Pontificia Universidad JAVERIANA Cali | Facultad de Ingeniería y Ciencias

Acta de Correcciones al Proyecto de Grado Ingeniería de Sistemas y Computación

Fecha: 22 de Julio del 2024

Autores: Jeremías Villalobos Tenorio

Nombre del Proyecto de Grado: Clasificador de sonidos que indiquen una alerta o amenaza para las personas con discapacidad auditiva

Director: Dr. Julián Gil González

Como indica el artículo 2.27 de las Directrices de Trabajo de Grado, he verificado que los estudiantes indicados arriba han implementado todas las correcciones que los Jurados del Proyecto de Grado definieron que se efectuarán, como consta en el Acta de Calificación correspondiente.

Firma de Director(a) del Proyecto de Grado

Nota de Aceptación

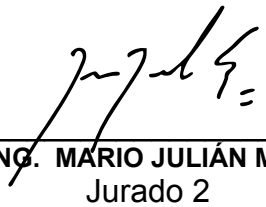
Aprobado por el Comité de Trabajo de Grado en cumplimiento de los requisitos exigidos por la Pontificia Universidad Javeriana para optar el título de Ingeniero de Sistemas y Computación.

Dr. CAMILO ROCHA
Decano de la Facultad de Ingeniería

ING. GERARDO MAURICIO SARRIA
Director Carrera Ingeniería Sistemas y Computación.

ING. JULIÁN GIL GONZÁLEZ
Director(a) Trabajo

ING. GERARDO MAURICIO SARRIA
Jurado 1



ING. MARIO JULIÁN MORA
Jurado 2

Pontificia Universidad
JAVERIANA
Cali

Universidad Javeriana Cali

Ingeniería de Sistemas y Computación

Trabajo de Grado

Clasificador de sonidos que indiquen una alerta o amenaza para las personas con discapacidad auditiva

Autor

Jeremías Villalobos Tenorio

Director

Dr. Julián Gil González

2024

Santiago de Cali, 24 de Junio del 2024.

Señores
Pontificia Universidad Javeriana Cali.
Dr. Gerardo Mauricio Sarria Montemiranda
Director Carrera de Ingeniería de Sistemas y Computación.
Cali.

Cordial Saludo.

Por medio de la presente me permito informarle que el estudiante de Ingeniería de Sistemas y Computación Jeremías Villalobos Tenorio (cod: 8939709) trabajó bajo mi dirección en el proyecto de grado titulado "Clasificador de sonidos que indiquen una alerta o amenaza para las personas con discapacidad auditiva".

Atentamente,

Dr. Julián Gil González

Santiago de Cali, 24 de Junio del 2024.

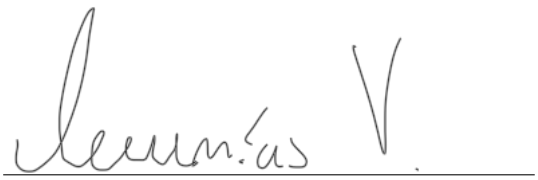
Señores
Pontificia Universidad Javeriana Cali.
Dr. Gerardo Mauricio Sarria Montemiranda
Director de Carrera de Ingeniería de Sistemas y Computación.
Cali.

Cordial Saludo.

Me permito presentar a su consideración el proyecto de grado titulado "Clasificador de sonidos que indiquen una alerta o amenaza para las personas con discapacidad auditiva" con el fin de cumplir con los requisitos exigidos por la Universidad para optar al título de Ingeniero de Sistemas y Computación.

Al firmar aquí, doy fe que entiendo y conozco las directrices para la presentación de trabajos de grado de la Facultad de Ingeniería aprobadas el 26 de Noviembre de 2009, donde se establecen los plazos y normas para el desarrollo del anteproyecto y del trabajo de grado.

Atentamente,



Jeremías Villalobos Tenorio
Código: 8939709

Resumen

Este trabajo de grado se enfoca en el entrenamiento de modelos de aprendizaje automático para clasificar algunos sonidos que se encuentran en el conjunto de datos AudioSet de Google. Estos sonidos fueron seleccionados en función de la cantidad de muestras disponibles y su relevancia para indicar una alerta o amenaza.

A través de este proyecto, se quiere documentar el proceso para llegar a entrenar un modelo que cumpla la tarea de clasificación de sonidos, y mostrar los obstáculos que se pueden presentar para lograrlo. También se busca dejar las puertas abiertas para un trabajo futuro donde se implemente un modelo de este tipo en dispositivos móviles con micrófono, y se logre ayudar a las personas con discapacidad auditiva a aprender a asociar lo que escuchan con su significado, o a que puedan identificar sonidos de su entorno físico que indiquen una alerta o amenaza para su integridad.

Para llegar a los resultados del proyecto, fue necesario generar espectrogramas a partir de los sonidos descargados y entrenar varios modelos con ayuda de *transfer learning*. En los resultados se presenta una comparación entre los modelos entrenados, su evaluación con distintas métricas de desempeño, y su comparación con algunos modelos del estado del arte.

Palabras clave: Aprendizaje automático, discapacidad auditiva, transfer learning, clasificación de sonidos, AudioSet, espectrogramas.

Abstract

This thesis focuses on training machine learning models to classify certain sounds found in Google's AudioSet dataset. These sounds were selected based on the number of available samples and their relevance for indicating an alert or threat.

Through this project, we aim to document the process of training a model to perform sound classification and highlight the obstacles that may arise in achieving this goal. Additionally, it aims to set the foundation for future work where such a model could be implemented in mobile devices with microphones, assisting people with hearing disabilities in learning to associate sounds with their meanings, or in identifying sounds in their physical environment that indicate an alert or threat to their safety.

To achieve the project results, it was necessary to generate spectrograms from the downloaded sounds and train several models using transfer learning. The results present a comparison between the trained models, their evaluation using various performance metrics, and their comparison with some state-of-the-art models.

Keywords: Machine learning, hearing disability, transfer learning, sound classification, AudioSet, spectrograms.

Índice general

1. Introducción **9**

2. Descripción del problema **11**

2.1. Planteamiento del Problema 11

2.1.1. Formulación 13

2.1.2. Sistematización 13

2.2. Objetivos 14

2.2.1. Objetivo General 14

2.2.2. Objetivos Específicos 14

2.3. Justificación 15

3. Marco Teórico **17**

3.1. Pérdida auditiva 17

3.1.1. Definición 17

3.1.2. Tipos de pérdida auditiva 17

3.1.3. Impacto 19

3.2. Espectrogramas 19

3.2.1. Espectrogramas en escala de Mel 20

3.3. Inteligencia Artificial 21

3.4. Aprendizaje Automático 21

3.5. Aprendizaje Profundo y redes neuronales 22

3.5.1. Función de Activación ReLU 24

3.5.2. Regularización L2 24

3.5.3. Regularización Dropout 24

3.5.4. Función de activación Softmax 25

3.5.5. Función de costo Categorical Cross Entropy 25

3.6. Redes neuronales convolucionales 25

3.7. Transfer Learning 26

3.7.1. VGG16 27

3.7.2. EfficientNetB7 27

3.7.3. EfficientNetV2M 27

3.7.4. MobileNetV3L 27
3.7.5. ConvNeXtBase 28
3.8. Hiperparámetros y su búsqueda 28
3.8.1. Random Search 28

4. Metodología **30**
4.1. Obtención del conjunto de datos 32
4.2. Generación de espectrogramas 34
4.2.1. Muestras de espectrogramas por clase 36
4.3. Extracción de características de los espectrogramas 40
4.4. Entrenamiento de los modelos 42
4.5. Optimización de los modelos 43
4.5.1. Optimización general 44
4.5.2. Optimización de los mejores modelos 45
4.6. Validación de los modelos 45

5. Resultados **48**
5.1. Primera fase de optimización 48
5.2. Segunda fase de optimización 51
5.3. Métricas de desempeño de los mejores modelos 53
5.3.1. Modelo entrenado con características de EfficientNetB7 . . . 54
5.3.2. Modelo entrenado con características de MobileNetV3L . . . 56
5.3.3. Modelo entrenado con características de ConvNeXtBase . . . 58
5.4. Comparación con el estado del arte 60

6. Conclusiones **61**
6.1. Trabajo futuro 62

7. Bibliografía y referencias **63**

Índice de figuras

1. Espectrograma de ejemplo [17] 20
2. Espectrograma Mel de ejemplo [17] 21
3. Diferencia entre programación clásica y aprendizaje automático. . . 22
4. Red neuronal básica. 23
5. Funcionamiento del aprendizaje profundo. 24
6. Comparación entre redes neuronales y redes neuronales convolucionales. 25
7. Arquitectura completa para el desarrollo de la metodología 31
8. Funcionamiento de la herramienta *audioset-processing* [27]. 33
9. Espectrograma mel de la muestra «Emergency Vehicle». 35
10. Espectrograma estándar de la muestra «Emergency Vehicle». 36
11. Muestra de espectrograma mel para la clase «Emergency Vehicle». . 37
12. Muestra de espectrograma mel para la clase «Explosion». 37
13. Muestra de espectrograma mel para la clase «Gunshot». 37
14. Muestra de espectrograma mel para la clase «Power Tool». 38
15. Muestra de espectrograma mel para la clase «Music». 38
16. Muestra de espectrograma mel para la clase «Bell». 38
17. Muestra de espectrograma mel para la clase «Speech». 39
18. Muestra de espectrograma mel para la clase «Silence». 39
19. Muestra de espectrograma mel para la clase «Sneeze». 39
20. Ejemplo de Average Pooling 2D. 41
21. Extracción de características de los espectrogramas. 41
22. Modelo secuencial básico. 43
23. Matriz de confusión para el mejor modelo de EfficientNetB7 56
24. Matriz de confusión para el mejor modelo de MobileNetV3L 58
25. Matriz de confusión para el mejor modelo de ConvNeXtBase 59

Índice de tablas

1. Tabla de comparación entre los modelos pre-entrenados utilizados [20]. 26
2. Ejemplo de estructura del CSV del conjunto de datos *Audio Set* [10] 32
3. Número de muestras obtenidas por clase 34
4. Parámetros y sus posibles valores 44
5. Parámetros y sus posibles valores 45
6. Resultados de los modelos primera iteración 49
7. Función de costo en entrenamiento y validación por las épocas de cada modelo. 50
8. Resultados de los modelos segunda iteración 52
9. Función de costo en entrenamiento y validación por las épocas de cada modelo. 53
10. Número de muestras por clase para la validación y obtención de métricas 54
11. Macro ROC-AUC (área bajo la curva ROC) de los mejores modelos 54
12. Métricas de desempeño por clase para el modelo EfficientNetB7 . . 55
13. Métricas de desempeño por clase para el modelo MobileNetV3L . . 57
14. Métricas de desempeño por clase para el modelo ConvNeXtBase . . 59
15. Comparación de métrica AUC con modelos entrenados del estado del arte. 60

1. Introducción

El uso de los sentidos es fundamental para la interacción diaria con el entorno, y entre ellos, la audición juega un papel crucial al permitir la detección de sonidos que pueden indicar situaciones de riesgo o amenaza. Sin embargo, las personas con pérdida auditiva enfrentan una desventaja significativa, aumentando el riesgo de sufrir accidentes, generando mayores costos económicos, dificultando el aprendizaje, reduciendo las oportunidades laborales y, en última instancia, deteriorando su calidad de vida.

Actualmente, alrededor del 5 % de la población mundial requiere rehabilitación para tratar la pérdida auditiva, y se estima que para el año 2050 más de 700 millones de personas tendrán una pérdida auditiva incapacitante. Este problema es más preocupante en países de ingresos medios y bajos, donde el acceso a tratamientos adecuados es limitado, aumentando el impacto negativo en la calidad de vida de las personas afectadas.

Existen dispositivos como implantes cocleares y audífonos que pueden mitigar los efectos de la pérdida auditiva. Sin embargo, estos dispositivos no son accesibles para todos debido a su costo, la necesidad de procedimientos quirúrgicos y terapias de aprendizaje del sonido. Además, no todos los problemas auditivos pueden ser resueltos con estas tecnologías, especialmente en casos de daño en el oído interno.

Ante esta problemática, surge la necesidad de desarrollar soluciones alternativas y accesibles. Una opción viable es un sistema computacional basado en aprendizaje de máquina que pueda describir textualmente eventos o señales sonoras indicativas de riesgo, utilizando el micrófono de un smartphone. Este sistema no solo reduciría la vulnerabilidad de las personas con discapacidad auditiva, sino que también podría ser una herramienta educativa para aquellos que están en proceso de aprender a interpretar sonidos.

En el desarrollo de este proyecto, se implementaron y optimizaron modelos de aprendizaje automático para la detección de sonidos relevantes para personas con discapacidad auditiva. Durante la primera fase de optimización, se utilizó una búsqueda de hiperparámetros mediante *Random Search*, obteniendo un desempeño

satisfactorio en términos de precisión categórica de validación. Los modelos que mostraron un mejor equilibrio entre precisión y estabilidad fueron *EfficientNetB7*, *MobileNetV3L* y *ConvNeXtBase*.

En la segunda fase de optimización, se mejoró aún más la precisión y estabilidad de estos modelos, incorporando capas adicionales de *Batch Normalization*. El modelo entrenado con *MobileNetV3L* destacó con mejor rendimiento general, logrando una alta capacidad de diferenciación entre clases, especialmente en sonidos críticos como vehículos de emergencia y herramientas eléctricas.

A pesar de que los resultados no alcanzan los niveles del estado del arte, las diferencias no son muy considerables. La investigación también identificó desafíos como la necesidad de técnicas efectivas de *data augmentation* para mejorar la cantidad y calidad de los datos de entrenamiento.

El trabajo futuro se enfocará en implementar mejoras como técnicas de *data augmentation*, capas convolucionales adicionales, y la implementación práctica del modelo en dispositivos móviles o plataformas web para su uso en tiempo real, lo que potencialmente ofrecerá una herramienta para mejorar la seguridad y calidad de vida de las personas con pérdida auditiva.

2. Descripción del problema

2.1. Planteamiento del Problema

En el día a día, se utilizan los sentidos como herramientas para obtener información del entorno. El sentido de la audición permite detectar sonidos que pueden comunicar si hay una situación de riesgo o amenaza para la vida, a través del oído. Cuando las personas se encuentran en distintos lugares, existe la posibilidad de que ocurran eventos inesperados, como un incendio, un perro mordiendo a una persona, un vehículo de emergencia pasando un semáforo en rojo, disparos, entre otros. Todos estos eventos mencionados pueden detectarse fácilmente con la ayuda del sentido de la audición cuando se tiene un conocimiento previo de su significado, lo que permite tomar decisiones a tiempo. Sin embargo, algunas personas tienen pérdida auditiva y se encuentran en desventaja, lo que aumenta el riesgo de sufrir un accidente, genera mayores gastos económicos, dificulta el aprendizaje, disminuyen las oportunidades laborales, y como consecuencia empeora la calidad de vida de estas personas.

Actualmente, alrededor del 5 % de la población mundial necesita rehabilitación para tratar su pérdida auditiva. Para el año 2050 se estima que más de 700 millones de personas (1 de cada 10) tendrán pérdida auditiva incapacitante. Además, casi el 80 % de las personas con pérdida auditiva incapacitante viven en países de ingresos medios y bajos [1], por lo cual no tienen las mejores oportunidades para acceder a buenos tratamientos o terapias, generando un impacto mayor en su calidad de vida.

Aunque existen algunos dispositivos como implantes cocleares y micrófonos que ayudan a reducir el impacto de la pérdida auditiva, estos implantes requieren un procedimiento quirúrgico y terapia para aprender o volver a aprender la sensación del sonido [2]. Para las personas que sufren problemas en el oído interno, los dispositivos existentes de asistencia auditiva convencionales pueden no ser suficientes [3]. Sumado a lo anterior, no todas las personas tienen acceso a estos dispositivos. Existen factores económicos que influyen en la calidad y velocidad del tratamiento. En el caso de los niños que no reciben el tratamiento adecuado a tiempo, pueden

enfrentar problemas en el desarrollo del lenguaje, lo que impacta en su alfabetización, autoestima y habilidades sociales. Estas dificultades pueden llevar a un bajo rendimiento académico y, como consecuencia, reducir las oportunidades de empleo en la adultez [4].

Es claro que la pérdida auditiva es un problema relevante [1][5], que aún requiere atención y trabajo para mejorar las condiciones de las personas con esta discapacidad [6], y quienes tienen menos acceso a los tratamientos, o quieren una alternativa, no cuentan con una herramienta de apoyo adicional. Una posible ayuda que impactaría positivamente a las personas que no tienen acceso a los dispositivos actuales, o no se benefician lo suficiente de estas soluciones, es un sistema computacional basado en aprendizaje de máquina, que describe textualmente los eventos o señales sonoras que permitan identificar un riesgo usando el micrófono de un smartphone. Esto ayudaría a reducir la vulnerabilidad de las personas con discapacidad auditiva en distintos espacios, y también, puede ayudar a las personas que reciben una ayuda tecnológica, pero primero deben aprender a procesar lo que escuchan, pues así podrán asociar los sonidos con su significado, apoyándose en la descripción textual que genera el sistema. Sumado a lo anterior, la posibilidad de tener un smartphone con acceso a internet es cada vez mayor, y es mucho mayor que la accesibilidad a los tratamientos y dispositivos que ayudan a tratar la discapacidad auditiva.

Por otro lado, se han desarrollado modelos de aprendizaje automático que permiten detectar distintos sonidos [7], pero estos no se centran en ayudar a personas con discapacidad auditiva, detectan sonidos que no son relevantes [8] para disminuir la vulnerabilidad de las personas con esta discapacidad.

Por lo tanto, este proyecto se centrará en implementar un modelo que aporte a la detección de los sonidos que indiquen una alerta o amenaza para las personas con esta discapacidad.

2.1.1. Formulación

¿Cómo implementar un modelo de aprendizaje automático que clasifique los sonidos en espacios abiertos orientado a apoyar a las personas con discapacidad auditiva?

2.1.2. Sistematización

- ¿Cómo preparar un dataset que incluya los sonidos más relevantes para la clasificación de estos sonidos?
- ¿Cómo entrenar un modelo de aprendizaje automático que logre clasificar correctamente distintos sonidos?
- ¿Cómo elegir los mejores hiperparámetros para mejorar el rendimiento del modelo?
- ¿Cómo evaluar el sistema implementado?

2.2. Objetivos

2.2.1. Objetivo General

Desarrollar un modelo de clasificación de sonidos a partir de técnicas de aprendizaje automático orientado a apoyar a las personas con discapacidad auditiva.

2.2.2. Objetivos Específicos

- Desarrollar un módulo de recolección y procesamiento de datos que permita extraer y preparar sonidos desde una base de datos existente para su clasificación.
- Implementar un modelo de clasificación a partir de técnicas de aprendizaje automático que permita la identificación de sonidos.
- Desarrollar una estrategia de búsqueda de hiperparámetros con el fin de maximizar el rendimiento del modelo entrenado.
- Realizar la validación de los modelos a través de métricas de desempeño según los algoritmos empleados.

2.3. Justificación

Este proyecto se lleva a cabo con el objetivo de brindar apoyo a las personas con discapacidad auditiva, quienes a menudo enfrentan barreras en la comunicación y dificultades en tareas del día a día debido a su condición. A través de la propuesta de un modelo de aprendizaje automático que clasifique distintos sonidos, se busca beneficiar a los miembros de esta comunidad, así como a aquellos interesados en el tema que puedan desarrollar nuevas alternativas o ayudas.

En particular, se espera que este modelo pueda beneficiar a las personas con discapacidad auditiva que buscan una herramienta de apoyo de mayor accesibilidad o alternativa a las existentes, quienes podrán utilizarlo como una herramienta para detectar sonidos que puedan indicar una amenaza. De esta manera, se espera que la investigación tenga un impacto positivo en el desarrollo de modelos de aprendizaje automático para clasificación de sonidos, y con esto, en futuras investigaciones o implementaciones, llevarlo a las personas que las necesiten.

Los aportes de esta investigación radican en la propuesta de un modelo que, a diferencia de los existentes, se centrará en compartir el conocimiento de forma clara, cómo se entrena e implementa el modelo, para que se pueda mejorar o implementar de diferentes formas. Además, se espera que los resultados de la investigación puedan ser utilizados por otros investigadores en el futuro, lo que contribuiría al avance del conocimiento en el área de *audio tagging*.

La viabilidad de la propuesta está respaldada por las soluciones similares existentes como la propietaria de Apple, el desarrollo existente de un modelo de clasificación de sonidos con técnicas clásicas y su implementación en una app móvil [9], los conocimientos de aprendizaje automático y aprendizaje profundo, los conocimientos de programación en Python 3, las librerías de Python para implementar estos modelos, un dataset etiquetado de Google (AudioSet) [10] de distintos sonidos que ha sido utilizado por diversas investigaciones para la tarea de reconocimiento de patrones de audio [11][8][12], y también, la investigación que se centra en cómo mejorar el entrenamiento de modelos que utilizan AudioSet dataset [13].

En resumen, esta investigación busca desarrollar y documentar un modelo entre-

nado de aprendizaje automático que pueda ser utilizado en el ámbito de audio tagging, con el fin de beneficiar a las personas con discapacidad auditiva y contribuir al avance del conocimiento en el área.

3. Marco Teórico

En esta sección, primero se abordará el contexto relevante para entender la problemática del proyecto, incluyendo los problemas que pueden enfrentar las personas con discapacidad auditiva, los diferentes tipos de discapacidad y su impacto. Posteriormente, se explicará lo necesario para comprender las herramientas y el conocimiento técnico empleado en el desarrollo del proyecto.

3.1. Pérdida auditiva

3.1.1. Definición

La pérdida auditiva se refiere a las personas que no logran escuchar al igual que quienes no tienen esta condición, es decir, que los umbrales de audición están por encima de 20 dB en ambos oídos [14], por lo cual no escuchan sonidos que se encuentren por debajo de 20 decibelios, cuando lo normal es tener el umbral auditivo en 0 dB, siendo este el nivel mínimo de presión sonora que el oído humano es capaz de percibir.

Según el *Servicio Nacional de Salud del Reino Unido* (National Health Service - NHS) [15], entre los tratamientos existentes para la pérdida auditiva permanente, se encuentran los audífonos e implantes, estos primeros no restablecen por completo la audición, pero pueden ayudar a amplificar los sonidos para que sean más fuertes y claros. En el caso de los implantes, se realiza una cirugía para adherir el dispositivo al cráneo o colocarlo en lo profundo del oído. Es importante aclarar que en algunos casos de pérdida auditiva, como en los que se tiene daño del nervio auditivo después del oído medio, los implantes o audífonos no benefician a las personas con esta condición [3].

3.1.2. Tipos de pérdida auditiva

Existen varios tipos de pérdida auditiva, que se clasifican de distintas maneras según la agencia nacional de salud pública (CDC) de Estados Unidos [16]. Para entender mejor los tipos de pérdida auditiva, primero es necesario conocer los componentes del sistema auditivo, entre los cuales se encuentra:

- **Oído externo:** Se compone de las orejas, el canal auditivo y la membrana timpánica que separa al oído externo del oído medio.
- **Oído medio:** Se compone del tímpano y tres huecesillos que envían ondas del tímpano al oído interno.
- **Oído interno:** Consiste del órgano de audición en forma de caracol llamado cóclea, canales semicirculares que contribuyen al equilibrio, y los nervios que se conectan al cerebro.
- **Nervio acústico:** Es el nervio que envía la información del sonido del oído al cerebro.

La composición de estos cuatro elementos se llama sistema auditivo, el cual procesa la información que viaja del oído al cerebro, pasando por vías nerviosas.

Es importante conocer los grados de pérdida auditiva, los cuales se clasifican de la siguiente manera:

- **Pérdida auditiva leve:** La persona puede escuchar algunos sonidos del habla pero no oye los susurros con claridad.
- **Pérdida auditiva moderada:** La persona puede que no escuche casi nada de lo que habla otra persona a un volumen normal.
- **Pérdida auditiva grave:** La persona no puede escuchar lo que dice una persona al hablar a un volumen normal y solo percibe algunos sonidos fuertes.
- **Pérdida auditiva profunda:** Aquí se clasifican las personas con sordera, la persona no oye nada de lo que se habla y solo logra oír algunos sonidos muy fuertes en el mejor de los casos, pues también puede que no oiga nada.

Por otro lado, la pérdida auditiva puede ocurrir de distintas maneras, como lo son:

- **Unilateral o bilateral:** Ocurre en un solo oído o en ambos.
- **Prelingüística o postlingüística:** Ocurre antes de que la persona aprenda a hablar o después. Cuando ocurre de manera prelingüística, aunque la persona logre beneficiarse de audífonos o implantes, debe aprender a procesar el sonido cuando reciba el tratamiento, pues aunque logre escuchar algo, no

sabrá identificar o asociar los sonidos, para este caso se requieren muchas terapias y aprendizaje.

- **Simétrica o asimétrica:** Ocurre con el mismo grado en ambos oídos o distinta en cada oído.
- **Gradual o repentina:** La pérdida auditiva empeora con el tiempo o ocurre de repente.
- **Fluctuante o estable:** La pérdida auditiva mejora o empeora con el tiempo o se mantiene igual.
- **Congénita o adquirida:** La pérdida auditiva está presenta al nacer o es adquirida más adelante en la vida.

3.1.3. Impacto

El impacto de la perdida auditiva puede ser muy profundo en la vida de las personas, según la OMS [14] puede causar la pérdida de la habilidad para comunicarse con los demás. En el caso de los niños retrasa el desarrollo del lenguaje, ocasionando diversas dificultades como problemas de salud mental y problemas económicos. También es importante saber que para el 2050 se estima más de 700 millones de personas con discapacidad auditiva o pérdida auditiva discapacitante, la cual se define cuando el umbral de la audición inicia después de los 35 decibelios en el mejor oído. Cerca del 80 % de las personas con pérdida auditiva discapacitante se encuentra en países de ingresos medios y bajos. La pérdida auditiva por lo general aumenta con la edad, más del 25 % de las personas mayores de 60 años son afectadas por la pérdida auditiva discapacitante.

3.2. Espectrogramas

El espectrograma es una herramienta utilizada para visualizar cómo cambian las frecuencias en una señal de audio a lo largo del tiempo [17]. Para generar un espectrograma, se toman múltiples Transformadas Discretas de Fourier (DFT) de pequeños fragmentos de la señal de audio, y los espectros resultantes se organizan uno tras otro. Esto permite que el espectrograma muestre las frecuencias del audio

a través del tiempo, mostrando en una sola gráfica la información de tiempo, frecuencia y amplitud. El algoritmo que permite hacer este representación se conoce como STFT o Transformada de Fourier de tiempo reducido.

La siguiente figura es tomada del curso de audio de la plataforma *Hugging Face* (*https://huggingface.co/learn/audio-course/es/chapter1/audio_ data*).

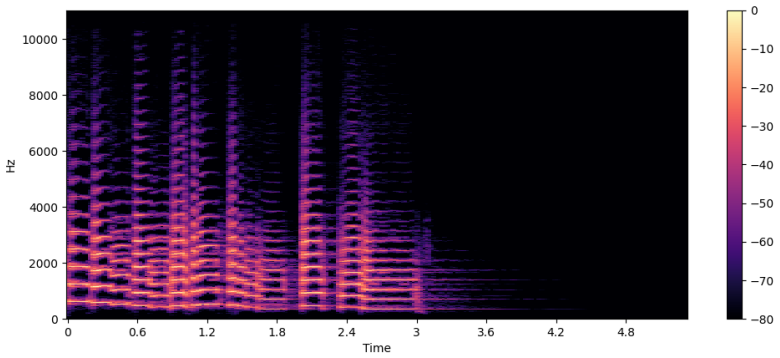


Fig. 1: Espectrograma de ejemplo [17]

En este espectrograma (ver figura 1), el eje x representa el tiempo y el eje y representa la frecuencia en hertz (Hz). Además, la intensidad del color representa el nivel en decibelios (dB) de la frecuencia en cada punto del tiempo. Es decir que para cualquier punto punto de la gráfica de un espectrograma, podemos saber en ese momento del tiempo los decibelios de cada frecuencia, en otras palabras sabemos qué tan fuerte se escucha cada frecuencia para cada instante temporal.

3.2.1. Espectrogramas en escala de Mel

Un espectrograma en escala de mel, o espectrograma mel es una variante de los espectrogramas que se utiliza comúnmente en tareas de aprendizaje automático. Es similar al espectrograma en la información que contiene, pero su eje de frecuencia es diferente. A diferencia del espectrograma estándar, se añade un paso ma2s en su generación, pues cada espectro de frecuencia generado con la STFT pasa luego por unos filtros para transformar las frecuencias a la escala mel.

La escala mel es una escala no lineal que permite acercar las frecuencias a las que

puede percibir el sistema auditivo humano, pues este es más sensible a los cambios en frecuencias bajas que a los cambios en las frecuencias altas.

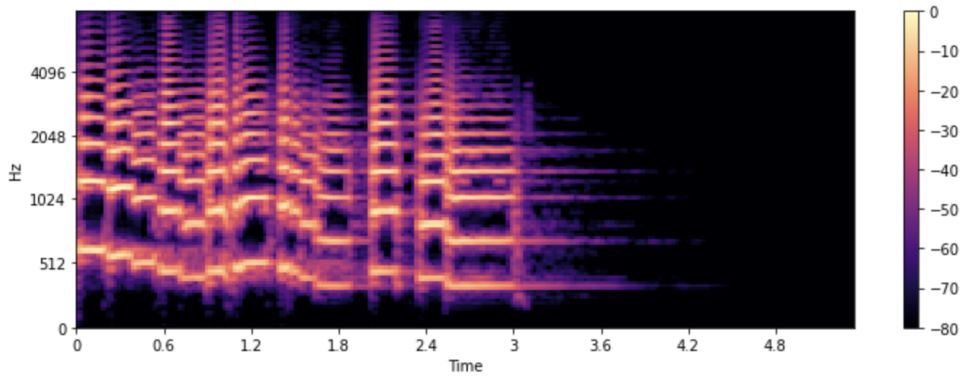


Fig. 2: Espectrograma Mel de ejemplo [17]

En la figura 2 se pueden observar tanto las similitudes como diferencias con el espectrograma estándar (ver figura 1). Si reducimos las diferencias a lo mínimo posible, se puede ver que el espectrograma mel parece un acercamiento o «zoom» al espectrograma estándar, recortando las frecuencias altas.

3.3. Inteligencia Artificial

En esta sección se definen los conceptos de inteligencia artificial, aprendizaje automático y aprendizaje profundo, según el libro *Deep learning with Python* [18].

La inteligencia artificial se puede describir como el esfuerzo por automatizar las tareas intelectuales que normalmente realizan los humanos. El campo de IA incluye el aprendizaje automático y el aprendizaje profundo.

3.4. Aprendizaje Automático

El aprendizaje automático introduce un nuevo paradigma a la programación, pues la forma usual de que un computador sea útil, es que un programador humano escriba las reglas (un programa de computador) para que la máquina procese la información de la entrada de datos a las salidas o respuestas deseadas.

En el caso del aprendizaje automático supervisado, la máquina recibe los datos de entrada, las salidas deseadas correspondientes, y con esto logra obtener las reglas para transformar la entrada en la salida deseada, es decir, el programador no escribe las reglas.

La siguiente figura 3 (tomada del libro [18]), muestra una representación clara de lo mencionado anteriormente.

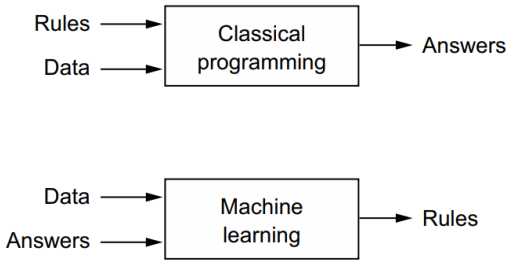


Fig. 3: Diferencia entre programación clásica y aprendizaje automático.

Para lograr el aprendizaje automático, se requieren 3 cosas:

1. Datos de entrada.
2. Ejemplos de la salida esperada.
3. Una forma de medir si el algoritmo está logrando el objetivo.

3.5. Aprendizaje Profundo y redes neuronales

El aprendizaje profundo es una de las ramas del aprendizaje automático, esta se centra en las redes neuronales artificiales. La palabra "profundo" se refiere a las capas sucesivas que procesan los datos de entrada, la cantidad de capas que contribuyen al modelo de los datos se llama profundidad del modelo.

Las redes neuronales consisten de 3 capas principales, la capa de entrada, las capas ocultas y la capa de salida. En las capas de entrada ingresan los datos para el entrenamiento de los modelos, en las capas ocultas ocurre el aprendizaje de diversas características y en la capa de salida se entrega los resultados, ya sea la clasificación, o la predicción de un valor. En la siguiente figura 4 se puede observar

una red neuronal básica, es importante tener en cuenta que esta cuenta con una sola capa oculta, pero por lo general se utilizan más de una capa oculta y puede tener diferentes cantidades de neuronas por capa.

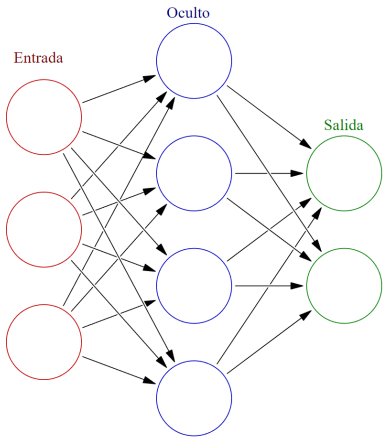


Fig. 4: Red neuronal básica.

Otra forma de entender el aprendizaje profundo, es con la siguiente figura 5 (del libro [18]), donde se puede ver la funcionalidad de las capas y la representación de los 3 puntos importantes del aprendizaje automático. Se puede observar que para una entrada de datos X se tiene una capa de entrada con unos pesos, luego otra capa consecutiva con pesos distintos, esto permite calcular diferentes pesos por capa y llegar al modelo que logrará predecir la salida correcta según la entrada de datos.

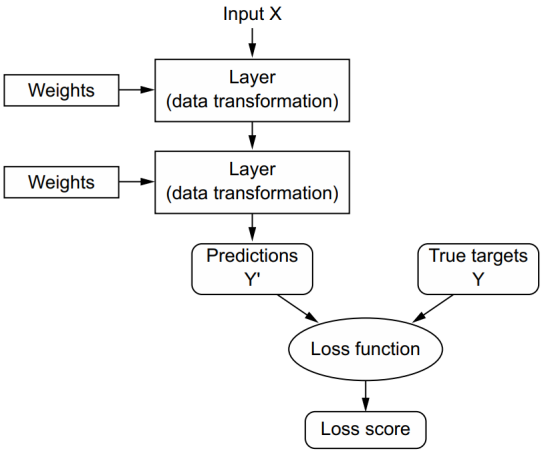


Fig. 5: Funcionamiento del aprendizaje profundo.

3.5.1. Función de Activación ReLU

La función de activación ReLU (Rectified Linear Unit) es una de las más utilizadas en redes neuronales. Si el valor de entrada es positivo, se mantiene, y si es negativo, se convierte en cero. Esta función permite que la red aprenda patrones complejos. Además, ayuda a mitigar el problema del desvanecimiento del gradiente, acelerando la convergencia del entrenamiento. [18]

3.5.2. Regularización L2

La regularización L2, es una técnica utilizada para reducir el sobre-ajuste en modelos de aprendizaje automático. Consiste en añadir una penalización al término de costo que es proporcional al cuadrado de los coeficientes de los pesos del modelo. Esto incentiva al modelo a mantener los pesos pequeños, promoviendo la simplicidad del modelo y mejorando su capacidad de generalización. [18]

3.5.3. Regularización Dropout

El Dropout es otra técnica de regularización que ayuda a prevenir el sobre-ajuste. Durante el entrenamiento de la red neuronal, el Dropout desactiva aleatoriamente un porcentaje de neuronas en cada capa en cada paso de entrenamiento. Esto obliga a la red a aprender redundancias y a ser más robusta, ya que no puede

depender de ninguna neurona específica para hacer sus predicciones. [18]

3.5.4. Función de activación Softmax

La función de activación softmax se utiliza en la capa de salida de una red neuronal para problemas de clasificación multiclase. Esta función convierte los valores de las neuronas de salida en probabilidades, asignando a cada clase un valor entre 0 y 1 que suma 1 en total. De esta manera, se puede interpretar la salida del modelo como la probabilidad de que una muestra pertenezca a cada una de las clases posibles. [18]

3.5.5. Función de costo Categorical Cross Entropy

La función de costo categorical cross entropy se utiliza para evaluar el rendimiento de modelos de clasificación multiclase. Compara la distribución de probabilidad de las predicciones por el modelo con la distribución real (donde la clase correcta tiene probabilidad 1 y las demás 0). La entropía cruzada mide la diferencia entre estas distribuciones, penalizando fuertemente las predicciones incorrectas y proporcionando una medida clara de qué tan bien está funcionando el modelo en términos de clasificación. [18]

3.6. Redes neuronales convolucionales

La principal diferencia entre una red neuronal y una red neuronal convolucional, son las capas convolucionales, en la siguiente figura 6 (tomada del libro [19]) se puede observar cómo se diferencian, es importante saber que la primera fila con el nombre *ANN* se refiere a las redes neuronales y la segunda fila con *CNN* se refiere a las redes neuronales convolucionales.

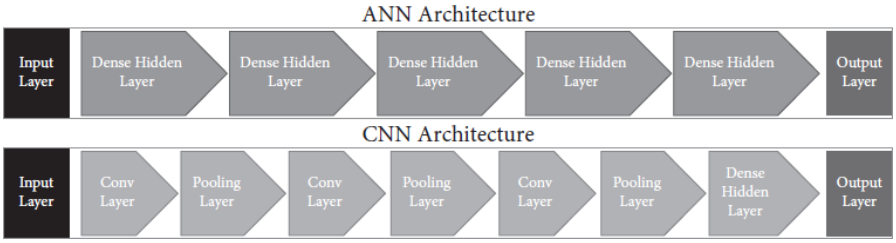


Fig. 6: Comparación entre redes neuronales y redes neuronales convolucionales.

En una red neuronal artificial (ANN), tenemos una capa de entrada, y la capa después de la capa de entrada es una capa oculta completamente conectada. Sin embargo, en el caso de una red neuronal convolucional (CNN), aplicamos un filtro en la capa de entrada y creamos características convolucionadas.

Aunque se pierde algo de información en la capa de convolución de una CNN, usar suficientes filtros puede minimizar esta pérdida y capturar más características útiles. [19]

3.7. Transfer Learning

Un enfoque común y altamente efectivo para el aprendizaje profundo en pequeños conjuntos de datos de imágenes es utilizar una red pre-entrenada. Una red pre-entrenada es una red guardada que se entrenó previamente en un conjunto de datos grande, típicamente en una tarea de clasificación de imágenes a gran escala. Si este conjunto de datos original es lo suficientemente grande y general, el modelo o red pre-entrenada puede considerarse un modelo genérico del mundo visual, y por lo tanto, sus características pueden ser útiles para muchos problemas diferentes de clasificación de imágenes, aunque estos nuevos problemas puedan involucrar clases completamente diferentes a las de la tarea original. [18]

El uso de modelos pre-entrenados para otras tareas de clasificación en el mismo dominio, se conoce como *transfer learning*, y la base de datos comúnmente utilizada para entrenar estos grandes modelos en la clasificación de imágenes, se conoce como *ImageNet*.

Modelo	**Tamaño (MB)**	**Top 1 Accuracy**	**Parámetros**
VGG16	528	71.3 %	138.4M
EfficientNetB7	256	84.3 %	66.7M
EfficientNetV2M	220	85.3 %	54.4M
MobileNetV3L	-	75.6 %	5.4M
ConvNeXtBase	338.58	85.3 %	88.5M

Tabla 1: Tabla de comparación entre los modelos pre-entrenados utilizados [20].

La métrica *Top 1 Accuracy* de la tabla 1 hace referencia al rendimiento de los

modelos con el conjunto de datos de validación de ImageNet.

3.7.1. VGG16

VGG16 es un modelo de redes neuronales convolucionales desarrollado por Visual Geometry Group (VGG) de la Universidad de Oxford. Se caracteriza por su simplicidad y arquitectura uniforme, utilizando capas convolucionales de 3x3. VGG16 se usa ampliamente para tareas de clasificación de imágenes y ha logrado un rendimiento de óptimo en el conjunto de datos ImageNet [21].

3.7.2. EfficientNetB7

EfficientNetB7 es parte de la familia EfficientNet, que mejora la capacidad de las redes neuronales convolucionales de una manera más eficiente. Los modelos EfficientNet equilibran la profundidad, el ancho y la resolución de la red para lograr un mejor rendimiento con menos parámetros. EfficientNetB7 es el modelo más grande de la serie EfficientNet original y proporciona alta precisión para tareas de clasificación de imágenes [22].

3.7.3. EfficientNetV2M

EfficientNetV2 es una familia de redes convolucionales que tienen una velocidad de entrenamiento más rápida y una mejor eficiencia de parámetros que los modelos anteriores. Para desarrollar esta familia de modelos, se utiliza una combinación de búsqueda y escalado de arquitectura neuronal consciente del entrenamiento, para optimizar conjuntamente la velocidad del entrenamiento y la eficiencia de los parámetros. Los modelos EfficientNetV2 se entrenan mucho más rápido que los modelos de última generación y son hasta 6,8 veces más pequeños [23].

EfficientNetV2M es una variante de tamaño mediano que ofrece un equilibrio entre velocidad y precisión.

3.7.4. MobileNetV3L

MobileNetV3 Large (MobileNetV3L) es parte de la familia MobileNet diseñada para entornos móviles y entornos con recursos limitados. MobileNetV3L combina

los avances de MobileNetV2 con técnicas de búsqueda de arquitectura novedosas y no linealidades mejoradas, como la función de activación h-swish. Proporciona alta eficiencia y rendimiento para aplicaciones de visión móviles [24].

MobileNetV3-Large es un 3,2 % más preciso en la clasificación de ImageNet y reduce la latencia en un 20 % en comparación con MobileNetV2

3.7.5. ConvNeXtBase

ConvNeXtBase es parte de la serie ConvNeXt, cuyo objetivo es modernizar las redes neuronales convolucionales estándar con elementos de diseño inspirados en transformadores de visión (ViT). Los modelos ConvNeXt mantienen la simplicidad de las CNN tradicionales al tiempo que integran funciones avanzadas como la normalización de capas y un diseño de cuello de botella invertido. ConvNeXtBase ofrece un equilibrio entre eficiencia y precisión computacional [25].

3.8. Hiperparámetros y su búsqueda

Los algoritmos de aprendizaje automático tienen hiperparámetros que, en algunos casos, deben ser ajustados para optimizar su rendimiento. Los algoritmos básicos suelen tener pocos hiperparámetros, mientras que los algoritmos de aprendizaje profundo tienen muchos más, afectando tanto la precisión como el tiempo de ejecución [19].

Para la búsqueda de hiperparámetros existen diversos algoritmos que de manera general, construyen conjuntos de hiperparámetros y realizan entrenamientos de los modelos utilizando estos conjuntos, luego se guarda el conjunto de hiperparámetros con el que se obtuvo mejor rendimiento según una métrica de desempeño y se utilizan para construir el modelo.

3.8.1. Random Search

La búsqueda aleatoria o *Random Search*, es un algoritmo de búsqueda de hiperparámetros aleatoria, es decir, se construyen los posibles conjuntos con una selección de parámetros aleatoria y se guarda el conjunto que obtenga el mejor rendimiento. Es importante notar que los conjuntos se construyen a partir de una selección

específica de posibles valores, porque de lo contrario serían infinitos valores para algunos parámetros numéricos.

Este método de búsqueda consume menos tiempo y recursos, pues en lugar de explorar exhaustivamente todas las combinaciones posibles como en la búsqueda en cuadrícula, selecciona aleatoriamente los hiperparámetros, lo que permite evaluar menos combinaciones. Aunque la búsqueda aleatoria puede no encontrar el conjunto óptimo de hiperparámetros, es más probable que encuentre un conjunto cercano al mejor en un tiempo significativamente menor. En muchos casos esta es la única opción debido a limitaciones de recursos computacionales o el tiempo disponible para la búsqueda. [26]

También es importante saber que en el *Random Search*, se define el número máximo de pruebas que se realizaran en la búsqueda, entonces esto permite aumentar el número de posibles combinaciones sin afectar negativamente la eficiencia temporal del algoritmo.

4. Metodología

Para cumplir con los objetivos de este trabajo, fue necesario obtener los datos del conjunto de datos *Audio Set* [10] como fragmentos de audio de 10 segundos. Luego, se convierten estos fragmentos en espectrogramas para que puedan ser procesados como imágenes por las redes neuronales convolucionales. Dado que el entrenamiento utilizando modelos preentrenados para clasificación de imágenes requiere una alta capacidad computacional, fue necesario aplicar una técnica de extracción de características sobre los espectrogramas para cada modelo entrenado. Los pesos obtenidos de la extracción de características se utilizan como entradas en los modelos construidos. Se utiliza la técnica de optimización Random Search para la búsqueda de los mejores hiperparámetros y, finalmente, se comparan los modelos obtenidos, se eligen tres y se obtienen sus métricas de desempeño. En el siguiente diagrama (ver figura 7), se podrá observar una síntesis de toda la arquitectura implementada para el desarrollo y metodología del proyecto. Posteriormente, en cada una de las sub-secciones se explicará a detalle cada uno de los procesos o fases mencionadas.

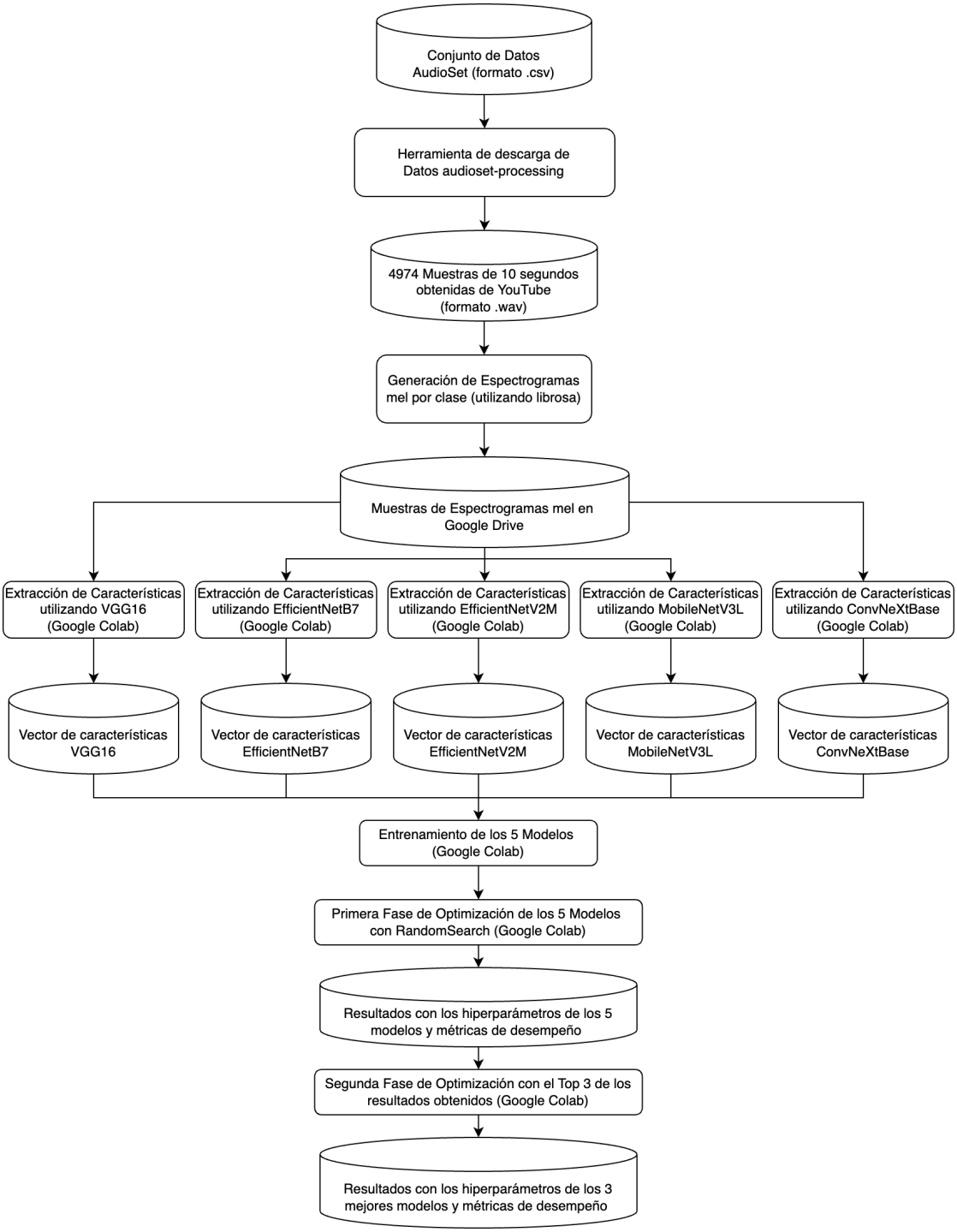


Fig. 7: Arquitectura completa para el desarrollo de la metodología

4.1. Obtención del conjunto de datos

Los datos utilizados para el entrenamiento de este trabajo fueron obtenidos del conjunto de datos *Audio Set* [10] de Google. Se eligió este conjunto de datos debido a la cantidad de muestras disponibles, las investigaciones previas que obtuvieron buenos resultados con su utilización, y la facilidad de acceso a las muestras. Estos datos son fragmentos de audio de 10 segundos recopilados de vídeos encontrados en YouTube. Google proporciona acceso a ellos a través de archivos CSV (comma separated values). En estos archivos se encuentran varias filas, donde cada una contiene el ID del vídeo en YouTube, los segundos en los que inicia y termina el sonido dentro del vídeo, y las etiquetas de clasificación para cada vídeo, que fueron anotadas por humanos. Para facilitar la descarga de los datos necesarios, se utiliza la herramienta open source *audioset-processing*, desarrollada por una usuaria de GitHub [27]. Se decide utilizar esta herramienta porque el conjunto de datos no se encuentra directamente disponible como pistas de audio, sino como fragmentos de audio extraídos de videos de YouTube (ver tabla 2). Por lo tanto, es necesario descargar cada una de las muestras, y la herramienta open source proporciona esta funcionalidad.

YTID	**Start Seconds**	**End Seconds**	**Positive Labels**
–PJHxphWEs	30.000	40.000	/m/09x0r, /t/dd00088
–ZhevVpy1s	50.000	60.000	/m/012xff
–aE2O5G5WE	0.000	10.000	/m/03fwl, /m/04rlf, /m/09x0r
–aO5cdqSAg	30.000	40.000	/t/dd00003, /t/dd00005
–aaILOrkII	200.000	210.000	/m/032s66, /m/073cg4
		⋮	

Tabla 2: Ejemplo de estructura del CSV del conjunto de datos *Audio Set* [10]

Al utilizar la herramienta, se encontraron fallos: una de las dependencias para la descarga de los vídeos de YouTube había dejado de funcionar, por lo tanto, fue necesario hacer una copia del proyecto de GitHub para modificarlo y solucionar estos problemas. Se decide arreglar la herramienta para permitir que otros investigadores puedan hacer uso de ella. Además, se agregó una nueva funcionalidad para limitar cuántos archivos se descargan por cada clase. Estas modificaciones y

mejoras se encuentran en la copia del repositorio *audioset-processing* [28]. En la siguiente figura proporcionada por la autora de la herramienta se puede observar el funcionamiento para la obtención del conjunto de datos:

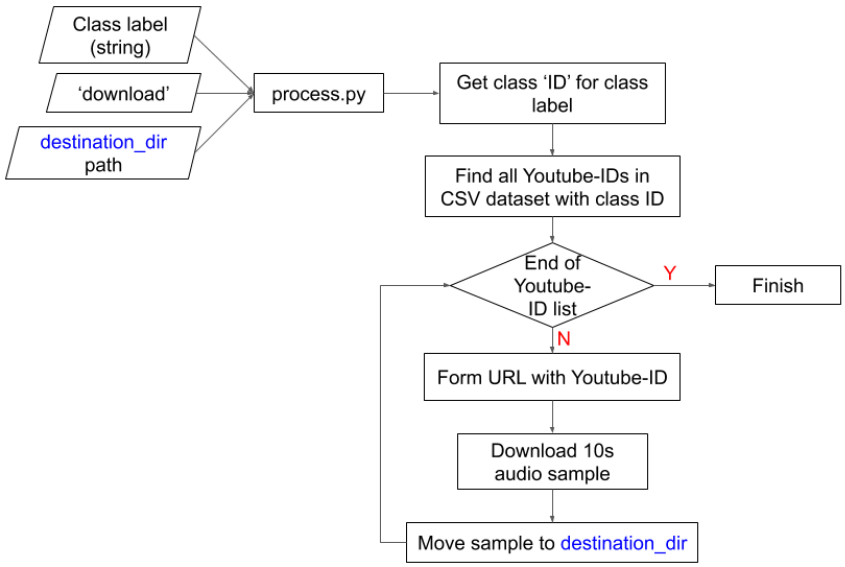


Fig. 8: Funcionamiento de la herramienta *audioset-processing* [27].

En la figura 8 se explica que, utilizando un programa codificado en Python, se ingresa la etiqueta de la clase a descargar y el directorio destinado para la descarga. Luego, el programa se encarga de obtener el ID de la clase y de encontrar todos los IDs de los vídeos en YouTube dentro de los CSVs que coincidan con dicho ID de la clase. Después, se generan las URLs de YouTube y se inicia la descarga de cada fragmento de audio de 10 segundos en el formato *.wav*.

Para este trabajo, se eligieron como principales y se descargaron las clases «Explosion», «Power Tool», «Gunshot» y «Emergency vehicle», esto debido a que eran las clases disponibles que representan un audio de alerta o amenaza para la integridad física y, además, se encontraron al menos 1000 muestras de cada una. Para evitar que los modelos entrenados clasificaran cualquier otro sonido que no correspondiera a estas clases en una de ellas, se construyó la clase «Other», que se compone de las clases «Music», «Bell», «Speech», «Silence» y «Sneeze», cada una con 200 muestras, para un total de 1000 muestras entre las cinco clases.

Es importante resaltar que para muchas de estas clases, según la información de los CSVs, se encontraban más muestras disponibles de las que fueron descargadas, esto es porque en varios casos, el vídeo de YouTube que contenía la muestra ya no existía en la plataforma o no se podía acceder para su descarga. Por lo tanto, se perdieron esas muestras y la cantidad disponible según la información del conjunto de datos ya no corresponde con la real. Es por esto que al final la clase «Explosion» quedó con 974 muestras para un total de 4974 muestras entre todas las clases como se observa en la tabla 3.

Clase	**Número de Muestras**
Emergency Vehicle	1000
Explosion	974
Gunshot	1000
Power Tool	1000
Music	200
Bell	200
Speech	200
Silence	200
Sneeze	200
Total	**4974**

Tabla 3: Número de muestras obtenidas por clase

4.2. Generación de espectrogramas

Luego de tener cada una de las clases organizadas por carpetas según su clase y cantidad de muestras descargadas, se desarrolla una herramienta [29] para llevar los fragmentos de audio a los histogramas. Esto se logró utilizando el módulo *Librosa* [30] de *Python 3*, el cual se especializa en análisis de música y audio. Al utilizar este módulo, se asegura una buena implementación debido a que está especializado en estas tareas.

Para generar los espectrogramas, utilizando la funcionalidad *librosa_load* se carga cada uno de los audios descargados con una frecuencia de muestreo de 16 KHz y se utiliza la técnica de *zero padding* [31][13] para que cada archivo quede como un vector de exactamente 10 segundos, esto se logra calculando la longitud deseada

del vector con la frecuencia de muestreo multiplicado por la duración en segundos, es decir $10s \cdot 16000\frac{1}{s} = 160000$. Entonces cada muestra del conjunto de datos se convierte en un vector de 160000 elementos representando una serie temporal de punto flotante, y las muestras que no llegan a los 160000 elementos son completadas con ceros al final del vector.

A partir de los vectores de las muestras, se genera cada espectrograma como una imagen de de 205×226 píxeles RGB, esta decisión se toma debido a que es necesario generar las imágenes con dimensiones pequeñas, para no tener problemas de almacenamiento ni de recursos computacionales en el momento del entrenamiento. En la siguiente figura 9 se puede observar una muestra de la clase «Emergency Vehicle».

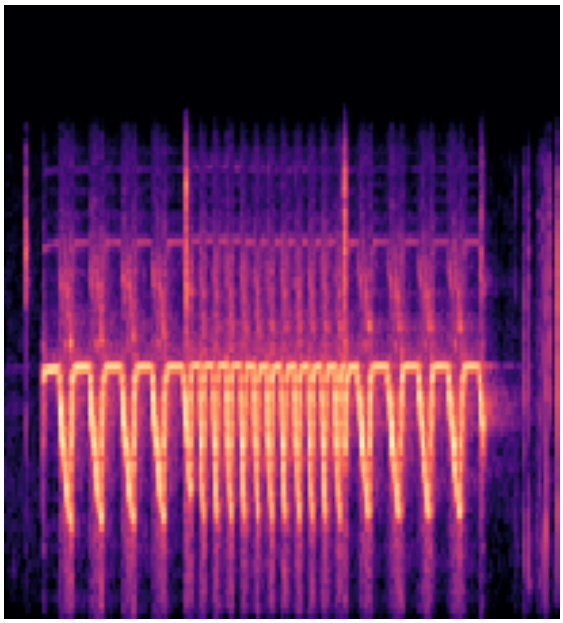


Fig. 9: Espectrograma mel de la muestra «Emergency Vehicle».

La elección de un espectrograma mel y no un espectrograma estándar se debe a que «el sistema auditivo humano es más sensible a los cambios en las frecuencias bajas que en las frecuencias altas, y esta sensibilidad disminuye de manera logarítmica a medida que la frecuencia aumenta» [17], esto también ayuda a la tarea de aprendizaje automático porque la imagen contiene más información relevante,

como se podrá observar en la siguiente figura 10 de la misma muestra pero con un espectrograma estándar. Por otro lado, se exploró también generando gráficos de la forma de la onda a través del tiempo (en inglés «waveform»), pero estos gráficos contenían muy poca información para la extracción de características y se descartó su uso.

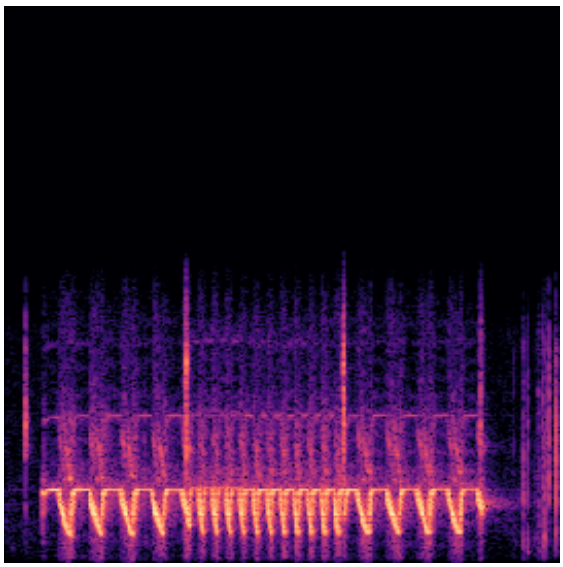


Fig. 10: Espectrograma estándar de la muestra «Emergency Vehicle».

En la figura 10 es claro que la imagen generada contiene menos información que la figura 9, pues gran parte de la imagen son píxeles negros sin información, y la parte visible contiene la misma información pero a menor detalle por la limitación de cantidad de píxeles en la imagen.

4.2.1. Muestras de espectrogramas por clase

A continuación se presentará una muestra de un espectrograma mel por cada clase del conjunto de datos.

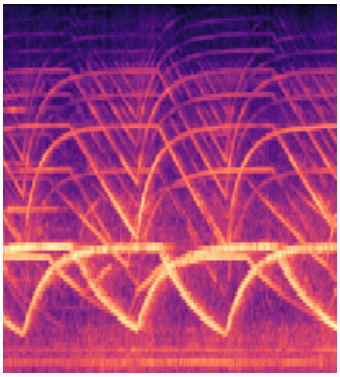


Fig. 11: Muestra de espectrograma mel para la clase «Emergency Vehicle».

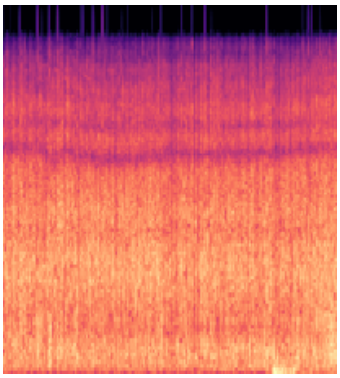


Fig. 12: Muestra de espectrograma mel para la clase «Explosion».

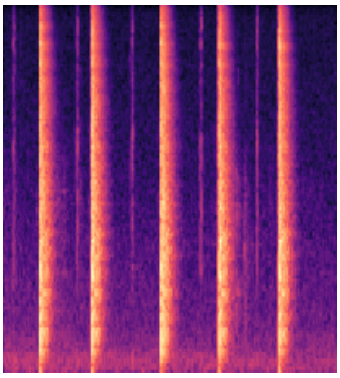


Fig. 13: Muestra de espectrograma mel para la clase «Gunshot».

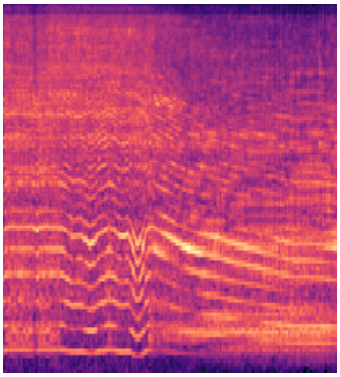


Fig. 14: Muestra de espectrograma mel para la clase «Power Tool».

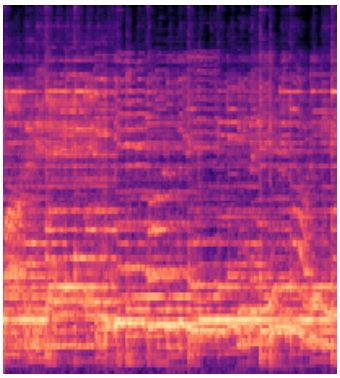


Fig. 15: Muestra de espectrograma mel para la clase «Music».

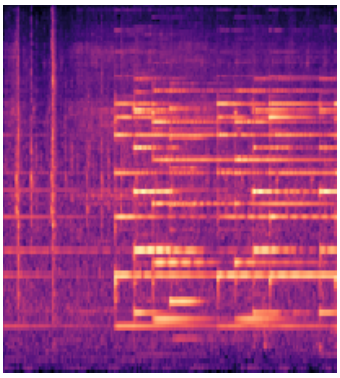


Fig. 16: Muestra de espectrograma mel para la clase «Bell».

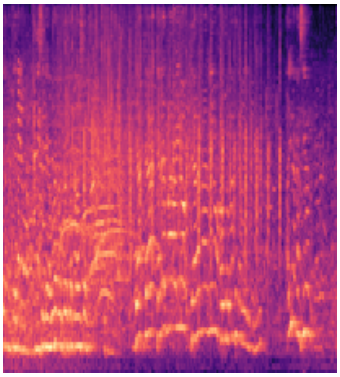


Fig. 17: Muestra de espectrograma mel para la clase «Speech».

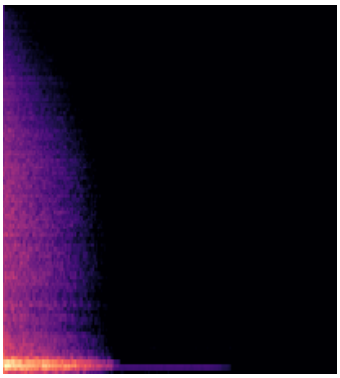


Fig. 18: Muestra de espectrograma mel para la clase «Silence».

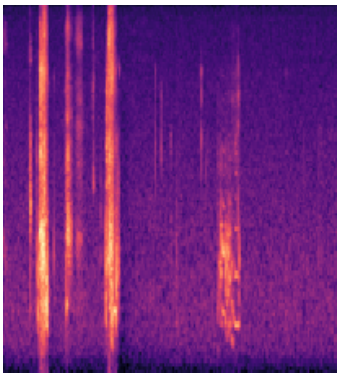


Fig. 19: Muestra de espectrograma mel para la clase «Sneeze».

4.3. Extracción de características de los espectrogramas

Teniendo todas las muestras en espectrogramas mel, se toma la decisión de utilizar transfer learning debido a los resultados positivos encontrados en el estado del arte [8] [11]. Para poder utilizar los modelos pre-entrenados con imágenes, fue necesario realizar una extracción de características de los espectrogramas porque entrenar los modelos utilizando el modelo base completo es muy costoso a nivel computacional y con el hardware que se tiene disponible para este trabajo, no sería viable.

La extracción de características permite que no sea necesario extraer las características de cada una de las imágenes más de una vez, con esto se obtienen los vectores con los pesos que se guardan para luego ser utilizados en el entrenamiento con otras capas ocultas. Esto se logra pasando cada imagen del conjunto de datos por cada uno de los modelos pre-entrenados, es decir se obtienen los pesos para: *VGG16*, *EfficientNetB7*, *EfficientNetV2M*, *MobileNetV3L* y *ConvNeXtBase*.

Los modelos pre-entrenados fueron elegidos debido a sus cantidad de parámetros y precisión presentes en la tabla 1. Además, se eligen porque tienen diferencias en su arquitectura y se encuentran disponibles en la reconocida herramienta *Keras* que fue la utilizada en este proyecto debido a su gran reconocimiento en el ámbito de aprendizaje automático con *Python 3* [20].

Para la extracción de características se carga el modelo pre-entrenado y se congelan sus pesos para que no se modifiquen, en otras palabras se congela el modelo base, esto se logra utilizando el módulo *Keras*. Luego, se utiliza el modelo base junto con una capa de `AveragePooling2D` con la siguiente configuración:

- `pool_size=(7, 7)` reduce la dimensionalidad tomando ventanas de 7x7 píxeles y obteniendo el valor promedio de esas ventanas.
- `strides=(7,7)` moverá la ventana de pooling en pasos de 7 píxeles.
- `padding="same"` completa la matriz con ceros cuando al mover la ventana quedan faltando datos para completar su tamaño.

La anterior configuración se eligió porque permite reducir la dimensionalidad en una magnitud de 7, para acelerar el proceso de aprendizaje de los modelos sin

perder características importantes.

En la siguiente figura 20 se puede ver un ejemplo del Average Pooling 2D aplicado a una matriz de 5x5 utilizando un pool size o ventanilla de (2,2), strides o pasos de (2,2) y padding «same».

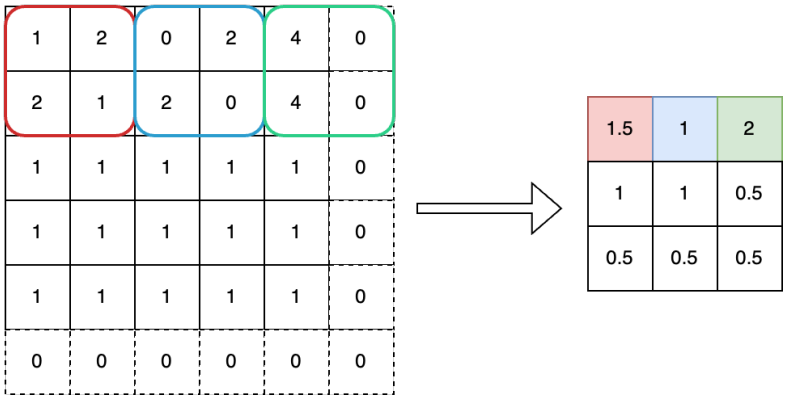


Fig. 20: Ejemplo de Average Pooling 2D.

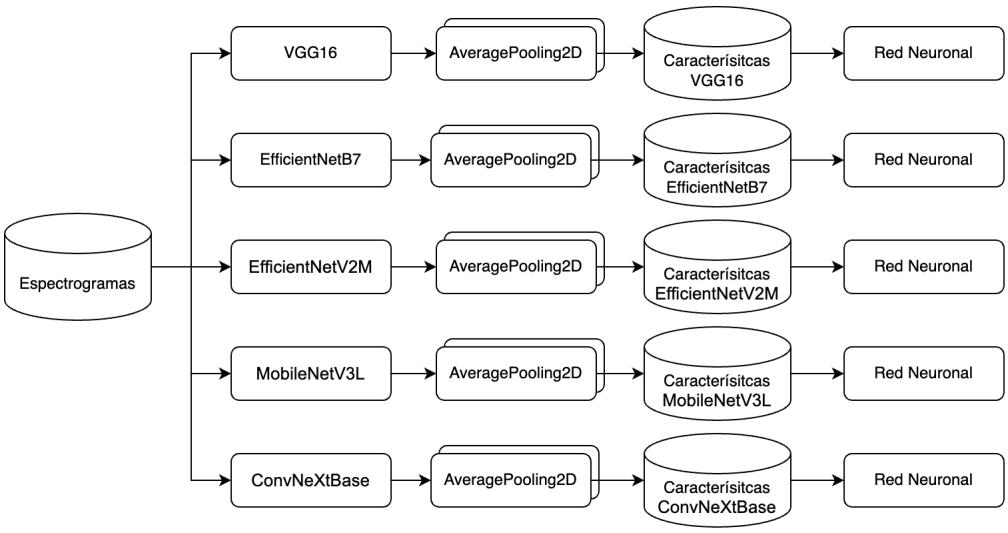


Fig. 21: Extracción de características de los espectrogramas.

En la figura 21 podemos observar que al final se usan las características extraídas para entrenar un modelo de redes neuronales para cada uno de los modelos pre-entrenados, a continuación se explicarán los detalles del entrenamiento.

4.4. Entrenamiento de los modelos

Todo el entrenamiento se encuentra en el repositorio de GitHub *thesis-model-training* [32], este se realiza utilizando la herramienta gratuita *Google Colab*, que accede a el conjunto de datos guardado en *Google Drive*, se guardan en *Google Drive* porque de esta manera se pueden utilizar directamente los datos desde la herramienta *Google Colab*, que es la herramienta gratuita que permitió el desarrollo de estre proyecto. Teniendo esto en cuenta, a continuación se enumeran los pasos que se siguieron para entrenar los modelos.

- **Carga de los datos:** Se leen los vectores de las características obtenidos en los pasos anteriores utilizando la función `numpy.load(path)` del módulo *Numpy* [33], y se cargan las etiquetas de cada uno de los elementos dentro de los vectores usando la función `pd.read_csv(path)` del módulo *Pandas* [34]. Se adaptan las etiquetas de las clases al formato binario *One Hot Encoding*.
- **División de los datos:** Para la división se decide partir las 4974 muestras en dos conjuntos, *Train* y *Validation*, el primero con el 80 % de las muestras y el segundo con el 20 %. Esto se logra utilizando la función `train_test_split()` del módulo *Scikit Learn* [35].
- **Entrenamiento de un modelo secuencial:** Se entrena un modelo secuencial sencillo con una capa de 512 neuronas que usa ReLU como activación y aplica regularización L2 para controlar el sobreajuste, una capa de Dropout para apagar aleatoriamente un porcentaje de las neuronas durante cada paso de entrenamiento, y una capa de salida de 5 neuronas (por las 5 clases) que utiliza la función de activación softmax donde cada neurona representa la probabilidad de pertenecer a cada una de las clases. El objetivo de este modelo es ver el rendimiento general y luego utilizar la técnica de optimización de hiperparámetros Random Search para obtener un mejor modelo.

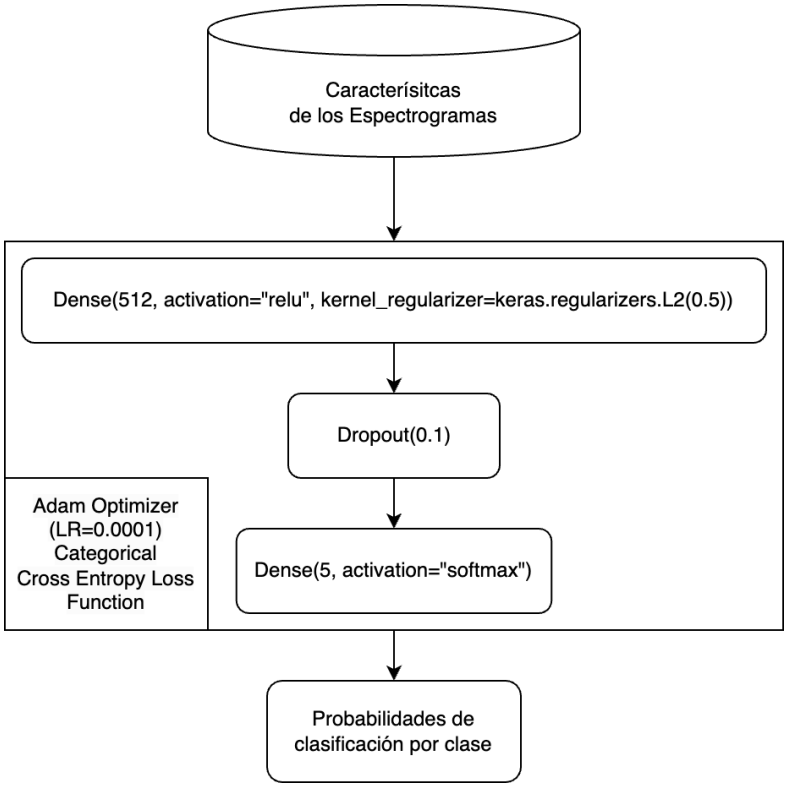


Fig. 22: Modelo secuencial básico.

La validación del desempeño de este modelo se realiza utilizando la métrica *CategoricalAccuracy* y se revisa la gráfica de la función de pérdida durante cada época del entrenamiento comparado con la gráfica de función de pérdida en la validación. Con la información anterior podemos saber qué tan bueno es el modelo para clasificar correctamente y si hay sobreajuste o subajuste.

4.5. Optimización de los modelos

Luego de lograr entrenar un modelo base, se empieza la tarea de optimización, esta se divide en 2 partes:

1. Optimización de los 5 modelos utilizando *RandomSearch* con una función que construye el modelo.

2. Elección y optimización de los 3 mejores modelos obtenidos en el paso ante-

rior. Se utiliza *RandomSearch* con un «HyperModel» que permite encontrar el mejor *batch_size* y se utiliza early stopping sobre el *validation_loss*.

A continuación se profundiza en cada una de estas partes.

4.5.1. Optimización general

Para la primera parte de la optimización, se utiliza el modelo secuencial de la figura 22 con los siguientes posibles parámetros:

Parámetro	**Valores**
learning_rate	$[1 \times 10^{-2}, 1 \times 10^{-3}, 1 \times 10^{-4}, 1 \times 10^{-5}]$
kernel_regularizer	[0,5, 0,05, 0,005, 0,0005]
dropout_rate	[0,6, 0,5, 0,4, 0,3, 0,2, 0,1, 0,005]

Tabla 4: Parámetros y sus posibles valores

Con la técnica de optimización de hiperparámetros *Random Search* se obtienen los mejores hiperparámetros que maximicen el *validation categorical accuracy* de cada modelo entrenado. Es importante resaltar que la búsqueda de los hiperparámetros se realiza utilizando el conjunto de datos *Train* que corresponde al 80 % de los datos totales, y este se parte nuevamente en 80 % para el train y 20 % para el validation, por lo cual en esta fase se está utilizando 64 % del total de los datos para el entrenamiento, y 16 % del total de los datos para la validación, esto ayuda a que al validar el modelo después de la búsqueda de hiperparámetros, se utilice un conjunto de datos «desconocido» para el modelo entrenado obteniendo una métrica de validación de mayor calidad.

El *Random Search* se ejecuta durante 20 trials con 2 ejecuciones por cada trial para reducir el impacto de la aleatoriedad y cada trial se ejecuta durante 100 épocas.

Lo importante de esta fase de optimización es identificar los modelos que obtuvieron el mejor desempeño según la métrica de *validation categorical accuracy* y que el modelo entrenado no tenga problemas de sobreajuste o subajuste.

4.5.2. Optimización de los mejores modelos

En la segunda parte de la optimización se eligen los 3 mejores modelos de la primera fase y se añade el *batch_size* a la búsqueda de hiperparámetros, por lo cual búsqueda queda de con las siguientes opciones en los parámetros:

Parámetro	**Valores**
learning_rate	$[1 \times 10^{-2}, 1 \times 10^{-3}, 1 \times 10^{-4}, 1 \times 10^{-5}, 1 \times 10^{-6}]$
kernel_regularizer	[0,5, 0,05, 0,005, 0,0005]
dropout_rate	[0,6, 0,5, 0,4, 0,3, 0,2, 0,1, 0,005]
batch_size	[16, 32, 64, 128]

Tabla 5: Parámetros y sus posibles valores

Se realiza la búsqueda de hiperparámetros con *Random Search* al igual que en la optimización general, pero ahora con 50 trials y con early stopping. Luego se validan los hiperparámetros creando un modelo que los utilice y validando con el conjunto de datos que no fue utilizado en el entrenamiento. También se hace la prueba de la búsqueda de hiperparámetros con una capa adicional de *Batch Normalization* después de la primera capa densa en los 3 mejores modelos y se determina si dejar o no la capa según el resultado obtenido.

4.6. Validación de los modelos

Teniendo los 3 mejores modelos con sus mejores hiperparámetros, se realiza la validación mediante la obtención de sus métricas.

Inicialmente se utilizan los mejores hiperparámetros para cada modelo y se construye el modelo base que se utilizado anteriormente con la única diferencia de que algunos utilizan la capa de *Batch Normalization* y otros no, todo eso será presentado en la sección de resultados.

Luego de construir los modelos con sus hiperparámetros, se entrenan durante 100 épocas utilizando early stopping de acuerdo a su *validation categorical accuracy*, además, se utiliza el parámetro de *patience* con un valor de 20 para que se detenga el entrenamiento si se entrena por más de 20 épocas y no mejora la métrica de va-

lidación. Finalmente, se emplea el parámetro *restore_best_weights* para restaurar los pesos del modelo a la época con mejor valor en la métrica de validación.

Se recopilan las métricas de *validation accuracy*, *ROC-AUC* por clase, matriz de confusión, *precision*, *recall* y *f1-score* de cada clase, y finalmente, el *ROC-AUC Macro*, es decir de todas las clases.

Cada una de las anteriores métricas, se eligieron por las siguientes características:

- La precisión de validación (validation accuracy) permite medir el porcentaje de ejemplos correctamente clasificados por el modelo en el conjunto de datos de validación, proporcionando una visión general del rendimiento. Sin embargo, en problemas de clasificación desequilibrada, esta métrica puede ser engañosa, ya que el modelo puede lograr una alta precisión simplemente prediciendo siempre la clase mayoritaria.
- El ROC-AUC por clase proporciona una evaluación del rendimiento del modelo en términos de su capacidad para distinguir entre las muestras positivas y negativas para cada clase. Es particularmente útil porque es independiente del umbral de decisión y proporciona una medida de la capacidad del modelo para clasificar correctamente las clases en un rango de diferentes umbrales. Un ROC-AUC alto indica una buena capacidad de diferenciación entren clases.
- La matriz de confusión es una tabla que visualiza el rendimiento del algoritmo de clasificación, mostrando las verdaderas clasificaciones positivas, negativas, falsas positivas y falsas negativas. Es crucial para identificar qué tipos de errores está cometiendo el modelo y permite una comprensión detallada del rendimiento en cada clase.
- La precisión (precision) mide la proporción de verdaderos positivos sobre el total de instancias clasificadas como positivas y es especialmente útil cuando el costo de los falsos positivos es alto. Proporciona información sobre la exactitud de las predicciones positivas del modelo.
- El recall (sensibilidad o tasa de verdaderos positivos) mide la proporción de verdaderos positivos sobre el total de instancias que realmente son posi-

tivas, siendo importante en situaciones donde es crítico capturar todas las instancias positivas, incluso a costa de tener más falsos positivos.

- El F1-score es la media de la precisión y el recall, proporcionando un balance entre ambos. Es útil cuando se necesita un equilibrio entre precisión y recall.
- El ROC-AUC Macro calcula el promedio del AUC para todas las clases, proporcionando una visión general del rendimiento del modelo en problemas multiclase, asegurando que el modelo tiene una buena capacidad de discriminación en todas las clases y no está sesgado hacia ninguna clase específica.

En el contexto de transfer learning, donde el modelo puede haber sido pre-entrenado en un dominio y adaptado a otro, estas métricas ayudan a identificar cómo el modelo se está generalizando y adaptando al nuevo dominio.

5. Resultados

En esta sección se presentan los resultados obtenidos para cubrir el objetivo de desarrollar un modelo de clasificación de sonidos a partir de técnicas de aprendizaje automático para apoyar a las personas con discapacidad auditiva.

Como se mencionó en la anterior sección, se realizaron dos fases para la optimización de los modelos, es por esto que se presentarán a continuación los resultados obtenidos en la primera fase. Esto ayudará a explicar cuáles fueron los modelos elegidos para la segunda fase de optimización y por qué se eligieron esos modelos.

5.1. Primera fase de optimización

En esta sección, se presentan los resultados obtenidos tras la primera fase de la optimización, es decir, la búsqueda de hiperparámetros con *Random Search* y el modelo base de la figura 22.

Como se puede observar en la tabla 6, se obtuvo un desempeño por encima del 70 % para la métrica de validación precisión categórica, o también llamada validation categorial accuracy. Esto nos indica que todos los modelos son capaces de clasificar correctamente un sonido de alguna clase por lo menos en 7 de cada 10 muestras del conjunto de validación.

Ahora entrando a detalle, si observamos también la tabla 7, los comportamientos de acuerdo a la función de costo o función de pérdida nos muestran que los modelos más estables fueron los que utilizaron los pesos de los modelos *VGG16*, *EfficientNetV2M* y *ConvNeXtBase*.

Sin embargo, al hacer un contraste con el validation categorical accuracy, se puede observar que los mejores modelos en cuanto a esa métrica, son: *EfficientNetB7*, *MobileNetV3L* y *ConvNeXtBase*. Estos tres últimos no coinciden con los 3 mejores en cuanto a estabilidad, es decir, los que menos presentaron indicios de sobreajuste o subajuste, pero gracias a las técnicas búsquedas de hiperparámetros, se logra una mejora en este aspecto para la segunda iteración.

Base Model	**Validation Categorical Accuracy**	**Hiperparámetros**
VGG16	70.65 %	learning_rate: 0.001 kernel_regularizer: 0.005 dropout_rate: 0.6
EfficientNetB7	75.08 %	learning_rate: 0.0001 kernel_regularizer: 0.0005 dropout_rate: 0.3
EfficientNetV2M	73.27 %	learning_rate: 0.0001 kernel_regularizer: 0.005 dropout_rate: 0.1
MobileNetV3L	76.68 %	learning_rate: 0.0001 kernel_regularizer: 0.0005 dropout_rate: 0.1
ConvNeXtBase	75.98 %	learning_rate: 0.0001 kernel_regularizer: 0.005 dropout_rate: 0.6

Tabla 6: Resultados de los modelos primera iteración

Gráficas función de costo: primera iteración

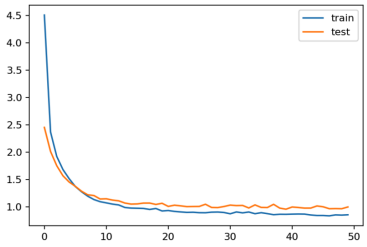
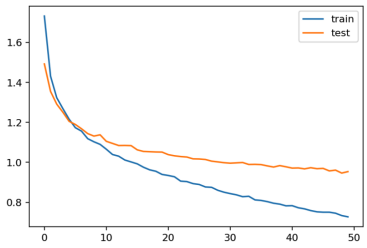
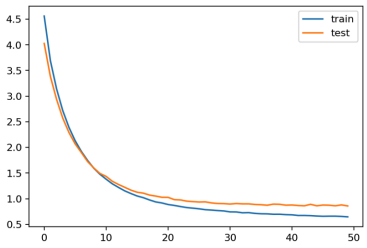
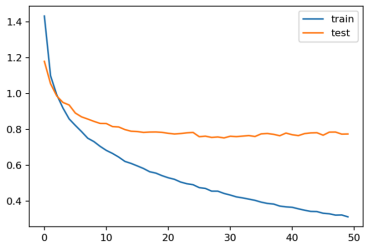
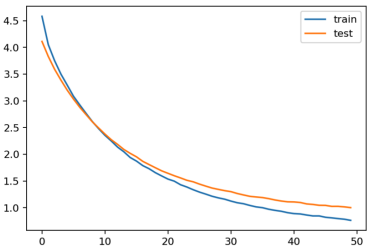
Modelo Base	Loss vs. Validation Loss
VGG16	
EfficientNetB7	
EfficientNetV2M	
MobileNetV3L	
ConvNeXtBase	

Tabla 7: Función de costo en entrenamiento y validación por las épocas de cada modelo.

5.2. Segunda fase de optimización

Debido a los resultados positivos en la métrica de validación de los modelos entrenados con los pesos de *EfficientNetB7*, *MobileNetV3L* y *ConvNeXtBase*, se decide utilizar estos mismos para obtener una mejor versión donde mejore el validation categorical accuracy y la estabilidad de los modelos.

Para los resultados de esta iteración, es importante recordar que el modelo base para la búsqueda de hiperparámetros cambió al agregar una capa de *Batch Normalization* con respecto al modelo de la figura 22, es decir, en la primera fase siempre se utilizó el modelo secuencial básico y en esta segunda se agregó la capa adicional.

Como se observa en la tabla 8, se ha añadido una nueva columna para la arquitectura del modelo. Esto se debe al cambio en la capa mencionado anteriormente. Además, dado que el modelo *ConvNeXtBase* no mejoró con la adición de la nueva capa, conserva la arquitectura original mostrada en la figura 22, entonces la columna permite diferenciar esa parte de la arquitectura que en la primera iteración no era necesario.

En cuanto a la métrica de desempeño de validation categorical accuracy, en los 3 modelos se logró una mejora.

Para el modelo entrenado con los pesos del modelo base *EfficientNetB7*, mejoró la métrica de desempeño en 0.40 %, y su estabilidad tuvo una mejora considerable como se puede observar en la tabla 9.

Para el modelo entrenado con los pesos del modelo base *MobileNetV3L*, mejoró la métrica de desempeño en 0.41 %, y su estabilidad mejoró por completo, pues según a gráfica encontrada en la tabla 9, se observa una estabilización completa de la función de costo tanto en el entrenamiento como en la validación. Para el modelo entrenado con los pesos del modelo base *MobileNetV3L*,

Para el modelo entrenado con los pesos del modelo base *ConvNeXtBase*, mejoró la métrica de desempeño en 0.80 %, y su estabilidad se mantuvo muy similar en relación a la primera iteración, sin ser esta negativa.

Según la métrica de desempeño validation categorical accuracy encontrada en la tabla 8, el mejor modelo fue *MobileNetV3L*, seguido de *ConvNeXtBase* y por último *EfficientNetB7*, pero es necesario revisar otras métricas de desempeño para hacer más clara esta diferencia, en la siguiente sección de resultados encontraremos la comparación.

Base Model	**Arquitectura**	**Validation Categorical Accuracy**	**Hiperparámetros**
EfficientNetB7	Base Model Dense(512, L2 Reg) Batch Normalization Dropout Dense (5, Softmax) - Early Stopping	75.48 %	learning_rate: 0.00001 kernel_regularizer: 0.05 dropout_rate: 0.6 batch_size: 16
MobileNetV3L	Base Model Dense(512, L2 Reg) Batch Normalization Dropout Dense (5, Softmax) - Early Stopping	77.09 %	learning_rate: 0.00001 kernel_regularizer: 0.5 dropout_rate: 0.4 batch_size: 16
ConvNeXtBase	Base Model Dense(512, L2 Reg) Dropout Dense (5, Softmax) - Early Stopping	76.78 %	learning_rate: 0.0001 kernel_regularizer: 0.005 dropout_rate: 0.6 batch_size: 32

Tabla 8: Resultados de los modelos segunda iteración

Gráficas función de costo: segunda iteración

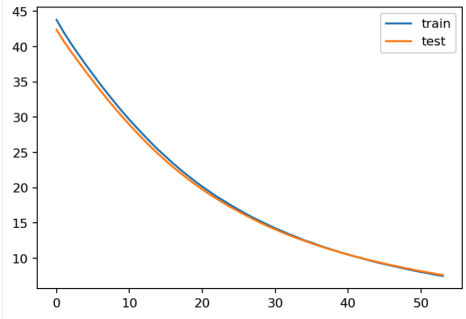
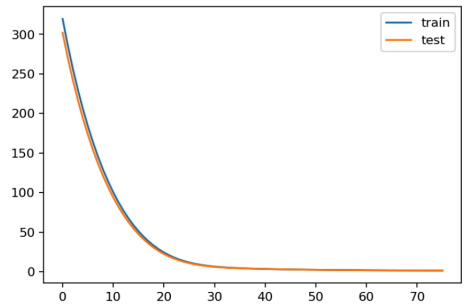
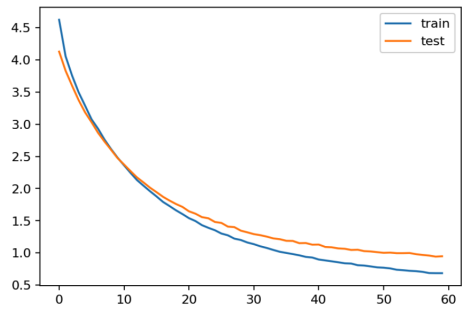
Modelo Base	Loss vs. Validation Loss
EfficientNetB7	
MobileNetV3L	
ConvNeXtBase	

Tabla 9: Función de costo en entrenamiento y validación por las épocas de cada modelo.

5.3. Métricas de desempeño de los mejores modelos

En esta sección se presentan las métricas de desempeño obtenidas para cada uno de los mejores modelos. Es importante aclarar que para cada uno de los siguientes modelos, se obtuvieron sus métricas con la misma cantidad de muestras es decir el 20 % del conjunto de datos, en la tabla 10 se puede observar la distribución de

las muestras:

Clase	Número de Muestras
Emergency Vehicle	210
Explosion	200
Gunshot	189
Power Tool	202
Other	194
Total	995

Tabla 10: Número de muestras por clase para la validación y obtención de métricas

En cuanto a la métrica Macro ROC-AUC que mide la capacidad del modelo para distinguir entre clases se obtuvo los siguientes resultados (ver tabla 11):.

Base Model	Macro ROC-AUC
MobileNetV3L	0.9432
ConvNeXtBase	0.9424
EfficientNetB7	0.9318

Tabla 11: Macro ROC-AUC (área bajo la curva ROC) de los mejores modelos

Con base en la métrica de desempeño Macro ROC-AUC, podemos observar que el modelo con los pesos de *MobileNetV3L* es el que mejor puede diferenciar eficazmente entre cada una de las clases, pues lo logra en el 94.32 % de los casos, lo que significa que cuando el modelo clasifica una muestra, tiene una alta probabilidad de que realmente la muestra sea de esa clase y no de otra. En segundo lugar encontramos el modelo con los pesos de *ConvNeXtBase* y finalmente el modelo *EfficientNet*.

5.3.1. Modelo entrenado con características de EfficientNetB7

Para el modelo entrenado con los pesos de *EfficientNetB7*, se encuentra que la clase que mejor clasifica es «Emergency Vehicle», con una precisión del 91 % indicando que de todas las predicciones en las que el modelo clasificó esta clase como positiva,

el 91 % de ellas son correctas. Las clases con menor precisión son «Gunshot» y «Explosion», sin embargo, teniendo en cuenta el recall o la métrica F1-score que combina las dos anteriores, la clase «Explosion» es la de menor rendimiento para este modelo (ver tabla 12).

Métricas de desempeño: EfficientNetB7

Clase	**ROC-AUC**	**Precision**	**Recall**	**F1-score**
Emergency Vehicle	0.9724	91 %	89 %	90 %
Explosion	0.8930	66 %	59 %	63 %
Gunshot	0.9050	64 %	66 %	65 %
Power Tool	0.9673	80 %	89 %	84 %
Other	0.9211	75 %	72 %	73 %

Tabla 12: Métricas de desempeño por clase para el modelo EfficientNetB7

Al observar la matriz de confusión (ver figura 23), es claro que el modelo en general clasifica correctamente, pues la diagonal de la matriz contiene la mayoría de los datos. También se ve claramente que las clases «Emergency Vehicle» y «Power Tool» son las que mejor rendimiento tienen, por otro lado «Explosion» y «Gunshot» tienen el menor rendimiento.

También es importante notar que la mayor cantidad de muestras mal clasificada, se concentra entre «Explosion» y «Gunshot», pues 41 muestras de «Explosion» fueron clasificadas como «Gunshot» y 38 muestras de «Gunshot» fueron clasificadas como «Explosion», esto puede ser debido a que el sonido que representan las dos clases contiene similitudes.

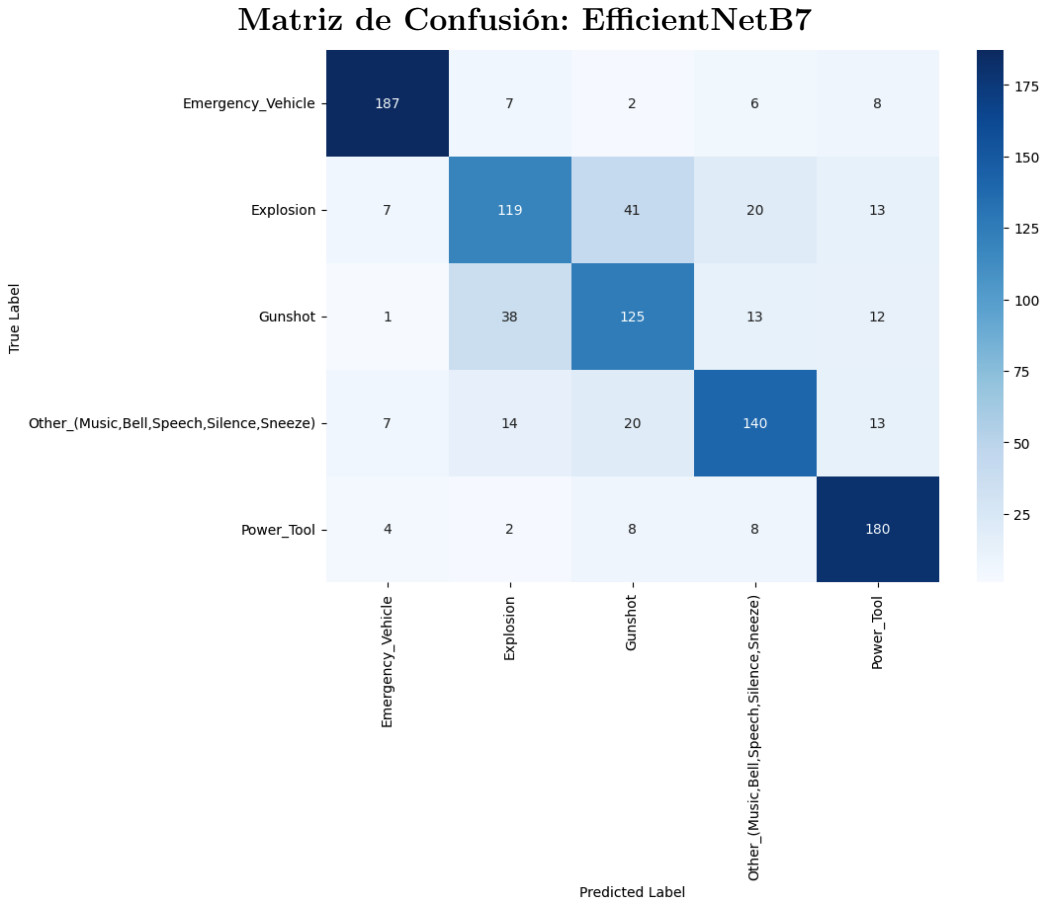


Fig. 23: Matriz de confusión para el mejor modelo de EfficientNetB7

5.3.2. Modelo entrenado con características de MobileNetV3L

Para el modelo entrenado con los pesos de *MobileNetV3L*, la clase que mejor logra clasificar es «Emergency Vehicle», y la clase con menor rendimiento es «Explosion» seguido de «Gunshot» (ver tabla 13). Este mismo comportamiento se presentó en el modelo entrenado con los pesos de *EfficientNetB7*.

Métricas de desempeño: MobileNetV3L

Class	ROC-AUC	Precision	Recall	F1-score
Emergency Vehicle	0.9782	92 %	86 %	89 %
Explosion	0.9015	73 %	62 %	67 %
Gunshot	0.9183	64 %	76 %	69 %
Power Tool	0.9741	83 %	87 %	85 %
Other	0.9440	75 %	74 %	75 %

Tabla 13: Métricas de desempeño por clase para el modelo MobileNetV3L

En cuanto a la matriz de confusión (ver figura 24) las clases con mejor rendimiento fueron «Emergency Vehicle» y «Power Tool», y las de menor rendimiento «Explosion» y «Gunshot». Aquí, a diferencia de la matriz de confusión de *EfficientNetB7* 23, la clase «Gunshot» tiene mayor cantidad de muestras clasificadas correctamente, es por eso mismo que el F1-score para esa clase en este modelo es 4 % mayor que en la misma clase en el modelo *EfficientNetB7*.

Al igual que en el anterior modelo, las clases «Explosion» y «Gunshot» son las que presentan más problemas posiblemente por su similitud en el sonido que ocasionan.

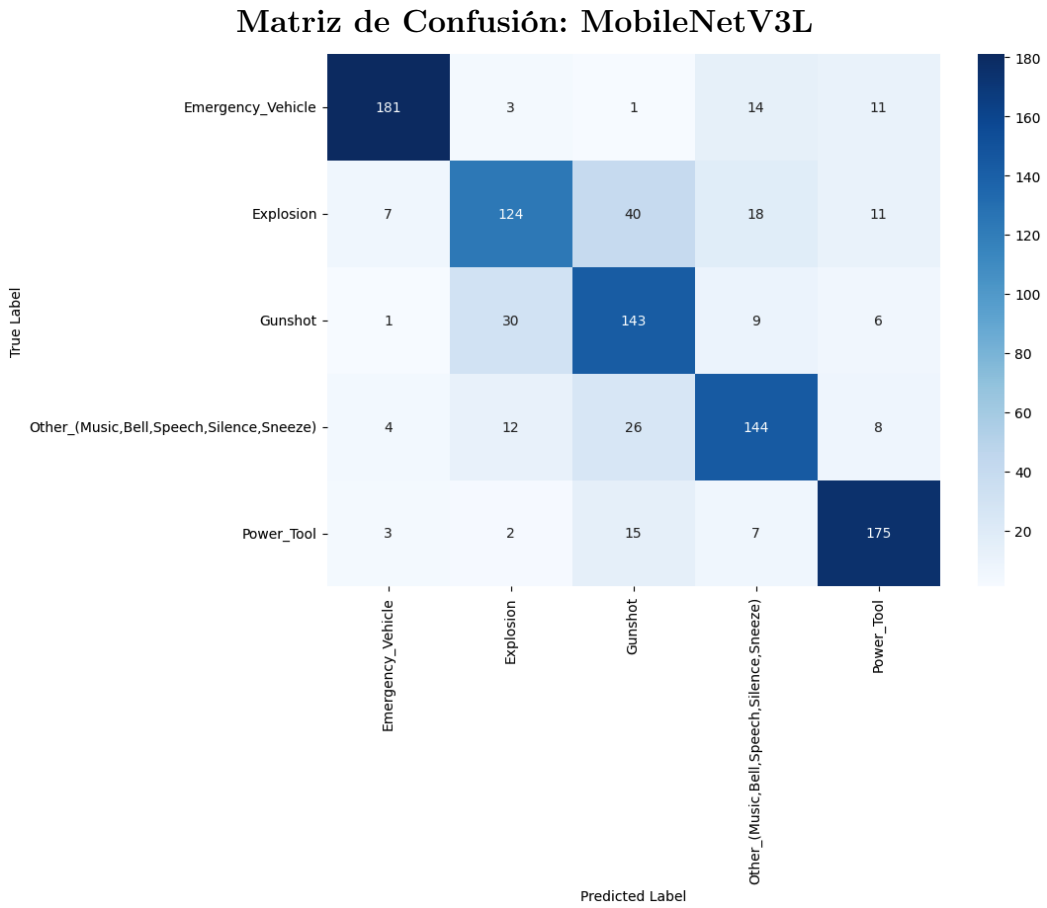


Fig. 24: Matriz de confusión para el mejor modelo de MobileNetV3L

5.3.3. Modelo entrenado con características de ConvNeXtBase

El comportamiento de modelo entrenado con las características de ConvNeXtBase, es similar a los otros dos modelos mencionados, nuevamente la clase «Emergency Vehicle» es la clase con mejor rendimiento para su clasificación (ver tabla 14), una de las posibles razones es que el sonido que esta genera es monótono y repetitivo, lo cual puede generar un patrón muy claro en los espectrogramas. De hecho, en la figura 11 presentada en la sección de generación de espectrogramas, se puede observar que hay una forma muy específica del patrón que se puede generar con el sonido de alarma de un vehículo de emergencia. Por otro lado, la clase «Explosion» es claramente la de menor rendimiento.

Métricas de desempeño: ConvNeXtBase

Class	ROC-AUC	Precision	Recall	F1-score
Emergency Vehicle	0.9840	91 %	90 %	90 %
Explosion	0.9028	69 %	58 %	63 %
Gunshot	0.9145	69 %	70 %	70 %
Power Tool	0.9687	86 %	83 %	84 %
Other	0.9421	68 %	82 %	74 %

Tabla 14: Métricas de desempeño por clase para el modelo ConvNeXtBase

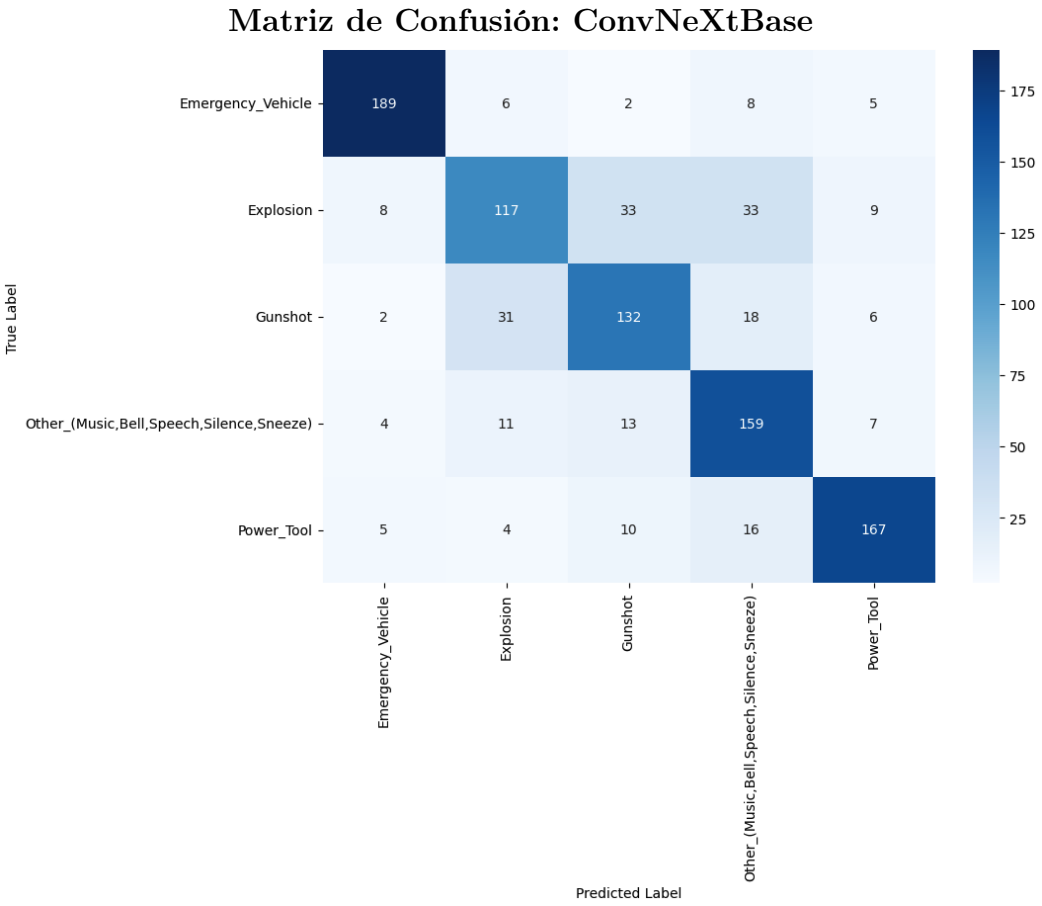


Fig. 25: Matriz de confusión para el mejor modelo de ConvNeXtBase

En cuanto a la matriz de confusión (ver figura 25), esta se asemeja más a la del modelo entrenado con los pesos de *EfficientNetB7*, pues hay menor similitud

entre la cantidad de muestras correctamente clasificadas para la clase «Gunshot» y «Other», sin embargo, a diferencia de los dos modelos anteriores, este logra la mayor cantidad de muestras clasificdas correctamente para la clase other.

5.4. Comparación con el estado del arte

En la siguiente tabla encontraremos los resultados de la métrica *AUC* para diferentes modelos del estado del arte. Cada uno de ellos fue entrenado con *AudioSet* en su totalidad (632 clases), a diferencia de este trabajo donde se escogieron 9 clases (ver tabla 3). En la primera columna encontraremos los modelos o una referencia al mejor modelo del artículo citado, en la parte inferior separado por una línea, los resultados de los mejores modelos en este trabajo.

Modelo	**AUC**
AudioSet Baseline [10]	0.959
Kong [36], 2018	0.965
Yu [37], 2018	0.970
TALNet [38], 2019	0.965
Kong [39], 2019	0.969
DeepRes [40], 2019	0.971
PANNs [11], 2020	0.973
PSLA [13], 2021	0.981
MobileNetV3L	0.943
ConvNeXtBase	0.942
EfficientNetB7	0.932

Tabla 15: Comparación de métrica AUC con modelos entrenados del estado del arte.

Al observar la anterior tabla 15, es claro que los resultados de los modelos entrenados para este trabajo, no se encuentran al nivel del estado del arte, sin embargo, los valores no son tan lejanos, para el modelo entrenado con *MobileNetV3*, se obtuvo una diferencia de 0.038 con el mejor modelo de la tabla, en cuanto al modelo entrenado con *ConvNeXtBase* difiere por 0.039 del mejor modelo, y, el modelo entrenado con *EfficientNetB7* se diferencia por 0.049, es decir, fue el más lejano al mejor modelo consultado del estado del arte por esta métrica.

6. Conclusiones

El objetivo final de este trabajo fue entrenar modelos de aprendizaje automático capaces de clasificar clases específicas del conjunto de datos AudioSet de Google. Estas clases fueron elegidas para asistir a personas con discapacidad auditiva, con la idea de identificar alertas y amenazas en su entorno.

Es importante resaltar que uno de los principales retos, fue adaptar el conjunto de datos a un formato de imágenes, pues el proceso es costoso computacionalmente. Aunque se logró encontrar y utilizar una herramienta para la descarga de las muestras de audio [28], fue necesario desarrollar herramientas adicionales para la transformación de esas muestras a imágenes [29].

Uno de los problemas que se logró identificar durante el desarrollo, fue la dificultad para utilizar *data augmentation*, esto debido a que aunque aumentar los datos de un conjunto de imágenes no es complicado si se realizan rotaciones, reflejos, entre otras, en este caso no era válido realizar este tipo de operaciones sobre las imágenes porque representan una gráfica en donde importa la dirección. Es decir, si tenemos una imagen de un gato y la reflejamos sobre uno de sus ejes, sigue siendo la imagen de un gato, pero si reflejamos un espectrograma que representa el sonido de una de nuestras clases, ya no se tendría una representación válida de estos sonidos sino de un sonido completamente diferente, lo que puede influir negativamente en el modelo. Una posible solución a este problema, es aplicar transformaciones a las muestras de audio originales antes de transformarlas en imágenes, por ejemplo añadir ruido, cambiar la frecuencia, etc., generando así, muestras adicionales válidas.

En cuanto al rendimiento de los modelos, se podría decir que puede ser mejorable. Este fue un trabajo de exploración y como la idea es entrenar un modelo capaz de identificar los sonidos sin ser este muy costoso computacionalmente, funcionaría mejor centrarse en entrenar el modelo *MobileNetV3L*, que como se sabe es el mejor para este tipo de tareas con recursos limitados [24]. Además, este fue el modelo con el que se alcanzó mejor rendimiento general de los entrenados en esta investigación (ver tabla 11).

6.1. Trabajo futuro

Para un trabajo futuro se proponen las siguientes mejoras y/o implementaciones:

- **Data augmentation:** Utilizar una técnica de data augmentation para generar más muestras de audio por clase. De esta manera se podría lograr obtener más muestras válidas y luego transformarlas a los espectrogramas. Esto disminuiría el sobreajuste y aumentaría la precisión de los modelos.
- **Capas convolucionales:** Al utilizar capas convolucionales adicionales al modelo pre-entrenado, se pueden extraer distintas características que podrían beneficiar el entrenamiento y la calidad del modelo.
- **Implementación de modelo entrenado**: Si se quiere llevar el modelo entrenado a clasificar sonidos de un ambiente real, es necesario implementar el modelo en un dispositivo móvil o página web que permita la carga de archivos y este identifique la clase a la que pertenece la muestra en tiempo real.

7. Bibliografía y referencias

[1] *Deafness and hearing loss*, en, feb. de 2023. dirección: https://www.who.int/news-room/fact-sheets/detail/deafness-and-hearing-loss (visitado 22-03-2023).

[2] *Implantes cocleares: Cómo funciona, quién los recibe | NIDCD*, es, mar. de 2021. dirección: https://www.nidcd.nih.gov/es/espanol/implantes-cocleares (visitado 22-03-2023).

[3] A. Pavlidou y B. Lo, «Artificial ear - a wearable device for the hearing impaired,» en *2021 IEEE 17th International Conference on Wearable and Implantable Body Sensor Networks (BSN)*, Athens, Greece: IEEE, jul. de 2021, págs. 1-4, ISBN: 9781665403627. DOI: 10.1109/BSN51625.2021.9507021. dirección: https://ieeexplore.ieee.org/document/9507021/ (visitado 22-03-2023).

[4] World Health Organization, *Childhood hearing loss: strategies for prevention and care*, en. Geneva: World Health Organization, 2016, ISBN: 9789241510325. dirección: https://apps.who.int/iris/handle/10665/204632 (visitado 22-03-2023).

[5] A. S. Jallu, T. Hussain, W. U. Hamid y R. A. Pampori, «Prelingual Deafness: An Overview of Treatment Outcome,» en, *Indian Journal of Otolaryngology and Head & Neck Surgery*, vol. 71, n.º S2, págs. 1078-1089, nov. de 2019, ISSN: 2231-3796, 0973-7707. DOI: 10.1007/s12070-017-1181-7. dirección: http://link.springer.com/10.1007/s12070-017-1181-7 (visitado 22-03-2023).

[6] N. A. Lesica, «Hearing Aids: Limitations and Opportunities,» en, *The Hearing Journal*, vol. 71, n.º 5, pág. 43, mayo de 2018, ISSN: 0745-7472. DOI: 10.1097/01.HJ.0000533807.87095.2a. dirección: http://journals.lww.com/00025572-201805000-00012 (visitado 22-03-2023).

[7] S. A. Romanov, N. A. Kharkovchuk, M. R. Sinelnikov, M. R. Abrash y V. Filinkov, «Development of an Non-Speech Audio Event Detection System,» en *2020 IEEE Conference of Russian Young Researchers in Electrical and Electronic Engineering (EIConRus)*, St. Petersburg y Moscow, Russia: IEEE, ene. de 2020, págs. 1421-1423, ISBN: 9781728157610. DOI: 10.1109/EIConRus49466.2020.9039115. dirección: https://ieeexplore.ieee.org/document/9039115/ (visitado 22-03-2023).

[8] S. Park y G. Kim, «Pretrained network-based sound event recognition for audio surveillance applications,» en *2021 International Conference on Information and Communication Technology Convergence (ICTC)*, Jeju Island, Korea, Republic of: IEEE, oct. de 2021, págs. 1306-1309, ISBN: 9781665423830. DOI: 10.1109/ICTC52510.2021.9621184. dirección: https://ieeexplore.ieee.org/document/9621184/ (visitado 22-03-2023).

[9] A. I. S. M. Ayu y K. K. Karyono, «Audio detection (Audition): Android based sound detection application for hearing-impaired using AdaBoostM1 classifier with REPTree weaklearner,» en *2014 Asia-Pacific Conference on Computer Aided System Engineering (APCASE)*, 2014, págs. 136-140. DOI: 10.1109/APCASE.2014.6924487.

[10] J. F. Gemmeke, D. P. W. Ellis, D. Freedman et al., «Audio Set: An ontology and human-labeled dataset for audio events,» en *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, New Orleans, LA: IEEE, mar. de 2017, págs. 776-780, ISBN: 9781509041176. DOI: 10.1109/ICASSP.2017.7952261. dirección: http://ieeexplore.ieee.org/document/7952261/ (visitado 22-03-2023).

[11] Q. Kong, Y. Cao, T. Iqbal, Y. Wang, W. Wang y 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, págs. 2880-2894, 2020, ISSN: 2329-9304. DOI: 10.1109/TASLP.2020.3030497.

[12] F. Schmid, K. Koutini y G. Widmer, *Efficient Large-scale Audio Tagging via Transformer-to-CNN Knowledge Distillation*, arXiv:2211.04772 [cs, eess], feb. de 2023. dirección: http://arxiv.org/abs/2211.04772 (visitado 02-04-2023).

[13] Y. Gong, Y.-A. Chung y J. Glass, «PSLA: Improving Audio Tagging With Pretraining, Sampling, Labeling, and Aggregation,» *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 29, págs. 3292-3306, 2021, ISSN: 2329-9304. DOI: 10.1109/TASLP.2021.3120633.

[14] *Deafness and hearing loss*, en. dirección: https://www.who.int/health-topics/hearing-loss (visitado 02-04-2023).

[15] *Hearing loss*, en, oct. de 2017. dirección: https://www.nhs.uk/conditions/hearing-loss/ (visitado 02-04-2023).

[16] CDC, *Tipos de pérdida auditiva (sordera) | CDC*, es-us, mayo de 2020. dirección: https://www.cdc.gov/ncbddd/spanish/hearingloss/types.html (visitado 02-04-2023).

[17] *Introducción a los datos de audio - Hugging Face Audio Course — huggingface.co*, https://huggingface.co/learn/audio-course/es/chapter1/audio_data, [Accessed 15-06-2024].

[18] F. Chollet, *Deep learning with Python*, Second edition. Shelter Island: Manning Publications, 2021, OCLC: on1289290141, ISBN: 9781617296864.

[19] V. R. Konasani y S. Kadre, *Machine learning and deep learning using python and TensorFlow*, en. Columbus, OH: McGraw-Hill Education, mar. de 2021.

[20] F. Chollet et al., *Keras*, https://keras.io, 2015.

[21] K. Simonyan y A. Zisserman, *Very Deep Convolutional Networks for Large-Scale Image Recognition*, 2015. arXiv: 1409.1556.

[22] M. Tan y Q. V. Le, *EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks*, 2020. arXiv: 1905.11946.

[23] M. Tan y Q. V. Le, *EfficientNetV2: Smaller Models and Faster Training*, 2021. arXiv: 2104.00298.

[24] A. Howard, M. Sandler, B. Chen et al., «Searching for MobileNetV3,» en *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, 2019, págs. 1314-1324. DOI: 10.1109/ICCV.2019.00140.

[25] Z. Liu, H. Mao, C.-Y. Wu, C. Feichtenhofer, T. Darrell y S. Xie, *A ConvNet for the 2020s*, 2022. arXiv: 2201.03545.

[26] T. Agrawal, *Hyperparameter Optimization in Machine Learning: Make Your Machine Learning and Deep Learning Models More Efficient*. Apress, 2021, ISBN: 9781484265796. DOI: 10.1007/978-1-4842-6579-6. dirección: http://dx.doi.org/10.1007/978-1-4842-6579-6.

[27] A. McDonagh, *aoifemcdonagh/audioset-processing*. 24 de jul. de 2023. dirección: https://github.com/aoifemcdonagh/audioset-processing.

[28] A. McDonagh y J. Villalobos, *Jeremias-V/audioset-processing*. 22 de oct. de 2023. dirección: https://github.com/Jeremias-V/audioset-processing.

[29] J. Villalobos, *Jeremias-V/audioset_thesis*. 22 de oct. de 2023. dirección: https://github.com/Jeremias-V/audioset_thesis.

[30] B. McFee, M. McVicar, D. Faronbi et al., *librosa/librosa: 0.10.2.post1*, ver. 0.10.2.post1, mayo de 2024. DOI: 10.5281/zenodo.11192913. dirección: https://doi.org/10.5281/zenodo.11192913.

[31] B. Zhu, K. Xu, Q. Kong, H. Wang e Y. Peng, «Audio Tagging by Cross Filtering Noisy Labels,» *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 28, págs. 2073-2083, 2020, ISSN: 2329-9304. DOI: 10.1109/TASLP.2020.3008832.

[32] J. Villalobos, *Jeremias-V/thesis-model-training*. 15 de jun. de 2024. dirección: https://github.com/Jeremias-V/thesis-model-training.

[33] C. R. Harris, K. J. Millman, S. J. van der Walt et al., «Array programming with NumPy,» *Nature*, vol. 585, n.º 7825, págs. 357-362, sep. de 2020. DOI: 10.1038/s41586-020-2649-2. dirección: https://doi.org/10.1038/s41586-020-2649-2.

[34] T. pandas development team, *pandas-dev/pandas: Pandas*, ver. latest, feb. de 2020. DOI: 10.5281/zenodo.3509134. dirección: https://doi.org/10.5281/zenodo.3509134.

[35] F. Pedregosa, G. Varoquaux, A. Gramfort et al., «Scikit-learn: Machine Learning in Python,» *Journal of Machine Learning Research*, vol. 12, págs. 2825-2830, 2011.

[36] Q. Kong, Y. Xu, W. Wang y M. D. Plumbley, «Audio Set Classification with Attention Model: A Probabilistic Perspective,» en *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2018, págs. 316-320. DOI: 10.1109/ICASSP.2018.8461392.

[37] C. Yu, K. S. Barsim, Q. Kong y B. Yang, *Multi-level Attention Model for Weakly Supervised Audio Classification*, 2018. arXiv: 1803.02353.

[38] Y. Wang, J. Li y F. Metze, *A Comparison of Five Multiple Instance Learning Pooling Functions for Sound Event Detection with Weak Labeling*, 2019. arXiv: 1810.09050.

[39] Q. Kong, C. Yu, Y. Xu, T. Iqbal, W. Wang y M. D. Plumbley, «Weakly Labelled AudioSet Tagging With Attention Neural Networks,» *IEEE/ACM*

Transactions on Audio, Speech, and Language Processing, vol. 27, n.º 11, págs. 1791-1802, nov. de 2019, ISSN: 2329-9304. DOI: 10.1109/taslp.2019.2930913. dirección: http://dx.doi.org/10.1109/TASLP.2019.2930913.

[40] L. Ford, H. Tang, F. Grondin y J. Glass, «A Deep Residual Network for Large-Scale Acoustic Scene Analysis,» en *Proc. Interspeech 2019*, 2019, págs. 2568-2572. DOI: 10.21437/Interspeech.2019-2731.