Universidad de Buenos Aires
Facultad de Ciencias Exactas y Naturales
Departamento de Computación

Análisis y desarrollo de representaciones generales de audio

Tesis presentada para optar por el título de
Doctor de la Universidad de Buenos Aires
en el área Ciencias de la Computación

Leonardo Pepino

Directora de tesis: Dra. Luciana Ferrer
Director adjunto de tesis: Dr. Pablo Riera
Consejero de estudios: Dr. Pablo Negri
Lugar de trabajo: Laboratorio de Inteligencia Artificial Aplicada (LIAA), Departamento de Computación (DC) e Instituto UBA-CONICET de Ciencias de la Computación (ICC), Facultad de Ciencias Exactas y Naturales (FCEyN), Universidad de Buenos Aires (UBA).

Ciudad Autónoma de Buenos Aires, Argentina, 2025


Leonardo Pepino


Luciana Ferrer


Pablo Riera

Análisis y desarrollo de representaciones generales de audio

Resumen

El aprendizaje de representaciones ha sido un pilar fundamental en el avance del aprendizaje profundo, al permitir la reutilización de modelos para resolver tareas diversas de manera eficiente en términos de cómputo y datos. En particular, el aprendizaje no supervisado ha permitido aprovechar la abundancia de datos sin etiquetar para aprender representaciones ricas y útiles en escenarios con escasez de datos anotados, simplificando el desarrollo de modelos y favoreciendo la democratización de la inteligencia artificial.

Este trabajo doctoral se desarrolló entre los años 2020 y 2025, en un período marcado por el auge del aprendizaje auto-supervisado de representaciones de habla y, hacia el final, por la extensión de los modelos de lenguaje a otras modalidades distintas al texto, como el audio e imágenes. En ese contexto de éxito de los modelos de representaciones de habla, y la tendencia a crear sistemas cada vez más generalistas, este trabajo consistió en el desarrollo de un modelo de representaciones de audio que fuera útil para múltiples tareas del dominio del habla, y de otros dominios como la música y los sonidos ambientales. Este desarrollo culminó con EnCodecMAE, un modelo basado en *masked autoencoders* y pre-entrenado con señales de audio diversas provenientes de Audioset, LibriLight y Free Music Archive. Este modelo tiene la particularidad de utilizar EnCodec, un codec de audio neuronal, como señal objetivo. EnCodecMAE alcanzó resultados comparables o superiores al estado del arte en varias tareas de habla, música y audio general.

Además de presentar EnCodecMAE y mostrar los resultados de su evaluación, realizamos un estudio de ablación, explorando el efecto que tienen distintos aspectos de su diseño, como parámetros de enmascarado, representaciones de entrada, conjuntos de preentrenamiento, etc. Por último, realizamos un análisis del alineamiento entre distintas representaciones de audio, incluyendo EnCodecMAE, y representaciones cerebrales obtenidas a partir de resonancias magnéticas funcionales de la corteza auditiva. Mostramos que estos modelos, a pesar de no ser entrenados explícitamente para aproximar la actividad cerebral, exhiben un alineamiento creciente con la misma durante el preentrenamiento, y que el desempeño de las representaciones en distintas tareas se correlaciona con la similaridad con las representaciones cerebrales. Una implicancia práctica de este hallazgo es la posibilidad de utilizar medidas de alineamiento con la corteza auditiva como un indicador del desempeño general del modelo, con un costo computacional inferior al de benchmarks exhaustivos como HEAREval.

Palabras claves: aprendizaje de representaciones, auto-supervisión, procesamiento de audio, neuroconexionismo, transferencia de aprendizaje.

Analysis and design of general audio representations

Abstract

Representation learning has been fundamental for the advancement of deep learning, enabling the reuse of models to efficiently solve diverse tasks in terms of computation and data. In particular, unsupervised learning has made it possible to leverage the abundance of unlabeled data to learn rich and useful representations in scenarios where annotated data is scarce, simplifying model development and promoting the democratization of artificial intelligence.

This doctoral work was carried out between 2020 and 2025, a period marked by the rise of self-supervised learning of speech representations and, towards the end, by the extension of language models to other modalities beyond text, such as audio and images. Given the success of speech representation models and the trend towards general-purpose models, this work focused on developing an audio representation model that would be useful for multiple tasks in the speech domain, as well as in other domains such as music and environmental sounds. This effort culminated in EnCodecMAE, a model based on masked autoencoders and pre-trained with diverse audio signals from Audioset, LibriLight, and Free Music Archive. A distinctive feature of this model is the use of EnCodec, a neural audio codec, as the target signal. EnCodecMAE achieved results comparable to or surpassing the state of the art in various speech, music, and general audio tasks.

In addition to presenting EnCodecMAE and showcasing its evaluation results, we conducted an ablation study, exploring the effects of various design choices, such as masking parameters, input representations, and pretraining datasets. Finally, we analyzed the alignment between different audio representations—including EnCodecMAE—and brain representations obtained from functional magnetic resonance imaging (fMRI) of the auditory cortex. We show that these models, despite not being explicitly trained to approximate brain activity, exhibit increasing alignment with it during pretraining, and that the performance of the representations across tasks correlates with their similarity to brain representations. A practical implication of this finding is the possibility of using the alignment with the auditory cortex as a proxy of a model's general performance, at a lower computational cost than exhaustive benchmarks such as HEAREval.

Keywords: representation learning, self-supervision, audio processing, neuroconnectionism, transfer learning.

Agradecimientos

A continuación presentaré una lista no exhaustiva de personas que me acompañaron, apoyaron y ayudaron a crecer como persona y académicamente durante este doctorado.

Quiero agradecer a Luciana Ferrer, Pablo Riera y Agustín Gravano por confiar en mí y permitirme ser parte del maravilloso grupo de personas que conforman el Laboratorio de Inteligencia Artificial Aplicada (LIAA). A Luciana y Pablo por apoyarme en mis ideas locas con redes neuronales y audio y brindar su tiempo, conocimiento y compañía durante más de 5 años. Gracias a quienes compartieron oficina, charlas y juntadas en el LIAA: Diego, Juan, Enzo, Brusco, Jaz, Lara, Cyntia, Lauti, Eddy, Ramiro, Juanma, March, Facu, Petroni, Chechu, Martín, Bruno, Nico, Lao, Gus, Gastón, Dami, Octi, Fermín, Joaco, Agus, Gonzas, Fran, Pablos, Tomás, Erik y Manu.

Gracias a quienes me abrieron las puertas de sus laboratorios y me permitieron integrarme por un tiempo en sus equipos y seguir aprendiendo: Aren, Dan y el equipo de Sound Understanding en Google, al grupo de habla del BUT, al equipo de habla de ASAPP: Prashant, Suwon, Pablo, Marce, Cristina, Kwangyoun y Yi-Te, al equipo de habla en Google: Andrew, Bhuvana, Gary y Kyle, y a Martín Rocamora de la UPF.

Gracias a quienes hacen posible que exista una universidad pública, gratuita y de calidad, sin la cual no hubiera tenido la oportunidad de formarme. Agradezco a los docentes, no docentes, personal administrativo y de apoyo que, con su esfuerzo diario, sostienen una institución que abre puertas y transforma vidas.

También quiero agradecer a la comunidad científica en su sentido más amplio, y en particular a quienes desinteresadamente comparten de manera pública y gratuita su trabajo, ya sea en forma de artículos, conjuntos de datos o código abierto. Sin ese esfuerzo colaborativo, mi doctorado hubiera sido mucho más difícil y menos enriquecedor. Espero que el producto de esta tesis, a su vez, inspire, contribuya y sea de utilidad para la comunidad.

Por último, pero no por eso menos importante, gracias a mis padres Alberto y Alicia, a mis hermanos Martín y Ana, a mi compañera Belén, y a mis amigos Fabio, Flor, Caro, Hernan, Gastón, Germán y Ale por apoyarme en todos estos años a pesar de no entender mucho qué era lo que estaba haciendo. Este logro también es de ellos, porque sin su apoyo constante no hubiera llegado hasta aquí.

Índice general

1. Introducción 1
   1.1. Motivación 1
   1.2. Objetivo y contribuciones de la tesis 3
   1.3. Estructura de la tesis 4
   1.4. Trayectoria personal del doctorado 5
   1.5. Listado de publicaciones 6

2. Preliminares 7
   2.1. Atributos expertos 8
      2.1.1. Representaciones Tiempo-Frecuencia 8
      2.1.2. Descriptores acústicos 12
   2.2. Modelado de audio 14
      2.2.1. Aprender de los datos 16
      2.2.2. Redes neuronales artificiales 18
      2.2.3. Redes neuronales convolucionales 23
      2.2.4. Transformers 25
   2.3. Aprendizaje de representaciones 29
      2.3.1. Aprendizaje supervisado de representaciones 32
      2.3.2. Autoencoders de audio 33
      2.3.3. Aprendizaje por contraste 38
      2.3.4. Modelos de Lenguaje Enmascarado 42
      2.3.5. Aprendizaje por supervisión de modalidad cruzada 44
      2.3.6. Tabla resumen 47

3. EnCodecMAE 49
   3.1. Modelo propuesto 50
      3.1.1. EnCodec 50
      3.1.2. EnCodecMAE 51
   3.2. Evaluación de los modelos 53
   3.3. Detalles experimentales 56
   3.4. Comparación con el Estado del Arte 61
   3.5. Evaluación completa en HEAREval 67
   3.6. Exploración de Downstreams alternativos 70

4. Estudios de ablación y variantes 76
   4.1. Efecto de la representación de entrada 76
   4.2. ¿Cuál es la mejor capa? 79

4.3. ¿Cuándo detener el pre-entrenamiento? . . . 81
4.4. Efecto de la pre-post normalización . . . 82
4.5. Efecto de la señal objetivo . . . 84
4.6. Efecto del enmascarado . . . 88
4.7. Efecto del tamaño del decoder . . . 93
4.8. Variantes de EnCodecMAE . . . 94
4.8.1. MLM con mezclas . . . 94
4.8.2. Conexión residual larga . . . 95
4.8.3. Uso de registros . . . 96
4.8.4. Resultados . . . 96
4.9. Efecto de la semilla . . . 97
4.10. ¿Qué hiperparámetros afectaron más al desempeño? . . . 99

5. Alineamiento con representaciones cerebrales . . . 101
5.1. Introducción y Motivación . . . 101
5.1.1. Trabajo base . . . 103
5.1.2. Aportes . . . 104
5.2. Metodología . . . 106
5.2.1. Conjuntos de datos y preprocesamiento . . . 106
5.2.2. Análisis de regresión . . . 108
5.2.3. Análisis de regresión por componentes . . . 109
5.2.4. Análisis de similaridad de representaciones . . . 110
5.3. Resultados . . . 111
5.3.1. Análisis de regresión . . . 111
5.3.2. Análisis de regresión por componentes . . . 114
5.3.3. Análisis de similaridad de representaciones . . . 115
5.3.4. Efecto del tamaño de los modelos . . . 116
5.3.5. Efecto de la calidad de las representaciones . . . 118
5.3.6. Correspondencia estructural . . . 122
5.3.7. Alineamiento a lo largo del preentrenamiento . . . 125

6. Otros trabajos relacionados a la tesis . . . 128
6.1. Reconocimiento de emociones mediante fusión de texto y audio . . . 128
6.2. Reconocimiento de emociones con Wav2Vec 2.0 . . . 131
6.3. Embeddings posicionales para Audio Spectrogram Transformers . . . 135
6.4. Aprendizaje de prototipos musicales . . . 138

7. Conclusiones y trabajos futuros . . . 144
7.1. EnCodecMAE . . . 144
7.1.1. Señal objetivo y de entrada . . . 144
7.1.2. Eficiencia computacional . . . 145
7.1.3. Desempeño . . . 146
7.1.4. Análisis del diseño . . . 149
7.2. Alineamiento con representaciones cerebrales . . . 150
7.2.1. Resumen de los resultados . . . 150

7.2.2. Limitaciones del estudio 152
7.2.3. Trabajos futuros 154

Apéndice 157

A. Evaluación completa en HEAREval 158
A.1. Descripción de las tareas 158
A.2. Resultados 161

B. Error de reconstrucción de decoders 164

C. Corrección por atenuación 165

1

Introducción

People talking without speaking
People hearing without listening
People writing songs that voices never shared
No one dared disturb the sound of silence.

Simon & Garfunkel, *The Sound of Silence*

Resulta difícil imaginarnos un mundo sin sonido, sin música, sin habla, con lluvia y truenos mudos. Es que en pocos segundos, un sonido nos puede erizar la piel, cambiarle la cara al día o alertarnos de un peligro. Cada sonido lleva consigo un significado, y en consecuencia no solo se oye, también se comprende. Es aproximar ese deshilvanar del sonido hacia sus significados el objetivo quizás escondido detrás de este trabajo doctoral.

1.1. Motivación

Muchas de las tareas que realizamos cotidianamente dependen de nuestra capacidad de entender los sonidos que nos rodean. Sin embargo, adquirir ese entendimiento no es inmediato. Por ejemplo, aprender a transcribir piezas musicales puede llevar años de entrenamiento, y lo mismo ocurre a la hora de aprender a hablar un nuevo idioma. En contraste, un modelo de aprendizaje profundo puede aprender a resolver estas tareas en pocas horas o días de entrenamiento. Además, una vez entrenado, puede, en muchos casos, transcribir canciones o conversaciones en menos tiempo del que nos llevaría escucharlas, sin fatigarse.

Desarrollar modelos que entiendan los sonidos que los rodean tiene múltiples aplicaciones prácticas. En el campo de la accesibilidad, por ejemplo, se pueden generar subtítulos automáticos que no solo transcriban el habla, sino que también incorporen información relevante del entorno, como la presencia de una alarma sonando. En el dominio del habla, estas tecnologías pueden facilitar las comunicaciones entre personas, por ejemplo traduciendo automáticamente lo que alguien dice en un idioma que no entendemos. Del mismo modo, se podría mejorar la comunicación entre humanos y máquinas si estas últimas fueran capaces de entender nuestras emociones al hablar. También se pueden automatizar tareas de análisis que son tediosas o

propensas a la subjetividad, como el análisis de llamadas a centros de atención al cliente para identificar si los usuarios se sintieron satisfechos y si sus problemas fueron efectivamente resueltos.

En la actualidad, con los avances en el aprendizaje profundo, y con suficientes datos etiquetados y poder de cómputo, es posible lograr un buen desempeño en la mayoría de estas tareas. Una manera de hacerlo es optimizando los parámetros de un modelo matemático, en este caso una red neuronal artificial (ANN), para aproximar una función que transforme un conjunto de sonidos en las salidas deseadas para cada tarea específica. Existen algunos desafíos con este enfoque. Principalmente, es necesario contar con un conjunto de datos anotados. Dependiendo de la cantidad de parámetros que optimicemos, y de la dificultad de la tarea, serán necesarios más o menos datos para obtener sistemas que generalicen a nuevas instancias y posean un buen desempeño. Dado el elevado costo de realizar anotaciones, y la disponibilidad masiva de datos sin etiquetar, resulta deseable hacer uso de los mismos para mejorar el desempeño de los modelos.

Una de las principales técnicas para aprovechar datos sin etiquetas es la de transferencia de aprendizaje, la cual consiste en resolver una tarea T' utilizando un modelo que fue entrenado para resolver una tarea T. Particularmente, un tema activo de investigación en el campo del aprendizaje profundo es el de diseñar tareas de pretexto T que no requieran de anotaciones manuales y resulten en modelos reutilizables para distintas tareas T'_i. Internamente, las redes neuronales profundas aprenden a transformar las señales de entrada en alguna representación que resulta "útil" para resolver las distintas tareas. Por ejemplo, si la tarea fuera clasificar si hay actividad vocal o no en un audio, sería deseable que la última capa de la red neuronal reciba una representación en la que los datos de ambas clases sean linealmente separables. La forma de onda o un espectrograma probablemente no exhiban esta propiedad y requieran de una frontera de decisión mucho más compleja. Sin embargo, las capas de una red profunda pueden transformar ese espectrograma en distintas representaciones intermedias que resulten finalmente en una representación con esta propiedad. Luego, si queremos resolver otra tarea, es posible que esa representación también le sea útil, o al menos más útil que utilizar directamente un espectrograma. El principal objetivo de este trabajo doctoral es encontrar mecanismos que permitan aprender, mediante una tarea de pretexto T, representaciones que luego puedan utilizarse para resolver tareas diversas de audio, las cuales pueden ser del dominio del habla, la música o los sonidos ambientales. Para esto, debemos encontrar tareas de pretexto que den origen a representaciones generales del audio.

Por otro lado, una vez obtenidas representaciones de audio que resultan útiles para resolver múltiples tareas auditivas humanas, es interesante preguntarse si los modelos convergen hacia representaciones similares a las de nuestra corteza auditiva, responsable de interpretar y procesar los estímulos sonoros. Este tipo de análisis, en el que se emplean redes neuronales artificiales para modelar datos neuronales o de comportamiento humano, se enmarca dentro del campo emergente del neuroconexionismo [1]. Este campo tiene el potencial de ayudarnos a comprender mejor tanto el funcionamiento de las redes artificiales como el de las biológicas que

inspiran su diseño.

1.2. Objetivo y contribuciones de la tesis

El objetivo central de este trabajo doctoral es desarrollar un modelo de audio cuyas representaciones puedan ser utilizadas para resolver tareas diversas de audio, abarcando distintos dominios como el habla, música y sonidos ambientales.

Los pasos que dimos en esta dirección fueron:

- Proponer EnCodecMAE, un modelo general de audio basado en redes transformers [2] y la arquitectura de Masked Autoencoder [3], y entrenarlo con aproximadamente 12000 horas de audio diverso para reconstruir segmentos de audio enmascarados.
- Evaluar la calidad de las representaciones resultantes entrenando distintos modelos sencillos que toman como entrada estas representaciones. Específicamente evaluamos nuestros modelos en el benchmark HEAREval [4], y también en la tarea de reconocimiento del habla (ASR) del benchmark SUPERB [5].
- Explorar el uso de modelos alternativos a los utilizados en el benchmark HEAREval, particularmente regresiones logísticas, K vecinos más cercanos, y un modelo que combina representaciones provenientes de múltiples capas del modelo.
- Estudiar cómo impactan distintas decisiones de diseño en la calidad de las representaciones y proponer variantes de EnCodecMAE.

Un objetivo secundario, orientado a entender mejor qué están representando los distintos modelos, y a corroborar y expandir resultados de trabajos previos, es el de analizar si las representaciones de distintos modelos de audio, incluido EnCodecMAE, son similares a las representaciones cerebrales de sujetos escuchando estímulos sonoros. En esta dirección cumplimos los siguientes objetivos:

- Realizar un análisis de similaridad de representaciones entre las activaciones de modelos de audio y mediciones obtenidas a partir de resonancias magnéticas funcionales (fMRI) de sujetos escuchando estímulos sonoros.
- Analizar si el desempeño en distintas tareas de audio se correlaciona con la similaridad entre las representaciones de audio y las cerebrales, es decir, si cuanto más se parece un modelo a un cerebro, mejor funciona.
- Analizar si existe una correspondencia estructural entre los modelos de audio y la corteza auditiva, es decir, si las salidas de las primeras capas de una red neuronal se asemejan más a etapas tempranas del procesamiento auditivo, mientras que las últimas se asemejan más a etapas tardías.
- Analizar si la similaridad con las representaciones cerebrales aumenta durante el preentrenamiento, a pesar de no efectuarse una optimización directa de la misma.

1.3. Estructura de la tesis

Este trabajo está organizado en 7 capítulos, incluyendo esta introducción:

Capítulo 2. Preliminares. Se presenta el marco teórico necesario para comprender este trabajo, organizado en tres secciones. En la primera sección se explican conceptos fundamentales del procesamiento de audio, en la segunda se introducen temáticas relacionadas al aprendizaje automático y las redes neuronales profundas, abarcando las arquitecturas de redes neuronales convolucionales y transformers, y por último, en la tercera sección se realiza una revisión de la literatura relacionada al aprendizaje de representaciones de audio.

Capítulo 3. EnCodecMAE. Se presenta el modelo de representaciones de audio desarrollado durante este trabajo doctoral, explicando las motivaciones detrás de su diseño, la metodología de entrenamiento y evaluación, y presentando resultados del desempeño en distintas tareas de audio.

Capítulo 4. Estudios de ablación y variantes. En esta sección se busca entender cómo impactan los distintos factores de diseño en la calidad de las representaciones de EnCodecMAE. Para esto, se realiza un estudio de ablación modificando distintos hiperparámetros del modelo y viendo su influencia en el desempeño de la representación al ser utilizada para resolver distintas tareas de audio. A su vez, se proponen distintas variantes de EnCodecMAE y se analiza el desempeño de las mismas.

Capítulo 5. Alineamiento con representaciones cerebrales. En esta sección se intenta responder a la pregunta: "A medida que las representaciones de audio se vuelven más útiles para resolver tareas que los humanos resolvemos, ¿también se vuelven más similares a nuestras representaciones cerebrales del sonido?". Para responder a esta pregunta, hacemos uso de distintas técnicas del campo del neuroconexionismo, particularmente el análisis de similaridad de representaciones (RSA), y el uso de regresores lineales regularizados, para comparar las activaciones de los distintos modelos de audio y mediciones de fMRI de la corteza auditiva de sujetos escuchando estímulos sonoros.

Capítulo 6. En este capítulo se recopilan y resumen algunos aportes realizados durante el trabajo doctoral, que si bien están relacionados con el tema principal de la tesis, creemos que harían a este documento muy extenso y romperían con el flujo de lectura. Particularmente, en las dos primeras secciones se comentarán resultados obtenidos en la tarea de reconocimiento de emociones a partir del habla, utilizando representaciones de modelos pre-entrenados de texto y habla. Luego, se presentan algunos resultados sobre el efecto de utilizar distintas representaciones posicionales en transformers de audio para detección de eventos acústicos. Finalmente, se comentan resultados de un proyecto colaborativo con investigadores de la Universidad de Pompeu Fabra (España) y la Universidad de la República (Uruguay) utilizando representaciones de EnCodecMAE para mejorar la interpretabilidad de clasificadores de audio utilizando redes prototípicas.

Capítulo 7. Conclusiones y trabajo futuro. Se presentan las conclusiones del trabajo realizado y se proponen líneas de trabajo futuro.

1.4. Trayectoria personal del doctorado

Antes de cerrar la introducción, creemos que puede resultar útil comprender el contexto cronológico detrás de este trabajo doctoral, el cual no se condice con el orden de los capítulos.

Hacia el año 2019 me recibí de ingeniero de sonido en la Universidad Nacional de Tres de Febrero, y en simultáneo comencé a trabajar en un proyecto de reconocimiento de emociones a partir del habla, bajo la dirección de mi directora Luciana Ferrer y del Dr. Agustín Gravano. Es por esto que mis primeras publicaciones durante el doctorado [6, 7] se asocian a este proyecto y van incorporando gradualmente la idea de utilizar modelos preentrenados. Estos primeros trabajos tuvieron una buena recepción en la comunidad de procesamiento de habla, al haber aplicado técnicas de transferencia de aprendizaje en el campo del reconocimiento de emociones, alcanzando o superando al estado del arte.

Dado el éxito que tuvimos al reutilizar modelos de habla en la tarea de reconocimiento de emociones, fue natural preguntarnos: "¿Podemos construir mejores representaciones que sean aplicables a otros dominios del audio más allá del procesamiento del habla? ¿Podemos aprovechar los avances en el campo del procesamiento del lenguaje natural y el aprendizaje auto-supervisado?". Responder estas preguntas resultó mucho más complejo de lo esperado y conllevó cumplir otros objetivos de carácter más logístico:

- Entrenar este tipo de modelos que utiliza arquitecturas transformer y una gran cantidad de datos es computacionalmente muy costoso. Al mismo tiempo, no contábamos con la infraestructura necesaria en el laboratorio. Un primer paso fue conseguir financiamiento para adquirir equipos con los que poder realizar este trabajo. Afortunadamente, conseguimos dos premios: Google Research Award 2019 y Amazon Research Award 2019, que nos ayudaron a adquirir dos servidores, y también nos alentaron a continuar en esta dirección de investigación.
- Al ser una nueva línea de investigación en el laboratorio, se debió comenzar a construir una base de código con la cual poder manejar un gran caudal de datos y experimentar de una forma organizada. Esto supuso una inversión de tiempo importante, que retrospectivamente también resultó en un gran aprendizaje y posibilitó experimentar con muchos modelos de forma eficiente.

Plantearnos estas preguntas y resolver los problemas logísticos, resultó en el desarrollo de nuestro propio modelo de audio, EnCodecMAE. Este trabajo fue publicado primero en arXiV en el 2023 y luego presentado en Interspeech 2025. Poco después de publicar este trabajo en arXiV, investigadores del campo de la neurociencia de la universidad de MIT publicaron un trabajo mostrando cómo los modelos de aprendizaje profundo de habla y audio aprenden representaciones similares a las observadas en el cerebro mediante fMRIs [8], y liberaron el código. Una semana después de su publicación, utilicé este código para calcular cuan similares eran las representaciones de EnCodecMAE a las cerebrales, y para mi sorpresa mostraban una similaridad mayor a los modelos que habían sido analizados en el trabajo original. En ese momento me encontraba enfocado en mejorar EnCodecMAE y realizando

pasantías, por lo que el proyecto quedó en un segundo plano. Hacia mediados del 2024 cursé la materia de Procesamiento de Imagenes Cerebrales y Machine Learning dictada por Federico Raimondo, lo cual me permitió entender mejor los detalles y metodología del trabajo y retomar y ampliar los resultados que había obtenido. Actualmente nos encontramos terminando la redacción de un artículo científico con estos resultados, los cuales se describen en el capítulo 5. También planeamos publicar un artículo de revisión ampliando la sección 2.3, en el que se hará una revisión de trabajos en el campo del aprendizaje de representaciones de audio, y una comparación entre las representaciones de los distintos modelos de audio.

En paralelo, adquirí experiencia en la industria mediante pasantías: una en Hipcam (2021), dos en Google (2022 y 2024) y una en ASAPP (2024); y en la academia realizando una pasantía en el prestigioso laboratorio de procesamiento del habla de la Brno University of Technology (BUT) en 2023. Todos estos trabajos tuvieron alguna relación con el aprendizaje de representaciones de audio y aplicaciones de las mismas, pero no son mostrados en este documento ya que algunos resultados aún no son públicos, y otros romperían el flujo natural de esta tesis, el cual busco conservar.

1.5. Listado de publicaciones

A continuación se listan las publicaciones más relevantes realizadas durante el doctorado.

- L. Pepino, P. Riera and L. Ferrer, *"Encodecmae: Leveraging neural codecs for universal audio representation learning"*. En Proc. Interspeech 2025, pp. 3519-3523.
- P. Alonso-Jimenez, L. Pepino, R. Batlle-Roca, P. Zinemanas, D. Bogdanov, X. Serra, M. Rocamora, *"Leveraging pre-trained autoencoders for interpretable prototype learning of music audio"*. En *ICASSP Workshop on Explainable AI for Speech and Audio (XAI-SA)*, 2024.
- L. Pepino, P. Riera and L. Ferrer, *"Study of Positional Encoding Approaches for Audio Spectrogram Transformers"*. En ICASSP 2022, pp. 3713-3717.
- L. Pepino, P. Riera y L. Ferrer, *"Emotion recognition from speech using wav2vec 2.0 embeddings"*. En *Proceedings de Interspeech 2021*, pp. 3400–3404.
- L. Pepino, P. Riera, L. Ferrer y A. Gravano, *"Fusion approaches for emotion recognition from speech using acoustic and text-based features"*. En *ICASSP 2020*, pp. 6484-6488.

2

Preliminares

> "Con cuatro parámetros puedo ajustar un elefante, con cinco puedo hacer que mueva su trompa."
>
> *John Von Neumann*

Según Beranek [9], el sonido es una perturbación propagada por un medio elástico, como el aire o el agua, que causa un cambio en la presión o el desplazamiento de las partículas del mismo, y puede ser detectado por una persona o un instrumento. Por ejemplo, un micrófono es un instrumento que puede transformar estos cambios de presión en señales eléctricas, las cuales pueden ser posteriormente digitalizadas y almacenadas en una computadora.

Dado que la presión sonora, o las señales eléctricas de un micrófono, son magnitudes continuas, para su almacenamiento en una computadora es necesario realizar un proceso de digitalización en el que se muestree la señal tanto en el tiempo como en amplitud. Este proceso es llevado a cabo normalmente por un conversor de señal analógica a digital, el cual posee una **frecuencia de muestreo** que determina cuántas muestras de la señal se tomarán en un segundo, y una **profundidad de bits** que determina cuántas amplitudes posibles se almacenarán.

Según el **teorema de Nyquist**, dada una señal continua con una frecuencia máxima f_{max}, se necesita una frecuencia de muestreo de al menos $2f_{max}$ para no perder información de la señal continua al discretizar. Dado que el rango de audición del ser humano se encuentra entre los 20 y 20000 Hz, es común que las señales de audio se capturen con una frecuencia de muestreo de 44100 Hz, que corresponde a un poco más del doble de la máxima frecuencia audible[1]. Si tuvieramos el oído de un delfín, el cual puede percibir hasta 150 kHz, tendríamos que utilizar frecuencias de muestreo mucho mayores. Por otro lado, si queremos capturar sonidos como habla, que ocupan un menor rango de frecuencia, podemos utilizar una frecuencia de muestreo menor y reducir la memoria utilizada. En el caso de habla, se suelen usar

[1] La mayoría de las señales continuas no están acotadas en ancho de banda (es decir, no tienen una frecuencia máxima), por ende, hay que utilizar un filtro con frecuencia de corte $f_{cut} = f_{max}$ que remueva las frecuencias mayores a f_{max}. Este proceso de filtrado no es perfecto por lo que no atenúa completamente las frecuencias mayores a f_{max}, generando ruido de aliasing si se muestrease con $2f_{max}$. Para minimizar este efecto, se elige una frecuencia de muestreo ligeramente mayor al doble de la frecuencia de corte del filtro antialiasing.

frecuencias de muestreo de 16 kHz u 8 kHz en el caso de habla telefónica. Respecto a la profundidad de bits, se suelen utilizar 16 bits, lo cual implica registrar $2^{16} = 65536$ valores posibles de amplitud. Utilizar una mayor cantidad de bits permite reducir el error (o ruido) de cuantización, aumentando la relación señal a ruido. Para la mayoría de las situaciones de escucha, el ruido de cuantización es inaudible utilizando 16 o más bits.

2.1. Atributos expertos

A la hora de analizar señales de audio, o intentar resolver problemas en donde el sonido está involucrado, hasta hace unos años se solía evitar el uso de la forma de onda como representación de audio. El principal motivo por el que resultaba desafiante modelar directamente la forma de onda es su alta resolución temporal. Un solo segundo de habla puede contener 16000 muestras, lo cual complica la extracción de información de la misma, o su procesamiento con métodos de aprendizaje automático que modelan secuencias, como Cadenas ocultas de Markov (HMM) [10], redes neuronales recurrentes (RNN) [11] o de memoria a corto y largo plazo (LSTM) [12]. Una alternativa a modelar directamente la forma de onda, es extraer información de la misma que sea útil para resolver el problema de interés. Expertos en la tarea a resolver solían diseñar manualmente técnicas de extracción de atributos o características de la señal, reemplazando la forma de onda por un conjunto de estos atributos. Si bien en ocasiones se extraen atributos directamente de la forma de onda, en general resulta más conveniente trabajar en el dominio de las frecuencias.

2.1.1. Representaciones Tiempo-Frecuencia

Una de las transformaciones de la forma de onda más utilizadas es la **Transformada Discreta de Fourier (DFT)** que permite pasar de una secuencia de N muestras en el dominio del tiempo $x \in \mathbb{C}^N$, a una secuencia $X \in \mathbb{C}^K$ en el dominio de las frecuencias:

$$X[k] = \sum_{n=0}^{N-1} x[n] e^{-i\frac{2\pi}{N}kn}, \quad n = 0, 1, \ldots, N-1 \tag{2.1}$$

Particularmente, la DFT devuelve tantos valores distintos de frecuencia como valores distintos de tiempo haya en la señal a transformar, es decir $N = K$. Normalmente, en el dominio del audio, se trabajará con señales reales, y dado que la DFT de una señal real posee simetría hermítica, es decir, $X[k] = \overline{X[-k]}$, conservar $K/2 + 1$ frecuencias es suficiente para recuperar el espectro completo.

Los sonidos suelen cambiar a lo largo del tiempo, por ejemplo, al hablar estamos cambiando los fonemas, entonación, energía, etc. Si suponemos que en un lapso muy corto de tiempo, normalmente del orden de las decenas de milisegundos, la distribución de las muestras de habla no cambia, es decir se mantiene estacionaria, podemos calcular la DFT sobre segmentos cortos a lo largo de la señal. La secuencia

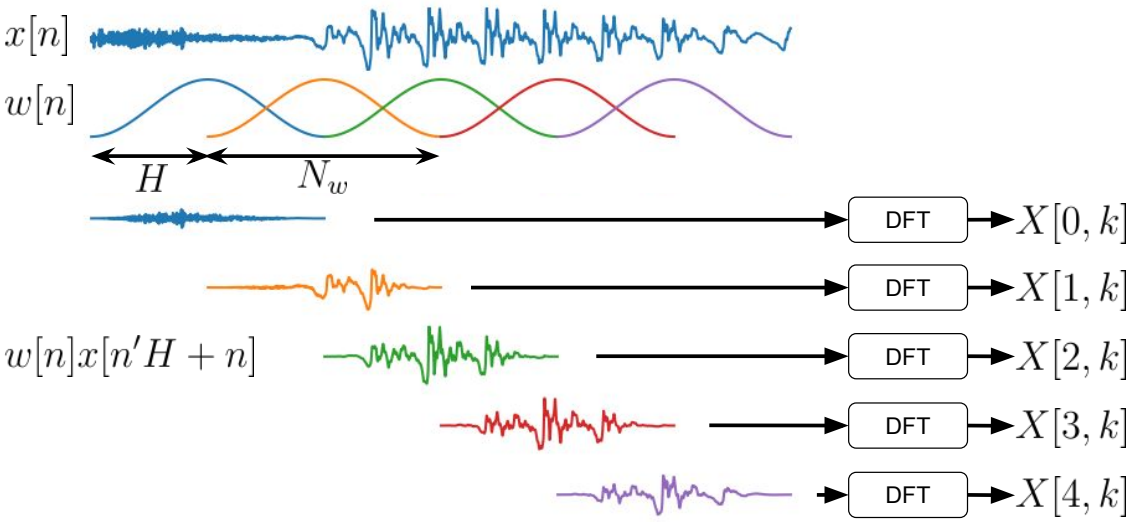

Figura 2.1: Proceso de cálculo de la STFT mediante enventanado de la señal $x[n]$ con ventanas w de tamaño N_w y salto H, y posterior cálculo de la DFT sobre las señales enventanadas.

de DFTs resultante representa cómo varía el espectro del habla, o cualquier sonido, en el tiempo. Este método se denomina **Transformada de Tiempo Corto de Fourier**, o STFT por sus siglas en inglés, y forma parte del concepto más general de análisis de tiempo corto. Matemáticamente, la STFT se puede expresar como:

$$X[n',k] = \sum_{n=0}^{N_w-1} w[n]x[n'H+n]e^{-i\frac{2\pi}{N_w}kn}, \quad n' = 0,1,\ldots,\lceil N/H \rceil, \quad k = 0,1,\ldots,N_w-1 \tag{2.2}$$

donde w es una ventana de largo N_w, que podría ser rectangular (un valor constante de 1) o con otra forma como Hanning o Blackman, H es el tamaño de salto e indica cuántas muestras se desplaza la ventana para calcular cada cuadro n' de la STFT. Cuando $H < N_w$ se produce un solapamiento de las ventanas. La STFT resultante poseerá N_w frecuencias distintas (o $N_w/2+1$ si x es real), y $\lceil \frac{N}{H} \rceil$ muestras temporales o cuadros. En la práctica, se suele trabajar con la DFT y STFT utilizando coordenadas polares, expresando los valores complejos en función de su fase y magnitud. Muchas veces la fase se descarta y se trabaja solamente sobre la magnitud de la STFT, la cual puede visualizarse con un espectrograma. En la Figura 2.1 se puede observar un diagrama explicando el proceso de cálculo de la STFT, y en la Figura 2.2a) y b), se puede observar un ejemplo de la magnitud y fase de la STFT de una señal de habla.

Una propiedad importante de la DFT es que se puede recuperar la señal temporal original aplicando la transformada discreta de Fourier inversa (IDFT); es decir, la DFT es una transformación inversible. La STFT también es inversible, siempre y cuando se utilicen ciertas combinaciones de ventana, H y N_w que cumplan con el criterio de solapamiento y suma constante (COLA) [13].

En la literatura existen muchas representaciones tiempo-frecuencia alternativas a la STFT. Por ejemplo, la **transformada de Q constante (CQT)** [14] es amplia-

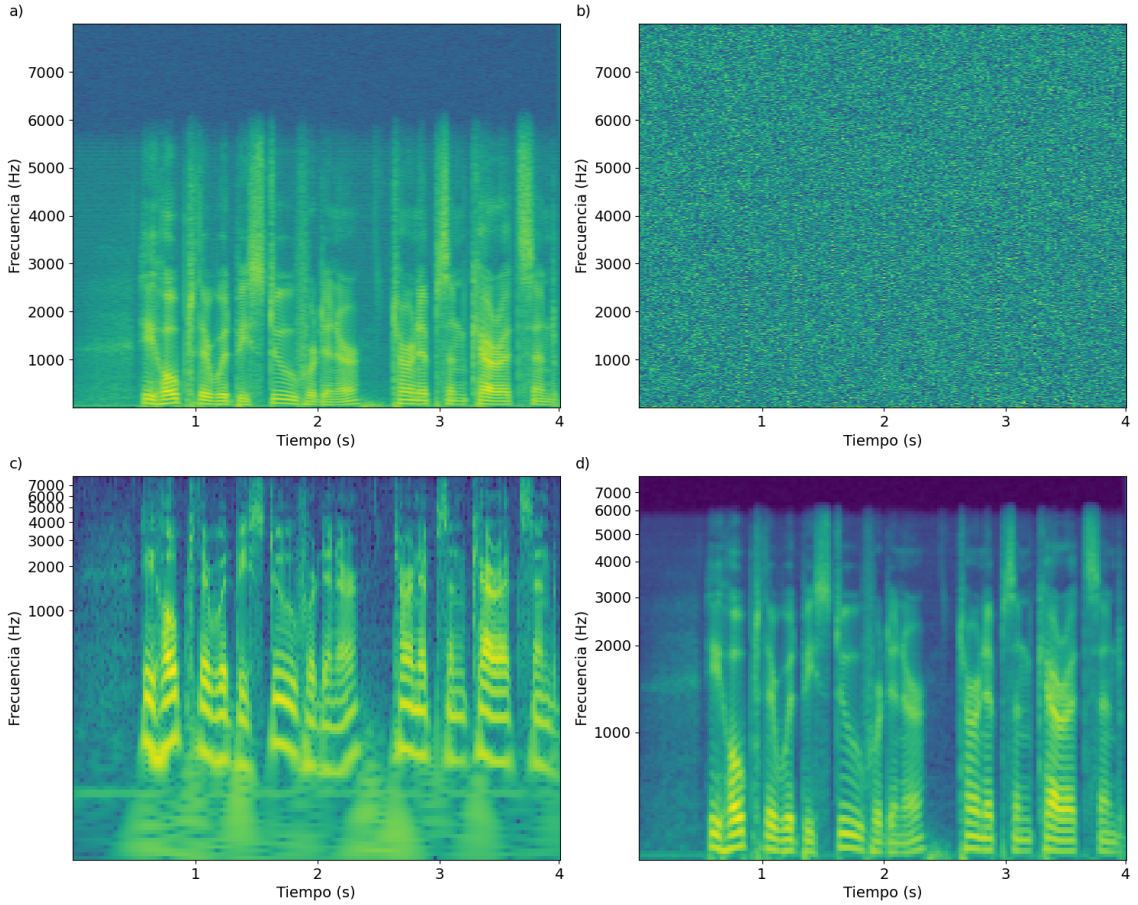

Figura 2.2: Representaciones tiempo-frecuencia para una señal de habla. a) Magnitud de la STFT en escala logarítmica; b) Fase de la STFT; c) Transformada de Q constante con 12 bins por octava; d) Melspectrograma con 128 filtros

mente utilizada para el análisis de señales musicales, debido a que las frecuencias no se distribuyen de manera equiespaciada como en la STFT, sino que de manera geométrica, lo cual está mejor alineado con conceptos musicales (como octavas) y de percepción sonora. Esta representación se puede observar en la Figura 2.2c), en donde se puede ver el espaciado de frecuencias no uniforme. Esta caracteristica permite una mayor resolución en bajas frecuencias que en altas. Otra representación tiempo-frecuencia muy utilizada, especialmente en el dominio del habla, es el **melspectrograma**, que utiliza la escala mel en el eje de frecuencias. Esta representación puede observarse en la Figura 2.2d), y al igual que en la CQT, las frecuencias no se encuentran equiespaciadas. La escala mel es una escala perceptual de pitch, construida para que las frecuencias se perciban equidistantes. Existen varias formulas que relacionan frecuencia f con mels, siendo la de Slaney [15] una de las más utilizadas:

$$mel(f) = \begin{cases} 3\frac{f}{200} & f < 1000 \\ 15 + 27\log_{6.4}(\frac{f}{100}) & f \geq 1000 \end{cases}$$

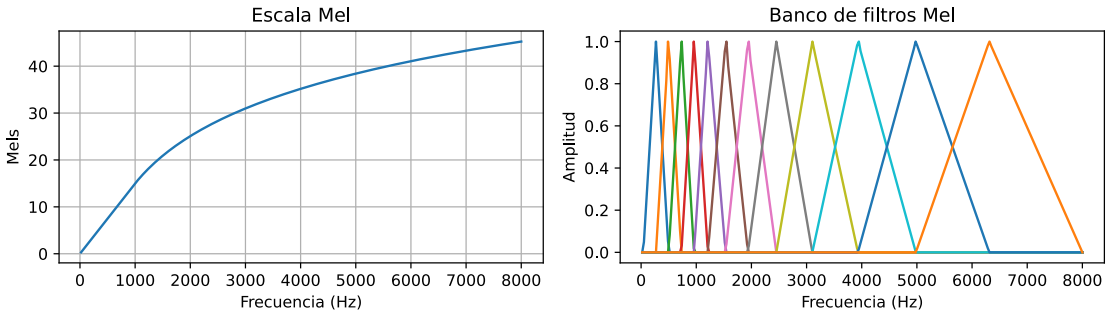

Figura 2.3: Izquierda: mapeo de frecuencia a mels usando la fórmula de Slaney. Derecha: banco de filtros triangulares en escala mel.

En la práctica, para representar un espectrograma en escala mel, se construye un banco de filtros triangulares en donde cada filtro se espacia de acuerdo a la escala mel, y luego se multiplica con el espectrograma e integra la energía. En la Figura 2.3, se puede observar estos filtros y la equivalencia entre frecuencia y mels utilizando la fórmula de Slaney.

Si bien muchas de estas transformaciones pueden facilitar el análisis de ciertos tipos de señales, la inversibilidad de las mismas no siempre puede garantizarse como en el caso de la STFT. Si bien se puede aproximar la señal original a partir de un melspectrograma, por ejemplo mediante la aplicación de una pseudo-inversa de la matriz de filtros mel y el uso de Griffin-Lim [16], o usando Vocoders basados en redes neuronales [17], la reconstrucción no es perfecta. Lo mismo puede decirse para la CQT, con la excepción de ciertas versiones modificadas donde es posible una reconstrucción perfecta [18].

2.1.2. Descriptores acústicos

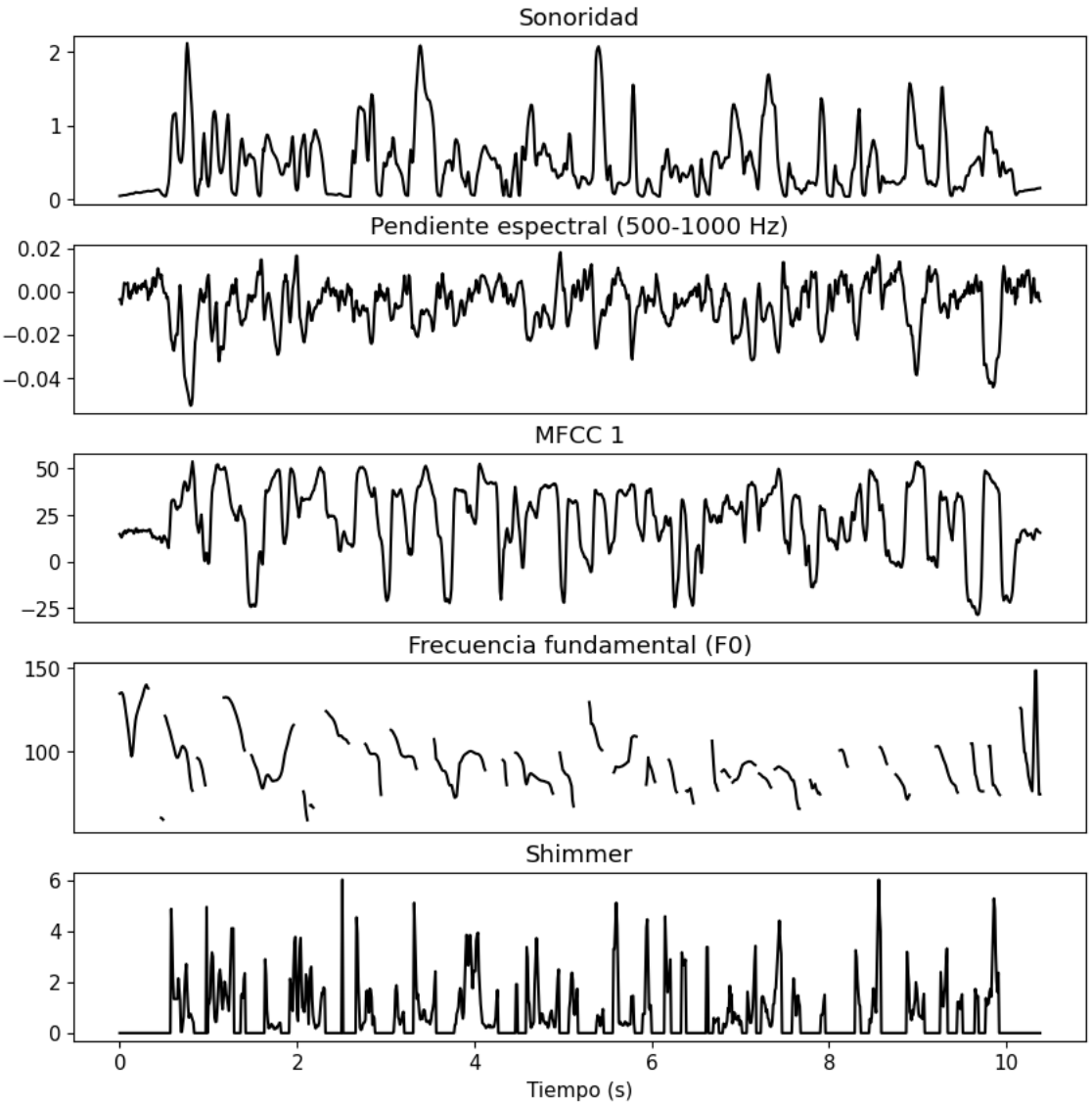

Figura 2.4: Ejemplos de descriptores acústicos de bajo nivel pertenecientes a eGeMAPS y calculados con OpenSmile sobre la señal de habla de la Figura 2.2.

Aunque utilizar representaciones tiempo-frecuencia simplifica el análisis de audio, muchas veces se quiere extraer información más específica que pueda ser útil para resolver un problema determinado. Por ejemplo, se sabe que al hablar, las emociones afectan la respiración, fonación y articulación del habla [19]. Esto se traduce en cambios en la señal de audio que podrían ser extraídos a partir de la forma de onda o espectrogramas haciendo uso de conocimiento experto. Con este objetivo, para muchas tareas se pueden diseñar descriptores acústicos que capturan características del habla relevantes al problema a resolver y usualmente interpretables. En el caso de reconocimiento de emociones, un conjunto de descriptores acústicos comúnmente

utilizado es el conjunto minimalístico de parámetros acústicos de Genova (o eGeMAPS por sus siglas en inglés) [20], el cual consiste de 18 descriptores acústicos de bajo nivel (o LLD por Low Level Descriptors). Estos LLD son secuencias de escalares calculados a partir del audio en el dominio del tiempo o frecuencias, y pueden ser obtenidos utilizando el paquete OpenSmile [21]. Algunos ejemplos relevantes son:

- **Energía normalizada**: para una señal con media cero se corresponde con su varianza, $E(x) = \frac{1}{T}\sum_{t=0}^{T-1} |x[t]|^2$.
- **Sonoridad**: o loudness en ingles, es la percepción subjetiva de la presión sonora. Se corresponde con lo que informalmente denominamos 'volumen'. Existen muchas maneras de aproximar la sonoridad, siendo común integrar la energía en un espectrograma corregido según criterios perceptuales, como las curvas de igual sonoridad [22], para tener en cuenta la mayor o menor sensibilidad del oído a ciertas regiones del espectro, y aplicar compresión aproximando algunas características de la audición humana [23].
- **Frecuencia fundamental (F0)**: es la frecuencia más baja de una onda periódica. En habla, es la frecuencia a la que vibran las cuerdas vocales, mientras que en música define qué nota está siendo tocada. Existen muchas técnicas para estimar F0, algunas analizan directamente la serie temporal, como YIN [24] y PYIN [25], otras el espectro y cepstrum como Harvest [26] y SWIPE [27], y métodos más recientes utilizan redes neuronales como CREPE [28] y SPICE [29].
- **Shimmer**: es la diferencia de amplitud pico entre dos períodos consecutivos de la señal.
- **Pendiente espectral**: es la pendiente obtenida mediante una regresión lineal sobre la potencia espectral en algún rango de frecuencias.
- **Coeficientes cepstrales en frecuencia mel (MFCC)** [30]: se obtienen al aplicar la Transformada Coseno Discreta (DCT) sobre el logaritmo de un melspectrograma. En general, calcular el espectro de un espectro en escala logarítmica da lugar a lo que se conoce como **cepstrum**, y permite una mejor visualización de formantes y frecuencia fundamental. Particularmente los primeros 12 MFCC, junto con la energía y sus deltas, han sido ampliamente utilizados en el campo del reconocimiento del habla [30]. Utilizar más coeficientes también ha sido útil para otros problemas dentro del campo de procesamiento de habla, como identificación de hablante [31], y fuera de este, como en clasificación de género musical [32] y de escenas acústicas [33, 34].

En la Figura 2.4 se pueden observar estos atributos calculados sobre la misma señal de habla de la Figura 2.2. En la tesis de Florian Eyben [35] se pueden encontrar más LLD normalmente utilizados en análisis de habla, junto con detalles de su implementación.

Como se mencionó previamente, los LLD son secuencias de escalares y suelen ser útiles para posteriormente aplicar modelos secuenciales. Sin embargo, algunos modelos sencillos como máquinas de vectores de soporte (SVM) [36], o random

forests [37], no son capaces a priori de modelar secuencias. Para adaptar los LLD a este tipo de modelos, un enfoque utilizado comunmente es resumir las secuencias en valores globales. Por ejemplo, eGeMAPS resume los LLD calculando su media, coeficiente de variación y percentiles, entre otras funciones, transformando los 18 LLD originales en 62 valores escalares globales. A modo ilustrativo, si las LLD son secuencias calculadas sobre ventanas de 50 ms sin solapamiento, y se tiene un audio de 1 segundo, eso implica caracterizar a un audio utilizando 18 secuencias de largo 20 cada una (360 valores en total). Si el audio durara 10 segundos, se tendrían 3600 valores. Sin embargo, si en lugar de utilizar los LLD nos quedamos solo con los 62 valores escalares, sin importar el largo del audio, podremos caracterizarlo con un vector de 62 dimensiones. Esto es muy útil ya que permite aplicar una gran variedad de modelos de aprendizaje automático para modelar audio, pero al mismo tiempo descarta mucha información temporal potencialmente relevante del audio al comprimirlo (o resumirlo) en solo 62 valores.

2.2. Modelado de audio

"Numquam ponenda est pluralitas sine necessitate"

William de Ockham

Antes de explicar cómo podemos modelar sonidos, veamos para qué puede servirnos. Supongamos que estamos a cargo de un parque nacional extenso y lleno de vegetación, y queremos asegurarnos de que no se talen árboles. Se podrían utilizar cámaras y detectar la actividad de tala a partir de la imagen, pero esto sería muy costoso ya que se necesitarían muchas cámaras para cubrir un espacio tan extenso. Una alternativa menos costosa es colocar micrófonos en distintos puntos del parque, y monitorear la presencia de ruidos de tala. Dada la intensidad de este tipo de ruidos, los micrófonos se pueden espaciar mucho más que las cámaras, reduciendo el costo significativamente. Ahora bien, tenemos 100 micrófonos a lo largo y ancho de nuestro parque nacional; pongamos a 100 personas a escuchar las 24 horas del día, los 7 días de la semana cada uno de los micrófonos. Como podrá imaginarse el lector, esta solución es poco práctica. Sin embargo, podemos construir un modelo de esa persona que escucharía las grabaciones. Propongamos una función matemática:

$$\hat{f}^* : \mathbb{R}^T \to \{0, 1\}$$

Esta función toma como entrada una señal de audio $x \in \mathbb{R}^T$ capturada por el micrófono con una cierta cantidad T de muestras, y devuelve 0 si no detecta ruidos de tala, y 1 si los detecta.

Si logramos encontrar esta función $\hat{f}^*$ que sea buena detectando el ruido de tala, y puede calcularse rápidamente en una computadora, o incluso en un dispositivo embebido en los micrófonos, podemos implementar un sistema que alerte a los guardaparques en tiempo real y evitar la tala de árboles.

La función $\hat{f^*}$ fue encontrada, el sistema es un éxito. Ahora llegó un grupo de biólogos al parque nacional. Nuestros micrófonos grabaron durante un año todos los sonidos del parque, y los biólogos quieren analizar los mismos para conocer qué especies están presente, y estudiar sus vocalizaciones. Los biólogos se encuentran con la tarea ardua de escuchar 100 años de grabaciones, buscando el momento exacto en el que algún pajarito se pone a cantar. Al mismo tiempo, los biólogos no tienen en claro qué es lo que buscan, quieren entender, explorar el parque y sus sonidos. Por suerte son amigos de un científico de datos que les da una función $\hat{g^*}$, esta vez con una forma distinta:

$$\hat{g^*} : \mathbb{R}^T \to \mathbb{R}^D$$

Al igual que la función anterior, esta toma un segmento de audio como entrada, pero ahora devuelve un vector con D dimensiones. Esta función es muy útil para los biólogos, ya que cuando dos sonidos se parecen, devuelve vectores parecidos, pero cuando no lo hacen, los vectores son muy distintos. Ahora los biólogos pueden buscar algún segmento de audio que no tenga ningún sonido interesante (el sonido habitual del parque), pasarlo por $\hat{g^*}$, y obtener un vector de referencia v_r. Luego repiten el proceso con los 100 años de grabaciones, y con suerte, si la computadora es potente y la función $\hat{g^*}$ no es muy complicada, en poco tiempo pueden obtener muchos vectores v_i correspondientes a los 100 años de grabación. Luego, los biólogos pueden encontrar los segmentos 'interesantes' de audio, que son los que están más lejos de v_r, es decir que se diferencian mucho del sonido habitual del parque. No solo eso, sino que pueden llamar a su amigo científico de datos y pedirle que haga reducción de dimensionalidad y/o clustering, y ahora los biólogos cuentan con un mapa interactivo de sonidos en el que se puede ver cada sonido como un punto bidimensional y escucharlo al hacer click[2]. Este tipo de procedimiento consistente en encontrar una representación de sonidos y luego utilizar técnicas de clustering o reducción de dimensionalidad, es muy común para el análisis de vocalizaciones animales [38, 39].

Esta sección de la tesis se enfocará en cómo encontrar estas funciones a partir de datos. Si bien se presentó el problema con dos aplicaciones específicas de ecología, el modelado de audio tiene muchisimas más aplicaciones, y muchas de ellas se pueden resolver estimando funciones. En problemas de **clasificación multiclase** queremos determinar a cuál de K categorías pertenece un audio, y para eso podemos buscar una función $\hat{f*} : \mathbb{R}^T \to \{0,1,...,K-1\}$. Algunos ejemplos de problemas de clasificación de audio son el reconocimiento de emociones [40] y enfermedades a partir del habla [41], detección de lenguaje [42], clasificación de escenas acústicas [43] y clasificación de género musical [44]. Otros problemas son de **clasificación multietiqueta** en los cuales un audio puede pertenecer a más de una categoría al mismo tiempo, teniendo $\hat{f*} : \mathbb{R}^T \to \{0,1\}^K$. Algunos ejemplos son el de detección de eventos acústicos [45] y el de tagging de canciones [46], ya que en un mismo audio puede haber un ladrido de perros y una persona hablando y una canción puede tener múltiples tags como

[2] Un ejemplo de mapa interactivo de sonidos que me gusta mucho se puede acceder en `https://experiments.withgoogle.com/ai/bird-sounds/view/` y permite visualizar y escuchar distintos cantos de aves.

jazz, relaxing, 70s. Si en lugar de determinar a qué categoría pertenece una señal, buscamos determinar un valor real, es decir $\hat{f}* : \mathbb{R}^T \to \mathbb{R}$, decimos que estamos ante un problema de **regresión**. Algunos ejemplos son estimación de tempo [47], estimación de frecuencia fundamental [29] y predicción de excitación y valencia [48].

Muchas veces, nos interesa que el codominio de $\hat{f}*$ sea otra secuencia temporal. En este caso se dice que estamos ante un problema de **transducción de secuencias**. En algunos casos la secuencia que queremos predecir puede ser binaria, por ejemplo, en el caso de detección de actividad vocal (VAD) [49]. En otros puede ser multiclase o multietiqueta, como en transcripción automática de música [50], o real al igual que la entrada, como cuando queremos remover el ruido de un audio, en cuyo caso la secuencia de salida será el mismo audio de entrada sin el ruido. En esta categoría entran muchos otros problemas interesantes, como el de separación de fuentes [51], dereverberación [52] y conversión de hablante [53]. Algo que podemos notar en las aplicaciones mencionadas es que el largo de la secuencia de salida es proporcional al de la secuencia de entrada. Por ejemplo, si quitamos el ruido de un audio, la secuencia de salida tiene el mismo largo que la de entrada, o si hacemos VAD tendrá el largo de la señal de entrada dividido por el largo de la ventana de análisis. Sin embargo, existen aplicaciones donde la relación entre largo de salida y entrada no es fija. Un ejemplo muy importante es el de reconocimiento del habla (ASR) [54]; se puede decir una misma frase en 3 segundos o en 5, resultando en dos audios con longitudes distintas pero la misma transcripción. Otros ejemplos en esta categoría son los de audio captioning [55], en el que dado un audio se predice una descripción textual del mismo, y el de traducción de habla a habla [56], en la que se convierte el audio de una persona hablando en un idioma en otro de la misma persona en otro idioma.

Por último, en todas las aplicaciones descritas el dominio de $\hat{f}*$, o la entrada, era un audio. Sin embargo, en muchas aplicaciones, se quiere convertir otro tipo de entrada en un audio. Estos son problemas de **generación de audio**, y podemos pensarlos como tomar muestras de la distribución de probabilidad $P(x|y)$ de las señales de audio x, dada cierta variable de condicionamiento y. Algunos ejemplos son la síntesis de texto a habla [57], en la que y es un texto que queremos que se lea. Otro ejemplo es el de la generación de música [58], en donde y puede ser una descripción de la canción, o la letra de la misma, género musical, etc.

2.2.1. Aprender de los datos

Según Arthur Samuel, el aprendizaje automático, o Machine Learning, es el campo de estudio que le brinda a las computadoras la habilidad de aprender sin ser explicitamente programadas [59]. Tom Mitchell agrega detalles sobre qué es el concepto de aprender: "Un programa de computadora se dice que aprende de una experiencia E con respecto a un conjunto de tareas T y medida de desempeño P, si su desempeño P medido en T mejora con la experiencia E" [60].

La experiencia suele provenir de procesar un conjunto de datos de entrenamiento, el cual puede contener etiquetas o no, definiendo si el aprendizaje es **supervisado**

o **no supervisado**[3]. Como ejemplo, podemos tener millones de audios, de los que no sabemos qué contienen (es decir no tenemos etiquetas), o podemos tener quizás cientos de audios de los que tenemos anotados los eventos acústicos que ocurrieron en cada uno (es decir tenemos etiquetas). El uso de millones para el caso sin etiqueta, y cientos para el caso con etiqueta, es porque en la práctica, contar con etiquetas implica un costo no menor, que suele ser el de anotar el dato. Anotar unas pocas horas de audio contratando anotadores expertos o no expertos, dependiendo de la tarea y calidad de anotación requerida, es costoso pero realizable. Sin embargo, en general cuanto más datos se tengan, mejor será el desempeño P de los modelos; y actualmente los datos abundan. Por ejemplo, Susan Wojcicki, CEO de YouTube, comentó en la VidCon 2015 que cada minuto se suben 400 horas de video a la plataforma. Esto significa que en un día, hay más de medio millón de horas de datos nuevos. Etiquetar esta cantidad de datos sería impracticable, lo cual trae a relevancia el potencial del aprendizaje no supervisado, el cual será un tema central a lo largo de esta tesis, justamente por su potencial para aprender modelos usando cantidades masivas de datos.

Habiendo clarificado qué es la experiencia E en el contexto de la definición de Mitchell de aprendizaje automático, nos queda especificar el conjunto de tareas T y medida de desempeño P. En cuanto a las tareas T, muchos ejemplos fueron dados en la sección 2.2 en el contexto de audio, y las formas de medir el desempeño P dependerán de cada tarea y contexto de uso. Posteriormente en el texto se presentarán tareas específicas de audio y las métricas utilizadas para medir el desempeño en ellas.

Existen muchos algoritmos de aprendizaje automático, tanto para aprendizaje supervisado como no supervisado. En esta tesis doctoral nos centraremos en las técnicas de aprendizaje profundo. Antes de entrar en el tema, podemos volver a la motivación inicial de encontrar la $\hat{f}^* : \mathcal{X} \to \mathcal{Y}$, que resuelve nuestra tarea de interés. En el marco del aprendizaje automático, se define un espacio de hipótesis $\mathcal{H}$, el cual es un conjunto de funciones candidatas $\hat{f} : \mathcal{X} \to \mathcal{Y}$. Queremos encontrar una $\hat{f}^* \in \mathcal{H}$ que mejor aproxime a la f verdadera bajo algún criterio. A modo de ejemplo, en regresión lineal, el espacio de hipótesis es

$$\mathcal{H} = \{\hat{f}(x) = ax + b \mid a, b \in \mathbb{R}\},$$

donde a y b son parámetros del modelo. A su vez, dado que tenemos un conjunto de datos,

$$\mathcal{D} = \{(x_i, y_i)\}_{i=1}^{n}, \quad x_i \in \mathcal{X},\ y_i \in \mathcal{Y},$$

podemos suponer que $f(x_i) = y_i$, y por ende, como queremos que $\hat{f}^*$ aproxime bien a f, idealmente querríamos una función $\hat{f}^*$ tal que $\hat{f}^*(x_i) = y_i$ para todo i. Para esto, definimos una función de costo L tal que alcance un valor mínimo cuando esto ocurre. Un ejemplo es el error cuadrático medio:

$$\mathcal{L}(\mathcal{D}, \hat{f}) = \frac{1}{|\mathcal{D}|} \sum_i (y_i - \hat{f}(x_i))^2$$

[3] Si bien está fuera del alcance de este trabajo, existen otras formas de adquirir experiencia. Por ejemplo, en el **aprendizaje por refuerzo**, la experiencia proviene de la interacción del algoritmo de aprendizaje y un entorno.

Y la función $\hat{f}^*$ que buscamos será:

$$\hat{f}^* = \arg\min_{\hat{f}\in\mathcal{H}} \mathcal{L}(\mathcal{D},\hat{f}).$$

o específicamente para el caso de regresión lineal, siendo $\theta = \{a,b\}$:

$$\hat{f}^* = \arg\min_{\theta} \mathcal{L}(\mathcal{D},\theta).$$

2.2.2. Redes neuronales artificiales

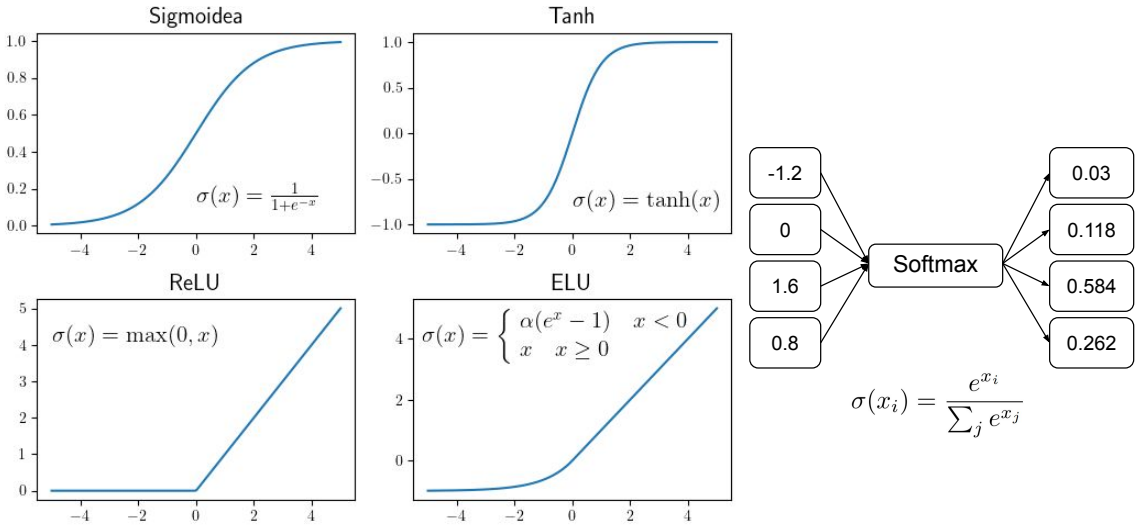

Figura 2.5: Funciones de activación comúnmente utilizadas: función sigmoidea o logística, tangente hiperbólica, ReLU, ELU y softmax.

Si bien el modelo de regresión lineal es extremadamente útil, popular y fácil de interpretar y ajustar, resulta demasiado sencillo para muchos problemas al asumir linealidad; o en términos de aprendizaje automático, el modelo está muy sesgado lo que provoca un subajuste. En cambio, en esta tesis se hará mayormente uso de redes neuronales artificiales, cuyo espacio de hipótesis es mucho más expresivo y contiene funciones que pueden aproximar a cualquier función continua definida en un conjunto compacto en un espacio n-dimensional [61].

La capa de una red neuronal artificial con n neuronas puede definirse como:

$$\hat{f}(x) = \sigma(Wx + B)$$

En donde $x \in \mathbb{R}^d$ es la señal de entrada con d dimensiones, $W \in \mathbb{R}^{n\times d}$ es una matriz conteniendo los pesos de cada una de las n neuronas de la capa, $B \in \mathbb{R}^n$ es un vector de sesgos (o bias), y σ es una función de activación. Notese que si σ es la identidad, $\hat{f}(x)$ es un modelo lineal, no más expresivo que el de regresión lineal. En la práctica, σ introduce no linealidades al modelo, siendo las funciones sigmoidea, softmax, ReLU y algunas de sus variantes como ELU o Leaky ReLU

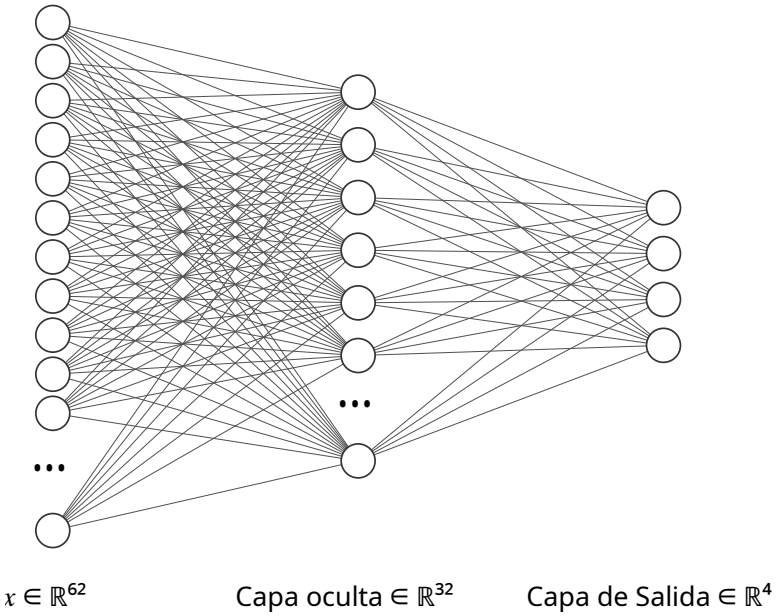

Figura 2.6: Diagrama de la red neuronal con una capa oculta descrita. Cada neurona se conecta con todas las de la capa previa.

las más utilizadas. En la Figura 2.5 se pueden ver gráficas y expresiones de estas funciones; nótese que la elección de σ determinará la imagen de $\hat{f}$, por ejemplo, si σ es sigmoidea, los valores de salida estarán en el intervalo (0,1).

Supongamos que nuestra tarea es predecir la emoción de un segmento de habla, siendo las emociones posibles: felicidad, enojo, tristeza, o ausencia de emociones (neutralidad). Este problema es de clasificación con 4 clases, y por ende queremos una función de la forma:

$$\hat{f} : \mathbb{R}^n \to \{0,1,2,3\}$$

Un espacio de hipótesis posible es:

$$\hat{f}(x) = \arg\max_k \text{softmax}(Wx + B)$$

En donde $x \in \mathbb{R}^{62}$ son atributos resumen de eGeMAPS calculados sobre un segmento de habla, $W \in \mathbb{R}^{4\times62}$ es la matriz de pesos y $B \in \mathbb{R}^4$ es el vector de sesgos. Es decir, estamos utilizando una capa de red neuronal para transformar los atributos x y obtener 4 valores. La función softmax actúa como función de activación, y posee la propiedad de que devuelve valores en el intervalo (0,1) cuya suma es 1. Esto permite pensar a la salida como una distribución de probabilidad, particularmente en este caso, $P(y = c_k|x)$; la probabilidad de que la clase y sea c_k dados los atributos de entrada x. Finalmente, la clase predicha es aquella con mayor probabilidad. Particularmente, este modelo se corresponde con una regresión logística multinomial, para la cual se maximiza la verosimilitud al utilizar entropía cruzada como función de costo. En términos de redes neuronales, es un modelo que cuenta solamente con una capa de salida (transforma su entrada en las salidas).

En un perceptrón multicapa (MLP), se agregan una o más capas ocultas antes de la de salida. Estas capas transforman la señal de entrada x en una representación intermedia h, la cual es finalmente transformada en y por la capa de salida. Podemos ampliar el modelo anterior agregando una capa oculta con 32 neuronas y activación ReLU, resultando en:

$$f(x) = \arg\max_k \text{softmax}(W_2 \ \text{ReLU}(W_1 x + B_1) + B_2)$$

En donde $W_1 \in \mathbb{R}^{32\times 62}$ y $B_1 \in \mathbb{R}^{32}$ son los pesos y sesgos de la capa oculta, y $W_2 \in \mathbb{R}^{4\times 32}$ y $B_2 \in \mathbb{R}^4$ son los pesos y sesgos de la capa de salida. Se puede obtener fácilmente la cantidad de parámetros de este modelo, calculando la cantidad total de elementos de las matrices de pesos y sesgos, lo cual en este caso da 2144 parámetros.

El uso en el ejemplo de atributos de eGeMAPS como entrada en lugar de la forma de onda es deliberado. Si quisiéramos modelar directamente la forma de onda, y nos limitáramos, por ejemplo, a audios de 1 segundo con una frecuencia de muestreo de 16kHz, eso implicaría una entrada $x \in \mathbb{R}^{16000}$. Cada neurona que agreguemos en la capa oculta supondrá añadir 16001 parámetros al modelo. A su vez, el modelo tendría que ser capaz de extraer mediante una transformación lineal y una activación no lineal, patrones que sean útiles para la clasificación. En la práctica, para aprender a extraer patrones útiles de una forma de onda se suele requerir funciones más expresivas, las cuales podríamos obtener agregando más capas ocultas. Sin embargo, en la práctica esto implica una explosión de parámetros, complicando la búsqueda de $\hat{f}*$ en el espacio de hipótesis. En las siguientes secciones, se explicarán otros tipos de redes neuronales que escalan mejor con la profundidad y cantidad de parámetros, y suelen utilizarse extensivamente en el campo del aprendizaje profundo.

Volviendo al ejemplo, tenemos definido el espacio de hipótesis $\mathcal{H}$, y ahora nos queda definir un conjunto de datos $\mathcal{D}$ y una función de costo $\mathcal{L}$. Existen varios conjuntos de datos en la literatura, que cuentan con segmentos de habla etiquetados según la emoción. A modo de ejemplo podemos seleccionar Interactive Emotional Dyadic Motion Capture (IEMOCAP) [62], el cual cuenta con unas 12 horas de habla emocional de 10 actores distintos. Respecto a la función de costo, deseamos encontrar los parámetros que maximicen la verosimilitud $P(Y = y, X = x|\theta)$, donde y es la clase correcta y viene dada por el conjunto de datos $\mathcal{D}$. Normalmente en problemas multiclase como este, se utiliza la entropía cruzada, que al minimizarse maximizará la verosimilitud:

$$\mathcal{L}(\mathcal{D}, \theta) = -\frac{1}{|\mathcal{D}|} \sum_{(x, c_{\text{true}})\in\mathcal{D}} \log \hat{f}_{c_{\text{true}}}(x; \theta)$$

Por último queda definir cómo encontrar la $\hat{f}^*(x) \in \mathcal{H}$ que minimice la entropía cruzada. Este proceso de encontrar $\hat{f}^*$ es lo que se denomina **entrenamiento**, e implica ajustar los parámetros de la red neuronal de forma que minimicen la función de costo. Se suele utilizar la técnica de **descenso por gradiente**, en la que se ajustan iterativamente los parámetros θ moviéndolos en la dirección opuesta al gradiente:

$$\theta_{t+1} = \theta_t - \lambda \nabla_{\theta_t} \mathcal{L}(\mathcal{D}, \theta_t)$$

La tasa de aprendizaje λ, controla cuánto se ajustan los parámetros en cada iteración del descenso por gradiente. Para calcular el gradiente de la función de costo respecto a los parámetros, se utiliza la técnica de retropropagación de error (o backpropagation en inglés), la cual implica pasar las entradas x por la red neuronal, es decir aplicar un paso hacia adelante (o forward pass), obteniendo el valor de la función de costo y almacenando en memoria los valores de las activaciones. Luego se realiza un paso hacia atrás (backward pass), en el que por regla de la cadena se van calculando los gradientes y ajustando los parámetros θ.

En la práctica hay que tener algunas consideraciones:

- Calcular el gradiente utilizando el conjunto de datos completo (12 horas de audio de IEMOCAP), en general no es posible debido a los requerimientos de cómputo. En la práctica, cada iteración del descenso por gradiente se hace sobre una muestra de tamaño N del conjunto de datos. Esta muestra se toma de forma aleatoria sin reposición hasta agotar el conjunto de datos y completar una **época**. Cuando N, el tamaño de lote (o batch size), es 1, hablamos de descenso por gradiente estocástico (SGD), y cuando $N > 1$ pero menor al tamaño del conjunto de datos, se habla de descenso por gradiente con minibatches.
- Al mismo tiempo, utilizar una sola vez el conjunto de datos, lo cual equivale a una época, no siempre es suficiente para que la función de costo converja. En ese caso, se suele continuar el entrenamiento repitiendo el conjunto de datos en un órden aleatorio diferente.
- El método de descenso por gradiente no garantiza encontrar el mínimo global si la función a optimizar no es convexa. La configuración de tasa de aprendizaje y tamaño de lote, puede ayudar a 'escapar' de algunos mínimos locales. A su vez, se utilizan variantes del descenso por gradiente como ADAM [63], que aplican técnicas como el uso de momento y la posibilidad de ajustarlo de manera distinta para cada parámetro, ayudando a evitar mínimos locales y puntos de ensilladura.
- La tasa de aprendizaje no necesariamente es un valor constante, sino que puede ir ajustándose a lo largo del entrenamiento.

Por último, una vez finalizado el entrenamiento, hay que evaluar la $\hat{f}^*$ encontrada. Para eso, debemos utilizar un conjunto de datos distinto, que no haya sido utilizado durante el entrenamiento. Esto es porque lo que queremos medir es la capacidad de generalización de nuestros modelos, es decir, no cuan bien funciona en los datos con los que se entrenó, sino en nuevos datos que encontrará el modelo cuando se utilice en la práctica. Además debemos definir una métrica de evaluación que sea relevante para el problema que estamos queriendo resolver. Una métrica muy utilizada es la precisión (o accuracy):

$$Acc(y, \hat{y}) = \frac{1}{N} \sum_{i=1}^{N} \mathbb{1}(y_i = \hat{y}_i)$$

en donde, N es la cantidad de muestras de evaluación, y son las etiquetas

verdaderas, $\hat{y}$ las clases predichas por el modelo, y $\mathbb{1}$ es una función indicadora que vale 1 cuando la etiqueta y predicción coinciden, y 0 en caso contrario.

Podría ocurrir que las métricas mejoren en el conjunto de entrenamiento pero empeoren en el de evaluación, en cuyo caso, se dice que el modelo sufre de un **sobreajuste** a los datos de entrenamiento, ya que no solo está capturando los patrones relevantes para poder generalizar, sino que también los irrelevantes (o ruido), llevándolo a realizar predicciones erróneas en un conjunto de datos no vistos.

En el otro extremo, tanto las métricas de entrenamiento como evaluación pueden no ser satisfactorias, en cuyo caso, el modelo puede estar subajustandose a los datos de entrenamiento, es decir, no tener suficiente capacidad para encontrar los patrones relevantes para la tarea, o puede haber mucho error irreducible, por lo que incluso el mejor modelo posible no es capaz de lograr un buen desempeño. Por ejemplo, si usamos regresión lineal pero la f que queremos estimar es no lineal, el modelo sufrirá de subajuste debido a su capacidad expresiva limitada. Por otro lado, si se solapan las distribuciones de cada clase, como cuando $P(c_1|x) = P(c_2|x)$, el mejor clasificador posible elegirá a la clase por azar.

En general, queremos hacer una **búsqueda de hiperparámetros**, como por ejemplo, la cantidad de capas, la cantidad de neuronas en la capa oculta, los atributos expertos que usaremos en las entradas, la tasa de aprendizaje o tamaño de lote. Estos hiperparámetros definen distintos espacios de hipótesis $\mathcal{H}$ con distintos niveles de sesgo y varianza, y también definen distintas formas de buscar $\hat{f}^*$ en esos espacios. Si probamos muchas combinaciones de hiperparámetros y elegimos la que de mejor métrica en el conjunto de evaluación, podemos pensar este proceso como una nueva optimización o búsqueda (solo que manual en vez de por medio del descenso por gradiente), y podríamos terminar sobreajustando al conjunto de evaluación. Es por esto que en la práctica necesitaremos tres conjuntos de datos:

- Conjunto de entrenamiento: en este conjunto se realiza el entrenamiento, es decir se busca $\hat{f}^* \in \mathcal{H}$. Se suele detener el entrenamiento cuando la función de costo deja de decrecer, o luego de una cantidad fija de épocas o iteraciones.
- Conjunto de validación: en este conjunto se evalúan las distintas combinaciones de hiperparámetros exploradas, y en general la combinación con mejor desempeño en este conjunto será la seleccionada.
- Conjunto de evaluación: finalmente, el modelo con el mejor desempeño en el conjunto de validación es evaluado en un nuevo conjunto de datos, que no ha sido utilizado durante la fase de desarrollo (entrenamiento y validación).

En la práctica, esta división puede venir dada por el conjunto de datos, o la debe realizar uno. Es importante tener en cuenta algunos detalles a la hora de dividir los datos, como por ejemplo el balance de clases, y si se desea evaluar con el mismo balance que en entrenamiento. Una opción es hacer una división estratificada, manteniendo el balance en los tres conjuntos. Otro detalle importante es definir qué es lo que queremos que generalice. Por ejemplo, en el caso de IEMOCAP se tienen 10 actores distintos. Uno desearía que el modelo pueda detectar emociones en personas distintas a estos actores. Una forma de simular el escenario, es utilizar

actores distintos a los de entrenamiento para la evaluación. Un error muy común en el area es dividir de manera aleatoria, lo que provoca que segmentos de habla pertenecientes a un hablante, esten tanto en el conjunto de entrenamiento como el de evaluación, y se dice que hay una **fuga de datos**.

Por último, muchos conjuntos de datos son muy pequeños, por lo que se intenta utilizar la mayor cantidad de datos posibles para el entrenamiento, y por consiguiente, se complica tener datos suficientes para validar y evaluar. En esos casos, se suele utilizar la técnica de **validación cruzada de K iteraciones**, la cual consiste en dividir los datos en K subconjuntos disjuntos, entrenando en $K-1$ subconjuntos, y evaluando en el restante. Este proceso se repite K veces, de modo que se haya hecho predicciones en cada uno de los K subconjuntos de evaluación. Finalmente se reporta el desempeño calculado sobre la unión de las predicciones en los K subconjuntos evaluados.

2.2.3. Redes neuronales convolucionales

Las redes neuronales convolucionales (CNN) fueron probablemente las primeras que escalaron exitosamente en términos de profundidad, parámetros y cantidad de datos, produciendo una revolución en 2012 en el campo de visión por computadora con AlexNet [64]. Este modelo mejoró ampliamente el estado del arte en la competencia de clasificación de imágenes ImageNet [65], demostrando el potencial de los modelos de aprendizaje profundo, y sentando las bases de las arquitecturas modernas de redes neuronales convolucionales.

Las neuronas en una capa densamente conectada de la subsección 2.2.2, responden a la entrada completa, lo cual dificulta su escalado a entradas de mayor dimensionalidad como imágenes o audio. Las capas convolucionales, en cambio, solo responden a una región de la entrada, introduciendo un sesgo de localidad. A su vez, debido al uso de parámetros atados, son invariantes a la traslación; si desplazo la región de la entrada, se obtendrá la misma salida desplazada. Estos dos sesgos son muy útiles, especialmente en el dominio de imagenes, ya que por ejemplo, quiero poder detectar un rostro sin importar si está en una esquina de la imagen, o en el centro. También pixeles que están cerca probablemente estén correlacionados, haciendo útil al sesgo de localidad. Al mismo tiempo, atar parámetros y responder solo a una región de la entrada permite reducir significantemente la cantidad de parámetros en comparación con un perceptrón multicapa.

En el caso de una capa convolucional 2D, para una entrada $x \in \mathbb{R}^{H \times W \times C}$ con H elementos de alto, W de ancho, y C canales, cada elemento de salida $y[i,j,k]$ será el resultado de:

$$y[i,j,k] = \sigma\left(\sum_{c=0}^{C-1}\sum_{h=0}^{K_H-1}\sum_{w=0}^{K_W-1} x[i \cdot S_h + h, j \cdot S_w + w, c] \cdot K[h,w,c,k] + b[k]\right)$$

En donde $K \in \mathbb{R}^{K_H \times K_W \times C \times K_C}$ es un **kernel** con K_C canales de salida o filtros y tamaño $K_H \times K_W$, y $b \in \mathbb{R}^{K_C}$ es equivalente al bias en una red MLP. σ es la función

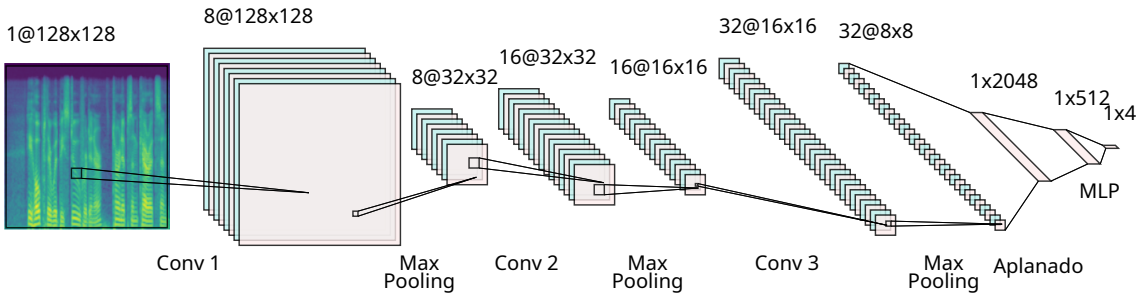

Figura 2.7: Ejemplo de una red convolucional tomando un espectrograma como entrada para clasificación.

de activación, y S_h, S_w definen el salto o **stride**. Un paso o stride mayor a 1 permite reducir la dimensionalidad de la entrada. y, denominado mapa de activaciones, tendrá una dimensionalidad $\left\lfloor \frac{H-K_H}{S_h} \right\rfloor + 1 \times \left\lfloor \frac{W-K_W}{S_w} \right\rfloor + 1 \times K_C$. Muchas veces, para evitar una reducción en el tamaño de entrada (al no usar stride) debido a que los filtros se escapan de los bordes, usualmente se agregan $(K_H - 1)/2$ y $(K_W - 1)/2$ ceros en cada lado de la imagen.

Al igual que con las capas densamente conectadas, se suelen utilizar múltiples capas convolucionales en serie. Existen muchos patrones de diseño que se utilizan tanto en imágenes como en procesamiento de audio.

En la Figura 2.7 se puede ver un ejemplo de clasificador de audio utilizando una arquitectura CNN. La entrada suele ser alguna representación tiempo frecuencia, como la magnitud de la STFT, un melspectrograma o la magnitud de una CQT. En este ejemplo, es un melspectrograma con 128 filtros mel y 128 cuadros. Luego, un patrón común de diseño [66, 67] consiste en aplicar una serie de capas convolucionales que van a incrementar la cantidad de canales y aprender a detectar patrones en el espectrograma, seguido de una capa de agrupamiento por máximo, que desliza una ventana normalmente sin solapamiento y la reduce a un solo valor que es el máximo. Otra opción para reducir la dimensionalidad de los mapas de activaciones es usar capas convolucionales con paso mayor a 1. Este patrón se aplica varias veces, incrementando la cantidad de canales al mismo tiempo que se achican los mapas de activación. Esto se puede observar en la Figura 2.7, con cada capa convolucional incrementando la cantidad de canales (de 1 a 8, de 8 a 16 y de 16 a 32), y cada capa de agrupamiento por máximo, o max pooling, reduciendo el tamaño de los mapas de activaciones en factores de 4, 2 y 2. Finalmente, se aplana el último mapa de activaciones transformándolo en un vector de 2048 atributos que sirve de entrada a un perceptrón multicapa que devuelve $P(c_k|x)$. El sistema se entrena conjuntamente de punta a punta, y se puede pensar que las capas convolucionales agregadas al principio de la red, sirven como un extractor de atributos que se aprende de los datos, y luego estos atributos son usados como entrada para un clasificador MLP. Este concepto, de las redes profundas aprendiendo a extraer atributos o representaciones de forma automática, es clave en el campo del aprendizaje profundo y es central en esta tesis.

A lo largo del tiempo, se han propuesto distintas variantes de este patrón de diseño, como por ejemplo, el uso de conexiones residuales, que dió origen a ResNet [68], el uso de capas de convolución transpuesta o upsampling para poder resolver problemas de, por ejemplo, segmentación de imágenes [69] o en el campo del audio, separación de fuentes [70] y reducción de ruido [71], o el desarrollo de capas convolucionales con menor cantidad de parámetros, como las convoluciones separables, que pueden ser utilizadas en sistemas embebidos con limitaciones de hardware [72].

Por último, el uso de convoluciones 1D es cada vez más común en audio, siendo utilizado para procesar directamente la forma de onda en la entrada, o para procesar secuencias de atributos, como podrían ser los LLD de eGeMAPS. Una de las primeras arquitecturas utilizada exitosamente para modelar la forma de onda es WaveNet [73], la cual utiliza filtros dilatados, es decir, los elementos que se multiplican con el filtro no son contiguos sino que están separados por K_D muestras, donde K_D es el factor de dilatación. Esta modificación permite ampliar el campo receptivo de forma exponencial con la profundidad, lo cual es importante a la hora de trabajar con formas de onda que poseen una alta resolución temporal.

Otra motivación para utilizar convoluciones 1D sobre la forma de onda, es reemplazar el cálculo de espectrogramas por una o más de estas capas convolucionales, que pueden aprender una representación mejor alineada con la tarea de interés. Por ejemplo, ConvTasNet [74] utiliza una capa convolucional inicial con un tamaño de filtro y paso imitando el tamaño de ventana y salto de la STFT. Los autores muestran que el tipo de representación aprendida se alinea con principios de percepción auditiva, al por ejemplo poseer mayor resolución en baja frecuencia que en alta, y capturar información de fase principalmente en baja frecuencia. Por otro lado, Wav2Vec 2.0 [75] y HuBERT [76] utilizan una red convolucional 1D de 7 capas para transformar la forma de onda en una secuencia de 50 atributos por segundo, que luego será procesada por una red transformer.

2.2.4. Transformers

Así como la adopción de redes neuronales convolucionales revolucionó inicialmente al área de visión por computadora, y luego se trasladó a otros dominios como el audio y el procesamiento del lenguaje natural, las redes transformers, introducidas en el año 2017 revolucionaron inicialmente el campo del NLP y posteriormente otros campos.

El uso de Transformers da algunas ventajas a la hora de modelar secuencias respecto a las RNN o LSTM, que eran los modelos más utilizados previamente en ese escenario:

- Permite paralelizar el cálculo ya que las operaciones son matriciales y no hay un estado que se actualiza secuencialmente en el tiempo.
- Una capa transformer tiene acceso directo a toda la secuencia de entrada para generar cada elemento de la secuencia de salida. En contraste, para el caso de una RNN, si por ejemplo el último elemento de la secuencia quiere utilizar

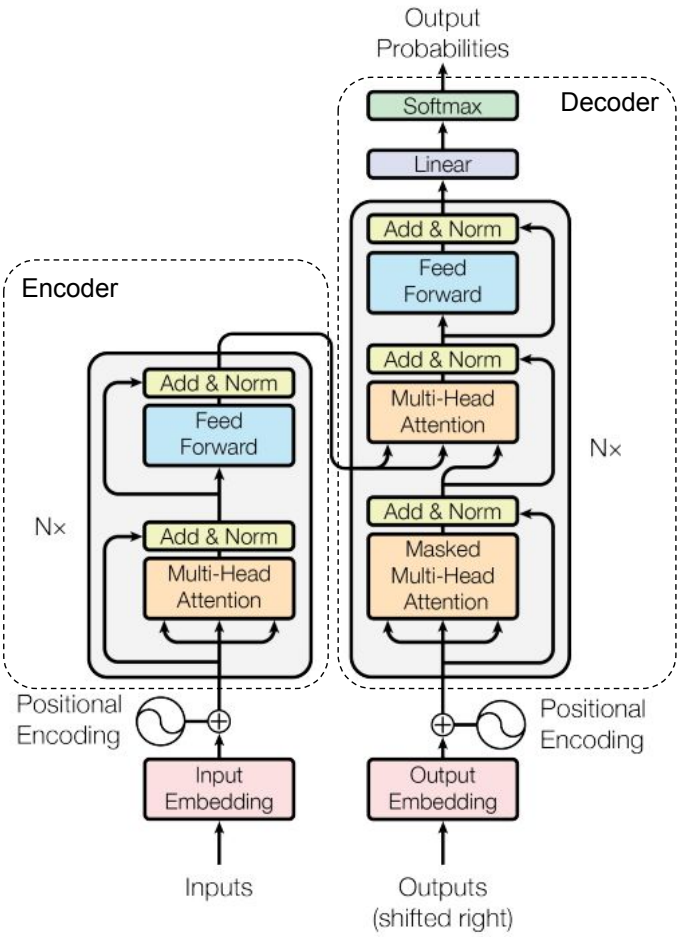

Figura 2.8: Diagrama de la arquitectura Transformer. Adaptado de [2].

información del primero, deberá guardarla en el estado y conservarla a medida que se procesa la secuencia.

- La complejidad computacional es cuadrática con la longitud n y lineal con la dimensionalidad d, es decir, $\mathcal{O}(n^2 d)$, mientras que en una RNN es $\mathcal{O}(nd^2)$. Esto hace que utilizar transformers sea más eficiente en el caso en que la longitud de las secuencias es menor que la dimensionalidad de la misma.

Originalmente la arquitectura transformer fue desarrollada para resolver el problema de traducción automática. Este es un problema de transducción de secuencias, también llamado sequence to sequence (seq2seq), en el que se quiere tomar un texto en un idioma, con una cierta longitud, y generar su traducción, con potencialmente una longitud distinta. En la Figura 2.8 se puede ver un esquema de la arquitectura transformer, la cual consiste de un encoder y un decoder. El encoder tiene la finalidad de procesar la secuencia de entrada (frase a traducir) y generar una representación de la misma, que luego el decoder utilizará para generar de manera autoregresiva la secuencia de salida (traducción). Las capas de embedding de entrada y salida son específicas del problema de traducción debido a que se trabaja con texto, y es

necesario asignar vectores a cada una de las V palabras, subpalabras o símbolos del vocabulario. Con este fín, se aprenden matrices de tamaño $V \times D_{model}$, donde D_{model} es la dimensionalidad de los vectores.

El bloque central de la arquitectura transformer es la atención de producto interno, la cual tiene la siguiente forma:

$$Att(Q,K,V) = \text{Softmax}(\frac{QK^T}{\sqrt{d_k}})V,$$

en donde Q es una matriz de consultas o queries, K es una matriz de claves o keys, y V es una matriz de valores o values. En el encoder, tanto Q como K y V son transformaciones lineales de la secuencia de entrada x. Este caso se conoce como self-attention, y se aprenden matrices de pesos W_q, W_k y W_v para realizar estas transformaciones:

$$Q = xW_q, \quad K = xW_k, \quad V = xW_v$$

En el caso del decoder, K y V son transformaciones lineales de la salida del encoder, mientras que Q proviene de la entrada. Este caso se conoce como atención cruzada. Intuitivamente, el mecanismo de atención aprende a realizar un promedio pesado de los valores V para calcular cada elemento de la secuencia de salida. Los pesos de atención vienen dados por el producto escalar entre el query y cada uno de los keys, y la función softmax asegura que la suma de los pesos sea 1. Una de las innovaciones de la arquitectura transformer fue utilizar múltiples cabezas de atención, es decir, se aprenden tantas proyecciones W_{qi}, W_{ki} y W_{vi} como cabezas de atención se tengan. Esto da origen a múltiples secuencias de salida o_i, las cuales se concatenan y se proyectan a la dimensionalidad original mediante una matriz $W_o \in \mathbb{R}^{N_h D_q \times D_{model}}$, donde N_h es la cantidad de cabezas de atención y D_q es la dimensionalidad de los queries originales. A su vez, en el caso del decoder, como se busca que genere la secuencia de salida de manera autoregresiva, es necesario forzar causalidad, evitando que se utilicen keys de una posición mayor a la del query. Para realizar esto, se utiliza una máscara de atención que tiene una forma de matriz triangular inferior y coloca en cero a los pesos de atención que violan causalidad. Si bien la formulación inicial de transformers utiliza atención cruzada, debido a que en la tarea de traducción hay una idea de secuencia de entrada y salida, en otros modelos decoder-only, como GPT [77], la atención cruzada está ausente y es el caracter autoregresivo y la máscara causal las que definen al decoder.

El mecanismo de atención no posee información sobre las posiciones de los elementos en la secuencia; si se permutaran los queries se tendría la misma salida permutada. Si queremos modelar secuencias es importante tener en cuenta esta información posicional, por lo que existen diversos mecanismos en la literatura para incorporarla a la red Transformer. Originalmente se utilizaron embeddings posicionales absolutos, en donde cada posición tiene asociado un vector único que se suma a la secuencia de entrada, agregándole la información posicional necesaria. En el paper original, estos embeddings son funciones sinusoidales de la siguiente forma:

$$PE_{(pos,2i)} = \sin\left(\frac{pos}{10000^{\frac{2i}{d_{\text{model}}}}}\right)$$

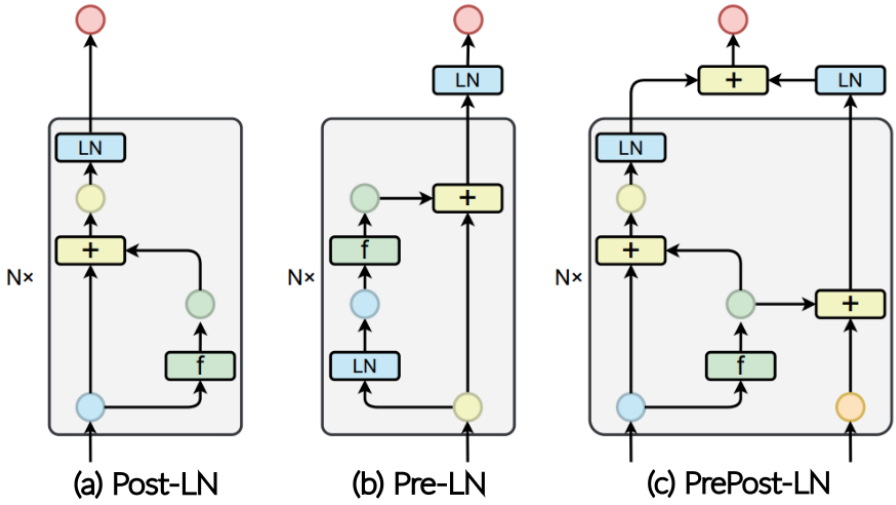

Figura 2.9: Distintas disposiciones de la capa de normalización y las conexiones residuales. Extraído de [83].

$$PE_{(pos,2i+1)} = \cos\left(\frac{pos}{10000^{\frac{2i}{d_{\text{model}}}}}\right)$$

En trabajos posteriores se exploraron otros tipos de embeddings, como aprendidos por la red [78] y rotativos [79].

Otra alternativa para introducir información posicional es usar embeddings relativos. En lugar de asociar cada posición absoluta con un vector, se modifica el mecanismo de atención agregándole información de la posición relativa (o distancia) entre el query y los keys. En la atención de Shaw [80, 81], se aprende un tensor de posiciones relativas $E^r \in \mathbb{R}^{T \times D_{model}}$, y a partir de él se calcula un tensor $R \in \mathbb{R}^{T \times T \times D_{model}}$ que para cada distancia relativa Δt entre queries y keys toma el vector $E^r[\Delta t]$. Finalmente, el cálculo de atención se modifica de la siguiente manera:

$$Att(Q,K,V) = \text{Softmax}(\frac{QK^T + S^{rel}}{\sqrt{d_k}})V$$

donde $S^{rel} = QR^T$. Es decir, se suma a la matriz de atención original, otra matriz dependiente de la distancia entre keys y queries. Se han propuesto en la literatura muchas formas de calcular S^{rel}, por ejemplo, en el caso de ALiBi [82] se utiliza un S^{rel} fijo, cuyo valor disminuye al aumentar la distancia relativa, introduciendo un sesgo de localidad; se le da mayor peso a interacciones entre elementos cercanos en la secuencia. A su vez, introducen pendientes distintas para cada cabeza de atención, promoviendo que algunas cabezas tengan un mayor sesgo de localidad que otras.

Luego de aplicarse el mecanismo de atención hay una conexión residual y se aplica un MLP, el cual introduce la mayor parte de los parámetros en un Transformer. A modo de ejemplo, en cada bloque de transformer de Llama 3 8B, 176M de 218M de parámetros (el 80.7%) pertenecen a este MLP. La red MLP suele consistir de dos capas densamente conectadas con forma de cuello de botella invertido, por ejemplo, en el paper original lleva la dimensionalidad de 512 a 2048 (la primer capa tiene 2048

neuronas), aplica activación ReLU, y luego la siguiente capa con 512 neuronas lleva la dimensionalidad nuevamente a 512. En el transformer propuesto originalmente, cada bloque posee dos conexiones residuales seguidas de Layer Normalization [84]. Posteriormente, distintos trabajos han encontrado que esta configuración, denominada post-normalización, lleva a inestabilidades en el entrenamiento y a la necesidad de incrementar progresivamente la tasa de aprendizaje al comienzo del entrenamiento (warmup). En consecuencia, se han propuesto alternativas, siendo la más común la de pre-normalización; es decir, primero se normaliza la salida de cada bloque, y luego se suma en la conexión residual. Recientemente, se ha propuesto combinar ambos métodos de normalización en lo que se denomina conexiones residuales duales (ResiDual) [83], y los autores han mostrado que el método es superador de los anteriores en términos de estabilidad en el entrenamiento y desempeño alcanzado. A lo largo del escrito, nos referiremos a este método como pre-post normalización. En la Figura 2.9 se muestran las distintas configuraciones de normalización presentadas.

Por último, si bien las primeras aplicaciones de transformers fueron exitosas en el campo del procesamiento del lenguaje natural, con la introducción de la arquitectura Vision Transformer [85], también se ha vuelto popular su utilización en el contexto de visión por computadora, y por consiguiente, en el dominio del procesamiento de audio si se reemplaza la imagen por un espectrograma. En un Vision Transformer (ViT), para poder utilizar imágenes en la entrada, se dividen en patches rectangulares (originalmente de 16x16 pixeles), y cada patch se aplana formando un vector de 256 dimensiones, y proyecta con una transformación lineal a la dimensionalidad con la que el transformer va a trabajar (d_{model}). Finalmente una imagen de entrada se expresa como una secuencia de patches, se agrega el embedding posicional, y es procesado por el encoder de un transformer. Cuando la tarea es de clasificación, se antepone un token de clasificación a la secuencia de patches de entrada, y se agrega una red MLP que toma como entrada este token de clasificación a la salida y devuelve la clase predicha. Una alternativa al token de clasificación es promediar la secuencia de salida en el eje temporal obteniendo un vector único. La arquitectura Audio Spectrogram Transformer (AST) [86], es un ejemplo de esta idea llevada al campo del procesamiento del audio para resolver la tarea de detección de eventos acústicos. En la siguiente sección, se presentarán distintos modelos de audio que suelen combinar transformers y capas convolucionales para aprender representaciones de habla y audio.

2.3. Aprendizaje de representaciones

> "Si he visto más, es poniéndome sobre los hombros de Gigantes."
>
> *Isaac Newton*

Según Bengio et al. [87], el aprendizaje de representaciones consiste en aprender representaciones de los datos que faciliten la extracción de información a la hora

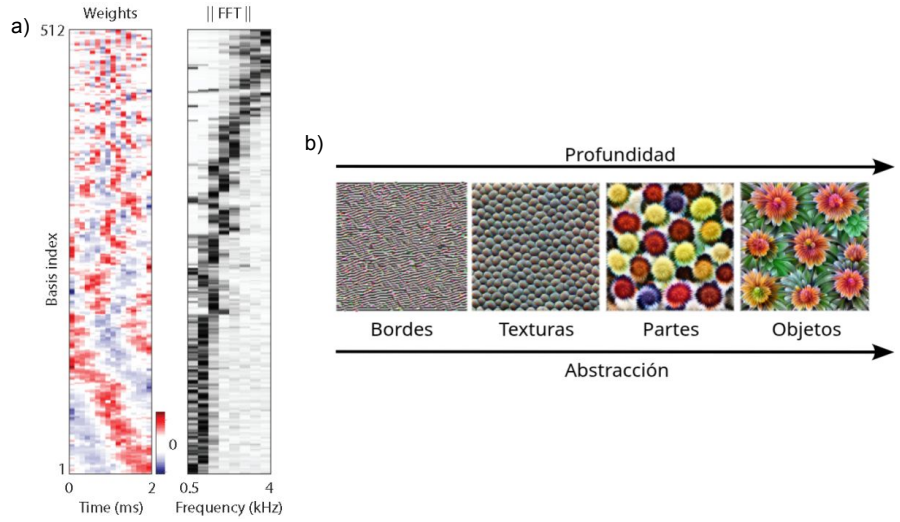

Figura 2.10: a) Ejemplo de kernels junto a la magnitud de su espectro, aprendidos por la primera capa convolucional de ConvTasnet [74]. b) Ejemplo de mapas de atributos en una red convolucional de imágenes (adaptado de [93])

de construir clasificadores u otros predictores. En el libro del aprendizaje profundo [88] se da el ejemplo de hacer una división aritmética con números romanos, la cual es una tarea que se simplifica si representamos los números en el sistema arábico. En la sección 2.2.1 se mostró como el aprendizaje automático nos permite aprender funciones que transformen una representación de los datos, como pueden ser atributos expertos, en una salida deseada. En esta sección se presentarán técnicas para aprender representaciones de audio utilizando aprendizaje automático. Existe una gran variedad de algorítmos de aprendizaje de representaciones como el análisis de componentes principales (PCA) [89], factorización no negativa de matrices (NMF) [90], análisis de componentes independientes (ICA) [91] y matching pursuit [92]. Sin embargo, pondremos el foco en las técnicas de aprendizaje de representaciones basadas en redes neuronales profundas.

Como se explicó previamente, a medida que las técnicas de aprendizaje profundo fueron mejorando, el diseño de atributos expertos se volvió cada vez menos frecuente. Por ejemplo, con las redes tipo perceptrón multicapa no era posible trabajar directamente con espectrogramas o la forma de onda como entradas, y se hacía necesario extraer atributos a partir de estas señales utilizando conocimiento experto. Sin embargo, con el auge de las redes neuronales convolucionales, se comenzó a trabajar directamente sobre espectrogramas, y posteriormente sobre la forma de onda. Se puede pensar que internamente, el modelo está aprendiendo a extraer atributos, o a representar esos espectrogramas o formas de onda de manera que sean útiles para las siguiente capas, lo que lleva a una organización jerárquica de las activaciones de las capas de redes profundas. Este fenómeno se ilustra en la Figura 2.10a) en la que se muestra cómo los filtros de la primera capa convolucional de un modelo de audio son selectivos a frecuencias específicas, y asignan mayor resolución en bajas frecuencias que en altas. También conservan información de fase en estas bajas fre-

cuencias (los primeros filtros poseen magnitudes de espectro similares pero el kernel muestra un desplazamiento en el tiempo), lo cual conserva similitudes con ciertos atributos perceptuales expertos como los filtros gammatone [94]. En la Figura 2.10b), se muestran activaciones de distintas capas de una red convolucional entrenada con imágenes. Se puede observar que las primeras capas detectan patrones de bajo nivel como bordes y las siguientes comienzan a detectar patrones más abstractos de alto nivel como texturas, partes de objetos u objetos específicos. En estos casos, se han entrenado redes neuronales profundas con el objetivo de resolver una tarea T, y en consecuencia, una vez entrenada la red sus representaciones internas pueden ser pensadas como atributos. Particularmente, si se tiene otra tarea T' que es similar a T, se podría pensar que las representaciones que fueron útiles para resolver T, también lo serán para resolver T'. Muchas veces queremos resolver tareas en las que los datos anotados son escasos, lo cual impide entrenar de forma apropiada una red profunda desde cero. Si contamos con una tarea T en la que disponemos de muchos datos, quizás podemos aprovechar las representaciones aprendidas por un modelo en esta tarea, para obtener un mejor desempeño en la tarea con pocos datos. Esta idea se conoce como **transferencia de aprendizaje**, y tiene una gran relevancia en el aprendizaje automático moderno, permitiendo resolver una gran cantidad y variedad de problemas para los que no se tienen abundantes datos. El proceso de aprender representaciones en la tarea T, se suele denominar **preentrenamiento**. Las representaciones aprendidas pueden ser utilizadas de distintas maneras para resolver T', siendo las dos principales la **extracción de activaciones** y el **finetuning**.

Dada una señal de entrada x, la extracción de activaciones consiste en obtener la salida correspondiente con la red neuronal preentrenada en T y utilizar como representación las activaciones de alguna capa intermedia. La elección de la capa de la que extraer las activaciones es importante, ya que por ejemplo en un modelo preentrenado mediante clasificación de imágenes, las primeras capas representarán conceptos de bajo nivel y más generales (por ejemplo bordes), mientras que las últimas capas, al estar más cerca de la salida que será utilizada para resolver la tarea T, codificarán información de más alto nivel y alineada a la tarea T. Por este motivo, si para la tarea T' se necesita información que para la tarea T no es necesaria, es posible que en las últimas capas no esté disponible. Por ejemplo, si la tarea T es reconocimiento de habla, es esperable que el modelo resultante no necesite información del timbre del hablante para determinar qué se está diciendo. Por ende, si la tarea T' fuera identificación de hablante, es más probable que la información necesaria para discriminar hablantes se encuentre en las primeras capas y no en las últimas que están ajustadas para discriminar fonemas y palabras, y no hablantes. Una vez extraídas las activaciones, estas serán utilizadas como atributos de entrada a otro modelo, normalmente más pequeño, como puede ser un perceptrón multicapa. En la literatura se suele denominar como **upstream** al modelo del que se extraen las activaciones, y **downstream** al modelo entrenado utilizando estas activaciones como atributos de entrada.

Por otro lado, el finetuning consiste en seguir entrenando el modelo upstream en la nueva tarea T'. Es decir, en lugar de entrenar con pesos inicializados aleatoriamente,

se aprovechan los pesos aprendidos por la red en la tarea T. Normalmente es necesario reemplazar la última capa del modelo upstream por una apropiada para T' (por ejemplo, si la cantidad de clases a predecir es distinta), o agregar más capas o un modelo downstream al final del upstream. El principal beneficio de realizar finetuning respecto a extraer activaciones, es que el modelo upstream puede adaptar sus pesos a la tarea T' aprendiendo a mejorar las representaciones que aprendió para T. Un problema de este método es que es computacionalmente más costoso y que se corre el riesgo de que el modelo upstream se sobreajuste a la nueva tarea, "olvidando" los conceptos aprendidos en T. En cierta manera, se replica el balance entre sesgo y varianza que ocurre con cualquier modelo de aprendizaje automático; al hacer finetuning, se tienen más parámetros para optimizar dando lugar a un modelo de mayor varianza que si solo se entrenara el downstream utilizando extracción de activaciones. Algunas estrategias para mitigar este problema, o para controlar el balance sesgo-varianza son: el uso de una tasa de aprendizaje menor para los parámetros del modelo upstream, ajustar solo las últimas capas del modelo upstream, ir gradualmente permitiendo que capas cada vez más profundas se ajusten (gradual unfreezing) [95], o utilizar técnicas de finetuning eficientes en parámetros como LoRA [96].

2.3.1. Aprendizaje supervisado de representaciones

Existen varios trabajos en el campo del audio en los que T es una tarea supervisada, es decir que hace uso de etiquetas. Uno de los primeros modelos consistió en entrenar diversas arquitecturas exitosas en el campo de la visión por computadora, sobre melspectrogramas para predecir categorías de eventos acústicos [66]. Algunos de los modelos explorados fueron AlexNet, VGG, Inception y ResNET, los cuales fueron entrenados sobre el dataset YouTube-100M, que consiste de 5.24 millones de horas de video con etiquetas de video (30871 valores posibles). Al ser etiquetas de video, algunas serán acusticamente relevantes, como trompeta, mientras que otras, como "webpage", no lo serán. Finalmente, una vez entrenado el modelo, entrenan un MLP que toma como entrada las salidas de la penúltima capa para realizar detección de eventos acústicos (AED) en Audioset (521 categorías) [97]. Los autores muestran que utilizar los atributos aprendidos para la tarea de detección de eventos de video, es mejor que utilizar log-melspectrogramas para AED.

De manera similar, al ser Audioset un conjunto de datos relativamente masivo (2 millones de audios de 10 segundos), la tarea misma de AED puede utilizarse para preentrenamiento. Por ejemplo, en **PANNs** [98], los autores entrenan redes neuronales convolucionales para la tarea de AED, y luego evaluan cómo transfieren su conocimiento a otras tareas, ya sea utilizando el modelo como extractor de atributos o realizando finetuning. Los resultados mostrados son comparables al estado del arte en diversas tareas de audio como clasificación de eventos, clasificación de escenas acústicas, AED, clasificación de género musical y reconocimiento de emociones en el habla.

En **Audio Spectrogram Transformer (AST)** [86], como se ilustra en la Figura

2.11, se utiliza un Vision Transformer en lugar de CNNs, que toma melspectrogramas como entrada y devuelve probabilidades para las 521 clases de eventos en Audioset. Además de obtener un desempeño estado del arte en ese entonces para la tarea de AED, al realizar finetuning del modelo se obtienen resultados estado del arte también en clasificación de eventos en ESC-50 [99], y reconocimiento de comandos de voz (Google Speech Commands) [100]. Resulta interesante que AST no solo se beneficia del preentrenamiento en Audioset, sino que utilizar los pesos aprendidos originalmente por ViT en ImageNet resulta importante para un buen desempeño del modelo. Este detalle revela que T' se puede beneficiar de T incluso cuando los dominios y tareas no son similares. Al mismo tiempo, los resultados revelan que las más de 5000 horas de Audioset no saturan el desempeño en un ViT. En **Patchout faSt Spectrogram Transformer (PaSST)** [101] se modifica ligeramente AST para incrementar su eficiencia. Especificamente, durante el entrenamiento se descarta una proporción de los patches (técnica que denominan Patchout), lo cual permite acortar la secuencia de entrada del transformer reduciendo significativamente los tiempos de entrenamiento. Es importante destacar que el Patchout se realiza luego de agregar los embeddings posicionales, preservando la información de posición absoluta dentro de la secuencia para los patches no descartados. Este método logra superar a AST, tanto en términos de desempeño en tareas downstream, como en eficiencia computacional. **MAEST** [102] especializa PaSST para aplicaciones musicales, usando como datos de entrenamiento 3.3 millones de canciones y como tarea predecir tags asociados a estas canciones, logrando superar en diversas tareas de recuperación de información musical (MIR), a modelos especializados estado del arte.

Por último, en el campo de verificación del hablante, es común entrenar modelos para la tarea de identificación de hablante, y luego utilizar las activaciones de este modelo como atributos de entrada en un sistema de verificación del hablante. Algunos trabajos en esta línea son los **x-vectors** [103], **d-vectors** [104], **DNN/i-vectors** [105] y **DeepSpeaker** [106]. En este caso, T y T' son tareas cercanas en las que es esperable que los atributos aprendidos para T también sean útiles para T'.

2.3.2. Autoencoders de audio

Si bien utilizar supervisión para aprender representaciones puede ser muy útil, especialmente cuando T y T' son parecidas, se requiere disponer de un conjunto de datos etiquetados de gran escala, lo cual normalmente no está disponible. Diseñar métodos capaces de aprender representaciones de manera no supervisada puede desbloquear el acceso a conjuntos de datos mucho más masivos que los existentes con etiquetas.

Uno de los métodos más utilizados para aprender representaciones de forma no supervisada es el **autoencoder**. Este tipo de modelo consiste de un **encoder**, que transforma la señal de entrada x en una representación z, y un **decoder** que toma esta representación z y devuelve una reconstrucción $\hat{x}$ de x. Un aspecto importante de esta arquitectura es que z suele poseer una dimensionalidad mucho menor que x, resultando en un cuello de botella de información. De esta manera, z, también

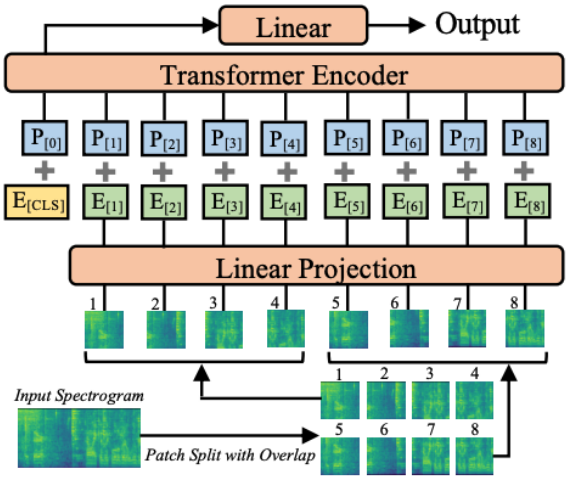

Figura 2.11: Arquitectura de Audio Spectrogram Transformers. Un espectrograma es dividido en patches rectangulares y se procesa por un ViT como si fuera una imágen. Finalmente se utiliza el token CLS en la salida para tareas de clasificación de audio. Extraído de [86]

.

denominado **espacio latente**, deberá capturar las características más relevantes de x de manera de que el decoder pueda realizar la reconstrucción de la forma más fiel posible.

Para promover una reconstrucción fiel, se suele utilizar como función de costo el error cuadrático medio (MSE), el error absoluto medio (MAE) o variantes de los mismos, calculados entre la entrada x y la salida $\hat{x}$ del autoencoder. No siempre es suficiente utilizar una pérdida de reconstrucción como MSE, por lo que se suele combinar con pérdidas adversariales. En [107] se define **distorsión** como el error entre la señal original y la estimada, y las pérdidas de reconstrucción como MSE y MAE buscan minimizarla. Por otro lado, se define **calidad perceptual** como cuán valida es la señal estimada, o cuán distinta es la distribución de la reconstrucción $\hat{x}$ respecto a la de x. Por ejemplo, un autoencoder puede ser muy bueno reconstruyendo x en términos de MSE o MAE, sin embargo, $\hat{x}$ podría contener artefactos que no se encuentran en señales de audio naturales, y que si bien no suman mucho error en términos de MSE, son fácilmente perceptibles y poco naturales. En el caso de utilizar MSE como función de costo, los autores demuestran que las estimaciones son un promedio de todas las reconstrucciones válidas dado z, y este promedio no es necesariamente una señal válida. En el caso de espectrogramas esto suele dar como resultado espectrogramas 'desenfocados' o suavizados, los cuales no se observan en audios naturales. Para resolver estos problemas y tener en cuenta la calidad perceptual en la reconstrucción, **EnCodec** [108] por ejemplo utiliza múltiples discriminadores adversarios que toman como entrada espectrogramas complejos con distintos tamaños de ventana y salto. Estos discriminadores deberán clasificar si el espectrograma de entrada es real o si en cambio es generado por el autoencoder. La función de costo se minimiza cuando el discriminador no es capaz de distinguir las

estimaciones del autoencoder de espectrogramas reales, y como resultado la calidad perceptual de los audios reconstruidos mejora con respecto a solo usar MSE.

Algunos trabajos, en lugar de usar autoencoders determinísticos usan **Autoencoders Variacionales (VAE)**, en los cuales z representa parámetros de una distribución de probabilidad, y el decoder reconstruye la señal de entrada dada una muestra de esa distribución. Este cambio junto con agregar un término de regularización en la función de costo, transforman el VAE en un modelo generativo. En el campo del audio se han propuesto muchos modelos de tipo autoencoder o VAE, utilizando distintas arquitecturas como redes recurrentes [109] que permiten trabajar con audios de longitud variable, redes neuronales convolucionales aplicadas sobre espectrogramas [110–112] y redes neuronales convolucionales tipo WaveNet aplicadas directamente sobre la forma de onda [113, 114].

Al mismo tiempo, se han explorado distintas técnicas de regularización para promover estructura en el espacio latente de estos modelos. Por ejemplo, una propiedad deseable es que si z es una secuencia, cambie lentamente (o sea suave). Esta propiedad promueve potencialmente capturar ciertas características como fonemas, los cuales pueden abarcar múltiples ventanas de análisis. En [113] se aplica jitter en el tiempo, el cual consiste de aleatoriamente hacer que z_{i+1} y/o z_{i-1} sean reemplazados por z_i, y en [115] se promueve group sparsity y penaliza la diferencia entre elementos sucesivos de z.

Otros trabajos intentan 'desenredar' (en inglés, disentangle) variables generativas del espacio latente. Por ejemplo, para señales de habla es deseable que el timbre y contenido estén codificados en dimensiones disjuntas. Esto permite, por ejemplo, controlar de manera independiente el contenido y hablante de la señal. Algunas técnicas utilizadas para lograr esto consisten en imponer estructuras temporales; por ejemplo el hablante suele ser constante a lo largo de una frase y puede ser representado con un vector z único, mientras que el contenido varía rapidamente en el tiempo y puede ser representado con una secuencia latente [116, 117].

En la práctica, las representaciones aprendidas por autoencoders que solo reconstruyen la entrada no suelen ser muy útiles para resolver tareas. Algunas técnicas para mejorar las representaciones de un autoencoder consisten en cambiar la función objetivo, y por ejemplo agregar ruido en la señal de entrada y entrenar al autoencoder para que no solo reconstruya la entrada sino que le remueva el ruido, dando lugar a los **Denoising AutoEncoders** [118]. Este tipo de transformaciones de la entrada fuerzan al modelo a capturar patrones más relevantes de la señal. Enmascarar partes de la señal suele ser una de las técnicas más utilizadas y exitosas para aprender representaciones. En la Sección 2.3.4 se explicará en profundidad esta técnica y distintas variantes e implementaciones de la misma.

Otra técnica comúnmente empleada en autoencoders de audio es el uso de cuantización vectorial en el espacio latente, dando lugar a la arquitectura **VQ-VAE** [119], la cual se ilustra en la Figura 2.12. La cuantización se realiza introduciendo un cuantizador vectorial (VQ), que tiene asociado un codebook $C \in \mathbb{R}^{K \times D}$ con K códigos de D dimensiones. Se busca la fila C_i que esté más cerca en distancia euclídea de la salida del encoder z, y con ella se obtiene la variable latente cuantizada z_q.

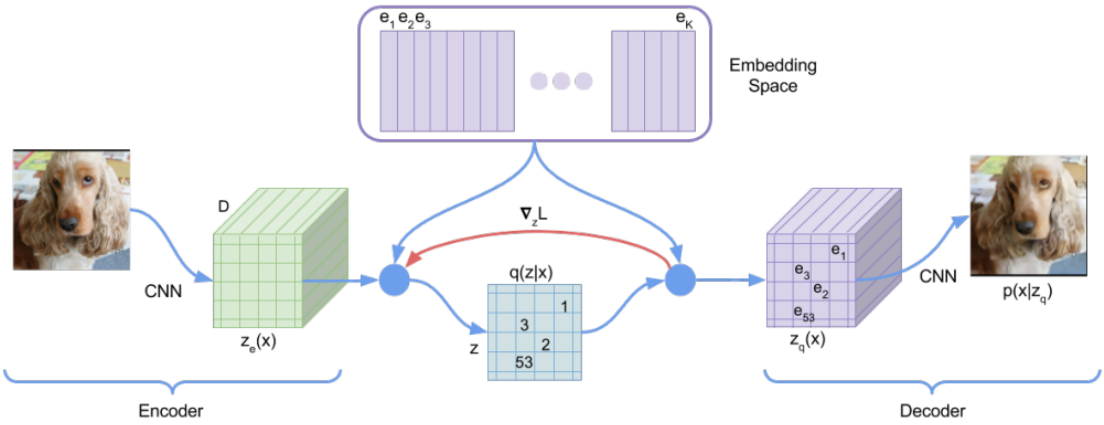

Figura 2.12: Esquema de una red VQ-VAE. La señal de entrada es procesada por un encoder generando z. Se reemplazan los elementos de z z_i por los códigos e_i más cercanos generando z_q, el cual es procesado por el decoder para reconstruir la señal de entrada. El gradiente fluye normalmente de adelante hacia atrás, con excepción de la capa de cuantización, la cual es salteada (curva roja) copiándose directamente el gradiente a la entrada del decoder a la salida del encoder. Extraído de [119]

.

De esta manera cada valor de z puede representarse con el índice correspondiente al código más cercano en el codebook, discretizando z. Dado que la operación de argmin no es diferenciable, se copia el gradiente en la entrada del decoder a la salida del encoder, "puenteando" esa operación. Luego, se introduce un término de costo adicional para actualizar solamente el codebook, que consiste en el error cuadrático entre la salida del encoder y el código seleccionado, y un término de commitment que actualiza al encoder para que sus salidas se acerquen al código más cercano. Otra opción más sencilla para actualizar cada código es utilizar un promedio móvil de las salidas del encoder asignadas al código, lo cual es análogo a K-Means. El uso de VQ-VAE para audio permite obtener representaciones muy comprimidas, y en el caso de habla promueve que se capture información fonética, ya que los fonemas también se pueden pensar como señales discretas en el tiempo.

Trabajos más recientes han extendido VQ-VAE para que se puedan aprender múltiples secuencias latentes discretas, ya que utilizar una sola puede ser muy restrictivo a la hora de obtener una buena reconstrucción en señales complejas como música. La técnica más utilizada es la de **Cuantizadores Vectoriales Residuales (RVQ)** [108, 120], en donde se toma z y se le aplica Q cuantizadores vectoriales, de forma que cada uno cuantiza el residuo del anterior, aproximando progresivamente a z. En consecuencia, x queda condensado en Q secuencias de enteros con K valores posibles, en donde K es el tamaño de cada codebook.

EnCodec [108] es un ejemplo de autoencoder de audio que utiliza RVQ, y que utilizaremos en esta tesis. Podemos analizar EnCodec en términos de compresión; la forma de onda original consiste de 24000 muestras por segundo con una profundidad

de 16 bits. Por ende, la tasa de información es $24000 \cdot 16 bits/s = 384 kbits/s$. Por otro lado, EnCodec utiliza $Q = 24$ cuantizadores como máximo, cada uno con $K = 1024$ códigos, y z tiene una longitud de 75 embeddings por segundo. Por lo tanto, la tasa de información es de $75 \cdot 24 \cdot \log_2(1024) = 18 kbits/s$, y constituye una compresión mayor a 21 veces de la tasa de información original. De esta manera, los autoencoders con RVQ son buenas alternativas a VQ-VAE si lo que se busca es discretizar x perdiendo la menor cantidad de información posible. Estas arquitecturas son excelentes compresores de audio obteniendo tasas de información menores a las de codecs tradicionales como MP3 y al mismo tiempo manteniendo una buena calidad de audio, por lo que en la literatura se las denomina **Neural Audio Codecs**.

Recientemente, debido al auge de los LLM y el deseo de darles capacidades de escucha y habla, el desarrollo de este tipo de representaciones se ha vuelto popular, ya que permiten representar sonido como secuencias discretas, lo cual es compatible con las secuencias de tokens utilizadas como entrada y salida en los LLM de texto. De esta forma, el encoder junto a las capas de cuantización funcionan como tokenizadores de audio, mientras que el decoder funciona como sintetizador de audio. Algunas propiedades deseables de un tokenizador de audio en el contexto de LLMs son:

- **Secuencia única de tokens**: si se tienen múltiples secuencias como en un RVQ, se les debe dar un formato en el que se respete el ordenamiento temporal y de cuantizadores, y que pueda ser consumido y generado por un LLM. Algunas técnicas consisten en intercalar los distintos cuantizadores [121], pero esto da lugar a secuencias más largas. Otras utilizan un patrón de retardo [122] o generan los cuantizadores de mayor órden a partir del primero [123]. En todos los casos, se agrega complejidad al modelo y se incrementa el largo de secuencia, por lo que lograr tokenizar audio con una única secuencia resulta deseable. Por ejemplo, **WavTokenizer** [124] y **UniCodec** [125] son capaces de tokenizar audio con una única secuencia de 40 o 75 tokens por segundo, mediante un incremento del tamaño de codebook (por ejemplo de 1024 en EnCodec a 16384 en UniCodec), y realizando mejoras en las técnicas de cuantización y arquitectura del modelo. Otros trabajos, en vez de aprender codebooks, utilizan **Finite Scalar Quantization (FSQ)** para cuantizar, pudiendo incrementar el tamaño de vocabulario y su utilización [126].
- **Secuencias cortas**: tener secuencias más cortas es deseable dado que los transformers tienen un costo computacional que es cuadrático con la longitud. A su vez, si cada token representa segmentos más largos de audio, probablemente se alinee mejor con los tokens de texto que espera un LLM. A modo de ejemplo, un hablante de inglés suele producir en promedio unas 2 o 3 palabras por segundo, mientras que una representación como EnCodec contiene 75 tokens por segundo. Tokenizadores más recientes como **Mimi** [127] son capaces de reducir el largo de secuencia a 12.5 tokens por segundo mediante la incorporación de Transformers. Otros trabajos como **SNAC** [128] utilizan una variante de RVQ en la que los primeros cuantizadores utilizan menos tokens por segundo mientras que los últimos tienen mayor resolución temporal.

- **Tokens semánticos**: es deseable que un tokenizador de audio no solo conserve la mayor cantidad de información posible del audio original, sino que también facilite el aprendizaje del LLM. Para esto, se intenta incorporar información semántica en los tokenizadores, que no solo brinden información local necesaria para la reconstrucción del audio original, sino que también posean información más contextual referente al significado de los sonidos. En algunos trabajos [127, 129] se incorpora semántica mediante un término en la función de costo que intente acercar el primer cuantizador a representaciones de algún modelo preentrenado como WavLM [130]. De forma similar, UniCodec [125] incorpora semántica mediante una etapa de modelado de lenguaje enmascarado (MLM), prescindiendo del uso de modelos preentrenados.

2.3.3. Aprendizaje por contraste

El aprendizaje por contraste consiste en aprender representaciones de una señal $f(x)$ de manera que señales similares se encuentren cerca en este espacio de representaciones, mientras que señales disimilares estén lejos. La definición de cuándo dos señales son similares entre si depende del problema a resolver. Por ejemplo, si se trabaja en la tarea de verificación de hablante, definimos que dos señales son similares si el hablante es el mismo, mientras que si la tarea es reconocimiento de comandos de voz, definimos similaridad en base al contenido fonético y no al hablante.

Pérdida de triplas

Así como podemos definir qué características queremos que varíen en la representación, también podemos definir aquellas que queremos que no afecten a la representación, o se mantengan invariantes. En el caso de reconocimiento de comandos de voz, desearíamos que el hablante no altere a la representación, así, sin importar quién diga el comando, siempre será reconocido. Del mismo modo, queremos que sea invariante al ruido y a las condiciones acústicas, así un clasificador entrenado con estos atributos es robusto a cambios en el ruido de fondo o micrófonos utilizados. Del mismo modo, en un lenguaje no tonal, podríamos querer que sean invariantes a la frecuencia fundamental, ya que el contenido del habla no cambiará al cambiar esta. Matemáticamente, nuestro modelo f será invariante ante ciertas transformaciones de la señal de entrada $T(x)$ si $f(x) \approx f(T(x))$.

Esto se puede lograr de manera explícita utilizando técnicas de aprendizaje por contraste. Por ejemplo, **TRILL** [131] forma triplas (x_i, x_j, x_k), donde x_i es un segmento del audio x comenzando en el instante de tiempo i. La tripla se forma de manera que $|i - j| \leq \tau$ y $|i - k| > \tau$, donde τ es un hiperparámetro que define a partir de qué distancia temporal queremos que la representación cambie. Lo que se busca durante el aprendizaje es que la representación de un segmento x_i sea más similar a la de un segmento que está temporalmente cerca que a la de un segmento que está a una distancia temporal mayor a τ. Esto se expresa matematicamente mediante la **funcion de costo de triplas**:

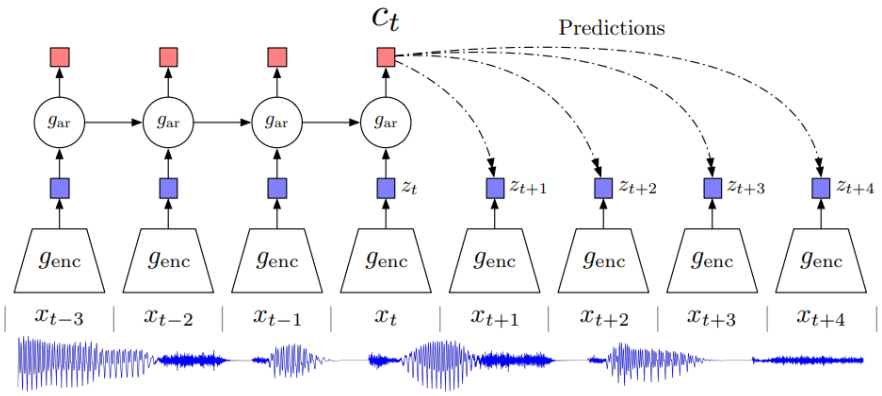

Figura 2.13: Diagrama de la técnica de Contrastive Predictive Coding. Una red g_ar obtiene una representación de la secuencia hasta el instante t, la cual intentará acercar a representaciones de la señal de audio z_{t+k} a distancia k en el tiempo y alejarlo de otras representaciones $z_{i\neq t+k}$. Extraído de [133].

$$L(x_i,x_j,x_k) = \max(0, \|f(x_i) - f(x_j)\|_2^2 - \|f(x_i) - f(x_k)\|_2^2 + \sigma),$$

en donde σ es el margen y es un hiperparámetro que define cuánto más lejos de x_i debe estar x_k respecto a x_j para que la función de pérdida sea cero.

En terminologías de aprendizaje por contraste, x_i se denomina ancla, x_j es un ejemplo positivo y x_k es un ejemplo negativo. Distintas definiciones de qué es un ejemplo positivo y negativo, llevan a distintas invarianzas en las representaciones resultantes. En [132] los autores experimentan con distintas formas de obtener las triplas: usando como ancla y ejemplo negativo audios tomados al azar, y como ejemplo positivo al ancla con ruido gaussiano; al ancla con un desplazamiento en tiempo o frecuencia; a un segmento temporalmente cercano al ancla (mismo criterio que TRILL); o una suma pesada del ancla y el ejemplo negativo. Los autores muestran que la tripla que lleva a representaciones más útiles para clasificación y recuperación de sonidos es la de proximidad temporal (misma que TRILL), y combinar todas las estrategias mejora los resultados.

Codificación predictiva por contraste (CPC)

La **codificación predictiva por contraste (CPC)** [133] ha sido utilizada extensivamente para el aprendizaje de representaciones de audio. Los modelos de CPC consisten de una función no lineal g_{enc} que transforma la entrada x_t en una secuencia de representaciones ocultas $z_t = g_{enc}(x_t)$, y un modelo autoregresivo g_{ar} que devuelve una representación contextual $c_t = g_{ar}(z_{\leq t})$ dados todos los z_t pasados. El objetivo del modelo es maximizar la información mutua entre c_t y x_{t+k}, es decir c_t tiene que contener información del futuro de la señal. La motivación detrás del método es que las variables de interés en el habla, como fonemas y entonación, cambian lentamente por lo que abarcarán múltiples instantes t de tiempo. Idealmente, g_{enc} aprenderá a descartar información local y ruido conservando solamente estos atributos

'lentos'. Para entrenar el modelo se minimiza la función de costo **InfoNCE**,

$$L_N = -\mathbb{E}_Z[\log \frac{f_k(z_{t+k},c_t)}{\sum_{z_j \in Z} f_k(z_j,c_t)}]$$

en donde $Z = \{z_1, ..., z_N\}$ es el conjunto de N variables aleatorias conteniendo una muestra positiva de $p(z_{t+k}|c_t)$ y $N-1$ muestras negativas de $p(z_{t+k})$, k es la cantidad de cuadros a futuro y $f_k(z_{t+k},c_t) = e^{z_{t+k}^T W_k c_t}$. Se puede probar que minimizar InfoNCE es equivalente a maximizar una cota inferior de la información mutua. En términos de aprendizaje por contraste, optimizar InfoNCE hará que c_t, el ancla, se acerque a z_{t+k}, y por ende a x_{t+k}, el ejemplo positivo, mientras que se aleje de los ejemplos negativos x_j. Otra manera de interpretar InfoNCE es como la entropía cruzada de clasificar a la muestra correcta z_{t+k}, siendo $\frac{f_k(z,c_t)}{\sum_z f_k(z,c_t)}$ la predicción del modelo. Se proponen diversas maneras de muestrear los negativos de $p(z_{t+k})$, como utilizar representaciones correspondientes a otros audios del batch, que pueden tener al mismo hablante o a uno distinto, o utilizar representaciones provenientes de otro segmento del mismo audio. En [133], los autores logran un desempeño superior a MFCCs utilizando c_t como atributos y entrenando clasificadores lineales, y un desempeño similar a modelos entrenados de forma supervisada en las tareas de clasificación de fonemas e identificación de hablantes. **Wav2Vec** [134] toma esta idea y la aplica al problema de reconocimiento de habla, logrando resultados estado del arte en ese entonces. En ambos trabajos, g_{enc} es una CNN, mientras que g_{ar} es una red recurrente en [133] y una CNN en Wav2Vec.

Otro enfoque es el de **codificación predictiva autoregresiva (APC)** [135–137], el cual tiene un objetivo similar a CPC; predecir a partir de una representación del pasado los cuadros futuros del espectrograma del audio de entrada. La principal diferencia es que en lugar de usar InfoNCE, los autores realizan regresión utilizando la norma L_1 entre las predicciones del modelo y el log-melspectrograma del audio, evitando el uso de pérdidas contrastivas y ejemplos negativos. Uno de los argumentos en contra de CPC es que las representaciones solo necesitan contener información que distinga a los ejemplos positivos de los negativos, mientras que el uso de regresión incentiva a codificar más información, lo cual puede ayudar si se quiere utilizar la representación en muchas tareas distintas.

Uso de redes maestra-estudiante

El aprendizaje por contraste requiere muestrear ejemplos negativos, lo cual puede hacer trivial al aprendizaje si estos ejemplos negativos son muy distintos al ancla, y por ende "fáciles". Muestrear ejemplos negativos difíciles (hard negative mining), es un problema en sí mismo [139]. Por esta razón, prescindir de los ejemplos negativos y solo contrastar el ancla con los positivos es algo deseable. **Bootstrap your own latent (BYOL)** [140] logra evitar el uso de ejemplos negativos y ha sido aplicado al ámbito del aprendizaje de representaciones de audio en **BYOL for audio (BYOL-A)** [138]. Este modelo se ilustra en la Figura 2.14 y consiste en transformar de dos formas distintas un mismo segmento de audio x, generando 2 vistas v y v'. Estas dos

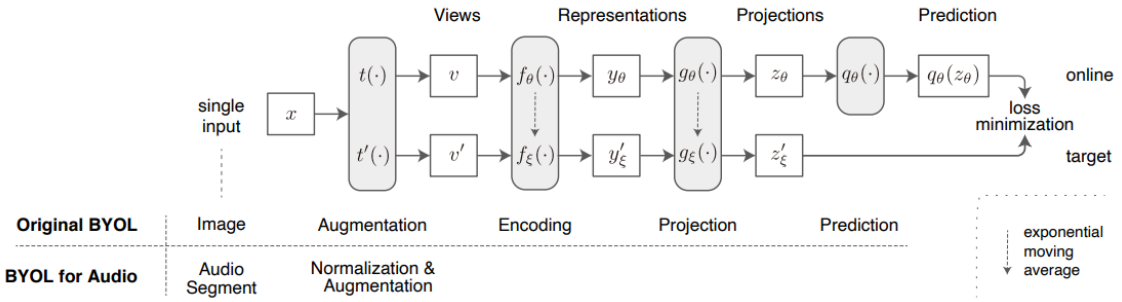

Figura 2.14: Arquitectura de BYOL-A. Un segmento de audio es transformado de dos maneras distintas y cada vista resultante es procesada por la red maestro y la estudiante. El objetivo del preentrenamiento es hacer que las salidas de ambas redes coincidan. Extraído de [138]

.

vistas serán procesadas por 2 redes neuronales, una maestra (f_θ) y otra estudiante (f_ξ), generando z_θ y z'_ξ respectivamente. El modelo se entrenará de manera que la red estudiante sea capaz de predecir z'_ξ a partir de z_θ. Una particularidad del modelo es que la red maestra y estudiante comparten la misma arquitectura, salvo por la capa extra de predicción q_θ del estudiante, y que los pesos ξ de la red maestra son un promedio móvil exponencial de los pesos θ de la red estudiante. Se puede pensar que la dupla (v,v') son el ancla y el ejemplo positivo, dado que durante el aprendizaje se intenta minimizar la distancia entre las representaciones de ambas vistas. En BYOL-A el segmento de audio es representado como un log-melspectrograma, y las transformaciones aplicadas consisten en muestrear aleatoriamente un audio n, que funcionará como ruido, y sumarlo a x cuidando que n tenga menor energía que x; y recortar un rectángulo aleatorio del melspectrograma y mapearlo al tamaño original mediante un upsampling. La primera transformación introduce ruido, mientras que la segunda simula de forma simplificada cambios en la frecuencia fundamental y velocidad del audio.

Data2Vec [141] utiliza un método similar, pero la vista v es la señal de audio x con segmentos enmascarados, mientras que v' es el audio original sin enmascarar. Otra diferencia es que se utiliza el audio crudo como representación de x, y la arquitectura es un Transformer que predice el promedio de las activaciones de las últimas K capas de la red maestra para los cuadros no enmascarados. **Data2Vec 2.0** [142] vuelve más eficiente al modelo al reemplazar la red por un **Masked Autoencoder (MAE)** [3]. Originalmente en Data2Vec, al enmascarar audio se utilizaban tokens de máscara como entrada, por lo que el largo de secuencia de entrada a la red es el mismo que sin enmascarar. Los MAE en cambio, eliminan los segmentos enmascarados de la secuencia de entrada reduciendo el largo de las mismas de manera similar a Patchout. Una vez procesada la secuencia de elementos visibles por el encoder, se agregan los tokens de máscara en las posiciones correspondientes, y se utiliza un decoder de pocas capas para predecir los segmentos enmascarados a partir de los visibles, ya procesados por el encoder. **Efficient Audio Transformer** (EAT) [143], es similar a Data2Vec

2.0, pero utiliza un MAE que toma patches de melspectrograma como entrada en lugar de audio crudo. Al mismo tiempo, utiliza una estrategia de enmascarado más agresiva que le permite acelerar el entrenamiento, y utiliza una función de costo no solo a nivel patch, sino también global. **DINO** [144] comparte muchas características con BYOL, siendo sus principales diferencias no utilizar un predictor; es decir la red maestra y estudiante son idénticas en términos de arquitectura, y en lugar de usar MSE como función de costo, utiliza entropía cruzada, centrando previamente las salidas de la red maestra y aplicándoles softmax con temperatura. **DinoSR** [145] aplica Dino a señales de habla, utilizando enmascaramiento como aumentación (al igual que Data2Vec), y realizando un clustering online de las salidas de la red maestra en lugar de un paso de centrado y softmax con temperatura.

En varios trabajos descriptos en esta sección se concluye que la transformación $T(x)$ consistente en enmascarar segmentos de un audio x es exitosa en el campo del aprendizaje por contraste, y ha permitido mejorar el aprendizaje no supervisado de representaciones de audio. En la sección siguiente se profundizará en esta técnica y se presentarán otros modelos que hacen uso de la misma.

2.3.4. Modelos de Lenguaje Enmascarado

Con la introducción de **BERT** [146] en el campo del procesamiento del lenguaje natural (NLP), pronto la comunidad de procesamiento de habla y audio comenzó a proponer variantes que pudieran utilizarse con señales de audio. BERT es un modelo de aprendizaje auto-supervisado, en el que se enmascara una cierta proporción de las palabras de un texto, y un transformer bidireccional (sólo la parte del encoder de la Figura 2.8) debe aprender a predecir la palabra que fue enmascarada. Esta tarea de pretexto, llamada modelado de lenguaje enmascarado (MLM) o prueba de Cloze, incentiva al modelo a aprender representaciones contextualizadas, que aprovechan la totalidad del texto para determinar qué palabras tienen mayor probabilidad en la posición enmascarada. Usando esta tarea de pretexto, y luego realizando un finetuning del modelo, los autores logran mejorar el estado del arte en 11 tareas de NLP marcando un hito en el aprendizaje no supervisado de representaciones.

Dado que el texto es discreto, mientras que el audio es continuo, adaptar BERT al dominio de los sonidos no es trivial. Una opción es transformar la señal de audio en una secuencia discreta para luego poder operar sobre ella como si fuera texto. **VQ-Wav2Vec** [147] genera representaciones discretas del habla agregando un cuantizador Gumbel-Softmax [148] a Wav2Vec. Dado que las representaciones discretas son calculadas cada 30 ms, cuadros continuos contendrán información similar volviendo trivial la tarea original de enmascarado de BERT. Para adaptar el enmascarado a señales de audio, se muestrea aleatoriamente comienzos de máscara, y luego se enmascaran 10 cuadros consecutivos en vez de uno solo. **DiscreteBERT** [149] extiende VQ-Wav2Vec explorando otras formas de discretizar la señal de audio: k-means de MFCCs y de log-melspectrogramas, llegando a la conclusión de que VQ-Wav2Vec genera mejores representaciones para ASR.

Otro enfoque posible es discretizar solamente la señal objetivo, dado que de todas

formas BERT transforma el texto de entrada en una señal continua mediante el uso de una tabla de Embeddings (o Lookup Table). Por ende, resulta trivial trabajar directamente sobre el audio continuo, y en cambio utilizar una representación discreta del mismo como objetivo. En **HuBERT** [76], se utiliza una CNN que transforma el audio crudo en una secuencia de representaciones z_t (50 por segundo), y luego se enmascaran estas representaciones como en VQ-Wav2Vec y sirven de entrada a un transformer similar al usado en BERT. La señal objetivo se obtiene calculando MFCCs de la señal de entrada y cuantizándolos mediante K-Means, obteniendo pseudo-fonemas. Luego de entrenado el modelo, mejoran la señal objetivo realizando K-Means nuevamente, pero esta vez sobre las activaciones del modelo ya entrenado, las cuales deberían ser señales más contextuales y con mejor discriminación fonética que los MFCCs. Los autores denominan a este procedimiento como **refinamiento iterativo de asignación de clusters**, y lo realizan dos veces. Con este método los autores logran mejorar el estado del arte en la tarea de reconocimiento del habla. **WavLM** [130] extiende HuBERT simulando habla con ruido y múltiples hablantes solapados. De esta manera, el modelo debe aprender a estimar las pseudoetiquetas del audio original, a partir de una versión ruidosa y enmascarada del mismo, añadiendo robustez al modelo que implícitamente deberá aprender a separar fuentes y reducir el ruido de la señal. Con este cambio y el uso de gated relative position bias [150], los autores logran el mejor desempeño en la mayoría de las tareas de Superb.

Best-RQ [151] explora proyectar la señal de entrada con una matriz aleatoria, y utilizar un codebook aleatorio para asignar un índice. De esta manera, se obtiene una señal objetivo discreta que es una proyección y cuantización aleatoria de la entrada. A pesar de la simplicidad del modelo, y el uso de proyecciones aleatorias, el modelo obtiene resultados similares a HuBERT. Un detalle no menor es que Best-RQ utiliza codebooks con tamaño 8192, en lugar de los 50,100 o 500 centroides utilizados en HuBERT. De manera similar, **BEATs** [152] utiliza la estrategia de Best-RQ utilizando un cuantizador aleatorio para generar la señal objetivo, pero luego mejora iterativamente este cuantizador de forma parecida a HuBERT, utilizando activaciones intermedias del modelo y entrenando un tokenizador que aprenda a cuantizarlas.

Por último, se puede prescindir de la discretización del audio por completo y realizar regresión de los segmentos enmascarados. **DeCOAR** [153] y **DeCOAR 2.0** [154] reconstruyen un segmento único enmascarado de espectrograma a partir de las porciones de espectrograma anteriores y posteriores al segmento enmascarado. En [155–158] se enmascaran espectrogramas con un patrón similar a VQ-Wav2Vec y se entrena un modelo, normalmente RNN o Transformer, para reconstruir los segmentos enmascarados, utilizando el error cuadrático o absoluto medio como función de costo. En el caso de [155] el enmascaramiento es tanto en el tiempo como por bandas de frecuencia. En **TERA** [159], se exploran otras formas de alterar la señal de entrada además de enmascarándola en el tiempo, como reemplazar segmentos de audio por segmentos muestreados al azar, enmascarar bandas de frecuencias y sumar ruido en la entrada.

Otros trabajos como **SSAST** [161] utilizan una arquitectura similar a AST, y enmascaran patches rectangulares de espectrogramas, para luego reconstruirlos.

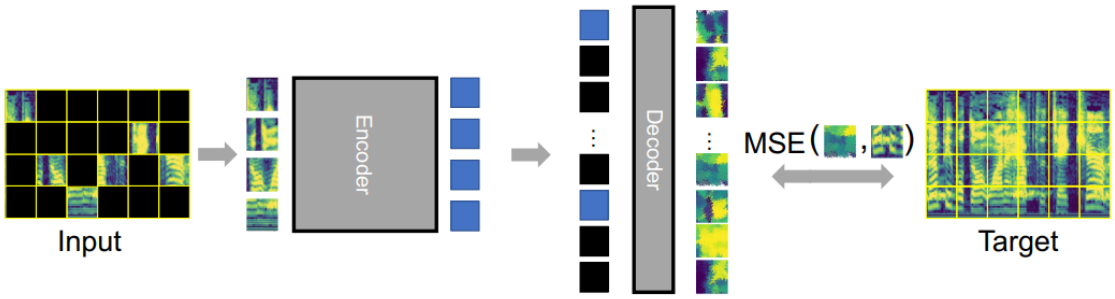

Figura 2.15: Aprendizaje de representaciones mediante MAE. El espectrograma de entrada es dividido en patches y una proporción de estos son enmascarados. El encoder procesa solamente los patches no enmascarados y genera una representación contextualizada de ellos. Luego, tokens de máscara son insertados en las posiciones donde habían patches enmascarados y el decoder intenta reconstruír los patches originales del espectrograma. Extraído de [160].

De forma similar, en **AudioMAE** [160], **MSM-MAE** [162], **MAE-AST** [163] y **MaskSpec** [164], se reemplaza el transformer de SSAST por un MAE, el cual tiene la ventaja de ser más eficiente, dando lugar a arquitecturas como la ilustrada en la Figura 2.15. Recientemente se ha explorado escalar el tamaño de este tipo de modelos y de la cantidad de datos de preentrenamiento dando lugar a **Dasheng** [165], el cual posee 3 tamaños: base con 12 capas y 83M de parámetros, 0.6B con 32 capas y 600M de parámetros, y 1.2B con 40 capas y 1200M de parámetros. En cuanto a los datos de preentrenamiento, se utilizaron 272k horas de audio consistentes de Audioset, VGGSound, Jamendo y ACAV100M. A diferencia de otros modelos similares, en lugar de utilizar patches rectangulares de melspectrogramas, cada entrada al MAE consiste de 4 cuadros consecutivos de melspectrograma concatenados.

La tarea de MLM permite a los modelos incorporar información que se encuentra antes y después de la máscara, por lo que es beneficioso que sean bidireccionales. Otros trabajos combinan la tarea de MLM con CPC [75, 166, 167], dando lugar a la tarea de **Masked Predictive Coding (MPC)**. El trabajo más relevante es **wav2vec 2.0** [75], el cual de manera análoga a HuBERT transforma el audio crudo en una secuencia z_t mediante una CNN, luego la enmascara y un transformer similar a BERT debe inferir los segmentos enmascarados. Las principales diferencias con HuBERT son que la señal objetivo q_t es aprendida en conjunto con el resto del modelo, cuantizando z_t mediante Gumbel-Softmax, y que la función de costo es InfoNCE, en donde el ejemplo positivo es el vector q_t correspondiente al instante de tiempo enmascarado, y los ejemplos negativos son otros q_t correspondientes a otros instantes de tiempo enmascarados de la misma señal de habla.

2.3.5. Aprendizaje por supervisión de modalidad cruzada

Asociar audio con otras modalidades es un aspecto fundamental de la percepción humana. A medida que los bebés se desarrollan, la sincronización y correspondencia de sonidos permite la asociación multimodal: una voz con un rostro, y un "muu" con

una vaca [168]. Más adelante, al adquirir el lenguaje, se asocian palabras habladas con los objetos que estas representan [169, 170]. Sorprendentemente, estas habilidades de asociación, que constituyen el reconocimiento del habla, el reconocimiento de eventos sonoros y el reconocimiento de objetos visuales, se desarrollan sin necesidad de mucha supervisión directa. Resulta entonces natural intentar aprovechar la inmensa cantidad de datos audiovisuales que existen para aprender representaciones mediante la asociación de imagen y sonido.

En **Soundnet** [171], se utilizan clasificadores de imágenes preentrenados como fuente de supervisión para aprender representaciones. El clasificador de imágenes recibe el cuadro correspondiente como entrada y predice a qué clase pertenece la imágen. Al mismo tiempo, una CNN toma como entrada el audio correspondiente a ese cuadro, y es entrenada para predecir las salidas del clasificador de imágenes. Se busca minimizar la divergencia KL entre la salida del clasificador de imágenes y la salida de la CNN. De esta forma, el clasificador de imágenes funciona como pseudo-etiquetador de los audios correspondientes. Este enfoque se puede ver como una forma de supervisión débil.

En **L3Net** [172] se prescinde de tener un clasificador de imagenes preentrenado, y en cambio, se entrenan desde cero 2 encoders, uno para la imagen y otro para el audio. Luego, las salidas de los encoders se concatenan y sirven de entrada a un MLP que debe predecir si la imagen y el audio provienen del mismo segmento de video o no. De esta manera, se utiliza la tarea de **predicción de correspondencia audio-visual** como T, o tarea de pretexto. Luego, las representaciones del encoder de audio son utilizadas para resolver tareas de detección y clasificación de eventos acústicos. En una siguiente iteración [173], los autores analizan el efecto de las decisiones de diseño, llegando a algunas conclusiones interesantes como que representar el audio de entrada usando melspectrogramas en lugar de espectrogramas mejora el desempeño en las tareas downstream, que preentrenar con datos de música (fuera de dominio) y luego evaluar en tareas de eventos acústicos da un desempeño similar a utilizar datos de sonido ambiente (en dominio), y que L3Net posee un mejor desempeño que SoundNet y VGGish [66] en tareas de clasificación y detección de eventos acústicos, a pesar de ser 10 veces menor a VGGish en términos de parámetros, y ser entrenado con un conjunto de datos 100 veces menor.

DenseAV [174] también es entrenado en la tarea de correspondencia audiovisual, optimizando InfoNCE, la cual buscará acercar las señales con correspondencia audio-visual, y alejar las que no se corresponden. La similaridad entre modalidades es calculada entre cada cuadro de HuBERT y cada elemento del mapa de activación de la imágen (proveniente de DiNO [175]), la cual luego es agregada en un valor único. En **CAVMAE** [176], aprenden un MAE que reconstruye una concatenación de patches de imágen y patches de espectrograma que es enmascarada, y al mismo tiempo promueve que los patches pooleados de la imágen y su sonido correspondiente sean similares entre si, combinando MLM con aprendizaje contrastivo multimodal. En [177–179] se aprenden o enriquecen representaciones de habla al promover que las representaciones de las descripciones habladas de una imágen sean similares a la representación de la imágen. Este grounding visual permite obtener representaciones

de habla con mayor contenido semántico.

Por otro lado, se puede utilizar texto en lugar de imágenes para enriquecer las representaciones de audio. Por ejemplo, en **CLAP** [180], se busca alinear las representaciones de un encoder de audio con las de un encoder de texto. De manera análoga a CLIP [181] para imágenes, se posee una base de datos multimodal, conteniendo audios y sus descripciones textuales, y se calcula la similaridad entre cada audio y cada descripción. La función de costo busca maximizar la similaridad entre los audios y sus descripciones, y minimizar la similaridad de cada audio con las descripciones que no le corresponden. De esta manera, el modelo de audio aprende representaciones que se alinean con los conceptos presentes en la descripción textual, logrando un buen desempeño en tareas de clasificación de audio.

Una de las limitaciones de este enfoque es que son necesarias descripciones o transcripciones y alineamientos de los audios, lo cual es costoso de obtener. En **SpeechClip** [182], los autores finetunean HuBERT y su salida es utilizada como entrada al encoder de texto de CLIP. Luego, se utiliza la función de pérdida de CLIP entre la salida del encoder de texto de CLIP y la salida del encoder de imagen de CLIP. Los autores también experimentan utilizando la función de pérdida directamente entre la salida de HuBERT y la de CLIP para imágen, ya que implícitamente su representación es equivalente a la de la descripción de la imagen, haciendo de puente con la modalidad textual. Similarmente en **ImageBind** [183] y **AudioCLIP** [184] se explotan las tres modalidades al mismo tiempo (texto, imágen y audio), buscando que se alineen en un espacio de representación común. Sin embargo, cabe notar que estos métodos requieren de un conjunto de datos con pares de audio y texto, los cuales suelen ser costosos de producir. La tarea de correspondencia audiovisual, o utilizar modelos preentrenados de imágenes para producir etiquetas débiles, son métodos mejor alineados con el aprendizaje no supervisado de representaciones de audio, y permiten la utilización de cantidades masivas de datos sin etiquetar.

2.3.6. Tabla resumen

Modelo	Entrada	Manipulación	Arquitectura	Objetivo	Dataset
VGG-ish [66]	Mel	-	CNN (VGG)	Tagging	YT-8M
PANNs [98]	Mel	-	CNN	AED	AS
AST [161]	Mel	-	ViT	AED	AS
PaSST [101]	Mel	Patchout	ViT	AED	AS
TRILL [131]	Mel	-	ResNet-50	Triplas	AS (Habla)
CPC [133]	Raw	-	CNN + GRU	CPC	LS - 100h
Wav2Vec [134]	Raw	-	CNN	CPC	LS
BYOL-A [138]	Mel	Mixup + RRC	CNN	Maestro	AS
Data2Vec [141]	Raw	MLM	Transformer	Maestro	LS
Data2Vec 2.0 [142]	Raw	MLM	Transformer MAE	Maestro	LS
EAT [143]	Mel	MLM	ViT + MAE	Maestro	AS
DinoSR [145]	Raw	MLM	Transformer	Clusters Maestro	LS
DiscreteBERT [149]	VQ-W2V	MLM	Transformer	VQ-Wav2Vec	LS
HuBERT [76]	Raw	MLM	Transformer	KM MFCC + Iter	LS
WavLM [130]	Raw	MLM + ruido	Transformer	KM MFCC + Iter	LS
BestRQ [151]	Mel	MLM	Transformer	RQ	LS
BEATs [152]	Mel	MLM	ViT + MAE	RQ + Iter	AS
DeCOAR [153]	Mel	Única máscara	BiLSTM	Mel	LS
DeCOAR 2 [154]	Mel	Única máscara	Transformer	Mel	LS
SSAST [161]	Mel	MLM	ViT	Mel	AS
AudioMAE [160]	Mel	MLM	ViT + MAE	Mel	AS
Wav2Vec 2.0 [75]	Raw	MLM	Transformer	CPC	LS
Dasheng [165]	Mel	MLM	Transformer + MAE	Mel	Mezcla

Tabla 2.1: Tabla resumen de los distintos métodos de aprendizaje de representaciones de audio discutidos. El primer bloque corresponde a modelos entrenados de forma supervisada, el segundo a modelos entrenados mediante aprendizaje por contraste, y el tercero a modelos entrenados mediante modelado de lenguaje enmascarado (MLM). RQ se refiere a cuantizador aleatorio, Iter a refinamiento iterativo, KM a K-Means, AED a detección de eventos acústicos, AS a Audioset, LS a LibriSpeech y mezcla al dataset de Dasheng compuesto por ACAV100M, AudioSet, VGGSound y Jamendo.

A modo de resumen, en la Tabla 2.1 se condensan algunos de los modelos mencionados en esta sección, comparando los aspectos que consideramos más relevantes y diferenciadores de los distintos métodos: la forma en que se representa el audio como entrada del modelo; la transformación aplicada a dicha entrada; la arquitectura del modelo; la señal objetivo con la que se aprenden las representaciones; y los datasets utilizados para su entrenamiento. El primer bloque corresponde a modelos entrenados con supervisión, por lo que las señales objetivos son anotaciones: tags o etiquetas de eventos acústicos. En el segundo bloque se presentan modelos entrenados mediante objetivos contrastivos, ya sea mediante pérdida de triplas, CPC o redes maestro-estudiante. Se puede observar que entre modelos con un mismo objetivo de preentrenamiento, las diferencias principales radican en las arquitecturas utilizadas y el dominio de aplicación, el cual queda determinado por el conjunto de preentrenamiento: LibriSpeech para habla y Audioset para audio general o eventos acústicos. El último bloque corresponde a modelos entrenados mediante enmascaramiento de las entradas, que deben de alguna manera reconstruir la información faltante. Entre estos modelos, las principales diferencias radican en cómo se representa la entrada y

la señal objetivo, con algunos modelos discretizando la señal objetivo y formulando el problema como una clasificación, y otros modelos prediciendo la misma entrada sin enmascarar, y por ende realizando regresión. Se puede observar que en este bloque la mayoría de los modelos utilizan una arquitectura Transformer, heredando metodologías de trabajos del campo del NLP como BERT. Se puede ver que en general, los modelos que trabajan con Audioset utilizan melspectrogramas como representación de entrada, mientras que los modelos de habla en algunos casos procesan directamente el audio crudo. El patrón más común de diseño al realizar modelos generales de audio es preentrenar con Audioset, utilizando melspectrogramas como representación de entrada, y Vision Transformers como arquitectura, enmascarando patches de la entrada.

Algunos modelos son dificiles de encasillar en un bloque. Por ejemplo, Data2Vec y EAT colocados en el segundo bloque también enmascaran sus entradas, pero al utilizar redes maestro-estudiante decidimos ubicarlos en el bloque de modelos contrastivos. Del mismo modo, Wav2Vec 2.0 podría pertenecer al segundo bloque, ya que utiliza InfoNCE como función de costo.

3

EnCodecMAE

Como se comentó en la sección 2.3, la mayor parte de los modelos de aprendizaje de representaciones están pensados específicamente para habla, y el foco en general está en mejorar la tarea de reconocimiento automático del habla. Por otro lado, los modelos enfocados en aprender representaciones más generales de audio, usualmente están basados en la arquitectura AST y utilizan patches de espectrogramas como señal de entrada, pensando a la misma como imágenes. En esta sección proponemos un modelo del cual deseamos las siguientes características:

- Que sus representaciones sean útiles para tareas en el dominio del habla, música y sonido ambiental, resultando en una representación de audio para uso general.
- Que el modelo pueda entrenarse con pocos recursos computacionales, permitiendo a universidades y organizaciones con menores recursos entrenar variantes del mismo, y en consecuencia contribuyendo positivamente a la democratización de la IA y reduciendo el impacto ambiental.

Para incentivar la primera característica, utilizamos una mezcla de datos provenientes del dominio de habla, música y audio general. A su vez, utilizamos como señal objetivo la salida de codecs de audio neuronales que, a diferencia de otras señales objetivos como los clusters de MFCCs usados en HuBERT, creemos que son más genéricas ya que al surgir de un objetivo de reconstrucción del audio original, no se alinean tan fuertemente con alguna característica particular de la señal como los fonemas. Para mejorar la eficiencia del modelo, utilizamos la arquitectura de Masked Autoencoder y audios de 4 segundos de duración, logrando entrenar los modelos de tamaño Base con una sola GPU en 5 días (120 horas GPU), y los Large con dos GPUs en la misma cantidad de tiempo (240 horas GPU). Esto contrasta con otros trabajos como por ejemplo HuBERT, que utilizan 32 GPUs para entrenar el modelo base durante 62 horas, y 128 para el modelo large durante 38 horas, lo cual equivale aproximadamente a 2000 y 4900 horas GPU respectivamente.

Nos enfocaremos en la tarea de modelado de lenguaje enmascarado (MLM), utilizando señales discretas como objetivo, y desviándonos de la mayoría de los trabajos de aprendizaje de representaciones generales de audio, los cuales suelen utilizar regresión como objetivo. En este sentido, nuestro trabajo se acerca más al de representaciones de habla como HuBERT y DiscreteBERT, y a BEATs en el caso de representaciones generales de audio. Al mismo tiempo, evitaremos el uso de patches rectangulares de espectrogramas, ya que queremos que las representaciones

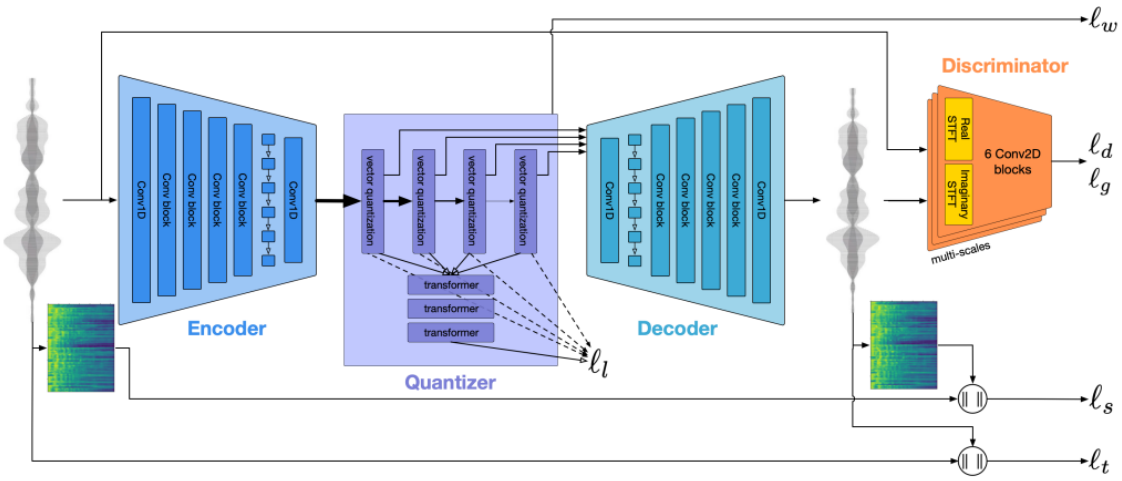

Figura 3.1: Arquitectura de EnCodec. Extraído de [108].

resultantes mantengan una buena resolución y estructura temporal. A su vez, creemos que el uso de patches de espectrogramas es una decisión que se trajo del mundo del procesamiento de imágenes, y que es subóptima para trabajar con señales de audio.

3.1. Modelo propuesto

A grandes rasgos, el modelo propuesto consiste de dos redes. Una red genera la señal objetivo discreta a partir del audio sin enmascarar, mientras que la otra red debe predecir esta señal objetivo a partir de una versión enmascarada del audio. La red que genera la señal objetivo es EnCodec, un codec neuronal que transforma las señales de audio en secuencias discretas. La red que intenta predecir estas señales discretas consiste de un Masked Autoencoder (MAE). A continuación se describen los distintos componentes del modelo propuesto.

3.1.1. EnCodec

Como se discutió en la sección 2.3.2, es posible aprender representaciones discretas de audio y obtener una buena calidad de reconstrucción mediante el uso de cuantizadores residuales (RVQ). EnCodec [108] es uno de los primeros modelos en utilizar esta técnica y contar con código y pesos de acceso libre. Este modelo es causal y capaz de comprimir audio con una tasa de muestreo de 24kHz a distintas tasas de información: 1.5, 3, 6, 12 o 24 kbps. También posee una variante no causal para audios estereofónicos de 48kHz entrenada solamente con música. Debido a que queremos un modelo capaz de representar no solamente señales de música sino también de habla y sonidos ambientales, utilizamos la versión causal de 24kHz que fue entrenada con segmentos de habla de DNS challenge 4 [185] y CommonVoice [186], audio genérico de AudioSet [97] y FSD50K [187], y canciones de Jamendo [188].

Como se puede observar en la Figura 3.1, EnCodec consiste de un encoder

compuesto de 4 bloques convolucionales que reducen la señal de audio en el eje temporal de 24kHz a 75Hz (un factor de 320). Los bloques convolucionales son seguidos por 2 capas LSTM y una capa convolucional final. La salida del encoder x_e es una secuencia de 75 vectores 128-dimensionales por segundo, los cuales son cuantizados mediante la capa de RVQ (llamada Quantizer en la Figura 3.1). La capa de RVQ consiste de $Q = 32$ codebooks, cada uno con $K = 1024$ códigos. El primer cuantizador de RVQ buscará los códigos del primer codebook que sean más cercanos en distancia euclidea a x_e. Habrá un error o residuo de cuantización que surge de la diferencia entre los códigos asignados y la señal original x_e. El siguiente cuantizador intentará aproximar este residuo dando lugar a una nueva secuencia de índices discretos. De esta forma, cada cuantizador refinará la aproximación de x_e reduciendo el error de cuantización y en consecuencia mejorando la calidad de reconstrucción. Finalmente, el audio de entrada x queda codificado mediante una matriz discreta $Y \in \mathbb{N}_{<K}^{Q \times T_f}$ donde $\mathbb{N}_{<K} = \{0,1,\ldots,K-1\}$ y T_f depende del largo de la señal.

Un decoder que posee una arquitectura similar al encoder pero espejada, toma como entrada la aproximación mediante RVQ de x_e y devuelve una aproximación $\hat{x}$ de x. Se optimizan una serie de pérdidas de reconstrucción para asegurar que x y $\hat{x}$ sean similares. Por un lado, se busca minimizar una pérdida de reconstrucción ℓ_t la cual consiste de una distancia L_1 en el dominio del tiempo, y una pérdida de reconstrucción ℓ_s en el dominio tiempo-frecuencia utilizando una combinación de distancia L_1 y L_2 entre múltiples espectrogramas con distintos tamaños de ventana y salto. Por otro lado, se utilizan discriminadores que operan sobre la STFT compleja de las señales introduciendo una pérdida adversaria para el generador (ℓ_g) y los discriminadores (ℓ_d) con el fin de mejorar la calidad perceptual de las reconstrucciones. También se introduce ℓ_w, que es la pérdida de commitment que actualiza el encoder para que sus salidas se acerquen a las del cuantizador, y ℓ_l que es una función de costo opcional para entrenar un modelo de lenguaje sobre los códigos que permite reducir aún más la tasa de información al utilizar codificación por entropía. Por último, para poder utilizar un solo modelo que opere a distintas tasas de compresión durante inferencia, se elige aleatoriamente una cantidad de cuantizadores en cada paso del entrenamiento, por lo que el decoder aprende a reconstruir la señal a partir de distintas cantidades de cuantizadores correspondientes a distintas tasas de compresión. A modo de ejemplo, en un paso de entrenamiento podrían usarse solo los primeros 16 cuantizadores, en el siguiente solo los primeros 8, y en el siguiente los 32. Esto induce una organización jerárquica en los cuantizadores, de manera que los primeros cuantizadores serán los más importantes en términos de reconstrucción, mientras que los últimos serán útiles para reconstruir detalles más finos, ya que solo serán utilizados cuando se usen todos los cuantizadores.

3.1.2. EnCodecMAE

En este trabajo, utilizamos EnCodec como una manera de representar de forma discreta la señal de audio, de forma que se pueda entrenar un modelo análogo a BERT,

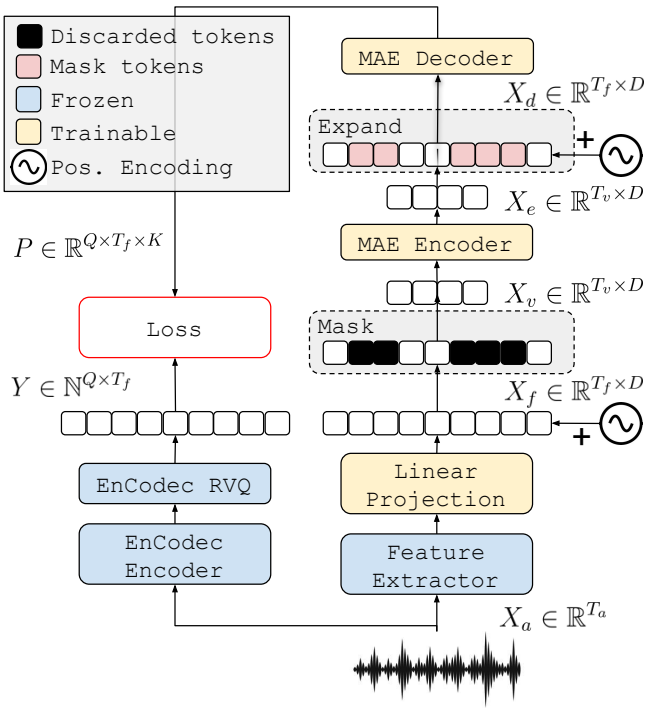

Figura 3.2: Estructura general del modelo propuesto, EnCodecMAE.

utilizando esta señal como objetivo. Esto es similar a DiscreteBERT y HuBERT, con la diferencia de que EnCodec es un modelo de audio general y utiliza RVQ por lo que es esperable que capture más detalles de la señal original que VQ-Wav2Vec, el cual fue entrenado solo con habla y sin un objetivo de reconstrucción, o que K-Means aplicado sobre MFCCs que favorecerá capturar ciertas características de la señal de entrada, como el contenido fonético.

Como puede observarse en la Figura 3.2, el modelo propuesto, al cual nos referiremos como EnCodecMAE, consiste de una etapa de extracción de atributos, en la que la señal de audio X_a es transformada en una representación local y proyectada linealmente a una dimensión D dando lugar a $X_f \in \mathbb{R}^{T_f \times D}$. Se suman embeddings posicionales sinusoidales a la secuencia X_f y posteriormente, un porcentaje de la señal M_{prop} es enmascarado mediante el muestreo aleatorio de índices de comienzo de máscara, y el enmascaramiento de los siguientes M_{gap} cuadros. A diferencia de wav2vec 2.0 o HuBERT, en lugar de utilizar un token de máscara, los cuadros enmascarados directamente se descartan de la secuencia y no se pierde información posicional dado que se agregó antes del enmascaramiento. Esta estrategia de enmascarado, como se discutió en la sección 2.3, permite reducir efectivamente el largo de la secuencia, lo cual es beneficioso en arquitecturas Transformer permitiéndonos acelerar el entrenamiento de manera significativa. Siguiendo la arquitectura de Masked Autoencoder, un transformer (el encoder), devuelve una representación contextualizada X_e a partir de los tokens visibles X_v. Luego, se insertan tokens de máscara en las posiciones que habían sido descartadas y se suman los embeddings posicionales generando X_d. Finalmente, una red transformer más pequeña (el decoder) toma como

entrada X_d y una última capa lineal transforma la salida del decoder en un tensor $P \in \mathbb{R}^{Q \times T_f \times K}$ con probabilidades para cada uno de los K códigos correspondientes a los Q cuantizadores residuales de EnCodec, y para cada cuadro de la señal objetivo.

Finalmente, se calcula la entropía cruzada C entre cada salida del decoder P y cada código de EnCodec Y. Se introducen pesos para las partes enmascaradas y sin enmascarar, y para cada cuantizador, dando lugar a la siguiente función de costo:

$$L = \sum_{q=0}^{Q} \gamma_q \left[\alpha \sum_{t \in M} C(y_q(t), P_q(t)) + \beta \sum_{t \notin M} C(y_q(t), P_q(t)) \right],$$

en donde γ_q es el peso dado a cada cuantizador, M es el conjunto de cuadros enmascarados, $\alpha = \frac{\delta}{|M|}$ es el peso dado a las partes enmascaradas en la función de costo y $\beta = \frac{1-\delta}{|M|}$ es el peso dado a las partes no enmascaradas. δ controla el balance en la función de costo entre las partes enmascaradas y no enmascaradas y C es la entropía cruzada $C(y_q(t), P_q(t)) = -\log(P_{q,y_q(t)}(t))$. Si $\delta = 0$ se está en el caso en que solo se penaliza la reconstrucción de los cuadros visibles, mientras que si $\delta = 1$, solo importan en la reconstrucción los cuadros que fueron enmascarados.

3.2. Evaluación de los modelos

> "Si no puedes medirlo, no puedes mejorarlo."
>
> *Lord Kelvin*

Los modelos aquí propuestos, y los utilizados en otros trabajos, generarán distintas representaciones de audio. A la hora de guiar el diseño de estos modelos, hay que poder definir alguna medida de qué representación es mejor. De esta forma, tras la experimentación, podremos tomar decisiones en el diseño que lleven a mejores representaciones. El principal problema en este campo es que no es claro qué representación es mejor, o más bien la respuesta dependerá principalmente de la aplicación, y la forma en la que se utilice la representación.

Nuestro objetivo en este trabajo es generar una representación que sea un buen punto de partida a la hora de resolver tareas de audio. Queremos que idealmente sea útil para cualquier tarea, constituyendo un modelo universal de audio. Esto puede sonar muy ambicioso, sin embargo es parte de una tendencia en el campo del aprendizaje automático en la que se intentan generar modelos cada vez más generalistas. Un ejemplo claro se da en el campo del NLP, en donde inicialmente se contaba con modelos muy específicos que requerían conocimiento experto y mucha manipulación de datos para que funcionaran. Estos modelos solo podían resolver una tarea muy específica y la manera de evaluarlas era clara; maximizar el desempeño en uno o más conjuntos de datos relevantes a la tarea. Con el auge de la transferencia de aprendizaje se buscó unificar la manera de representar y manipular los datos (primero utilizando representaciones como Word2Vec, y más tarde BERT y sus derivados), por lo que resolver tareas de distintos dominios del NLP se volvió progresivamente

una "receta" más uniforme: se extraen representaciones con BERT y se entrena un modelo pequeño que toma su salida como entrada o se finetunea directamente el modelo. Ese procedimiento resulta, en muchos casos, en un nuevo estado del arte en la tarea de interés.

Dado que la cantidad de tareas donde estas representaciones son útiles es cada vez mayor, ¿cómo evaluamos la representación en sí misma? Una de las formas de evaluación más comunes es el uso de benchmarks extensivos, en donde se utilizan representaciones para resolver una cantidad fija de tareas y se reporta el desempeño en cada una de ellas. Algunos ejemplos de benchmarks en el campo de NLP son GLUE [189], SuperGLUE [190] y MTEB [191]. En cada benchmark se debe definir qué tareas y conjuntos de datos se evaluarán, qué modelos downstream se utilizarán y qué metricas se reportarán. Respecto al modelo downstream, las opciones más comunes son:

- **Modelo Lineal**: permite determinar si las clases son linealmente separables en la representación. En general, solo requiere hacer una búsqueda del peso de regularización adecuado, por lo que la exploración de hiperparámetros es sencilla. La principal desventaja es que no puede capturar relaciones no lineales en la representación, aunque esto ayuda a mitigar el sobreajuste, en especial cuando los conjuntos de datos son pequeños. Particularmente en nuestro caso, en el que queremos resolver tareas diversas, es esperable que no se puedan separar linealmente las clases de todas las tareas al mismo tiempo para una única representación.
- **Perceptrón Multicapa**: introduce una o más capas ocultas, lo cual permite modelar alinealidades. Hay que determinar un número de capas y neuronas ocultas. Sigue siendo un modelo sencillo y computacionalmente barato de entrenar, aunque se corre mayor riesgo de sobreajuste comparado con el modelo lineal.
- **K-Vecinos más cercanos**: la predicción es determinada por la clase a la que pertenecen los K vecinos más cercanos a la representación. Posee pocos hiperparámetros, aunque requiere almacenar el conjunto de datos de entrenamiento completo en memoria, lo cual puede ser prohibitivo para bases de datos muy grandes, y se corre riesgo de sobreajuste al utilizar valores de K chicos.
- **Finetuning**: en este procedimiento, el modelo, y en consecuencia la representación, pueden ajustarse para resolver la tarea. En este sentido, lo que se evalúa no es tanto la representación sino la capacidad del modelo de adaptar la representación a otra nueva que sea útil para la tarea. Esta técnica es en general computacionalmente más costosa que las demás dado que se entrenan parámetros del modelo upstream, el cual suele poseer una cantidad de parámetros mucho mayor que el downstream. A su vez, se corre mayor riesgo de sobreajuste y requiere más cuidado en la selección de hiperparámetros.

En el campo del procesamiento de audio y habla se tomaron direcciones similares a las de NLP dando lugar a varios benchmarks:

- **SUPERB** [5]: evalúa representaciones en 10 tareas de procesamiento del habla: ASR, reconocimiento de fonemas, keyword spotting, recuperación de términos hablados, identificación y verificación de hablante, diarización, clasificación de intención, slot-filling y reconocimiento de emociones. En la mayoría de las tareas se utiliza un modelo lineal como downstream, mientras que otras más complejas como ASR utilizan una red Bi-LSTM. A su vez, en lugar de utilizar las activaciones de una única capa del modelo upstream, aprenden un promedio pesado de las activaciones de todas las capas, evitando tener que realizar una búsqueda de la mejor capa para cada tarea. Una de las ventajas de este benchmark es su popularidad y que cuenta con un leaderboard y código abierto, permitiendo una fácil comparación de distintas representaciones. Su principal desventaja es que la evaluación es muy costosa pudiendo tomar varios días evaluar un modelo en todas las tareas [192].
- **HEAREval** [4]: evalúa representaciones de audio en 19 tareas diversas relacionadas a análisis de pitch, habla, eventos acústicos, instrumentos musicales, géneros musicales y vocalizaciones entre otras. En el anexo A se brindan más detalles de las 19 tareas evaluadas. Respecto al modelo downstream, se utiliza un perceptrón multicapa y se exploran distintos hiperparámetros como la cantidad de capas ocultas (1 o 2), la tasa de aprendizaje y el tipo de inicialización. Al igual que SUPERB cuentan con un leaderboard y código. Además la evaluación no es muy costosa y varios trabajos han reportado su desempeño en este benchmark [162, 165, 193, 194]
- **HARES** [195]: consiste de 12 tareas diversas de audio general, incluyendo reconocimiento de escenas y eventos acústicos, clasificación de sonidos de aves, tareas de habla como identificación de hablante, lenguaje y comandos, y tareas relacionadas a música como reconocimiento de nota musical, instrumento y tagging. El modelo downstream es lineal excepto por la tarea de detección de eventos acústicos que utiliza un MLP.
- **NOSS** [131]: evalúa representaciones en 6 tareas de habla no semánticas: identificación de hablante, de lenguaje, de comando, detección de demencia y reconocimiento de emociones en dos bases de datos distintas. Exploran distintos clasificadores pequeños: regresión logística, random forests y linear discriminant analysis (LDA).

Para este trabajo, optamos por realizar nuestros experimentos iniciales utilizando un subconjunto de HEAREval. Esta elección se fundamenta en que el costo computacional de evaluación en este benchmark es relativamente bajo, existen resultados previamente reportados en la literatura, y cubre tareas diversas de audio. Elegimos un subconjunto de 6 tareas que se basan en datasets ampliamente aceptados por la comunidad científica, permitiéndonos realizar comparaciones más confiables y evitando la dependencia en conjuntos de datos menos estandarizados o de uso marginal:

- **NSynth (NS)** [114]: consiste de notas individuales ejecutadas mediante distintos instrumentos musicales. La tarea a resolver es la de clasificar qué nota se está

tocando, habiendo 88 notas posibles. Se utiliza accuracy como métrica de evaluación.

- **GTZAN (GC)** [32]: consiste en segmentos de 30 segundos de canciones que pueden corresponderse a 10 géneros musicales distintos: blues, música clásica, country, disco, hiphop, jazz, metal, pop, reggae y rock. La tarea consiste en clasificar a qué género pertenece una canción. El conjunto de datos se organiza en 10 folds y se reporta accuracy.
- **Google Speech Commands (SC)** [196]: consiste en clasificar comandos de voz, pertenecientes a un conjunto de 10 posibles comandos, ruido y desconocido. Se utiliza accuracy como métrica de evaluación.
- **CREMA-D (ER)** [197]: consiste en 12 oraciones dichas por 91 actores distintos con alguna de las siguientes emociones: enojo, asco, miedo, felicidad, neutro y triste; y distintas intensidades de emoción. La tarea consiste en clasificar la emoción a partir de la señal de habla. La métrica utilizada es accuracy y el conjunto de datos se organiza en 10 folds.
- **Freesound Dataset 50K (FSD)** [187]: contiene 100 horas de audios con anotaciones de eventos acústicos, los cuales pueden pertenecer a 200 categorías distintas. La tarea consiste en determinar qué eventos ocurrieron en un segmento de audio, y es multi-etiqueta. Se utiliza mean Average Precision (mAP) como métrica de evaluación.
- **ESC-50 (ESC)** [198]: contiene 2000 grabaciones de 5 segundos en los que ocurre un único sonido, el cual pertenece a una de 50 categorías posibles. Algunos ejemplos de categorías son: lluvia, perro, bebe llorando, golpe de puerta y helicóptero. Se organiza en 5 folds y utiliza accuracy como métrica.

3.3. Detalles experimentales

Datos de pre-entrenamiento

Para el pre-entrenamiento del modelo utilizamos una mezcla de datos consistente de AudioSet [97], Libri-Light Medium [199] y Free Music Archive [200]. AudioSet consiste de unos 2 millones de segmentos de audio de 10 segundos provenientes de YouTube abarcando sonidos diversos. Debido a que el mismo consiste de URLs a los videos, y que algunos videos pueden ser eliminados a lo largo del tiempo o no estar disponibles en ciertas regiones geográficas, generalmente se logra descargar un subconjunto de los 2M de videos. En nuestro caso, logramos descargar alrededor de 1.6M de videos, sumando más de 4500 horas de audio. Free Music Archive consiste de 106574 segmentos de canciones con una duración de 30 segundos, totalizando unas 800 horas de música. Por último, LibriLight consiste de habla en idioma inglés proveniente de audiolibros, y en el caso del subconjunto Medium, totaliza 6k horas de habla. Decidimos utilizar este subconjunto en vez de las 60k horas del conjunto completo para que los audios de habla no predominen. En total, nuestra mezcla de conjuntos de datos totaliza alrededor de 11300 horas o 1.29 años de audio. A

su vez, para poder compararnos con otros trabajos que utilizaron solo Audioset o LibriLight, realizamos experimentos utilizando cada conjunto de datos por separado y analizamos el efecto de los mismos en el desempeño de cada tarea.

Hiperparámetros y detalles de pre-entrenamiento

Los parámetros de EnCodec fueron descargados de HuggingFace[1] y no fueron actualizados durante el pre-entrenamiento. En cuanto a la cantidad de cuantizadores utilizados como señal objetivo, escuchamos reconstrucciones de audios diversos utilizando distintas cantidades de cuantizadores, y concluímos que con 8 cuantizadores era suficiente para conservar los detalles perceptualmente más relevantes de la señal original. Por otro lado, en el trabajo de EnCodec se realizaron pruebas subjetivas MUSHRA con señales de música y habla, utilizando distintas tasas de bits, correspondientes a distintas cantidades de cuantizadores. En la Figura 3.3 se muestra la relación entre tasa de bits y puntaje MUSHRA. Se puede observar que pasar de 2 a 4 cuantizadores (1.5 kbps a 3.0 kbps) tiene un impacto importante en el valor de MUSHRA (un aumento relativo del 36.2%), mientras que pasar de 8 a 16 cuantizadores solo trae un aumento relativo del 5.5%, lo cual respalda nuestra decisión de utilizar 8 cuantizadores.

Determinamos algunos de los hiperparámetros en una serie de pruebas preliminares en las que pre-entrenamos modelos por solo 150000 pasos de entrenamiento y los evaluamos en los conjuntos de validación de HEAREval. Este proceso resultó en un peso de enmascarado $\sigma = 0.9$, una proporción de enmascarado $M_{prop} = 0.5$ y un largo de máscaras $M_{gap} = 15$. A su vez, encontramos que utilizar pesos uniformes γ_q en la función de costo de EnCodecMAE, resultaba en un desempeño inferior a sólo utilizar el primer cuantizador (Q=1), y atribuimos este comportamiento a que el modelo estaba utilizando su capacidad de igual manera para predecir el primero o el último cuantizador, cuando el primero es el que codifica la mayor parte de la señal. Dada esta hipótesis, probamos utilizar pesos no uniformes, dándole mayor peso a los primeros cuantizadores. Para definir los pesos, tomamos 150 muestras aleatorias del conjunto de pre-entrenamiento y medimos el error de cuantización en cada nivel del RVQ. Luego, normalizamos estos valores para que su suma sea 1 obteniéndose los γ_q. Este pesado no uniforme dió ventajas por sobre utilizar solo el primer cuantizador. En el capítulo 4 se mostrará el efecto de cambiar los valores de σ, M_{prop}, M_{gap} y γ_q en el modelo final, junto con otros análisis y estudios de ablación.

Entrenamos el modelo con tres tamaños distintos: un modelo Base utilizando 10 capas de transformer en el encoder con representaciones de 768 dimensiones, un modelo Large utilizando 20 con 1024 dimensiones, y una versión más pequeña, Small, con solo 5 capas y 768 dimensiones. Todas las variantes utilizan un decoder con 2 capas. Originalmente, deseabamos mantener los tamaños de modelo estándar de la literatura de Transformers, que suelen utilizar 12 capas para su modelo Base y 24 para el Large. La lógica de razonamiento inicial fue que el modelo Base tiene efectivamente 12 capas (10 del encoder y 2 del decoder), sin embargo, esto difiere

[1] https://huggingface.co/facebook/encodec_24khz

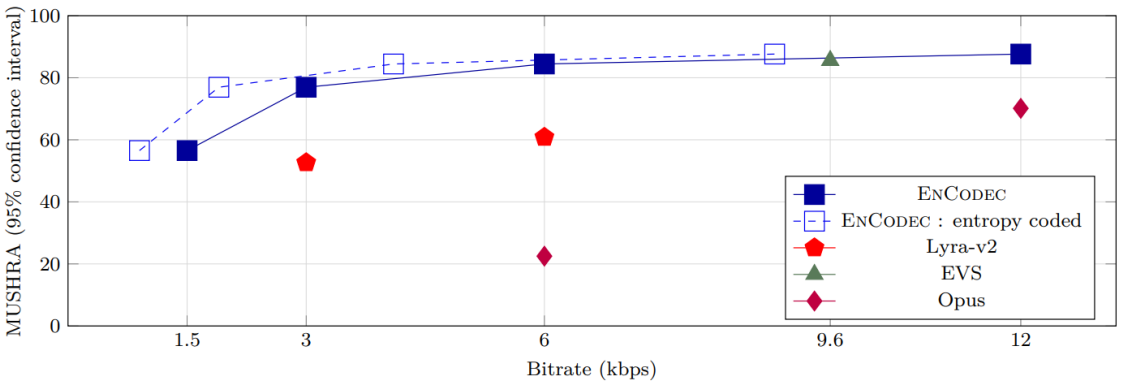

Figura 3.3: Puntaje subjetivo MUSHRA de EnCodec y otros codecs de audio en función de la tasa de bits. Extraído de [108].

un poco de la literatura de MAE, en la que solo se tiene en cuenta el tamaño del encoder, el cual posee 12 capas en el tamaño Base y 24 en el Large. A pesar de estas diferencias, que nos pondrían en desventaja al comparar con otros trabajos, la cantidad de parámetros de nuestros modelos es similar a la de otros trabajos: 43.6M para el modelo Small, 86.6M para el modelo Base y 261.6M para el Large. Esto se debe a que utilizamos una dimensionalidad de 4096 en lugar de 2048 en las capas feed-forward del Transformer. También utilizamos $N_h = 12$ cabezas de atención en lugar de 8, aunque esta decisión no altera la cantidad de parámetros del modelo ya que la dimensionalidad del modelo D_{model} se divide entre las N_h cabezas de atención, dando lugar a W_q,W_k y W_v con dimensiones $D_{model} \times N_h \cdot D_{model}/N_h$, las cuales no dependen de N_h. Del mismo modo, la matriz W_o tendrá dimensiones $D_{model} \times D_{model}$ sin importar el valor de N_h.

Respecto al extractor de atributos, en un principio utilizabamos el mismo encoder de EnCodec, tomando sus salidas antes de ser cuantizadas mediante RVQ, y manteníamos fijos sus pesos durante el pre-entrenamiento, pero al notar mejoras significativas al utilizar melspectrogramas, optamos por esta representación de entrada. Los melspectrogramas se calculan mediante torchaudio, utilizando una ventana Hanning de 640 muestras con un salto de 320, y 256 filtros mel. El tamaño de salto se eligió para tener secuencias con la misma longitud que la señal objetivo de EnCodec (75 muestras por segundo), mientras que el tamaño de ventana se eligió para que haya un solapamiento del 50% de la ventana. En la Sección 4.1 se realiza un análisis más profundo de ablación respecto al efecto de distintas representaciones de entrada.

Nuestros modelos fueron entrenados en 500k pasos de entrenamiento utilizando el optimizador AdamW con hiperparámetros $\beta_1 = 0.9, \beta_2 = 0.95$ y regularización L2 con un peso $\lambda = 0.05$. Los hiperparámetros del optimizador fueron tomados del trabajo original de MAE [3], mientras que la cantidad de pasos se determinó en base a las curvas de pre-entrenamiento, las cuales se pueden observar en la Figura 3.4 y al tiempo de pre-entrenamiento requerido para esa cantidad de pasos. Los 500k pasos de entrenamiento, con nuestras configuraciones de tamaño de lote y conjuntos de pre-entrenamiento corresponden a aproximadamente 7 épocas y unos 5 días de

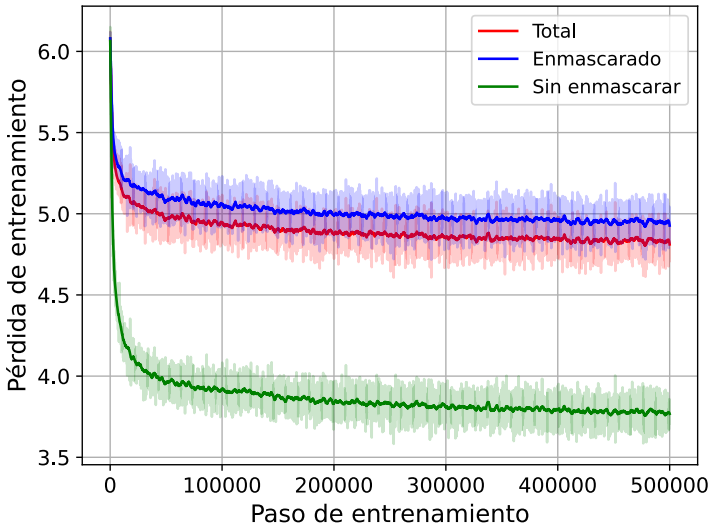

Figura 3.4: Función de pérdida durante el pre-entrenamiento del modelo base con melspectrograma como entradas. Se muestran desglosados los términos provenientes de los segmentos enmascarados y los no enmascarados.

pre-entrenamiento. En la Sección 4.3 mostraremos un análisis de cómo impacta en el desempeño downstream detener el pre-entrenamiento en distintos puntos, para distintos tamaños de EnCodecMAE. La tasa de aprendizaje se mantuvo fija en 10^{-4}, tras haber observado que utilizar warm-up o decay exponencial no daba ventajas en las curvas de entrenamiento. Creemos que esta estabilidad en el entrenamiento se atribuye a que utilizamos pre-post normalización en lugar de pre o postnormalización dentro de los transformers.

Se utilizó un tamaño de batch de 128 segmentos de audio con duración de 4 segundos. Estos valores se determinaron en base a los recursos computacionales disponibles; tamaños de batch o longitudes de audio mayores generaban problemas de memoria. Para archivos de audio de duración mayor a 4 segundos, se eligió aleatoriamente un segmento del mismo. A su vez, cada audio es elegido durante una época de entrenamiento $\lfloor T/4 \rfloor$ veces, en donde T es la duración del audio. Por ejemplo, algunos audios de LibriLight pueden durar 5 minutos por lo que serán elegidos 75 veces, mientras que los audios de Audioset suelen durar 10 segundos, y por ende se toman 2 segmentos aleatorios de cada uno por época.

Los experimentos se realizaron principalmente en dos equipos con las siguientes especificaciones:

- Computadora de escritorio con 2 tarjetas gráficas RTX 3090 de 24GB de memoria VRAM cada una, un procesador AMD Ryzen Threadripper 3960X con 24 núcleos y 128 GB de memoria principal.
- Servidor con 2 tarjetas gráficas RTX A5000 de 24GB de memoria VRAM cada una, un procesador AMD Ryzen 9 7950X de 16 núcleos y 128 GB de memoria

principal.

Los modelos de tamaño Base se entrenaron con una sola GPU en 5 días, mientras que los de tamaño Large se entrenaron con 2 GPUs en la misma cantidad de tiempo. La razón de utilizar 2 GPUs para el modelo Large fue que con el tamaño de batch de 128 se incurría en problemas de memoria por lo que paralelizamos a nivel de datos entre las 2 GPUs logrando mantener el tamaño de batch. En la Tabla 3.1 se resumen los distintos hiperparámetros utilizados para EnCodecMAE.

	Small	Base	Large
Optimizador	AdamW		
Momento del Optimizador	$\beta_1 = 0.9, \beta_2 = 0.95$		
Weight Decay	0.05		
Programación de LR	Constante		
LR	1e-4		
Pasos de entrenamiento	500000		
GPUs	1		2
Tamaño de Lote	128		
Duración de segmentos	4s		
Parámetros de enmascarado	$M_{prop} = 0.5, M_{gap} = 15, \sigma = 0.9$		
Q	8		
γ_q	[0.224, 0.176, 0.145, 0.121, 0.103, 0.088, 0.076, 0.066]		
Capas del Encoder	5	10	20
Capas del Decoder	2		
D_{model}	768		1024
Cabezas de atención	12		
Dropout	0		

Tabla 3.1: Resumen de los hiperparámetros utilizados en EnCodecMAE.

Para la evaluación de los modelos ya pre-entrenados, optamos por utilizar la salida del Encoder como representación. Esta salida es una secuencia temporal, mientras que los downstreams de HearEval operan sobre vectores con dimensionalidad fija. Siguiendo la metodología de la mayoría de los trabajos en el área [160, 161, 195], resumimos la secuencia en un vector de tamaño fijo mediante un promediado en el tiempo.

Cuando los audios tenían duración mayor a 4 segundos, se particionaron en segmentos de 4 segundos sin solapamiento y para cada uno de estos segmentos se extrajeron las salidas del Encoder, se concatenaron, y se promediaron en el tiempo. Este procedimiento se alinea con el pre-entrenamiento, en el que el modelo aprendió a representar segmentos de audio de 4 segundos. Si no realizáramos este enventanado, utilizaríamos embeddings posicionales no vistos durante el pre-entrenamiento, lo cual podría perjudicar a la calidad de las representaciones, además de que se incrementaría el costo computacional durante la inferencia respecto a realizar el enventanado.

Para obtener un intervalo de confianza alrededor de las métricas reportadas en HearEval, realizamos la técnica de bootstrapping [201] con 100 iteraciones utilizando el paquete de Python `confidence_intervals` [202]. Esta técnica consiste en generar N conjuntos de evaluación mediante un muestreo aleatorio con reposición del conjunto

de evaluación original, y calcular el desempeño en estos N conjuntos, obteniendo una distribución empírica de la métrica a partir de la cual se puede estimar un intervalo de confianza mediante el uso de percentiles.

Refinamiento iterativo de la señal objetivo

En nuestro trabajo decidimos explorar la técnica de refinamiento iterativo de la señal objetivo, la cual se utiliza en trabajos previos como HuBERT y BEATs. En una etapa inicial, HuBERT utiliza como señal objetivo clusters de MFCCs, mientras que BEATs utiliza proyecciones cuantizadas aleatorias. Esta primera iteración da lugar a un modelo cuyo desempeño es razonable pero no alcanza al estado del arte. En una segunda iteración, se utilizan como señal objetivo representaciones intermedias del modelo de la primera iteración. En el caso de HuBERT, se realiza clustering sobre la sexta capa mediante K-Means. En el caso de BEATs, en lugar de utilizar K-Means, entrenan un modelo similar a VQ-VAE para aproximar la salida del modelo de la primera iteración. Descartando el decoder, se obtiene un tokenizador que genera las señales objetivos.

Para mejorar las representaciones de EnCodecMAE, optamos por adoptar la misma estrategia y realizar una segunda iteración de entrenamiento. Para esto entrenamos un modelo de K-Means con $K = 1024$ utilizando 10000 señales de audio (balanceadas por conjunto de datos) y extrayendo activaciones de la última capa del encoder de EnCodecMAE. Continuamos entrenando EnCodecMAE con el conjunto de pre-entrenamiento completo por 150000 iteraciones más utilizando como señal objetivo las salidas del modelo de K-Means que entrenamos. Denominamos al modelo resultante Mel→EC (Base + ST). En el caso de los modelos Large, siguiendo la lógica de HuBERT, realizamos K-Means sobre las activaciones de la última capa del modelo Mel→EC (Base + ST), por ende, este modelo puede pensarse como el resultado de una tercera iteración.

3.4. Comparación con el Estado del Arte

Una vez pre-entrenados nuestros modelos, los comparamos con otros modelos de representación de audio de la literatura. Algunos de estos trabajos reportan resultados en HearEval, mientras que para otros, utilizamos su código oficial y realizamos la evaluación. El criterio general para elegir estos modelos fue:

- Modelos que hayan reportado un desempeño fuerte en varias tareas generales de audio o en el benchmark de HearEval.
- Para una comparación justa, sólo consideramos modelos que no utilizaron etiquetas durante su pre-entrenamiento, es decir, realizaron un pre-entrenamiento no supervisado. Cabe notar que modelos entrenados de forma supervisada en AudioSet es esperable que alcancen un mejor desempeño en FSD50K, ESC y GC, ya que las etiquetas de AudioSet contienen información de eventos acústicos y de metadatos musicales como género e instrumentos. Del mismo modo se

	Music		Speech		Env		Global	#P(M)	Datos
	NS	GC	SC	ER	FSD	ESC			
(1) Mel→EC (Small)	$85.9_{\pm 1.2}$	$87.5_{\pm 1.9}$	$95.2_{\pm 0.7}$	$74.8_{\pm 0.9}$	$48.5_{\pm 1.2}$	$80.1_{\pm 1.7}$	94.8	43.6	A+F+L6
(2) EC→EC (Base)	$\mathbf{91.7_{\pm 1.0}}$	$85.6_{\pm 1.9}$	$92.2_{\pm 1.0}$	$72.0_{\pm 1.0}$	$44.1_{\pm 1.3}$	$74.6_{\pm 2.1}$	83.4	94.0	A+F+L6
(3) Mel→EC (Base)	$84.6_{\pm 1.0}$	$87.6_{\pm 1.9}$	$96.3_{\pm 0.5}$	$75.5_{\pm 1.0}$	$49.5_{\pm 1.1}$	$79.8_{\pm 1.8}$	95.9	86.6	A+F+L6
(4) AS only	$86.1_{\pm 1.1}$	$87.1_{\pm 1.9}$	$93.6_{\pm 0.7}$	$73.8_{\pm 1.1}$	$51.7_{\pm 1.3}$	$81.5_{\pm 2.1}$	97.0	86.6	A
(5) FMA only	$87.7_{\pm 1.1}$	$\mathbf{89.2_{\pm 2.0}}$	$87.4_{\pm 0.9}$	$68.7_{\pm 1.1}$	$48.5_{\pm 1.3}$	$80.4_{\pm 2.2}$	89.1	86.6	F
(6) FMA+JAM	$89.6_{\pm 1.0}$	$\mathbf{89.5_{\pm 1.8}}$	$85.9_{\pm 1.1}$	$66.9_{\pm 1.2}$	$48.5_{\pm 1.1}$	$80.6_{\pm 2.1}$	85.4	86.6	F+J
(7) LL6K only	$83.4_{\pm 1.3}$	$83.4_{\pm 2.0}$	$95.4_{\pm 0.6}$	$75.6_{\pm 1.1}$	$41.3_{\pm 1.2}$	$68.8_{\pm 1.6}$	75.3	86.6	L6
(8) Mel→EC (Base + ST)	$81.7_{\pm 1.4}$	$\mathbf{87.8_{\pm 2.1}}$	$\mathbf{96.9_{\pm 0.5}}$	$\mathbf{76.3_{\pm 1.0}}$	$50.7_{\pm 1.2}$	$81.6_{\pm 2.1}$	97.6	86.6	A+F+L6
(9) AS only	$83.5_{\pm 1.3}$	$\mathbf{87.7_{\pm 2.3}}$	$94.6_{\pm 0.6}$	$75.1_{\pm 1.1}$	$\mathbf{52.9_{\pm 1.1}}$	$\mathbf{84.6_{\pm 1.6}}$	99.2	86.6	A
(10) Mel→EC (Large)	$85.3_{\pm 1.4}$	$85.8_{\pm 2.2}$	$96.4_{\pm 0.5}$	$\mathbf{77.1_{\pm 0.9}}$	$50.3_{\pm 1.1}$	$80.2_{\pm 1.9}$	97.1	261.6	A+F+L6
(11) Mel→EC (Large + ST)	$83.4_{\pm 1.0}$	$87.2_{\pm 2.1}$	$\mathbf{97.0_{\pm 0.4}}$	$\mathbf{77.2_{\pm 0.9}}$	$51.0_{\pm 1.0}$	$84.1_{\pm 1.8}$	**100.0**	261.6	A+F+L6
(12) AS only	$86.4_{\pm 1.3}$	$86.4_{\pm 1.7}$	$94.4_{\pm 0.7}$	$75.7_{\pm 0.9}$	$\mathbf{53.4_{\pm 1.0}}$	$\mathbf{86.0_{\pm 1.8}}$	99.1	261.6	A
(13) BEATS (Iter 1) [152]	$82.4_{\pm 1.3}$	$85.4_{\pm 2.4}$	$91.0_{\pm 0.8}$	$68.7_{\pm 1.1}$	$50.6_{\pm 0.9}$	$82.3_{\pm 1.7}$	94.4	90	A
(14) BEATS (Iter 3) [152]	$82.9_{\pm 1.0}$	$86.2_{\pm 2.9}$	$90.4_{\pm 0.8}$	$69.0_{\pm 1.1}$	$\mathbf{53.5}_{\pm 1.1}$	$\mathbf{85.1_{\pm 1.5}}$	96.0	90	A
(15) BYOL-A [138]	$68.3_{\pm 1.4}$	$86.0_{\pm 2.0}$	$93.1_{\pm 0.7}$	$65.9_{\pm 1.1}$	$51.3_{\pm 0.9}$	$83.6_{\pm 2.0}$	82.6	5.3	A
(16) AudioMAE (PT) [160]	$67.2_{\pm 1.5}$	$72.0_{\pm 3.1}$	$45.2_{\pm 1.6}$	$39.3_{\pm 1.0}$	$37.6_{\pm 1.2}$	$58.5_{\pm 2.1}$	-32.1	86	A
(17) MSM-MAE-512 [162] †	81.2	86.1	86.4	73.4	52.2	**85.6**	91.7	86	A
(18) Fuse HuBERT [203] †	68.8	79.6	95.7	75.2	41.3	74.3	62.9	1000	L60
(19) Fuse Wav2Vec2 [203] †	60.6	79.3	**96.9**	69.2	40.3	69.5	51.4	317	L60
HEAR SOTA	90.0	90.8	97.8	75.2	65.5	96.6	100.0	-	-

Tabla 3.2: Desempeño de distintos modelos en un subconjunto de HearEval. El primer bloque corresponde a nuestros modelos, mientras que el segundo corresponde a modelos de la literatura. Nombramos nuestros modelos con la sintáxis Entrada→Objetivo, por ejemplo, Mel→EC toma melspectrogramas como entrada y predice tokens de EnCodec. ST se refiere al refinamiento iterativo. Las líneas con † muestran resultados copiados de los trabajos originales. Los trabajos sin este símbolo cuentan con código y pesos abiertos, los cuales utilizamos para la evaluación. Las desviaciones reportadas muestran la diferencia entre el valor de métrica obtenida y el borde del intervalo de confianza más lejano a ella, obtenido con $\alpha = 0.05$ mediante bootstrapping. Por último, se muestra el desempeño del mejor modelo del leaderboard de HEAREval en cada tarea, incluyendo modelos entrenados con supervisión y agregados más recientemente, y se marcan en negrita aquellos modelos cuyo desempeño promedio en una tarea cae dentro del intervalo de confianza del mejor modelo.

excluyeron modelos que hacen uso de información textual, como CLAP, ya que el texto suele contener etiquetas relacionadas con las tareas downstream.

- Modelos que hayan reportado desempeño en HearEval o que hayan liberado el código y pesos. Esto es importante para poder reportar métricas asociadas a su desempeño y realizar una comparación justa.

En la Tabla 3.2 se pueden observar los resultados obtenidos en HEAREval por los distintos modelos propios y de la literatura. Además de reportar el desempeño en cada tarea, decidimos reportar una medida de desempeño global. Uno podría realizar un promedio de los desempeños, sin embargo, al poseer cada tarea y métrica rangos distintos, se le estaría dando mayor peso a tareas con un desempeño normalmente elevado como SC, y poco peso a tareas como FSD, la cual reporta mAP en lugar de accuracy. Por esta razón, siguiendo la metodología en HEAREval, calculamos

z-scores utilizando la media y desviación estándar por tarea de las métricas obtenidas por los modelos del leaderboard HEAREval 2021. Esto nos da para cada tarea un valor que refleja a cuántas desviaciones estándar de la media en esa tarea se encuentra la métrica obtenida. Para evitar que alguna tarea en la que el modelo se desempeña muy bien predomine en la métrica, los valores se limitan al rango [-1,1], y finalmente se promedian y multiplica por 100. Como ejemplo, nuestro modelo Large con refinamiento iterativo de la señal objetivo (fila 11), posee un puntaje global de 100, siendo el más alto de la Tabla 3.2. Este puntaje indica que el modelo obtuvo métricas al menos una desviación estándar mejores que la media de HEAREval 2021 en todas las tareas. En cambio, AudioMAE (fila 16), obtuvo un valor global negativo, indicando que su desempeño fue en promedio menor a la media de HEAREval 2021. Una crítica a esta métrica es que las representaciones de audio están mejorando constantemente, y depender del resultado de modelos del año 2021 hará que pronto saturen en 100. Una posible solución es calcular las estadísticas para el z-score utilizando modelos más recientes, los cuales harán que la métrica global se reduzca y sea más desafiante llegar a un puntaje de 100. Sin embargo, esto requiere generar un nuevo leaderboard actualizado con modelos estado del arte.

Lo primero que se puede observar es que utilizar melspectrogramas (fila 3) como entrada en lugar de representaciones de la salida del encoder de EnCodec (fila 2) lleva a una mejora importante en todas las tareas excepto clasificación de nota musical (NS). Incluso, un modelo con la mitad de parámetros entrenado con melspectrogramas (fila 1) logra un mejor desempeño, indicando que la representación de entrada podría tener un rol incluso más importante que el tamaño de los modelos. En la sección 4.1 analizaremos en mayor profundidad el efecto de distintas representaciones de entrada en la calidad de las representaciones. A pesar de que en general utilizar melspectrogramas resulta en un mejor desempeño que utilizar EnCodec como entrada, la excepción en NS es remarcable. EC→EC obtiene el mejor desempeño para NS, incluso superando a Crepe [28] que fue específicamente entrenado para estimación de pitch y obtiene una accuracy de 90%. Al mismo tiempo, logra superar a otros modelos de la literatura como BYOL-A (fila 15), AudioMAE (fila 16), Fuse HuBERT (fila 18) y Fuse Wav2Vec2 (fila 19) en términos de desempeño global.

Respecto al tamaño de los modelos, se observa que incrementar su tamaño trae mejoras en términos de desempeño global; el modelo Base mejora al Small llevando el desempeño de 94.8 a 95.9 (filas 1 y 3) y el Large al Base llevándolo de 95.9 a 97.1 (filas 3 y 10) . Las tareas de reconocimiento de emociones (ER), reconocimiento de comandos de voz (SC) y detección de eventos acústicos (FSD) son las que muestran una mejora más marcada en su desempeño.

Por otro lado, la etapa de refinamiento iterativo de la señal objetivo también proporcionó mejoras a lo largo de distintos tamaños de modelo. En el caso del modelo base, el desempeño global aumentó de 95.9 a 97.6 (ver filas 3 y 8), mientras que en el modelo Large aumentó de 97.1 a 100.0 (ver filas 10 y 11). Cabe destacar que la etapa de refinamiento iterativo permitió al modelo base superar el desempeño global del modelo Large, sin incrementar la cantidad de parámetros o tiempo de inferencia.

En cuanto al efecto de los conjuntos de entrenamiento utilizados, se observa que

utilizar sólo AudioSet (filas 4, 9 y 12), mejora el desempeño en las tareas relacionadas a eventos acústicos (FSD y ESC) en comparación con los modelos que utilizan la mezcla de conjuntos de datos (filas 3, 8 y 11) u otros conjuntos de datos (filas 5, 6 y 7). Sin embargo, el desempeño en las tareas relacionadas al habla se ve deteriorado respecto a utilizar la mezcla de conjuntos de datos. Al utilizar el conjunto de datos FMA solo (fila 5) o en combinación con Jamendo (fila 6), los cuales consisten de canciones, se alcanza un mayor desempeño en clasificación de género musical y de notas musicales, pero se observa un deterioro importante en las tareas de habla. Particularmente, incluir canciones de Jamendo mejora el desempeño del modelo entrenado solo con FMA en NS, pero lo empeora en SC y ER. Creemos que esto puede ocurrir debido a que Jamendo tiene una menor proporción de canciones con voz cantada, haciendo que EnCodecMAE se enfoque más en la parte instrumental de la música que en el canto, y en consecuencia empeore el desempeño en tareas relacionadas al habla. Del mismo modo, al utilizar solo el conjunto de datos LibriLight (fila 7), el cual consiste de habla de audiolibros, se observa un buen desempeño en tareas de habla pero un deterioro importante en las tareas relacionadas a música y eventos acústicos. Esto indica un resultado que esperábamos—el tipo de sonidos utilizados durante el pre-entrenamiento determina el tipo de tareas en las que la representación será buena. En este sentido, la mezcla de datos y AudioSet son los conjuntos que dieron mejores resultados globales, con AudioSet beneficiando a las tareas relacionadas con eventos acústicos, y la mezcla beneficiando a tareas de habla, e incluso superando al modelo entrenado con LibriLight en estas (ver fila 3 vs 7 para tareas SC y ER).

BEATs, BYOL-A, AudioMAE y MSM-MAE-512 son modelos pre-entrenados con Audioset y con un tamaño Base, por lo que los compararemos con nuestros modelos Base entrenados solo en AudioSet (filas 4 y 9). Se puede observar que nuestros modelos se desempeñan significativamente mejor en las tareas de música y habla, y de forma comparable en las de eventos acústicos. Nuestras principales diferencias con BEATs son la definición de la señal objetivo inicial (EnCodec en nuestro caso, y cuantizadores aleatorios en el de BEATs), y el formato de la señal de entrada (cuadros en nuestro caso, y patches rectangulares en BEATs). Trabajos previos [161–163] encontraron que usar cuadros en lugar de patches rectangulares daba mejores resultados en tareas de habla pero afectaba negativamente a tareas de eventos acústicos. Una posible explicación es que el uso de patches rectangulares conlleva una pérdida de resolución temporal—se están abarcando múltiples cuadros en el tiempo y una banda de frecuencias—y una ganancia en resolución en frecuencia—distintos elementos de la secuencia capturan distintas bandas de frecuencia—. La pérdida de resolución temporal podría afectar a tareas de procesamiento del habla donde hay que discriminar señales de corta duración como fonemas, mientras que la ganancia de resolución en frecuencia podría ayudar a la discriminación de eventos acústicos. Evitar el uso de patches rectangulares, los cuales provienen de la comunidad de visión por computadora [85], y en cambio usar cuadros temporales, se alinea mejor con técnicas tradicionales de análisis de audio con ventanas, y representaciones como espectrogramas. El uso de cuadros temporales en nuestros modelos podría

Modelo	sin LM	con LM	#P(M)
Wav2Vec [134]	15.86	11.00	32.5
VQ-Wav2Vec [**?**]	17.71	12.80	34.1
DeCOAR 2.0 [154]	13.02	9.07	89.8
Wav2Vec 2.0 Base [75]	6.43	4.79	95.0
Wav2Vec 2.0 Large [75]	3.75	3.10	317.4
HuBERT Base [76]	6.42	4.79	94.7
HuBERT Large [76]	**3.62**	**2.94**	316.6
Mel→EC (Base)	$15.53_{\pm 1.29}$	$10.42_{\pm 1.04}$	86.6
LL6K only	$14.91_{\pm 1.21}$	$10.12_{\pm 0.96}$	86.6
Mel→EC (Base+ST)	$14.41_{\pm 1.28}$	$9.90_{\pm 0.99}$	86.6
LL6K only	$13.72_{\pm 1.28}$	$9.29_{\pm 0.96}$	86.6
Mel→EC (Large+ST)	$12.44_{\pm 0.99}$	$8.59_{\pm 0.79}$	261.6

Tabla 3.3: Tasas de error a Nivel Palabra (WER) obtenidas en la tarea de ASR en SUPERB con el conjunto test-clean de LibriSpeech. Reportamos métricas con y sin modelo de lenguaje (LM) y con intervalo de confianza ($\alpha = 0.05$) obtenido mediante bootstrapping.

explicar por qué se desempeñan mejor en tareas de habla en comparación con modelos entrenados con patches rectangulares. A su vez, atribuimos el buen desempeño en las tareas de eventos acústicos, a pesar del uso de cuadros, a nuestra definición de la señal objetivo. Se puede observar que nuestro modelo que utiliza EnCodec como señal objetivo (fila 4) logra un desempeño mayor a BEATs cuando utiliza cuantizadores aleatorios (fila 13) en la tarea FSD. Ambos modelos se benefician del refinamiento iterativo (ver fila 4 vs 9 y fila 13 vs 14).

Fuse HuBERT y Fuse Wav2Vec2 son dos representaciones encontradas en el leaderboard de HearEval. Ambas generan representaciones promediando las activaciones de todas las capas de HuBERT XL [76] y Wav2Vec2 Large [75], respectivamente. Estos modelos son pre-entrenados exclusivamente con datos de habla provenientes de LibriLight con un foco puesto en aprender representaciones para ASR. Nuestro modelo base entrenado con LibriLight se desempeña mejor en todas las tareas que no son de habla, con excepción de ESC, a pesar de estar ambos modelos entrenados solo con habla (fila 7 vs 18 y 19). Esto puede deberse a que la señal objetivo, EnCodec, que se preentrenó no solo con habla sino que también con datasets de eventos acústicos y de música, podría indirectamente facilitar el modelado de esos dominios. Nuestros modelos base (filas 3 y 8) lograron un desempeño similar a Fuse HuBERT y Wav2Vec2 en las tareas de habla, a pesar de estar entrenados en menos datos y poseer muchos menos parámetros. A su vez, nuestro modelo Large con refinamiento iterativo (fila 11) logra superar a estos modelos en todas las tareas, incluyendo las de habla, a pesar de ser un modelo de tamaño menor y más generalista.

Dado el desempeño de nuestros modelos en tareas de habla, decidimos también realizar una evaluación en la tarea de ASR siguiendo el protocolo de SUPERB. Este consiste en entrenar una red BiLSTM con dos capas de 1024 neuronas optimizando la función de pérdida CTC [204] y utilizando el subconjunto de 100 horas limpias de LibriSpeech. Esta red toma como entrada las salidas de las 10 capas de transformer

de EnCodecMAE, y aprende un promedio pesado de estas 10 secuencias resultando en una única secuencia temporal de embeddings. Para acelerar el entrenamiento, hicimos un promedio local tomando 2 embeddings consecutivos y reduciendo el largo de secuencia a la mitad. Para decodificar las secuencias transcriptas, utilizamos el modelo de lenguaje 4-gramas oficial de LibriSpeech [205]. Cabe destacar que a diferencia de las otras tareas, la tarea de ASR nos permite realizar una evaluación de la representación a nivel secuencia, antes de promediarla en el tiempo. Los resultados en la Tabla 3.3 indican que nuestros modelos aún poseen un desempeño menor a modelos de SSL recientes como HuBERT y Wav2Vec 2.0, pero son comparables con modelos de SSL previos como DeCOAR 2.0 y Wav2Vec. Al mismo tiempo, no conocemos de otros trabajos que aprendan representaciones generales de audio y reporten el desempeño en la tarea de ASR. Uno podría pensar que la tarea de reconocimento de comandos de voz (SC) es una aproximación de la de ASR, sin embargo, son muy distintas. En el caso de SC basta con discriminar un conjunto acotado de comandos de un conjunto acotado de 'palabras negativas'. La dificultad del problema quedará determinada principalmente por cuán parecidos sean los comandos entre si y con las demás palabras que no son comandos (ejemplos negativos). Si el conjunto de comandos y ejemplos negativos es muy acotado, incluso aspectos como la duración de la palabra, cantidad de sílabas o un reconocimiento básico de vocales podrían ser suficientes para discriminar los comandos, lo cual resulta en una tarea más fácil que la de transcripción. A su vez, en la tarea de SC cada instancia contiene una única palabra, lo cual facilita la tarea respecto a reconocimiento de habla ya que no es necesario aprender a segmentar palabras.

3.5. Evaluación completa en HEAREval

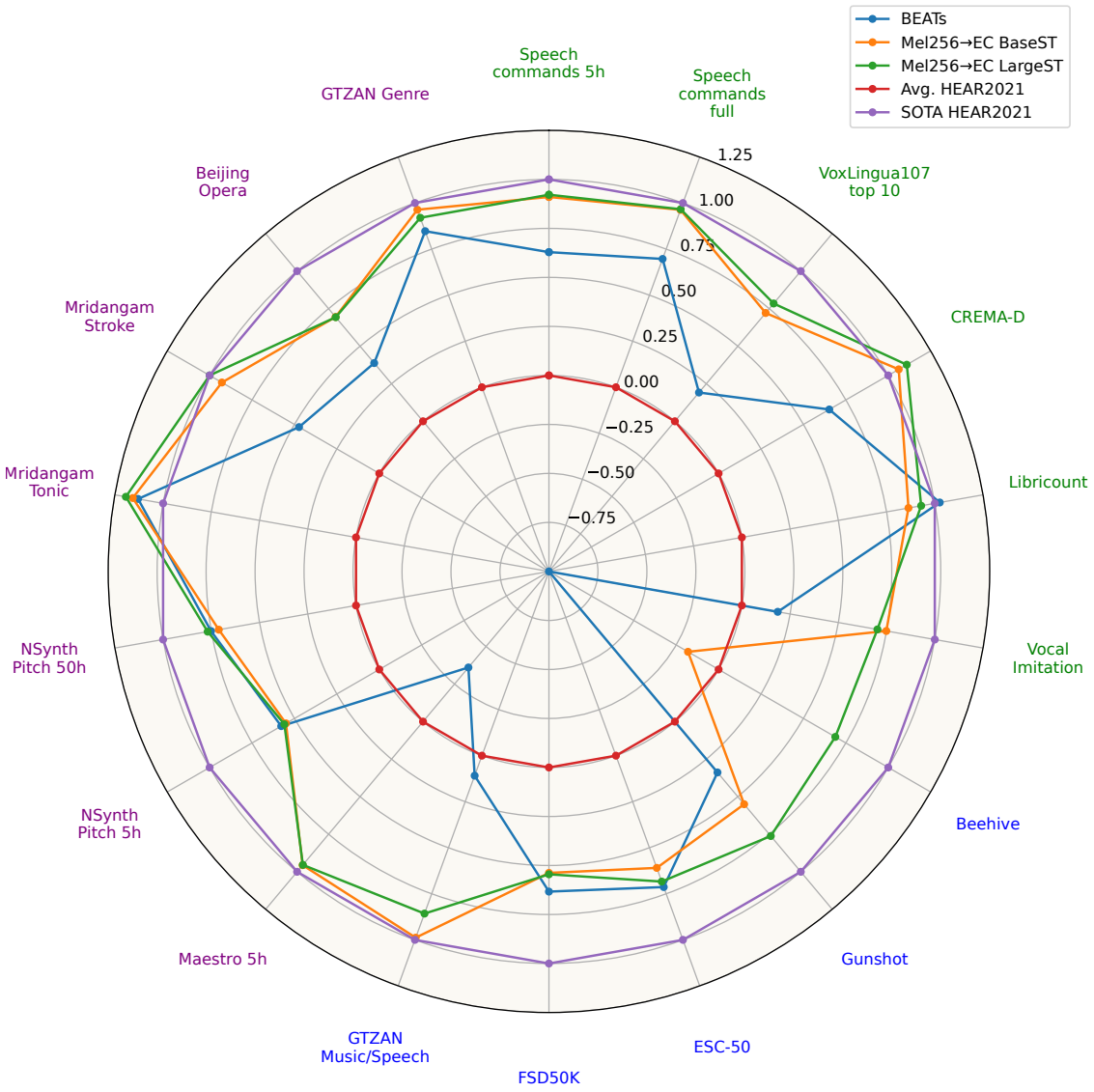

Figura 3.5: Desempeño obtenido por dos variantes de EnCodecMAE y BEATs (iter 3) comparado con el desempeño promedio y máximo obtenido en cada tarea de HEAREval 2021. Se muestra el desempeño centrado en la media y escalado por el máximo. Se utilizan etiquetas violetas para las tareas de música, verdes para las de habla, y azul para las de sonidos ambientales.

Hasta este momento trabajamos con un subconjunto de 6 tareas de HEAREval, sin embargo el benchmark posee 19 tareas distintas. En la Figura 3.5 se puede observar un gráfico resumen del desempeño de los mejores modelos de EnCodecMAE, BEATs (iteración 3), el estado del arte en HEAREval 2021 para cada tarea, y el desempeño promedio en cada tarea. Elegimos compararnos con BEATs ya que es el modelo externo que logró un mejor desempeño global en las 6 tareas analizadas

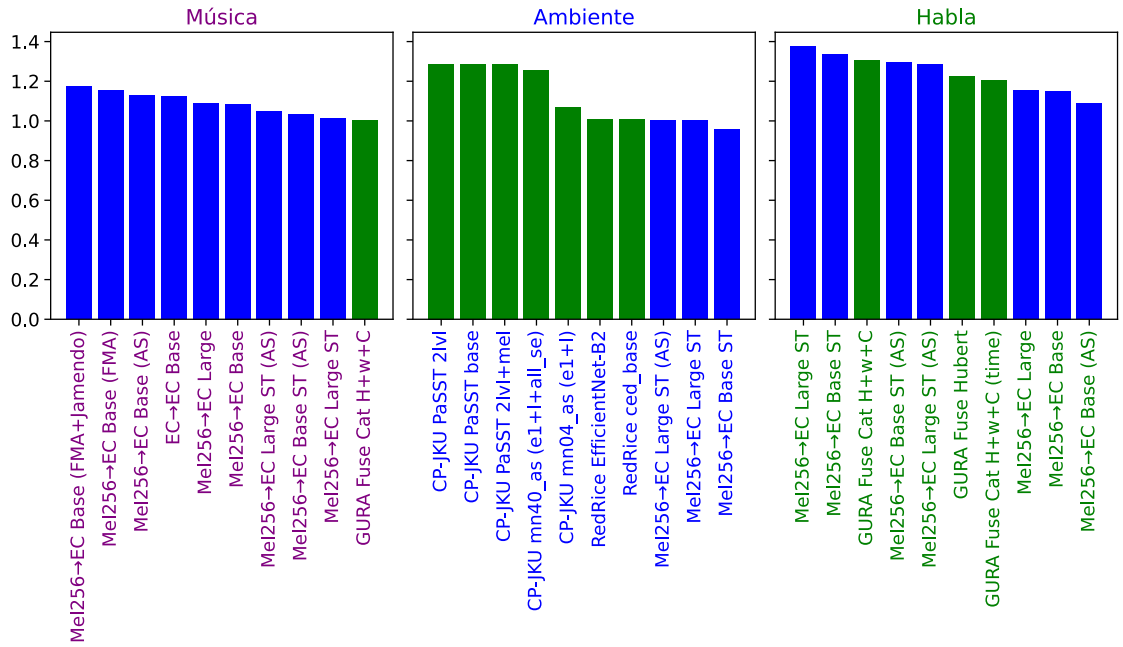

Figura 3.6: Z-Scores promedios obtenidos en las tareas de HEAREval en cada dominio. Las barras azules corresponden a modelos propios, mientras que las verdes a modelos externos. Los colores de las etiquetas reflejan el dominio y coinciden con la paleta utilizada en la Figura 3.5.

previamente (ver Tabla 3.2). El puntaje s^* mostrado para cada tarea surge de la siguiente operación:

$$\mathrm{s}^*(\mathrm{s}) = \frac{s - \overline{s}}{s_{max} - \overline{s}},$$

donde $\overline{s}$ es el desempeño promedio y s_{max} es el mejor desempeño en el leaderboard para esa tarea. Este valor tiene las propiedades de que un desempeño igual al promedio en HEAREval 2021 mapea a 0: $s^*(\overline{s}) = 0$, y un desempeño igual al máximo en HEAREval 2021 mapea a 1: $s^*(s_{max}) = 1$. En el anexo A se muestra una tabla completa con las métricas sin normalizar para cada tarea, y una lista más exhaustiva de modelos upstream, junto con una descripción breve de cada tarea del benchmark.

En la Figura 3.5 se puede ver que los modelos de EnCodecMAE mostrados, en general, se desempeñan por encima del promedio en todas las tareas. Particularmente, la tarea de Beehive, donde se observa mayor variabilidad de desempeño, es considerada en varios trabajos muy ruidosa, y en algunos descartada [165, 193]. Es esta tarea en la que nuestro modelo Base con refinamiento iterativo se desempeña peor que el promedio. En el resto de las tareas, ambos modelos obtienen resultados cercanos o superadores del estado del arte en HEAREval. Específicamente, nuestros modelos superan al estado del arte en la tarea Mridangam Tonic, la cual consiste en clasificar el tono de un mridangam (instrumento percusivo tonal de la India) y en CREMA-D, la cual es de reconocimiento de emociones. Por otro lado, BEATs posee un desempeño inferior a EnCodecMAE en todas las tareas exceptuando FSD50K, ESC50

y Libricount. Particularmente, en Maestro 5h, BEATs obtiene un desempeño inferior al promedio. Creemos que esto se debe a que esta tarea de transcripción musical tiene etiquetas en el tiempo. Como se discutió previamente, el uso de patches rectangulares como los usados en BEATs conlleva una pérdida de resolución temporal, lo cual es perjudicial en tareas donde se realiza una clasificación a nivel de ventanas en lugar de a nivel de instancia.

En la Figura 3.6 se puede observar los 10 mejores modelos en cada dominio. Los puntajes reportados en la Figura 3.6 de cada modelo en los distintos dominios, se calculan obteniendo z-scores del desempeño en cada tarea y promediándolos entre las tareas pertenecientes a cada dominio. Este puntaje es muy similar al desempeño global reportado en la Tabla 3.2, con la diferencia que no limitamos los z-scores a valores en el intervalo [-1,1]. En el anexo A se muestra a qué dominio corresponde cada tarea, y también la Figura 3.5 indica esto con los colores de las etiquetas.

Se puede observar que en las tareas musicales los modelos de EnCodecMAE son los que dominan, siendo aquellos entrenados sólo en datos de música los que mejor desempeño alcanzan. También se puede ver que el refinamiento iterativo no parece ser beneficioso en este dominio, probablemente porque algunas de las tareas están relacionadas a la estimación de pitch, y vimos que el refinamiento iterativo no beneficiaba en esta tarea. El único modelo externo que llega al top-10 en estas tareas consiste de una combinación de Hubert, Wav2Vec 2.0 y Crepe, siendo esta última representación la que probablemente le da una ventaja en el dominio musical, al incorporar información de pitch.

En las tareas relacionadas a sonidos ambientes, los modelos entrenados de forma supervisada en detección de eventos acústicos son los que mejor desempeño alcanzan. Esto se debe a que al menos dos tareas, ESC-50 y FSD50K, están muy alineadas con la tarea de detección de eventos acústicos con Audioset, solapándose en el espacio de etiquetas. Los modelos de EnCodecMAE son los que mejor desempeño logran sin uso de supervisión. En este dominio, a diferencia de música, los modelos de EnCodecMAE con refinamiento iterativo de la señal objetivo son los que mejor desempeño muestran.

Por último, en el dominio del habla, los modelos de EnCodecMAE entrenados en la mezcla de datos y con refinamiento iterativo de la señal objetivo son los que mejor desempeño muestran, superando a todos los modelos externos. Resulta interesante que las variantes de EnCodecMAE entrenadas solo con habla (LibriLight) no se encuentran en el top 10, y son superadas por las variantes entrenadas solo con Audioset. Creemos que la presencia de ruido en los datos de pre-entrenamiento puede ser importante para lograr un buen desempeño en algunas tareas de habla, y Audioset contiene muchos audios con habla y ruidos de fondo, mientras que Librilight es habla sin otras fuentes solapándose. Los modelos externos en el top 10 en habla son aquellos que incorporan modelos de representaciones de habla, como Hubert y Wav2Vec 2.0.

Retomando el objetivo central propuesto en esta tesis, que consistía en desarrollar un modelo cuyas representaciones puedan ser utilizadas para resolver tareas diversas de audio, abarcando distintos dominios como el habla, música y sonidos ambientales, podemos afirmar que lo hemos cumplido. Nuestro modelo Mel256→EC Large ST es el único que logra

estar en el top-10 de los 3 dominios, y como vimos en la Figura 3.5, alcanza un desempeño mejor al promedio y cercano al estado del arte en HEAREval para todas las tareas, a pesar de ser estas muy diversas y pertenecer a distintos dominios. A su vez, en la Figura 3.6 se puede observar que los modelos de EnCodecMAE, en azul, están en el top-1 si solo consideramos modelos pre-entrenados sin supervisión.

3.6. Exploración de Downstreams alternativos

En esta sección exploraremos diferentes modelos downstream para determinar su influencia a la hora de evaluar modelos. En particular queremos ver cómo interactúa la elección del modelo downstream con el modelo upstream y cada tarea en términos del desempeño obtenido.

KNN

Como mencionamos en la sección 3.2, utilizar KNN como modelo downstream es una práctica común y nos puede dar una idea de cómo está organizado localmente el espacio de la representación. La metodología es análoga a HearEval, intercambiando el MLP por un modelo de KNN, y eligiendo el valor de K en los conjuntos de validación. En el caso de FSD que es una tarea con múltiples etiquetas, utilizamos una variante multi-etiqueta de KNN [206]. El hiperparámetro más importante a determinar al usar KNN como downstream es el valor de K. Este determina cuántos vecinos se tendrán en cuenta para determinar la clase a la que pertenece una instancia. En la Figura 3.7 se puede observar que el valor óptimo de K depende principalmente de la tarea, y que el cambio de upstream solo desplaza la curva en el eje del desempeño sin alterar mucho su forma. Algunas tareas como NS, ESC y GC poseen un mayor desempeño cuando K toma valores chicos, mientras que FSD y ER parecen beneficiarse de K más grandes. En general, BEATs, Dasheng y EnCodecMAE parecen organizar sus representaciones en base a la frecuencia fundamental, dado que poseen una precisión mucho mayor en la tarea de clasificación de notas musicales (NS) que los demás modelos. Esto implica que dado un audio de un instrumento tocando una nota musical, los audios más cercanos contendrán la misma nota. Esto no parece ocurrir en representaciones de habla (Fuse HuBERT y Fuse Wav2Vec2), ni en BYOL-A, cuyo desempeño es significativamente menor en esta tarea. Los modelos de habla poseen un desempeño superior en la tarea de reconocimiento de comandos de voz (SC), sugiriendo que su espacio de representaciones podría estar organizado principalmente en base al contenido fonético. Si bien reconocimiento de emociones (ER) es una tarea de habla, EnCodecMAE es el modelo con mejor desempeño en ella. Hipotetizamos que aunque el contenido fonético es importante, el acceso a la frecuencia fundamental es relevante y puede ser el factor que le permite superar a Fuse HuBERT y Fuse Wav2Vec2.

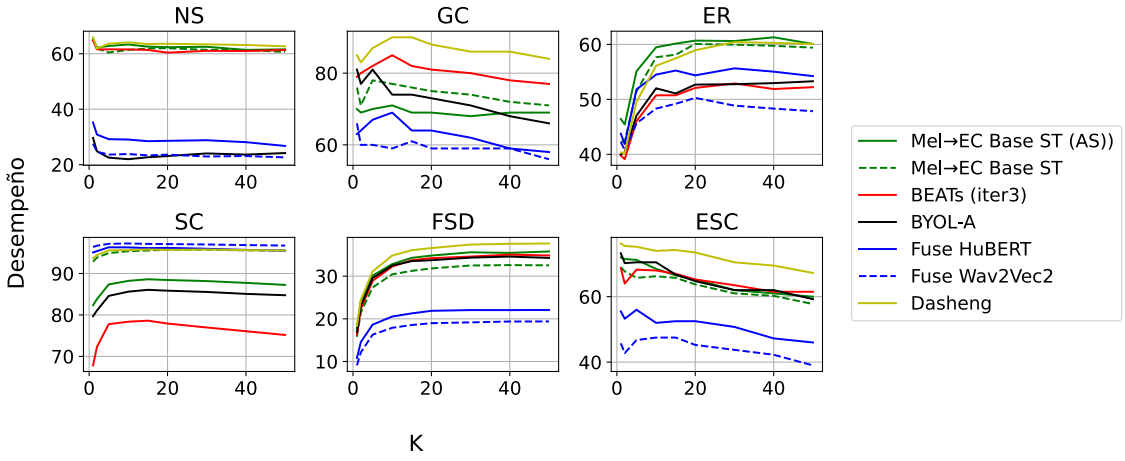

Figura 3.7: Efecto del K en el desempeño en validación para distintos modelos upstream.

	Music		Speech		Env	
	NS	GC	SC	ER	FSD	ESC
(1) BEATs (Iter 3)	65.5±1.2	84.3±3.5	53.0±0.7	53.1±1.4	32.8±1.1	71.5±1.7
(2) BYOL-A	29.7±1.4	78.6±4.1	68.9±1.1	53.2±1.2	31.3±1.0	70.8±2.0
(3) Dasheng	**66.9±1.0**	**87.6±1.8**	93.8±0.5	**60.6±0.6**	**34.5±1.1**	**77.8±1.6**
(4) Fuse HuBERT	35.0±1.3	68.4±2.3	94.0±0.5	55.8±0.7	20.1±0.9	55.8±1.9
(5) Fuse Wav2Vec2	26.9±0.9	65.5±2.4	**95.2±0.5**	50.6±1.4	18.1±1.2	48.0±2.1
(6) Mel→EC Base + ST	**66.1±1.5**	78.2±2.0	91.0±0.7	58.9±0.6	29.9±1.1	70.0±1.4
(7) AS Only	65.5±1.3	75.0±2.5	74.2±1.1	**61.0±1.0**	32.8±1.0	73.5±1.4

Tabla 3.4: Desempeño de distintos modelos en un subconjunto de HearEval utilizando KNN como modelo downstream. Las desviaciones reportadas muestran la diferencia entre el valor de métrica obtenida y el intervalo de confianza más lejano a ella al realizar bootstrapping.

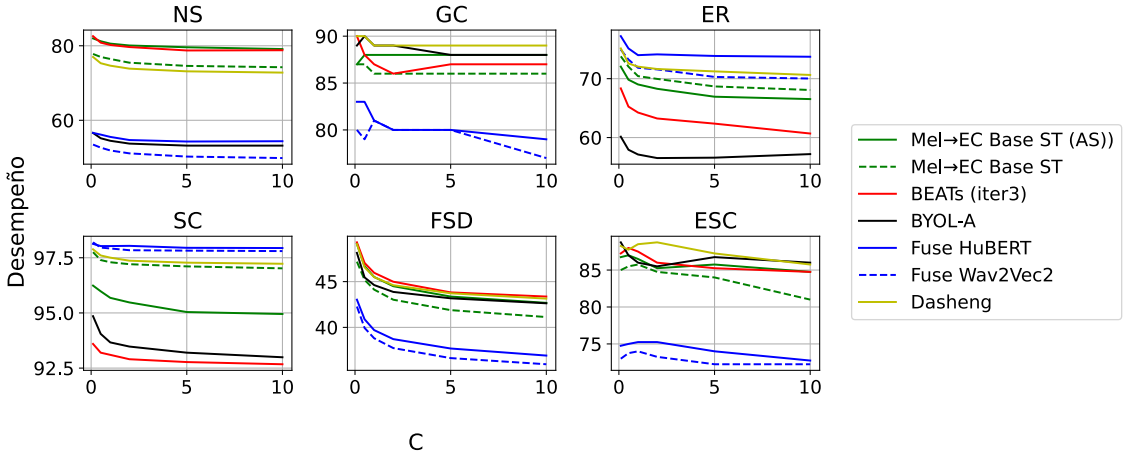

Figura 3.8: Efecto del C en el desempeño en validación para distintos modelos upstream.

Regresión Logística

	Music		Speech		Env	
	NS	GC	SC	ER	FSD	ESC
(1) BEATs (Iter 3)	**82.5**$_{\pm 0.9}$	**88.2**$_{\pm 2.0}$	89.3$_{\pm 0.8}$	69.4$_{\pm 0.8}$	**48.6**$_{\pm 1.2}$	**88.7**$_{\pm 0.8}$
(2) BYOL-A	56.1$_{\pm 1.5}$	**88.1**$_{\pm 2.7}$	90.8$_{\pm 0.5}$	60.1$_{\pm 0.8}$	45.3$_{\pm 0.8}$	87.0$_{\pm 0.9}$
(3) Dasheng	78.4$_{\pm 1.2}$	**89.7**$_{\pm 2.1}$	**97.2**$_{\pm 0.4}$	**77.0**$_{\pm 0.9}$	46.7$_{\pm 0.8}$	**88.0**$_{\pm 0.5}$
(4) Fuse HuBERT	57.7$_{\pm 1.6}$	82.7$_{\pm 2.0}$	**97.3**$_{\pm 0.4}$	**77.1**$_{\pm 0.5}$	40.7$_{\pm 0.9}$	76.8$_{\pm 1.7}$
(5) Fuse Wav2Vec2	53.8$_{\pm 2.0}$	81.1$_{\pm 2.2}$	**97.2**$_{\pm 0.3}$	74.5$_{\pm 0.7}$	39.5$_{\pm 1.0}$	74.7$_{\pm 1.2}$
(6) Mel→EC Base + ST	77.4$_{\pm 1.3}$	**88.2**$_{\pm 2.4}$	96.8$_{\pm 0.6}$	75.5$_{\pm 0.7}$	44.7$_{\pm 1.1}$	85.2$_{\pm 0.9}$
(6) AS Only	**82.6**$_{\pm 1.1}$	**89.0**$_{\pm 2.2}$	93.7$_{\pm 0.5}$	73.9$_{\pm 0.7}$	47.0$_{\pm 1.0}$	**88.0**$_{\pm 1.0}$

Tabla 3.5: Desempeño de distintos modelos en un subconjunto de HearEval utilizando regresión logística como modelo downstream. Las desviaciones reportadas muestran la diferencia entre el valor de métrica obtenida y el intervalo de confianza más lejano a ella al realizar bootstrapping.

Otro modelo downstream comúnmente utilizado es el lineal. A diferencia de KNN, este modelo puede aprender a utilizar un subconjunto de las dimensiones de la representación, ignorando direcciones no informativas para la tarea. Además, no es capaz de capturar relaciones no lineales entre las variables de entrada y salida. Nuevamente, respetamos la metodología de HearEval y reemplazamos el downstream por un modelo de regresión logística implementado mediante scikit-learn. Utilizamos regularización L_2 y exploramos el valor de C, el cual equivale a $1/\lambda$, en donde λ es el peso de regularización. En la Figura 3.8 se puede observar el efecto de este hiperparámetro. Se puede ver que un valor bajo de C es beneficioso en todas las tareas y modelos. Esto indica que una regularización fuerte ayuda en las tareas exploradas.

	Music		Speech		Env	
	NS	GC	SC	ER	FSD	ESC
(1) BEATs (Iter 3)	**84.9**$_{\pm 1.2}$	87.3$_{\pm 2.5}$	90.6$_{\pm 0.9}$	66.8$_{\pm 1.2}$	**54.1**$_{\pm 1.1}$	83.0$_{\pm 1.6}$
(2) BYOL-A	80.3$_{\pm 1.4}$	85.4$_{\pm 2.5}$	93.3$_{\pm 0.7}$	66.6$_{\pm 1.2}$	52.2$_{\pm 1.0}$	**83.1**$_{\pm 1.6}$
(3) Dasheng	**85.3**$_{\pm 1.0}$	**88.2**$_{\pm 2.1}$	**97.6**$_{\pm 0.5}$	**78.1**$_{\pm 0.9}$	52.0$_{\pm 1.1}$	81.4$_{\pm 1.9}$
(4) HuBERT XL	78.2$_{\pm 1.4}$	80.2$_{\pm 3.0}$	96.1$_{\pm 0.5}$	74.2$_{\pm 0.9}$	42.1$_{\pm 1.2}$	74.0$_{\pm 1.7}$
(5) Wav2Vec 2.0 L	76.1$_{\pm 1.4}$	78.7$_{\pm 3.3}$	96.5$_{\pm 0.5}$	69.5$_{\pm 1.0}$	42.8$_{\pm 1.1}$	70.1$_{\pm 2.0}$
(6) Mel→EC Base + ST	**84.7**$_{\pm 1.0}$	**89.3**$_{\pm 2.3}$	97.0$_{\pm 0.6}$	75.5$_{\pm 1.0}$	51.1$_{\pm 1.2}$	82.9$_{\pm 1.5}$
(7) AS Only	**85.8**$_{\pm 1.2}$	**89.6**$_{\pm 2.2}$	95.5$_{\pm 0.6}$	74.8$_{\pm 0.9}$	**53.6**$_{\pm 1.1}$	**84.9**$_{\pm 1.8}$

Tabla 3.6: Desempeño de distintos modelos en un subconjunto de HearEval introduciendo un promedio pesado entrenable en el modelo downstream de HearEval. Las desviaciones reportadas muestran la diferencia entre el valor de métrica obtenida y el intervalo de confianza más lejano a ella al realizar bootstrapping.

Promedio pesado de capas

La evaluación de HEAREval utiliza una representación fija durante su evaluación, la cual se suele extraer de la última capa del modelo, o de una concatenación o promedio de representaciones de distintas capas. Proponemos expandir HEAREval utilizando un promedio pesado aprendido de representaciones. Esto permite al modelo downstream utilizar representaciones de capas intermedias y combinarlas. En este downstream, la representación de entrada es:

$$x = \frac{\sum_i |\lambda_i| x_i}{\sum_i |\lambda_i|},$$

en donde x_i son las activaciones de la capa i, y λ_i son los pesos asociados, los cuales son aprendidos durante el entrenamiento del downstream y se inicializan con unos, lo cual equivale a un promedio de todas las activaciones.

Este tipo de downstream fue utilizado previamente en NLP por ELMO [207], y luego en el dominio del habla en nuestro trabajo en reconocimiento de emociones [7] y en SUPERB [5].

En la Tabla 3.6 se pueden observar los resultados obtenidos utilizando este promedio pesado de capas, los cuales se compararán contra distintos downstream en la siguiente subsección. Por otro lado, en la Figura 3.9 se pueden observar los pesos asignados a cada una de las capas de cada modelo. Dado que cada uno de los modelos analizados posee una cantidad de capas distinta, normalizamos el eje de abscisas para que tenga un rango único de 0 a 1, donde $1/L$ es la primera capa y 1 la última, con L la cantidad de capas del modelo. Se puede ver que para algunas tareas y modelos upstream, el modelo downstream asigna distintos pesos a las capas. Por ejemplo, se puede observar en general que las primeras y última capa reciben un mayor peso en la tarea de clasificación de notas musicales (NS), mientras que en la tarea de clasificación de comandos de voz (SC), los modelos generales de audio dan un mayor peso a las últimas capas, y los modelos de habla a capas intermedias cercanas a la entrada.

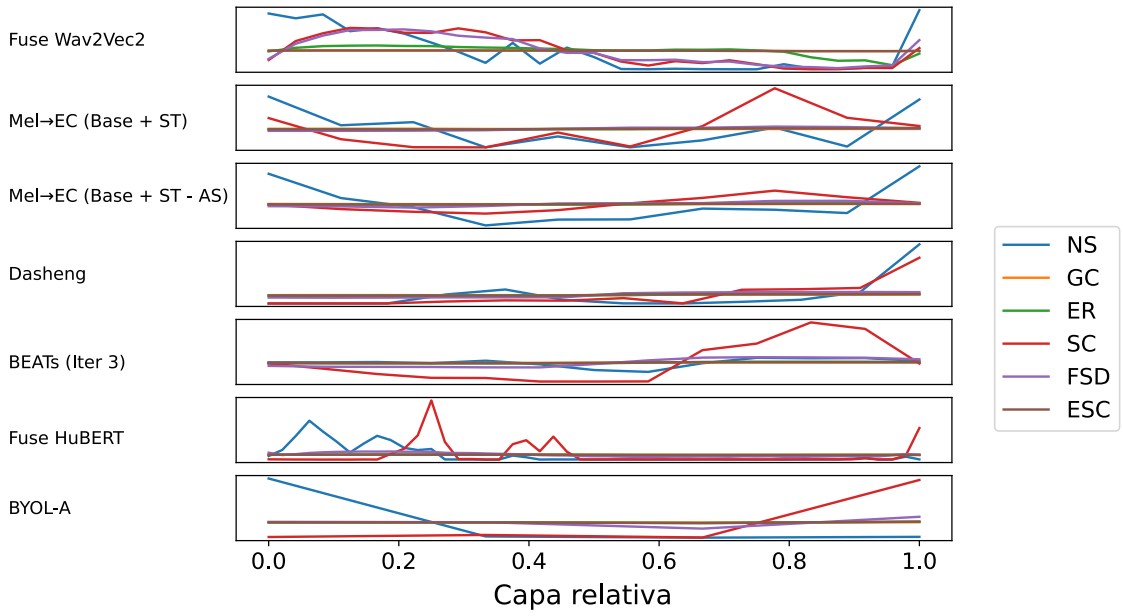

Figura 3.9: Pesos aprendidos para las distintas tareas y modelos de audio utilizados.

Comparación de Downstreams

Lo primero que podemos observar es que utilizar regresión lineal lleva a un mayor desempeño en general que utilizar KNN como modelo downstream. Esto se observa en la Figura 3.10 (Izquierda) en donde todos los puntos, correspondientes a un modelo y tarea, se encuentran por encima de la identidad. Esto implica que el puntaje del modelo lineal es superior al de KNN para todas las tareas y modelos analizados. Si bien KNN puede dar otras ventajas como la posibilidad de analizar qué vecinos son los más cercanos a una instancia en una representación, a la hora de resolver un problema de interés, resulta más efectivo utilizar un modelo lineal. Esto puede deberse a que en KNN se tiene en cuenta la representación completa, y las componentes de mayor varianza van a dominar en la búsqueda de vecinos, mientras que un regresor lineal, especialmente si se utiliza regularización, es capaz de encontrar un subespacio de la representación que sea útil para la tarea a resolver. A su vez, podemos observar en la Figura 3.10 (Centro), que en general el modelo downstream utilizado por HEAREval (el cual puede ser un perceptrón multicapa), obtiene un mejor desempeño que el modelo lineal. Las excepciones ocurren con las tareas cuyos conjuntos de datos son más pequeños (ESC y GC). En esos casos, el modelo lineal supera a los modelos de HEAREval, posiblemente gracias a la regularización que promueve modelos más sencillos con menor sobreajuste. Por último, se puede observar en la Figura 3.10 (Derecha), que introducir el promedio pesado de capas en los modelos downstream de HEAREval da resultados similares al downstream original de HEAREval. La principal diferencia se observa en la tareas de clasificación de notas musicales (NS), la cual se ve beneficiada al introducir el promedio pesado de capas. Como habíamos visto en la Figura 4.2, para EnCodecMAE, las representaciones de la última capa

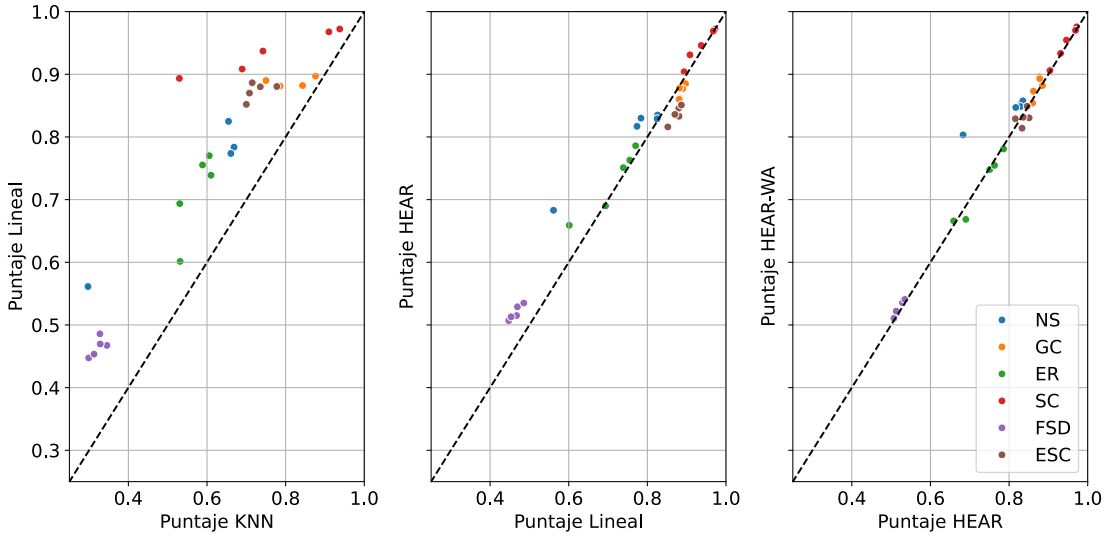

Figura 3.10: Puntajes obtenidos para cada tarea y upstream utilizando distintos modelos downstreams. Se muestran distintas combinaciones de modelos downstream con sus puntajes correspondientes. Si un punto cae en la identidad, significa que para ese modelo y tarea el desempeño es el mismo en los dos downstreams comparados. HEAR-WA se refiere a la utilización de un promedio pesado de capas.

obtiene un buen desempeño en la mayoría de las tareas analizadas, lo cual explica que no haya una diferencia notable entre ambos modelos downstream. Sin embargo, para el caso de NS, se podía observar que las primeras capas llevaban a un mejor desempeño que las últimas. Esto tiene sentido ya que la frecuencia fundamental se puede pensar como un atributo de bajo nivel que debería ser capturado en las primeras etapas de procesamiento de la red neuronal. Creemos que el promedio pesado de capas puede dar una ventaja en estos casos en los que la última capa puede no ser la óptima, o que las primeras capas pueden complementar a las últimas. En ese sentido, la ventaja es que el modelo puede aprender a seleccionar o combinar las representaciones que le sean más útiles para resolver la tarea. La desventaja puede darse cuando no se tienen suficientes datos para aprender esta selección, o cuando la última capa era la mejor representación y los pesos no nulos de las demás capas añaden ruido.

4

Estudios de ablación y variantes

4.1. Efecto de la representación de entrada

Se pueden utilizar distintos extractores de atributos para transformar la señal cruda de audio x en una secuencia con menor resolución temporal y más apropiada para ser procesada por un transformer. En la literatura, como se puede ver en la Tabla 2.1, la representación de entrada varía dependiendo del trabajo, y en general no existe un análisis de qué representación de entrada lleva a las mejores representaciones de salida. Es por esto que decidimos explorar el impacto que tiene este bloque en el desempeño de EnCodecMAE en las distintas tareas de HEAREval.

Los distintos tipos de representación que exploramos fueron:

- Encoder de Encodec: esta representación debería ser la más alineada con las etiquetas a predecir, ya que consisten en las mismas antes de cuantizar. Esto hace trivial la tarea de reconstruir las zonas no enmascaradas, aunque sigue siendo desafiante para el modelo aprender a reconstruir las partes enmascaradas. Al mismo tiempo, es una representación con poca pérdida de información respecto al audio crudo, ya que es posible reconstruir el mismo con alta calidad a partir de esta representación. Como se explicó en la sección 3.1.1, su salida consiste de 75 embeddings por segundo con 128 dimensiones.
- Magnitud de Espectrograma Lineal: esta representación es muy utilizada en modelos de audio y resulta una manera natural de representar audios que está alineada con el sonido como fenómeno físico al descomponerlo en sinusoidales con distintas frecuencias y fases y medir su energía. Como se mencionó en la sección 2.1.1, es posible recuperar la señal original a partir de un espectrograma completo (magnitud y fase). Sin embargo, al descartarse la información de fase, podrían haber detalles contenidos en las etiquetas de EnCodec que no se encuentran disponibles en la magnitud del espectrograma.
- Melspectrograma: esta representación, explicada en la Sección 2.1.1, es comúnmente utilizada como entrada a modelos fundacionales, como por ejemplo, BEATs [152], EAT [143], BYOL-A [138], AudioMAE [160] y BestRQ [151] (se pueden ver más ejemplos en la Tabla 2.1). Al utilizar una escala de frecuencias más alineada con la percepción, descarta información que estaría presente en un espectrograma pero que probablemente no sería percibida por una persona.

	Music		Speech		Env		
	NS	GC	SC	ER	FSD	ESC	Global
(1) EnCodec	63.9±1.0	65.9±5.3	33.7±1.1	45.9±0.7	20.2±0.8	39.9±2.5	-76.2
(2) Mel128	77.8±1.9	64.0±2.4	25.1±0.8	46.7±0.9	14.9±0.6	32.9±2.2	-68.6
(3) Mel256	77.8±1.0	66.2±1.9	23.8±0.4	46.1±0.7	15.2±0.6	32.9±1.8	-66.1
(4) Spec	72.4±1.4	66.0±3.5	17.1±0.4	40.0±1.0	14.6±0.7	32.6±1.4	-70.9
(5) CNN	6.1±1.0	75.8±2.9	34.7±1.0	17.5±0.6	30.5±0.7	3.0±0.2	-69.9
(6) EC→EC (R)	72.7±1.4	73.6±1.7	57.7±1.3	45.3±1.1	21.6±0.6	40.6±1.8	-51.1
(7) Mel128→EC (R)	77.6±1.7	66.9±2.4	61.4±1.0	48.5±1.2	18.1±0.5	35.6±2.0	-54.2
(8) Mel256→EC (R)	77.2±1.3	66.2±2.2	57.5±1.3	50.6±1.2	18.5±0.8	36.7±1.4	-55.5
(9) Spec→EC (R)	75.8±1.1	66.7±2.7	56.7±1.0	49.6±1.1	18.0±0.6	35.5±1.7	-57.8
(10) CNN→EC (R)	78.4±1.7	75.8±2.7	65.4±1.1	57.5±1.3	26.8±0.8	49.3±2.6	-8.8
(11) EC→EC	**91.7±1.1**	85.6±1.5	92.2±1.3	72.0±0.8	44.1±1.0	74.6±3.3	83.4
(12) Mel128→EC	84.6±1.4	**86.8±1.2**	**96.1±0.6**	**76.5±0.8**	**49.4±0.7**	**79.0±3.0**	95.3
(13) Mel256→EC	84.6±1.2	**87.6±1.0**	**96.3±0.5**	75.5±0.8	**49.5±0.7**	**79.8±2.2**	**95.9**
(14) Spec→EC	84.5±1.1	86.5±1.4	**96.0±0.3**	**76.2±1.1**	**49.1±0.7**	**78.8±2.8**	94.6
(15) CNN→EC	85.5±1.3	**87.5±2.1**	94.4±0.7	75.2±0.8	48.0±0.9	**78.7±3.0**	92.6

Tabla 4.1: Desempeño de distintas representaciones de entrada, utilizadas sin alterar (1-5), tras ser transformadas mediante EnCodecMAE con pesos aleatorios (6-10), y tras ser utilizadas como entrada en EnCodecMAE tras su pre-entrenamiento (11-15).

- CNN: debido a que muchos trabajos en el campo [76, 130, 141] hacen uso del encoder convolucional de Wav2Vec 2.0, es interesante analizar cuán importante es esta decisión de diseño y cuánto influencia en el desempeño final. Este encoder consiste de 7 capas convolucionales con layer normalization y activación GeLU. Cada convolución posee 512 canales, paso de 5 en la primer capa y 2 en el resto, y un tamaño de kernel de 10 para la primer capa, 3 para las siguientes 4, y 2 para las restantes. Debido a los pasos, el encoder reduce la longitud de X_a en un factor de 320 mientras que aumenta la dimensión de las características de 1 a 512.

Cabe notar que entrenar el encoder de Wav2Vec 2.0 implica la introducción de nuevos parámetros en el modelo y el tiempo de entrenamiento pasa de 5 a 8 días.

En la Tabla 4.1 se puede observar que las representaciones expertas (melspectrogramas y espectrogramas) usadas directamente como entrada al modelo downstream, sin la etapa de EnCodecMAE, se desempeñan muy por debajo de la media de HEAREval, siendo los melspectrogramas con 256 bins los que mejor puntaje global obtienen (fila 3). El modelo CNN (fila 5) surge tras entrenar EnCodecMAE utilizando audio crudo como entrada, el cual es procesado por la CNN de Wav2Vec 2.0. Para estos resultados se extraen las activaciones de la salida de la CNN ya entrenada. Para EnCodec (fila 1), se extraen las activaciones a la salida del encoder, antes de cuantizarlas.

En las filas 6 a 10, se puede observar el desempeño de las distintas variantes de EnCodecMAE en su inicialización (es decir, con pesos aleatorios en el encoder y decoder). Interesantemente, incluso sin entrenar al modelo, el desempeño mejora por

sobre utilizar directamente las representaciones de entrada. Esto implica que hay un sesgo en el modelo, probablemente en las capas de transformers, que beneficia a las representaciones, incluso si se trata de proyecciones aleatorias de las mismas. Este fenómeno en el que redes neuronales sin entrenar poseen un mejor desempeño que utilizar los atributos de entrada, se ha observado en trabajos previos, tanto en el dominio de audio [208] como en otros dominios [209, 210]. Resulta particularmente interesante el caso en donde se utiliza la salida de la CNN como representación de entrada (fila 10), ya que el modelo con pesos aleatorios es capaz de alcanzar un desempeño global similar al promedio en HEAREval, y la mejora en algunas tareas, como por ejemplo reconocimiento de comandos de voz (SC), y reconocimiento de emociones en el habla (ER) es notable.

Finalmente, como era esperable, entrenar el modelo en la tarea de pretexto de modelado de lenguaje enmascarado, genera representaciones muy superiores a los modelos con pesos aleatorios o utilizar directamente las representaciones de entrada. Puede observarse que el mejor modelo en términos de desempeño global es el que utiliza como entrada melspectrogramas con 256 bins (fila 13). Este modelo obtiene el mejor o cerca del mejor desempeño en todas las tareas, excepto la de clasificación de notas musicales (NS). En esta tarea, el modelo que utiliza a las representaciones de EnCodec como entrada (fila 11) es el que mejor desempeño alcanza. Una hipótesis posible es que la resolución de pitch necesaria para clasificar notas musicales no está presente en los melspectrogramas de entrada, pero sí lo está en EnCodec, que fue entrenado para reconstruir audio general, incluyendo música. Sin embargo, tanto los espectrogramas lineales como la CNN podrían contener esta información pero su desempeño es inferior al obtenido con EnCodec.

Una observación importante de hacer es que el desempeño de la representación de entrada por si sola, no guarda una relación con el desempeño obtenido tras entrenar EnCodecMAE. Por ejemplo, EnCodec (fila 1) obtiene 63.9% de precisión en NSynth, que es un resultado peor que el obtenido con melspectrogramas (filas 2 y 3) o espectrogramas lineales (fila 4). Sin embargo, EnCodecMAE utilizando representaciones de EnCodec (fila 11) como entrada obtiene el mejor desempeño en esa tarea. Lo opuesto ocurre en las tareas de reconocimiento de comandos de voz (SC), detección de eventos acústicos (FSD) y clasificación de eventos (ESC), en donde EnCodec tiene un mejor desempeño que los melspectrogramas y el espectrograma lineal, pero luego al ser utilizado como entrada de EnCodecMAE obtiene el peor desempeño en estas tareas. Esto es un resultado desalentador, ya que no nos permite ganar una intuición de cual va a ser el comportamiento de EnCodecMAE al cambiar la representación de entrada, y la única manera de saberlo será entrenándolo.

Este mismo problema ocurre al observar el desempeño de los modelos en la tarea de pretexto. Un mejor desempeño en la misma, no se corresponde con un mejor desempeño en las tareas downstream. Por esta razón, el pre-entrenamiento de modelos mediante autosupervisión puede ser como caminar a ciegas, complicando la búsqueda de hiperparámetros y el diseño de los mismos. En la Figura 4.1 se puede observar el desempeño en la tarea de pretexto para cada cuantizador en el conjunto de datos ESC-50, que consiste de eventos acústicos y no fue utilizado en el conjunto

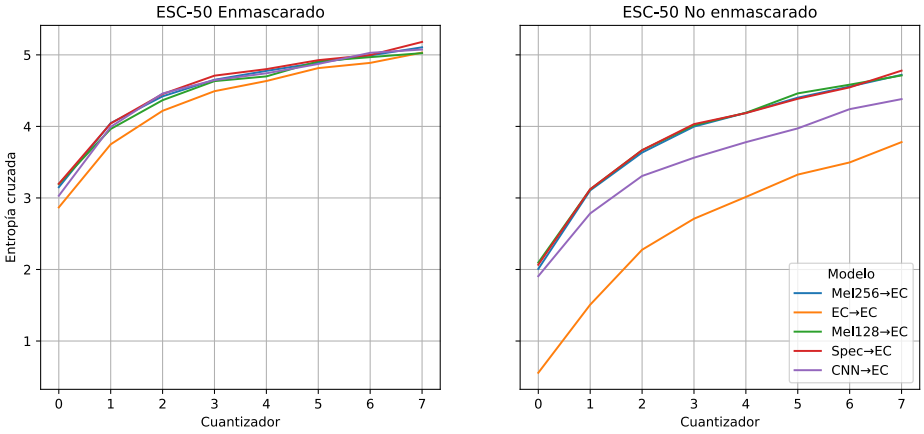

Figura 4.1: Entropía cruzada obtenida en el conjunto de datos ESC-50 para la tarea de pretexto, para los segmentos que fueron enmascarados y los que no, desglosados por cuantizador.

de pre-entrenamiento. Como era esperable, los cuantizadores más profundos, al estar capturando detalles cada vez más irrelevantes de la señal, son también los más difíciles de predecir para los modelos resultando en entropías cruzadas mayores. Por otro lado, el modelo que utiliza EnCodec como entrada (EC→EC) es el que mejor desempeño tiene, tanto a la hora de predecir a partir de segmentos enmascarados como no enmascarados. Esto se debe a que es la representación más cercana al objetivo a reconstruir, que también es EnCodec. La diferencia se acentúa más en los segmentos sin enmascarar, donde la tarea se hace más trivial y es equivalente a una autocodificación. El modelo que utiliza una CNN logra un desempeño similar al que usa EnCodec, dado que su entrada es el audio crudo y puede aprender a mantener información relevante para predecir los cuantizadores de EnCodec. Notablemente, como se observaba previamente, el modelo con el peor desempeño global en las tareas downstream es el que tiene el mejor desempeño en la tarea de pretexto. En otras palabras, lamentablemente, no es posible predecir cuál modelo va a tener el mejor rendimiento a partir de su rendimiento en la tarea de pretexto.

4.2. ¿Cuál es la mejor capa?

Para saber la salida de qué capa de un modelo pre-entrenado es óptima para una tarea downstream es necesario evaluar cada capa por separado. Esto puede consumir mucho tiempo, especialmente si el benchmark es costoso de computar. Dado que HEAREval es rápido y estamos evaluando sólo en un subconjunto de tareas, podemos realizar este tipo de análisis. Cabe resaltar que en trabajos previos se ha demostrado que no siempre la última capa es la mejor [211], que esta varía según la tarea a resolver y según el modelo [130, 212]. En modelos con arquitectura Masked Autoencoder, como EnCodecMAE, se suele utilizar la última capa del encoder como representación [3]. Esto último se debe al diseño mismo de la arquitectura, en la que

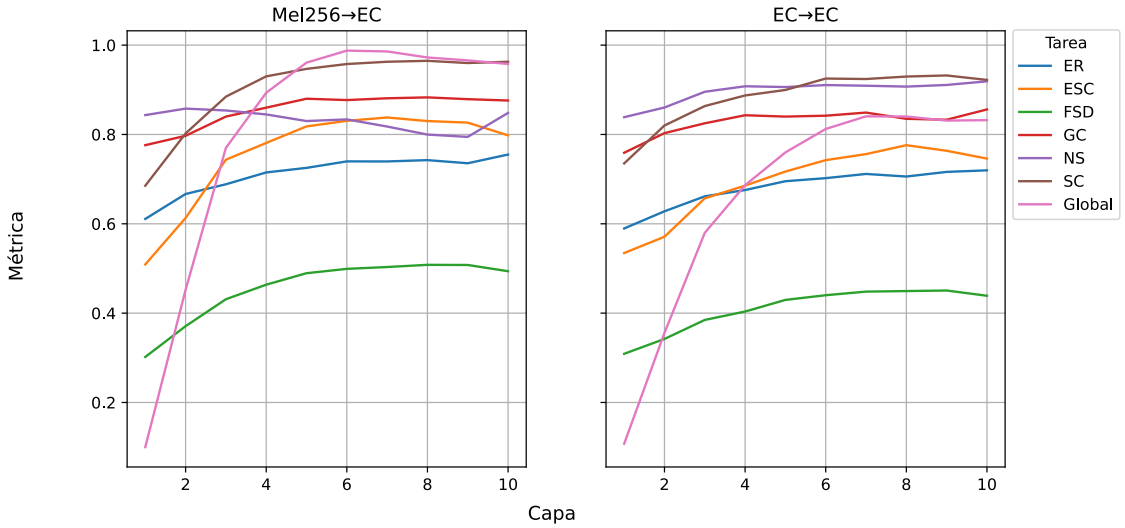

Figura 4.2: Desempeño del modelo Mel256→EC y EC→EC al utilizar salidas de distintas capas del Encoder como representación.

	Mel256→EC			EC→EC		
Tarea	Mejor capa	Métrica	Última capa	Mejor capa	Métrica	Última capa
ER	10	75.5	75.5	10	72.0	72.0
ESC	7	83.8	79.8	8	77.6	74.6
FSD	8	50.8	49.5	9	45.1	43.9
GC	8	88.3	87.6	10	85.6	85.6
NS	2	85.8	84.6	10	91.9	91.9
SC	8	96.5	96.3	9	93.2	92.2
Global	6	98.8	95.9	7	84.1	83.2

Tabla 4.2: Desempeño de la mejor capa y la última capa de Mel256→EC y EC→EC para cada tarea analizada.

se desacopla la etapa de representar y codificar los segmentos no enmascarados de la etapa de reconstruir los enmascarados. Por todo esto, resulta de interés analizar cuales son las mejores capas para cada tarea, y corroborar si la última capa del encoder es óptima.

En la Figura 4.2 se puede observar que la mejor capa en términos globales, no es la última sino que la sexta. También se observa que dependiendo de la tarea, la capa óptima es distinta. Esto se puede ver en la Tabla 4.2, donde se muestra la mejor capa para cada tarea, y su desempeño asociado. Con excepción de la clasificación de notas musicales (NS), para las demás tareas las mejores capas se encuentran hacia el final del encoder (capas 7,8 y 10). Una posible explicación de que NS se comporte de manera distinta, es que para clasificar notas musicales no se necesita de información contextual, sino más bien local y de corto plazo que permita determinar la frecuencia fundamental. Esto se evidencia también en la Tabla 4.1, en donde se observa que los melspectrogramas por si mismos dan buenos resultados en la tarea.

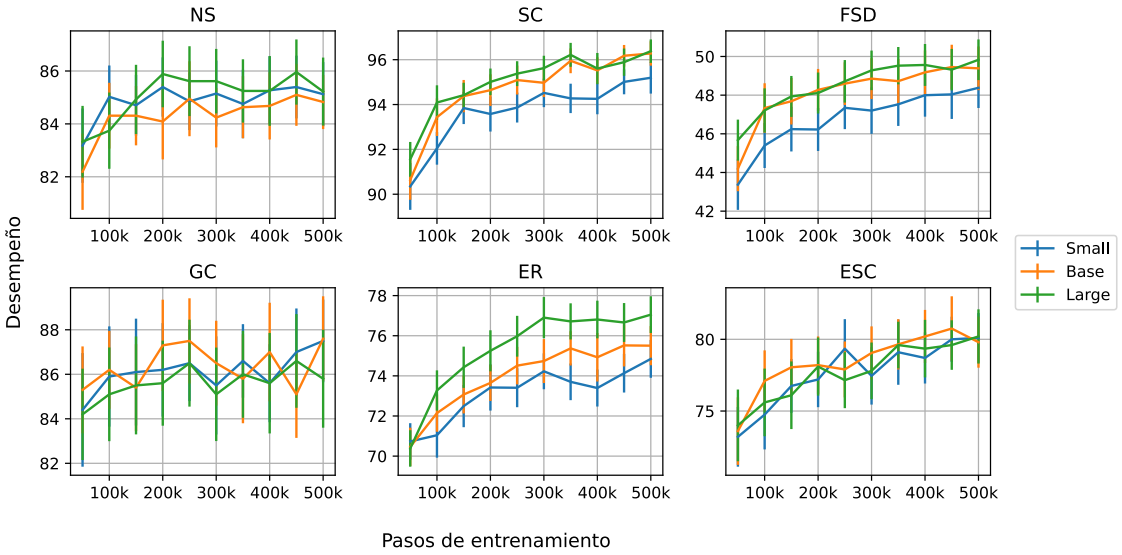

Figura 4.3: Desempeño en el subconjunto de HEAREval obtenido en distintos puntos del pre-entrenamiento de EnCodecMAE (Mel256→EC small, base y large).

4.3. ¿Cuándo detener el pre-entrenamiento?

Una pregunta que puede surgir es si es suficiente pre-entrenar los modelos por 500000 pasos. Si bien las curvas de entrenamiento no parecen decrecer de manera significativa, es posible que pequeños cambios en las representaciones internas tengan un efecto grande en su calidad para utilizarse en otras tareas, lográndose mejoras si se entrenara por más tiempo, o pudiéndose detener el pre-entrenamiento antes ahorrando recursos computacionales. Durante el pre-entrenamiento de nuestros modelos se realizaron checkpoints cada 50000 pasos de entrenamiento, por lo que podemos realizar un análisis del desempeño downstream con cada uno de estos checkpoints intermedios. En la Figura 4.3 se puede observar el desempeño en cada tarea en función de la cantidad de pasos de pre-entrenamiento. Se puede ver que para algunas tareas como clasificación de notas musicales y de género musical, el desempeño máximo parece haber sido alcanzado temprano durante el pre-entrenamiento, mientras que las tareas relacionadas al habla (SC y ER) y a eventos acústicos (FSD y ESC) podrían verse beneficiadas por un pre-entrenamiento más largo. Una observación positiva es que ninguna tarea parece verse perjudicada por un pre-entrenamiento más largo, sino que todas mantienen o mejoran su desempeño al agregar pasos de pre-entrenamiento. También se puede observar que en algunas tareas como NS, GC y ESC, el tamaño del modelo no parecería impactar fuertemente, mientras que en otras como SC y FSD, el impacto se ve principalmente al pasar de tamaño Small a Base. La tarea de reconocimiento de emociones (ER), es la que se ve más beneficiada por el incremento del tamaño de los modelos. Hipotetizamos que modelos más grandes son capaces de aprender a reconocer palabras de manera no supervisada, lo cual trae una ventaja en la tarea de reconocimiento de emociones. Por ejemplo, en [213], muestran que modelos de SSL recientes como Wav2Vec 2.0 y HuBERT son mejores

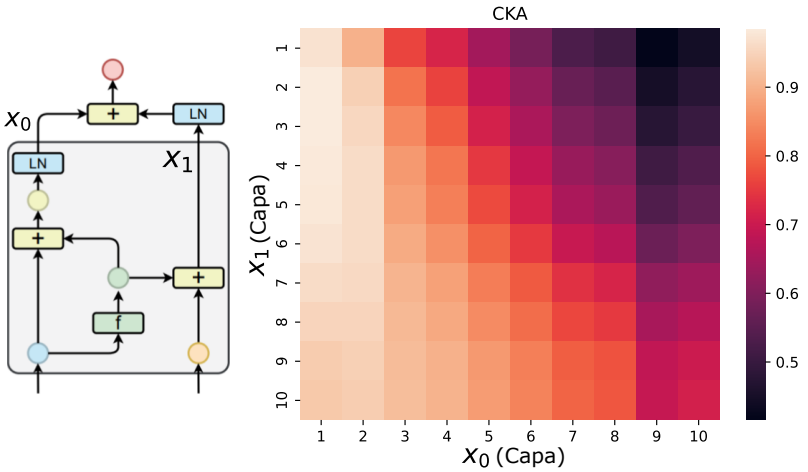

Figura 4.4: Izquierda: diagrama de la configuración utilizada en la pre-post normalización. CKAs entre la rama izquierda (X_0) y derecha (X_1) de la pre-post normalización en cada capa.

prediciendo valencia respecto a modelos más tradicionales como CNNs, y logran alcanzar a modelos que fusionan texto con audio, sugiriendo que implícitamente estos modelos están capturando información lingüística. Esto lo hemos podido mostrar para Wav2Vec 2.0, BEATs y EnCodecMAE utilizando técnicas de análisis de similaridad de representaciones en la tesis de licenciatura de datos de Cecilia Bolaños [214].

4.4. Efecto de la pre-post normalización

Se puede observar en las Figuras 4.2 que la última capa parece romper la tendencia en muchas tareas. Una explicación de este fenómeno es el uso de la pre-post normalización en nuestro modelo, la cual explicamos en la Sección 2.2.4. La salida del encoder es la suma de dos ramas, e hipotetizamos que una de las ramas puede funcionar como "skip connection" llevando información de las primeras capas hacia la última. Una forma de probar esta hipótesis es comparando la representación de cada rama por separado, con la representación de entrada. Una técnica muy utilizada para comparar activaciones de redes neuronales es Centered Kernel Alignment (CKA):

$$CKA(X,Y) = \frac{||Y^T X||_F^2}{||X^T X||_F^2 ||Y^T Y||_F^2},$$

en donde X e Y son dos representaciones distintas con forma $N \times D_X$ y $N \times D_Y$, con N la cantidad de instancias comparadas, y D_X y D_Y las dimensionalidades de las representaciones. Se puede mostrar que $||Y^T X||_F^2 = \langle vec(XX^T), vec(YY^T) \rangle$, lo cual intuitivamente equivale a comparar mediante un producto escalar las matrices de Gram de ambos vectores, que expresan la similaridad entre cada par de instancias de la representación. Por ende, CKA será alto si las distancias entre instancias es similar en ambas representaciones. Por ejemplo, si Y es una rotación de X, las distancias entre puntos se preservan y $CKA(X,Y) = 1$, mientras que si se realiza

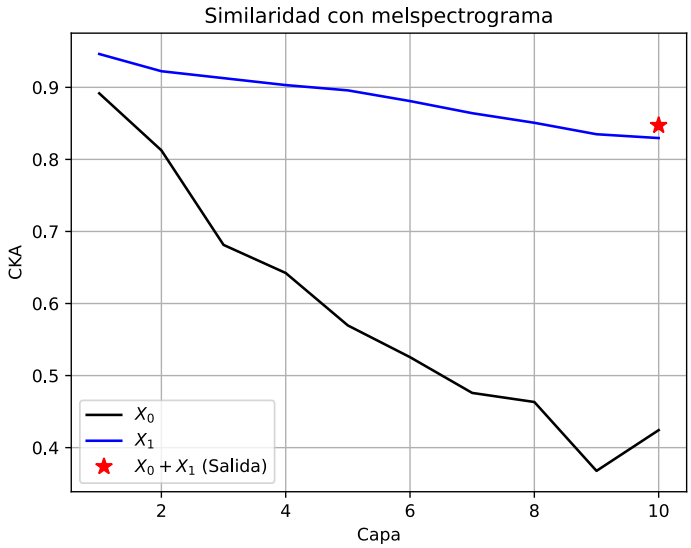

Figura 4.5: Similaridad calculada mediante CKA entre las distintas salidas de EnCodecMAE Mel256→EC y los melspectrogramas de entrada.

un escalamiento anisotrópico, es decir que se escalan distintas direcciones utilizando distintos factores, las distancias entre puntos dejan de conservarse y el valor de CKA puede ser menor a 1.

Podemos observar en la Figura 4.4 que las salidas de la rama derecha (X_1) en todas las capas son más similares a las salidas de las primeras capas de la rama izquierda (X_0). Esto indica que X_1 está conservando información de las primeras capas y propagándola hacia las últimas. Este comportamiento tiene sentido: al decoder le resulta útil conservar la información local de los tokens visibles. Por último, como se puede observar en el diagrama de pre-post normalización en la Figura 4.4, la última capa combina ambas ramas, por lo que al extraer la activación de esta última capa estamos combinando información de las primeras capas con la de las últimas. Esto explicaría porqué el comportamiento de la última capa muchas veces es distinto al resto en EnCodecMAE, debido a que para el resto de las capas se está utilizando solo la rama izquierda X_0.

En la Figura 4.5 se puede observar un análisis complementario, comparando las distintas salidas de las capas con el melspectrograma de entrada. Se puede observar que la similaridad decrece a mayor velocidad en la rama izquierda (X_0) que la derecha (X_1), confirmando que la rama derecha propaga información local hacia la salida. Por último, a efectos prácticos se muestra la salida de la última capa en la que se suman X_0 y X_1 tras aplicar layer normalization. Esta es la capa que se utiliza al extraer representaciones de EnCodecMAE, y se puede observar que al reincorporar la información local mediante X_1 se vuelve más similar al melspectrograma y exhibe un comportamiento diferente al de las capas anteriores.

4.5. Efecto de la señal objetivo

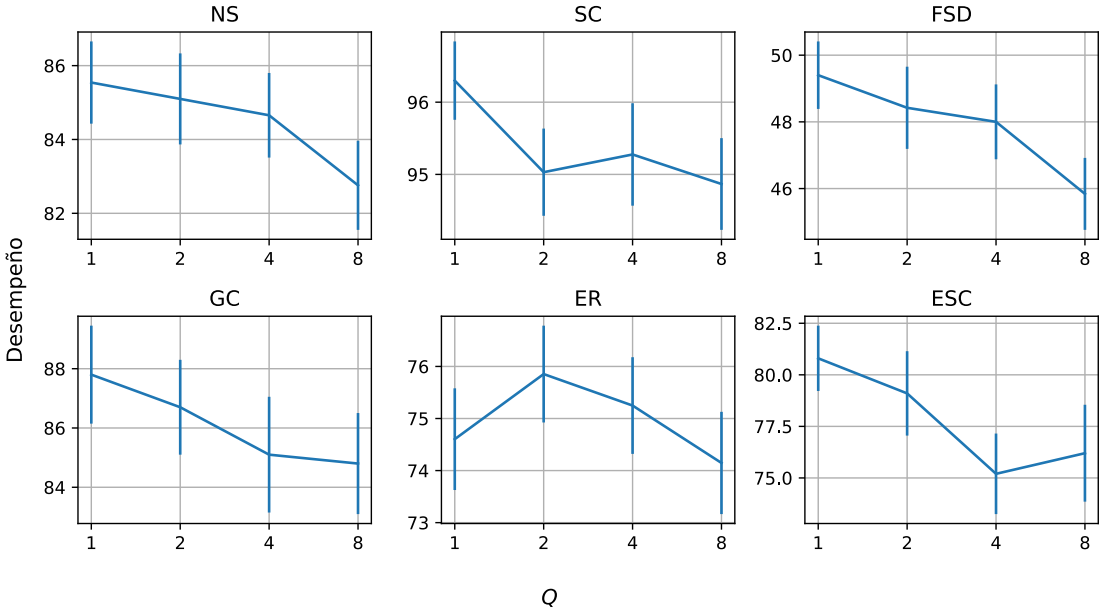

Figura 4.6: Efecto del cuantizador utilizado como señal objetivo.

Creemos que la señal elegida como objetivo es muy importante. En nuestro trabajo, nos diferenciamos de la literatura al utilizar un codec de audio neuronal como señal objetivo. Esto nos permite formular el problema de MLM como uno de clasificación, sin perder información relevante al discretizar la señal objetivo, a diferencia de otros trabajos como HuBERT, que emplean clusters de MFCCs como objetivo, a partir de los cuales no es posible reconstruir fielmente la señal original. EnCodec, como otros codecs de audio neuronales, codifica la señal de audio como múltiples secuencias discretas. Particularmente, EnCodec utiliza 32 cuantizadores, dando lugar a 32 secuencias discretas. Una pregunta que puede surgir es cuál de estos 32 cuantizadores funciona mejor como objetivo. Nuestra hipótesis es que los primeros cuantizadores poseen información más gruesa y relevante del audio, mientras que los últimos codifican detalles que aportan poca energía a la señal. Es por esto que nos inclinamos a pensar que lo más beneficioso es utilizar el primer cuantizador. En la Figura 4.6 se puede observar el desempeño en las tareas downstream al utilizar distintos cuantizadores como objetivo. Se puede confirmar que para todas las tareas, con excepción de reconocimiento de emociones (ER), el mejor desempeño se obtiene al utilizar el primer cuantizador como objetivo. En general se observa la tendencia de que el desempeño disminuye con la profundidad del cuantizador, lo cual se explica por la organización jerárquica de los mismos, codificando información menos relevante para la reconstrucción a medida que se avanza en profundidad. No queda claro por qué el segundo cuantizador da lugar a un mejor desempeño en emociones que el primero. Hipotetizamos que si bien el primer cuantizador debería contener la

información más relevante, algunos atributos necesarios para la tarea podrían no estar capturados por este cuantizador pero si por el siguiente. A modo de ejemplo, pensemos en cuantizar señales de habla de una manera "ideal" o "intuitiva". Podríamos suponer que el primer cuantizador captura los 44 fonemas del idioma inglés. Sin embargo, la información fonética no es suficiente para reconstruir habla; debemos codificar también la frecuencia fundamental. Suponiendo 2 octavas de rango y una resolución de semitono, podemos pensar en 24 valores de F0 posibles. Con todas estas simplificaciones ya tenemos $24 \cdot 44 = 1056$ índices necesarios para representar en una secuencia discreta estos atributos. Este valor se encuentra en el orden de los 1024 códigos que utiliza cada cuantizador de EnCodec. Sin embargo, aún no podremos reconstruir el habla; nos falta codificar información del timbre de la voz, de la energía, del canal acústico, etc. Es por esto que tiene sentido que otros cuantizadores que no sean el primero, sean relevantes para tareas específicas, y que combinar cuantizadores en la señal objetivo sea superador a solo utilizar el primer cuantizador.

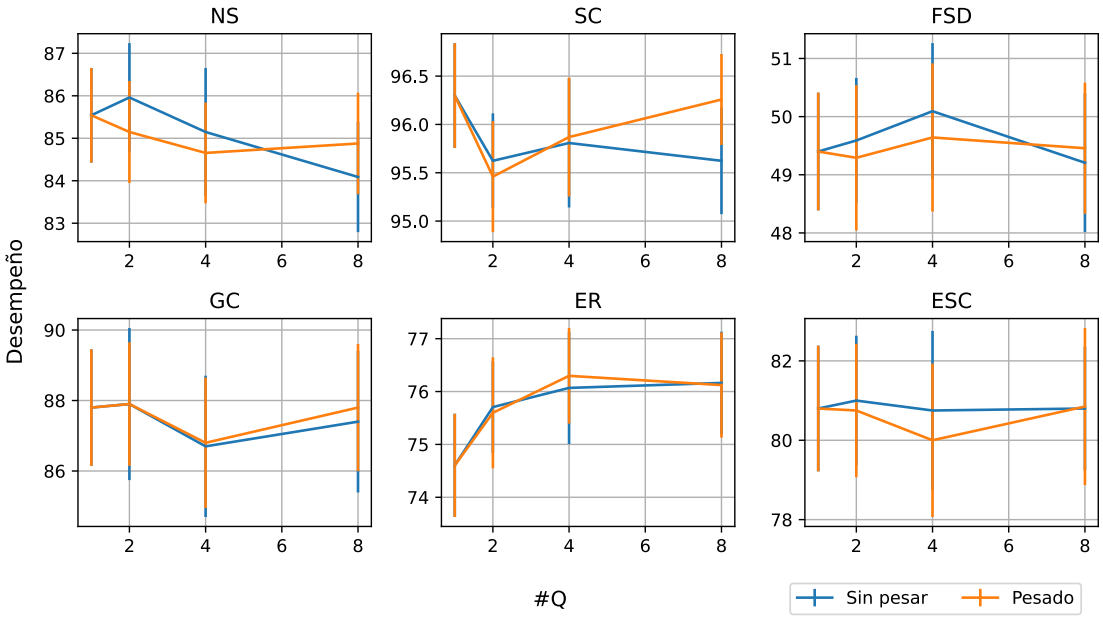

Figura 4.7: Efecto de la cantidad de cuantizadores utilizados en la señal objetivo y el peso asignado.

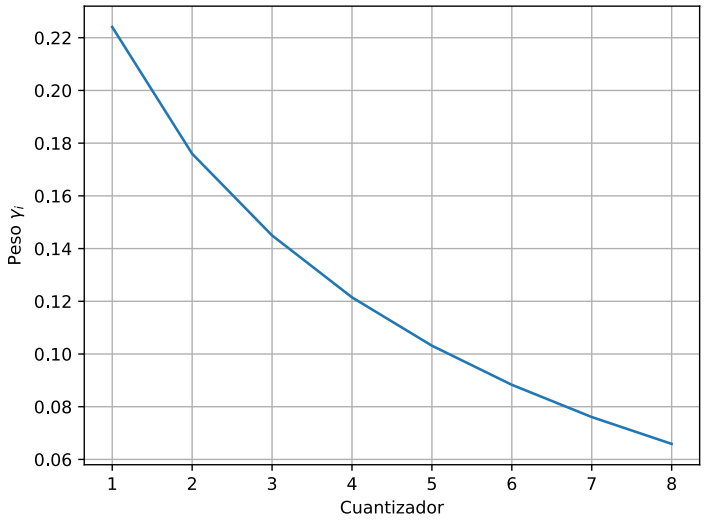

Figura 4.8: Pesos de los cuantizadores.

A continuación probamos combinar distintos cuantizadores utilizando pesos uniformes. Los resultados se pueden observar en la Figura 4.7 (curva azul). La tendencia no es clara en este caso y depende de la tarea. Algunas tareas como ER son claramente beneficiadas por el uso de cuantizadores más allá del primero, mientras otras como GC y ESC no se ven afectadas de manera significativa, y otras como SC y NS son perjudicadas al incorporar más cuantizadores. Hipotetizamos que el hecho de que todos los cuantizadores tengan el mismo efecto en la función objetivo debido a los pesos uniformes, hace que el modelo no aprenda bien a reconstruir el primer cuantizador, ya que debe utilizar parte de su capacidad para reconstruir los otros cuantizadores. Para darle mayor peso a los primeros cuantizadores, que deberían representar lo más relevante de la señal, decidimos pesar cada cuantizador por el error de cuantización promedio a su salida, calculado sobre 150 muestras aleatorias del conjunto de entrenamiento, y normalizado para que sume 1. El error de cuantización es monótonamente decreciente ya que cada cuantizador reduce el error del anterior. En la Figura 4.8 se pueden observar los pesos obtenidos, y en la Figura 4.7 se puede ver el efecto de utilizar distintas cantidades de cuantizadores pesados (curva naranja). Para la mayoría de las tareas, el desempeño no se ve muy afectado respecto a utilizar solo el primer cuantizador, con excepción de la tarea de ER, en donde utilizar 4 u 8 cuantizadores mejora el desempeño de manera significativa, y la tarea de NS, la cual empeora ligeramente al utilizar más de un cuantizador. A diferencia de utilizar pesos uniformes, no hay una degradación en comandos de voz (SC) al utilizar los 8 cuantizadores. Más allá de los resultados, los cuales no son concluyentes respecto a una ventaja o desventaja en términos de desempeño, creemos que es conveniente incorporar más de un cuantizador en la señal objetivo para lograr una representación que no solo capture información gruesa de la señal de audio. A su vez, incorporar más cuantizadores en la señal objetivo no introduce parámetros nuevos en el encoder

ni ralentiza significativamente el entrenamiento, y de la misma manera en la que produjo una mejora en ER, podría traer mejoras en otras tareas que no evaluamos.

Si bien durante el desarrollo de EnCodecMAE, el único codec neuronal de audio genérico con código abierto era EnCodec, en la actualidad nuevos modelos han sido liberados los cuales poseen características distintivas que son interesantes de analizar en el contexto de esta tesis. Particularmente, nos enfocamos en WavTokenizer [124], el cual no hace uso de Residual Vector Quantization (RVQ) y utiliza un solo cuantizador, diferenciándose de EnCodec. A pesar de su alta tasa de compresión, WavTokenizer es capaz de reconstruir audio fielmente, incluyendo música, habla y sonidos ambientales. WavTokenizer utiliza el mismo encoder que EnCodec, lo cual facilita la adaptación de EnCodecMAE a este nuevo codec ya que genera la misma cantidad de tokens por segundo. Las principales diferencias con EnCodec radican en que utilizan un solo cuantizador con 4096 códigos en lugar de 32 cuantizadores con 1024 códigos cada uno, y emplean un decoder asimétrico respecto al encoder. El decoder en lugar de utilizar capas de convolución transpuesta para upsamplear los tokens a la longitud del audio, utiliza una transformada inversa de Fourier a su salida, y se mantiene la dimensionalidad entre las distintas capas del decoder.

A su vez, otros trabajos como BEATs y BestRQ [151] han utilizado cuantizadores aleatorios como señal objetivo. Decidimos utilizar la implementación de cuantizador aleatorio de BestRQ, en el que se realiza una proyección lineal con pesos aleatorios del melspectrograma de entrada, y se utilizan codebooks con 8192 códigos aleatorios para discretizar estas proyecciones aleatorias de 16 dimensiones. A diferencia de BestRQ, no concatenamos cada 4 cuadros consecutivos de melspectrogramas ya que queríamos mantener la resolución temporal de nuestros experimentos previos.

En la Tabla 4.3 se puede observar el efecto de utilizar estos distintos codecs de audio neuronales como señales objetivo. EnCodec resultó ser la mejor señal objetivo entre las que exploramos. En un principio hipotetizamos que WavTokenizer, al lograr una mejor compresión y utilizar un solo cuantizador, sería una mejor señal objetivo. Los resultados muestran lo contrario, y creemos que se debe a que la estructura jerárquica de los cuantizadores en EnCodec, y el uso de pesos, permite al modelo enfocarse principalmente en la reconstrucción del primer cuantizador, y no poner el foco en detalles que quizás no sean relevantes a la semántica de la señal. WavTokenizer, al utilizar un solo cuantizador no sigue una estructura jerárquica y no le permite a nuestros modelos ponerle menor peso a los detalles finos. Finalmente, utilizar cuantizadores aleatorios para generar la señal objetivo (llamado "Random" en la tabla) tampoco fue superador de EnCodec, justificando nuestra elección de señal objetivo.

Objetivo	NS	GC	SC	ER	FSD	ESC	Global
(1) EnCodec	84.6±1.0	**87.6±1.9**	**96.3±0.5**	**75.5±1.0**	**49.5±1.1**	**79.8±1.8**	**95.9**
(2) WavTokenizer	**87.7±1.0**	85.3±1.8	95.5±0.6	74.4±0.9	46.9±1.1	76.0±2.0	89.5
(3) Random	81.1±1.3	**86.1±2.4**	95.4±0.7	74.4±1.0	47.3±1.1	**78.3±1.8**	90.9

Tabla 4.3: Desempeño con distintas señales objetivo.

4.6. Efecto del enmascarado

En esta sección se describirá con mayor detalle el proceso de enmascarado, distintas variantes del mismo, y el efecto de los hiperparámetros asociados en el desempeño de la representación. En el Algoritmo 1 se describe el proceso de generación de máscaras implementado, el cual está inspirado en Wav2Vec 2.0 y la mayoría de los trabajos aplicando MLM a señales de audio o habla.

Algoritmo 1: Proceso de generación de máscaras

Dados: un batch de B representaciones de audios $A = \{a_i\}_{i=1}^{B}$ con longitud $\{l_i\}_{i=1}^{B}$; una proporción de enmascarado M_{prop}; y un tamaño de gap M_{gap}.

for each *representación* $a_i \in A$ **do**
- $m_i \leftarrow \lfloor l_i * M_{prop} \rfloor$; `// Cantidad de cuadros a enmascarar.`
- $mask \leftarrow zeros(l_i)$;
- $mf \leftarrow 0$; `// Cantidad de cuadros ya enmascarados.`
- $start\_idxs \leftarrow []$;
- **while** $mf < m_i$ **do**
  - $start\_idx \leftarrow randint(0, max(0, l_i - M_{gap}))$;
  - $start\_idxs.append(start\_idx)$;
  - $mask[start\_idx : start\_idx + M_{gap}] \leftarrow 1$;
  - $mf \leftarrow sum(mask)$
- **if** $mf > m_i$ **do**
  - `// Revierto cuadros enmascarados extra`
  - $mask[start\_idxs[0] : start\_idxs[0] + (mf - m_i)] \leftarrow 0$;

En la Figura 4.9 se puede observar un ejemplo de máscara generada con hiperparámetros $M_{gap} = 15$ y $M_{prop} = 0.5$. Esto significa que por cada índice muestreado, se enmascaran los 15 cuadros consecutivos, y en total se enmascara el 50% de la señal. Se muestran 3 detalles relevantes:

1. Si dos índices muestreados están a una distancia menor a M_{gap} entonces ocurrirá un solapamiento, dando lugar a una región enmascarada con una longitud mayor a M_{gap}.
2. Debido a que en nuestra implementación somos estrictos en la cantidad de enmascarado, puede ocurrir que se desenmascare una parte de una región previamente enmascarada, dando lugar a una región enmascarada con longitud menor a M_{gap}. Cabe destacar que esto ocurrirá como máximo una vez por cada señal.
3. Debido a que no hay ninguna restricción respecto a la ubicación de los índices muestreados, las regiones visibles podrían ser segmentos contigüos muy cortos.

La Figura 4.10 muestra los resultados obtenidos usando distintos valores de M_{prop} para el modelo base Mel256→EC, dejando fijo el tamaño de gap M_{gap} en 15. Se puede observar que las tareas de reconocimiento de comandos de voz (SC) y la de reconocimiento de emociones (ER), son las más afectadas por la proporción de

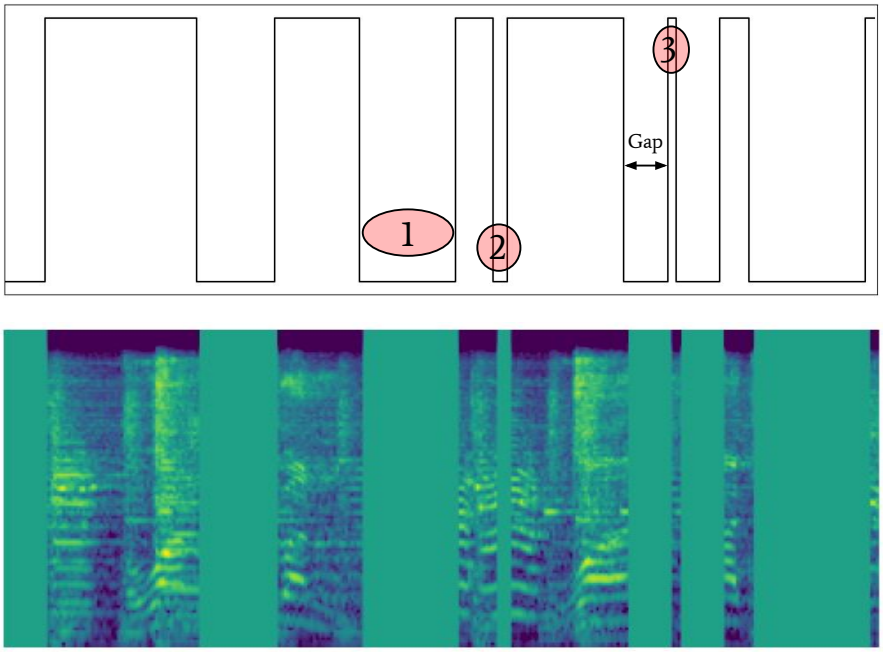

Figura 4.9: Ejemplo de máscara y efecto sobre la entrada.

enmascarado junto con la de detección de eventos acústicos (FSD). Hipotetizamos que distintas proporciones de enmascarado van a promover que el modelo aprenda a modelar distintas dependencias temporales. Particularmente, podemos imaginar que si enmascaramos solo una pequeña parte de la señal, se volverá trivial reconstruir la información faltante. Al contrario, si se pierde la mayor parte de la señal será muy difícil aprender dependencias temporales a partir de la poca señal visible. Esto es lo que se observa en las tareas donde este hiperparámetro más afecta al desempeño; los valores óptimos de M_{prop} se encuentran alrededor de 0.5, y los valores más extremos de M_{prop} (0.2 y 0.8), son los que llevan a un peor desempeño. Creemos que en las tareas de clasificación de nota musical (NS), clasificación de género musical (GC) y clasificación de eventos acústicos (ESC), este hiperparámetro no afecta tanto porque son tareas que podrían no necesitar de modelar dependencias temporales a largo plazo (la información local es suficiente). Por ejemplo, NS contiene audios de 4 segundos en los que suena una única nota de un instrumento musical. Para identificar cuál nota es, uno podría utilizar un segmento más corto proveniente de casi cualquier parte de la señal (quizás exceptuando el ataque). Del mismo modo, algunos géneros musicales se pueden reconocer a partir de la envolvente espectral, la cual se encuentra disponible en todo el audio. En cambio, si utilizamos un segmento de una conversación, podemos estar perdiendo información clave para identificar una emoción. Del mismo modo, para identificar un comando de voz, es necesario en general escuchar el comando entero o una gran parte del mismo.

Análisis similares previos de la literatura, realizados en otros modelos y dominios, muestran que la proporción de enmascarado óptima depende mucho del tipo de señal, del tipo de enmascarado, y de la tarea downstream. Por ejemplo, en modelos

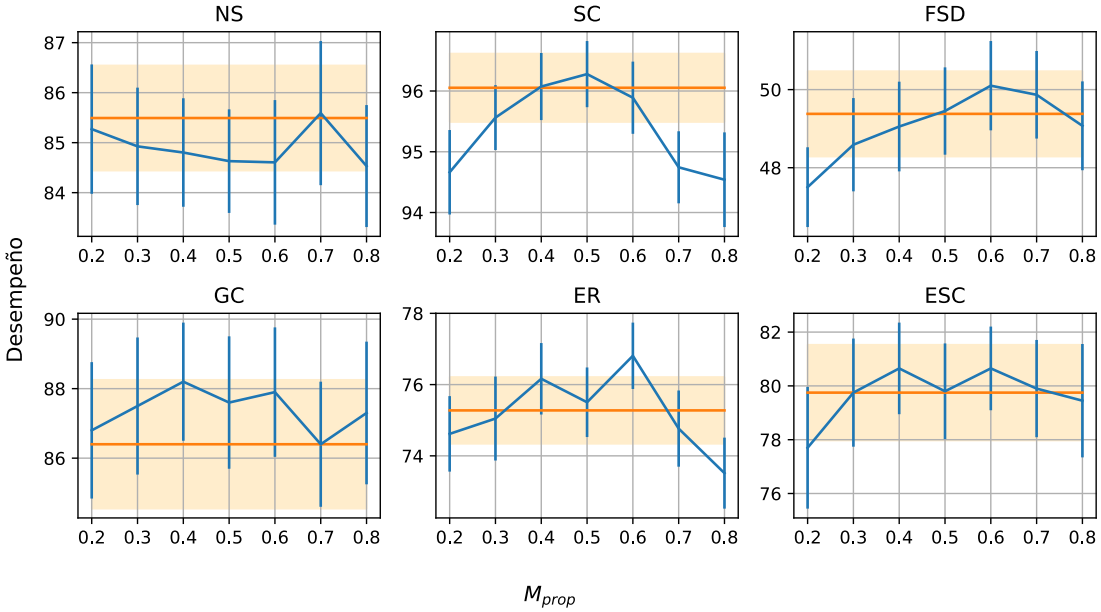

Figura 4.10: Efecto de la proporción de enmascarado en el desempeño de las distintas tareas evaluadas para el modelo base Mel256→EC. La línea naranja indica el desempeño del modelo entrenado con una proporción de enmascarado aleatoria entre 0.2 y 0.8, y la sombra indica el intervalo de confianza para ese caso. Los intervalos son estimados mediante bootstrapping.

de imágenes, el enmascaramiento óptimo suele ser de alrededor de un 70% de la imagen [3], mientras que en video, el enmascaramiento óptimo puede llegar a ser del 90% [215]. Una explicación de este fenómeno es que las señales de video poseen mayores correlaciones que las de imagen por la presencia del eje temporal, donde normalmente cuadros consecutivos son muy similares entre sí. En el dominio del audio, el enmascaramiento óptimo depende mucho de la estrategia de enmascarado. En general, los modelos de MAE trabajan sobre espectrogramas y enmascaran patches del mismo, en lugar de enmascarar segmentos temporales por completo. Algunos trabajos como AudioMAE [160], encuentran que el enmascaramiento óptimo para la tarea de detección de eventos acústicos, si es por patches es cercano al 80% del espectrograma, mientras que si se enmascaran segmentos temporales completos, es de un 40%. También exploran enmascarar bandas completas del espectro, llegando a la conclusión de que un enmascaramiento de alrededor del 50% es óptimo. A su vez, el enmascaramiento por patches parece dar mejores resultados que por segmentos temporales o bandas espectrales. Este análisis se expande en MSM-MAE [162], en donde muestran que para tareas relacionadas a eventos acústicos, el enmascaramiento por patches funciona mejor que el temporal, mientras que en tareas de habla, donde es necesaria mayor resolución temporal, ocurre lo contrario. Por último, hipotetizamos que utilizar múltiples proporciones de enmascaramiento durante el pre-entrenamiento podría ser beneficioso para el aprendizaje de representaciones, ya que podrían promover el aprendizaje de dependencias temporales a distintas escalas.

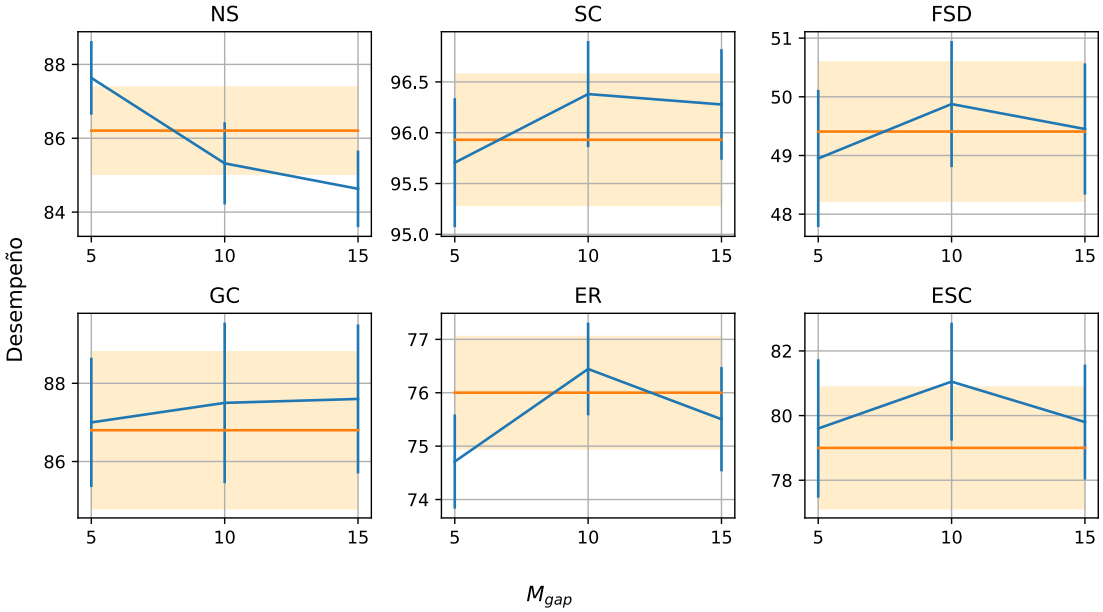

Figura 4.11: Efecto del largo del gap en el desempeño de las distintas tareas evaluadas para el modelo base Mel256→EC. Enmascarar 5, 10 y 15 cuadros consecutivos equivale a 66, 133 y 200 ms respectivamente. La línea naranja indica el desempeño del modelo entrenado con un largo de gap aleatorio entre 2 y 20.

Motivados por esta hipótesis, y por la falta de trabajos que exploren utilizar múltiples proporciones de enmascarado en simultáneo, para cada batch de pre-entrenamiento muestreamos $M_{prop} \sim U(0.2,0.8)$ siguiendo una distribución uniforme entre 0.2 y 0.8. Se puede observar en la Figura 4.10 que las representaciones resultantes (graficadas como una línea naranja), en lugar de mostrar una sinergía y mejorar al modelo con el mejor M_{prop} para cada tarea, se mantienen en general con un desempeño cercano al promedio de los distintos M_{prop}.

En la Figura 4.11, se observa el efecto de M_{gap} en el desempeño de los modelos para una proporción de enmascarado fija $M_{prop} = 0.5$. Nuevamente, el efecto es más pronunciado en las tareas de habla (SC y ER), y en las de eventos acústicos (FSD y ESC). En todos estos casos, un $M_{gap} = 10$ brinda el mejor desempeño. En cambio, para NS la cual es una tarea que no requiere de contextualización, el mejor desempeño se obtiene con gaps chicos, y para clasificación de género musical el tamaño de gap no parece afectar de manera significativa. Nuevamente exploramos variar el M_{gap} aleatoriamente durante el pre-entrenamiento, utilizando una distribución uniforme en el intervalo [2,20]. Se puede observar en la Figura 4.11 que esta estrategia no fue superadora a utilizar valores fijos de M_{gap}. Otra opción que no exploramos y que podría dar mejores resultados que utilizar valores aleatorios es realizar **curriculum learning** [216], aumentando progresivamente la dificultad de la tarea mediante un incremento de M_{gap} o M_{prop}.

Un hiperparámetro relacionado al enmascarado es el peso σ que se le da a las partes enmascaradas respecto a las no enmascaradas en la función de costo. Es decir,

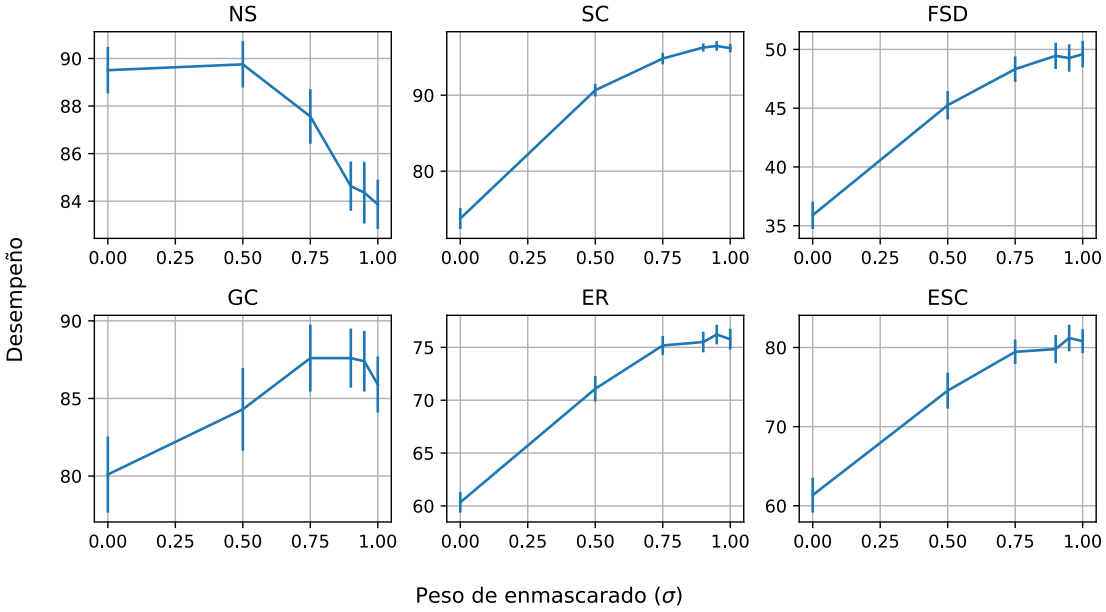

Figura 4.12: Efecto del peso de enmascarado σ en el desempeño.

cuando $\sigma = 0$, solamente se tienen en cuenta los segmentos sin enmascarar en la función de costo, haciendo que la tarea sea muy similar a la de un autoencoder clásico con una señal objetivo igual a la de entrada. Por el contrario, si $\sigma = 1$, solamente se tienen en cuenta los segmentos enmascarados. Nuestra hipótesis es que controlando esta variable se puede hacer que las representaciones sean más o menos locales; con un $\sigma = 0$ el modelo no necesita contextualizar para resolver la tarea, mientras que con un $\sigma = 1$ el modelo se ve forzado a utilizar la información del contexto. Al mismo tiempo creemos que ciertas tareas necesitan de información local mientras que otras se ven beneficiadas de una contextualización. En la Figura 4.12 se puede observar el desempeño del modelo base en función de σ. Tal como intuíamos, el desempeño del modelo se ve muy deteriorado cuando $\sigma = 0$, y la tarea de pretexto es equivalente a la de un autoencoder salvo que en nuestro caso no hay un cuello de botella sino que, por el contrario, se pasa de 24000 a $75 \cdot 768 = 57600$ muestras por segundo. Con excepción de la tarea de clasificación de notas musicales (NS), todas las tareas se vieron beneficiadas al incrementar σ, obligando al modelo a predecir segmentos enmascarados. Tal como intuíamos, utilizar un σ ligeramente menor a 1 ayuda a algunas tareas como clasificación de género musical (GC), aunque los resultados no son concluyentes en cuanto a si es conveniente darle peso a los segmentos no enmascarados. Lo que sí es claro es que σ deberá encontrarse más cerca de 1 que de 0 si deseamos una representación general de audio, ya que un valor bajo perjudica a todas las tareas excepto NS.

Volviendo a los 3 detalles relevantes de la Figura 4.9, que hayan regiones enmascaradas más largas que M_{gap} por un solapamiento de las máscaras (item 1) es común en la literatura, aunque no se ha explorado si este solapamiento ayuda al pre-entrenamiento. Respecto a ser estrictos en la cantidad de enmascarado (item 2),

resulta fácil cuantificar el impacto de esta decisión relajando el algoritmo y permitiendo que se enmascare una proporción ligeramente mayor a M_{prop} de la entrada. Por último, trabajos recientes han notado el problema de que las regiones visibles pueden ser segmentos muy cortos (item 3), y la propuesta es invertir la máscara, es decir, lo que se muestrean son índices de comienzo de zonas no enmascaradas (o visibles). Este método alternativo de enmascarado se denomina *Inverse Block Masking* y fue introducido en Data2Vec 2.0 y aplicado en EAT.

En la Tabla 4.4 se muestra el desempeño de estas variantes de enmascarado. Se puede observar que todas las variantes de enmascarado poseen un desempeño similar, indicando que los distintos cambios propuestos no afectan mucho en las representaciones aprendidas.

Tipo	NS	GC	SC	ER	FSD	ESC	Global
(1) Estricta	**84.6**$\pm$1.0	**87.6**$\pm$1.9	**96.3**$\pm$0.5	75.5$\pm$1.0	**49.5**$\pm$1.1	79.8$\pm$1.8	95.9
(2) No estricta	**84.7**$\pm$1.4	**87.8**$\pm$1.8	**96.0**$\pm$0.6	**77.4**$\pm$1.0	**49.7**$\pm$1.1	**81.5**$\pm$1.7	**97.4**
(3) (2) + Inverse Block	**83.7**$\pm$1.1	**87.1**$\pm$2.0	**95.9**$\pm$0.5	76.4$\pm$0.9	**49.2**$\pm$1.2	**81.4**$\pm$1.8	96.7

Tabla 4.4: Desempeño con distintas variantes de enmascarado.

4.7. Efecto del tamaño del decoder

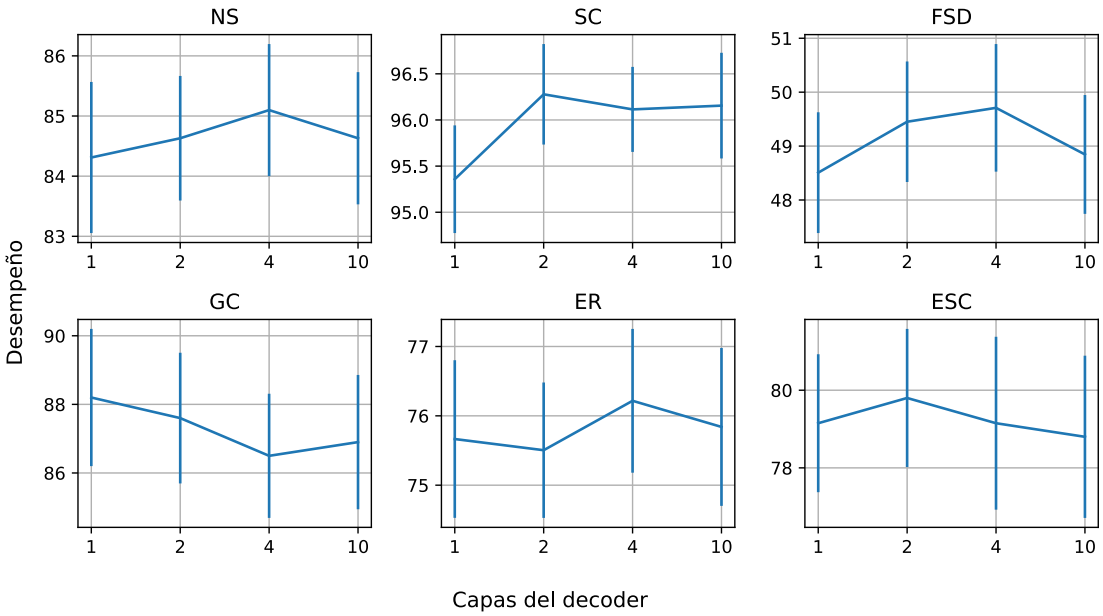

Figura 4.13: Efecto de la cantidad de capas del decoder en el desempeño.

Otro de los hiperparámetros que creíamos que podía afectar al aprendizaje de representaciones era el tamaño del decoder. Si el decoder es muy sencillo, hipotetizamos que la representación del encoder va a perder información quizás relevante para

algunas tareas, ya que el decoder no iba a ser lo suficientemente capaz de reconstruir la señal. Por otro lado, si el decoder es muy grande, el modelo podría empezar a utilizarlo como un encoder, y la salida del encoder podría dejar de ser una buena representación.

Por esto, decidimos variar la cantidad de capas del decoder y medir el desempeño en las distintas tareas downstream. Como se puede ver en la Figura 4.13, como nuestra intuición sugería, los modelos con decoders de tamaños intermedios (2 y 4 capas), casi siempre son los que mejor desempeño tienen, aunque la diferencia no es la que esperábamos con respecto a usar un decoder de 1 o 10 capas.

Al inspeccionar el desempeño de los modelos en la tarea de pretexto, se pudo observar que el tamaño del decoder no influye mucho (ver Anexo B). Esto implica que agregar más capas al decoder no hace que la capacidad de reconstrucción, tanto para segmentos enmascarados como sin enmascarar, se vea afectada, indicando que un decoder con una sola capa es suficiente. Esto se ve respaldado por trabajos como Data2Vec 2.0 [142] y EAT [143] en donde el típico decoder basado en transformers se reemplaza por una red convolucional liviana, o incluso por una sola capa lineal. Queda como trabajo futuro explorar estas arquitecturas de decoder más livianas para acelerar el pre-entrenamiento de EnCodecMAE.

4.8. Variantes de EnCodecMAE

4.8.1. MLM con mezclas

Una variante de la tarea de MLM que ha tenido éxito en el aprendizaje de representaciones de habla, es la utilizada en WavLM [130] donde el modelo predice la señal objetivo a partir de una versión enmascarada y ruidosa de la entrada. Específicamente, se generan las entradas simulando habla solapada con ruido, y el modelo predice las etiquetas correspondientes a la señal original. En el caso de habla solapada, el modelo debe ser capaz de discernir cuál es el segmento de habla original que se quiere reconstruir. La manera de diferenciar la señal original de la agregada es haciendo que la señal agregada tenga una duración máxima del 50% de la original. De esta forma, la señal objetivo corresponde a la que está presente durante todo el segmento de audio.

Para nuestro modelo, debemos adaptar el proceso de mezcla ya que queremos no solo modelar habla, sino cualquier tipo de sonido. Por este motivo, en lugar de sumar habla a la señal original, sumamos un audio al azar proveniente de Audioset, el cual contiene audios diversos. El mecanismo para generar la señal de entrada se describe en el Algoritmo 2.

Algoritmo 2: Proceso de generación de entradas ruidosas

Dados: un batch de B audios $A = \{a^i\}_{i=1}^{B}$ con largo L; una probabilidad de mezcla p_m; un conjunto de M ruidos $N = \{n^i\}_{i=1}^{M}$.

Elegir S audios $A^S \subset A$ mediante un proceso de Bernoulli con probabilidad p_m;

for each *audio* $a^{pri} \in A^S$ **do**

- Elegir con probabilidad uniforme un ruido n^i de duración L_N^i ;
- Elegir una relación señal a ruido en decibelios (SNR) con una distribución uniforme $\mathcal{U}(-5{,}20)$;
- Elegir una duración de solapamiento l a partir de una distribución uniforme discreta $\mathcal{U}(1, \frac{L}{2})$;
- Corregir si el ruido tiene menor duración que el solapamiento: $l = \min(l, L_N)$;
- Elegir un comienzo del solapamiento s^{pri} a partir de una distribución uniforme discreta $\mathcal{U}(1, L - l)$;
- Elegir un comienzo en el audio de ruido s^{sec} a partir de una distribución uniforme discreta $\mathcal{U}(1, L_N - l)$;
- $a^{sec} \leftarrow a^{sec}[s^{sec} : s^{sec} + l]$;
- $E^{pri} \leftarrow \frac{1}{l} \sum_{i=0}^{l-1} (a^{pri}[s^{pri} + i])^2$;
- $E^{sec} \leftarrow \frac{1}{l} \sum_{i=0}^{l-1} (a^{sec}[i])^2$;
- $\alpha \leftarrow \sqrt{\frac{E^{pri}}{10^{\frac{SNR}{10}} E^{sec}}}$;
- $a^{pri}[s^{pri} : s^{pri} + l] \leftarrow a^{pri}[s^{pri} : s^{pri} + l] + \alpha \cdot a^{sec}$;

Utilizamos una probabilidad de mezcla $p_m = 0.2$ y los ruidos se muestrearon de el conjunto de entrenamiento no balanceado de Audioset.

4.8.2. Conexión residual larga

Como hemos visto en el estudio de ablación en la Sección 4.4, el encoder intenta conservar información local mediante la conexión residual introducida por la pre-post normalización. Para evitar que el encoder tenga que conservar la información de los tokens visibles y trasladarla desde la entrada hacia la salida del encoder, le proveemos al encoder una conexión residual larga entre la salida del primer bloque Transformer (x_{t1}) y la salida del Encoder (x_{last}). De esta manera, el decoder recibe la suma de información potencialmente más local (x_{t1}) y la salida del encoder (x_{last}), pudiendo el encoder enfocarse en capturar relaciones más abstractas y contextualizadas ya que la conexión residual le provee directamente al decoder de la información más local que pueda necesitar. Para entrenar los modelos downstream utilizamos la salida del encoder (x_{last}) antes de sumar la conexión residual.

4.8.3. Uso de registros

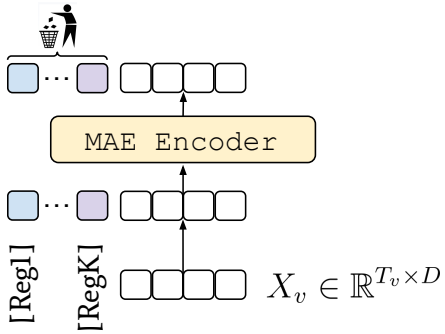

Figura 4.14: Modificación propuesta al encoder de EnCodecMAE. Se agregan K registros, es decir vectores inicializados de forma aleatoria al comienzo de la secuencia de tokens visibles. Estos registros pueden ser utilizados en capas intermedias del Transformer para almacenar información. Luego, los registros se descartan de la secuencia de salida, conservando solo los embeddings correspondientes al audio.

En Vision Transformers se encontró que patches poco informativos de la imagen de entrada, presentaban valores inusuales introduciendo artefactos a la hora de querer interpretarlos. En Vision Transformers Need Registers [217], los autores hipotetizan que el modelo utiliza partes de la imagen poco informativas para almacenar información relevante, generándose estos artefactos. Motivados por esta hipótesis, introducen el concepto de registros, los cuales toman el rol asumido por las partes poco informativas de la imagen, evitando la aparición de artefactos en las representaciones de la imagen. En la Figura 4.14 se puede observar la implementación de estos registros, los cuales son vectores entrenables que se inicializan aleatoriamente y se concatenan al comienzo de la secuencia de tokens visibles X_v. Luego, estos registros se descartan de la salida del encoder de EnCodecMAE. A pesar de que son descartados y no forman parte de la representación utilizada al entrenar modelos downstream, tienen una influencia en esta representación. Esto se debe a que en un transformer, cada vector en la secuencia de salida es generado teniendo en cuenta todos los vectores de la secuencia de entrada (incluyendo a los registros).

Dado que esta técnica no ha sido explorada en el dominio de audio, resulta interesante analizar si es útil y cómo afecta a las representaciones aprendidas por EnCodecMAE. Basados en los resultados del paper [217], utilizamos 4 registros en nuestro experimento, y mantuvimos el resto de los hiperparámetros del modelo base.

4.8.4. Resultados

En la Tabla 4.5, podemos observar el desempeño de las variantes propuestas de EnCodecMAE. Se puede ver que en términos de desempeño global, las estrategias propuestas (filas 2,3 y 5) fueron superadoras del modelo base. Particularmente, el uso de conexiones residuales largas (fila 3) fue la estrategia más exitosa, llevando el

desempeño global de 95.9 a 99.3. Las tareas más beneficiadas por el uso de la conexión residual fueron las relacionadas a detección y clasificación de eventos acústicos (FSD y ESC), mientras que la tarea de clasificación de notas musicales (NS) fue la más perjudicada. Esto tiene sentido ya que nuestra motivación inicial para utilizar una conexión residual, era hacer que el encoder se enfoque en contextualizar la señal visible en lugar de mantener la información local de las primeras capas. Al utilizar la salida después de la conexión residual larga (suma de las activaciones de la primera y última capa), se observa un deterioro del desempeño (ver fila 4), salvo en la tarea de clasificación de notas musicales (NS). En términos de no empeorar el desempeño del modelo base en ninguna tarea, el modelo entrenado con mezclas (fila 2), fue el más exitoso, manteniendo o mejorando el desempeño del modelo base en todas las tareas. Para el modelo con registros, utilizamos dos variantes: la primera consiste en promediar la salida del encoder descartando los registros (fila 5), mientras que la segunda consiste en promediar los registros descartando el resto de la secuencia (fila 6), hipotetizando que estos registros podrían funcionar como resúmenes de la secuencia. Sin embargo, usar sólo los registros no trae mejoras en el desempeño. Por último, combinamos las 3 técnicas: el uso de registros, el pre-entrenamiento con mezclas y la incorporación de la conexión residual. El modelo resultante (fila 7) obtiene el mejor desempeño global trayendo mejoras principalmente en las tareas relacionadas a eventos acústicos (FSD y ESC).

	NS	GC	SC	ER	FSD	ESC	Global
(1) Mel256→EC (Base)	84.6±1.0	**87.6±1.9**	**96.3±0.5**	75.5±1.0	49.5±1.1	79.8±1.8	95.9
(2) + Mix	**85.0±1.0**	**87.6±1.9**	**96.0±0.6**	**76.8±1.0**	**49.9±1.2**	81.2±2.1	97.4
(3) + Skip	82.1±1.3	**87.7±1.5**	**96.3±0.6**	75.6±1.1	**50.7±1.1**	**83.8±2.2**	99.3
(4) + Skip (suma)	**86.2±1.3**	86.2±2.1	**95.8±0.5**	75.7±0.9	48.8±1.1	79.0±2.1	94.6
(5) + Reg	84.8±1.4	**86.7±2.2**	95.7±0.5	**76.8±1.0**	49.5±1.0	81.5±2.0	97.1
(6) + Reg (global)	**85.1±1.0**	**86.3±1.9**	95.3±0.6	**76.5±1.0**	48.7±1.1	79.3±1.8	94.4
(7) + Reg + Mix + Skip	83.9±1.2	**86.5±2.0**	**95.8±0.5**	75.4±1.0	**51.0±1.1**	**84.1±1.8**	**100.0**

Tabla 4.5: Desempeño de las distintas variantes de EnCodecMAE propuestas.

4.9. Efecto de la semilla

Por último, nos preguntamos cuánto afecta al desempeño la semilla utilizada para entrenar el modelo upstream y el downstream. En particular, la semilla va a afectar en:

- Los pesos iniciales del modelo upstream y downstream.
- El orden de los datos durante el entrenamiento del modelo upstream y downstream.
- El proceso de generación de las máscaras.
- El dropout y otras capas que puedan contener componentes aleatorias.

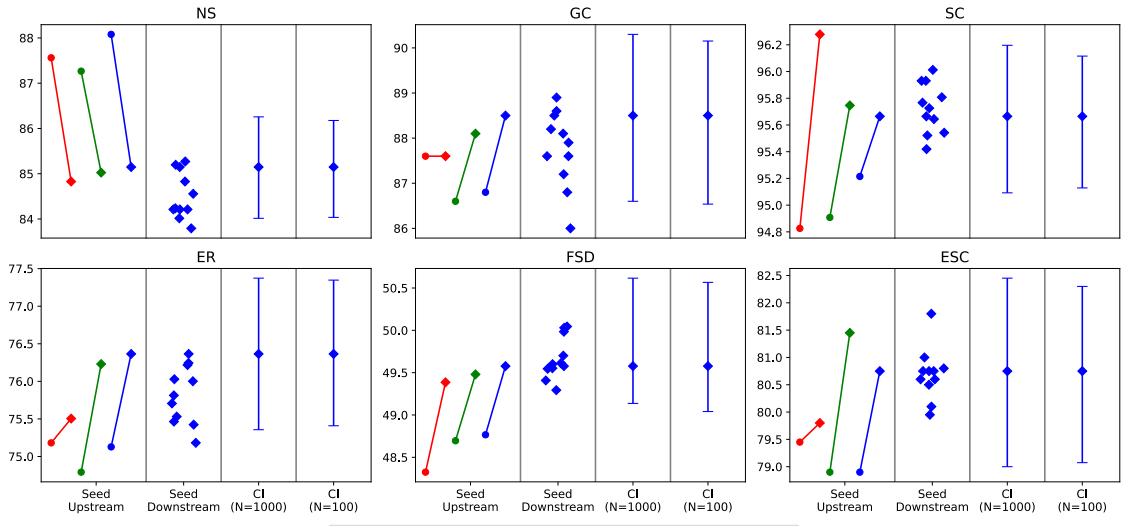

Figura 4.15: Efecto de la semilla utilizada en dos modelos upstream, de la semilla utilizada en el modelo downstream para un modelo upstream fijo, e intervalos de confianza ($\alpha = 0.05$) obtenidos mediante bootstrap con 100 y 1000 iteraciones.

- En el caso de MLM con mezclas, el proceso de generación de entradas ruidosas tiene componentes aleatorias, como la elección del sonido a mezclar.

En consecuencia, una pregunta que surge es si cambiar la semilla tiene un efecto mayor a cambiar otros hiperparámetros que hemos explorado. Se sabe que distintas semillas pueden llevar a resultados diferentes y en consecuencia a conclusiones erróneas sobre, por ejemplo, el efecto de cambiar la arquitectura de una red neuronal. Esta problemática fue estudiada y diversas soluciones fueron propuestas, como por ejemplo, optimizar la semilla o suavizar los pesos mediante un promediado móvil durante el entrenamiento [218–221]. Debido al elevado costo computacional de pre-entrenar nuestros modelos, resulta prohibitivo optimizar la semilla para cada experimento que desarrollamos, pero resulta interesante y útil cuantificar el efecto de la semilla en un modelo específico.

Para esto, pre-entrenamos dos variantes de EnCodecMAE utilizando 3 semillas distintas y realizamos la evaluación utilizando una semilla fija. Las variantes de EnCodecMAE que utilizamos fueron Mel256→EC (Base) y su variante utilizando un peso de enmascarado $\sigma = 0.75$, en lugar de $\sigma = 0.9$. A su vez, para una semilla fija del modelo base con $\sigma = 0.9$, realizamos la evaluación downstream utilizando 11 semillas distintas. Los resultados se pueden observar en la Figura 4.15 y sugieren que el desempeño es más sensible a la semilla utilizada durante la evaluación que a la semilla utilizada durante el pre-entrenamiento. De todas formas, podría ocurrir que el efecto de la semilla en el modelo upstream nos lleve a una conclusión erronea. Por ejemplo, si observamos la semilla correspondiente a la curva roja en la tarea de clasificación de género musical (GC), podríamos concluir que cambiar el peso de enmascarado no afecta al desempeño, mientras que si utilizamos la semilla correspondiente a las curvas verdes y azules, concluiríamos que el desempeño mejora al utilizar un mayor

σ.

Durante la experimentación en este trabajo doctoral, utilizamos intervalos de confianza obtenidos mediante bootstrap con $N = 100$ iteraciones como un criterio para determinar si dos modelos tenían un desempeño similar o si había una diferencia significativa. Estos intervalos de confianza nos dan una idea de la variabilidad en el desempeño de un modelo si el conjunto de evaluación fuera distinto, pero no contempla la variabilidad debido a distintas componentes aleatorias durante el entrenamiento del upstream y downstream. A pesar de esto, se puede observar que los intervalos de confianza obtenidos mediante bootstrap, en general, son más anchos que la variabilidad observada al cambiar la semilla en el upstream y downstream. Este resultado valida el criterio que utilizamos al analizar los resultados obtenidos durante este trabajo doctoral. Por último, nos preguntamos si $N = 100$ iteraciones de bootstrap eran suficientes para estimar el intervalo de confianza, dado que la literatura suele sugerir utilizar un N en el orden de los miles [222]. Se puede observar en la Figura 4.15 que si bien se agrandan los intervalos de confianza al utilizar más iteraciones, la diferencia es mínima a fines prácticos, y justifica la ganancia en tiempo de evaluación.

4.10. ¿Qué hiperparámetros afectaron más al desempeño?

En las distintas secciones de este capítulo, mostramos el efecto de cada hiperparámetro por separado, sin embargo, resulta quizás dificil para el lector darse una idea de cuáles hiperparámetros tuvieron un mayor efecto en el desempeño. En la Figura 4.16 comparamos el efecto en el desempeño que tienen los distintos hiperparámetros explorados, representando cada cruz una configuración distinta del hiperparámetro.

Se puede observar que el peso de enmascarado (σ) es el hiperparámetro que más afecta al desempeño del modelo, perjudicándole principalmente cuando $\sigma = 0$, y por ende, el modelo solo aprende a reconstruir segmentos no enmascarados. La representación de entrada es otro de los hiperparámetros que mayor efecto tuvo en el desempeño, siendo en general el modelo que utiliza EnCodec como entrada el que peor desempeño alcanza, con excepción de la tarea de clasificación de notas musicales (NS), en donde este modelo logra un desempeño mayor al resto.

Por otro lado, la cantidad de cuantizadores pesados (#Q), y la cantidad de capas del decoder, tienen un efecto menor en el desempeño, siendo comparable al efecto de la semilla utilizada para pre-entrenar los modelos.

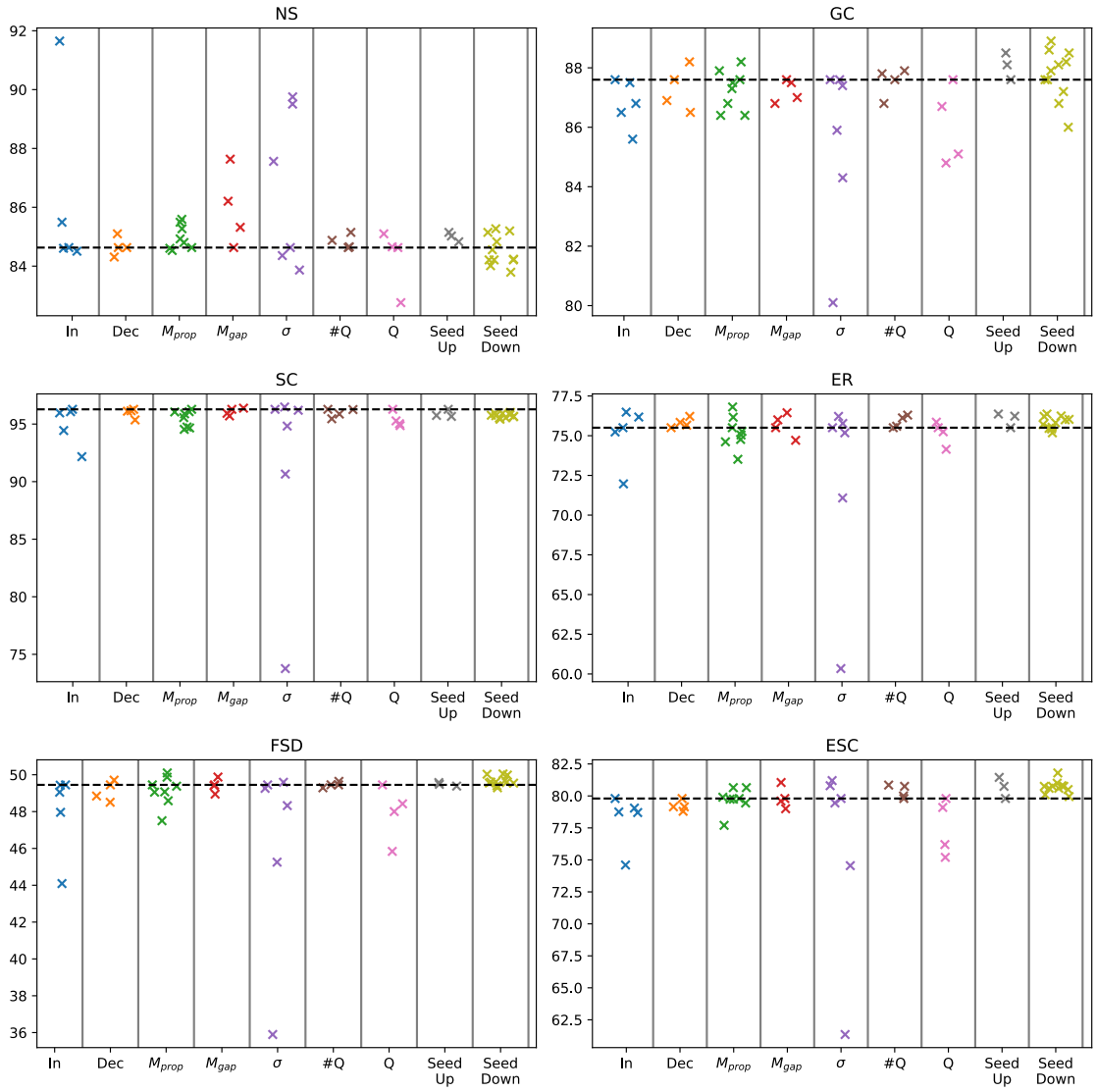

Figura 4.16: Efecto de los distintos hiperparámetros explorados en este capítulo. Cada cruz corresponde a un modelo con un valor distinto del hiperparámetro. La línea punteada representa el modelo base a partir del cual fuimos alterando cada hiperparámetro. Se muestran los efectos de la representación de entrada (In), cantidad de capas del decoder (Dec), proporción de enmascarado (M_{prop}), tamaño de gap (M_{gap}), cantidad de cuantizadores pesados (Q), cuantizador objetivo (Q), y semilla de upstream y downstream (Seed Up y Seed Down)

5

Alineamiento con representaciones cerebrales

> "No vemos las cosas tal como son, las vemos tal como somos"
>
> Anaïs Nin

5.1. Introducción y Motivación

Dado que los modelos de aprendizaje profundo logran resolver tareas que requieren de inteligencia, tiene sentido preguntarnos si emerge una estructura en sus representaciones internas similar a la de nuestras redes neuronales biológicas. Se ha observado, por ejemplo, que redes neuronales entrenadas para separación de hablantes, aprenden representaciones en su primer capa que siguen una escala de frecuencias similar a la mel, y poseen sensibilidad a la fase en baja frecuencia [74] replicando características de la percepción auditiva humana, sin haber sido explícitamente programadas para ello. A su vez, distintos trabajos [8, 223–226] han revelado que modelos preentrenados de audio son capaces de predecir la **señal dependiente del nivel de oxígeno en sangre (BOLD)** obtenida con una resonancia magnética funcional (fMRI) mejor que un modelo computacional clásico de la audición [227] y que sus capas exhiben una organización jerárquica similar a la del cerebro; las primeras capas son buenas predictoras de la actividad en la corteza auditiva primaria, mientras que las capas más profundas son buenas prediciendo la actividad en regiones no primarias asociadas a un procesamiento de más alto nivel, como del lenguaje. Una explicación de este fenómeno se da en [228], donde se plantea la **hipótesis de representaciones platónicas**. Se ha observado que distintas representaciones, provenientes de distintas modalidades como texto, imagen y audio, se vuelven cada vez más similares entre si a medida que estos modelos mejoran. Esto implica que los modelos estarían convergiendo a una representación común platónica, a partir de distintas observaciones o vistas de la realidad. Una de las posibles explicaciones que dan a esta convergencia es que al ser estos modelos capaces de resolver cada vez más tareas, existen menos representaciones que puedan resolverlas a todas

simultáneamente. Además, en general las tareas y datos utilizados para entrenar estos modelos son similares a los estímulos y tareas a las que estamos expuestos cotidianamente, llevando a un creciente alineamiento entre modelo y cerebro. Si bien explícitamente la tarea de pretexto suele ser una sola, optimizar por ejemplo la tarea de MLM puede pensarse como una optimización de múltiples tareas, del mismo modo que se piensa que los modelos de lenguaje al predecir el siguiente token están implícitamente resolviendo múltiples tareas [229]. Llevándolo al caso de MLM de audio, para completar un segmento faltante de habla es necesario conservar al hablante, y por ende detectarlo, entender el contexto de lo que se está diciendo para predecir los fonemas faltantes y no romper el estilo de habla y prosodia. Si fuera una señal de música, el modelo debe detectar qué instrumentos musicales estaban presentes, la ecualización general de la canción, estructura musical, etc.

Las redes neuronales artificiales (ANN) son consideradas buenas candidatas como modelos del procesamiento en el cerebro, ya que capturan varios aspectos del mismo: emergen comportamientos complejos a partir de la acción colectiva de unidades simples (neuronas artificiales); se optimiza la conectividad de un gran número de unidades en el tiempo para aprender; pueden procesar estímulos sensoriales como audio e imagen, sin necesidad de abstraerlos mediante reglas e ingeniería de atributos; y puede experimentarse con distintas funciones objetivo y arquitecturas que impondrán distintas restricciones en el procesamiento de los estímulos. De esta manera, se pueden probar hipótesis complejas sobre el procesamiento cerebral y evaluarlas en conjuntos de datos obtenidos a partir de mediciones cerebrales o de comportamiento, dando lugar al reciente campo del **neuroconexionismo** [1].

Para medir cuán bien los modelos de redes neuronales artificiales aproximan a los biológicos se realizan distintos tipos de análisis midiendo la correspondencia en el comportamiento, correspondencia con mediciones fisiológicas, electrofisiología in-silico y correspondencia en el desarrollo. Un ejemplo de análisis de correspondencia en el comportamiento consiste en verificar si distintos modelos exhiben los mismos metamerismos que los seres humanos, es decir, si el conjunto de estímulos que llevan a una misma representación en el modelo son también percibidos como pertenecientes a una misma categoría semántica por personas [230]. Respecto a la correspondencia con mediciones fisiológicas, se suelen utilizar técnicas como análisis de similaridad de representaciones (RSA) o predicción de actividad neuronal mediante modelos lineales regularizados, recolectando los datos neurológicos mediante neuroimágenes o mediciones electrofisiológicas. La electrofisiología in-silico consiste en experimentar sobre redes neuronales artificiales para entender mejor su funcionamiento; por ejemplo se pueden simular lesiones cerebrales removiendo conexiones en la ANN y ver cómo impacta en el comportamiento o medidas de correspondencia. Por último, la correspondencia en el desarrollo consiste en aplicar alguno de los métodos anteriores en distintos puntos del entrenamiento de la ANN y comparar estas trayectorias con distintas etapas del desarrollo humano.

5.1.1. Trabajo base

Nuestro trabajo extiende principalmente el estudio de Tuckute et al. [8] del año 2023 en el que analizaron la correspondencia entre distintos modelos de audio y la señal BOLD medida en la corteza auditiva. Para esto, utilizaron dos datasets de fMRI con mediciones realizadas mientras los sujetos escuchaban estímulos sonoros, y extrajeron activaciones de los distintos modelos utilizando como entradas los mismos estímulos sonoros. Luego, compararon estas representaciones con la cerebral utilizando RSA o prediciendo la señal de fMRI a partir de las distintas activaciones y midiendo el coeficiente de determinación. De esta manera, pudieron concluir que muchos de los modelos de audio analizados, a pesar de no haber sido entrenados explícitamente para aproximar al cerebro, exhiben una alta correspondencia con el mismo, y superan a modelos computacionales de la audición clásicos. Al mismo tiempo, al realizar un análisis por capas, observan que muchos de estos modelos exhiben una correspondencia estructural con la corteza auditiva—la corteza auditiva primaria se corresponde mejor con las primeras capas, y etapas más tardías del procesamiento con las últimas. Los modelos de audio que utilizaron en su análisis fueron:

- **Audio Spectrogram Transformer (AST) [86]**: como se explicó en la Sección 2.3.1, consiste de un Vision Transformer entrenado para la tarea de detección de eventos acústicos a partir de melspectrogramas.
- **DCASE2020 Baseline [231]**: consiste de una red GRU bidireccional entrenada para audio captioning con el conjunto de datos Clotho.
- **Deepspeech 2 [232]**: es una red con 2 capas convolucionales y 5 BiLSTM, entrenada para la tarea de reconocimiento de habla en el conjunto de datos LibriSpeech utilizando CTC como función objetivo.
- **MetricGAN [233]**: consiste de 2 capas BiLSTM seguidas por 2 capas completamente conectadas. El modelo se entrena para estimar máscaras espectrales y realizar reducción de ruido en el habla.
- **S2T [234]**: es una red Transformer de 12 capas entrenada para las tareas de reconocimiento de habla y traducción de habla. Los autores solo analizan el encoder de esta red.
- **SepFormer [235]**: es una red Transformer con 32 capas, algunas modelando dependencias de corto plazo (Intra-Transformer) y otras de largo plazo (Inter-Transformer). El modelo fue entrenado para separación de hablantes.
- **VGGish [66]**: como se explicó en la Sección 2.3.1, consiste de una red convolucional de tipo VGG entrenada para clasificar eventos acústicos a partir de melspectrogramas.
- **VQ-VAE [236]**: es un autoencoder variacional con cuantización vectorial entrenado para reconstruir habla representada mediante melspectrogramas. El encoder consiste de una red convolucional mientras que el decoder es una RNN. Solo se analizaron capas del encoder.

- **Wav2Vec 2.0 [75]**: como se explicó en la Sección 2.3.4, consiste de una red Transformer entrenada con una función de pérdida InfoNCE para distinguir las representaciones correspondientes a segmentos enmascarados de otras representaciones de audio. Particularmente analizaron la versión finetuneada para reconocimiento de habla en LibriSpeech.
- **Modelos in-house [8]**: estos modelos, diseñados por los autores, utilizan como entrada cocleagramas, los cuales se calculan con un banco de 211 filtros pasabanda imitando la respuesta del oído humano, seguido de una extracción de envolvente mediante una transformada de Hilbert, y por una compresión simulando la realizada por la membrana basilar. Se aplica un filtro pasabajo a las envolventes resultantes y se remuestrean a 200 Hz. Se utilizan 2 arquitecturas convolucionales que denominan CochCNN9 y CochResNet50, y se entrenan con un conjunto de datos en el que cada ejemplo consiste de una palabra con ruido de fondo proveniente de Audioset. Esto permite entrenar el modelo en 3 tareas utilizando los mismos datos: reconocimiento de palabra, identificación de hablante y detección de eventos acústicos (a partir del ruido de fondo). También experimentan entrenando el modelo para resolver las 3 tareas en simultáneo (multitask), y en una tarea de clasificación de género musical. Esto origina 10 modelos in-house al combinar las 2 arquitecturas con las 4 tareas y el multitasking.
- **Modelo espectro-temporal clásico [227]**: este modelo computacional de análisis auditivo es utilizado como baseline. El mismo se implementa con una serie de convoluciones 2D que analizan distintos patrones de modulación espectro-temporal en el cocleagrama de entrada. Estas convoluciones 2D son fijas y están diseñadas para capturar distintos fenómenos psicoacústicos y neurofisiológicos.

5.1.2. Aportes

Nuestros aportes expandiendo el trabajo original fueron:

- Incorporar modelos de audio más recientes, que exhiben un mejor desempeño en distintas tareas de audio que los modelos analizados en el trabajo original, los cuales son previos al 2022. Particularmente añadimos BEATs, Dasheng y EnCodecMAE, los cuales son entrenados de forma no supervisada en la tarea de modelado de lenguaje enmascarado y consisten en redes Transformer. En el trabajo original, sólo Wav2Vec 2.0 y VQ-VAE fueron entrenados sin etiquetas, aunque Wav2Vec 2.0 fue finetuneado en ASR. VQ-VAE fue entrenado exclusivamente con señales de habla, al igual que muchos de los modelos analizados en el trabajo original. En contraste, los modelos que incorporamos utilizan audios diversos provenientes del dominio del habla, música y sonidos ambiente.
- Agregar un análisis de correspondencia comportamental al medir la correlación entre el alineamiento con el cerebro y el desempeño de los modelos en distintas tareas downstream.

- Verificar si el alineamiento con el cerebro es una característica que emerge durante el preentrenamiento de EnCodecMAE. Particularmente, medimos el alineamiento a lo largo del preentrenamiento, observando que este crece a pesar de no optimizarse de manera explícita.

Una motivación para incorporar nuevos modelos de audio es que creemos que tanto utilizar la tarea de lenguaje enmascarado como utilizar datos de audio diversos, llevan a representaciones mejor alineadas con el cerebro, ya que son objetivos más similares a la experiencia humana —cotidianamente escuchamos sonidos diversos: desde una canción, una conversación hasta el sonido de la lluvia, animales, maquinarias, etc. A su vez, se cree que estamos constantemente prediciendo el futuro, o infiriendo información faltante, lo cual se alinea con la tarea de MLM. Un ejemplo muy estudiado en el campo del habla es el de la restauración fonémica [237]: si se reemplazan fonemas de una palabra por ruido, los sujetos tienden a completar mentalmente la información faltante y, en muchos casos, ni siquiera son conscientes de que hubo un reemplazo. Esta habilidad es muy útil especialmente en conversaciones con un nivel de ruido elevado, y la intensidad del efecto depende de la información presente en el contexto para desambiguar el fonema faltante [238, 239].

Los modelos analizados por Tuckute et al. difieren entre sí en muchos factores. Por esta razón, los autores sugieren en trabajos futuros explorar factores por separado para analizar su impacto en la correspondencia con la señal de fMRI. En este trabajo, al haber elegido estratégicamente los modelos a incorporar, podemos analizar algunos de estos factores:

- En EnCodecMAE variamos el conjunto de datos de preentrenamiento: sólo habla limpia, sólo música, sólo audios de YouTube (Audioset) o una combinación de los 3 tipos de datos.
- En EnCodecMAE y Dasheng variamos el tamaño del modelo sin alterar otros factores. Particularmente, EnCodecMAE presenta los tamaños Small (43.6M), Base (86.6M), y Large (261.6M). Por otro lado, Dasheng explora variantes con una mayor cantidad de parámetros: Base (86M), una variante con 600M y otra con 1.2B.
- En EnCodecMAE y BEATs se cuenta con modelos entrenados con y sin refinamiento iterativo de la señal objetivo, por lo que se puede explorar si este proceso incrementa el alineamiento con las representaciones cerebrales.
- En BEATs y Dasheng se cuenta con modelos finetuneados en la tarea de detección de eventos acústicos además de los preentrenados sin finetunear. Esto nos permite comparar el alineamiento con la señal BOLD entre un modelo que fue preentrenado sin supervisión y su versión especializada en una tarea particular.
- En EnCodecMAE contamos con checkpoints intermedios, con los cuales podemos analizar el alineamiento a lo largo del preentrenamiento.

Finalmente, tras obtener buenos resultados con modelos que utilizan cocleagramas en su entrada, los autores sugieren que modelos con los mismos sesgos y restricciones

perceptuales que un ser humano, podrían resultar en un mayor alineamiento con el cerebro. Del mismo modo, los autores obtuvieron representaciones mejor alineadas al utilizar modelos entrenados para resolver múltiples tareas, sugiriendo que esta estrategia promueve el alineamiento entre las representaciones y las mediciones de fMRI.

Este tipo de análisis también podemos realizarlo con EnCodecMAE, ya que contamos con variantes entrenadas con melspectrogramas, los cuales están inspirados en la percepción auditiva, y variantes utilizando espectrogramas tradicionales y EnCodec, los cuales no buscan aproximar el proceso de audición humana. Sin embargo, cabe destacar que EnCodec podría haber aprendido una representación alineada a la audición humana, como se ha observado en otros modelos entrenados sobre formas de onda en tareas de audio [74].

Quizás el resultado más novedoso de nuestro trabajo es haber determinado que existe una correlación entre el desempeño de los modelos de audio en ciertas tareas downstream de HEAREval y el alineamiento con el cerebro. Este resultado replica el de trabajos en otros dominios como NLP. Por ejemplo, en [240] los autores muestran que a medida que distintos LLMs tienen un mejor desempeño en el benchmark Massive Multitask Language Understanding (MMLU) [241], también exhiben un mayor alineamiento con el cerebro. Hasta donde conocemos, este tipo de análisis no se ha realizado previamente con modelos de audio, y nos permite ganar una intuición de qué tareas deberíamos optimizar para generar representaciones mejor alineadas con las de la corteza auditiva.

5.2. Metodología

La metodología aplicada en este trabajo es la misma utilizada por Tuckute et al. en el trabajo original. A continuación se describe la misma.

5.2.1. Conjuntos de datos y preprocesamiento

Al igual que en el trabajo original, utilizamos dos conjuntos de datos de resonancia magnética funcional (fMRI):

- **NH2015** [242]: contiene fMRIs de 8 participantes sin experiencia musical, hablantes nativos de inglés y con edades entre 19 y 25 años. Los participantes escucharon $N = 165$ estímulos sonoros de 2 segundos de duración por sesión, mientras se les realizaba el fMRI. Estos estímulos corresponden a sonidos cotidianos y se presentan en bloques en los que se repite 5 veces cada estímulo. Cada participante completó 3 sesiones de aproximadamente una hora y media de duración, y cada sesión se dividió en 11 partes con 15 bloques de estímulo cada una, y 4 de silencio. A su vez, se monitoreó que los participantes estuvieran prestando atención haciendo que una de las repeticiones del estímulo esté 7 dB más baja que el resto. Los sujetos debían presionar un pulsador cuando este estímulo era presentado.

- **B2021** [243]: contiene fMRIs de 20 participantes, 10 con formación musical (8 de sexo femenino y 2 masculino, con un promedio de 23.5 años de edad y una desviación estándar de 3.3 años, y con entre 11 y 23 años de formación musical) y 10 sin ella (6 de sexo femenino y 4 masculino, con un promedio de 25.8 años de edad y una desviación estándar de 4.1 años). Los estímulos sonoros son los mismos que los utilizados en NH2015, y el diseño de los experimentos es similar. Las principales diferencias radican en que cada estímulo se repitió 3 en vez de 5 veces por bloque, se utilizaron 2 bloques por estímulo (en total cada estímulo se presenta 6 veces por sesión), se organizó el experimento en 16 partes, y se usó un estímulo 12 dB más bajo que el resto para monitorear la atención de los sujetos.

En ambos conjuntos de datos se registró mediante fMRI el nivel de oxígeno en sangre (BOLD) en cada voxel de una región del cerebro incluyendo la corteza auditiva. Se midió con una orientación paralela al plano temporal superior cubriendo la porción desde el lóbulo temporal superior al surco temporal superior. Para cada sesión, participante y estímulo, se descarta la primera adquisición, y en el caso de NH2015 se promedian las 4 respuestas restantes correspondientes a las repeticiones del estímulo. Para B2021, en lugar de realizar un promediado se utilizó un modelo lineal generalizado (GLM). A partir de estas señales combinadas se calcula un Percent Signal Change (PSC) restando y dividiendo por la respuesta al silencio de cada voxel. Finalmente, se realiza un promedio en el tiempo resumiendo la actividad en un escalar.

Luego, para cada participante se seleccionaron aquellos voxels que tengan una respuesta significativamente distinta a la del silencio utilizando un t-test ($p < 0.001$) y que respondan de manera consistente a lo largo de las 3 sesiones. Para esto último, se calcula la siguiente medida de reliability para cada participante:

$$rel = 1 - \frac{||v_{12} - \frac{v_3 \cdot v_{12}}{||v_3||_2^2} v_3||_2}{||v_{12}||_2}$$

En el caso de NH2015, v_{12} es la respuesta de un voxel a los 165 estímulos promediada en las dos primeras sesiones, y v_3 la respuesta del mismo voxel obtenida en la tercer sesión. En B2021, v_{12} es la respuesta de un voxel a los 165 estímulos estimada con las primeras 3 repeticiones del estímulo en las primeras 24 partes, y v_3 la respuesta del mismo voxel obtenida con las respuestas a las últimas 3 repeticiones adquiridas en las partes 24 a 48. Para cada participante se incluyeron aquellos voxels con un $rel > 0.3$, y este proceso de selección resultó en un promedio de 961.75 voxels por participante (rango de 637 a 1221) en NH2015, y 1340 (rango 1020 a 1828) en B2021. Finalmente, la actividad de los voxels seleccionados se promedia entre sesiones.

Por otro lado, para cada modelo de audio, se extraen y promedian en el tiempo las activaciones de distintas capas intermedias para cada uno de los N estímulos.

De esta forma, se obtienen, para una cierta capa l de un modelo de audio m y un cierto participante s:

- **Representación de audio**: es un tensor $X_{m,l} \in \mathbb{R}^{D_m \times N}$, en donde D_m es la

dimensión de las activaciones de la capa l del modelo m y N es la cantidad de estímulos sonoros.

- **Representación de fMRI**: es una matriz $Y_s \in \mathbb{R}^{V_s \times N}$ con las respuestas promediadas en el tiempo y entre sesiones de los V_s voxels seleccionados para el participante s para los N estímulos sonoros.

Se utilizaron dos técnicas para medir el alineamiento entre las representaciones de los modelos de audio y las representaciones del cerebro: un análisis de regresión y un análisis de similaridad de representaciones (RSA). Ambas técnicas son utilizadas ampliamente en la literatura y proporcionan información complementaria sobre el alineamiento de las representaciones. En la Figura 5.1 se muestran de manera resumida las dos técnicas de análisis y a continuación se describe la metodología utilizada para ambos análisis, la cual se corresponde con los métodos utilizados en [8].

5.2.2. Análisis de regresión

En este análisis se intenta predecir la representación de cada vóxel del fMRI a partir de la representación del modelo de audio. Se utiliza un regresor ridge (con regularización L2), cuyo peso de regularización α es elegido mediante una búsqueda de hiperparámetros. Se replica la metodología del trabajo de Tuckute et al., la cual consistió en generar aleatoriamente subconjuntos de entrenamiento y evaluación, utilizando 83 y 82 estímulos respectivamente. Para elegir el valor de α, se utiliza validación cruzada Leave-one-out: para un cierto valor de α se realiza una predicción para cada uno de los 83 estímulos del conjunto de entrenamiento, entrenando con los 82 restantes. Este procedimiento origina un conjunto de 83 predicciones sobre el cual se calcula el coeficiente de determinación de Pearson r^2 para el valor de α, y se repite para cada uno de los 100 valores de α explorados, los cuales están espaciados logarítmicamente entre 10^{-50} y 10^{49}. Finalmente, se elige el valor de α que resulta en el mejor valor de r^2. Para el valor de α elegido se entrena el regresor ridge sobre el conjunto de entrenamiento completo y se obtiene un valor de r^2 en el conjunto de evaluación. Este procedimiento se repite $R = 10$ veces generando distintos conjuntos de entrenamiento y evaluación de manera aleatoria, y se reporta la mediana a lo largo de estas 10 repeticiones. Este proceso se repite para cada uno de los V_s voxels de cada sujeto s y las L_m capas del modelo de audio. Es decir, cada regresor ridge se entrena para predecir la actividad de un voxel específico a partir de las representaciones de una capa del modelo de audio. Tanto las representaciones de audio como los valores en los voxels son centrados, calculando la media en el conjunto de entrenamiento.

Cabe destacar el alto costo computacional de este proceso, en el que se entrenan $R \cdot L_m \cdot V \cdot 100 \cdot 83$ modelos, en donde típicamente $L_m = 12$ y para NH2015, $V = 7694$. El proceso completo demora unas 8 horas y media con un procesador AMD Ryzen 9 con 16 núcleos, para EnCodecMAE Base, el cual cuenta con 10 capas.

Una vez realizados los entrenamientos y evaluaciones, se obtiene un valor $r^2_{s,l,v}$, el cual indica cuánta de la varianza en el voxel v del sujeto s puede ser predicha con un regresor lineal ridge a partir de las activaciones de la capa l del modelo de audio m.

Como la señal de fMRI es ruidosa, ya que por ejemplo algunos voxels se encuentran más cerca que otros de las bobinas del resonador, siguiendo la metodología del trabajo original, se realiza una corrección por atenuación de los valores de r^2, obteniéndose $\tilde{r}^2_{s,l,v}$. La corrección por atenuación se explica en el Anexo C.

5.2.3. Análisis de regresión por componentes

Además de hacer regresión directamente sobre los voxels, en el trabajo de Tuckute et al. se realiza regresión sobre componentes independientes obtenidas en un estudio

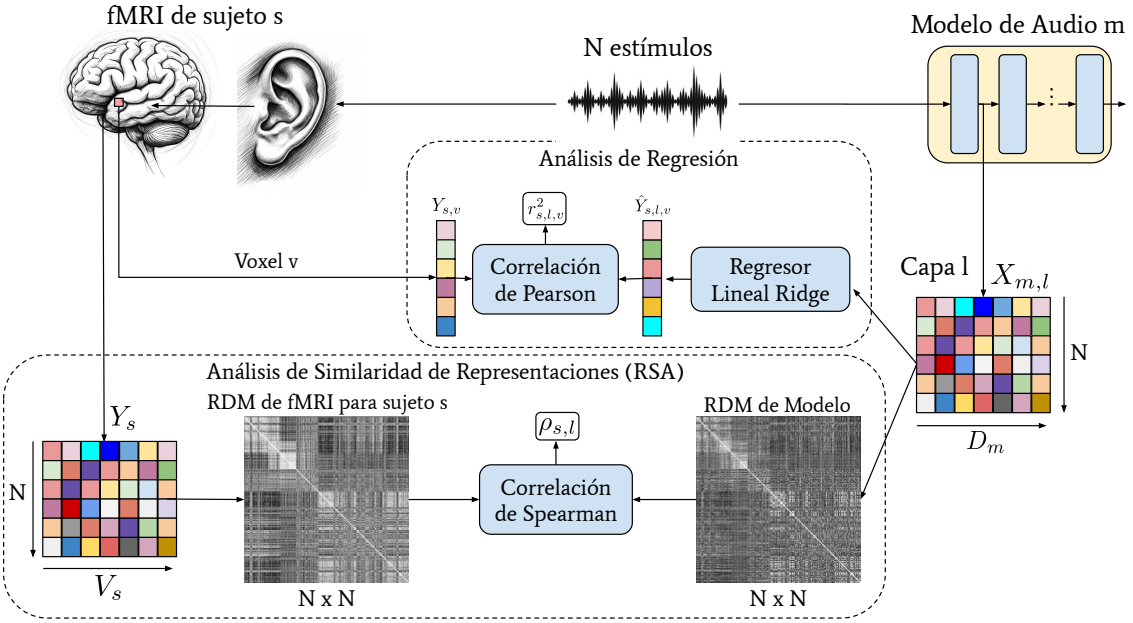

Figura 5.1: Diagrama conceptual de los análisis realizados. En ambos se utiliza un conjunto de datos con N estímulos sonoros y las correspondientes mediciones de fMRI con las respuestas del sujeto s para cada uno de los V_s voxeles. A partir de los estímulos sonoros se extraen activaciones de cada capa del modelo de audio. En el análisis de regresión se intenta predecir la actividad resumida $Y_{s,v}$ para un voxel v y sujeto s ante cada estímulo, dadas las activaciones X_l de la capa l del modelo de audio para los mismos estímulos. El coeficiente de determinación $r^2_{s,l,v}$ entre la actividad predicha $\hat{Y}_{s,l,v}$ y $Y_{s,v}$ se usa como medida de alineamiento entre el voxel v del fMRI del sujeto s y la capa l del modelo de audio m. En el análisis de similaridad de representaciones, se calculan matrices de disimilaridad (RDM) para las activaciones de la capa l y para las mediciones del fMRI de un sujeto s, que indican cuan disimilares (utilizando correlación de Pearson) son cada posible par de estímulos en ambos espacios de representación. Finalmente, se calcula la correlación de Spearman $\rho_{s,l}$ entre las dos RDM con dimensiones $N \times N$, y esta métrica sirve como medida de alineamiento entre la capa l del modelo m y el sujeto s. Notar que, mientras que el análisis de regresión produce un valor de alineamiento por voxel, RSA produce un solo valor global, sobre todos los voxels.

previo [242]. En este estudio encuentran que con 6 componentes independientes obtenidas mediante una variante de análisis de componentes independientes (ICA), y mostradas en la Figura 5.2, es posible explicar el 80% de la varianza de los voxels en el conjunto de datos NH2015 realizando combinaciones lineales de las 6 componentes. De esta forma, se puede simplificar el análisis de regresión de voxels, prediciendo el valor de esas 6 componentes en lugar del valor de cada voxel individual. Además, los autores lograron identificar a qué tipos de estímulos responde cada una de estas 6 componentes:

- **Componente 1 (LF)**: da mayor peso a voxels que respondieron a tonos puros de baja frecuencia en una medición de tonotopía.
- **Componente 2 (HF)**: de forma similar a la componente 1, da mayor peso a voxels que respondieron a tonos puros de alta frecuencia en una medición de tonotopía.
- **Componente 3 (Broadband)**: los estímulos que generan una mayor respuesta en esta componente se caracterizan por tener un espectro de banda ancha y tener modulaciones temporales rápidas, exhibiendo patrones verticales en un espectrograma.
- **Componente 4 (Pitch)**: los estímulos que generan una mayor respuesta en esta componente se caracterizan por ser tonales, exhibiendo líneas horizontales en un espectrograma.
- **Componente 5 (Habla)**: esta componente muestra una respuesta selectiva al habla, siendo los estímulos categorizados como habla los que más activan esta componente, seguidos de música con voz cantada y vocalizaciones.
- **Componente 6 (Música)**: esta componente muestra una respuesta selectiva a la música.

Cabe destacar que ordenamos las componentes de la misma manera que los autores originales, para ser consistentes, y que al ser obtenidas con un análisis de componentes independientes (ICA), no se encuentran ordenadas por varianza explicada como en el caso de PCA. A su vez, a diferencia de la regresión de voxels, no se realiza una corrección por atenuación sino que se reporta directamente el r^2 obtenido mediante correlación de Pearson, y las componentes no están disponibles para cada sujeto sino que son calculadas sobre el promedio entre sujetos de los voxels.

5.2.4. Análisis de similaridad de representaciones

Dada una matriz M, la matriz de disimilaridad (RDM) está dada por:

$$D_{ij} = 1 - r(M_i, M_j),$$

en donde $r(M_i, M_j)$ es el coeficiente de correlación de Pearson entre M_i y M_j que corresponden a las filas i y j de la matriz M.

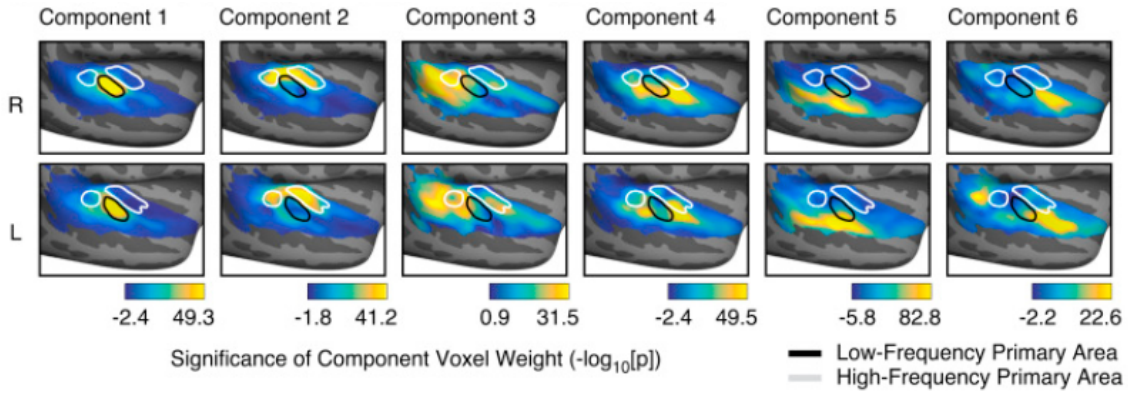

Figura 5.2: Pesos asociados a los componentes encontrados en [242].

Luego, para realizar el análisis de similaridad de representaciones (RSA), se calculan RDMs para $M = X_{m,l}$ y para $M = Y_s$. Ambas matrices M son normalizadas (z-score) antes de calcular las RDM. Dadas las dos matrices de disimilaridad, se calcula la correlación de Spearman ρ entre ellas, resultando en una métrica que mide alineamiento entre las representaciones de la capa l de un modelo m y las de fMRI de un sujeto s. Para obtener un sólo valor de alineamiento para un cierto modelo m, se elige la capa con mejor alineamiento (mayor correlación de Spearman) utilizando los 83 datos de entrenamiento. A continuación, se repite el proceso 10 veces para cada sujeto (generando distintas divisiones de datos de entrenamiento y evaluación, como para el modelo de regresión), y se calcula la mediana del ρ de Spearman obtenido en los distintos splits. Este procedimiento se repite para cada sujeto por separado y finalmente se reporta el promedio del coeficiente resultante sobre todos los sujetos.

Para obtener una cota superior de similaridad, se compara el RDM del fMRI de un sujeto held-out contra el RDM promedio de los demás sujetos utilizando las mismas 10 divisiones de datos y tomando la mediana del ρ de Spearman. Este proceso se repite dejando a cada sujeto como held-out y luego se reporta el promedio del ρ de Spearman. Como cota inferior, se utiliza el ρ de Spearman obtenido con el modelo espectro-temporal clásico (ver sección 5.1.1).

5.3. Resultados

A continuación se muestran distintos resultados que expanden el trabajo original de Tuckute et al. agregando modelos de audio más recientes y analizando el efecto del desempeño downstream y cantidad de parámetros.

5.3.1. Análisis de regresión

El primer análisis que realizamos consiste en comparar EnCodecMAE, BEATs y Dasheng con los modelos utilizados por el trabajo original, en términos de la capacidad general que tienen de predecir la representación del fMRI. Para obtener

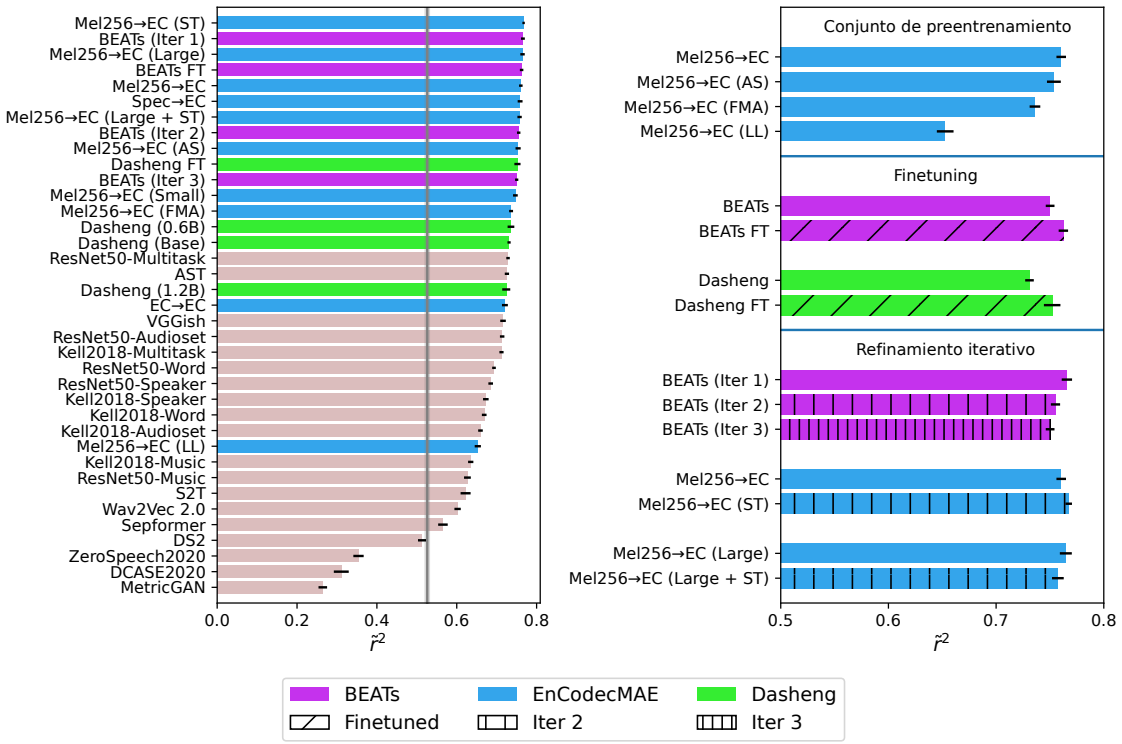

Figura 5.3: Izquierda: $\tilde{r}^2$ para distintas representaciones en el dataset NH2015. Se muestra el desempeño de un modelo computacional de la audición [227] como una línea gris. Derecha: detalle comparando modelos que sólo difieren en el conjunto de datos de preentrenamiento, en el finetuning, y en la cantidad de iteraciones de refinamiento de la señal objetivo.

un valor único por modelo que resuma su capacidad de predecir la actividad cerebral se realiza el siguiente procedimiento:

1. Se utilizan 5 de las divisiones aleatorias para seleccionar la mejor capa de cada voxel, y las otras 5 para reportar el valor de $\tilde{r}^2$ en cada voxel utilizando la capa seleccionada. Se repite este proceso 10 veces seleccionando distintas divisiones para elegir la mejor capa y para evaluar, y se reporta la media a lo largo de estas 10 iteraciones, dando lugar a un $\tilde{r}^2_{s,v}$ resumen de todas las capas l del modelo de audio.
2. Se toma la mediana de $\tilde{r}^2_{s,v}$ sobre todos los voxels v de cada sujeto s. Esto resulta en un $\tilde{r}^2_s$ que resume $\tilde{r}^2$ por sujeto.
3. Finalmente, se calcula la media de los $\tilde{r}^2_s$ combinando los resultados obtenidos para todos los sujetos.

Debido a que este análisis es el más costoso en términos computacionales, solo lo realizamos en el conjunto de datos NH2015. Del mismo modo, para los modelos que poseían muchas capas, en lugar de calcular el alineamiento en todas las capas, lo calculamos en capas equiespaciadas. De esta manera, la cantidad de capas analizadas es más uniforme entre modelos de distinto tamaño. Específicamente, para las versiones

Large de EnCodecMAE utilizamos las capas impares (1,3,5,...,19) y la última capa, para Dasheng 0.6B y 1.2B usamos 1 cada 4 capas (1,5,9,13,...,K) con $K = 29$ para Dasheng 0.6B y $K = 37$ para Dasheng 1.2B e incorporando también la última capa (32 para Dasheng 0.6B y 40 para Dasheng 1.2B).

En la Figura 5.3 (izquierda) se pueden observar los resultados obtenidos para los distintos modelos analizados. Se puede ver que los modelos introducidos en el análisis (barras azules, violetas y verdes), superan a la mayoría de los modelos analizados en el trabajo original (barras rosas). Esto indica que los modelos más recientes, los cuales exhiben un mejor desempeño general en tareas de audio, efectivamente se están alineando mejor con representaciones cerebrales.

Algunos resultados interesantes que se pueden ver con mayor detalle en la Figura 5.3 (derecha) son:

- El conjunto de preentrenamiento es importante. Cuando sólo se utiliza habla limpia (LibriLight) para entrenar EnCodecMAE, el alineamiento es muy inferior a cuando se utiliza señales musicales (FMA) o una mezcla de música, sonido ambiente y habla limpia. Particularmente, los modelos que utilizan una mezcla de datos o Audioset son los que mejor alineamiento exhiben. Una explicación de por qué Dasheng posee un peor alineamiento (una de las barras verdes en la Figura 5.3) a pesar de ser un modelo nuevo de buen rendimiento puede provenir de que utiliza un conjunto de preentrenamiento en donde ACAV100M [244] predomina (constituye el 96.6% de los datos de preentrenamiento). Este conjunto de datos puede introducir un sesgo debido a cómo fue construido. El proceso de recolección consistió en seleccionar videos de YouTube en los que la información mutua entre el audio y la imagen sea elevada. Esto implica que eventos donde el sonido e imagen estén muy acoplados, como una voz frontal o cantantes, serán sobrerepresentados, mientras que sonidos que suelen ocurrir "de fondo", como voces de fondo, ruido urbano, lluvia o música de fondo pueden ser ignorados por esta metodología de recolección.
- Realizar finetuning en la tarea de detección de eventos acústicos parece llevar a un incremento de similaridad entre la representación del modelo y la de la corteza auditiva. Esto puede deberse a que las categorías de Audioset son muy diversas y representan un gran rango de sonidos, a los que los 165 estímulos pertenecen. Al ajustarse el modelo para resolver esta tarea, se le está informando de qué diferencias sonoras llevan a un significado distinto para un ser humano. A modo de ejemplo, podríamos decir que el golpe de una puerta y el golpe de un bombo son espectrotemporalmente similares. Sin embargo, al ser útil la discriminación de ambos sonidos en la vida cotidiana, y por ende, ser parte de las clases de Audioset, se está informando implícitamente a la representación sobre cómo discriminar sonidos que pertenecen a distintas categorías semánticas.
- Para BEATs y EnCodecMAE Large, se observa que el alineamiento empeora ligeramente a medida que se refina iterativamente la señal objetivo. Por el contrario, para EnCodecMAE Base mejora con el refinamiento de la señal objetivo. En ambos casos, las diferencias son pequeñas y no parece haber una

tendencia clara.

También se puede observar que modelos con un mayor número de parámetros no necesariamente se alinean mejor con las representaciones cerebrales. Por ejemplo, Dasheng 1.2B contiene el doble de parámetros que Dasheng 0.6B y unas 14 veces más de parámetros que Dasheng Base, sin embargo, es el modelo de Dasheng que peor se alínea con representaciones cerebrales, siendo superado incluso por algunos modelos analizados en el trabajo original de Tuckute como AST. En la sección 5.3.4 se realizará un análisis más completo de la interacción entre el tamaño del modelo y la similaridad con las representaciones cerebrales.

5.3.2. Análisis de regresión por componentes

Se puede hacer un análisis de la capacidad de predicción de actividad en la corteza auditiva descomponiendo el fMRI en las 6 componentes descritas en la sección 5.2.3. Esto tiene algunas ventajas y desventajas:

- Las componentes son más interpretables que voxels específicos y podemos hacer un análisis de qué modelo predice mejor cada componente.
- El análisis es más eficiente al reducirse la cantidad de variables a predecir a 6.
- Esta simplificación trae la desventaja de que hay aproximadamente un 20% de la varianza en el fMRI que no se está considerando.
- Otra desventaja es que se pierde información de la región anatómica que se está prediciendo.

En la Figura 5.4 se puede observar el desempeño de los distintos modelos de audio a la hora de predecir las 6 componentes. Resulta interesante que para las primeras componentes, las cuales responden a características espectrales como contenido en alta y baja frecuencia, los mejores modelos no son los más recientes, sino que los que utilizan cocleagramas como representación de entrada. Por otro lado, para las componentes selectivas a estímulos de habla y música, los mejores modelos son los más recientes que incorporamos en este estudio. Este resultado parecería indicar que estos modelos, que son entrenados con mayores cantidades y diversidad de datos de manera no supervisada y utilizando redes transformers, son mejores predictores de estas componentes selectivas que los modelos analizados en el trabajo original. Otro resultado interesante es que, a diferencia de lo que uno esperaría, los modelos entrenados sólo con música o habla, Mel256→EC Base (FMA) y Mel256→EC Base (LL), no son los mejores predictores de las componentes de música y habla respectivamente. Los mejores modelos son los entrenados con una combinación de bases de datos. Del mismo modo que el modelo entrenado solo con habla tenía un desempeño bajo a la hora de predecir la actividad en voxels, también lo tiene a la hora de predecir la componente de habla.

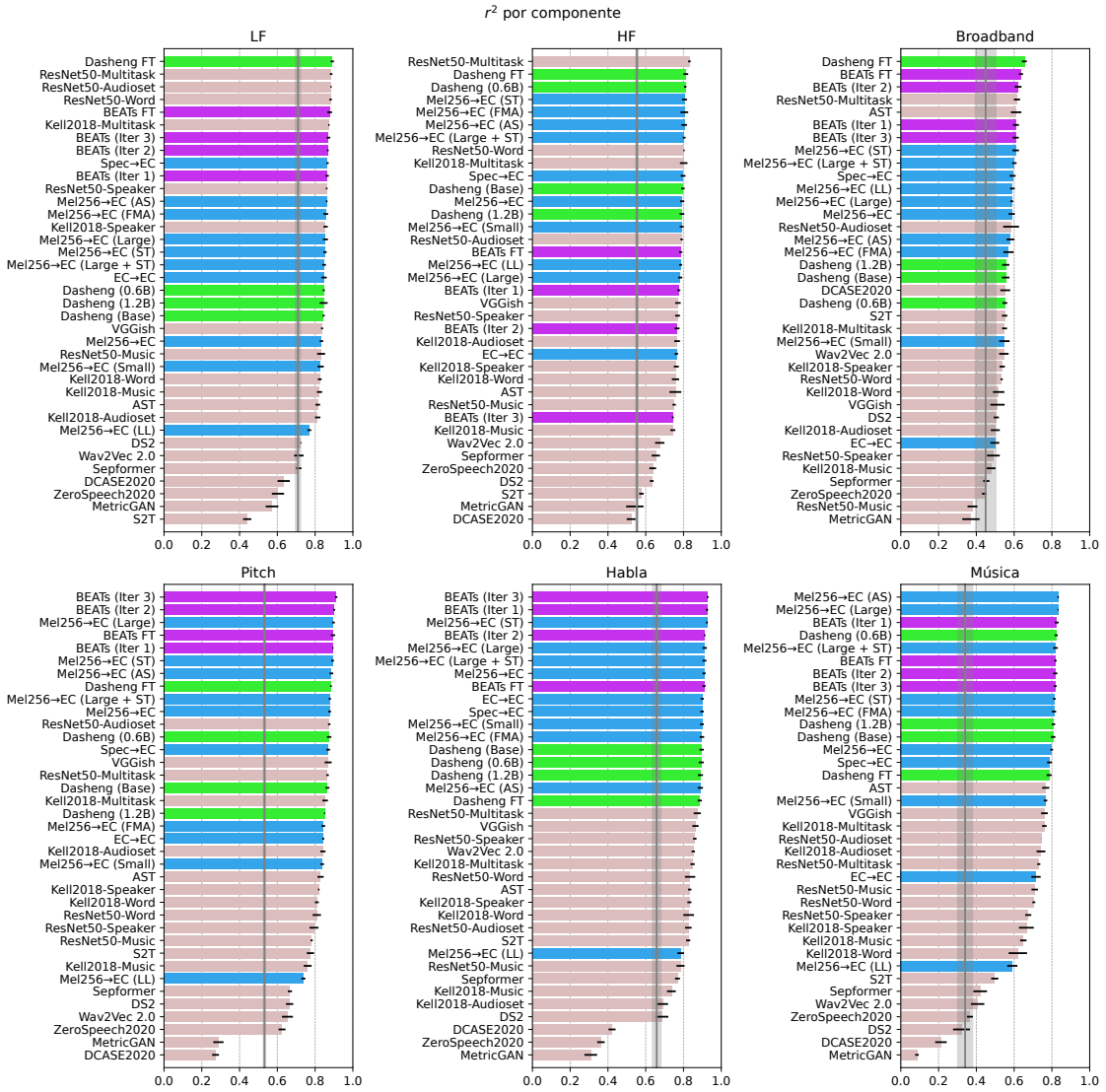

Figura 5.4: r^2 para las distintas componentes y modelos de audio analizados.

5.3.3. Análisis de similaridad de representaciones

Se puede observar en la Figura 5.5, que al igual que en el análisis de regresión de la sección 5.3.1, los modelos más recientes también son los que mayor similaridad presentan al usar RSA, y los modelos entrenados solamente con habla como MetricGAN, DeepSpeech2, VQ-VAE y Wav2Vec2 son los menos similares. Particularmente, la versión Large de EnCodecMAE es la que mayor similaridad alcanza en ambos conjuntos de datos. Cabe destacar que, a diferencia de los resultados de regresión, algunos modelos del trabajo original, como CochResNet50-MultiTask, alcanzan un nivel de similaridad superior a muchos modelos recientes como BEATs y Dasheng. Otros resultados que se conservan respecto al análisis de regresión son:

- EnCodecMAE preentrenado en conjuntos de datos diversos posee un mejor

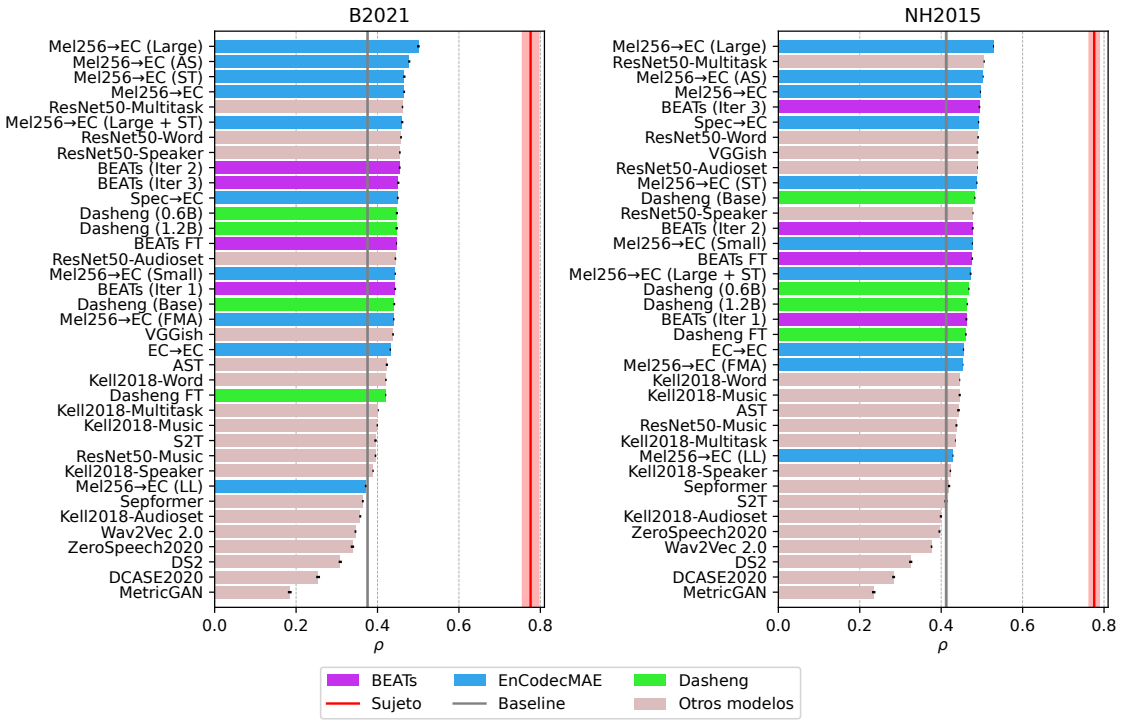

Figura 5.5: Correlación de Spearman entre RDMs de humano y modelo para la mejor capa.

alineamiento que los modelos preentrenados solamente con habla o música.

- Como se puede observar en los modelos de Dasheng, no es claro que incrementar el tamaño se traduzca en un mayor alineamiento, a pesar de que esto resulte en modelos con mejor desempeño downstream.
- El refinamiento de la señal objetivo, tanto en BEATs como en EnCodecMAE, no parece tener un efecto importante en la similaridad.

En la Figura 5.6 se puede observar un ejemplo de la RDM obtenida a partir del fMRI, y la obtenida a partir del modelo que exhibió mayor similaridad (EnCodecMAE Large). Los valores reportados en la Figura 5.5 corresponden a la correlación de Spearman entre estas matrices de disimilaridad. Se puede ver que las matrices mostradas en la figura muestran patrones similares, encontrando similitud (puntos más claros) entre estímulos del mismo tipo, y por ende resultando en un ρ de Spearman alto. Por ejemplo, para los estímulos correspondientes a la música instrumental, los mismos estímulos muestran valores altos de disimilitud (rayas más oscuras en la esquina izquierda superior) para el fMRI y la capa 4 del modelo de EnCodecMAE.

5.3.4. Efecto del tamaño de los modelos

Se ha observado, tanto en el trabajo de representaciones platónicas [228] (ver Sección 5.1), como en trabajos utilizando LLMs, que la cantidad de parámetros exhibe una correlación alta con el alineamiento de un modelo con otros que trabajan

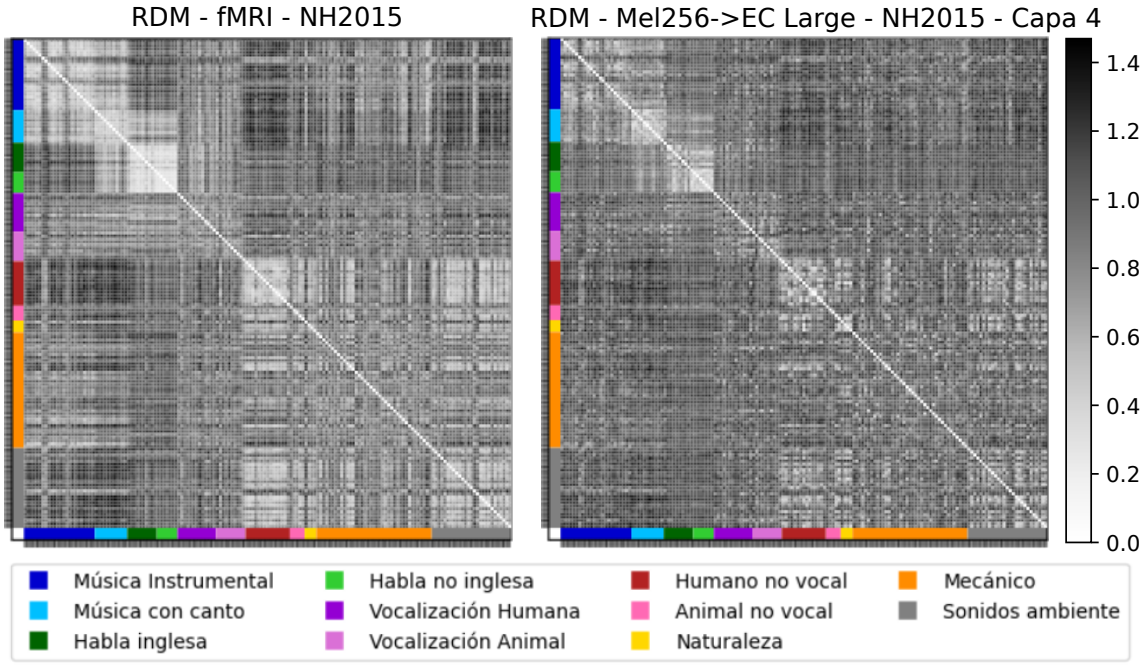

Figura 5.6: Ejemplo de RDM obtenida a partir de las mediciones de fMRI y la obtenida a partir del modelo que exhibió mayor similaridad. Las filas y columnas corresponden a los 165 estímulos, que fueron agrupados por su tipo, indicado por el color.

con distintas modalidades, y con representaciones cerebrales. Por este motivo, decidimos explorar esta variable e intentar responder a la pregunta: ¿se alinean más las representaciones de los modelos de audio con las de la corteza auditiva al incrementar la cantidad de parámetros?

Para responder esta pregunta, consideramos los modelos de la Figura 5.3 y visualizamos en la Figura 5.7 la cantidad de parámetros que poseen contra el alineamiento basado en regresión lineal ($\tilde{r}^2$). Se puede observar una tendencia global a que modelos con mayor cantidad de parámetros se alinean mejor con el cerebro. Específicamente, ambas variables exhiben una correlación de Pearson significativa de 0.736. Sin embargo, se puede observar que para tamaños compartidos por varios modelos, hay una dispersión importante. A su vez, para los modelos en los que sólo alteramos el tamaño, no siempre se observa que a mayor cantidad de parámetros el alineamiento es mayor. Por ejemplo, si bien Dasheng exhibe un mayor alineamiento al pasar de 80M a 0.6B de parámetros, su alineamiento decrece a un nivel inferior al de 80M al incrementar a 1.2B de parámetros. Lo mismo ocurre con la versión de EnCodecMAE que utiliza refinamiento iterativo de la señal objetivo, la cual ve su $\tilde{r}^2$ disminuído al aumentar la cantidad de parámetros.

Como observamos previamente, para una misma cantidad de parámetros se pueden obtener distintos niveles de desempeño, ya que el mismo también depende de otros factores como la calidad de los datos de preentrenamiento, la función de costo y la arquitectura utilizada. Hipotetizamos que, en realidad, no es el tamaño del

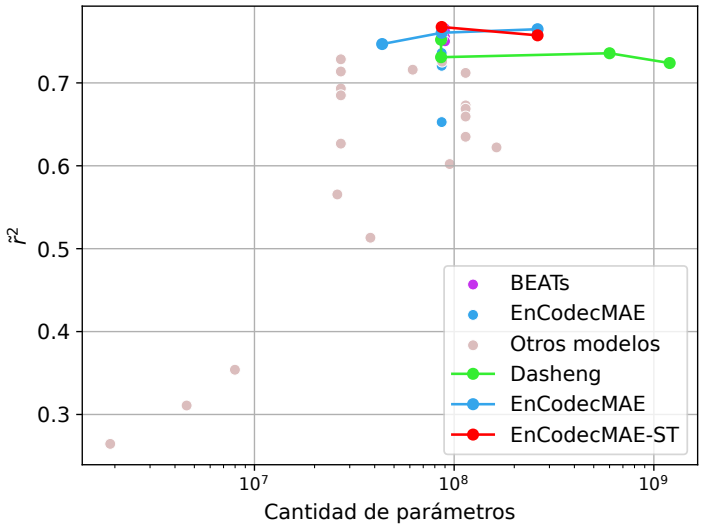

Figura 5.7: Diagrama de dispersión mostrando la cantidad de parámetros de cada modelo contra su alineamiento con las representaciones de la corteza auditiva ($\tilde{r}^2$). Para los modelos en los que sólo variaron la cantidad de parámetros, manteniendo el resto de sus características constantes, trazamos líneas sólidas.

modelo sino su utilidad para resolver tareas downstream lo que mejor correlaciona con el alineamiento con el fMRI.

5.3.5. Efecto de la calidad de las representaciones

En consecuencia, decidimos medir la calidad de las representaciones de la Figura 5.3 para poder realizar un análisis de correlación entre la calidad de las representaciones y el alineamiento con la corteza auditiva. Para medir la calidad, conservamos las técnicas de evaluación utilizadas en los capítulos 3 y 4, evaluando en las tareas de clasificación de notas musicales (NS), clasificación de género musical (GC), reconocimiento de emociones en el habla (ER), reconocimiento de comandos de voz (SC), detección de eventos acústicos (FSD) y clasificación de eventos acústicos (ESC). Introducimos dos diferencias en la metodología:

1. En lugar de utilizar solo la representación de la última capa, utilizamos las representaciones de las capas usadas en la predicción de la señal BOLD. Para no tener que explorar todas las capas y elegir la de mejor desempeño, y dado que distintos voxels pueden estar asociados a distintas capas, aprendemos pesos para la representación de cada capa y realizamos un promedio pesado de las mismas. Luego, la metodología es equivalente a la evaluación de HEAREval, solo que la entrada es un promedio pesado de representaciones en lugar de ser la salida de la última capa. Una dificultad que surge es que en algunos modelos, la dimensionalidad de las representaciones varía según la capa. En estos casos aprendemos las componentes principales de las representaciones de cada capa

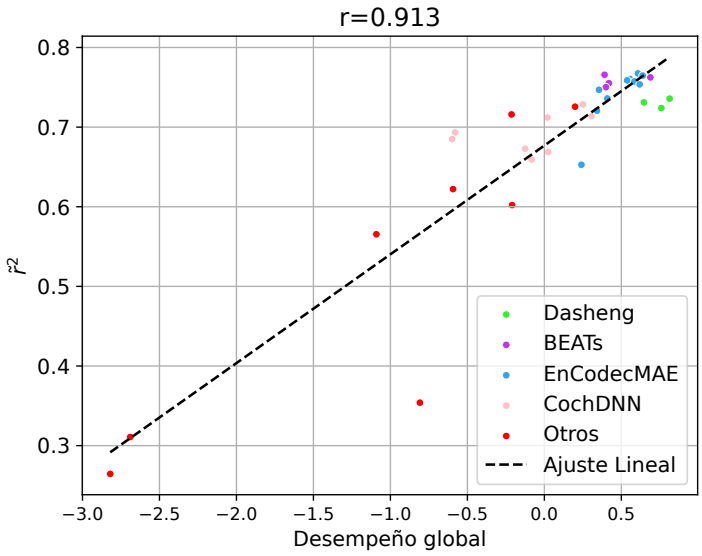

Figura 5.8: Relación entre el desempeño de los modelos analizados en tareas downstream y el alineamiento con la corteza auditiva mediante regresión de voxeles ($\tilde{r}^2$) para el dataset NH2015.

usando el conjunto de entrenamiento de cada tarea downstream, y calculamos la cantidad de dimensiones D^i_{90} necesarias para explicar el 90% de la varianza de cada representación. Finalmente utilizamos como representación de cada capa las primeras $\min(\max_i(D^i_{90}), D_{min})$ componentes, en donde D_{min} corresponde a la cantidad de dimensiones en la capa con menor dimensionalidad. Una vez aplicado PCA, dado que las dimensiones de las representaciones de cada capa coinciden, podemos entrenar el promedio pesado de capas junto con el modelo downstream.

2. Para calcular una medida de desempeño global, en lugar de utilizar la media y desviación estándar de los modelos del leaderboard de HEAREval, calculamos esas estadísticas a partir de los modelos analizados. A diferencia de análisis anteriores, no limitamos los puntajes de cada tarea al rango [-1, 1] antes de promediarlos, ya que en este caso no buscamos comparar directamente el desempeño global entre modelos. Además, restringir el rango de solo una de las dos variables podría introducir un sesgo en el análisis de correlación con la medida de alineamiento cerebral y distorsionar la relación lineal en caso de que exista.

En la Figura 5.8 se puede observar que la medida de desempeño global exhibe una correlación de Pearson positiva ($r = 0.913$) con el $\tilde{r}^2$. En general, se puede observar la tendencia de que cuando los modelos poseen un mayor desempeño general en las tareas downstream, el alineamiento con la corteza auditiva crece. Si bien se observan algunos modelos con un desempeño particularmente malo (puntos rojos correspondientes a MetricGAN, DCASE2020, SepFormer y ZeroSpeech2020), descartando estos modelos

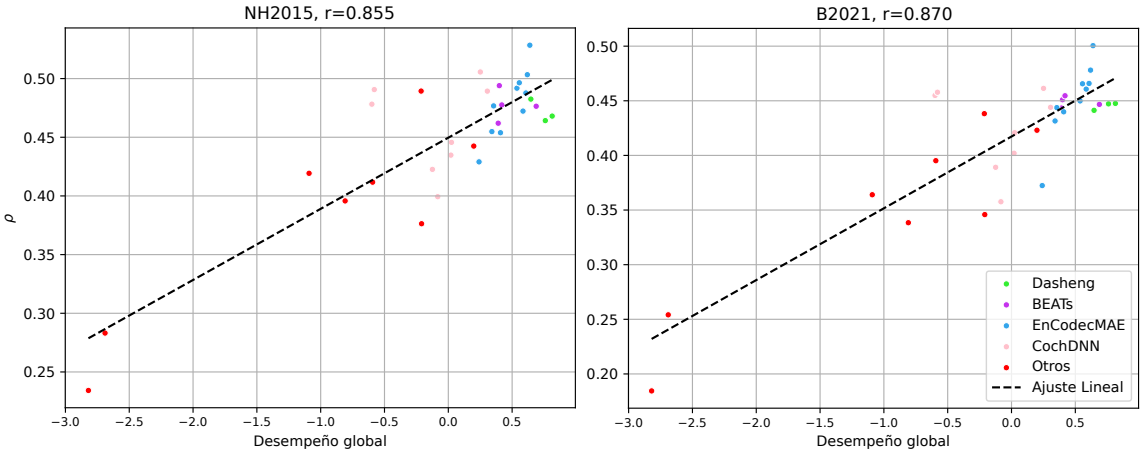

Figura 5.9: Relación entre el desempeño de los modelos analizados en tareas downstream y el alineamiento con la corteza auditiva mediante RSA (ρ de Spearman) para los datasets NH2015 y B2021.

también la correlación es alta y significativa ($r = 0.746$).

Realizamos el mismo análisis utilizando como variable dependiente el ρ de Spearman obtenido con el análisis de RSA, cuyo resultado se puede observar en la Figura 5.9. Las conclusiones se conservan en ambos tipos de análisis, habiendo una clara correlación positiva entre el desempeño downstream y el grado de alineamiento con la corteza auditiva.

También analizamos la correlación entre el desempeño en cada tarea individual y el $\tilde{r}^2$ obtenido con los modelos. Los resultados se muestran en la Tabla 5.1 analizando todos los modelos y también descartando los que tienen un desempeño global menor a -0.7. Se puede observar que al descartar aquellos modelos con un muy bajo desempeño global, el desempeño en tareas relacionadas con el habla (SC y ER), no correlaciona con el alineamiento entre el modelo y las representaciones cerebrales. Las tareas que exhiben una mayor correlación son aquellas relacionadas a la clasificación y detección de eventos acústicos (FSD y ESC). Una posible explicación es que estas tareas comprenden un rango de sonidos más amplio y diverso, pudiendo abarcar posiblemente más patrones de actividad cerebral. Al mismo tiempo, el tipo de sonidos utilizados en estas tareas es similar a los estímulos sonoros utilizados en las pruebas experimentales. Por último, la combinación de todas las tareas mediante una métrica global resultó ser la variable que mejor correlaciona con el alineamiento, superando a todas las tareas individuales. Los resultados que obtuvimos al correlacionar con el ρ de Spearman obtenido mediante RSA son similares.

Estos análisis verifican la hipótesis de representación platónica:

- Al mejorar los modelos de audio, su alineamiento con la corteza auditiva aumenta.
- Particularmente, el alineamiento parece correlacionar mejor con el desempeño medido en múltiples tareas de audio de distintos dominios (habla, música, y sonidos ambientales), coincidiendo con la hipótesis de que modelos que son

Análisis	Dataset	Filtrado	NS	GC	SC	ER	FSD	ESC	Global
Regresión	NH2015	Todos	.696	.878	.749	.637	.873	.870	**.913**
		> -0.7	.449	.677	.096	.318	.697	.690	**.746**
RSA	NH2015	Todos	.681	.848	.714	.542	.816	.800	**.855**
		>-0.7	.376	.446	.002	.079	**.468**	.425	.454
	B2021	Todos	.650	.847	.755	.593	.827	.809	**.870**
		>-0.7	.310	.446	.096	.176	.467	.433	**.487**

Tabla 5.1: Correlaciones de Pearson entre el desempeño en cada tarea y el alineamiento con el cerebro. Se muestran en verde los resultados con un nivel de significancia $p < 0.01$, en amarillo con $0.01 \leq p \leq 0.05$ y en rojo con $p > 0.05$.

Componente	Filtrado	NS	GC	SC	ER	FSD	ESC	Global
LF	Todos	**.856**	.617	.451	.309	.704	.652	.697
	>-0.7	**.910**	.277	-.203	-.134	.436	.331	.436
HF	Todos	**.893**	.808	.635	.528	.806	.785	.865
	>-0.7	**.858**	.343	-.026	.041	.431	.376	.537
Broadband	Todos	.251	.630	.615	.518	**.769**	.737	.684
	>-0.7	-.054	.608	.046	.286	**.732**	.721	.581
Pitch	Todos	.748	**.934**	.739	.604	.888	.883	.931
	>-0.7	.446	.626	-.079	.104	**.674**	.651	.613
Habla	Todos	.572	.816	.769	.642	.792	.796	**.852**
	>-0.7	.066	.543	.339	.479	.433	.448	**.578**
Música	Todos	.748	.869	.706	.619	.891	.874	**.914**
	>-0.7	.573	.603	.000	.227	.668	.641	**.700**

Tabla 5.2: Correlaciones de Pearson entre el desempeño en cada tarea y el alineamiento con cada componente de fMRI. Se muestran en verde los resultados con un nivel de significancia $p < 0.01$, en amarillo con $0.01 \leq p \leq 0.05$ y en rojo con $p > 0.05$.

buenos en múltiples tareas se alinean más entre si.

Por último, podemos analizar qué tareas se correlacionan mejor con la predicción de cada componente derivado del fMRI. En la Tabla 5.2 se pueden ver los coeficientes de correlación entre el desempeño en cada tarea y el r^2. Se puede observar que la mayoría de las tareas exhiben una correlación significativa con las distintas componentes. Resulta interesante que el alineamiento con las primeras componentes (LF y HF), que responden a atributos espectrales, se correlaciona más fuertemente con la tarea de clasificación de notas musicales, mientras que la componente 3 (Broadband) relacionada con eventos impulsivos (cortos y de banda ancha), se correlaciona mejor con la tarea de detección de eventos acústicos. Por último, las componentes 5 y 6, asociadas a habla y música respectivamente, se correlacionan mejor con la métrica global, la cual tiene en cuenta el desempeño en todas las tareas. Algunas conclusiones que se desprenden de estos resultados son:

- Las tareas relacionadas a habla son las que menos se correlacionan con medidas de alineamiento entre las representaciones de modelos de audio y el cerebro. De esto se desprende que los modelos entrenados solamente con habla sean los que menos alineamiento exhiben. De todas formas, se puede observar que el máximo r obtenido con estas tareas se dió con la componente de habla.
- Los modelos que exhiben un mayor alineamiento con las componentes de habla y música son aquellos que tienen un buen desempeño en todas las tareas. Esto también se observa en la Figura 5.4, en donde los modelos incorporados en este trabajo, cuyo desempeño general es superior a los analizados en el trabajo original, son los que mejor se alinean con estas dos componentes.
- La métrica global y la tarea de detección de eventos acústicos (FSD) son las únicas dos que exhiben correlaciones positivas y significativas con todas las componentes. Particularmente la tarea de detección de eventos acústicos requiere reconocer una gran variedad de sonidos abarcando 200 clases. Esto sugiere que el alineamiento con el cerebro es mayor cuanto más diversa es la tarea, lo cual coincide con la hipótesis de múltiples tareas descrita previamente.

Por último, se puede desprender de estos resultados una implicación práctica respecto a la evaluación de representaciones de audio. Dada la rapidez y bajo costo computacional de calcular el alineamiento entre una representación y componentes, o realizar un análisis de similaridad de representaciones, estas tareas downstream podrían ser alternativas o complementarias a un benchmark como HEAREval. Por ejemplo, durante el preentrenamiento de un modelo de audio, se podrían monitorear estas métricas de alineamiento con la corteza auditiva, y usarse como un 'proxy' computacionalmente eficiente del desempeño que tendrá el modelo en otras tareas de audio.

5.3.6. Correspondencia estructural

Un análisis interesante y común en este tipo de trabajos es verificar si las redes neuronales profundas exhiben una organización jerárquica similar a la del cerebro. En este sentido, se esperaría que las primeras capas del modelo sean buenas predictoras de las primeras etapas de procesamiento auditivo—la corteza auditiva primaria (A1)— y que las capas más profundas sean buenas predictoras de etapas más tardías del procesamiento auditivo— la corteza auditiva secundaria (A2) o belt, y la terciaria (A3) o parabelt—.

Para verificar si existe esta organización jerárquica, para cada voxel del fMRI buscamos la capa de la red neuronal que mejor predice su actividad en términos de $\tilde{r}^2$. Luego, normalizamos este número de capa dividiendo por el total de capas del modelo, por ejemplo, la capa 6 en un modelo con 12 capas corresponde a la capa relativa 0.5. De esta manera obtenemos $L_{s,v}$, es decir, la mejor capa relativa para predecir el voxel v del sujeto s. Para cada región anatómica de interés calculamos la mediana de los $L_{s,v}$ correspondientes a voxeles de esa región y promediamos este valor entre sujetos. Las regiones anatómicas surgen de combinar subconjuntos de regiones provenientes de la parcelación de Glasser [245], y fueron definidas en un

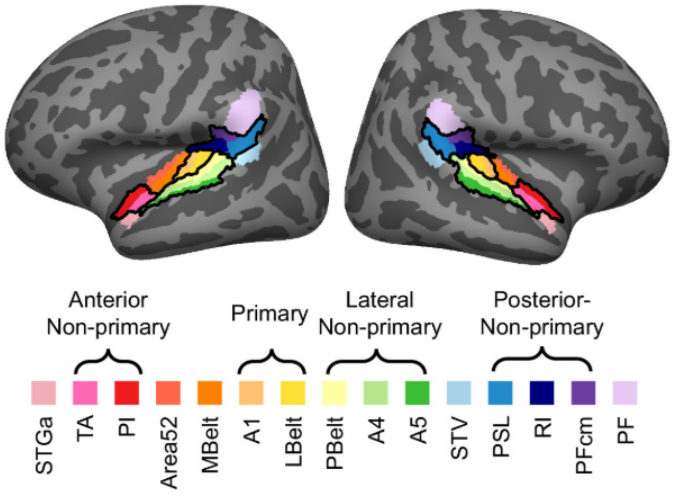

Figura 5.10: Regiones anatómicas consideradas para el análisis de correspondencia estructural. Figura extraída de [243].

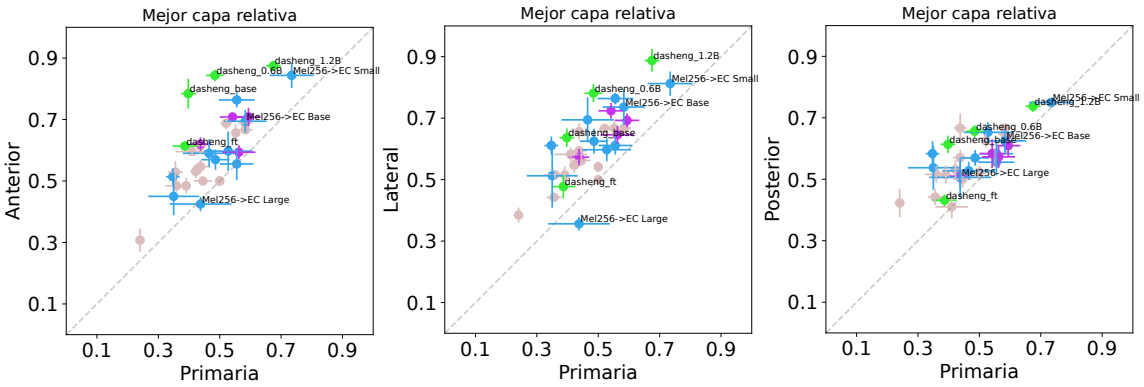

Figura 5.11: Mejor capa relativa para predecir distintas regiones anatómicas comparadas con la mejor capa relativa para predecir la actividad en la corteza auditiva primaria.

trabajo previo [243] independiente de este y del de Tuckute. Estas regiones son la primaria, lateral, anterior y posterior y se pueden visualizar en la Figura 5.10.

Luego, podemos comparar para cada modelo cuál es la capa que mejor predice la región primaria y otra región. Si hay una organización en la red neuronal artificial análoga a la organización de la corteza auditiva, uno esperaría que la capa que mejor predice a la región primaria sea menor que la capa que mejor predice a las regiones no primarias (anterior, posterior y lateral). En la Figura 5.11 se puede observar que efectivamente, la mayoría de los modelos se encuentran por encima de la diagonal. Esta diagonal, indicaría que la mejor capa predictora de ambas regiones es la misma. Puntos por encima de la diagonal indican que la mejor capa predictora de la corteza auditiva primaria se encuentra antes que la mejor capa predictora de otras regiones. Se puede observar que la mayoría de los modelos se encuentran por encima de la diagonal, y por lo tanto, sus primeras capas se alinean mejor con las primeras etapas

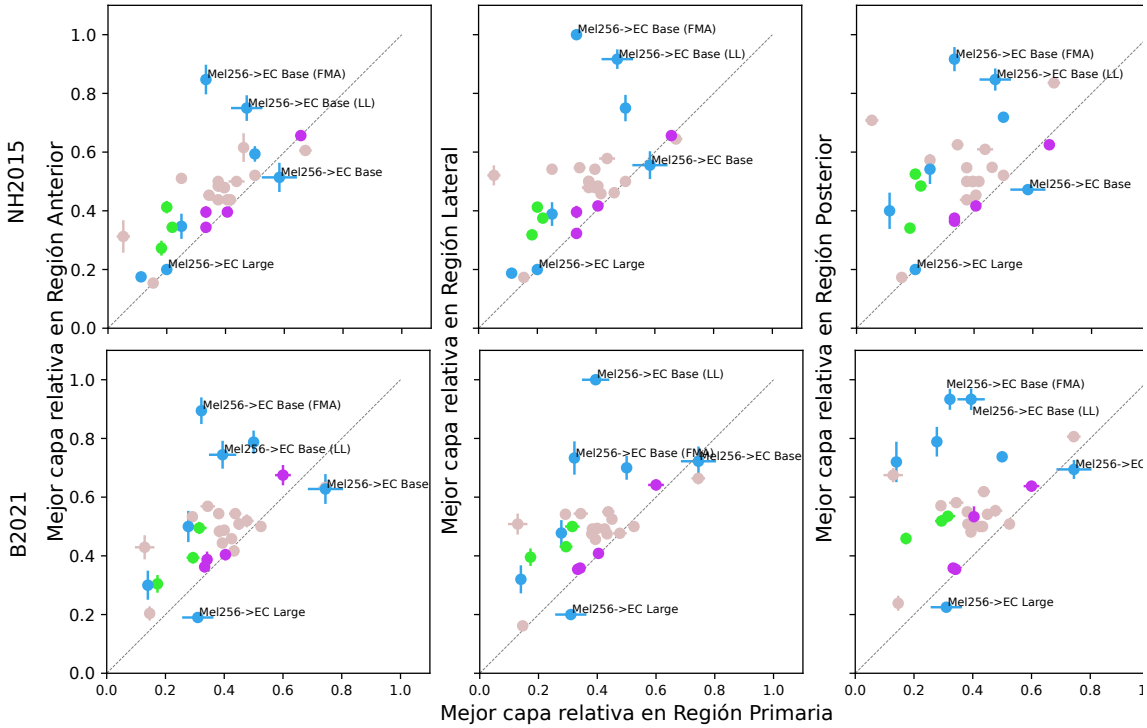

Figura 5.12: Mejor capa relativa para predecir distintas regiones anatómicas comparadas con la mejor capa relativa en términos de similaridad de representaciones.

del procesamiento auditivo, mientras que las últimas capas se alinean mejor con etapas más tardías. En particular, los modelos Dasheng son los que muestran una mayor diferenciación de etapas (están más alejados de la diagonal), mientras que los modelos de EnCodecMAE (puntos celestes) y BEATs (puntos violetas), exhiben un comportamiento similar a los modelos analizados en el trabajo original (puntos rosas). Estos resultados refuerzan los del trabajo original de Tuckute et al. al conservarse la tendencia de correspondencia estructural tras agregar modelos de audio más recientes. El modelo large de EnCodecMAE es el único debajo de la diagonal, particularmente al comparar la región lateral con la primaria.

Repetimos el análisis pero utilizando las mejores capas obtenidas en el análisis de similaridad de representaciones. Para esto se calcularon RDMs a partir de los voxeles del fMRI pertenecientes a cada región anatómica de interés, y se correlacionaron con las RDMs obtenidas a partir de las representaciones de audio. El resultado se puede observar en la Figura 5.12 para ambos conjuntos de datos (NH2015 y B2021). Nuevamente el modelo Large de EnCodecMAE es uno de los que menos se corresponde estructuralmente, invirtiendo la organización jerárquica del cerebro. Llama la atención de que a pesar de no corresponderse estructuralmente con el cerebro, este modelo es el que exhibe mayor similaridad con las representaciones del fMRI. Más allá de este resultado curioso y que va en contra de lo que esperábamos, la mayoría de los modelos exhiben una correspondencia entre etapas, aunque en este caso, no son los modelos de Dasheng los que más se separan de la diagonal, sino que modelos de EnCodecMAE entrenados sólo en música o en habla. Nuevamente,

resulta curioso que estos modelos no eran los más similares a la señal del fMRI.

Realizamos una prueba de rangos con signo de Wilcoxon para confirmar que hay diferencias entre la región primaria y las no primarias en términos de qué capa relativa es la que mejor predice. Mediante la prueba de Wilcoxon se confirmó que hay una diferencia significativa ($p < 0.001$) entre cada región no primaria y la primaria tanto para el análisis de regresión como el de RSA, y para ambos conjuntos de datos, confirmando lo observado en las Figuras 5.11 y 5.12.

Hipotetizamos que la correspondencia estructural va a depender de la arquitectura y la función objetivo utilizada. Por ejemplo, se ha observado que modelos como Wav2Vec 2.0 codifican información semántica en sus capas intermedias, mientras que las últimas capas codifican información local de manera similar a las primeras [211]. Esto tiene sentido ya que la tarea de pretexto de Wav2Vec 2.0 implica en cierto modo reconstruir una versión cuantizada de la señal de entrada. En cambio, un modelo entrenado para detectar eventos acústicos como AST, va a tener información más semántica y de alto nivel en las últimas capas, ya que su salida deberá corresponderse con estos eventos.

5.3.7. Alineamiento a lo largo del preentrenamiento

Un aspecto interesante de analizar es si optimizar la tarea de pretexto lleva a que las representaciones del modelo se alineen más con las de la corteza auditiva. Para esto, podemos analizar el alineamiento en función de la cantidad de pasos de preentrenamiento. Si la tarea de pretexto conduce naturalmente a un mayor alineamiento, deberíamos observar que este crece a medida que se avanza en el preentrenamiento. Decidimos realizar este análisis con el modelo EnCodecMAE, particularmente Mel256→EC Base.

En la Figura 5.13 se puede observar que las representaciones de las distintas capas se vuelven más similares a las representaciones cerebrales obtenidas mediante el fMRI a medida que el modelo se preentrena. Cabe notar que durante el preentrenamiento del modelo, no hay ninguna optimización explícita de la similaridad con representaciones cerebrales, ni se hace uso de conjuntos de datos obtenidos mediante fMRI. El alineamiento, al igual que el buen desempeño obtenido en distintas tareas downstream, es un producto del modelo aprendiendo a reconstruir segmentos faltantes de audio a partir de su contexto. Resulta interesante que la mayor parte del alineamiento ocurre durante los primeros 100000 pasos de preentrenamiento, y luego este oscila alrededor del valor máximo. A su vez, los valores máximos de similaridad se alcanzan en las capas intermedias (5 y 6). Un resultado importante es que el alineamiento sigue patrones muy similares en ambos conjuntos de datos, a pesar de corresponder a sujetos distintos, lo cual habla de una robustez en la cuantificación de la similaridad de las representaciones.

Dado que la mayor parte del alineamiento ocurre temprano en el preentrenamiento, decidimos repetir el preentrenamiento del modelo Mel256→EC Base, guardando los pesos cada 1000 pasos de preentrenamiento en lugar de 50000. Lo que observamos es que temprano en el preentrenamiento ocurre una diferenciación estructural que se

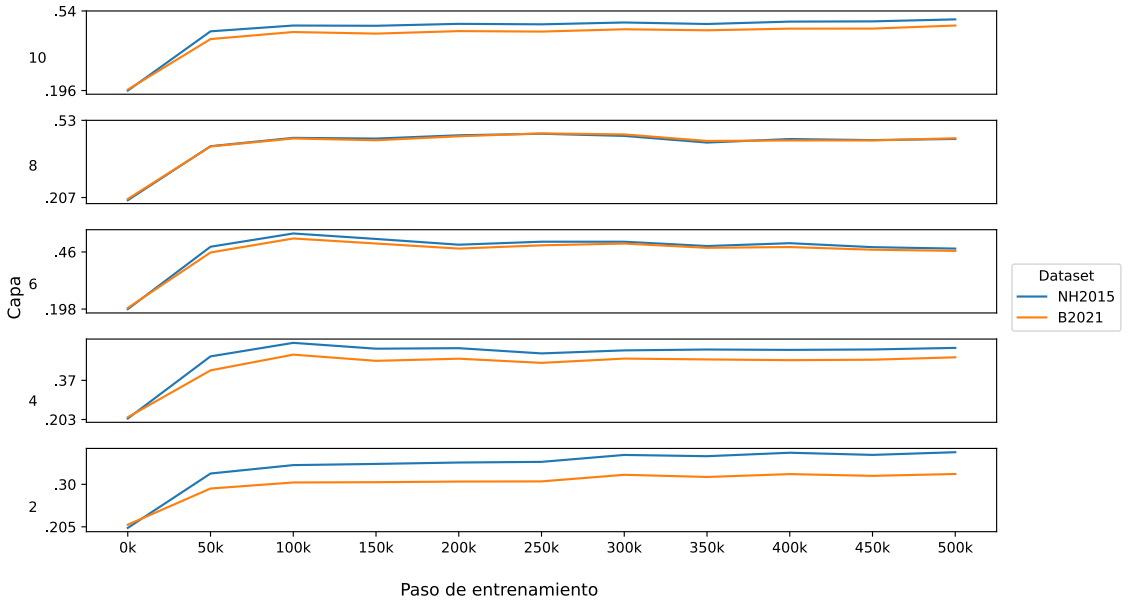

Figura 5.13: ρ de Spearman del análisis de RSA en el conjunto de datos B2021 y NH2015 para cada capa a lo largo del preentrenamiento.

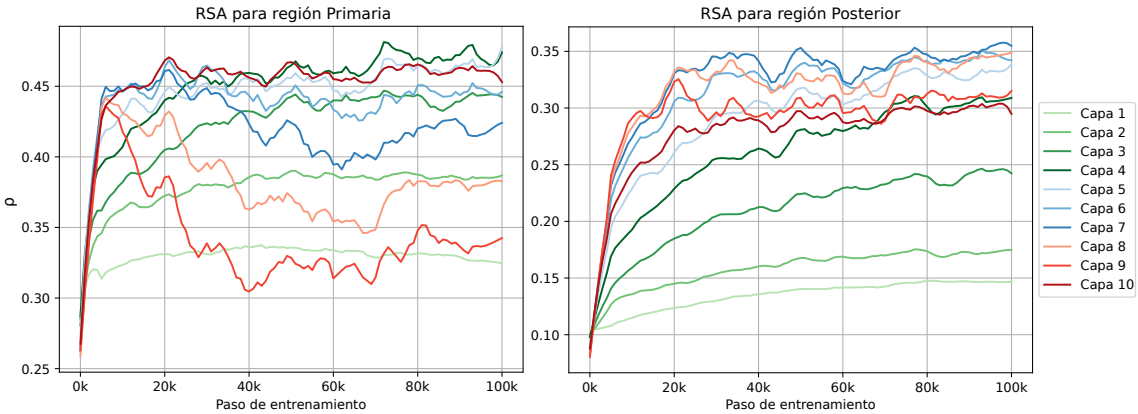

Figura 5.14: Evolución de la similaridad entre representaciones de audio y cerebrales correspondientes a las regiones primaria y posterior obtenidas en el conjunto de datos NH2015 para cada una de las capas del modelo Mel256→EC Base.

corresponde con la observada en la corteza auditiva. Esto se puede ver en la Figura 5.14, donde las capas 7 y 8 disminuyen su similaridad con las representaciones de la región primaria temprano en el preentrenamiento, siendo menos similares a las representaciones de la corteza auditiva primaria que la mayoría de las capas. Por otro lado, la similaridad de estas capas con la región posterior no decrece y se encuentra entre las mayores del modelo.

Es notable que la capa final no sigue la tendencia de las capas previas en su similaridad con la región primaria, mostrando una alta similaridad a lo largo de todo el entrenamiento. Hipotetizamos que esto se debe a la pre-post normalización que le permite combinar información local de las primeras capas. Esta incorporación de información local causa a su vez que la similaridad en la región posterior sea inferior a la de las capas anteriores (8 y 9). Si bien no mostramos la evolución del ρ de Spearman a lo largo del preentrenamiento para las regiones lateral y anterior, exhiben un comportamiento similar al de la región posterior, mostrada en la Figura 5.14.

6

Otros trabajos relacionados a la tesis

6.1. Reconocimiento de emociones mediante fusión de texto y audio

Este fue el primer proyecto que realicé durante mi doctorado entre el año 2019 y 2020, y fue un disparador para explorar el aprendizaje de representaciones y la transferencia de aprendizaje. El objetivo era mejorar sistemas de reconocimiento de emociones a partir del habla, utilizando transcripciones. De esta manera, se puede combinar la señal de audio con el texto correspondiente. En esta época ocurrió el despegue del procesamiento del lenguaje natural con los primeros modelos auto-supervisados de texto basados en redes transformer. Previamente a este trabajo, las transcripciones se codificaban con embeddings de texto no contextualizados, como GloVE [246] y Word2Vec [247]. Resultaba natural entonces explorar el uso de BERT [78], un modelo popular en aquel entonces para resolver tareas de NLP, para codificar las transcripciones y mejorar los sistemas de reconocimiento de emociones. Específicamente, nuestras contribuciones con este trabajo fueron: i) Mostrar que el uso de transcripciones ayuda a mejorar el reconocimiento de emociones en el conjunto de datos MSP-PODCAST [248], en el cual solo se habían hecho experimentos con audio, ii) Incorporar representaciones de texto contextualizadas, ya que en el campo solo se habían utilizado representaciones estáticas, que dependen solamente de la palabra y no del contexto en el que se utiliza, iii) proponer una manera novedosa de fusionar las modalidades, preentrenando cada modalidad por separado, y luego realizando un finetuning del modelo completo y iv) mostrar que no es suficiente separar los datos por hablante en IEMOCAP si se utilizan transcripciones, y que es necesario también separar por guión o utilizar el subconjunto de improvisaciones.

Utilizamos los conjuntos de datos IEMOCAP [62] y MSP-PODCAST [248], los cuales contienen diálogos y las correspondientes anotaciones de emociones. Particularmente, trabajamos con 4 clases de emociones: felicidad, enojo, tristeza y neutral. IEMOCAP contiene aproximadamente 12 horas de diálogos guionados e improvisados organizados en 5 sesiones, cada una con turnos de diálogo entre un actor y una actriz. Para la evaluación se utiliza validación cruzada de 5 particiones, asegurándonos que los conjuntos de entrenamiento y evaluación no comparten actores ni guiones. MSP-PODCAST contiene segmentos de grabaciones de podcasts, anotados mediante crowdsourcing. Luego de quedarnos solo con las 4 clases de interés, el conjunto de

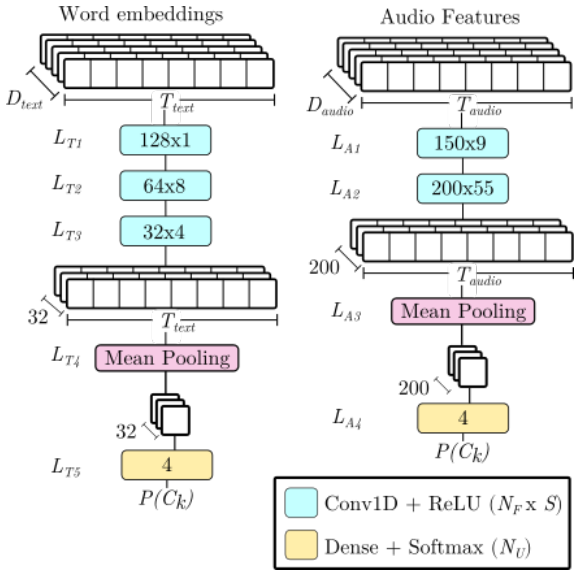

Figura 6.1: Arquitecturas utilizadas para procesar la modalidad de texto y audio. Ambas ramas consisten de una serie de capas convolucionales, seguidas de un promedio temporal y una capa de salida.

entrenamiento resultante contiene 12078 segmentos de habla de 601 hablantes, y el de evaluación 5557 de 50 hablantes no presentes en el de entrenamiento. Respecto a las transcripciones, en el caso de IEMOCAP utilizamos manuales provistas en el mismo conjunto de datos, mientras que en MSP-PODCAST utilizamos la API de Google Cloud de habla a texto.

Como se puede observar en la Figura 6.1, para la modalidad de audio y texto se utilizaron redes neuronales convolucionales, que en el caso del audio reciben secuencias de atributos extraídos con OpenSmile: pitch, jitter, shimmer, logHNR, sonoridad y los primeros 13 MFCCs, y en el caso de texto recibe secuencias de embeddings del modelo BERT base uncased obtenidas sumando las activaciones de las últimas 4 capas. Esto se mostró en el trabajo original de BERT que resultaba en general en un desempeño similar a realizar ajuste fino. Luego, exploramos distintas estrategias para combinar las dos redes:

- Fusión temprana (EF): se concatenan las representaciones obtenidas tras el promediado temporal (L_{T4} y L_{A3}), resultando en una nueva representación con 232 dimensiones. Esta representación es procesada por una capa oculta de 128 neuronas con activación ReLU y por la capa de salida que devuelve probabilidades para cada emoción.
- Fusión tardía (LF): se concatenan los logits de ambas modalidades (L_{T5} y L_{A4}) y se envían a la capa de salida, la cual aprende a combinar los logits para resolver la tarea.

A su vez, exploramos 3 estrategias de entrenamiento:

- Inicio frío (CS): se entrena el modelo completo desde cero

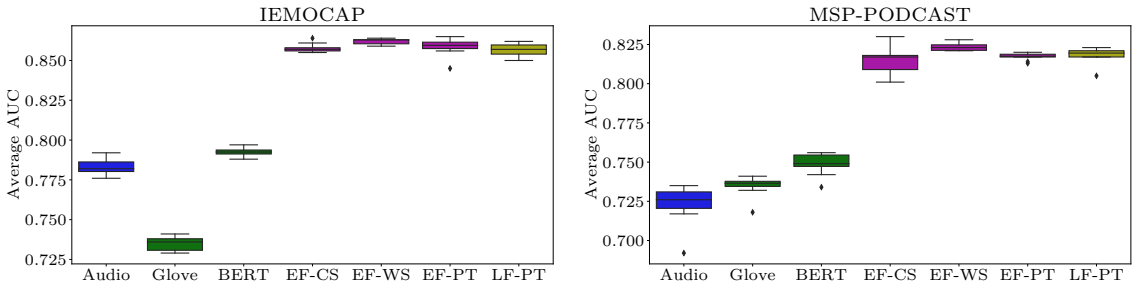

Figura 6.2: Resultados obtenidos en ambos conjuntos de datos utilizando los distintos modelos descritos. Las barras de error corresponden al rango intercuartil obtenido al entrenar con 10 semillas distintas.

- Preentrenado (PT): se entrena cada modalidad por separado, y luego se entrenan solamente las capas agregadas para la fusión.
- Inicio tibio (WS): es similar a Preentrenado, pero también se actualizan los pesos de las subredes de cada modalidad.

Se puede observar en la Figura 6.2 que para ambos conjuntos de datos, utilizar las representaciones contextualizadas de BERT mejoran a las estáticas de GloVE. Al mismo tiempo, utilizar sólo las transcripciones resulta en mejores resultados que utilizar sólo el audio. Sin embargo, ambas modalidades se complementan satisfactoriamente al utilizar las distintas estrategias de fusión. Para todos los casos, utilizar ambas modalidades mejora frente a utilizar una sola de ellas. Entre las distintas estrategias, realizar fusión temprana y un inicio tibio (EF-WS) parece ser la más efectiva.

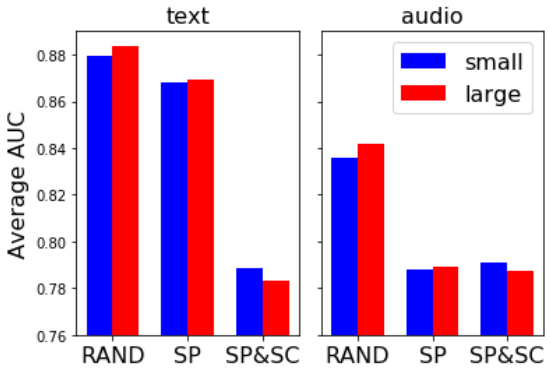

Figura 6.3: Efecto de distintos criterios para definir las particiones en IEMOCAP usando modelos de audio y de texto con dos tamaños distintos (Small y Large). RAND: particiones aleatorias. SP: división por hablante. SP&SC: división por hablante y guión. Los modelos Small y Large poseen la mitad y el doble de neuronas en cada capa del modelo de la Figura 6.1 respectivamente.

En este trabajo también mostramos que utilizar divisiones sólo por hablante (SP) puede resultar optimista en sistemas que utilizan transcripciones. Esto se puede

observar en la Figura 6.3, en donde se ve que realizar la división por hablante y guión al utilizar BERT resulta en un desempeño mucho menor y más realista que realizar divisiones aleatorias (RAND) o por hablante.

6.2. Reconocimiento de emociones con Wav2Vec 2.0

En este trabajo, nuevamente hacemos uso de representaciones obtenidas de modelos preentrenados para mejorar el desempeño en la tarea de reconocimiento de emociones a partir de habla. Particularmente, dado que se habían comenzado a adoptar algunas de las técnicas que habían sido exitosas en NLP en el campo del habla, resultaba natural explorar estos nuevos modelos de habla aplicados en la tarea de reconocimiento de emociones. Previo a nuestra publicación realizada en 2021, los modelos utilizados para realizar reconocimiento de emociones consistían de:

- Modelos entrenados desde cero para la tarea. Normalmente se utilizaban redes convolucionales o recurrentes tomando como entrada atributos de audio o espectrogramas.
- Modelos combinando modalidades. Se solía utilizar transcripciones manuales o automáticas de los audios y combinarlas con la señal acústica, utilizando modelos de texto en conjunto con modelos de audio.
- Transferencia de aprendizaje. Este es el tipo de trabajo que mejor se alinea con el nuestro. Se solían utilizar modelos preentrenados para ASR o modelos de autosupervisión con arquitecturas recurrentes o convolucionales. El trabajo más cercano al nuestro en esta dirección es el de Boigne et al. [249], en el que se utilizan atributos de Wav2Vec y BERT.

Nuestras contribuciones fueron:

- Primer trabajo en explorar el uso de Wav2Vec 2.0 para la tarea de reconocimiento de emociones a partir del habla, y uno de los primeros en utilizar representaciones de modelos de habla preentrenados basados en transformers.
- Uno de los primeros trabajos en utilizar Wav2Vec 2.0 para una tarea que no sea reconocimiento del habla. Otras tareas que se exploraron concurrentemente a nuestro trabajo fueron: verificación de hablante e identificación de idioma [250], y traducción de habla [251].
- Proponemos un modelo downstream que permite utilizar todas las activaciones de Wav2Vec 2.0 al mismo tiempo, lo cual resulta en un mejor desempeño que utilizar sólo la última capa, y es más eficiente computacionalmente que buscar la mejor capa del modelo.

Utilizamos los conjuntos de datos IEMOCAP y RAVDESS [252] para realizar los distintos experimentos. En el caso de IEMOCAP predecimos 4 emociones al igual que se describió en la sección 6.1, mientras que para RAVDESS predecimos 7 emociones tras fusionar las emociones neutral y calma, siguiendo la metodología en [253].

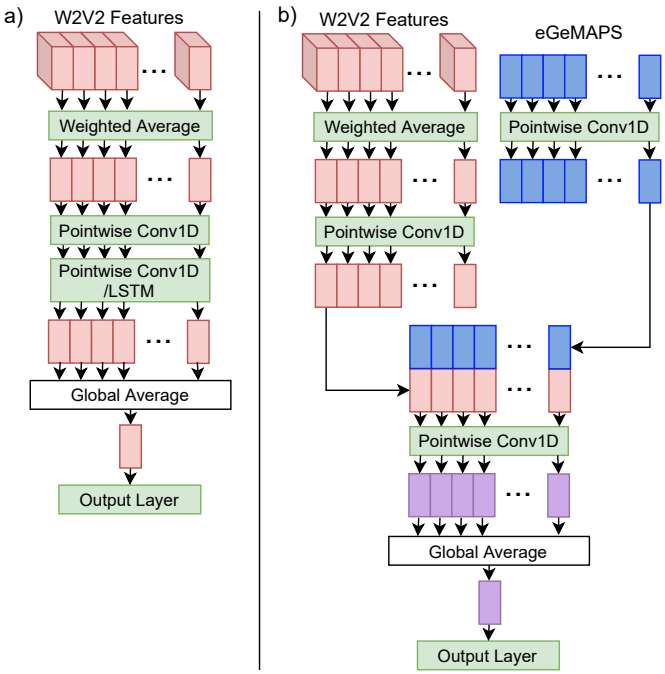

Figura 6.4: Modelos downstream utilizados en este trabajo. Los bloques verdes se pueden entrenar. El modelo a) lo llamamos Dense al utilizar una capa convolucional con tamaño de kernel 1 (pointwise convolution), o LSTM al utilizar esta capa en lugar de la convolucional. El modelo b) lo denominamos Fusion.

La arquitectura del modelo se puede observar en la Figura 6.4. Se parte de las activaciones de Wav2Vec 2.0, extraídas de la salida del encoder convolucional (o entrada al transformer), y de cada una de las 12 capas de Transformer. Esto resulta en un tensor con dimensiones $13 \times \frac{T}{320} \times 768$. El modelo downstream puede combinar las distintas activaciones al aprender un promedio pesado, resultando en un tensor con dimensiones $\frac{T}{320} \times 768$. Luego, 2 capas convolucionales, o una convolucional y una LSTM, procesan esta secuencia de atributos. Finalmente se colapsa la dimensión temporal mediante un promedio, obteniéndose una representación de tamaño fijo con 768 dimensiones, la cual sirve de entrada a la capa de salida que devuelve las probabilidades para las distintas emociones. Ambas capas convolucionales poseen 128 canales, activación ReLU y dropout con probabilidad de 0.2, imitando la configuración utilizada en [249]. También experimentamos combinando los atributos de wav2vec 2.0 con eGeMAPS. Con esta arquitectura queríamos responder a la pregunta ¿Son los atributos expertos de eGeMAPS complementarios a las representaciones de wav2vec 2.0? Si el desempeño no se viera alterado, esto significaría que wav2vec 2.0 ya codifica internamente los distintos atributos de eGeMAPS, mientras que si el desempeño mejora, significaría que algunos aspectos de eGeMAPS útiles para el reconocimiento de emociones, no están presentes en las activaciones de Wav2Vec 2.0. Por otro lado, se experimentó normalizando las activaciones de Wav2Vec 2.0 de dos maneras:

- Global: se normalizan las representaciones de Wav2Vec 2.0 y de eGeMAPS

Upstream	Atributos	Dataset	
		IEMOCAP	RAVDESS
None	eGeMAPS	52.4 ± 0.1	57.0 ± 2.4
	Espectrograma	49.8 ± 1.0	44.5 ± 0.8
Wav2vec2-PT	Capa 0	60.3 ± 0.7	65.4 ± 1.7
	Capa 12	58.5 ± 0.6	69.0 ± 0.2
	Capas 0 a 12	**67.2 ± 0.7**	**84.3 ± 1.7**
Wav2vec2-FT	Capa 0	57.3 ± 1.0	58.8 ± 2.7
	Capa 12	44.6 ± 1.0	37.5 ± 3.0
	Capas 0 a 12	63.8 ± 0.3	68.7 ± 0.9

Tabla 6.1: Recall promedio (%) obtenido para los distintos modelos explorados en IEMOCAP y RAVDESS. Todos los resultados se obtienen utilizando el downstream con capas convolucionales y normalización por hablante.

utilizando la media y desviación estándar calculada sobre el conjunto de entrenamiento completo.

- Hablante: se normalizan las representaciones de Wav2Vec 2.0 y de eGeMAPS utilizando la media y desviación estándar calculada sobre los audios correspondientes al hablante. Esta normalización requiere de estadísticas del hablante durante la inferencia.

En la Tabla 6.1 se pueden ver los resultados obtenidos con normalización por hablante. Se puede observar que utilizar los atributos de Wav2Vec 2.0 da mejores resultados que utilizar atributos expertos como eGeMAPS y espectrogramas. A su vez, se puede observar que el modelo de Wav2Vec 2.0 que no fue finetuneado para la tarea de ASR (Wav2vec2-PT) posee un mejor desempeño que el que se finetuneo en Librispeech para reconocimiento de habla (Wav2vec2-FT). Una hipótesis es que, si bien ajustar a la tarea de ASR podría ayudar al modelo a identificar mejor palabras que pueden ser útiles para reconocer emociones, también puede hacer que las representaciones estén muy alineadas con esta tarea, y dejen de ser útiles para reconocimiento de emociones. Por ejemplo, el pitch es un atributo que es útil para el reconocimiento de emociones, pero no es necesario para reconocer habla, e incluso puede perjudicar a modelos de ASR. En este sentido, Wav2vec2-FT podría no codificar este tipo de atributos prosódicos, llevando a un mal desempeño en reconocimiento de emociones. Otro resultado interesante es que las representaciones de la última capa no parecen ser útiles para la tarea, y en la mayoría de los casos, la salida del encoder convolucional en Wav2Vec 2.0 (capa 0) alcanza un mejor desempeño. Este tipo de resultados se ha observado en otros trabajos posteriores como [211], en donde notan que las últimas capas de Wav2Vec 2.0 suelen estar muy ajustadas a la tarea de preentrenamiento, y utilizar capas intermedias resulta en un mejor desempeño. A su vez, la tarea de pretexto de Wav2Vec 2.0 induce a que las representaciones de la última capa se parezcan a las de la capa 0, ya que debe predecir

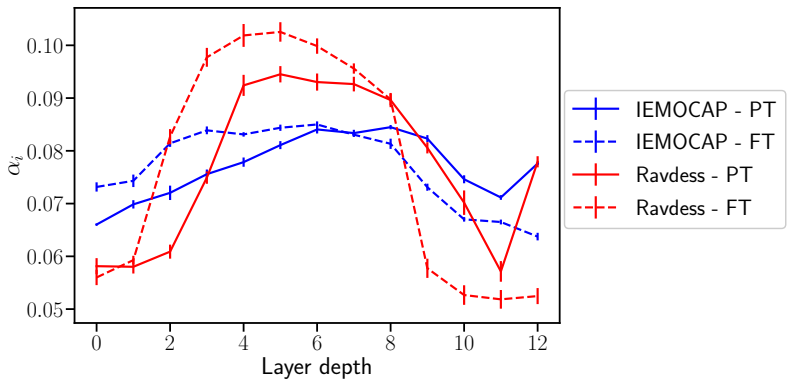

Figura 6.5: Pesos aprendidos para las distintas capas del modelo

Modelo	IEMOCAP	RAVDESS
CNN	65.8 ± 0.3	75.7 ± 2.3
LSTM	64.8 ± 1.9	74.6 ± 3.7
Fusión	**66.3 ± 0.7**	**77.5 ± 1.0**
BiLSTM con atención [254]	58.8	-
CNN-BiLSTM [255]	59.4	-
TDNN-LSTM con attention [256]	60.7	-
Wav2Vec [249]	64.3	-
eGeMAPS + BERT [6]	65.1†	-

Tabla 6.2: Recall promedio utilizando wav2vec2-PT como upstream, y atributos de todas sus capas. En negrita se marcan los mejores resultados con normalización global. El resultado marcado con † no es directamente comparable con el resto ya que la metodología de evaluación es distinta.

estos atributos cuantizados. Por último, confirmamos que utilizar las salidas de todas las capas aprendiendo pesos para las mismas es el método que mejores resultados da, superando en todos los casos a utilizar la capa 0 o la última. En la Figura 6.5 se puede observar que las capas que reciben pesos más altos son las intermedias. Resulta interesante que no hay tanta variabilidad entre semillas de inicialización, indicando que la forma de la curva es consistente entre distintas inicializaciones. Si bien se observan diferencias entre IEMOCAP y RAVDESS, el perfil sigue dándole mayor peso a las capas intermedias que a las primeras y últimas.

Por último, para realizar una comparación justa con el estado del arte, en la Tabla 6.2 se pueden observar los resultados obtenidos mediante normalización global. Se puede observar que utilizar la capa de LSTM en lugar de la convolucional no produjo mejoras en el desempeño, y que en cambio, combinar las activaciones de Wav2Vec 2.0 con atributos de eGeMAPS resultó en el mejor modelo para ambos conjuntos de datos. También se puede observar que los modelos propuestos superaron al estado del arte de ese momento, e incluso obtuvieron mejor desempeño que el modelo de nuestro trabajo previo que combinaba audio con transcripciones y describimos en

la sección 6.1, aunque la comparación no es justa ya que la división en ese trabajo era por hablante y guión, mientras que en este trabajo utilizamos la división típica por sesión. Trabajos posteriores como [213] fueron mostrando que el uso de modelos autosupervisados de habla como Wav2Vec 2.0 y WavLM efectivamente logran achicar la diferencia de desempeño entre modelos puros de audio y modelos que hacen uso de transcripciones, indicando que Wav2Vec 2.0 podría estar efectivamente capturando aspectos de la semántica y el contenido de los diálogos sin necesidad de utilizar transcripciones.

6.3. Embeddings posicionales para Audio Spectrogram Transformers

Este trabajo [257] se realizó poco después de que se publicara el trabajo de Audio Spectrogram Transformers (AST) [86]. La idea era explorar distintos embeddings posicionales para esta arquitectura dado que:

- Los autores solo utilizan embeddings posicionales absolutos.
- Los autores muestran la importancia de utilizar como pesos iniciales los de un ViT entrenado en Imagenet. Particularmente encuentran que es importante inicializar los embeddings posicionales con los del modelo de imágenes.
- La arquitectura de Vision Transformer al utilizar patches de imágenes o espectrogramas, y luego transformarlos en una secuencia plana, pierde la estructura 2D de la señal original, forzando a los embeddings posicionales a recuperar esta estructura.
- En la literatura se han propuesto una gran variedad de alternativas a los embeddings posicionales absolutos, como atención relativa [80, 81], ALiBi [258] y embeddings posicionales condicionales [259], los cuales no han sido explorado extensivamente en el dominio del audio.

En este trabajo entrenamos AST desde cero utilizando Audioset y experimentando con distintos embeddings posicionales. Particularmente, no inicializamos el modelo con los pesos de Imagenet sino que con pesos aleatorios.

Entrenamos AST con distintas variantes de embeddings posicionales:

- **Embeddings posicionales absolutos**: para cada posición posible de la secuencia, se aprende un vector. Luego se suma esta secuencia de vectores posicionales a la secuencia de entrada. La principal limitación de este enfoque es que se debe predefinir un largo máximo de secuencia y no es trivial hacer inferencia en secuencias más largas una vez entrenado el modelo.
- **Embeddings condicionales**: en lugar de aprender un conjunto fijo de embeddings, los mismos se generan dinámicamente a partir de la entrada. Se utiliza una capa convolucional 2D separable en profundidad (depthwise convolution) como generador de embeddings posicionales (PEG). Una de las ventajas respecto al embedding posicional absoluto típico es que se conserva la estructura 2D del

espectrograma a la hora de generar los embeddings posicionales. Sin embargo, no queda claro si introducir esta capa solamente brinda información posicional o si puede también ayudar al modelo en otros aspectos. De todas formas, cabe destacar que esta modificación solo introduce 38.4K parámetros nuevos. Los autores del trabajo [259] encontraron que los mejores resultados se obtenían al introducir los PEG en las salidas de las primeras 5 capas de transformer.

- **Atención relativa 2D**: en los mecanismos de atención relativa, el producto de atención se modifica según la siguiente ecuación:

$$\text{Att}(Q,K,V) = \text{Softmax}(\frac{QK^T + R}{\sqrt{d_k}})V$$

en donde R es la matriz de atención relativa, la cual va a modificar los pesos de atención según la distancia entre cada query Q_i y key K_j. En este trabajo adaptamos el mecanismo de atención propuesto en [81] incorporando distancia no solo en el eje temporal sino que también en el de frecuencias. De esta forma $R_{ij} = Q_i E^t_{\Delta t(i,j)} + Q_i E^f_{\Delta f(i,j)}$, en donde E^t y E^f son embeddings aprendidos por el modelo para cada distancia posible temporal y frecuencial, y $\Delta t(i,j)$ y $\Delta f(i,j)$ indican la diferencia entre dos posiciones temporales o frecuenciales. Compartimos las matrices E^t y E^f entre las distintas cabezas de atención, pero se aprenden distintas matrices para cada capa de transformer.

- **ALiBi**: extendimos el método de Attention with Linear Biases (ALiBi) [258] al caso no autoregresivo y para señales 2D. En ALiBi la matriz R es fija dando mayor peso a keys que se encuentran a corta distancia del query. Especificamente, en nuestra extensión 2D, la mitad de las cabezas de atención responde a la distancia temporal $R_{ij} = -m|\Delta t(i,j)|$, mientras que la otra mitad responde a la distancia en frecuencia $R_{ij} = -m|\Delta f(i,j)|$. m posee un valor distinto para cada cabeza de atención, siendo $m = 0.5^{16h/N_h}$ con h el índice de la cabeza actual y N_h la cantidad de cabezas de atención. Este parámetro permite que algunas cabezas de atención tengan un mayor sesgo a la localidad al tener un valor de m más alto.

Los resultados obtenidos con los distintos tipos de embedding posicional se detallan en la Tabla 6.3. Se puede observar que no utilizar embeddings posicionales lleva al peor desempeño. La extensión propuesta a ALiBi no mejoró los resultados de utilizar embeddings posicionales absolutos. Hipotetizamos que el sesgo de localidad en el eje de frecuencias puede ser perjudicial. Al mismo tiempo, esto podría quitarle cierta capacidad al modelo ya que sólo la mitad de las cabezas de atención agregan información posicional en el tiempo. Esto nos llevo a probar otra variante de ALiBi en la que todas las cabezas de atención responden solo a la distancia temporal. Para introducir información posicional respecto a las frecuencias, incorporamos esta información utilizando embeddings posicionales absolutos. Este enfoque (Time ALiBi) brindó mejoras respecto a utilizar embeddings posicionales absolutos. Nuestra extensión 2D de atención relativa produjo aún más mejoras, mientras que el método más efectivo fue el consistente en embeddings posicionales condicionales.

PE Method	Audioset (mAP)	ESC-50 (Accuracy)
None	0.286	81.2
Absolute	0.313	87.5
ALiBi 2D	0.307	86.3
Time ALiBi	0.319	87.6
Learned Relative	0.329	87.8
Conditional	0.343	**91.4**
Conditional + Absolute	**0.344**	90.0
AST [260]	**0.485**	**95.7**
WEANET [261]	0.398	94.1
EfficientNet [262]	-	89.5

Tabla 6.3: Desempeño obtenido en Audioset y ESC-50 utilizando los distintos métodos de embeddings posicionales. Los valores corresponden al mean Average Precision (mAP) en Audioset y accuracy en ESC-50. También mostramos resultados de modelos estado del arte.

En la Figura 6.6 a) y b) se puede observar que los embeddings posicionales absolutos aprenden a discriminar principalmente a las distintas bandas de frecuencia, mientras que el modelo parece exhibir cierta invarianza en el eje del tiempo. Esto tiene cierto sentido dado que la tarea de detección de eventos acústicos requiere solo confirmar la presencia de un evento en el segmento de audio, y no su ubicación, por lo tanto exhibir invariancia a la traslación en el eje temporal puede ser beneficioso. Por otro lado, los embeddings aprendidos mediante atención relativa parecen discriminar distintas regiones tanto en tiempo como en frecuencia. Por ejemplo, parecen discriminar entre posiciones muy cercanas en el tiempo, posiciones del futuro y posiciones a una distancia media y alta en el pasado. Un comportamiento similar se da en el eje de frecuencias, en donde los embeddings son más discriminativos en bandas de frecuencia cercanas que en las lejanas.

Si bien estas adaptaciones a los embeddings posicionales son directas de implementar para un transformer como el utilizado en AST, surgen algunas complicaciones al querer aplicarlas a un MAE:

- Al eliminar los tokens no visibles, las posiciones en la secuencia dejan de ser contiguas, lo cual complica la implementación de mecanismos de atención relativa.
- Del mismo modo, si se quisiera usar embeddings condicionales, tras descartar elementos de la secuencia no sería natural aplicar una capa convolucional para la generación de la información posicional.
- Si se generaran los embeddings posicionales antes de descartar elementos de la secuencia, se corre riesgo de conservar en el embedding posicional condicional información de los elementos que se descartaron facilitando la tarea de pretexto.

Por estos motivos, no experimentamos con variantes de EnCodecMAE que utilicen otros tipos de embedding posicionales distintos del sinusoidal.

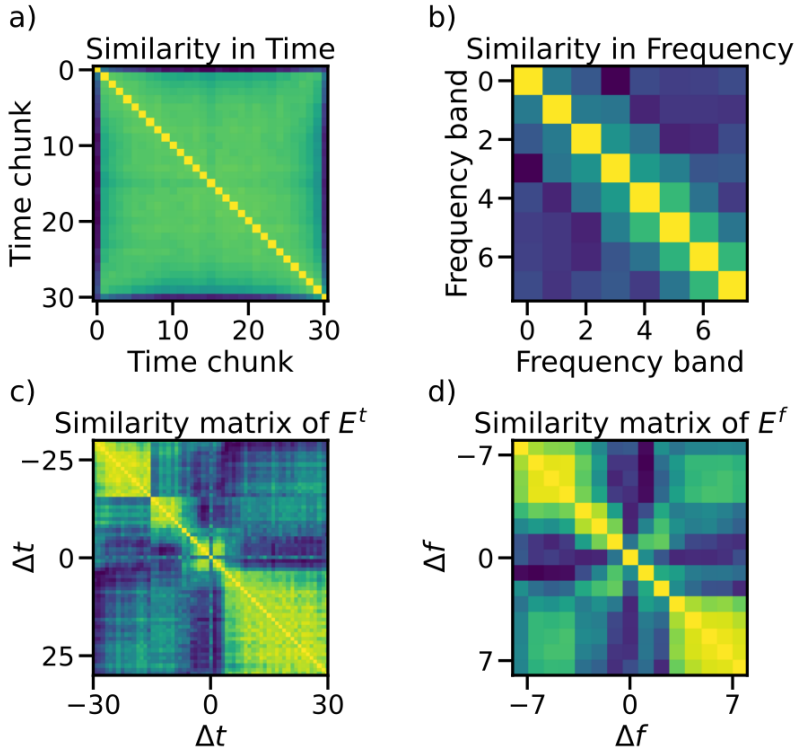

Figura 6.6: Matrices de similaridad coseno entre distintos embeddings posicionales. a) y b) corresponden a embeddings posicionales absolutos, en a) se concatenan todos los embeddings correspondientes a la misma posición temporal (y distinta frecuencial), mientras que en b) se concatenan los correspondientes a la misma posición frecuencial. En c) y d) se muestran los embeddings posicionales relativos aprendidos mediante la técnica de atención relativa 2D.

6.4. Aprendizaje de prototipos musicales

Como se explicó en la sección 2.1, algunas representaciones son inversibles, es decir, dada una forma de onda x y una representación $y = f(x)$, existe una función g tal que $g(f(x)) = x$. Algunas representaciones como espectrogramas, son inversibles bajo ciertas condiciones, mientras que otras, como la frecuencia fundamental, no lo son. Una pregunta que puede surgir es si las representaciones dadas por una red neuronal se pueden invertir. En el caso de un autoencoder, en cierta manera el encoder es f mientras que el decoder cumple la función de g. Si bien $g(f(x))$ no será igual a x, el error puede llegar a ser lo suficientemente bajo como para conservar las características originales de x, y ser escuchable; o en otros términos, tener una distorsión baja y una calidad perceptual alta (ver Sección 2.3.2).

Una pregunta que se puede formular en el marco de esta investigación, es si las representaciones generadas por EnCodecMAE son inversibles. Esta pregunta la recibí poco después de publicar el paper de EnCodecMAE en arXiv y dió lugar a un trabajo colaborativo que culminó con un artículo científico [263]. Un grupo de investigadores del Music Technology Group de la Universitat Pompeu Fabra

(UPF) y de la Universidad de la República (UDELAR) se encontraba trabajando en redes prototípicas para clasificación de género e instrumento musical. En una red prototípica, dados atributos de audio $z_x \in \mathbb{R}^{N \times D}$, y un conjunto de prototipos $z_p \in \mathbb{R}^{M \times D}$, en donde N es la cantidad de audios de un batch, M es la cantidad de prototipos y D su dimensionalidad, se calcula un vector $S = e^{||z_x - z_p||_2^2}$ que refleja la distancia entre cada audio del batch y cada prototipo. Finalmente, una capa lineal toma como entrada S y devuelve los logits correspondientes a las clases, y se utiliza una suma pesada de la entropía cruzada y la distancia $L2$ entre cada audio y su prototipo más cercano como función de costo. Este segundo término en la función de costo promueve que los prototipos pertenezcan o sean similares a audios del conjunto de entrenamiento. Una vez entrenado el modelo, este tiene el beneficio de ser explicable, ya que dada una predicción se puede inspeccionar S para determinar qué prototipos contribuyeron más a la misma. Al mismo tiempo, si las representaciones z_x son **inversibles**, podemos escuchar los prototipos e identificar los patrones sonoros que son más útiles para distinguir alguna clase, por ejemplo, heavy metal en el caso de clasificación de género musical. Además, el usuario podría editar interactivamente los prototipos, eligiendo otros puntos del espacio de atributos que sean más representativos de la clase, y alterando o corrigiendo el comportamiento de un clasificador ya entrenado. Por ende, es deseable tener una representación de audio lo suficientemente semántica como para resolver la tarea de interés con un modelo de baja capacidad, y al mismo tiempo ser inversible para poder sonificar prototipos. Sabemos que por ejemplo un espectrograma es inversible, pero como se vió en la Tabla 4.1, su desempeño al utilizarse como atributo es mucho peor que EnCodecMAE y otras representaciones de audio. Si logramos invertir las representaciones de EnCodecMAE, satisfaceremos ambas propiedades deseables.

Como un primer acercamiento, entrenamos decoders g que dado un audio x, toman como entrada la representación $f(x)$ generada por EnCodecMAE y predicen las representaciones $e(x)$ generadas por el encoder de EnCodec. Una vez entrenado el decoder, se puede aplicar sobre la representación de EnCodecMAE $g(f(x))$ obteniendo una estimación de las representaciones internas de EnCodec. Luego, estas representaciones se pueden invertir utilizando el decoder de EnCodec d. Es decir, si EnCodec es un autoencoder que reconstruye de forma perfecta, y el decoder g estima de forma perfecta las representaciones de EnCodec, luego se tiene $\hat{x} = d(g(f(x))) = x$. Si bien la reconstrucción no será perfecta en ambos casos, se puede conseguir una buena aproximación si:

1. EnCodec es un buen autoencoder con baja distorsión y buena calidad perceptual.
2. Las representaciones de EnCodecMAE no pierden información del audio original.
3. El decoder g tiene capacidad suficiente para obtener las representaciones de EnCodec a partir de las representaciones de EnCodecMAE.

El primer item se cumple satisfactoriamente, dada la buena capacidad de reconstrucción de EnCodec [108]. El segundo item, en parte es lo que queremos averiguar en esta sección. Si bien no perder información puede ser un requisito muy fuerte, e incluso perjudicial para la representación en si, que se conserven los detalles perceptualmente

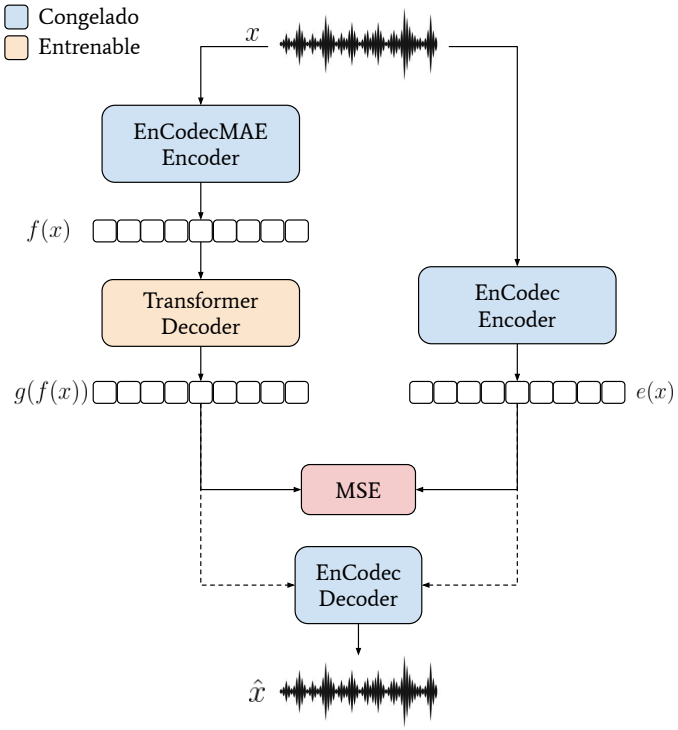

Figura 6.7: Sistema propuesto para invertir representaciones de EnCodecMAE.

relevantes es algo deseable si queremos ganar interpretabilidad sobre las representaciones, ya que nos permite escucharlas. También es deseable para ciertas aplicaciones, como por ejemplo, separación de fuentes o reducción de ruido. Estas tareas implican regresión de audio, y puede ser beneficioso realizar la separación o reducción de ruido en el espacio de representaciones y luego invertir esa representación en lugar de entrenar un modelo end-to-end que tome una forma de onda y devuelva otra.

Decidimos que el decoder g consista de una capa de Transformer con 12 cabezas de atención y dimensionalidad de 768, cuyo objetivo es minimizar el error cuadrático medio entre su salida y la del encoder de EnCodec. En la Figura 6.7 se ilustra el sistema de inversión propuesto. Cabe notar que tanto la salida del encoder de EnCodec como la de EnCodecMAE tienen la misma longitud, aunque distinta dimensionalidad. Para adaptar la dimensionalidad de la salida de EnCodecMAE a la de EnCodec (de 768 a 128) se utiliza una capa lineal. Además cabe notar que a diferencia del Decoder de EnCodecMAE, este decoder no ve tokens de máscara ya que durante inferencia no se realiza enmascarado de la señal. El modelo es entrenado con la mezcla de Audioset, FMA y LibriLight durante 50000 iteraciones utilizando una tasa de aprendizaje fija de $1 \cdot 10^{-4}$.

En la Figura 6.8 se puede observar un ejemplo de inversión de las representaciones de EnCodecMAE para el modelo EC→EC base. Viendo el espectrograma completo, se observa que la estimación del decoder conserva la estructura y detalles del audio original. Resulta interesante que el decoder tiene el efecto secundario de realizar super-resolución de audio; en este caso, la señal original estaba limitada en banda

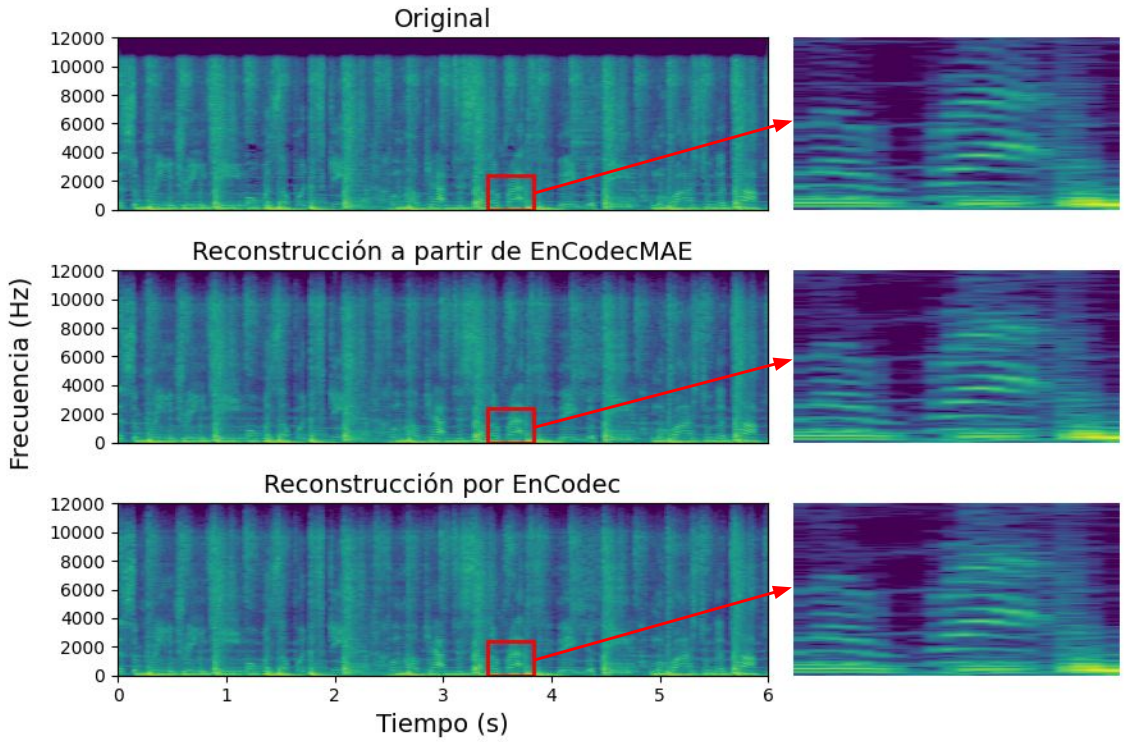

Figura 6.8: **Arriba:** espectrograma del audio original correspondiente a 6 segundos de una canción de Hip-Hop en GTZAN. **Medio:** espectrograma del audio obtenido tras invertir las representaciones del modelo EC→EC. **Abajo:** espectrograma del audio obtenido tras utilizar EnCodec para codificar y decodificar el audio original.

alrededor de los 10.5 kHz, pero la reconstrucción completó las partes faltantes del espectro. Se puede observar que este fenómeno también ocurre al codificar y decodificar con EnCodec, probablemente porque vió principalmente señales con ancho de banda completo durante el entrenamiento, por lo que tiene un sesgo a completar el espectro incluso si no estaba presente en el audio original. Mirando en más detalle el espectrograma, se puede observar que a diferencia de los primeros armónicos, los de mayor órden no están reconstruídos de forma precisa. Esto ocurre tanto en las reconstrucciones de EnCodec como las de EnCodecMAE, lo cual tiene sentido ya que estamos entrenando el decoder para predecir las representaciones de EnCodec.

Como se observa en la Figura 6.9, invertir las representaciones del modelo Mel256→EC no es tan sencillo. Se pueden ver los MSE durante el entrenamiento entre las representaciones predichas y las de EnCodec, utilizando decoders con distintas cantidades de capas. Se observa que crecer el tamaño del decoder no cambia la capacidad de reconstrucción del modelo cuando se usan melspectrogramas como entrada. Esto indicaría que la señal necesaria para reconstruír está ausente en la representación. Como se discutió en la sección 2.3.2, existe un compromiso entre distorsión y calidad perceptual. Hipotetizamos que a pesar del alto error de reconstrucción, las características relevantes del audio están presentes en la representación

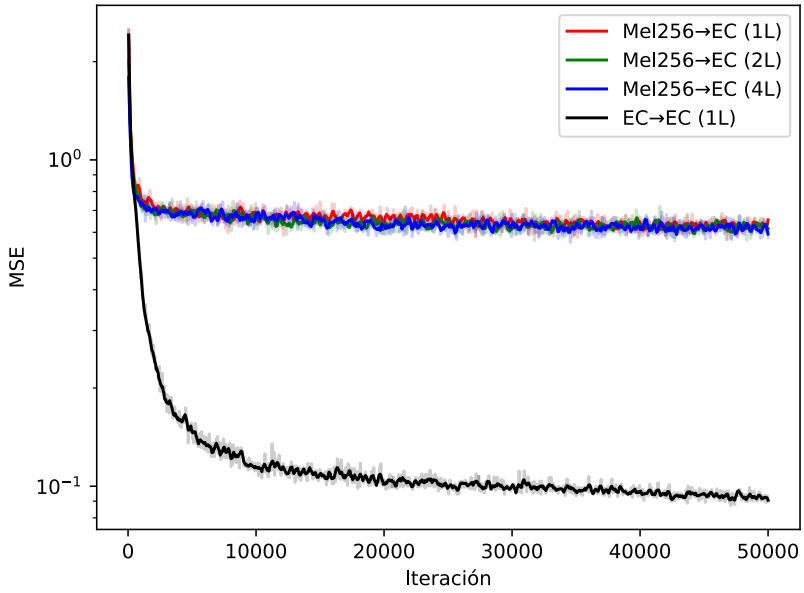

Figura 6.9: MSE durante el entrenamiento para distintos decoders del modelo EC→EC y Mel256→EC bases con distintas cantidades de capas (1,2 y 4).

de Mel256→EC, e introduciendo modelos generativos se podrían sonificar en trabajos futuros.

En el trabajo realizado junto a UPF, si bien las representaciones del modelo EC→EC se podían sonificar con buena calidad, resultaba difícil trabajar en el espacio de secuencias temporales con la resolución de 75 Hz. Esto es porque, si por ejemplo queremos sonificar un nuevo punto que es la interpolación lineal entre dos prototipos, ocurrirá que todos los elementos de la secuencia se alteran en la misma dirección, y se pierde coherencia temporal. Si bien se puede percibir el timbre y la envolvente espectral, los patrones temporales son dificiles de interpretar si el prototipo no se corresponde exactamente a un audio del conjunto de entrenamiento.

Para resolver este problema, comenzamos a trabajar en representaciones con menor resolución temporal, y por ende con mayor compresión y pérdida de información. Luego, mediante un modelo de difusión se lograron sonificar estas representaciones. En la Figura 6.10 se muestra el modelo de difusión propuesto. Este modelo toma las representaciones de EnCodec y les agrega progresivamente ruido gaussiano (proceso de difusión hacia adelante). Luego un transformer debe predecir el ruido gaussiano agregado en el instante t para poder revertir este proceso (proceso de difusión hacia atrás). Se informa a la red de cual es el instante t concatenando una representación del mismo a la secuencia de entrada del transformer. A su vez, se le concatena una representación de EnCodecMAE para usarla como señal de condicionamiento y hacer que el modelo de difusión genere una señal de audio parecida a la original durante inferencia. Esta representación de EnCodecMAE se obtiene tras agregar un CLS token al comienzo de la secuencia de embeddings a la salida de EnCodecMAE, y utilizar

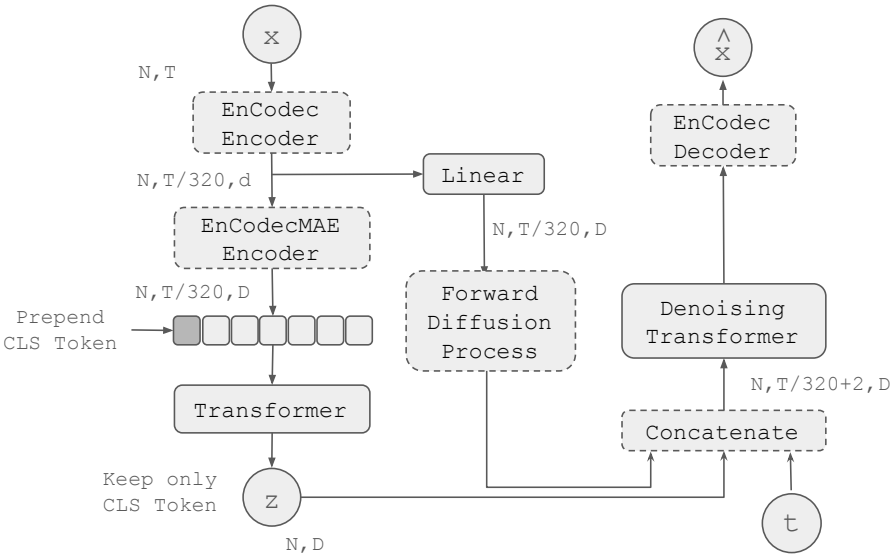

Figura 6.10: Modelo de difusión desarrollado para sonificar embeddings de EnCodecMAE.

otro transformer para que comprima la secuencia en ese único token. Este token luego sirve como resumen de la señal de audio y como señal de condicionamiento. El sistema se puede ver como un encoder que devuelve una representación z de la señal de audio x resumiendo las salidas de EnCodecMAE, y un decoder basado en difusión, que dado z puede generar un audio.

El sistema desarrollado logró un buen desempeño en las tareas de clasificación de género musical e instrumento musical, permitiendo al mismo tiempo escuchar audios prototípicos que definen a las distintas clases, y conduciendo a un modelo interpretable. En el sitio web[1] se pueden escuchar algunos ejemplos de prototipos y encontrar un análisis más detallado de los resultados.

Si bien en este trabajo se utilizó un modelo de difusión para la sonificación, he explorado otras alternativas como modelos de lenguaje obteniendo resultados satisfactorios. En este sitio web[2] se pueden escuchar algunos resultados.

[1] https://palonso.github.io/pecmae/
[2] https://leonardopepino.latentsound.com/blog/2023/audiolm/

7

Conclusiones y trabajos futuros

7.1. EnCodecMAE

En este trabajo doctoral, nos propusimos diseñar un modelo de representaciones de audio que sea útil para la resolución de tareas diversas relacionadas a la música, habla y sonidos ambiente. Esto llevó al desarrollo de EnCodecMAE, un modelo de representación de audio general entrenado con señales de audio enmascaradas para predecir representaciones de ese mismo audio sin enmascarar.

7.1.1. Señal objetivo y de entrada

Particularmente, las representaciones que se usan como objetivo para entrenar al modelo provienen de EnCodec, un codec de audio neuronal. Este codec, al generar secuencias discretas, nos permite plantear al problema como uno de clasificación y adaptar ideas aplicadas en el campo del procesamiento del habla a señales más generales de audio. Modelos como HuBERT, utilizan clusters de MFCCs como representación del audio objetivo, los cuales son atributos enfocados en capturar ciertas características del habla como el contenido fonético, pero que descartan información quizás relevante para otras tareas de habla u otro tipo de sonidos. Al utilizar EnCodec u otro codec de audio neuronal, como estas representaciones están enfocadas en lograr una buena reconstrucción de cualquier señal de audio, se evita descartar información que sea auditivamente perceptible, y por ende, no se descarta información que podría ser relevante para tareas que los humanos somos capaces de resolver (con nuestras limitaciones perceptuales y cognitivas).

Como trabajo futuro, se pueden pensar en alternativas a EnCodec como señal objetivo. Dado el creciente interés en desarrollar tokenizadores de audio, es esperable que en el futuro cercano surjan cada vez más codecs de audio neuronales, que incorporen semántica de los sonidos, sean más contextuales, posean mejor calidad de reconstrucción, o utilicen menos cuantizadores o secuencias más cortas para un mismo largo de señal. Si bien exploramos el uso de cuantizadores aleatorios y de WavTokenizer, creemos que sería interesante incorporar otros tipos de tokenizadores, que por ejemplo, utilicen transformers en su arquitectura y potencialmente sean menos locales. Otra alternativa es utilizar ensambles de representaciones objetivos, pensados para capturar distintas características de la señal. Por ejemplo, se podrían combinar: clusters de MFCCs (para capturar contenido fonético), clusters de capas

intermedias de WavLM (para capturar contenido fonético más contextualizado), y EnCodec (para capturar contenido general de audio). Del mismo modo, se pueden pensar otras representaciones objetivos, como clusters de pitch, que podrían ayudar en tareas de transcripción musical.

Por otro lado, continuar el preentrenamiento utilizando representaciones internas de EnCodecMAE como señal objetivo, lo cual llamamos refinamiento iterativo de la señal objetivo y fue una idea tomada de HuBERT, llevó a mejoras en la mayoría de las tareas downstream. Consideramos que la elección de la capa para generar esta señal objetivo refinada, junto con la elección del K en K-Means, puede tener un impacto grande en el desempeño en tareas downstreams. En trabajos futuros se puede realizar una búsqueda y análisis de estos hiperparámetros. En una etapa de este trabajo doctoral se intentó realizar un K-Means online que se ajuste durante el entrenamiento, aunque no tuvo éxito ya que probablemente requería un ajuste más fino de los hiperparámetros implicados. En cierto modo, esta extensión del refinamiento iterativo de la señal objetivo guarda similaridades con redes maestro-estudiante y trabajos como Data2Vec y DinoSR.

También, se pueden explorar representaciones de entrada alternativas. Nuestros resultados al respecto, en donde los melspectrogramas resultaron ser las mejores representaciones de entrada, sugieren que incorporar restricciones perceptuales en la señal de entrada es beneficioso. Por ejemplo, se podría experimentar con representaciones más fieles a la percepción como cocleagramas.

7.1.2. Eficiencia computacional

Por otro lado, nos encontramos con limitaciones de recursos computacionales, lo que llevó a adoptar ciertas estrategias que reduzcan el costo computacional del preentrenamiento. En particular, utilizamos una arquitectura de Masked Autoencoder y audios con un largo de 4 segundos. Estas dos decisiones reducen significativamente el largo de las secuencias de entrada, y en consecuencia, los tiempos de preentrenamiento. Si bien el modelo presentado es eficiente, pudiéndose entrenar la versión Large en 5 días con 2 GPUs RTX3090, aún hay espacio para mejoras. Por ejemplo, se puede experimentar con decoders convolucionales, los cuales serán más livianos y fueron exitosos en Data2Vec 2.0. También si la señal objetivo tuviera una menor resolución temporal, podríamos representar los 4 segundos de audio con secuencias más cortas reduciendo significativamente el costo computacional, o incorporar audios más largos en el preentrenamiento sin incrementar el costo computacional. Una manera en la que se podría reducir el largo de las secuencias generadas por EnCodec es realizando promedios locales en la representación sin cuantizar y utilizando el cuantizador de EnCodec, o realizando K-Means, para discretizar estas nuevas representaciones. De esta manera, si promediamos cada 2 vectores consecutivos de la secuencia, reducimos el largo a la mitad. Si bien esta estrategia reduce el costo computacional, hay que determinar si no se ve deteriorada la calidad de las representaciones.

En particular, hay muchas preguntas abiertas alrededor de este tema: ¿cuál es la duración óptima que deberían tener los audios de preentrenamiento?, ¿cuánto

realmente aprenden a contextualizar los transformers?, ¿cuál es la resolución temporal óptima? ¿75 embeddings por segundo? ¿1 por segundo? ¿20? En este sentido, creo que puede ser fructífero explorar largos de ventana dinámicos. Por ejemplo, si tenemos un segundo de silencio, quizás no sea necesario representarlo como 75 embeddings, sino más bien como uno solo, mientras que si tenemos habla rápida quizás queremos utilizar más embeddings por segundo. En cierto modo, explorar estas direcciones se interseca con problemas más clásicos del procesamiento de señales como determinar la estacionariedad de una señal o detectar cambios en la misma.

Por otro lado, se han explorado en la literatura arquitecturas más eficientes que un Transformer y que logran un desempeño comparable [264–267]. En el campo del procesamiento de audio, varios trabajos [268, 269] han reemplazado las capas de Transformers por convoluciones dinámicas, las cuales utilizan kernels dependientes de la entrada. A su vez, se pueden combinar este tipo de arquitecturas más eficientes con técnicas de destilación. Se podría entrenar una versión de EnCodecMAE con más parámetros y datos, y luego aproximar este modelo maestro con una red estudiante más pequeña que utilice convoluciones dinámicas o menos capas de transformer. Otras direcciones para incrementar la eficiencia del modelo puede ser el uso de cuellos de botella en la arquitectura Transformer [270, 271]. Por ejemplo, se puede realizar un promedio local u otro método para la reducción del largo de la secuencia en cada capa del modelo, lo cual lleva a una mayor eficiencia computacional y fuerza al modelo a contextualizar ventanas de mayor largo.

7.1.3. Desempeño

Uno de los grandes desafíos en este campo de investigación es cómo evaluar las distintas representaciones de audio. Especialmente, al buscarse un modelo general de audio, uno debería verificar que este es útil para todas las tareas posibles de audio. Sin embargo, esto no es viable y se hace necesario elegir un conjunto acotado de tareas. Al mismo tiempo, una vez elegidas las tareas hay que definir cómo se va a utilizar la representación para resolver esta tarea. Las opciones posibles en cuanto al diseño del modelo downstream son infinitas, y cuál es la mejor opción va a depender del modelo upstream y la tarea downstream entre otros factores. En general, hemos observado que hay dos corrientes opuestas:

- Dado un modelo upstream, se busca obtener el mayor desempeño posible en la tarea downstream. En general, estos trabajos realizan finetuning del modelo upstream y una exploración extensiva de hiperparámetros.
- Dado un modelo upstream, se busca analizar si las representaciones del mismo poseen propiedades deseables para resolver distintas tareas downstream. En este caso, se suelen extraer atributos del modelo upstream y utilizar modelos downstream sencillos, como regresión logística.

Ambos métodos tienen sus ventajas y desventajas. En el caso de realizar finetuning, es cuestionable si lo que se está evaluando es la representación en si o la capacidad del modelo upstream de adaptarse a los datos de la tarea downstream. Por ejemplo,

si se cuenta con muchos datos en la tarea downstream y el modelo upstream tiene mucha capacidad, es dificil determinar si un buen desempeño en la tarea downstream proviene de que las representaciones del modelo upstream eran buenas, o si el modelo upstream cambió por completo sus representaciones y solo fue beneficiado por la inicialización. Al mismo tiempo, el finetuning es costoso computacionalmente, al igual que explorar los distintos hiperparámetros. Si no hay un protocolo establecido de como realizar el finetuning, es posible que un modelo resulte 'mejor' que otro porque se realizó una exploración más extensiva de hiperparámetros, beneficiando a quienes tienen mayor poder de cómputo.

En el caso de realizar extracción de atributos y utilizar modelos downstream sencillos, es posible que el modelo downstream no esté pudiendo aprovechar de la mejor manera la representación del modelo upstream. Por ejemplo, utilizar un modelo downstream lineal puede resultar muy restrictivo, ya que estamos requiriendo que la representación este organizada de una forma en que sea linealmente separable para la tarea de interés. A su vez, en nuestro caso que deseamos una representación útil para muchas tareas, sería optimista pretender que exista una separación lineal para todas las clases de todas las tareas al mismo tiempo, y quizás se beneficiaría a representaciones que contengan un mayor número de dimensiones. Al utilizar un perceptrón multicapa, relajamos los requerimientos de la representación, aunque siga sin ser el modelo downstream que obtenga el mejor desempeño posible.

Creemos que aún hace falta un mayor trabajo por parte de la comunidad del procesamiento de habla y audio para uniformizar los protocolos de evaluación de representaciones. Si bien existen benchmarks que han permitido comparar distintos trabajos de la literatura como SUPERB en habla, y HEAREval en audio general, creemos que aún se pueden mejorar:

- SUPERB es muy costoso computacionalmente, haciendo que evaluar un modelo lleve varios días. Algunas variantes como MiniSUPERB han disminuído significativamente el costo computacional, aunque aún no han sido adoptadas masivamente.
- HEAREval es menos costoso computacionalmente que SUPERB y realiza una pequeña búsqueda de hiperparámetros. Sin embargo, algunas de las tareas no son muy reconocidas en la comunidad, como Beehive y Mridangam Tonic/Stroke. Al mismo tiempo, faltan otras tareas de interés como reconocimiento de habla, identificación de hablante, clasificación de instrumentos musicales, etc.
- Se pueden evaluar otros aspectos que no se relacionen con el desempeño, como la eficiencia computacional tanto en inferencia como preentrenamiento, o características intrínsecas de la representación como medidas de disentanglement o dimensión efectiva, que podrían enriquecer el análisis y llevar a un entendimiento más profundo de los compromisos que existen en el aprendizaje de representaciones.

El modelo propuesto fue evaluado en tareas diversas provenientes del benchmark HEAREval y en la tarea de ASR del benchmark Superb. Los resultados obtenidos en estas tareas fueron positivos, superando en algunos casos al estado del arte y

obteniendo un buen desempeño en la mayoría de las tareas, lo cual muestra que la representación a la que llegamos es "general". A pesar de esto, el desempeño en la tarea de ASR sigue siendo inferior al estado del arte, donde modelos especializados de habla logran un mejor desempeño. A pesar de esto, nuestro trabajo (hasta donde sabemos) fue el primero en plantear la evaluación de modelos generales de audio en una de las tareas más desafiantes del procesamiento del habla, y los resultados obtenidos son esperanzadores y comparables con los de modelos especializados anteriores al 2020.

Es posible que una búsqueda más extensiva de hiperparámetros, particularmente de la señal objetivo y la arquitectura, lleve a mejorar el desempeño en ASR sin perjudicar el desempeño en otras tareas. Sin embargo, creemos que existen algunos desafíos en esta dirección debido a que pueden haber características de la representación que beneficien a la tarea de ASR pero perjudiquen a otras tareas. Por ejemplo, hemos visto que cuando se logra un mejor desempeño en clasificación de notas musicales, también ocurre un deterioro en el desempeño en tareas relacionadas al reconocimiento de comandos de voz o eventos acústicos. Una posible explicación de este fenómeno es que la información de la frecuencia fundamental, si bien beneficia a la tarea de clasificación de notas musicales, podría perjudicar a otras tareas en donde deseamos una invariancia a la frecuencia fundamental. Por ejemplo, queremos que un sistema de reconocimiento de habla funcione ante distintas frecuencias fundamentales de la voz, distintos timbres, ruidos de fondo, etc. Es decir, queremos que sea invariante a estas características. Pero al mismo tiempo, si fueran invariantes a estas características, las representaciones no serían útiles para identificación de hablante o estimación de frecuencia fundamental. Creemos que la solución a este problema consiste en aprender una representación en donde las distintas características estén separadas. Por ejemplo, una dimensión del embedding podría corresponder a la frecuencia fundamental, mientras que otro subconjunto de dimensiones podría codificar los fonemas. De esta forma, la representación preservaría las distintas características de forma que un modelo downstream pueda acceder a las mismas y también aprender a ignorar fácilmente aquellos factores que no son necesarios para resolver la tarea downstream. Las técnicas y análisis de *disentanglement* se enfocan en lograr esta separación de los distintos factores generativos de la señal, aunque aún resulta desafiante, especialmente cuando no se utiliza supervisión.

Por otro lado, creemos que realizar un promediado en el tiempo es una operación subóptima para obtener una representación a nivel instancia a partir de una representación temporal. Si bien experimentamos brevemente con distintos métodos como aprender autoencoders que resuman la secuencia temporal en un vector único, las representaciones promediadas producían un mejor desempeño. Creemos que encontrar alternativas superadoras al promediado temporal es una línea de investigación interesante y que puede traer mejoras importantes en el desempeño. Particularmente, es un problema cuya solución podría impactar positivamente en otras áreas de investigación y aplicaciones que requieran resumir series de tiempo.

7.1.4. Análisis del diseño

En el capítulo 4 realizamos un análisis extensivo de cómo impactan distintos factores de diseño e hiperparámetros en el desempeño downstream. Algunos de los resultados desprendidos de este análisis fueron:

- La representación utilizada como entrada tiene un impacto importante en el desempeño del modelo, siendo los melspectrogramas las representaciones que mejor desempeño tuvieron (ver sección 4.1).
- La capa óptima para extraer las representaciones depende de la tarea downstream, y no siempre es la última (ver sección 4.2). Para EnCodecMAE, en general las activaciones de la última capa tienen un buen desempeño en la mayoría de las tareas, aunque para clasificación de notas musicales, las primeras capas son mejores. Intuímos que la capa óptima depende del nivel de contextualización requerido por la tarea downstream. En tareas que requieren de información fina local, las primeras capas pueden ser una mejor opción que las últimas, mientras que en tareas que requieren identificar patrones temporales de largo plazo, por ejemplo para reconocer palabras, las últimas capas suelen ser mejores.
- En general, entrenar más tiempo lleva a representaciones con un mejor desempeño downstream. Creemos que los 500k pasos de entrenamiento utilizados en este trabajo fueron adecuados dado el budget computacional (ver sección 4.3).
- Identificamos que el modelo propaga información local hacia el decoder mediante las conexiones residuales utilizadas en la normalización del transformer (ver sección 4.4), lo cual explica por qué la última capa del encoder se comporta distinto a la penúltima. A su vez, propusimos una variante de EnCodecMAE para evitar que el transformer tenga que propagar esta información local al proveer de una conexión residual directa entre la salida de la primera capa y la última del encoder (ver sección 4.8.2).
- Para la mayoría de las tareas downstream, el primer cuantizador de EnCodec es el que mejor funciona como señal objetivo. Utilizar los primeros 8 cuantizadores dándoles un peso decreciente (mayor peso al primero y menor peso al último) brinda algunas mejoras marginales en el desempeño de la mayoría de las tareas, y curiosamente una mejora importante en la tarea de reconocimiento de emociones (ver Figura 4.7).
- Mostramos que utilizar EnCodec como señal objetivo lleva a un mejor desempeño que utilizar cuantizadores aleatorios como los utilizados en BEATs (ver Tabla 4.3). Como trabajo futuro se pueden explorar nuevos tokenizadores y cuantificar su efecto en la calidad de las representaciones.
- Encontramos que para la mayoría de las tareas downstream, una proporción de enmascarado alrededor del 50% resulta óptima, mientras que un enmascarado escaso o excesivo lleva a un deterioro del desempeño downstream (ver Figura 4.10).

- Utilizar proporciones de enmascaramiento o largos de máscara aleatorios durante el preentrenamiento no trajo mejoras respecto a buscar un valor óptimo fijo. Como trabajo futuro se puede explorar el uso de curriculum learning, generando máscaras más difíciles (mayor proporción y largo de enmascaramiento) a lo largo del preentrenamiento.
- Se propusieron variantes de EnCodecMAE que produjeron mejoras en el desempeño. Se exploró la generación de señales ruidosas durante el preentrenamiento (siguiendo la receta de WavLM), el uso de una conexión residual entre la primera y última capa del encoder, y el uso de registros. Todas las variantes, incluída la combinación de las tres, produjeron mejoras en el desempeño. Queda como trabajo futuro explorar si con estos modelos también se producen mejoras al incrementar el tamaño (de Base a Large), y hacer un refinamiento iterativo de la señal objetivo.

7.2. Alineamiento con representaciones cerebrales

7.2.1. Resumen de los resultados

En el capítulo 5 analizamos si las representaciones aprendidas por EnCodecMAE, y otros modelos de audio, son similares a representaciones cerebrales obtenidas a partir de fMRIs de la corteza auditiva. Particularmente, continuamos el trabajo de Tuckute et al. [8] incorporando modelos de audio más recientes y de mejor rendimiento, y nuevos tipos de análisis. Siguiendo el trabajo original, estudiamos si son similares las representaciones, tanto usando regresores lineales regularizados como análisis de similaridad de representaciones (RSA), y analizamos si hay una correspondencia estructural entre los modelos de audio y la organización de la información en la corteza auditiva. Expandiendo las técnicas de análisis del trabajo original, medimos si existe una correlación entre el desempeño de los modelos en distintas tareas de audio y la similaridad con el cerebro, y medimos esta similaridad a lo largo del preentrenamiento. También al incorporar más modelos, pudimos adquirir ciertas intuiciones de qué criterios de diseño generan un mayor o menor grado de alineamiento entre las representaciones aprendidas y las cerebrales. Algunos resultados interesantes que obtuvimos son:

- Los modelos más recientes de audio, incluyendo EnCodecMAE, exhiben una mayor similaridad con las representaciones obtenidas a partir de fMRI. Esto indica que existe una tendencia a generar modelos con representaciones cada vez más alineadas a las de nuestra corteza auditiva. A su vez, la mayoría de los modelos de audio son mejores predictores de la representación cerebral que un modelo espectro-temporal clásico de la audición.
- Los modelos que son preentrenados con datos más diversos también exhiben un mayor alineamiento con el cerebro. Particularmente, EnCodecMAE entrenado sólo con datos de habla exhibe el menor alineamiento, mientras que las variantes

entrenadas con Audioset, que contiene una gran variedad de datos, y con la mezcla de datasets son las que mayor alineamiento exhiben.

- Se observa una correlación de Pearson positiva entre la cantidad de parámetros de un modelo y la similaridad con las representaciones cerebrales. Sin embargo, para una misma cantidad de parámetros se observa también una gran variabilidad en los valores de similaridad, lo cual indica que si bien hay una tendencia, no es un factor determinante.
- Observamos una correlación de Pearson elevada entre el desempeño de 33 modelos en distintas tareas downstream de audio y la similaridad con representaciones cerebrales (ver tabla 5.1). En particular, las tareas relacionadas con detección de eventos acústicos (FSD y ESC) fueron las que mayor correlación presentaron para ambas medidas de similaridad (RSA y regresión lineal regularizada) y ambos conjuntos de datos. La métrica que mayor correlación presentó fue la global, obtenida mediante un promedio de z-scores de las métricas obtenidas en las 6 tareas de HEAREval. En este caso, para todos los análisis y datasets la correlación fue significativa ($p < 0.05$).
- Al realizar el análisis de correlación del item anterior con las 6 componentes independientes identificadas en [242] a partir de fMRIs, observamos que el desempeño en la tarea de clasificación de notas musicales (NS) es el que mayor correlación con las componentes que responden a contenido en baja y alta frecuencia tiene. Creemos que esto se debe a que es una tarea que requiere de información más local, y que resuelven mejor las primeras capas de los modelos, ya que las componentes LF y HF responden a patrones en la corteza auditiva primaria relacionada a una etapa de procesamiento temprano de los sonidos. Por otro lado, la tarea de detección de eventos acústicos (FSD) muestra una mayor correlación con la componente asociada a patrones verticales en el espectrograma (corta duración y banda ancha) y horizontales (tonales y estacionarios). Por último, la medida de desempeño global es la que mayor correlación tiene con las componentes de habla y música. Cabe destacar que la medida de desempeño global es la única que exhibe una correlación significativa ($p < 0.05$) con todas las componentes. Por otro lado, en cuanto a las tareas de habla, solo reconocimiento de emociones muestra una correlación significativa con la componente de habla. De todas formas, es notable que las tareas de habla muestren mayor correlación con la componente asociada al habla.
- Las representaciones de los modelos incorporados exhiben en su mayoría una correspondencia estructural con las etapas de procesamiento en la corteza auditiva (ver Figuras 5.11 y 5.12). Estos resultados se alinean con los obtenidos por Tuckute y muestran una organización jerárquica emergente en los modelos de audio que se corresponde con la organización jerárquica en el cerebro.
- Durante el preentrenamiento de EnCodecMAE, observamos que el alineamiento con la corteza auditiva crece a pesar de que el modelo se está optimizando para una tarea distinta. Resulta interesante que las primeras capas se vuelven más similares a las representaciones de fMRI de la corteza auditiva primaria, mientras

que las últimas capas se vuelven menos similares a las representaciones de esta región, y en cambio se vuelven más similares a las representaciones cerebrales en la región posterior, asociada a un procesamiento más tardío de la señal auditiva. Este tipo de dinámica entre capas durante el preentrenamiento explica cómo se origina la correspondencia estructural durante el preentrenamiento del modelo, y también brinda pistas de cómo organiza la información EnCodecMAE a lo largo de las capas y el preentrenamiento.

7.2.2. Limitaciones del estudio

Una de las principales limitaciones de este estudio se encuentra en los datos. Realizar experimentos con fMRI es costoso en términos de tiempo y diseño experimental. A su vez, existen múltiples fuentes de ruido durante la adquisición, las cuales luego se buscó corregir mediante promediado de adquisiciones y corrección por atenuación. Sin embargo, estas no son soluciones totales y el ruido sigue presente e influye en los resultados. A pesar de esto, hay ciertas verificaciones básicas que sustentan que las observaciones no son resultado meramente del ruido:

- En el trabajo original se repitieron los distintos análisis de similaridad utilizando los modelos con sus parámetros permutados aleatoriamente. De esta forma, comprobaron que un modelo aleatorio posee una similaridad muy por debajo del modelo computacional de la audición que sirve de baseline. Esto indica que el preentrenamiento efectivamente lleva a un alineamiento entre las representaciones del modelo y las del cerebro humano. Nosotros no repetimos este análisis de permutación ya que el costo computacional era elevado y los resultados deberían ser similares permutando aleatoriamente otros modelos de redes neuronales.
- Si bien no realizamos la permutación aleatoria de pesos, pudimos comprobar en la sección 5.3.7 que EnCodecMAE con pesos aleatorios (sin entrenar), posee una similaridad (medida mediante RSA) muy baja con las representaciones de la corteza auditiva, y que a medida que el modelo se preentrena, esa similaridad aumenta hasta superar al baseline y a otros modelos. Esto indica que efectivamente la optimización del modelo upstream es la que lleva al alineamiento observado entre representaciones de audio y del cerebro.
- La mayoría de los modelos analizados exhibe una similaridad mayor que el modelo computacional de la audición utilizado como baseline. Este modelo baseline fue diseñado para aproximar aspectos de la audición humana, y sin embargo, es superado ampliamente por modelos de audio entrenados para resolver tareas sin un foco explícito en imitar aspectos de nuestra percepción y fisiología.

Otras limitaciones del estudio surgen de la resolución temporal de las señales de fMRI, las cuales fueron adquiridas usando un TR de 3.4 segundos con un tiempo de adquisición de 1 segundo. Esto significa que se realizó la medición por 1 segundo, y luego durante un intervalo de 2.4 segundos se presentó el estímulo sonoro con el

resonador en silencio[1]. Por lo tanto, las mediciones se toman cada 3.4 segundos y duran 1 segundo, y en consecuencia no podemos analizar si los modelos capturan detalles temporales finos en una forma similar al cerebro. Si quisiéramos comparar las representaciones con una mayor resolución temporal, deberíamos utilizar otro tipo de representaciones del cerebro, obtenidas mediante técnicas como electroencefalogramas (EEG) o magnetoencefalogramas (MEG) que permiten registrar la actividad neuronal con una mayor resolución temporal, a expensas de una menor resolución espacial.

Otro aspecto a tener en cuenta es la batería de estímulos utilizados, la cual representa un conjunto muy acotado de los sonidos que experimentamos cotidianamente. Particularmente, resulta difícil presentar a un sujeto una gran cantidad de estímulos mientras se encuentra en un resonador magnético, ya que se volvería muy extenso el experimento. En ese sentido, contar con solo 165 estímulos conlleva limitaciones en el análisis. Por ejemplo, al ser mucho mayor la dimensionalidad de las representaciones de audio que la cantidad de estímulos, se vuelve imprescindible el uso de regularización en los modelos lineales. Al mismo tiempo, los estímulos seleccionados, los cuales contienen música, habla en distintos idiomas, sonidos vocales y no vocales de humanos y animales, y sonidos mecánicos, de la naturaleza y ambiente, se van a alinear mejor con ciertos conjuntos de datos de preentrenamiento. Esto podría explicar por qué los modelos preentrenados con Audioset o mezclas de audios muestran una mayor similaridad con el cerebro que los modelos entrenados solo con habla. En este sentido, se complica llegar a la conclusión de que modelos preentrenados con sonidos más diversos son más similares al cerebro, ya que la medida de similaridad depende de los estímulos elegidos, y quizás si estos fueran solo de habla, los modelos de habla exhibirían una mayor similaridad.

Por otro lado, en el análisis de regresión los modelos lineales pueden ignorar contenido de la representación del modelo que se desvía del comportamiento del cerebro, y que puede por ejemplo llevar a ataques adversarios u otros comportamientos que difieren del humano. Por el contrario, el análisis de similaridad de representaciones (RSA) utiliza la totalidad de las representaciones para medir la similaridad, pero es sensible a ciertas transformaciones lineales como un escalamiento anisotrópico de la representación. En este sentido, RSA nos da una idea de si globalmente ambas representaciones son similares, lo cual implica que las direcciones de mayor varianza van a ser las que más impacto tienen. A pesar de la complementariedad de ambas métricas, se observaron resultados similares para ambos análisis y conjuntos de datos, lo cual nos da cierta tranquilidad sobre la robustez de las conclusiones. Cabe notar que el campo del análisis de similaridad de representaciones está en constante evolución y existen muchos métodos alternativos a los utilizados, cambiando la distancia utilizada para calcular las RDM y las técnicas para compararlas.

Si bien pudimos comprobar que la mayoría de los modelos exhiben una corres-

[1] Esto se conoce como *sparse scanning*, una técnica utilizada en estudios auditivos con fMRI para evitar que el ruido del resonador interfiera con la percepción del estímulo. Además, debido al retraso hemodinámico de la señal BOLD —que alcanza su pico típicamente entre 4 y 6 segundos después de la estimulación neuronal—, es posible capturar la respuesta cerebral al estímulo incluso si la adquisición se realiza algunos segundos después de su presentación.

pondencia de etapas, con las primeras capas siendo mejores predictoras de la corteza auditiva primaria y las últimas de las otras regiones, esta correspondencia sólo se analizó para la región primaria y no-primaria. La resolución del fMRI impone ciertas dificultades a la hora de buscar más asociaciones entre capas de modelos y distintas regiones del cerebro. Al mismo tiempo, no todos los modelos exhiben esta correspondencia de etapas, y aún no queda claro qué factores contribuyen a que ocurra o no la misma. Creemos que no tiene sentido pensar que todas las arquitecturas y objetivos de preentrenamiento van a llevar a una organización estructural similar a la del cerebro. A modo de ejemplo, en un autoencoder tanto las primeras como las últimas capas se van a parecer más al estímulo sonoro que las capas del medio. Esto se debe a que la señal objetivo y la de entrada son iguales, por lo que las representaciones intermedias más cercanas a la entrada y salida se van a parecer entre si. A pesar de ser la última capa una etapa de procesamiento 'tardía', por la arquitectura y objetivo utilizados, debería corresponderse mejor con una etapa temprana. Si bien en muchos modelos se hace una distinción entre encoder y decoder, y se espera que la salida del encoder sea la más semántica y 'alejada' de la entrada, en otros modelos como Wav2Vec 2.0 o HuBERT, la separación no es explícita ya que no hay un bottleneck. En estos casos, se suele observar que las capas más semánticas se encuentran cerca de la mitad o un poco más hacia el final de la red, dependiendo de cuál sea la señal objetivo. Del mismo modo, al hacer un finetuning de estos modelos, si la tarea es por ejemplo ASR, es esperable que las últimas capas que están más ancladas al lenguaje sean las más semánticas, habiéndose trasladado la etapa 'tardía' desde la mitad de la red hacia el final. Otra limitación en cuanto al análisis de correspondencia estructural es que algunas capas del modelo podrían corresponderse con regiones del cerebro que no fueron medidas con el fMRI.

En cuanto al estudio de correlación entre el desempeño downstream de los modelos y el alineamiento con el cerebro (ver sección 5.3.5), surgen varias limitaciones:

- Si bien se analizaron 33 modelos, algunos rangos de desempeño no tienen buena cobertura. Por ejemplo, hay pocos modelos con un desempeño global menor a -0.5 (ver Figura 5.8). Incorporar una variedad mayor de modelos que cubran distintos valores de desempeño ayudaría a reforzar los resultados mostrados.
- Las métricas de desempeño dependen del modelo downstream utilizado y de las tareas elegidas. Si bien nuestro análisis fue exitoso en mostrar la correlación entre el desempeño global y la similaridad con representaciones cerebrales, los resultados y conclusiones podrían diferir si se eligieran otras tareas y/o modelos downstream.

7.2.3. Trabajos futuros

En trabajos futuros se podría replicar este análisis utilizando estímulos puramente de habla o de música. En esta dirección, existen varios conjuntos de datos con mediciones de fMRI de sujetos escuchando cuentos, los cuales pueden ser útiles para enriquecer el análisis de este trabajo, e identificar si modelos de habla que son buenos

en tareas como reconocimiento del habla (ASR), también se alinean mejor con el cerebro utilizando este tipo de estímulos.

Resulta interesante buscar puntos de intercambio entre las disciplinas de la neurociencia y el aprendizaje automático. De cierta manera, muchas de las arquitecturas e ideas en el mundo del aprendizaje profundo están inspirados en conceptos biológicos. El lenguaje que utilizamos al referirnos a las distintas técnicas de modelado son una muestra de esto: hablamos de redes neuronales en lugar de grafos acíclicos dirigidos diferenciables, de atención en lugar de promedios pesados dinámicamente, de memoria en lugar de vectores de estado, de aprendizaje en lugar de optimización y generalización, etc. Del mismo modo, cabe preguntarnos ¿se pueden aprovechar las representaciones del cerebro para mejorar las representaciones de un modelo? En esta dirección, creemos que puede ser interesante utilizar las representaciones cerebrales obtenidas mediante fMRI para regularizar las representaciones aprendidas por distintos modelos. Por ejemplo, se podrían introducir términos en la función de costo de EnCodecMAE que promuevan que la RDM de sus representaciones se parezca a la RDM de mediciones de fMRI. Trabajos recientes han comenzado a experimentar en esta dirección con cierto éxito en el dominio del procesamiento del habla [272, 273].

Hemos observado que a medida que avanza el preentrenamiento de EnCodecMAE, tanto la similaridad con las representaciones cerebrales como el desempeño downstream en tareas humanas crece. Algo que podemos preguntarnos es si este comportamiento es una condición necesaria. En otras palabras, ¿es el alineamiento con el cerebro humano un requerimiento para poder resolver tareas humanas?, ¿existen representaciones que sean útiles para resolver una diversidad de tareas humanas y que no sean similares a las representaciones cerebrales? Una forma posible de responder esto es buscando un contraejemplo, es decir, un modelo cuyo desempeño global sea alto, mientras que su similaridad con las representaciones cerebrales sea baja. Esto se reflejaría en las Figuras 5.8 y 5.9 como un punto en la esquina inferior derecha. De manera similar, uno podría preguntarse si existen modelos que exhiben una similaridad alta con las representaciones cerebrales pero un bajo desempeño global, lo cual se correspondería con puntos en la esquina superior izquierda de las Figuras 5.8 y 5.9. En cierto modo, la hipótesis de representaciones platónicas [228] sugiere que encontrar estos contraejemplos puede ser difícil ya que existen menos representaciones que resulten en un buen desempeño para N tareas que para $M < N$ tareas, aunque no podemos garantizar que la solución a la que llegamos como humanos sea la única posible, y es en este punto en donde resulta interesante filosóficamente preguntarnos ¿existen otros 'cerebros' posibles que también permitan resolver los problemas que afrontamos cotidianamente?.

Resulta también interesante pensar en neuroconexionismo más allá del cerebro humano. ¿Qué ocurre si preentrenamos EnCodecMAE utilizando el canto de un ave y los sonidos que la rodean cotidianamente? ¿Podremos entender mejor cómo funciona el cerebro de distintos animales preentrenando modelos con el tipo de sonidos que escuchan? Así como los modelos entrenados con sonidos humanos aprenden en sus primeras capas a imitar las características perceptuales de la audición humana,

¿podríamos obtener información sobre los sistemas auditivos de otros animales entrenando modelos con sonidos característicos de su entorno? En esta dirección no hemos encontrado trabajos publicados, aunque sí existen trabajos que preentrenan modelos de audio con vocalizaciones de aves [274, 275] y de otros animales [276] para luego finetunear el modelo en tareas de bioacústica logrando mejoras respecto a utilizar modelos generales de audio.

Apéndice

A

Evaluación completa en HEAREval

A continuación se describen las 19 tareas de HEAREval, agrupándolas por dominio (música, habla o sonidos ambientales), y se muestran los resultados obtenidos por EnCodecMAE y otros modelos de la literatura.

A.1. Descripción de las tareas

Tareas de habla

- **Speech commands (SC)**: consiste en 10 comandos de voz, y las categorías 'Silencio' y 'Otros'. Se utiliza accuracy para su evaluación, y las divisiones oficiales en entrenamiento/validación/evaluación. Se evalúan dos versiones de este conjunto, uno con la totalidad de los datos, y otro con un subconjunto de 5 horas.
- **VoxLingua (LID)** : la tarea consiste en identificar el idioma hablado en un segmento de audio. Se seleccionaron los 10 idiomas más frecuentes del dataset VoxLingua107 [277]: árabe, danés, estonio, persa, finlandés, francés, armenio, letón, holandés y sueco. El conjunto resultante consiste de 972 segmentos de audio con una duración total de alrededor de 5 horas.
- **CREMA-D (ER)**: consiste en 12 oraciones dichas por 91 actores distintos con alguna de las siguientes emociones: enojo, asco, miedo, felicidad, neutro y triste; y distintas intensidades de emoción. La tarea consiste en clasificar la emoción a partir de la señal de habla. La métrica utilizada es accuracy y el conjunto de datos se organiza en 10 folds.
- **Libricount (Count)**: consiste en determinar la cantidad de hablantes distintos en un segmento de audio. Los segmentos son simulaciones de habla solapada con entre 0 y 10 hablantes y pertenecen al conjunto de datos Libricount [278]. Para realizar las simulaciones se utilizan audios provenientes del conjunto test-clean de LibriSpeech. Se reporta accuracy.
- **Vocal Imitation (VI)**: consiste en determinar el audio de un evento acústico que se está imitando vocalmente. El conjunto de datos presentado en [279] consiste de 5601 imitaciones vocales realizadas a partir de 302 audios de referencia, organizados según la ontología de Audioset. La tarea es multiclase y se deberá

Figura A.1: Mridangam e instrumentos musicales de percusión para ópera china.

predecir para una imitación vocal cual es el audio de referencia correspondiente. Se utiliza mean Average Precision (mAP) como métrica de evaluación.

Tareas de música

- **MAESTRO (AMT)**: la tarea consiste en realizar transcripción automática de piezas de piano. Se utiliza un subconjunto de 5 horas del dataset MAESTRO [280], y es una tarea de clasificación multi-etiqueta en el tiempo con 87 notas posibles. Se reporta el F1-score promedio tomando como nota válida si el onset se corresponde con el etiquetado con un margen de error de 50 ms.
- **NSynth (NS)**: consiste en clasificar notas musicales aisladas provenientes de distintos instrumentos musicales. Las clases corresponden a 88 notas (pitches) posibles. Se reporta el accuracy y utiliza el conjunto entero de datos, y un subconjunto de 5 horas.
- **Mridangam (Mri)**: el mridangam, que se puede observar en la Figura A.1, es un instrumento de percusión tonal utilizado en la música carnática india. Se realizan dos tareas a partir del conjunto de datos introducido en [281], y para ambas tareas se reporta el accuracy:
  - **Mri-S**: clasificar el tipo de golpe habiendo 10 clases posibles (bheem, cha, dheem, dhin, num, ta, tha, tham, thi y thom).
  - **Mri-T**: clasificar el tono habiendo 6 clases posibles.
- **Beijing Opera (BO)**: Se utiliza el conjunto de datos presentado en [282], el cual consiste en sonidos de instrumentos de percusión chinos. Se deben clasificar las instancias en 4 grupos de instrumentos de percusión mostrados en la Figura A.1: bangu, naobo, daluo y xiaoluo. Se reporta el accuracy.

Figura A.2: Ubicaciones de los micrófonos, tirador y objetivo utilizados en la tarea Gunshot [283].

- **GTZAN Genre (GC)**: se utiliza el conjunto GTZAN Genre Collection [32], el cual contiene 1000 canciones (segmentos de 30 segundos), categorizadas en 10 géneros musicales: blues, música clásica, country, disco, hiphop, jazz, metal, pop, reggae y rock. La tarea consiste en clasificar a qué género pertenece una canción. El conjunto de datos se organiza en 10 folds y se reporta accuracy.

Tareas de sonidos ambientales

- **Beehive**: es una tarea de clasificación binaria en la que hay que determinar si un panal de abejas contiene o no abeja reina. Se utiliza un conjunto de 576 grabaciones con una duración de 10 minutos cada una [284].
- **DCASE 2016**: es una tarea de detección de eventos de oficina en el tiempo, utilizando escenas sonoras sintéticas [285]. Los 11 eventos posibles son: aclarar la garganta, tos, cierre de puerta, sonido de cajones, sonido de teclas, sonido de llaves, golpe, risas, cambio de página, teléfono sonando y habla. Debido a problemas con el código, no evaluamos nuestros modelos con esta tarea.
- **Gunshot**: se cuenta con grabaciones de 22 disparos provenientes de 7 armas de fuego distintas y capturadas con 4 micrófonos ubicados en distintas posiciones [283] (ver Figura A.2). La tarea consiste en determinar qué micrófono se utilizó para capturar el disparo.
- **ESC-50** [99]: contiene 2000 grabaciones de 5 segundos en los que ocurre un único sonido, el cual pertenece a una de 50 categorías posibles. Algunos ejemplos de categorías son: lluvia, perro, bebe llorando, golpe de puerta y helicóptero. Se organiza en 5 folds y utiliza accuracy como métrica.

	Música							
	BO	GC	AMT	Mri-S	Mri-T	NS	NS-5h	M/S
(1) Mel→EC (Small)	95.8±2.3	87.5±1.9	47.7	**97.7±0.4**	**97.7±0.4**	85.9±1.2	73.0±4.6	**99.2±1.5**
(2) EC→EC (Base)	**97.0±1.9**	85.6±1.9	42.3	**97.5±0.4**	95.1±0.5	**91.7±1.0**	**86.0±3.1**	96.1±3.1
(3) Mel→EC (Base)	**96.2±2.3**	87.6±1.9	47.9	**97.6±0.3**	97.5±0.4	84.8±1.0	72.2±3.9	96.2±2.3
(4) AS only	**96.6±2.5**	87.1±1.9	**48.6**	**97.7±0.4**	**97.8±0.3**	86.1±1.1	76.8±3.7	**97.7±2.3**
(5) FMA only	**96.2±2.5**	**89.2±2.0**	47.9	**97.7±0.4**	**97.9±0.4**	87.7±1.1	77.6±3.4	97.6±2.4
(6) FMA+JAM	95.8±2.3	**89.5±1.8**	48.4	97.3±0.4	**97.6±0.4**	89.6±1.0	80.4±3.6	96.1±2.4
(7) LL6K only	**96.2±2.3**	83.4±2.0	47.5	97.4±0.4	96.0±0.5	83.4±1.3	70.4±4.5	92.9±3.8
(8) Mel→EC (Base + ST)	95.8±2.8	**87.8±2.1**	45.5	97.1±0.4	96.3±0.5	81.7±1.4	68.4±4.0	**99.2±1.7**
(9) AS only	**97.5±2.1**	87.7±2.3	43.3	97.1±0.4	95.8±0.5	83.5±1.3	71.2±3.8	97.6±2.3
(10) Mel→EC (Large)	**97.0±2.5**	85.8±2.2	47.6	**97.8±0.4**	**97.8±0.4**	85.3±1.4	73.2±4.1	**98.4±2.3**
(11) Mel→EC (Large + ST)	95.8±2.3	87.2±2.1	45.4	**97.5±0.4**	96.8±0.4	83.4±1.0	68.8±3.8	**98.5±1.9**
(12) AS only	95.3±2.1	86.4±1.7	45.8	97.2±0.4	96.9±0.4	86.4±1.3	75.6±3.7	**98.5±2.3**
(13) BEATS (Iter 1)[152]	95.3±2.7	85.4±2.4	3.1	96.2±0.4	95.9±0.4	82.4±1.3	68.4±3.8	96.9±3.5
(14) BEATS (Iter 3)[152]	94.1±3.0	86.2±2.9	3.1	94.7±0.5	95.9±0.5	82.9±1.0	69.6±4.6	94.4±4.8
(15) BYOL-A [138]	**97.9±1.9**	86.0±2.0	7.2	97.2±0.4	91.8±0.7	68.3±1.4	38.4±4.2	95.4±2.7
(16) MSM-MAE-512†	94.9	86.1	-	**97.5**	**98.3**	81.2	-	**99.2**
(17) Fuse HuBERT†	94.9	79.6	16.6	97.4	90.9	68.8	38.2	93.6
(18) Fuse Wav2Vec2†	94.5	79.3	11.1	96.2	83.8	60.6	33.0	95.3
HEAR SOTA	97.5	90.8‡	46.9‡	97.5	96.5	90.0‡	87.8‡	99.2‡

Tabla A.1: Evaluación completa en HEAREval para tareas musicales. Con † se muestran los resultados que fueron copiados del paper original, y con ‡ se muestran los resultados correspondientes a modelos preentrenados de forma supervisada.

- **FSD50K** [187]: contiene 100 horas de audios con anotaciones de eventos acústicos, los cuales pueden pertenecer a 200 categorías distintas. La tarea consiste en determinar qué eventos ocurrieron en un segmento de audio, y es multi-etiqueta. Se utiliza mean Average Precision (mAP) como métrica de evaluación.
- **GTZAN Music/Speech**: es una tarea de clasificación binaria en la que hay que determinar si un sonido es música o habla. El conjunto de datos contiene 60 segmentos de habla y 60 segmentos de música. Cada segmento tiene una duración de 30 segundos. Se reporta accuracy.

A.2. Resultados

Se puede observar en la Tabla A.1 que nuestros modelos poseen un desempeño similar o superior al estado del arte en HEAREval para todas las tareas de música, incluso superando a modelos que fueron preentrenados de forma supervisada en detección de eventos acústicos o regresión de frecuencia fundamental. Un ejemplo es el de transcripción automática de música en MAESTRO (AMT), en donde varios de nuestros modelos superan a modelos utilizando Crepe, que fue preentrenado en la tarea de regresión de frecuencia fundamental. En esta tarea, se puede observar la importancia de utilizar ventanas temporales en lugar de patches espectrotemporales, ya que los modelos de audio basados en patches rectangulares (filas 13,14,15 y 16) obtienen un desempeño muy bajo a comparación de EnCodecMAE, probablemente debido a la baja resolución temporal necesaria para detectar el comienzo y fin de las

notas musicales. En la tarea de clasificación de notas musicales (NS), nuestro modelo EC→EC (Base) (fila 2), supera a Crepe que es el estado del arte en HEAREval. Del mismo modo, varios de nuestros modelos superan el estado del arte de HEAREval en las tareas relacionadas a identificar el golpe y tono de un mridangam (Mri-S y Mri-T), y alcanzan un desempeño comparable al estado del arte en las tareas de clasificación de instrumentos de percusión (BO), clasificación de género musical (GC) y clasificación de música/habla (M/S).

	Habla						Ambiente			
	ER	Count	SC	SC-5h	VI	LID	Bee	ESC	FSD	Gun
(1) Mel→EC (Small)	74.8±0.9	76.9±1.0	95.2±0.7	93.3±0.8	15.7±2.9	50.5±3.2	51.0±8.1	80.1±1.7	48.5±1.2	**92.9±4.8**
(2) EC→EC (Base)	72.0±1.0	75.4±1.2	92.2±1.0	89.3±0.9	13.7±2.9	40.0±3.6	54.4±6.0	74.6±2.1	44.1±1.3	**91.7±6.0**
(3) Mel→EC (Base)	75.5±1.0	74.3±1.1	96.3±0.5	94.3±0.5	16.4±2.9	55.7±3.8	58.2±6.8	79.8±1.8	49.4±1.1	**90.5±4.8**
(4) AS only	73.8±1.1	77.2±1.2	93.6±0.7	90.2±0.7	17.0±3.1	48.4±3.4	8.1±2.6	81.5±2.1	51.7±1.3	89.3±6.0
(5) FMA only	68.7±1.1	74.7±1.0	87.4±0.9	82.1±1.0	17.4±3.1	34.1±2.7	15.4±5.6	80.4±2.2	48.5±1.3	**94.0±6.0**
(6) FMA+JAM	66.9±1.2	75.2±1.3	85.9±1.1	80.1±1.2	**19.0±3.2**	31.4±2.8	13.2±5.6	80.6±2.1	48.5±1.1	**91.1±7.1**
(7) LL6K only	75.6±1.1	74.3±1.3	95.4±0.6	93.8±0.7	13.4±2.8	51.9±3.3	36.5±7.5	68.8±1.6	41.3±1.2	**94.0±3.6**
(8) Mel→EC (Base + ST)	**76.3±1.0**	76.0±1.1	**96.9±0.5**	**95.2±0.5**	**19.1±3.3**	62.0±2.9	57.5±6.5	81.6±2.1	50.7±1.2	**91.7±6.6**
(9) AS only	75.1±1.1	77.0±1.1	94.6±0.6	92.6±0.8	**21.2±3.0**	53.8±2.7	37.9±7.2	**84.6±1.6**	**52.9±1.1**	87.2±4.0
(10) Mel→EC (Large)	**77.1±0.9**	76.6±1.2	96.4±0.5	94.3±0.6	15.4±2.9	52.0±3.2	50.7±7.2	80.2±1.9	50.3±1.1	82.1±6.1
(11) Mel→EC (Large + ST)	**77.2±0.9**	77.2±1.4	**97.0±0.4**	**95.5±0.5**	**18.7±3.2**	64.3±2.7	**79.8±5.7**	84.1±1.8	51.0±1.0	**94.0±5.4**
(12) AS only	75.1±1.1	77.0±1.1	94.6±0.6	92.6±0.8	**21.2±3.0**	53.8±2.7	37.9±7.2	**84.6±1.6**	**52.9±1.1**	87.2±4.0
(13) BEATS (Iter 1)[152]	68.7±1.1	73.5±1.2	91.0±0.8	86.5±1.0	13.5±2.9	46.0±2.9	30.3±6.1	82.3±1.7	50.6±0.9	**91.7±4.8**
(14) BEATS (Iter 3)[152]	69.0±1.1	**79.0±1.2**	90.4±0.8	87.6±1.0	13.7±2.9	43.1±3.2	8.4±4.2	**85.1±1.5**	**53.5±1.1**	89.3±8.3
(15) BYOL-A [138]	65.9±1.1	69.3±1.1	93.1±0.7	89.0±0.8	15.6±2.8	40.2±3.1	38.3±7.3	83.6±2.0	51.3±0.9	**92.0±4.8**
(17) MSM-MAE-512†	73.4	**77.8**	86.4	-	**18.3**	50.0	69.4	**85.6**	52.2	**96.4**
(18) Fuse HuBERT†	75.2	68.3	95.7	94.7	**18.5**	**71.4**	-	74.3	41.3	**92.9**
(19) Fuse Wav2Vec2†	69.2	65.3	**96.9**	**95.7**	17.4	**70.6**	-	69.5	40.3	**96.7**
HEAR SOTA	75.2	78.5	97.8‡	97.6‡	22.7‡	72.0‡	87.8	96.6‡	65.5‡	96.7

Tabla A.2: Evaluación completa en HEAREval para tareas de habla y sonidos ambiente. Con † se muestran los resultados que fueron copiados del paper original, y con ‡ se muestran los resultados correspondientes a modelos preentrenados de forma supervisada.

En la Tabla A.2 se puede observar que en la mayoría de las tareas de habla superamos a BEATs, BYOL-A y MSM-MAE-512, con excepción de la tarea de contado de hablantes (Count), en la que BEATs y MSM-MAE superan ligeramente a nuestros modelos (ver filas 14 y 16 vs 4 y 11). Particularmente, en esta tarea resulta interesante que en general los modelos entrenados solo con Audioset (filas 4, 9 y 12) parecen poseer una mejor capacidad de contar hablantes que los entrenados solo con habla (fila 7). Creemos que esto se debe a que Librilight contiene habla limpia y no solapada, a diferencia de Audioset, y por eso la distancia de dominio entre la tarea de pretexto y la downstream es mayor. A su vez, sonidos más diversos podrían exigir al modelo upstream a aprender a contar eventos para predecir los segmentos enmascarados. Si bien en la tarea de reconocimiento de comandos de voz (SC) superamos a los modelos de audio, los modelos específicos de habla (filas 18 y 19) resultan superiores, al igual que modelos de HEAR preentrenados para keyword spotting, los cuales alcanzan el estado del arte. Nuestros modelos con mejor desempeño en detección de eventos acústicos (filas 9 y 12) son también los que mejor desempeño alcanzan en reconocimiento de imitaciones vocales (VI), lo cual es razonable ya que las imitaciones vocales son de eventos acústicos de la ontología de

Audioset. Sin embargo, resulta notable que BEATs (fila 14), a pesar de alcanzar un desempeño similar en detección de eventos acústicos, no obtiene un buen desempeño en VI. Creemos que la resolución temporal puede tener un papel importante en esta tarea, ya que los imitadores están intentando recrear el sonido de referencia, no la clase semántica a la que pertenece, y deberán conservar la estructura temporal de la referencia. Por ejemplo, si el audio a imitar contiene 4 ladridos de perro, el imitador ladrará 4 veces con un timbre parecido al de la referencia, en lugar de realizar un ladrido genérico. En la tarea de identificación de idioma, los modelos específicos de habla (filas 17 y 18) alcanzan un desempeño superior al resto. Creemos que incorporar datos de habla en múltiples idiomas durante el preentrenamiento podría reducir esta brecha en el desempeño.

Respecto a las tareas relacionadas a sonidos ambiente, Beehive (Bee), consistente en reconocer si la abeja reina está presente en un panal, es la que mayor variabilidad presenta y es difícil determinar qué diseños favorecen o perjudican al desempeño en esta tarea. En trabajos anteriores [165,193] se ha notado este problema, probablemente ocasionado por la duración excesiva de las instancias (10 minutos) y la poca cantidad de muestras de entrenamiento. En cuanto a clasificación y detección de eventos acústicos (ESC y FSD), nuestros modelos preentrenados solo con Audioset (filas 9 y 12) alcanzan un desempeño comparable al de BEATs (fila 14) y MSM-MAE (fila 16), aunque todavía son inferiores a modelos preentrenados en Audioset de manera supervisada para la tarea de detección de eventos acústicos, los cuales alcanzan el estado del arte en HEAREval. Por último, para la tarea de triangulación de disparos (Gun), debido a la poca cantidad de muestras los intervalos de confianza son amplios, haciendo que la mayoría de los modelos obtengan un desempeño similar. Creemos que esta tarea, al igual que Beehive, son poco informativas de la calidad de las representaciones de audio.

B

Error de reconstrucción de decoders

Para estudiar si trae ventajas utilizar decoders con mayor capacidad en EnCodec-MAE, medimos su desempeño en la tarea de pretexto al utilizar distintas cantidades de capas. Utilizamos el conjunto de datos ESC-50, el cual no está contenido en el conjunto de preentrenamiento y consiste de sonidos de eventos acústicos. Se puede observar en la Figura B.1 que el desempeño es similar para los distintos tamaños de decoder en todos los cuantizadores, sugiriendo que no es necesario utilizar un decoder muy grande, dado el mismo encoder, para resolver la tarea de pretexto.

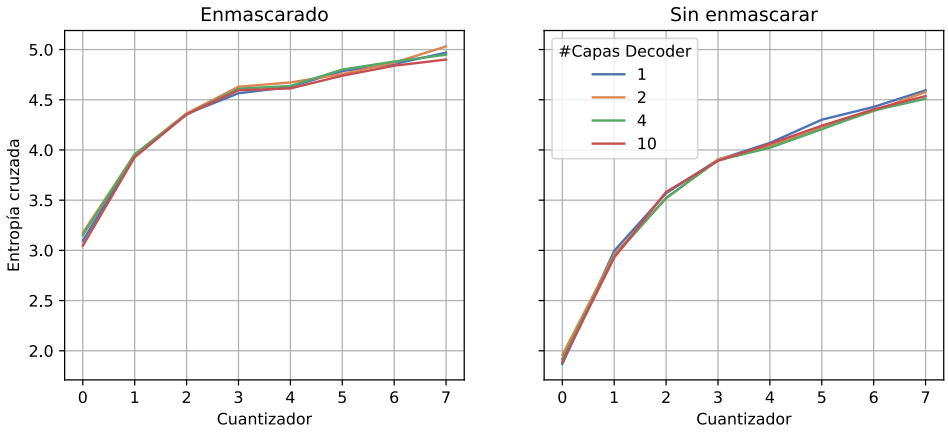

Figura B.1: Entropía cruzada por cuantizador obtenida en el conjunto de datos ESC-50 por versiones de Mel256→EC con distinta cantidad de capas en el decoder, tanto para segmentos enmascarados (Izquierda) como sin enmascarar (Derecha).

C

Corrección por atenuación

El uso del coeficiente de determinación como métrica de evaluación está limitado por la presencia de ruido en las mediciones. En Tuckute et al. se corrigió el ruido en las respuestas de cada voxel, lo cual es importante para poder realizar una comparación de los r^2 en distintos voxels. Por otro lado, el ruido también está presente en los datos de entrenamiento de cada modelo lineal. Por esto, las predicciones del modelo estarán afectadas por el ruido de forma distinta para cada voxel.

Para estimar el coeficiente de determinación si el ruido no estuviera presente, se utiliza el método de corrección por atenuación propuesto por Spearman [286], el cual estima la correlación entre dos variables independientemente del ruido de medición. Al realizar esta corrección, el máximo valor posible de r^2 es 1 en todos los voxels, en lugar de estar determinado por el ruido presente en cada voxel. El valor de r^2 corregido por atenuación teniendo en cuenta el efecto del ruido tanto en las mediciones como las predicciones está dado por:

$$r^2_{v,\hat{v}} = \frac{r^2}{r'_{\hat{v}} r'_v},$$

en donde, $r^2_{v,\hat{v}}$ es el valor de r^2 corregido, r^2 es el coeficiente de determinación sin corrección, el cual surge de la correlación de Pearson entre la respuesta del voxel v a los 82 estímulos del conjunto de evaluación, y las respuestas predichas por el modelo lineal. r'_v y $r'_{\hat{v}}$ se obtienen estimando la fiabilidad en cada voxel de las respuestas y predicciones respectivamente. Para calcular esta fiabilidad, se aprovecha que se realizaron múltiples sesiones y por ende se cuenta con varias mediciones de las respuestas y predicciones ante los 82 estímulos de evaluación para un mismo voxel y sujeto. Luego, r'_v se obtiene calculando la mediana de la correlación de las respuestas de un voxel y sujeto ante los 82 estímulos de evaluación para las distintas combinaciones de K sesiones. Se realiza el mismo procedimiento utilizando las respuestas predichas para obtener $r'_{\hat{v}}$. En ambos casos, se realiza una corrección de Spearman-Brown [287] para tener en cuenta el incremento de fiabilidad asociado a tener K veces más datos.

Intuitivamente, si las mediciones para un mismo voxel y sujeto en las distintas sesiones son consistentes entre sí, $r'_{\hat{v}} = 1$ y $r'_v = 1$, es decir no hay corrección y el valor de r^2 original se mantiene. Si hubiera ruido en las mediciones, el valor de $r'_{\hat{v}}$ y r'_v será menor a 1 trayendo un incremento en el valor de $r^2_{v,\hat{v}}$ respecto al r^2 medido. Esto podría resultar en un valor de $r^2_{v,\hat{v}} > 1$. Para evitar esto se determina una constante

k que sirva como valor mínimo del denominador. En este trabajo se utiliza un k que corresponde a la correlación entre dos distribuciones normales 83D para $r'_{\hat{v}}$ y 82D para r'_v con $p = 0.05$. Finalmente, los valores de $r^2_{v,\hat{v}}$ mayores a 1 se mapean a 1, mientras que los correspondientes a r negativos se mapean a 0.

Bibliografía

[1] A. Doerig, R. P. Sommers, K. Seeliger, B. Richards, J. Ismael, G. W. Lindsay, K. P. Kording, T. Konkle, M. A. Van Gerven, N. Kriegeskorte *et al.*, "The neuroconnectionist research programme," *Nature Reviews Neuroscience*, vol. 24, no. 7, pp. 431–450, 2023.

[2] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. u. Kaiser, and I. Polosukhin, "Attention is All you Need," in *Advances in Neural Information Processing Systems*, I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, Eds., vol. 30. Curran Associates, Inc., 2017. [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/2017/file3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf

[3] K. He, X. Chen, S. Xie, Y. Li, P. Dollár, and R. Girshick, "Masked autoencoders are scalable vision learners," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2022, pp. 16 000–16 009.

[4] J. Turian, J. Shier, H. R. Khan, B. Raj, B. W. Schuller, C. J. Steinmetz, C. Malloy, G. Tzanetakis, G. Velarde, K. McNally *et al.*, "Hear: Holistic evaluation of audio representations," in *NeurIPS 2021 Competitions and Demonstrations Track*. PMLR, 2022, pp. 125–145.

[5] S. wen Yang, P.-H. Chi, Y.-S. Chuang, C.-I. J. Lai, K. Lakhotia, Y. Y. Lin, A. T. Liu, J. Shi, X. Chang, G.-T. Lin, T.-H. Huang, W.-C. Tseng, K. tik Lee, D.-R. Liu, Z. Huang, S. Dong, S.-W. Li, S. Watanabe, A. Mohamed, and H. yi Lee, "SUPERB: Speech Processing Universal PERformance Benchmark," in *Proc. Interspeech 2021*, 2021, pp. 1194–1198.

[6] L. Pepino, P. Riera, L. Ferrer, and A. Gravano, "Fusion approaches for emotion recognition from speech using acoustic and text-based features," in *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2020, pp. 6484–6488.

[7] L. Pepino, P. Riera, and L. Ferrer, "Emotion Recognition from Speech Using wav2vec 2.0 Embeddings," in *Interspeech 2021*, 2021, pp. 3400–3404.

[8] G. Tuckute, J. Feather, D. Boebinger, and J. H. McDermott, "Many but not all deep neural network audio models capture brain responses and exhibit correspondence between model stages and brain regions," *Plos Biology*, vol. 21, no. 12, p. e3002366, 2023.

[9] L. L. Beranek, *Acoustics*. New York, NY: American Institute of Physics, Dec. 1986.

[10] L. E. Baum and T. Petrie, "Statistical inference for probabilistic functions of finite state Markov chains," *Annals of Mathematical Statistics*, vol. 37, pp. 1554–1563, 1966.

[11] J. L. Elman, "Finding structure in time," *Cognitive science*, vol. 14, no. 2, pp. 179–211, 1990.

[12] S. Hochreiter, "Long Short-term Memory," *Neural Computation MIT-Press*, 1997.

[13] J. B. Allen and L. R. Rabiner, "A unified approach to short-time Fourier analysis and synthesis," *Proceedings of the IEEE*, vol. 65, no. 11, pp. 1558–1564, 1977.

[14] J. C. Brown, "Calculation of a constant Q spectral transform," *The Journal of the Acoustical Society of America*, vol. 89, no. 1, pp. 425–434, 1991.

[15] M. Slaney, "Auditory toolbox," *Interval Research Corporation, Tech. Rep*, vol. 10, no. 1998, p. 1194, 1998.

[16] D. Griffin and J. Lim, "Signal estimation from modified short-time Fourier transform," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. 32, no. 2, pp. 236–243, 1984.

[17] J. Kong, J. Kim, and J. Bae, "Hifi-gan: Generative adversarial networks for efficient and high fidelity speech synthesis," *Advances in neural information processing systems*, vol. 33, pp. 17 022–17 033, 2020.

[18] G. A. Velasco, N. Holighaus, M. Dörfler, and T. Grill, "Constructing an invertible constant-Q transform with non-stationary Gabor frames," *Proceedings of DAFX11, Paris*, vol. 33, p. 81, 2011.

[19] R. Banse and K. R. Scherer, "Acoustic profiles in vocal emotion expression." *Journal of personality and social psychology*, vol. 70, no. 3, p. 614, 1996.

[20] F. Eyben, K. R. Scherer, B. W. Schuller, J. Sundberg, E. André, C. Busso, L. Y. Devillers, J. Epps, P. Laukka, S. S. Narayanan *et al.*, "The Geneva minimalistic acoustic parameter set (GeMAPS) for voice research and affective computing," *IEEE transactions on affective computing*, vol. 7, no. 2, pp. 190–202, 2015.

[21] F. Eyben, M. Wöllmer, and B. Schuller, "Opensmile: the munich versatile and fast open-source audio feature extractor," in *Proceedings of the 18th ACM international conference on Multimedia*, 2010, pp. 1459–1462.

[22] H. Fletcher and W. A. Munson, "Loudness, its definition, measurement and calculation," *Bell System Technical Journal*, vol. 12, no. 4, pp. 377–430, 1933.

[23] E. Zwicker and H. Fastl, *Psychoacoustics: Facts and models*. Springer Science & Business Media, 2013, vol. 22.

[24] A. De Cheveigné and H. Kawahara, "YIN, a fundamental frequency estimator for speech and music," *The Journal of the Acoustical Society of America*, vol. 111, no. 4, pp. 1917–1930, 2002.

[25] M. Mauch and S. Dixon, "pYIN: A fundamental frequency estimator using probabilistic threshold distributions," in *2014 ieee international conference on acoustics, speech and signal processing (icassp)*. IEEE, 2014, pp. 659–663.

[26] M. Morise *et al.*, "Harvest: A High-Performance Fundamental Frequency Estimator from Speech Signals." in *INTERSPEECH*, 2017, pp. 2321–2325.

[27] A. Camacho and J. G. Harris, "A sawtooth waveform inspired pitch estimator for speech and music," *The Journal of the Acoustical Society of America*, vol. 124, no. 3, pp. 1638–1652, 2008.

[28] J. W. Kim, J. Salamon, P. Li, and J. P. Bello, "Crepe: A convolutional representation for pitch estimation," in *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2018, pp. 161–165.

[29] B. Gfeller, C. Frank, D. Roblek, M. Sharifi, M. Tagliasacchi, and M. Velimirović, "SPICE: Self-supervised pitch estimation," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 28, pp. 1118–1128, 2020.

[30] S. Davis and P. Mermelstein, "Comparison of parametric representations for monosyllabic word recognition in continuously spoken sentences," *IEEE transactions on acoustics, speech, and signal processing*, vol. 28, no. 4, pp. 357–366, 1980.

[31] D. A. Reynolds and R. C. Rose, "Robust text-independent speaker identification using Gaussian mixture speaker models," *IEEE transactions on speech and audio processing*, vol. 3, no. 1, pp. 72–83, 1995.

[32] G. Tzanetakis and P. Cook, "Musical genre classification of audio signals," *IEEE Transactions on speech and audio processing*, vol. 10, no. 5, pp. 293–302, 2002.

[33] D. Barchiesi, D. Giannoulis, D. Stowell, and M. D. Plumbley, "Acoustic scene classification: Classifying environments from the sounds they produce," *IEEE Signal Processing Magazine*, vol. 32, no. 3, pp. 16–34, 2015.

[34] A. J. Eronen, V. T. Peltonen, J. T. Tuomi, A. P. Klapuri, S. Fagerlund, T. Sorsa, G. Lorho, and J. Huopaniemi, "Audio-based context recognition," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 14, no. 1, pp. 321–329, 2005.

[35] F. Eyben, *Real-time speech and music classification by large audio feature space extraction*. Springer, 2015.

[36] C. Cortes and V. Vapnik, "Support-vector networks," *Machine learning*, vol. 20, no. 3, pp. 273–297, 1995.

[37] L. Breiman, "Random forests," *Machine learning*, vol. 45, pp. 5–32, 2001.

[38] P. Best, S. Paris, H. Glotin, and R. Marxer, "Deep audio embeddings for vocalisation clustering," *Plos one*, vol. 18, no. 7, p. e0283396, 2023.

[39] A. Calonge, C. Parcerisas, E. Schall, and E. Debusschere, "Revised clusters of annotated unknown sounds in the Belgian part of the North Sea," *Frontiers in Remote Sensing*, vol. 5, p. 1384562, 2024.

[40] A. Hashem, M. Arif, and M. Alghamdi, "Speech emotion recognition approaches: A systematic review," *Speech Communication*, vol. 154, p. 102974, 2023. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0167639323001085

[41] M. Milling, F. B. Pokorny, K. D. Bartl-Pokorny, and B. W. Schuller, "Is speech the new blood? recent progress in ai-based disease detection from audio in a nutshell," *Frontiers in digital health*, vol. 4, p. 886615, 2022.

[42] H. Muralikrishna, P. Sapra, A. Jain, and D. A. Dinesh, "Spoken language identification using bidirectional lstm based lid sequential senones," in *2019 IEEE Automatic Speech Recognition and Understanding Workshop (ASRU)*. IEEE, 2019, pp. 320–326.

[43] J. Abeßer, "A review of deep learning based methods for acoustic scene classification," *Applied Sciences*, vol. 10, no. 6, p. 2020, 2020.

[44] N. Ndou, R. Ajoodha, and A. Jadhav, "Music genre classification: A review of deep-learning and traditional machine-learning approaches," in *2021 IEEE International IOT, Electronics and Mechatronics Conference (IEMTRONICS)*. IEEE, 2021, pp. 1–6.

[45] D. Vij, Y. Yogesh, D. Srivastava, and H. Shankar, "Detection of Acoustic Scenes and Events using Audio Analysis–A Survey," in *2023 3rd International Conference on Advance Computing and Innovative Technologies in Engineering (ICACITE)*. IEEE, 2023, pp. 316–320.

[46] J. Nam, K. Choi, J. Lee, S.-Y. Chou, and Y.-H. Yang, "Deep learning for audio-based music classification and tagging: Teaching computers to distinguish rock from bach," *IEEE signal processing magazine*, vol. 36, no. 1, pp. 41–51, 2018.

[47] H. Schreiber, J. Urbano, and M. Müller, "Music tempo estimation: Are we done yet?" *Trans. Int. Soc. Music. Inf. Retr.*, vol. 3, no. 1, p. 111, 2020.

[48] S. Parthasarathy and C. Busso, "Jointly Predicting Arousal, Valence and Dominance with Multi-Task Learning." in *Interspeech*, vol. 2017, 2017, pp. 1103–1107.

[49] Y. Korkmaz and A. Boyacı, "Hybrid voice activity detection system based on LSTM and auditory speech features," *Biomedical Signal Processing and Control*, vol. 80, p. 104408, 2023.

[50] B. Bhattarai and J. Lee, "A Comprehensive Review on Music Transcription," *Applied Sciences*, vol. 13, no. 21, p. 11882, 2023.

[51] J. Agrawal, M. Gupta, and H. Garg, "A review on speech separation in cocktail party environment: challenges and approaches," *Multimedia Tools and Applications*, vol. 82, no. 20, pp. 31 035–31 067, 2023.

[52] J. Richter, S. Welker, J.-M. Lemercier, B. Lay, and T. Gerkmann, "Speech enhancement and dereverberation with diffusion-based generative models," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 31, pp. 2351–2364, 2023.

[53] A. R. Bargum, S. Serafin, and C. Erkut, "Reimagining Speech: A Scoping Review of Deep Learning-Powered Voice Conversion," *arXiv preprint arXiv:2311.08104*, 2023.

[54] H. Kheddar, M. Hemis, and Y. Himeur, "Automatic speech recognition using advanced deep learning approaches: A survey," *Information Fusion*, p. 102422, 2024.

[55] X. Xu, Z. Xie, M. Wu, and K. Yu, "Beyond the status quo: A contemporary survey of advances and challenges in audio captioning," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 2023.

[56] X. Li, Y. Jia, and C.-C. Chiu, "Textless direct speech-to-speech translation with discrete speech representation," in *ICASSP 2023-2023 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2023, pp. 1–5.

[57] N. Kaur and P. Singh, "Conventional and contemporary approaches used in text to speech synthesis: A review," *Artificial Intelligence Review*, vol. 56, no. 7, pp. 5837–5880, 2023.

[58] L. Wang, Z. Zhao, H. Liu, J. Pang, Y. Qin, and Q. Wu, "A review of intelligent music generation systems," *Neural Computing and Applications*, vol. 36, no. 12, pp. 6381–6401, 2024.

[59] A. L. Samuel, "Some studies in machine learning using the game of checkers," *IBM Journal of research and development*, vol. 3, no. 3, pp. 210–229, 1959.

[60] *Machine learning*, vol. 1, no. 9.

[61] G. Cybenko, "Approximation by superpositions of a sigmoidal function," *Mathematics of control, signals and systems*, vol. 2, no. 4, pp. 303–314, 1989.

[62] C. Busso, M. Bulut, C.-C. Lee, A. Kazemzadeh, E. Mower, S. Kim, J. N. Chang, S. Lee, and S. S. Narayanan, "IEMOCAP: Interactive emotional dyadic motion capture database," *Language resources and evaluation*, vol. 42, pp. 335–359, 2008.

[63] D. P. Kingma, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[64] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," *Advances in neural information processing systems*, vol. 25, 2012.

[65] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "Imagenet: A large-scale hierarchical image database," in *2009 IEEE conference on computer vision and pattern recognition*. Ieee, 2009, pp. 248–255.

[66] S. Hershey, S. Chaudhuri, D. P. Ellis, J. F. Gemmeke, A. Jansen, R. C. Moore, M. Plakal, D. Platt, R. A. Saurous, B. Seybold *et al.*, "CNN architectures for large-scale audio classification," in *2017 ieee international conference on acoustics, speech and signal processing (icassp)*. IEEE, 2017, pp. 131–135.

[67] Y. Lee, S. Lim, and I.-Y. Kwak, "CNN-Based Acoustic Scene Classification System," *Electronics*, vol. 10, no. 4, 2021. [Online]. Available: https://www.mdpi.com/2079-9292/10/4/371

[68] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[69] O. Ronneberger, P. Fischer, and T. Brox, "U-net: Convolutional networks for biomedical image segmentation," in *Medical image computing and computer-assisted intervention–MICCAI 2015: 18th international conference, Munich, Germany, October 5-9, 2015, proceedings, part III 18*. Springer, 2015, pp. 234–241.

[70] J. Chen, S. Vekkot, and P. Shukla, "Music Source Separation Based on a Lightweight Deep Learning Framework (DTTNET: DUAL-PATH TFC-TDF UNET)," in *ICASSP 2024-2024 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2024, pp. 656–660.

[71] Y. Zhao and D. Wang, "Noisy-Reverberant Speech Enhancement Using DenseUNet with Time-Frequency Attention." in *Interspeech*, vol. 2020, 2020, pp. 3261–3265.

[72] P. M. Sørensen, B. Epp, and T. May, "A depthwise separable convolutional neural network for keyword spotting on an embedded system," *EURASIP Journal on Audio, Speech, and Music Processing*, vol. 2020, no. 1, p. 10, 2020.

[73] A. Van Den Oord, S. Dieleman, H. Zen, K. Simonyan, O. Vinyals, A. Graves, N. Kalchbrenner, A. Senior, K. Kavukcuoglu *et al.*, "Wavenet: A generative model for raw audio," *arXiv preprint arXiv:1609.03499*, vol. 12, 2016.

[74] Y. Luo and N. Mesgarani, "Conv-tasnet: Surpassing ideal time–frequency magnitude masking for speech separation," *IEEE/ACM transactions on audio, speech, and language processing*, vol. 27, no. 8, pp. 1256–1266, 2019.

[75] A. Baevski, Y. Zhou, A. Mohamed, and M. Auli, "wav2vec 2.0: A framework for self-supervised learning of speech representations," *Advances in neural information processing systems*, vol. 33, pp. 12 449–12 460, 2020.

[76] W.-N. Hsu, B. Bolte, Y.-H. H. Tsai, K. Lakhotia, R. Salakhutdinov, and A. Mohamed, "Hubert: Self-supervised speech representation learning by masked prediction of hidden units," *IEEE/ACM transactions on audio, speech, and language processing*, vol. 29, pp. 3451–3460, 2021.

[77] A. Radford, K. Narasimhan, T. Salimans, I. Sutskever *et al.*, "Improving language understanding by generative pre-training," 2018.

[78] J. Devlin, "Bert: Pre-training of deep bidirectional transformers for language understanding," *arXiv preprint arXiv:1810.04805*, 2018.

[79] J. Su, M. Ahmed, Y. Lu, S. Pan, W. Bo, and Y. Liu, "Roformer: Enhanced transformer with rotary position embedding," *Neurocomputing*, vol. 568, p. 127063, 2024.

[80] P. Shaw, J. Uszkoreit, and A. Vaswani, "Self-attention with relative position representations," *arXiv preprint arXiv:1803.02155*, 2018.

[81] C.-Z. A. Huang, A. Vaswani, J. Uszkoreit, N. Shazeer, I. Simon, C. Hawthorne, A. M. Dai, M. D. Hoffman, M. Dinculescu, and D. Eck, "Music transformer," *arXiv preprint arXiv:1809.04281*, 2018.

[82] O. Press, N. A. Smith, and M. Lewis, "Train short, test long: Attention with linear biases enables input length extrapolation," *arXiv preprint arXiv:2108.12409*, 2021.

[83] S. Xie, H. Zhang, J. Guo, X. Tan, J. Bian, H. H. Awadalla, A. Menezes, T. Qin, and R. Yan, "Residual: Transformer with dual residual connections," *arXiv preprint arXiv:2304.14802*, 2023.

[84] J. Lei Ba, J. R. Kiros, and G. E. Hinton, "Layer normalization," *ArXiv e-prints*, pp. arXiv–1607, 2016.

[85] A. Dosovitskiy, "An image is worth 16x16 words: Transformers for image recognition at scale," *arXiv preprint arXiv:2010.11929*, 2020.

[86] Y. Gong, Y.-A. Chung, and J. Glass, "Ast: Audio spectrogram transformer," *arXiv preprint arXiv:2104.01778*, 2021.

[87] Y. Bengio, A. Courville, and P. Vincent, "Representation learning: A review and new perspectives," *IEEE transactions on pattern analysis and machine intelligence*, vol. 35, no. 8, pp. 1798–1828, 2013.

[88] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press, 2016, http://www.deeplearningbook.org.

[89] K. Pearson, "LIII. On lines and planes of closest fit to systems of points in space," *The London, Edinburgh, and Dublin philosophical magazine and journal of science*, vol. 2, no. 11, pp. 559–572, 1901.

[90] D. D. Lee and H. S. Seung, "Learning the parts of objects by non-negative matrix factorization," *nature*, vol. 401, no. 6755, pp. 788–791, 1999.

[91] B. Ans, J. Hérault, and C. Jutten, "Architectures neuromimétiques adaptatives: Détection de primitives," *Proceedings of Cognitiva*, vol. 85, pp. 593–597, 1985.

[92] S. G. Mallat and Z. Zhang, "Matching pursuits with time-frequency dictionaries," *IEEE Transactions on signal processing*, vol. 41, no. 12, pp. 3397–3415, 1993.

[93] C. Olah, A. Mordvintsev, and L. Schubert, "Feature Visualization," *Distill*, 2017, https://distill.pub/2017/feature-visualization.

[94] P. Johannesma, "The pre-response stimulus ensemble of neurons in the cochlear nucleus," in *Symposium on Hearing Theory, 1972*. IPO, 1972.

[95] J. Howard and S. Ruder, "Universal language model fine-tuning for text classification," in *ACL 2018-56th Annual Meeting of the Association for Computational Linguistics, Proceedings of the Conference (Long Papers)*, vol. 1. Association for Computational Linguistics, 2018, pp. 328–339.

[96] E. J. Hu, Y. Shen, P. Wallis, Z. Allen-Zhu, Y. Li, S. Wang, L. Wang, and W. Chen, "LoRA: Low-Rank Adaptation of Large Language Models," in *International Conference on Learning Representations*, 2022. [Online]. Available: https://openreview.net/forum?id=nZeVKeeFYf9

[97] J. F. Gemmeke, D. P. W. Ellis, D. Freedman, A. Jansen, W. Lawrence, R. C. Moore, M. Plakal, and M. Ritter, "Audio Set: An ontology and human-labeled dataset for audio events," in *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2017, pp. 776–780.

[98] Q. Kong, Y. Cao, T. Iqbal, Y. Wang, W. Wang, and M. D. Plumbley, "Panns: Large-scale pretrained audio neural networks for audio pattern recognition," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 28, pp. 2880–2894, 2020.

[99] K. J. Piczak, "ESC: Dataset for Environmental Sound Classification," in *Proceedings of the 23rd Annual ACM Conference on Multimedia*. ACM Press, pp. 1015–1018. [Online]. Available: http://dl.acm.org/citation.cfm?doid=2733373.2806390

[100] P. Warden, "Speech Commands: A Dataset for Limited-Vocabulary Speech Recognition," *ArXiv e-prints*, Apr. 2018. [Online]. Available: https://arxiv.org/abs/1804.03209

[101] K. Koutini, J. Schlüter, H. Eghbal-zadeh, and G. Widmer, "Efficient Training of Audio Transformers with Patchout," in *Interspeech 2022, 23rd Annual Conference of the International Speech Communication Association, Incheon, Korea, 18-22 September 2022*. ISCA, 2022, pp. 2753–2757. [Online]. Available: https://doi.org/10.21437/Interspeech.2022-227

[102] P. Alonso-Jiménez, X. Serra, and D. Bogdanov, "Efficient supervised training of audio transformers for music representation learning," *arXiv preprint arXiv:2309.16418*, 2023.

[103] D. Snyder, D. Garcia-Romero, G. Sell, D. Povey, and S. Khudanpur, "X-vectors: Robust dnn embeddings for speaker recognition," in *2018 IEEE international conference on acoustics, speech and signal processing (ICASSP)*. IEEE, 2018, pp. 5329–5333.

[104] E. Variani, X. Lei, E. McDermott, I. L. Moreno, and J. Gonzalez-Dominguez, "Deep neural networks for small footprint text-dependent speaker verification," in *2014 IEEE international conference on acoustics, speech and signal processing (ICASSP)*. IEEE, 2014, pp. 4052–4056.

[105] Y. Lei, N. Scheffer, L. Ferrer, and M. McLaren, "A novel scheme for speaker recognition using a phonetically-aware deep neural network," in *2014 IEEE international conference on acoustics, speech and signal processing (ICASSP)*. IEEE, 2014, pp. 1695–1699.

[106] C. Li, X. Ma, B. Jiang, X. Li, X. Zhang, X. Liu, Y. Cao, A. Kannan, and Z. Zhu, "Deep speaker: an end-to-end neural speaker embedding system," *arXiv preprint arXiv:1705.02304*, 2017.

[107] Y. Blau and T. Michaeli, "The perception-distortion tradeoff," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 6228–6237.

[108] A. Défossez, J. Copet, G. Synnaeve, and Y. Adi, "High Fidelity Neural Audio Compression," *arXiv preprint arXiv:2210.13438*, 2022.

[109] Y.-A. Chung, C.-C. Wu, C.-H. Shen, H.-Y. Lee, and L.-S. Lee, "Audio word2vec: Unsupervised learning of audio segment representations using sequence-to-sequence autoencoder," *arXiv preprint arXiv:1603.00982*, 2016.

[110] J. T. Colonel, "Autoencoding Neural Networks as Musical Audio Synthesizers," Ph.D. dissertation, 2018, aAI13424678.

[111] J. T. Colonel, C. Curro, and S. Keene, "Improving Neural Net Auto Encoders for Music Synthesis," *Journal of The Audio Engineering Society*, 2017. [Online]. Available: https://api.semanticscholar.org/CorpusID:54765530

[112] F. Roche, T. Hueber, S. Limier, and L. Girin, "Autoencoders for music sound modeling: a comparison of linear, shallow, deep, recurrent and variational models," in *SMC 2019-16th Sound & Music Computing Conference*, no. 1-6, 2019.

[113] J. Chorowski, R. J. Weiss, S. Bengio, and A. Van Den Oord, "Unsupervised speech representation learning using wavenet autoencoders," *IEEE/ACM transactions on audio, speech, and language processing*, vol. 27, no. 12, pp. 2041–2053, 2019.

[114] J. Engel, C. Resnick, A. Roberts, S. Dieleman, M. Norouzi, D. Eck, and K. Simonyan, "Neural audio synthesis of musical notes with wavenet autoencoders," in *International Conference on Machine Learning*. PMLR, 2017, pp. 1068–1077.

[115] S. Dieleman, C. Nash, J. Engel, and K. Simonyan, "Variable-rate discrete representation learning," *arXiv preprint arXiv:2103.06089*, 2021.

[116] H. Lu, X. Wu, Z. Wu, and H. Meng, "SpeechTripleNet: End-to-End Disentangled Speech Representation Learning for Content, Timbre and Prosody," in *Proceedings of the 31st ACM International Conference on Multimedia*, 2023, pp. 2829–2837.

[117] W.-N. Hsu, Y. Zhang, and J. Glass, "Unsupervised learning of disentangled and interpretable representations from sequential data," *Advances in neural information processing systems*, vol. 30, 2017.

[118] P. Vincent, H. Larochelle, I. Lajoie, Y. Bengio, P.-A. Manzagol, and L. Bottou, "Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion." *Journal of machine learning research*, vol. 11, no. 12, 2010.

[119] A. Van Den Oord, O. Vinyals *et al.*, "Neural discrete representation learning," *Advances in neural information processing systems*, vol. 30, 2017.

[120] N. Zeghidour, A. Luebs, A. Omran, J. Skoglund, and M. Tagliasacchi, "Soundstream: An end-to-end neural audio codec," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 30, pp. 495–507, 2021.

[121] Z. Borsos, R. Marinier, D. Vincent, E. Kharitonov, O. Pietquin, M. Sharifi, D. Roblek, O. Teboul, D. Grangier, M. Tagliasacchi *et al.*, "Audiolm: a language modeling approach to audio generation," *IEEE/ACM transactions on audio, speech, and language processing*, vol. 31, pp. 2523–2533, 2023.

[122] J. Copet, F. Kreuk, I. Gat, T. Remez, D. Kant, G. Synnaeve, Y. Adi, and A. Défossez, "Simple and controllable music generation," *Advances in Neural Information Processing Systems*, vol. 36, pp. 47 704–47 720, 2023.

[123] C. Wang, S. Chen, Y. Wu, Z. Zhang, L. Zhou, S. Liu, Z. Chen, Y. Liu, H. Wang, J. Li *et al.*, "Neural codec language models are zero-shot text to speech synthesizers," *arXiv preprint arXiv:2301.02111*, 2023.

[124] S. Ji, Z. Jiang, W. Wang, Y. Chen, M. Fang, J. Zuo, Q. Yang, X. Cheng, Z. Wang, R. Li *et al.*, "Wavtokenizer: an efficient acoustic discrete codec tokenizer for audio language modeling," *arXiv preprint arXiv:2408.16532*, 2024.

[125] Y. Jiang, Q. Chen, S. Ji, Y. Xi, W. Wang, C. Zhang, X. Yue, S. Zhang, and H. Li, "UniCodec: Unified Audio Codec with Single Domain-Adaptive Codebook," *arXiv preprint arXiv:2502.20067*, 2025.

[126] J. D. Parker, A. Smirnov, J. Pons, C. Carr, Z. Zukowski, Z. Evans, and X. Liu, "Scaling transformers for low-bitrate high-quality speech coding," *arXiv preprint arXiv:2411.19842*, 2024.

[127] A. Défossez, L. Mazaré, M. Orsini, A. Royer, P. Pérez, H. Jégou, E. Grave, and N. Zeghidour, "Moshi: a speech-text foundation model for real-time dialogue," *arXiv preprint arXiv:2410.00037*, 2024.

[128] H. Siuzdak, F. Grötschla, and L. A. Lanzendörfer, "SNAC: Multi-Scale Neural Audio Codec," in *Audio Imagination: NeurIPS 2024 Workshop AI-Driven Speech, Music, and Sound Generation*.

[129] X. Zhang, D. Zhang, S. Li, Y. Zhou, and X. Qiu, "SpeechTokenizer: Unified Speech Tokenizer for Speech Language Models," in *ICLR*, 2024.

[130] S. Chen, C. Wang, Z. Chen, Y. Wu, S. Liu, Z. Chen, J. Li, N. Kanda, T. Yoshioka, X. Xiao *et al.*, "Wavlm: Large-scale self-supervised pre-training for full stack speech processing," *IEEE Journal of Selected Topics in Signal Processing*, vol. 16, no. 6, pp. 1505–1518, 2022.

[131] J. Shor, A. Jansen, R. Maor, O. Lang, O. Tuval, F. de Chaumont Quitry, M. Tagliasacchi, I. Shavitt, D. Emanuel, and Y. Haviv, "Towards Learning a Universal Non-Semantic Representation of Speech," 2020.

[132] A. Jansen, M. Plakal, R. Pandya, D. P. Ellis, S. Hershey, J. Liu, R. C. Moore, and R. A. Saurous, "Unsupervised learning of semantic audio representations," in *2018 IEEE international conference on acoustics, speech and signal processing (ICASSP)*. IEEE, 2018, pp. 126–130.

[133] A. v. d. Oord, Y. Li, and O. Vinyals, "Representation learning with contrastive predictive coding," *arXiv preprint arXiv:1807.03748*, 2018.

[134] S. Schneider, A. Baevski, R. Collobert, and M. Auli, "wav2vec: Unsupervised pre-training for speech recognition," *arXiv preprint arXiv:1904.05862*, 2019.

[135] Y.-A. Chung, W.-N. Hsu, H. Tang, and J. Glass, "An unsupervised autoregressive model for speech representation learning," *arXiv preprint arXiv:1904.03240*, 2019.

[136] Y.-A. Chung and J. Glass, "Generative pre-training for speech with autoregressive predictive coding," in *ICASSP*, 2020.

[137] Y.-A. Chung and J. Glass, "Improved Speech Representations with Multi-Target Autoregressive Predictive Coding," in *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, 2020, pp. 2353–2358.

[138] D. Niizumi, D. Takeuchi, Y. Ohishi, N. Harada, and K. Kashino, "Byol for audio: Self-supervised learning for general-purpose audio representation," in *2021 International Joint Conference on Neural Networks (IJCNN)*. IEEE, 2021, pp. 1–8.

[139] Z. Yang, M. Ding, T. Huang, Y. Cen, J. Song, B. Xu, Y. Dong, and J. Tang, "Does Negative Sampling Matter? a Review With Insights Into its Theory and Applications," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 46, no. 8, pp. 5692–5711, 2024.

[140] J.-B. Grill, F. Strub, F. Altché, C. Tallec, P. Richemond, E. Buchatskaya, C. Doersch, B. Avila Pires, Z. Guo, M. Gheshlaghi Azar *et al.*, "Bootstrap your own latent-a new approach to self-supervised learning," *Advances in neural information processing systems*, vol. 33, pp. 21 271–21 284, 2020.

[141] A. Baevski, W.-N. Hsu, Q. Xu, A. Babu, J. Gu, and M. Auli, "Data2vec: A general framework for self-supervised learning in speech, vision and language," in *International Conference on Machine Learning*. PMLR, 2022, pp. 1298–1312.

[142] A. Baevski, A. Babu, W.-N. Hsu, and M. Auli, "Efficient self-supervised learning with contextualized target representations for vision, speech and language," in *International Conference on Machine Learning*. PMLR, 2023, pp. 1416–1429.

[143] W. Chen, Y. Liang, Z. Ma, Z. Zheng, and X. Chen, "EAT: Self-supervised pre-training with efficient audio transformer," *arXiv preprint arXiv:2401.03497*, 2024.

[144] M. Caron, H. Touvron, I. Misra, H. Jégou, J. Mairal, P. Bojanowski, and A. Joulin, "Emerging properties in self-supervised vision transformers," in *Proceedings of the IEEE/CVF international conference on computer vision*, 2021, pp. 9650–9660.

[145] A. H. Liu, H.-J. Chang, M. Auli, W.-N. Hsu, and J. Glass, "Dinosr: Self-distillation and online clustering for self-supervised speech representation learning," *Advances in Neural Information Processing Systems*, vol. 36, pp. 58 346–58 362, 2023.

[146] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding," in *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, 2019, pp. 4171–4186.

[147] A. Baevski, S. Schneider, and M. Auli, "vq-wav2vec: Self-Supervised Learning of Discrete Speech Representations," in *International Conference on Learning Representations*.

[148] E. Jang, S. Gu, and B. Poole, "Categorical Reparametrization with Gumble-Softmax," in *International Conference on Learning Representations (ICLR 2017)*. OpenReview. net, 2017.

[149] A. Baevski, M. Auli, and A. Mohamed, "Effectiveness of self-supervised pre-training for speech recognition," *arXiv preprint arXiv:1911.03912*, 2019.

[150] Z. Chi, S. Huang, L. Dong, S. Ma, S. Singhal, P. Bajaj, X. Song, and F. Wei, "XLM-E: Cross-lingual Language Model Pre-training via ELECTRA," in *ACL 2022*, June 2021. [Online]. Available: https://www.microsoft.com/en-us/research/publication/xlm-e-cross-lingual-language-model-pre-training-via-electra/

[151] C.-C. Chiu, J. Qin, Y. Zhang, J. Yu, and Y. Wu, "Self-supervised learning with random-projection quantizer for speech recognition," in *International Conference on Machine Learning*. PMLR, 2022, pp. 3915–3924.

[152] S. Chen, Y. Wu, C. Wang, S. Liu, D. Tompkins, Z. Chen, W. Che, X. Yu, and F. Wei, "BEATs: audio pre-training with acoustic tokenizers," in *Proceedings of the 40th International Conference on Machine Learning*, 2023, pp. 5178–5193.

[153] S. Ling, Y. Liu, J. Salazar, and K. Kirchhoff, "Deep contextualized acoustic representations for semi-supervised speech recognition," in *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2020, pp. 6429–6433.

[154] S. Ling and Y. Liu, "Decoar 2.0: Deep contextualized acoustic representations with vector quantization," *arXiv preprint arXiv:2012.06659*, 2020.

[155] W. Wang, Q. Tang, and K. Livescu, "Unsupervised pre-training of bidirectional speech encoders via masked reconstruction," in *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2020, pp. 6889–6893.

[156] A. T. Liu, S.-w. Yang, P.-H. Chi, P.-c. Hsu, and H.-y. Lee, "Mockingjay: Unsupervised speech representation learning with deep bidirectional transformer encoders," in *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2020, pp. 6419–6423.

[157] P.-H. Chi, P.-H. Chung, T.-H. Wu, C.-C. Hsieh, Y.-H. Chen, S.-W. Li, and H.-y. Lee, "Audio albert: A lite bert for self-supervised learning of audio representation," in *2021 IEEE Spoken Language Technology Workshop (SLT)*. IEEE, 2021, pp. 344–350.

[158] L. Liu and Y. Huang, "Masked pre-trained encoder base on joint ctc-transformer," *arXiv preprint arXiv:2005.11978*, 2020.

[159] A. T. Liu, S.-W. Li, and H.-y. Lee, "Tera: Self-supervised learning of transformer encoder representation for speech," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 29, pp. 2351–2366, 2021.

[160] P.-Y. Huang, H. Xu, J. Li, A. Baevski, M. Auli, W. Galuba, F. Metze, and C. Feichtenhofer, "Masked autoencoders that listen," *Advances in Neural Information Processing Systems*, vol. 35, pp. 28 708–28 720, 2022.

[161] Y. Gong, C.-I. Lai, Y.-A. Chung, and J. Glass, "Ssast: Self-supervised audio spectrogram transformer," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 36, no. 10, 2022, pp. 10 699–10 709.

[162] D. Niizumi, D. Takeuchi, Y. Ohishi, N. Harada, and K. Kashino, "Masked spectrogram modeling using masked autoencoders for learning general-purpose audio representation," in *HEAR: Holistic Evaluation of Audio Representations*. PMLR, 2022, pp. 1–24.

[163] A. Baade, P. Peng, and D. Harwath, "MAE-AST: Masked Autoencoding Audio Spectrogram Transformer," in *Interspeech 2022*, 2022, pp. 2438–2442.

[164] D. Chong, H. Wang, P. Zhou, and Q. Zeng, "Masked Spectrogram Prediction for Self-Supervised Audio Pre-Training," in *ICASSP 2023 - 2023 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2023, pp. 1–5.

[165] H. Dinkel, Z. Yan, Y. Wang, J. Zhang, Y. Wang, and B. Wang, "Scaling up masked audio encoder learning for general audio classification," in *Interspeech 2024*, 2024.

[166] D. Jiang, X. Lei, W. Li, N. Luo, Y. Hu, W. Zou, and X. Li, "Improving transformer-based speech recognition using unsupervised pre-training," *arXiv preprint arXiv:1910.09932*, 2019.

[167] D. Jiang, W. Li, R. Zhang, M. Cao, N. Luo, Y. Han, W. Zou, K. Han, and X. Li, "A further study of unsupervised pretraining for transformer based speech recognition," in *ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2021, pp. 6538–6542.

[168] L. Smith and C. Yu, "Infants rapidly learn word-referent mappings via cross-situational statistics," *Cognition*, vol. 106, no. 3, pp. 1558–1568, 2008.

[169] G. K. Pullum and B. C. Scholz, "Empirical assessment of stimulus poverty arguments," *The linguistic review*, vol. 19, no. 1-2, pp. 9–50, 2002.

[170] N. Chomsky, *Language and problems of knowledge: The Managua lectures*. MIT press, 1987, vol. 16.

[171] Y. Aytar, C. Vondrick, and A. Torralba, "Soundnet: Learning sound representations from unlabeled video," *Advances in neural information processing systems*, vol. 29, 2016.

[172] R. Arandjelovic and A. Zisserman, "Look, listen and learn," in *Proceedings of the IEEE international conference on computer vision*, 2017, pp. 609–617.

[173] A. L. Cramer, H.-H. Wu, J. Salamon, and J. P. Bello, "Look, listen, and learn more: Design choices for deep audio embeddings," in *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2019, pp. 3852–3856.

[174] M. Hamilton, A. Zisserman, J. R. Hershey, and W. T. Freeman, "Separating theÇhirp"from theÇhat": Self-supervised Visual Grounding of Sound and Language," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2024, pp. 13 117–13 127.

[175] H. Zhang, F. Li, S. Liu, L. Zhang, H. Su, J. Zhu, L. Ni, and H.-Y. Shum, "DINO: DETR with Improved DeNoising Anchor Boxes for End-to-End Object Detection," in *The Eleventh International Conference on Learning Representations*.

[176] Y. Gong, A. Rouditchenko, A. H. Liu, D. Harwath, L. Karlinsky, H. Kuehne, and J. R. Glass, "Contrastive Audio-Visual Masked Autoencoder," in *The Eleventh International Conference on Learning Representations*.

[177] P. Peng and D. Harwath, "Self-supervised representation learning for speech using visual grounding and masked language modeling," *arXiv preprint arXiv:2202.03543*, 2022.

[178] D. Harwath, A. Recasens, D. Surís, G. Chuang, A. Torralba, and J. Glass, "Jointly discovering visual objects and spoken words from raw sensory input," in *Proceedings of the European conference on computer vision (ECCV)*, 2018, pp. 649–665.

[179] P. Peng and D. Harwath, "Word Discovery in Visually Grounded, Self-Supervised Speech Models," 2022.

[180] B. Elizalde, S. Deshmukh, M. Al Ismail, and H. Wang, "Clap learning audio concepts from natural language supervision," in *ICASSP 2023-2023 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2023, pp. 1–5.

[181] A. Radford, J. W. Kim, C. Hallacy, A. Ramesh, G. Goh, S. Agarwal, G. Sastry, A. Askell, P. Mishkin, J. Clark *et al.*, "Learning transferable visual models from natural language supervision," in *International conference on machine learning*. PMLR, 2021, pp. 8748–8763.

[182] Y.-J. Shih, H.-F. Wang, H.-J. Chang, L. Berry, H.-y. Lee, and D. Harwath, "Speechclip: Integrating speech with pre-trained vision and language model," in *2022 IEEE Spoken Language Technology Workshop (SLT)*. IEEE, 2023, pp. 715–722.

[183] R. Girdhar, A. El-Nouby, Z. Liu, M. Singh, K. V. Alwala, A. Joulin, and I. Misra, "Imagebind: One embedding space to bind them all," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2023, pp. 15 180–15 190.

[184] A. Guzhov, F. Raue, J. Hees, and A. Dengel, "Audioclip: Extending clip to image, text and audio," in *ICASSP 2022-2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2022, pp. 976–980.

[185] H. Dubey, V. Gopal, R. Cutler, S. Matusevych, S. Braun, E. S. Eskimez, M. Thakker, T. Yoshioka, H. Gamper, and R. Aichner, "ICASSP 2022 Deep Noise Suppression Challenge," in *ICASSP*, 2022.

[186] R. Ardila, M. Branson, K. Davis, M. Henretty, M. Kohler, J. Meyer, R. Morais, L. Saunders, F. M. Tyers, and G. Weber, "Common voice: A massively-multilingual speech corpus," *arXiv preprint arXiv:1912.06670*, 2019.

[187] E. Fonseca, X. Favory, J. Pons, F. Font, and X. Serra, "Fsd50k: an open dataset of human-labeled sound events," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 30, pp. 829–852, 2021.

[188] D. Bogdanov, M. Won, P. Tovstogan, A. Porter, and X. Serra, "The mtg-jamendo dataset for automatic music tagging," in *Machine learning for music discovery workshop, international conference on machine learning (ICML 2019)*, 2019, pp. 1–3.

[189] A. Wang, A. Singh, J. Michael, F. Hill, O. Levy, and S. R. Bowman, "GLUE: A Multi-Task Benchmark and Analysis Platform for Natural Language Understanding," 2019, in the Proceedings of ICLR.

[190] A. Wang, Y. Pruksachatkun, N. Nangia, A. Singh, J. Michael, F. Hill, O. Levy, and S. R. Bowman, "SuperGLUE: A Stickier Benchmark for General-Purpose Language Understanding Systems," *arXiv preprint 1905.00537*, 2019.

[191] N. Muennighoff, N. Tazi, L. Magne, and N. Reimers, "MTEB: Massive Text Embedding Benchmark," *arXiv preprint arXiv:2210.07316*, 2022.

[192] Y.-H. Wang, H.-Y. Chen, K.-W. Chang, W. Hsu, and H.-y. Lee, "Minisuperb: Lightweight benchmark for self-supervised speech models," in *2023 IEEE Automatic Speech Recognition and Understanding Workshop (ASRU)*. IEEE, 2023, pp. 1–8.

[193] J. Anton, H. Coppock, P. Shukla, and B. W. Schuller, "Audio Barlow Twins: Self-Supervised Audio Representation Learning," in *ICASSP 2023 - 2023 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2023, pp. 1–5.

[194] G. Elbanna, N. Scheidwasser-Clow, M. Kegler, P. Beckmann, K. El Hajal, and M. Cernak, "Byol-s: Learning self-supervised speech representations by bootstrapping," in *HEAR: Holistic Evaluation of Audio Representations*. PMLR, 2022, pp. 25–47.

[195] L. Wang, P. Luc, Y. Wu, A. Recasens, L. Smaira, A. Brock, A. Jaegle, J.-B. Alayrac, S. Dieleman, J. Carreira *et al.*, "Towards learning universal audio representations," in *ICASSP 2022-2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2022, pp. 4593–4597.

[196] P. Warden, "Speech commands: A dataset for limited-vocabulary speech recognition," *arXiv preprint arXiv:1804.03209*, 2018.

[197] H. Cao, D. G. Cooper, M. K. Keutmann, R. C. Gur, A. Nenkova, and R. Verma, "Crema-d: Crowd-sourced emotional multimodal actors dataset," *IEEE transactions on affective computing*, vol. 5, no. 4, pp. 377–390, 2014.

[198] K. J. Piczak, "ESC: Dataset for environmental sound classification," in *Proceedings of the 23rd ACM international conference on Multimedia*, 2015, pp. 1015–1018.

[199] J. Kahn, M. Rivière, W. Zheng, E. Kharitonov, Q. Xu, P. E. Mazaré, J. Karadayi, V. Liptchinsky, R. Collobert, C. Fuegen, T. Likhomanenko, G. Synnaeve, A. Joulin, A. Mohamed, and E. Dupoux, "Libri-Light: A Benchmark for ASR with Limited or No Supervision," in *ICASSP 2020 - 2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2020, pp. 7669–7673, https://github.com/facebookresearch/libri-light.

[200] M. Defferrard, K. Benzi, P. Vandergheynst, and X. Bresson, "FMA: A dataset for music analysis," *arXiv preprint arXiv:1612.01840*, 2016.

[201] M. Keller, S. Bengio, and S. Wong, "Benchmarking non-parametric statistical tests," *Advances in neural information processing systems*, vol. 18, 2005.

[202] L. Ferrer and P. Riera, "Confidence Intervals for evaluation in machine learning." [Online]. Available: https://github.com/luferrer/ConfidenceIntervals

[203] T.-Y. Wu, T.-Y. Hsu, C.-A. Li, T.-H. Lin, and H.-y. Lee, "The efficacy of self-supervised speech models for audio representations," in *HEAR: Holistic Evaluation of Audio Representations*. PMLR, 2022, pp. 90–110.

[204] A. Graves, S. Fernández, F. Gomez, and J. Schmidhuber, "Connectionist temporal classification: labelling unsegmented sequence data with recurrent neural networks," in *Proceedings of the 23rd international conference on Machine learning*, 2006, pp. 369–376.

[205] V. Panayotov, G. Chen, D. Povey, and S. Khudanpur, "Librispeech: an asr corpus based on public domain audio books," in *2015 IEEE international conference on acoustics, speech and signal processing (ICASSP)*. IEEE, 2015, pp. 5206–5210.

[206] M.-L. Zhang and Z.-H. Zhou, "ML-KNN: A lazy learning approach to multi-label learning," *Pattern recognition*, vol. 40, no. 7, pp. 2038–2048, 2007.

[207] M. E. Peters, M. Neumann, M. Iyyer, M. Gardner, C. Clark, K. Lee, and L. Zettlemoyer, "Deep Contextualized Word Representations," in *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, M. Walker, H. Ji, and A. Stent, Eds. New Orleans, Louisiana: Association for Computational Linguistics, Jun. 2018, pp. 2227–2237. [Online]. Available: https://aclanthology.org/N18-1202/

[208] J. Pons and X. Serra, "Randomly weighted cnns for (music) audio classification," in *ICASSP 2019-2019 IEEE international conference on acoustics, speech and signal processing (ICASSP)*. IEEE, 2019, pp. 336–340.

[209] A. M. Saxe, P. W. Koh, Z. Chen, M. Bhand, B. Suresh, and A. Y. Ng, "On random weights and unsupervised feature learning." in *Icml*, vol. 2, no. 3, 2011, p. 6.

[210] E. Amid, R. Anil, W. Kotłowski, and M. K. Warmuth, "Learning from randomly initialized neural network features," *arXiv preprint arXiv:2202.06438*, 2022.

[211] A. Pasad, J.-C. Chou, and K. Livescu, "Layer-wise analysis of a self-supervised speech representation model," in *2021 IEEE Automatic Speech Recognition and Understanding Workshop (ASRU)*. IEEE, 2021, pp. 914–921.

[212] Y. E. Kheir, Y. Samih, S. Maharjan, T. Polzehl, and S. Möller, "Comprehensive Layer-wise Analysis of SSL Models for Audio Deepfake Detection," *arXiv preprint arXiv:2502.03559*, 2025.

[213] J. Wagner, A. Triantafyllopoulos, H. Wierstorf, M. Schmitt, F. Burkhardt, F. Eyben, and B. W. Schuller, "Dawn of the transformer era in speech emotion recognition: closing the valence gap," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 45, no. 9, pp. 10 745–10 759, 2023.

[214] C. M. Bolaños, "Análisis de similaridad entre representaciones de palabras generadas por modelos de audio y de texto," 2024, director: Pablo Brusco, Codirector: Leonardo Pepino.

[215] Z. Tong, Y. Song, J. Wang, and L. Wang, "Videomae: Masked autoencoders are data-efficient learners for self-supervised video pre-training," *Advances in neural information processing systems*, vol. 35, pp. 10 078–10 093, 2022.

[216] Y. Bengio, J. Louradour, R. Collobert, and J. Weston, "Curriculum learning," in *Proceedings of the 26th annual international conference on machine learning*, 2009, pp. 41–48.

[217] T. Darcet, M. Oquab, J. Mairal, and P. Bojanowski, "Vision Transformers Need Registers," in *The Twelfth International Conference on Learning Representations*.

[218] S. Bethard, "We need to talk about random seeds," *arXiv preprint arXiv:2210.13393*, 2022.

[219] D. Picard, "Torch. manual_seed (3407) is all you need: On the influence of random seeds in deep learning architectures for computer vision," *arXiv preprint arXiv:2109.08203*, 2021.

[220] L. Schader, W. Song, R. Kempker, and D. Benkeser, "Don't let your analysis go to seed: on the impact of random seed on machine learning-based causal inference," *Epidemiology*, vol. 35, no. 6, pp. 764–778, 2024.

[221] P. S. Madhyastha and R. Jain, "On Model Stability as a Function of Random Seed," in *Proceedings of the 23rd Conference on Computational Natural Language Learning (CoNLL)*, 2019, pp. 929–939.

[222] A. C. Davison and D. V. Hinkley, *Bootstrap methods and their application*. Cambridge university press, 1997, no. 1.

[223] U. Güçlü, J. Thielen, M. Hanke, and M. Van Gerven, "Brains on beats," *Advances in Neural Information Processing Systems*, vol. 29, 2016.

[224] F. Khatami and M. A. Escabí, "Spiking network optimized for word recognition in noise predicts auditory system hierarchy," *PLOS Computational Biology*, vol. 16, no. 6, p. e1007558, 2020.

[225] J. Millet and J.-R. King, "Inductive biases, pretraining and fine-tuning jointly account for brain responses to speech," *arXiv preprint arXiv:2103.01032*, 2021.

[226] A. R. Vaidya, S. Jain, and A. Huth, "Self-Supervised Models of Audio Effectively Explain Human Cortical Responses to Speech," in *International Conference on Machine Learning*. PMLR, 2022, pp. 21 927–21 944.

[227] T. Chi, P. Ru, and S. A. Shamma, "Multiresolution spectrotemporal analysis of complex sounds," *The Journal of the Acoustical Society of America*, vol. 118, no. 2, pp. 887–906, 2005.

[228] M. Huh, B. Cheung, T. Wang, and P. Isola, "The platonic representation hypothesis," *arXiv preprint arXiv:2405.07987*, 2024.

[229] A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, I. Sutskever *et al.*, "Language models are unsupervised multitask learners," *OpenAI blog*, vol. 1, no. 8, p. 9, 2019.

[230] J. Feather, A. Durango, R. Gonzalez, and J. McDermott, "Metamers of neural networks reveal divergence from human perceptual systems," *Advances in Neural Information Processing Systems*, vol. 32, 2019.

[231] K. Drossos, S. Adavanne, and T. Virtanen, "Automated audio captioning with recurrent neural networks," in *2017 IEEE Workshop on Applications of Signal Processing to Audio and Acoustics (WASPAA)*. IEEE, 2017, pp. 374–378.

[232] D. Amodei, S. Ananthanarayanan, R. Anubhai, J. Bai, E. Battenberg, C. Case, J. Casper, B. Catanzaro, Q. Cheng, G. Chen *et al.*, "Deep speech 2: End-to-end speech recognition in english and mandarin," in *International conference on machine learning*. PMLR, 2016, pp. 173–182.

[233] Y. Tsao, P. Plantinga, M. Ravanelli, S.-W. Fu, C. Yu, T.-A. Hsieh, and X. Lu, "MetricGAN+: An Improved Version of MetricGAN for Speech Enhancement," *Interspeech 2021*, 2021.

[234] C. Wang, Y. Tang, X. Ma, A. Wu, D. Okhonko, and J. Pino, "Fairseq S2T: Fast Speech-to-Text Modeling with Fairseq," in *Proceedings of the 1st Conference of the Asia-Pacific Chapter of the Association for Computational Linguistics and the 10th International Joint Conference on Natural Language Processing: System Demonstrations*, 2020, pp. 33–39.

[235] C. Subakan, M. Ravanelli, S. Cornell, M. Bronzi, and J. Zhong, "Attention is all you need in speech separation," in *ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2021, pp. 21–25.

[236] B. van Niekerk, L. Nortje, and H. Kamper, "Vector-Quantized Neural Networks for Acoustic Unit Discovery in the ZeroSpeech 2020 Challenge," *Interspeech 2020*, 2020.

[237] R. M. Warren, "Perceptual restoration of missing speech sounds," *Science*, vol. 167, no. 3917, pp. 392–393, 1970.

[238] D. M. Groppe, M. Choi, T. Huang, J. Schilz, B. Topkins, T. P. Urbach, and M. Kutas, "The phonemic restoration effect reveals pre-N400 effect of supportive sentence context in speech perception," *Brain research*, vol. 1361, pp. 54–66, 2010.

[239] P. Sivonen, B. Maess, S. Lattner, and A. D. Friederici, "Phonemic restoration in a sentence context: evidence from early and late ERP effects," *Brain research*, vol. 1121, no. 1, pp. 177–189, 2006.

[240] K. L. Aw, S. Montariol, B. AlKhamissi, M. Schrimpf, and A. Bosselut, "Instruction-tuning Aligns LLMs to the Human Brain," in *First Conference on Language Modeling*.

[241] D. Hendrycks, C. Burns, S. Basart, A. Zou, M. Mazeika, D. Song, and J. Steinhardt, "Measuring Massive Multitask Language Understanding." in *ICLR*. OpenReview.net, 2021. [Online]. Available: http://dblp.uni-trier.de/db/conf/iclr/iclr2021.html#HendrycksBBZMSS21

[242] S. Norman-Haignere, N. G. Kanwisher, and J. H. McDermott, "Distinct cortical pathways for music and speech revealed by hypothesis-free voxel decomposition," *neuron*, vol. 88, no. 6, pp. 1281–1296, 2015.

[243] D. Boebinger, S. V. Norman-Haignere, J. H. McDermott, and N. Kanwisher, "Music-selective neural populations arise without musical training," *Journal of Neurophysiology*, vol. 125, no. 6, pp. 2237–2263, 2021.

[244] S. Lee, J. Chung, Y. Yu, G. Kim, T. Breuel, G. Chechik, and Y. Song, "Acav100m: Automatic curation of large-scale datasets for audio-visual video representation learning," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021, pp. 10 274–10 284.

[245] M. F. Glasser, T. S. Coalson, E. C. Robinson, C. D. Hacker, J. Harwell, E. Yacoub, K. Ugurbil, J. Andersson, C. F. Beckmann, M. Jenkinson *et al.*, "A multi-modal parcellation of human cerebral cortex," *Nature*, vol. 536, no. 7615, pp. 171–178, 2016.

[246] J. Pennington, R. Socher, and C. D. Manning, "Glove: Global vectors for word representation," in *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, 2014, pp. 1532–1543.

[247] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean, "Distributed representations of words and phrases and their compositionality," *Advances in neural information processing systems*, vol. 26, 2013.

[248] L. Martinez-Lucas, M. Abdelwahab, and C. Busso, "The MSP-conversation corpus," *Interspeech 2020*, 2020.

[249] J. Boigne, B. Liyanage, and T. Östrem, "Recognizing more emotions with less data using self-supervised transfer learning," *arXiv preprint arXiv:2011.05585*, 2020.

[250] Z. Fan, M. Li, S. Zhou, and B. Xu, "Exploring wav2vec 2.0 on Speaker Verification and Language Identification," *Interspeech 2021*, 2021.

[251] C. Wang, A. Wu, J. Pino, A. Baevski, M. Auli, and A. Conneau, "Large-Scale Self- and Semi-Supervised Learning for Speech Translation," in *Interspeech 2021*, 2021, pp. 2242–2246.

[252] S. R. Livingstone and F. A. Russo, "The Ryerson Audio-Visual Database of Emotional Speech and Song (RAVDESS): A dynamic, multimodal set of facial and vocal expressions in North American English," *PloS one*, vol. 13, no. 5, p. e0196391, 2018.

[253] K. Venkataramanan and H. R. Rajamohan, "Emotion recognition from speech," *arXiv preprint arXiv:1912.10458*, 2019.

[254] S. Mirsamadi, E. Barsoum, and C. Zhang, "Automatic speech emotion recognition using recurrent neural networks with local attention," in *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2017.

[255] A. Satt, S. Rozenberg, and R. Hoory, "Efficient Emotion Recognition from Speech Using Deep Learning on Spectrograms." in *Interspeech*, 2017.

[256] M. Sarma, P. Ghahremani, D. Povey, N. K. Goel, K. K. Sarma, and N. Dehak, "Emotion Identification from Raw Speech Signals Using DNNs." in *Interspeech*, 2018.

[257] L. Pepino, P. Riera, and L. Ferrer, "Study of positional encoding approaches for audio spectrogram transformers," in *ICASSP 2022-2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2022, pp. 3713–3717.

[258] O. Press, N. Smith, and M. Lewis, "Train Short, Test Long: Attention with Linear Biases Enables Input Length Extrapolation," in *International Conference on Learning Representations*, 2022.

[259] X. Chu, Z. Tian, B. Zhang, X. Wang, and C. Shen, "Conditional Positional Encodings for Vision Transformers," in *The Eleventh International Conference on Learning Representations*, 2021.

[260] Y. Gong, Y.-A. Chung, and J. Glass, "AST: Audio Spectrogram Transformer," in *Proc. Interspeech 2021*, 2021, pp. 571–575.

[261] A. Kumar and V. Ithapu, "A sequential self teaching approach for improving generalization in sound event recognition," in *ICML*, 2020.

[262] J. Kim, "Urban sound tagging using multi-channel audio feature with convolutional neural networks," *Proceedings of the Detection and Classification of Acoustic Scenes and Events*, 2020.

[263] P. Alonso-Jiménez, L. Pepino, R. Batlle-Roca, P. Zinemanas, D. Bogdanov, X. Serra, and M. Rocamora, "Leveraging pre-trained autoencoders for interpretable prototype learning of music audio," in *2024 IEEE International Conference on Acoustics, Speech, and Signal Processing Workshops (ICASSPW)*. IEEE, 2024, pp. 833–837.

[264] Y. Tay, M. Dehghani, D. Bahri, and D. Metzler, "Efficient transformers: A survey," *ACM Computing Surveys*, vol. 55, no. 6, pp. 1–28, 2022.

[265] Y. Chen, X. Dai, M. Liu, D. Chen, L. Yuan, and Z. Liu, "Dynamic convolution: Attention over convolution kernels," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2020, pp. 11 030–11 039.

[266] A. Gu and T. Dao, "Mamba: Linear-time sequence modeling with selective state spaces," *arXiv preprint arXiv:2312.00752*, 2023.

[267] M. Beck, K. Pöppel, M. Spanring, A. Auer, O. Prudnikova, M. Kopp, G. Klambauer, J. Brandstetter, and S. Hochreiter, "xLSTM: Extended Long Short-Term Memory," *Advances in Neural Information Processing Systems*, vol. 37, pp. 107 547–107 603, 2025.

[268] F. Schmid, K. Koutini, and G. Widmer, "Dynamic convolutional neural networks as efficient pre-trained audio models," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 2024.

[269] H. Nam, S. Kim, and Y.-H. Park, "Frequency Dynamic Convolution: Frequency-Adaptive Pattern Recognition for Sound Event Detection," in *23rd Annual Conference of the International Speech Communication Association, INTERSPEECH 2022*. ISCA, 2022, pp. 2763–2767.

[270] P. Nawrot, S. Tworkowski, M. Tyrolski, Ł. Kaiser, Y. Wu, C. Szegedy, and H. Michalewski, "Hierarchical Transformers Are More Efficient Language Models," in *Findings of the Association for Computational Linguistics: NAACL 2022*, 2022, pp. 1559–1571.

[271] K. Chen, X. Du, B. Zhu, Z. Ma, T. Berg-Kirkpatrick, and S. Dubnov, "HTS-AT: A hierarchical token-semantic audio transformer for sound classification and detection," in *ICASSP 2022-2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2022, pp. 646–650.

[272] O. Moussa, D. Klakow, and M. Toneva, "Improving semantic understanding in speech language models via brain-tuning," *arXiv preprint arXiv:2410.09230*, 2024.

[273] M. Freteault, M. Le Clei, L. Tetrel, L. Bellec, and N. Farrugia, "Alignment of auditory artificial networks with massive individual fMRI brain data leads to generalisable improvements in brain encoding and downstream tasks," *Imaging Neuroscience*, vol. 3, p. imag_a_00525, 2025.

[274] L. Rauch, I. Moummad, R. Heinrich, A. Joly, B. Sick, and C. Scholz, "Can Masked Autoencoders Also Listen to Birds?" *arXiv preprint arXiv:2504.12880*, 2025.

[275] I. Moummad, R. Serizel, E. Benetos, and N. Farrugia, "Domain-Invariant Representation Learning of Bird Sounds," *arXiv preprint arXiv:2409.08589*, 2024.

[276] M. Hagiwara, "AVES: Animal vocalization encoder based on self-supervision," in *ICASSP 2023-2023 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2023, pp. 1–5.

[277] J. Valk and T. Alumäe, "VoxLingua107: a dataset for spoken language recognition," in *2021 IEEE Spoken Language Technology Workshop (SLT)*. IEEE, 2021, pp. 652–658.

[278] F.-R. Stöter, S. Chakrabarty, E. Habets, and B. Edler, "LibriCount, a dataset for speaker count estimation," Apr. 2018. [Online]. Available: https://doi.org/10.5281/zenodo.1216072

[279] B. Kim, M. Ghei, B. Pardo, and Z. Duan, "Vocal Imitation Set: a dataset of vocally imitated sound events using the AudioSet ontology." in *DCASE*, 2018, pp. 148–152.

[280] C. Hawthorne, A. Stasyuk, A. Roberts, I. Simon, C.-Z. A. Huang, S. Dieleman, E. Elsen, J. Engel, and D. Eck, "Enabling Factorized Piano Music Modeling and Generation with the MAESTRO Dataset," in *International Conference on Learning Representations*.

[281] A. Anantapadmanabhan, A. Bellur, and H. A. Murthy, "Modal analysis and transcription of strokes of the mridangam using non-negative matrix factorization," in *2013 IEEE international conference on acoustics, speech and signal processing*. IEEE, 2013, pp. 181–185.

[282] M. Tian, A. Srinivasamurthy, M. Sandler, and X. Serra, "A study of instrument-wise onset detection in beijing opera percussion ensembles," in *2014 ieee international conference on acoustics, speech and signal processing (icassp)*. IEEE, 2014, pp. 2159–2163.

[283] S. Cooper and S. Shaw, "Gunshots recorded in an open field using ipod touch devices," *Dryad, Dataset*, p. 43, 2020.

[284] I. Nolasco, A. Terenzi, S. Cecchi, S. Orcioni, H. L. Bear, and E. Benetos, "Audio-based identification of beehive states," in *ICASSP 2019-2019 IEEE International conference on acoustics, speech and signal processing (ICASSP)*. IEEE, 2019, pp. 8256–8260.

[285] A. Mesaros, T. Heittola, E. Benetos, P. Foster, M. Lagrange, T. Virtanen, and M. D. Plumbley, "Detection and classification of acoustic scenes and events: Outcome of the DCASE 2016 challenge," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 26, no. 2, pp. 379–393, 2017.

[286] C. Spearman, "The proof and measurement of association between two things." 1961.

[287] C. Spearman, "Correlation calculated from faulty data," *British journal of psychology*, vol. 3, no. 3, p. 271, 1910.