

GRADO EN INGENIERÍA
TELE-INFO-MULTI / C. DATOS



VNIVERSITAT
DE VALÈNCIA

TRABAJO FIN DE GRADO

ANÁLISIS Y CLASIFICACIÓN DE GÉNEROS
MUSICALES A TRAVÉS DE REDES NEURONALES
CONVOLUCIONALES

AUTOR:
DIEGO GARCÍA CAÍNZOS

TUTOR:
JAUME SEGURA GARCÍA



VNIVERSITAT
DE VALÈNCIA [Q] Escola Tècnica Superior
d'Enginyeria ETSE-UV

TRABAJO FIN DE GRADO

ANÁLISIS Y CLASIFICACIÓN DE GÉNEROS MUSICALES A TRAVÉS DE REDES NEURONALES CONVOLUCIONALES

AUTOR: DIEGO GARCÍA CAÍNZOS

TUTOR: JAUME SEGURA GARCÍA

Declaración de autoría:

Yo, Diego García Caínzos, declaro la autoría del Trabajo Fin de Grado titulado “Análisis y Clasificación de Géneros Musicales a través de Redes Neuronales Convolucionales” y que el citado trabajo no infringe las leyes en vigor sobre propiedad intelectual. El material no original que figura en este trabajo ha sido atribuido a sus legítimos autores.

Valencia, 26 de junio de 2024

Fdo: Diego García Caínzos

Resumen:

Las plataformas de streaming como Spotify, Youtube Music o Apple Music se han convertido en las principales formas de consumir música. Ahora los usuarios cuentan con la posibilidad de subir sus propios trabajos a estas plataformas y ser sus propios distribuidores.

La aparición de este fenómeno ha provocado un aumento el número de artistas y canciones disponibles en la red, haciendo inviable el etiquetado manual de características musicales. Esto ha generado una creciente demanda en herramientas de reconocimiento y etiquetado automático de canciones. Inspirado por esta demanda se plantea la temática de este proyecto.

El objetivo del trabajo es crear un clasificador de géneros musicales con técnicas de aprendizaje automático. El conjunto de entrenamiento está formado por fragmentos de canciones de 30 segundos pertenecientes a 10 géneros diferentes, que posteriormente son transformados en espectrogramas de Mel. Los espectrogramas son representaciones visuales de la señal de audio en función del tiempo y la frecuencia, lo que permite capturar de manera detallada las características musicales, incluyendo el ritmo, la armonía, el tono y otros aspectos.

Una vez procesados correctamente, los espectrogramas son sujetos de entrenamiento y prueba para el diseño de una red neuronal convolucional (CNN) diseñada específicamente para esta tarea.

Abstract:

Streaming platforms such as Spotify, Youtube Music or Apple Music have become the main ways to consume music. Now users have the possibility to upload their own works to these platforms and be their own distributors.

The emergence of this phenomenon has led to an increase in the number of artists and songs available on the network, making the manual tagging of musical features unfeasible. This has generated a growing demand for automatic song recognition and tagging tools. Inspired by this demand, the subject of this project arises.

The objective of the work is to create a music genre classifier using machine learning techniques. The training set consists of 30-second song fragments belonging to 10 different genres, which are subsequently transformed into Mel spectrograms. Spectrograms are visual representations of the audio signal as a function of time and frequency, allowing detailed capture of musical characteristics, including rhythm, harmony, pitch and other aspects.

Once correctly processed, the spectrograms are then subjected to training and testing for the design of a convolutional neural network (CNN) designed specifically for this task.

Resum:

Les plataformes de streaming com Spotify, YouTube Music o Apple Music s'han convertit en les principals formes de consumir música. Ara els usuaris tenen la possibilitat de pujar els seus propis treballs a aquestes plataformes i ser els seus propis distribuïdors.

L'aparició d'aquest fenòmen ha augmentat el nombre d'artistes i cançons disponibles en la xarxa, fent inviable l'etiquetatge manual de característiques musicals. Això ha generat una creixent demanda en eines de reconeixement i etiquetatge automàtic de cançons. Inspirat per aquesta demanda, es planteja la temàtica d'aquest projecte.

L'objectiu del treball és crear un classificador de gèneres musicals amb tècniques d'aprenentatge automàtic. El conjunt d'entrenament està format per fragments de cançons de 30 segons pertanyents a 10 gèneres diferents, que posteriorment són transformats en espectrogrames de Mel. Els espectrogrames són representacions visuals del senyal d'àudio en funció del temps i la freqüència, la qual cosa permet capturar de manera detallada les característiques musicals, incloent-hi el ritme, l'harmonia, el to i altres aspectes.

Una vegada processats correctament, els espectrogrames són subjectes d'entrenament i prova per al disseny d'una xarxa neuronal convolucional (CNN) dissenyada específicament per a aquesta tasca.

Agradecimientos:

A mi familia y amigos, por su constante apoyo, comprensión y ánimo incondicional. Su confianza en mí ha sido una fuente de motivación inagotable.

Índice general

1. Introducción	17
1.1. Introducción	17
1.2. Motivación	17
1.3. Objetivos	18
1.4. Índice de acrónimos	18
2. Estado del arte	19
2.1. Análisis de aplicaciones similares	21
2.2. Tecnologías	22
3. Requisitos, especificaciones, coste, riesgos, viabilidad	25
3.1. Requisitos	25
3.1.1. Requisitos funcionales	25
3.1.2. Requisitos no funcionales	26
3.2. Costes	26
3.3. Riesgos	27
3.4. Viabilidad	28
3.5. Planificación	28
4. Análisis	31
4.1. Onda Sonora	31
4.1.1. Características de la onda sonora	31
4.1.2. Propiedades musicales del sonido	32
4.2. Escalas	32
4.3. Extracción de características	34
4.3.1. La transformada de Fourier	35
5. Diseño	37
5.1. Aprendizaje Automático	37
5.1.1. Aprendizaje no supervisado	37

5.1.2. Aprendizaje supervisado	37
5.2. Neurona Artificial	38
5.2.1. Estructura de la neurona	38
5.3. Perceptrón	39
5.3.1. Funcionamiento de la red	39
5.3.2. Función de activación	40
5.4. Tipos de redes neuronales	40
5.5. Inspiración biológica	41
5.5.1. Áreas de la corteza visual	41
5.6. Arquitectura	42
5.6.1. Componentes de una CNN	43
5.6.2. Operación de convolución	44
5.7. Aprendizaje	45
6. Implementación y pruebas	47
6.1. Implementación	47
6.1.1. Análisis Exploratorio de Datos	47
6.1.2. Procesado de audio	47
6.1.3. Aumento de datos	50
6.1.4. Arquitectura de la red	55
6.1.5. Interfaz	57
6.2. Resultados	57
6.2.1. Filtro de características	59
6.2.2. Tiempo de ejecución	60
6.2.3. Prueba de funcionalidad	60
7. Conclusiones	69
7.1. Conclusiones	69
7.2. Trabajo futuro	70
Bibliografía	71

Capítulo 1

Introducción

1.1. Introducción

La barrera de entrada a las tecnologías IA ha disminuido exponencialmente desde el lanzamiento de ChatGPT. El uso de ellas se está haciendo más común en todos los sectores, incluyendo usuarios no profesionales. Cada vez es más frecuente que un conocido use un asistente de voz como Siri o Alexa, desbloquee su smartphone con reconocimiento facial, conduzca un coche autónomo como los famosos Tesla de Elon Musk, o utilice un chatbot para realizar una consulta en el trabajo. El rango de acción aumenta cada día y va invadiendo el entorno cotidiano.

En el ámbito de la música, las plataformas de streaming se han convertido en los principales distribuidores. Una de las principales razones para este cambio ha sido la creación de algoritmos de recomendación de canciones y artistas, haciendo que la experiencia de escuchar música sea totalmente personalizada.

Una de las aplicaciones de la IA en la música es la clasificación automática de géneros musicales. Debido al gran número de pistas que son publicadas en la red, las herramientas de etiquetado automático son muy solicitadas. Las redes neuronales son el método con mejor rendimiento en las tareas de este tipo. Las CNN, cuentan con la habilidad de reconocer patrones en datos visuales y se utilizan para analizar espectrogramas de audio, ofreciendo una clasificación eficiente y precisa. Los espectrogramas representan señales de audio visualmente, y son ideales para capturar características musicales como la armonía, el timbre y el ritmo.

1.2. Motivación

Los últimos avances en inteligencia artificial (IA) han generado una ola de expectación e interés por parte del público y el sector empresarial. Hoy por hoy la IA se está acercando cada vez más al público general y es la tecnología que ha generado mayor repercusión desde la creación y masificación de Internet.

Nuevas herramientas IA que mejoran la calidad de vida de las personas se publican diariamente, en campos como la salud, educación o seguridad, entre muchas otras.

Es por ello que he tomado la decisión de especializarme en esta rama y realizar el TFG sobre esta temática fue un paso lógico que ha hecho mucho más ameno su desarrollo.

Además, una de mis mayores aficiones es la música, tanto su escucha como su producción, y desde el principio tuve la intención de mezclar ambos en mi TFG. Esta intersección entre la IA y la música además de ser interesante personalmente, es un campo de investigación prolífico que cuenta con la atención de las principales empresas del tecnológicas y musicales.

1.3. Objetivos

El objetivo consiste en presentar una aplicación capaz de detectar el género de una canción con alta precisión. Para lograr este objetivo, se llevan a cabo varias etapas clave.

- Diseñar y entrenar un modelo de red neuronal convolucional (CNN) capaz de analizar espectrogramas de audio y clasificar géneros musicales con alta precisión.
- Implementar una interfaz de usuario sencilla que permita la carga de archivos de audio y la visualización de las predicciones del modelo.
- Evaluar y optimizar el rendimiento del modelo utilizando métricas adecuadas para asegurar su robustez y eficiencia en diversas condiciones.

1.4. Índice de acrónimos

- **CNN:** Red Neuronal Convolutional
- **MFCC:** Mel Frequency Cepstral Coefficients
- **IA:** Inteligencia Artificial
- **STFT:** Short-time Fourier Transform

Capítulo 2

Estado del arte

En el campo musical, se ha apostado por la implementación de tecnologías de Machine Learning en los últimos años. El consumo musical ha pasado de tener forma de DVD a las ya familiares playlists, de las cuales muchas son creadas por algoritmos de recomendación basados en modelos de aprendizaje profundo.

La IA es la encargada de la gestión de las tendencias, lo que ha supuesto una democratización en un mercado en el que antes era muy difícil hacerse un hueco con pocos recursos, y cada vez es más accesible para el artista y el consumidor. Pero, no es solo la distribución musical la que ha dado un cambio radical, sino que en estos últimos meses, la creación musical ha comenzado a ser también responsabilidad de las máquinas. En el artículo [1], se explican las bases de los sistemas de generación de música simbólica y la amplia gama de modelos. La mayoría de estos sistemas utilizan modelos probabilísticos de Markov, ideales para procesar series temporales y muy útiles en la generación de melodías. Estos modelos se integran fácilmente con técnicas de aprendizaje profundo. Las arquitecturas utilizadas son las CNNs y las redes neuronales recurrentes(RNN). Las CNNs son aplicadas en la extracción de características y representaciones simbólicas. Por otro lado las RNN se especializan en las tareas secuenciales y son capaces de producir modelos que generen música a través de un prompt.

Una de las herramientas más conocidas es Suno AI, una aplicación gratuita que permite experimentar con esta novedosa tecnología. Ofrece la posibilidad de añadir letra, así como definir la voz y el género musical a través del prompt. Además, funciona como una red social, donde los usuarios pueden compartir sus canciones. Algunas de ellas suman cientos de miles de reproducciones.

La arquitectura escogida para el desarrollo de la aplicación ha sido la CNN, que está orientada principalmente a tareas con imágenes, aunque se pueden usar en una amplia gama de campos, por ejemplo, procesamiento de lenguaje natural, clasificación de imágenes, análisis de series temporales o medicina, un campo donde destaca por sus buenos resultados. Como introducción a las aplicaciones de las CNN comentaré los resultados de investigaciones que hacen uso de ellas en algunos de los campos ya mencionados.

Los investigadores Sumaiya Dabeer, Maha Mohammed Khan y Saiful Islam, presentan en [2], un clasificador binario sobre un dataset con imágenes histopatológicas de tejido mamario a diferentes aumentos del microscopio. Las clases de salida del clasificador son benigno y maligno. El objetivo del estudio es proporcionar un método de clasificación con mayor precisión y eficiencia que los métodos de detección manuales, con el objetivo de intervenir antes en el tratamiento del tumor. Los resultados alcanzan una precisión

y recall de 0.95 puntos en la clase maligna. La CNN presentada tiene una arquitectura simple con 3 capas y una salida a una red totalmente conectada de 1024 neuronas.

La detección preventiva de tumores es una de las tareas donde las CNN tienen una gran relevancia, son efectivas y simples de implementar. Cada vez es más común que centros de salud implementen técnicas de reconocimiento automático.

En [3] se elabora un sistema IoT basado en 5G para la recopilación e identificación de cantos de aves usando CNNs. Este estudio resalta la importancia de las herramientas de detección automática de especies para la biología y ecología, especialmente para la conservación de especies. El objetivo del sistema es realizar un monitoreo más preciso y eficiente de las poblaciones de aves.

La CNN propuesta se caracteriza por un bajo número de parámetros y consumo, uno de los requisitos del sistema es que se pueda ejecutar en pequeños kits o placas de desarrollo. Esta característica es especialmente relevante para aplicaciones en campo, donde la disponibilidad eléctrica y la instalación de infraestructura adicional pueden ser determinantes.

Para mejorar la calidad del conjunto de datos, los investigadores han empleado el paquete Audimentations, que permite realizar transformaciones sobre los cantos de los pájaros. Incrementando la variabilidad de los datos de entrada, la red se vuelve más robusta, mejorando su versatilidad contra datos nuevos.

El rendimiento de la red propuesta es comparado con el de otras redes preentrenadas, con un mayor número de parámetros. A través de diversas métricas observamos que el rendimiento del modelo es excelente, llegando a superar a la mayoría de redes preentrenadas. El modelo presentado no solo mantienen un nivel alto de precisión, sino que además los supera en términos de consumo energético y eficiencia.

Abul Abbas Barbhuiya, Ram Kumar Karsh y Rahul Jain presentan en [4], un sistema de reconocimiento de lengua de signos usando modelos CNN pre entrenados, AlexNet y VGG16. La investigación destaca que la mayoría de los modelos hasta el momento usados se centran en gestos simples y que mediante el uso de CNNs se puede alcanzar un nuevo marco de precisión. La arquitectura de la red, usa el método de transferencia de aprendizaje en el que se usan CNNs complejas que ya han sido previamente entrenadas y se adaptan las últimas capas de la red para adaptar nuestro modelo. Este proceso reduce el coste computacional de forma muy considerable y es muy eficiente para ciertos problemas, en este caso obtenemos una precisión casi perfecta de 0.99 al usar una validación 70-30.

En [5], Zoltán Szlávik y Tamás Szirányi publican un artículo acerca del desarrollo de un algoritmo on-chip para el reconocimiento facial usando CNNs. Se hace énfasis en la importancia del análisis facial para el desarrollo de interfaces humano-máquina y el reconocimiento de expresiones faciales en tiempo real. El empaquetado de estos algoritmos en chips no se enfoca en aumentar la precisión, que ya es de por sí muy alta, si no la eficiencia. El sistema ejecuta operaciones morfológicas para extraer las características faciales primarias, la posición de nariz, ojos, boca y el eje de simetría del rostro. El algoritmo fue probado con un conjunto de datos de 20 personas en diferentes poses, desde perfil hasta frontales, la capacidad de computación del chip le permite procesar hasta 50 imágenes por segundo con una alta tasa de precisión en las características faciales mencionadas previamente. Esto demuestra la efectividad del desarrollo de hardware especializado para tareas de reconocimiento facial, y muy posiblemente de otros objetos.

2.1. Análisis de aplicaciones similares

El conjunto de datos con el que se trabajará en este proyecto es GTZAN [6], un conjunto de 1000 fragmentos de audio de 30 segundos pertenecientes a 10 géneros musicales diferentes. Es uno de los principales conjuntos de entrenamiento y prueba para aplicaciones IA para la clasificación de audio.

Son tomados como referencia los artículos publicados que realizan tareas de clasificación de audio con CNNs. La precisión de los mejores modelos con CNNs simples en GTZAN rondan el 90 % de precisión, superando ampliamente el 70 % que los humanos podemos alcanzar en el test. [7] Sin embargo, existen modelos que combinan diferentes arquitecturas de redes como XGBoost-CNN, CNN-Transformer, CNN-LSTM o simplemente arquitecturas Transformer como la Broadcast Swin Transformer (BST) [8] que obtienen resultados mejores, a cambio de un coste computacional mucho mayor. A continuación mencionare alguno de los modelos más interesantes que he encontrado. Pese a que la mayoría de trabajos coinciden en bases y finalidad, existe una gran variedad en el desarrollo del modelo, implementación aumentado de datos, diseño de arquitectura, métricas a optimizar o el procesado del audio.

En [9] se proponen 3 arquitecturas, una CNN simple, el modelo preentrenado VGG16 y un clasificador eXtreme Gradient Boosting (XGBoost). Los investigadores utilizan dos métodos para la extracción de características, espectrogramas de Mel de 3 segundos en la red preentrenada, y MFCCs de 3 segundos para XGBoost.

Los resultados obtenidos son muy interesantes, VGG16 es superado ampliamente, por los otros modelos. Es una constante en los trabajos sobre GTZAN que los modelos con transferencia de aprendizaje no alcanzan los resultados deseados. Es posible que la causa sea que son normalmente entrenadas en clasificación de objetos reales, y por ellos no capaz de descomponer la imagen correctamente debido a que estamos tratando con una serie temporal en forma de onda.

Por otra parte, los otros modelos obtienen resultados satisfactorios, la CNN consigue un 0.91 de precisión y un 0.99 en AUC(Area Under Curve) en la gráfica ROC (Receiver Operating Characteristic Curve), mientras que XGBoost alcanza 0.97 y 0.95 respectivamente.

En [10], la entrada de los datos son espectrogramas STFT, algo bastante inusual, de 3 segundos de duración extraídos de GTZAN. Se proponen dos arquitecturas de CNN: nnet1 que combina Max-Pooling y Average-Pooling, y nnet2 que incorpora un bloque residual. La combinación de capas Max y Average Pooling se realiza para proporcionar más información en las capas superiores de la red, en lugar de utilizar solo uno de los métodos. La característica más relevante del trabajo es el uso de un bloque residual en la CNN. Se agrega una conexión directa que suma las salidas de las capas convolucionales primera y tercera con el objetivo de saltarse una o más capas. Esta técnica está inspirada por el aprendizaje residual [11]. Los resultados muestran que un buen rendimiento en ambas arquitecturas. Con nnet2 alcanzando un 87,4 % y superando los resultados previos a la publicación del artículo en 2016.

En [12] se proponen varios modelos de arquitectura mixta de CNN. El modelo híbrido entre CNN y GRU obtiene una precisión del 89 %. Lo más interesante del artículo es la comparación de los datos de entrada: espectrogramas de Mel y MFCC. Todos los modelos obtienen resultados mucho mejores cuando reciben espectrogramas de Mel como entrada. El estudio no usa ningún tipo de aumento de datos sobre el audio, pero sí utiliza un

solapamiento del 50 % al generar los espectrogramas en fragmentos de 5 segundos. Como resultado cada espectrograma contiene 2,5 segundos del anterior. Al final del artículo se discute la posibilidad de trabajar con la señal de audio sin utilizar espectrogramas, esta técnica es implementada con CNN 1D. Todos los modelos mencionados utilizan CNNs 2D, ya que tienen la capacidad de procesar representaciones de audio en forma de espectrogramas, lo que les permite capturar patrones espaciales complejos y aprender características discriminatorias relacionadas con la tarea de clasificación. Las CNNs 1D pueden procesar secuencias unidimensionales, como formas de onda de audio sin procesar [13]. Aunque la información espectral y temporal contenida en los espectrogramas bidimensionales resulta mucho más relevante para la clasificación de géneros musicales, existen modelos 1D que dan buenos resultados, e incluso modelos mixtos [14].

Existe una regla común en todos los artículos que trabajan con GTZAN sobre el procesado del audio. Los audios son divididos en segmentos de pequeña duración con el objetivo de obtener más ejemplares lo que causa un impacto enorme en el funcionamiento de la CNN. La precisión usando los 30 segundos de audio como entrada es muy inferior [15]. Los estudios han demostrado que la precisión de clasificación humana alcanza su punto máximo cuando escuchan 3 segundos de música, y la precisión no aumenta a medida que la música se prolonga. En conclusión, esta técnica aumenta el número de muestras con las que se nutre la red en el entrenamiento sin acarrear una pérdida significativa de rendimiento. [16]

2.2. Tecnologías

Todo el código del proyecto está escrito en Python 3.10, que es el lenguaje de programación más usado para aplicaciones de inteligencia artificial (IA). Python cuenta con un gran soporte, comunidad, numerosas librerías y herramientas desarrolladas por terceros.

La arquitectura y entrenamiento de la red usan la librería Tensorflow. Desarrollada por Google, Tensorflow es uno los frameworks más usadas en el campo del Machine Learning. Incluye herramientas muy potentes para el desarrollo de IA. Permite diseñar y desplegar modelos de manera eficiente. Entre sus ventajas, destaca la flexibilidad con la que se puede escalar, desde móviles a grandes sistemas de producción.

Dentro de Tensorflow, se ha hecho uso de la API Keras, que facilita el desarrollo gracias a una interfaz más accesible para la construcción de redes neuronales. Keras ofrece una gran flexibilidad al mismo tiempo que una sintaxis legible y concisa.

La librería Librosa es ampliamente utilizada para el análisis y procesado de señales de audio. Ha sido usada para la extracción de características y generación de espectrogramas. Aparece en casi todos los artículos mencionados anteriormente e indispensable a la hora procesar fragmentos de audio con Python.

Audiomentations es una librería con múltiples técnicas para la transformación de señales de audio. Por ejemplo cambio de tono, velocidad, volumen, ruido gaussiano, recorte o inversión temporal. Es una herramienta indispensable para el aumento de datos en el conjunto de datos, con el fin de reducir el sobreentrenamiento y obtener mejores resultados.

Para la visualización de datos y resultados, se ha usado la biblioteca Matplotlib, que es uno de los estándares en este campo. Gracias a su facilidad de uso y profundidad.

Referencia	Librosa	Pytorch	Tensorflow	Keras	Audiomentations
[3]	Sí	No	Sí	Sí	Si
[4]	Sí	No	Sí	Sí	No
[8]	Sí	No	No	No	No
[9]	Sí	No	Sí	Sí	No
[12]	Sí	No	Sí	Sí	No
[14]	Sí	Sí	No	No	No

Cuadro 2.1: Presencia de tecnologías en los artículos citados

Capítulo 3

Requisitos, especificaciones, coste, riesgos, viabilidad

En este capítulo se describirán los aspectos esenciales para la implementación y funcionamiento exitosos del sistema. Es imprescindible realizar un análisis en base a las líneas de la gestión de proyectos, antes del comienzo del desarrollo. Para ello, se procederá a realizar un desglose que especifique los márgenes de desarrollo, costes y especificaciones del producto final.

Además, se describirán las metodologías que seguirá el proceso de desarrollo, acompañada de una planificación inicial. Debido a la naturaleza del proyecto, la planificación estará sujeta a cambios determinados por los problemas o modificaciones que puedan surgir. Por ello, se analizará una lista de riesgos que son más probables que se experimenten durante el desarrollo.

El objetivo del presente capítulo es proporcionar una visión integral del proyecto, integrando todos los aspectos relevantes con el objetivo de garantizar el éxito del proyecto y la satisfacción de los requisitos.

3.1. Requisitos

Los requisitos funcionales definen el comportamiento del sistema, funcionales y tareas que realiza. Para definir los requisitos funcionales, debemos realizar un análisis detallado de los procesos de desarrollo, así como de las necesidades y expectativas del cliente.

mientras que los no funcionales describen las características y cualidades.

3.1.1. Requisitos funcionales

- Implementar una interfaz simple e intuitiva que permita la carga de un archivo de audio por parte del usuario y que muestre el resultado de la predicción.
- El sistema debe ser capaz de transformar archivos de audio en espectrogramas utilizando la librería Librosa.
- El sistema debe soportar diferentes formatos de audio como MP3, WAV y FLAC.

- El sistema debe clasificar los géneros musicales de los espectrogramas utilizando una CNN de arquitectura propia.
- El sistema debe ser capaz de manejar errores comunes, como archivos corruptos, y mostrar mensajes de error

3.1.2. Requisitos no funcionales

- En caso de que el audio proporcionado por el usuario tenga una duración igual o menor a 30 segundos. El archivo debe ser procesado y clasificado en menos de 1 segundos.
- El diseño de la red debe ser suficientemente óptimo como para que sea ejecutado en todo tipo de dispositivos, como dispositivos móviles o placas de desarrollo.
- La precisión del sistema en la clasificación de géneros debe ser de al menos un 90
- El código del sistema debe estar dividido en módulos y bien documentado. Con el objetivo de facilitar el mantenimiento y actualización.

3.2. Costes

Se ha planificado el comienzo del desarrollo de la aplicación a inicios de noviembre y su finalización a finales de junio, en los que se repartirán 300 horas de trabajo. Para encajar con la planificación la carga diaria de trabajo asignada es de una hora y tres cuartos, sin contar los fines de semana.

Costos Temporales

- **Duración del proyecto:** Desde inicios de noviembre hasta finales de junio (aproximadamente 8 meses).
- **Total de horas planificadas:** 300 horas.
- **Horas de trabajo diarias:** 1.75 horas (una hora y tres cuartos) por día hábil.

Costos Económicos

- **Coste por hora de desarrollo:** Suponiendo un costo de 50€ por hora de desarrollo, el coste total del desarrollo sería:

$$300 \text{ horas} \times 10/\text{hora} = 3,000$$

Infraestructura y Herramientas

El equipo principal utilizado en el desarrollo ha sido un PC valorado en 1200€. Para el entrenamiento de la red neuronal se requiere una maquina con cierta capacidad de computo, sin la optimización requerida el proceso de entrenamiento se puede volver tedioso y excesivamente largo.

En concreto, el uso de una tarjeta gráfica potente de NVIDIA con acceso a la tecnología CUDA, permite acelerar el entrenamiento gracias a la paralelización de los cálculos realizados por la red. Además, la memoria RAM GDDR6 permite una mayor velocidad de comunicación una vez el conjunto de datos es cargado.

La infraestructura utilizada en el proceso de entrenamiento en proyectos de visión por computadora y aprendizaje profundo es un aspecto clave para el éxito. Con acceso a un equipo con capacidad de computo la realización de prueba y testeo aumenta las posibilidades de que el rendimiento del sistema sea el máximo posible.

Componente	Especificación
Procesador	AMD Ryzen 7 5800X
Placa Base	ASUS ROG Strix X570-E Gaming
Memoria RAM	Corsair Vengeance RGB Pro 32 GB (2 x 16 GB) DDR4-3600 MHz
Tarjeta Gráfica	NVIDIA GeForce RTX 3080 Founders Edition
Almacenamiento SSD	Samsung 970 Evo Plus 1 TB NVMe M.2
Disco Duro Adicional	Seagate Barracuda 2 TB 7200 RPM
Fuente de Alimentación	Corsair RM850x 850W 80+ Gold
Enfriamiento CPU	Cooler Master Hyper 212 RGB Black Edition
Caja/Torre	NZXT H510 Elite
Sistema Operativo	Windows 10 Pro 64-bit

Costos Detallados

Concepto	Costo (€)
Desarrollo (300 horas)	3,000
Infraestructura (PC)	1,200
Documentación y Mantenimiento	500
Total Estimado	4,700

Cuadro 3.1: Desglose de costos del proyecto en euros

Concepto	Costo Mensual
Vivienda	200 €
Luz	30 €
Agua	25 €
Internet	6 €

Cuadro 3.2: Costes mensuales asociados a la vivienda

3.3. Riesgos

En el contexto del análisis y clasificación de géneros musicales, debemos considerar diferentes eventos que podrían influir negativamente en el rendimiento y entrenamiento del modelo.

Los riesgos planteados son:

- Variabilidad en la interpretación de características acústicas

- Complejidad y diversidad de subgéneros en las canciones presentadas.
- Insuficiente cantidad de datos de entrenamiento.
- Sensibilidad a la calidad del audio de entrada

3.4. Viabilidad

La razón principal por la que consideramos el proyecto como viable es que el uso de las CNNs para la clasificación de géneros musicales es una técnica que ya ha demostrado tener un buen rendimiento previamente. Además, GTZAN ha sido sujeto de estudio de innumerables y diferentes estudio en tareas de audio con aprendizaje automático. Por lo que la calidad del conjunto de datos es un hecho.

Por otro lado, la viabilidad técnica recae sobre la disponibilidad del hardware, se trata de una pieza fundamental para el éxito del desarrollo. En caso de falla del equipo principal, la sustitución por una maquina de menor potencia puede dificultar la optimización del diseño de la red.

Otra razón para confiar en la viabilidad del sistema aquí presentado, es la aplicación práctica y adaptabilidad. Una vez completado el diseño de la red, existe la posibilidad de exportar el modelo e implementarlo en diferentes aplicaciones de forma sencilla. Por otra parte, en caso de recibir una petición personalizada de un clasificador con géneros específicos, una opción a considerar sería el reentreno de la red con un nuevo conjunto de datos.

DAFO

DAFO hace de acrónimo para las palabras Debilidades, Amenazas y Oportunidades. Es un diagrama de análisis para la evaluación de un proyecto o idea, con el objetivo de identificar los riesgos y cualidades. El uso de un análisis DAFO es determinante a la hora de definir la planificación, maximizando las fortalezas y oportunidades, mientras minimizamos las amenazas y debilidades.

Es una herramienta versátil que debe ser revisada durante diferentes puntos del desarrollo y es que realizar un seguimiento constante durante el periodo de desarrollo, puede ser de gran utilidad para la adaptación y ajuste de los requisitos y especificaciones del sistema.

3.5. Planificación

El proyecto ha sido planificado utilizando la metodología Scrum, un marco de trabajo ágil que facilita la gestión de proyectos gracias a la división en fases o sprints. Cada fase comienza o termina con una reunión, en las que se valoran y reajustan los progresos realizados. Durante las reuniones se planifica el siguiente sprint, asegurando un desarrollo iterativo que mejora de forma continua.

Al comienzo del proyecto, se estableció un cronograma detallada con un diagrama de Gantt, este diagrama aparece en la figura 3.2, en el que podemos apreciar los marcos

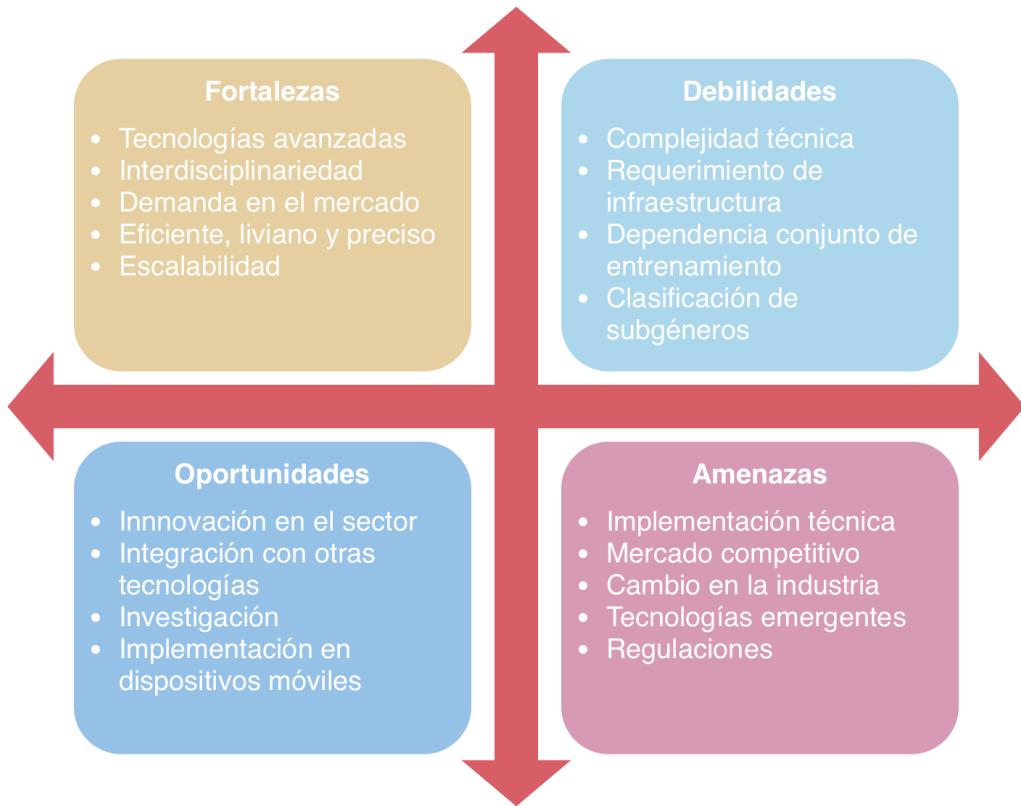


Figura 3.1: DAFO



Figura 3.2: Diagrama de Gantt antiguo

temporales de las fases del proyecto. Sin embargo, el proyecto sufrió un reajuste en las fechas de presentación y fue necesario hacer modificaciones.

En la nueva planificación, mostrada en 3.3, se cambian algunas fases del desarrollo que ya no eran necesarias y se actualiza la fecha de presentación. También se muestra el historial de reuniones con el tutor, aunque la comunicación continua a través de mensajería ha sido clave para ajustar la planificación y resolver cualquier imprevisto.

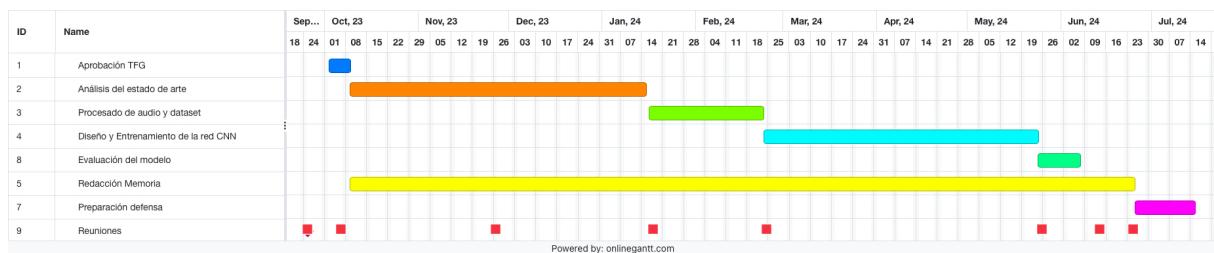


Figura 3.3: Diagrama de Gantt nuevo

Capítulo 4

Análisis

4.1. Onda Sonora

Las ondas sonoras son fenómenos físicos que desempeñan un papel fundamental en la propagación del sonido. Estas ondas, también conocidas como ondas de presión, se propagan a través de medios como el aire, el agua o los sólidos.^[17] Cuando un objeto vibra, por ejemplo, las cuerdas vocales, este transmite la vibración al medio o medios que tiene cerca, produciendo variaciones de ondas que viajan hasta que se extinguen, pudiendo alcanzar nuestro sistema auditivo por el camino. Las principales características de una onda sonora son: amplitud, frecuencia, longitud de onda y velocidad de propagación.^[18]

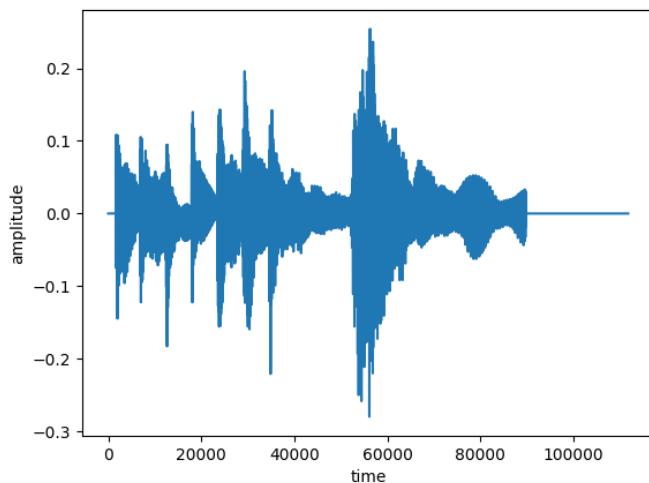


Figura 4.1: Esquema de una onda sonora

4.1.1. Características de la onda sonora

- **Amplitud:** Es el máximo desplazamiento que experimenta un punto de una onda en contraposición con su estado de equilibrio. Este concepto va ligado a la cantidad de energía que puede llevar la onda, pues si incrementa la energía, incrementa la amplitud de onda, y lo mismo ocurre a la inversa.^[19]

- **Frecuencia:** La frecuencia de onda es el número de ondas o ciclos completos que se producen en un segundo y se mide en Hertz. Podemos decir que la frecuencia es el número de vibraciones por segundo que una onda completa es producida por una vibración completa del cuerpo vibratorio. La frecuencia está relacionada con la altura tonal del sonido. A frecuencias más altas los sonidos serán más agudos, mientras que a frecuencias más bajas serán más graves.
- **Longitud de onda:** Es la distancia entre las dos crestas de una onda y se relaciona inversamente con la frecuencia.
- **Velocidad de propagación:** Es la velocidad de la onda a través del medio en el que se desplaza. Está influenciada por diferentes factores físicos como la temperatura, presión y densidad del medio en el que se desplaza.

4.1.2. Propiedades musicales del sonido

Son una percepción subjetiva de las características físicas de la onda sonora, es decir, son un puente entre la física y la experiencia subjetiva de nuestro sistema auditivo. Son la base fundamental de la teoría musical.

Tono: El tono o altura varía principalmente según la frecuencia, es decir, la cantidad de vibraciones que tenga la onda sonora, y nos permite diferenciar los sonidos graves (o bajos, de poca frecuencia) de los sonidos agudos (o altos, de mucha frecuencia). En música, las notas (Do, Re, Mi, Fa, Sol, La, Si) y las escalas determinan el tono.

Duración: La duración que percibimos de un sonido se define como ese intervalo temporal durante el cual persiste sin discontinuidad.

Armonía: Los armónicos son componentes de un sonido que se producen a frecuencias múltiples de la frecuencia fundamental. Los armónicos contribuyen al timbre o color de un sonido y son importantes en la música y la acústica.

Timbre: Otro elemento básico del sonido es el color o timbre. Es la característica por la que podemos distinguir cada sonido o instrumento en el caso de la música. Depende de la envolvente y la frecuencia en la mayoría de los casos. Las frecuencias secundarias que participan en el sonido se denominan armónicos.[\[20\]](#)

4.2. Escalas

Para representar las ondas sonoras de una forma similar a nuestro oído nos valemos de escalas psicoacústicas. El espectro audible por los seres humanos está formado por la gama de frecuencias entre 20Hz y 20000Hz. Además, este rango no es percibido de manera lineal por nuestro oído, los tonos bajos se perciben con mayor precisión que los altos. Las escalas comprimen el rango para adaptarlo a nuestra percepción.[\[21\]](#).

Son herramientas fundamentales en el procesamiento de sonido y la extracción de características, ya que tienen en cuenta la musicalidad. Dependiendo de la escala seleccionada, este proceso se hace de forma diferente, usando una u otra según las características de cada método.[\[22\]](#)

■ Escala de Mel

Propuesta en 1937 por Stevens, Volkmann y Newman. Se asigna un valor en mels a cada frecuencia en Hz, de tal manera que los tonos se distribuyen de manera equidistante. Es decir que dos tonos separados por la misma cantidad de mels serán igualmente espaciados aunque sus frecuencias no lo sean.[\[23\]](#)

El punto de referencia entre esta escala y la frecuencia normal se define equiparando un tono de 1000 Hz, 40 dB por encima del umbral de audición del oyente, con un tono de 1000 mels. Por encima de 500 Hz, los intervalos de frecuencia espaciados exponencialmente son percibidos como si estuvieran espaciados linealmente. En consecuencia, cuatro octavas en la escala de hercios por encima de 500 Hz se comprimen a alrededor de dos octavas en la escala Mel.

Formula para convertir frecuencia en Hertz (Hz) a escala Mel (mels):

$$m = 1127,01048 \cdot \log \left(\frac{1+f}{700} \right) \quad (4.1)$$

La escala Mel distingue con precisión las frecuencias bajas del sonido, que son las más importantes en la percepción del habla. Este concepto está definido por la Ley de Weber-Fechner, también conocida como Ley de la Percepción Sensorial, establece una relación cuantitativa entre la magnitud de un estímulo físico y la magnitud de lo que se percibe por los sentidos. Esta ley nos ayuda a entender cómo nuestro cerebro interpreta los cambios en la intensidad de los estímulos que percibimos a través de nuestros sentidos. Las vibraciones producidas por las cuerdas vocales en garganta y boca pertenecen a las frecuencias bajas del espectro de sonido.

Gracias a sus características enfocadas en la percepción del sistema auditivo es especialmente útil en tareas relacionadas con el reconocimiento de voz y codificación de audio.

Su aplicación es sencilla y su uso esta ampliamente extendido.

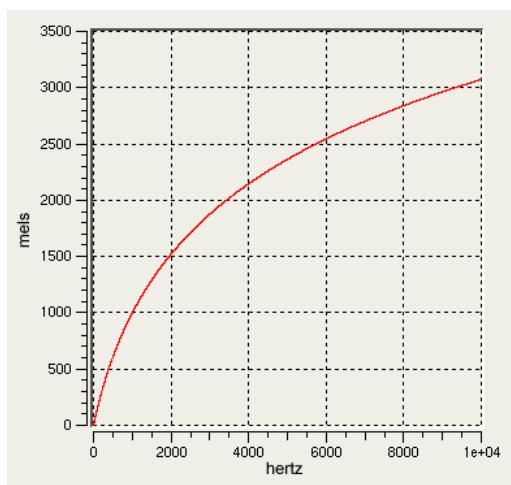


Figura 4.2: Escala de Mel

■ Escala de Bark.

Propuesta por Eberhard Zwicker en 1961. Se basa en el concepto de bandas críticas del oído humano. Las bandas críticas representan el ancho de banda de frecuencia

del "filtro auditivo" producido por la cóclea, el órgano del sentido auditivo dentro del oído interno. La banda crítica es aproximadamente la banda de frecuencias audibles en la que un segundo tono interferirá con la percepción del primer tono a través del enmascaramiento sonoro. [24] La escala tiene un rango de 1 a 24, y a cada intervalo se le asigna un valor específico. La formula para pasar de frecuencia en Hertz(Hz) a escala Bark:

La escala de Bark tiene aplicaciones similares a la escala Mel. Es menos usada debido a una mayor complejidad y un bajo uso histórico, pese a que se considera más precisa en la representación de las frecuencias bajas.

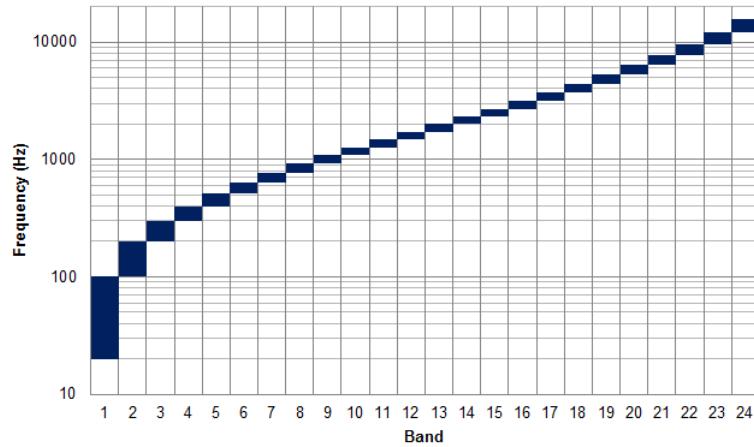


Figura 4.3: Escala de Bark

Escala ERB

Desarrollada por Brian C. J. Moore y Brian R. Glasberg en 1983. La Escala de Ancho de Banda Rectangular Equivalente (ERB) es una herramienta para medir el ancho de banda de los filtros en el oído humano. Se basa en la noción de que los filtros auditivos pueden modelarse en forma de pasabanda rectangular.[25] Trabaja sobre el mismo concepto de ancho de banda que la escala Bark.

El filtro auditivo, la frecuencia y el ancho de banda crucial están conectados por el concepto de ERB. Esta conexión explica cómo el ancho de banda de un filtro rectangular equivalente está relacionado con la frecuencia y la posición del filtro auditivo a lo largo de la membrana basilar del oído interno.[26]

Formula para pasar de frecuencia en Hertz(Hz) a escala ERB:

$$ERB = 21,4 \cdot \log_{10}(0,00437 \cdot f + 1) \quad (4.2)$$

Al igual que la escala de Bark, el cálculo de la escala ERB es más complejo que la escala Mel. No obstante su uso se ha popularizado en software de alta gama, como el desarrollo de códecs de alta fidelidad [27] o su uso en compañías de desarrollo hardware y software de audio como Dolby Laboratories.[28]

4.3. Extracción de características

El procesamiento de señales de audio es fundamental en el desarrollo de aplicaciones relacionadas con el análisis y detección de audio. Las técnicas más utilizadas y que me-

jores resultados ofrecen en el Estado del Arte son los coeficientes cepstrales de frecuencia Mel (MFCC) y los espectrogramas Mel. Ambas técnicas proceden de la aplicación de la transformada de Fourier de Tiempo Corto (Short-time Fourier transform, STFT), un algoritmo que permite calcular la transformada de Fourier discreta (DFT) y su inversa de manera más eficiente.

4.3.1. La transformada de Fourier

Es una herramienta matemática fundamental en el procesamiento de señales que sirve para descomponer una señal en sus componentes frecuenciales. Al aplicar la transformada de Fourier a una señal de audio, obtenemos su representación en el dominio de la frecuencia, con el objetivo de identificar las diferentes frecuencias que la componen. A partir de aquí podemos representar las señales de diferentes formas, siendo la más común los espectrogramas.[?]

El oido actúa de forma similar sobre las ondas sonoras, ya que descompone las diferentes frecuencias en el producto que finalmente escuchamos.

La STFT es un algoritmo capaz de calcular la Transformada Discreta de Fourier y su inversa, con un menor costo computacional. La complejidad de la Transformada Discreta de Fourier original para N puntos es $O(N^2)$ operaciones frente a la $O(N \log N)$ operaciones de STFT, es decir, una mejora muy significativa que permitió la incorporación de FFT en algoritmos informáticos.

MFCC(Mel-Frequency Cepstral Coefficients)

Son el resultado de aplicar la transformada del coseno del logaritmo real del espectro de energía a corto plazo, expresado en una escala de frecuencia Mel [29]. Son el resultado de un análisis cepstral.

El cálculo de las características MFCC se realiza siguiendo estos pasos [30]:

1. División del audio en segmentos superpuestos de igual longitud, normalmente de 10 a 50 ms. Estos segmentos de audio son llamados marcos.
2. Para cada marco se genera una representación de la potencia de la señal en diferentes frecuencias llamada periodograma. El periodograma es calculado con la transformada rápida de Fourier.
3. Transformación a escala de Mel, para enfatizar las características más importantes en el reconocimiento del habla.
4. Se aplican una serie de filtros de frecuencia Mel al periodograma, calculando la suma de la potencia para cada filtro. Obtenemos una representación de la energía para cada banda de Mel.
5. Compresión logarítmica sobre los filtros Mel. [31]
6. Finalmente, se aplica la transformada discreta del coseno (DCT) sobre energías del banco de filtros Mel con el objetivo de reducir la redundancia en los datos y devolver los MFCC.

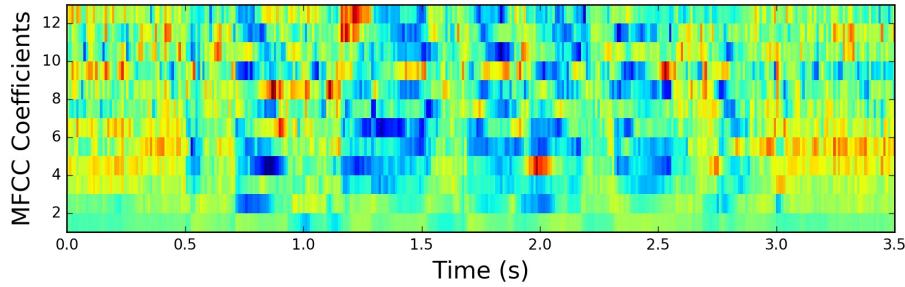


Figura 4.4: Representación de MFCC

La transformada del coseno es una variación de la transformada de Fourier que usa solamente números reales. Consigue concentrar la mayor parte de la información en pocos coeficientes transformados.[\[32\]](#)

Existen múltiples variaciones en el cálculo de los MFCC, debido a su gran cantidad de parámetros. En [\[33\]](#) se exploran diferentes implementaciones de MFCC según el número y el tipo de filtros e incluso el uso de la escala de Bark en vez de la escala de Mel. La conclusión es que la hiperparametrización de los MFCC depende en gran medida del conjunto de datos con el que estamos utilizando.

Espectrograma de Mel

Los espectrogramas de Mel son la representación visual de sonidos en escala Mel. El proceso de generación es más sencillo que el de MFCC. Los pasos son exactamente iguales, la diferencia es que no necesitamos aplicar la discreta del coseno. Tras aplicar la compresión logarítmica sobre los filtros Mel obtenemos una matriz que representa la energía en cada banda Mel a lo largo del tiempo, solo tenemos que convertir la matriz a un espectrograma con una biblioteca de visualización.

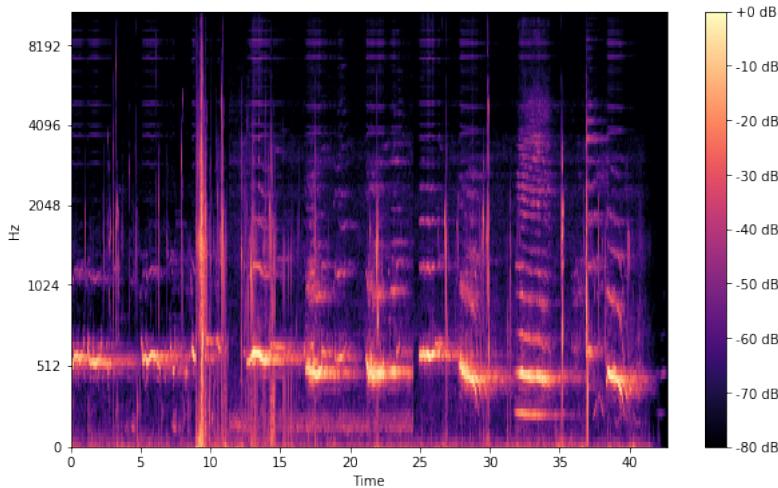


Figura 4.5: Espectrograma de Mel

Capítulo 5

Diseño

5.1. Aprendizaje Automático

Se trata de la rama de las ciencias de la computación que tiene como objetivo la replicación de inteligencia a partir de la experiencia, la cual puede tomar diferentes formas, como grandes conjuntos de datos etiquetados o interacciones con el entorno.

Buscamos que la máquina aprenda de la información proporcionada e intente replicar los conceptos asimilados en forma de predicciones, sin recurrir a una programación explícita de cada tarea. Con un entrenamiento adecuado es posible crear modelos con la capacidad de hacer predicciones con una precisión igual o mejor que la humana.^[34] Esto es posible gracias a una amplia variedad de algoritmos y modelos matemáticos que son capaces de identificar patrones y relaciones en los datos.^[35]

5.1.1. Aprendizaje no supervisado

Se centra en el análisis de la estructura del conjunto de datos no etiquetado. El objetivo es encontrar relaciones intrínsecas en los datos para realizar diferentes tareas.

- **Agrupamientos:** Consiste en agrupar elementos de un conjunto según sus características. Algunos algoritmos populares son k-medias, k-vecinos, DBSCAN u OPTICS.
- **Reducción de dimensionalidad:** Se busca reducir el número de características del conjunto de datos, tratando de conservar la mayor cantidad de información posible. Normalmente se hace con el objetivo de visualizar los datos en dos o tres dimensiones. Algunas técnicas populares son PCA y t-SNE.

5.1.2. Aprendizaje supervisado

Se caracteriza por el uso de conjuntos etiquetados como entrada, donde el valor a predecir ya está definido. A partir de estos datos se modela una función que predice la clase o el valor deseado sobre datos desconocidos. Este tipo de aprendizaje es útil en tareas de clasificación, donde se predice la pertenencia del sujeto a una clase, como la propuesta en este trabajo. También es útil en tareas de regresión, en las que se calcula

un valor numérico, como por ejemplo, calcular el precio de una vivienda en base a sus características.^[36]

5.2. Neurona Artificial

El concepto de neurona artificial fue introducido en 1943 por McCulloh y Pitts^[37]. Inspirados por el funcionamiento de la neurona propusieron un modelo matemático que intentaba replicar el comportamiento del sistema nervioso en base a una serie de consideraciones biológicas expresadas en lenguaje matemático. Pese a que McCulloh y Pitts pretendían revolucionar el campo neurofisiología, la investigación acabó teniendo una relevancia e impacto mucho mayor en las ciencias de la computación.^[38]

5.2.1. Estructura de la neurona

- **Pesos:** Son los parámetros que cuantifican las conexiones entre las neuronas en una red neuronal. Aquí es donde se almacena la mayor parte del "aprendizaje" de la red.
- **Función de activación:** Esta función computa la salida de la neurona basándose en la suma ponderada de los pesos y el bias. Introducen la no linealidad esencial. La elección de una función de activación sobre otra depende de las particularidades del problema y juega un papel crucial en la precisión del modelo.
- **Bias:** Es un valor aditivo que permite que la red se ajuste de una manera más refinada y compleja, proporcionando flexibilidad adicional al modelo.
- **Función de pérdida:** Esta función evalúa qué tan bien los valores de bias y pesos minimizan el error entre la salida actual y la esperada durante la fase de entrenamiento, ajustando los parámetros en consecuencia.

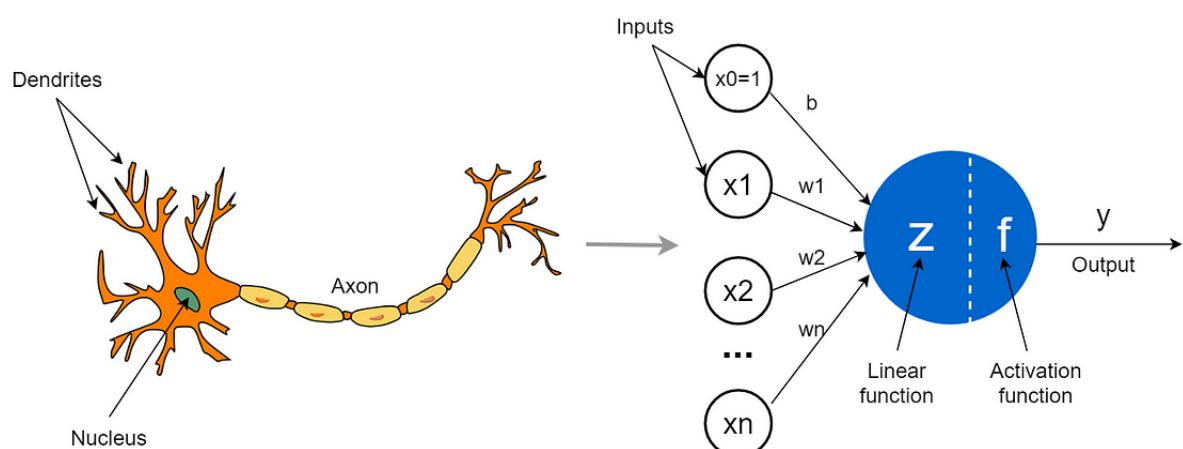


Figura 5.1: Operación de Max Pooling y Average Pooling

5.3. Perceptrón

Frank Rosenbatt impulsado por los descubrimientos de sus coetáneos, realizó la primera implementación del perceptrón en 1957, uno de los primeros modelos de aprendizaje automático. Se trata de un modelo primitivo de red neuronal artificial con una sola capa, por lo que su uso solo era adecuado para problemas linealmente separables.

La publicación del modelo causó inicialmente un gran revuelo en el campo de la ciencia de computación, pero se provocó un estancamiento en la investigación hasta la década de los ochenta, cuando se publicaron el algoritmo de retropropagación y el perceptrón multicapa, que permitieron abordar problemas no lineales.^[39]

Desde ese momento, la investigación y uso de redes neuronales no ha parado de crecer. Hasta el punto que cada pocos años se propone una nueva arquitectura que permite ampliar el abanico de problemas a los que las redes neuronales son aplicables. Este avance está acompañado de un desarrollo exponencial en el desarrollo de unidades de procesamiento adecuadas para la IA, las GPU y las TPUs, fundamentales para el entrenamiento y la ejecución de modelos de gran tamaño.^[40]

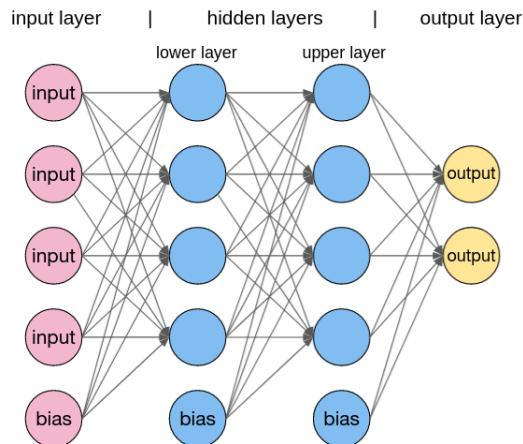


Figura 5.2: Esquema de un perceptrón multicapa

5.3.1. Funcionamiento de la red

Un perceptrón multicapa está formado por tres tipos diferentes de capas:

- **Capa de entrada:** Es la primera capa de la red y su función consiste en recibir los patrones de entrada y propagarlos a la capa posterior.
- **Capa oculta:** Reciben información de una capa anterior, la procesan usando un método asignado, y finalmente la transmiten a una capa posterior.
- **Capa de salida:** Neuronas con los valores de salida de la red.

La principal tarea al diseñar una red es averiguar el número de capas ocultas y neuronas. Es labor del diseñador averiguar qué parámetros se ajustan mejor a la arquitectura. En la mayoría de los casos esto se hace mediante prueba y error, pese a que podemos orientarnos según las características del problema.

La naturaleza oculta de los cálculos llevados a cabo por la red es comparada a una caja negra. Es prácticamente imposible extraer las relaciones entre los pesos. Esto sería útil para obtener un mejor conocimiento sobre el funcionamiento de la arquitectura.

La elaboración de métodos que optimicen estos parámetros se encuentra bajo investigación, se caracterizan por ser algoritmos evolutivos, como los algoritmos genéticos. Sin embargo su consumo computacional es muy intenso por lo que su aplicación no está extendida.

Normalmente, todas las neuronas de una capa están conectadas con las neuronas de la capa siguiente. Entonces, se dice que la red está totalmente conectada.[\[41\]](#)

5.3.2. Función de activación

La función de activación realiza una transformación sobre la salida de la suma ponderada de los pesos, ajustándola a un rango que facilite el manejo de los datos y su entrenamiento. Históricamente, la función sigmoide ha sido la más utilizada, pero se ha demostrado que la Función Unidad Rectificada Uniforme (ReLU) es más eficiente.[\[42\]](#)

ReLU

La RELU proporciona una serie de ventajas sobre el resto de funciones de activación. Su coste computacional es sencillo, lo que ayuda a reducir el tiempo de entrenamiento.

El algoritmo de retropropagación funciona mejor debido a la reducción de casos de desvanecimiento de gradiente. Por último, el número de activaciones es escaso comparado con otras funciones que tienden a la sobre saturación.

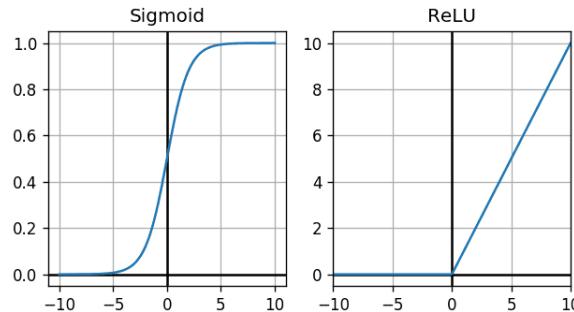


Figura 5.3: Funciones de activación RELU y Sigmoidal

5.4. Tipos de redes neuronales

El campo de aplicación de las redes neuronales es amplio, y nuevas arquitecturas se publican cada cierto tiempo, cada uno de ellas responde a un tipo problema diferente, por ejemplo, series temporales, imágenes, texto o audio. [\[43\]](#)

- **Redes neuronales artificiales (ANNs):** Se basan en el modelo del perceptrón multicapa. Cada neurona está conectada a todas las neuronas de la capa siguiente. Son versátiles y pueden adaptarse a una amplia rama de problemas, como regresión

y clasificación. En problemas donde las relaciones entre los datos son complejas y no lineales, son particularmente efectivas.

- **Redes neuronales recurrentes (RNNs):** A diferencia de otros modelos como las ANNs, una RNN tiene conexiones que forman ciclos, lo que le permite almacenar información de entradas anteriores. Esta característica las hace ideales para trabajar con secuencias de datos, como series temporales o procesado de lenguaje natural.
- **Redes Generativas Antagónicas (GANs):** Tienen una arquitectura diferente a las ya mencionadas, están formadas por 2 redes: una generadora y otra discriminadora. La generadora produce datos que tratan de imitar los datos reales, mientras que la discriminadora trata de distinguir entre los datos generados y los reales. Las GANs brillan en el campo de generación y mejora de imágenes sintéticas o modelos 3D, donde brillan por su rapidez.
- **Redes neuronales convolucionales (CNNs):** Son un algoritmo de aprendizaje automático orientado a imágenes. Si las redes neuronales artificiales tratan de imitar el comportamiento del sistema nervioso y la comunicación neuronal, las CNNs se centran específicamente en el comportamiento del córtex visual.

5.5. Inspiración biológica

Los trabajos de los doctores Hubel y Wiesel en los años 50 y 60, proporcionaron mucha información sobre el desarrollo y plasticidad del cerebro, centrándose en el córtex visual.^[44] ^[45] Realizaron experimentos con gatos, donde registraron las respuestas nerviosas provocadas por estímulos visuales. Encontraron que determinadas áreas se activaban en respuesta a bordes y contornos, este descubrimiento fue el primer indicio de como el cerebro procesa la información visual.

El trabajo de los investigadores fue recompensado con el premio Nobel de Medicina en 1981, por sentar las bases del comportamiento del cerebro durante la recepción sensorial. Esta investigación allanó el camino para el posterior desarrollo de las CNNs. Las CNNs están diseñadas para emular la estructura jerárquica del córtex visual. Cada capa se encarga de reconocer ciertos patrones concretos. Las primeras capas detectan características simples como bordes, mientras que capas más profundas reconocerán patrones más complejos y específicos.

5.5.1. Áreas de la corteza visual

A continuación, se exponen las diferentes áreas en las que se subdivide el córtex visual. Cada una es el encargado de una etapa en el procesado de la información visual. La identificación de cada una de estas áreas llevaron a la propuesta de un modelo en cascada para el reconocimiento de patrones, que eventualmente influiría en el diseño de las CNNs.

- **Corteza Visual Primaria (V1):** Recoge la información captada por los ojos y se encarga de preservar la ubicación espacial, es decir, la orientación de bordes y líneas.

- **Corteza Visual Secundaria (V2)**: Se retroalimenta de las conexiones con V1 y recopila información sobre la frecuencia espacial, el tamaño, el color y la forma del objeto. Esta información es compartida con V3, V4, V5.[46]
- **Tercera Corteza Visual (V3)**: Recibe entradas de V2, ayuda en el procesamiento del movimiento global y proporciona una representación visual completa.
- **Cuarta Corteza Visual (V4)**: También recibe entradas de V2. Reconoce formas geométricas simples y también forma el reconocimiento de objetos. No está sintonizada para objetos complejos como los rostros humanos.[47]
- **Área Visual Temporal Mediana (MT o V5)**: Su función es la percepción del movimiento, detecta la velocidad y dirección que toma un objeto en movimiento. También detecta el movimiento de características visuales complejas. Recibe conexiones directas de V1 y V2.
- **Área Dorsomedial (DM o V6)**: Al igual que V5, también recibe conexiones directas de V1. Se utiliza para detectar un amplio campo visual y para la estimulación del movimiento propio. Tiene una selección extremadamente nítida de la orientación de los contornos visuales.

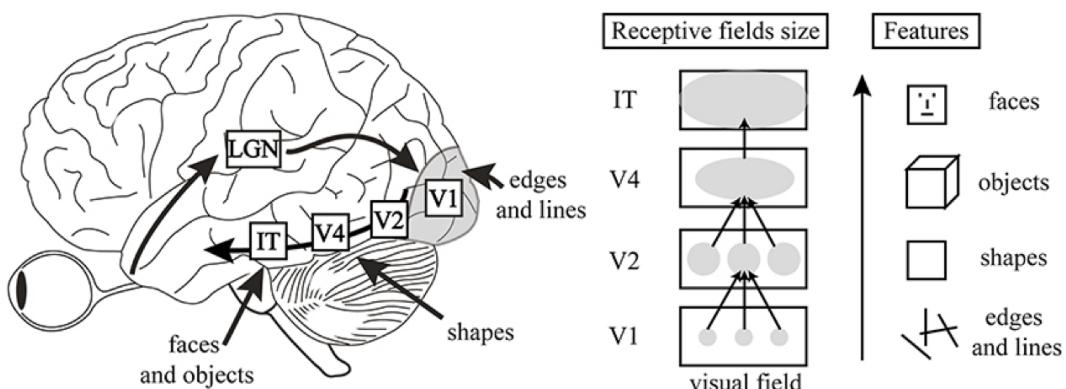


Figura 5.4: Representación del procesado de imágenes por el córtex visual

Las CNNs realizan un proceso en cascada, las primeras capas identifican elementos básicos de la imagen como bordes y orientación, de forma similar a V1. Las capas siguientes reconocen patrones más complejos como texturas y formas (V2). El reconocimiento de características de alto nivel se realiza las últimas capas convolucionales (V3) y el reconocimiento de objetos mediante capas densas conectadas (V4 y V5). V6 estaría relacionada con el reconocimiento de elementos en vídeo.[48]

5.6. Arquitectura

Las redes convolucionales intentan representar características de alto nivel en un espacio dimensional reducido, siguiendo una estructura en capas similar a la del córtex visual. Son capaces de reducir en varias ordenes de magnitud el numero de parámetros que serian necesarios en un perceptrón multicapa, es por esta razón que son usadas principalmente

en el campo de la visión por computador, donde la entrada de la red es una imagen está formada por un gran número de píxeles, cada uno de ellos es un parámetro de entrada.[49]

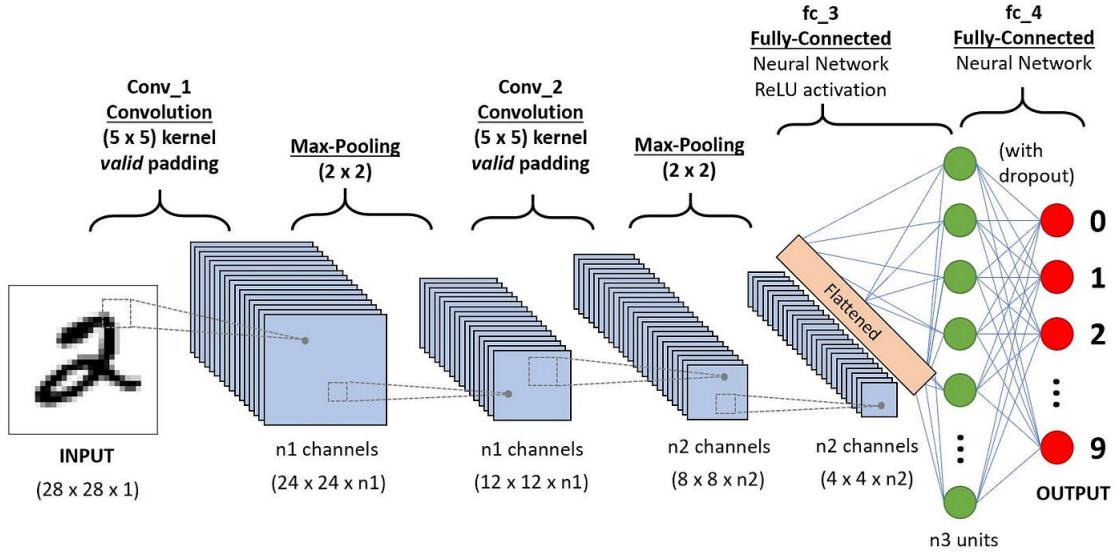


Figura 5.5: Esquema de una CNN

5.6.1. Componentes de una CNN

- **Capa convolutoria:** Es el elemento característico de la red. Está compuesta por un conjunto de filtros convolucionales llamados kernels. Estos filtros aplican la operación de convolución sobre la entrada y devuelven un mapa de características que captan diferentes aspectos de la imagen, como bordes, texturas o patrones específicos, dependiendo de los filtros utilizados. [50]
- **Kernel:** Un kernel es una matriz de números en el que cada valor representa un peso. La dimensión del filtro normalmente tiene un tamaño reduciendo como 2x2, 3x3 o 5x5, que permite simplificar las operaciones de la red. Sin embargo existen modelos con filtros de gran tamaño que han demostrado ser eficientes también.[51] Existen diferentes métodos para establecer los valores del kernel al principio del entrenamiento. Una de las técnicas más comunes es simplemente inicializar los valores de forma aleatoria[52].
- **Capa Pooling:** En todas las CNN, buscamos ir reduciendo poco a poco el tamaño de los mapas de características de modo que el número de operaciones escala a un número demasiado grande mientras que la red extrae la información de la entrada. Las capas de Pooling son una herramienta para reducir la dimensionalidad, intentando reducir al mínimo la perdida de información. Se aplica en forma de filtro con determinado tamaño, comúnmente, 2x2 y con un stride de 2. La salida de aplicar esta operación de pooling sobre una matriz de 28x28(784 pixels) sería 14x14(196 pixels). Al pasar el mapa reducido a la siguiente capa convolutoria, se consigue disminuir el número de cálculos que ésta debe realizar significativamente. Esta propiedad convierte a las capas Pooling en herramientas muy poderosas en modelos de grandes dimensiones donde reducir el tiempo de entrenamiento es crucial.

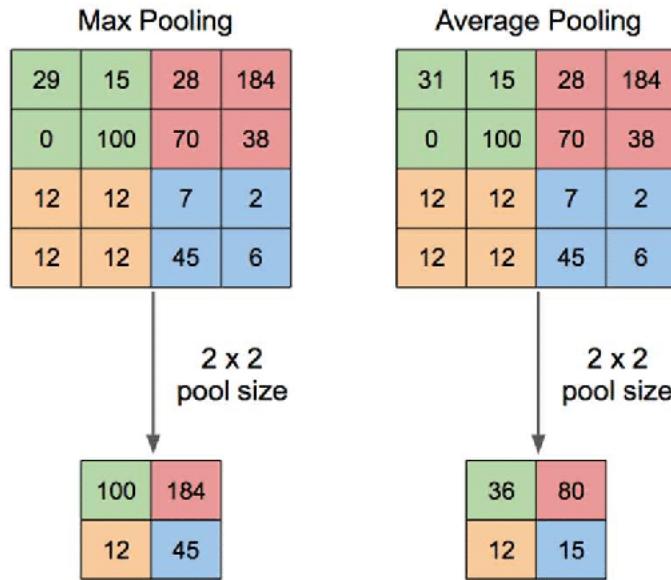


Figura 5.6: Operación de Max Pooling y Average Pooling

Existen diferentes métodos para calcular la operación de pooling, los más usados son:

- **Max Pooling:** El filtro selecciona el máximo local como representante de cada región. Este enfoque resalta las características más prominentes de cada área, permitiendo una mejor visualización y análisis de los datos.
- **Average Pooling:** Se calcula el promedio de los valores de los elementos. Esto puede ayudar a suavizar la representación y a reducir el ruido en los datos.

Ambos métodos tienen una eficacia similar, y son los más usados debido a la simplicidad de su implementación. Otros métodos más sofisticados no han demostrado tener un rendimiento suficientemente mayor como para reemplazarlos.^[53]

- **Capa totalmente conectada** Reciben como entrada un vector unidimensional, resultado de aplanar los mapas de características que da la última capa convolucional. Son responsables de la salida final de la red.

La cantidad de filtros usados es el parámetro más importante de la capa convolucional, de él dependen el número de parámetros a entrenar y el número de operaciones. A cada uno de los mapas de características resultantes se le aplica una función de activación, y se envían a la capa siguiente, que volverá a aplicar un filtro convolucional o otro tipo de operación.

Una de las propiedades más importantes de la capa convolucional es la compartición de pesos, por lo que la red solo tiene que aprender los pesos de los filtros. Esto permite una reducción en el número de operaciones a realizar respecto a una capa totalmente conectada y es clave para la eficacia de la arquitectura.

5.6.2. Operación de convolución

Es una operación que se utiliza en las redes convolucionales para extraer características de los datos de entrada. Esta operación se realiza mediante el deslizamiento del filtro

convolucional sobre la entrada y calculando los productos escalares en cada ubicación generando un mapa de características.

El proceso de convolución comienza colocando el filtro en la esquina superior izquierda de la entrada. El filtro se desliza (o convoluciona) sobre la entrada moviéndose en pasos definidos por una variable que llamamos 'stride'.

En cada posición donde se superpone el filtro con la entrada, se realiza una multiplicación punto a punto entre los elementos del filtro y los elementos correspondientes de la entrada. Después de realizar las multiplicaciones punto a punto, se suman los resultados de estas operaciones para obtener un valor único en cada ubicación donde se ha aplicado el filtro. Esto se conoce como el producto escalar entre el filtro y la región correspondiente de la entrada.

Es importante mencionar que la mayoría de implementaciones de la capas convolucionales reducen el costo computacional mediante el uso de la transformada rápida de Fourier, gracias a la propiedad de transformar la convolución que en principio pertenece al dominio temporal en una multiplicación en el dominio frecuencial. En lugar de realizar la convolución directamente en el dominio espacial, como se describe en el proceso anterior, se transforman la señal de entrada y el filtro al dominio de la frecuencia. Luego, se realiza la multiplicación de estas señales y se aplica la Transformada Inversa de Fourier (IFFT) para obtener el resultado en el dominio espacial nuevamente. De esta manera se reduce el coste de $O(N^2)$ a $O(N \log N)$.

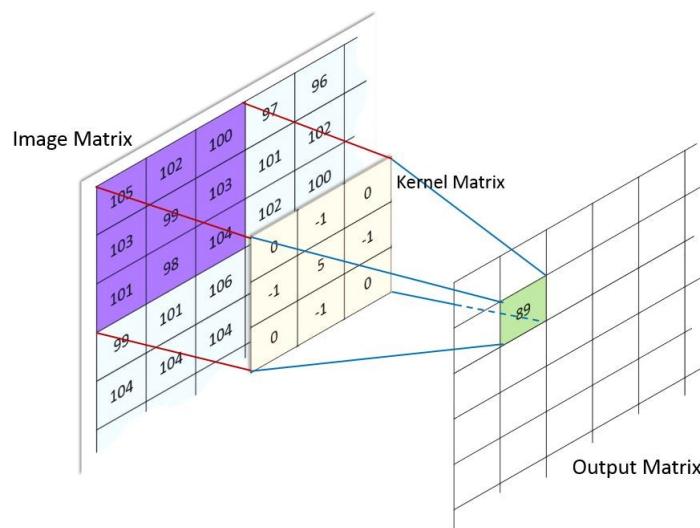


Figura 5.7: Operación de convolución

5.7. Aprendizaje

El aprendizaje de la CNN se basa en la optimización iterativa de los parámetros mediante un algoritmo conocido como retropropagación. Su objetivo es minimizar la diferencia entre las etiquetas del conjunto de datos y la salida de la red durante el proceso de entrenamiento, ajustando los pesos y sesgos de la red.

En una CNN, el proceso de entrenamiento se centra en el cálculo de los filtros en las capas convolucionales. Durante la retropropagación, se calculan los gradientes de la función de pérdida respecto a los valores de los filtros. En las capas totalmente conectadas,

se hace el mismo proceso pero sobre los sesgos y pesos en vez de sobre los filtros.

La retropropagación tiene 2 fases:

- **Propagación hacia adelante:** Los datos de entrada avanzan a través de la red neuronal, pasando por los kernels, capas de activación y pooling, luego se aplana para pasar a las capas totalmente conectadas. Finalmente se genera una salida, que es comparada con las etiquetas verdaderas utilizando una función de pérdida, en este caso entropía cruzada.
- **Propagación hacia atrás:** Se calcula el gradiente de la función de pérdida con respecto a la salida de la red. Estos gradientes se utilizan para actualizar los parámetros de la red de manera progresiva, comenzando desde la última capa y avanzando hacia atrás hasta llegar al inicio de la red. La actualización de los parámetros permite al modelo reducir la diferencia de la función perdida y mejorar la precisión, a este proceso le podemos llamar aprendizaje de la red.

Capítulo 6

Implementación y pruebas

En este capítulo se describe el proceso de implementación y pruebas del sistema. La implementación se divide en una serie de módulos que incluyen el análisis exploratorio de datos, el procesado de audio, el aumento de datos y el diseño de la red. Una vez expuestos sus componentes, se someterá a análisis la aplicación, para comprobar el cumplimiento de los requisitos.

6.1. Implementación

6.1.1. Análisis Exploratorio de Datos

GTZAN es uno de los conjuntos de datos más relevantes para la creación y prueba de modelos relacionados con audio. Es un conjunto de datos considerablemente limpio, con una distribución de clases balanceada, donde todos los clips tienen la misma duración. Se pueden procesar usando el mismo método, todos los audios tienen el mismo formato y no hay archivos corruptos.

Tiene un largo historial de uso en la industria desde 2002, sin embargo, en la última década ha sido sujeto de críticas se que posicionan a favor de usar datasets más recientes. Los argumentos de estas críticas tratan sobre la corta duración de las canciones, que la selección de canciones no representa la música que se escucha en la vida cotidiana, ya que han sido sacadas de colecciones académicas, y los géneros musicales definidos en el conjunto de datos son demasiado amplios y vagos. Por ejemplo, el género 'bluesén GTZAN incluye tanto el blues tradicional como el blues eléctrico moderno, que tienen características musicales muy diferentes.

6.1.2. Procesado de audio

La tarea de este modulo consiste en convertir los archivos de audio del conjunto de datos GTZAN en espectrogramas de Mel. La librería Librosa permite la generación de espectrogramas con gran variedad de parámetros.

El audio se muestrea con un sample rate de 22050 Hz con un solo canal. El tamaño de ventana de la FFT y la longitud de salto del espectrograma han sido definidos con valores de 2048 y 512 respectivamente. En algunos artículos se usan diferentes valores como 1024 y 256, que también da buenos resultados, lo que si debe ser constante es la proporción 4 a

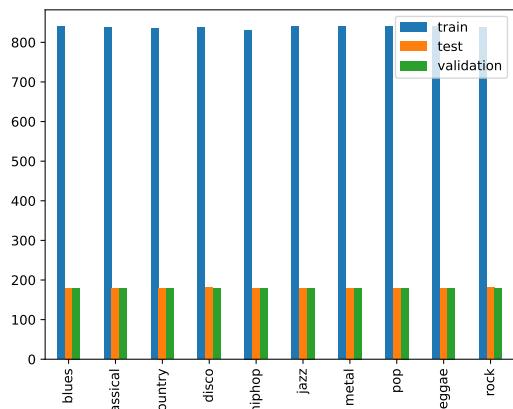


Figura 6.1: Distribución de sets

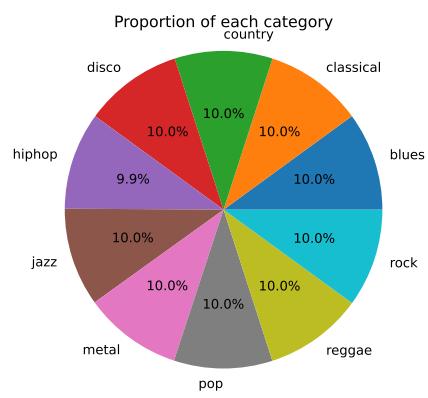


Figura 6.2: Distribución clases

1 entre las variables.

La duración de los audios del conjunto es de 30 segundos, a los que aplicaremos una división. En este caso estaremos generando 6 espectrogramas de Mel de 5 segundos de duración por cada audio. También se ha probado la división del audio en fragmentos de 3 segundos, que es otro valor frecuentemente usado en los estudios de esta temática, y el resultado es similar pese al incremento del número de muestras, por ello la duración elegida ha sido 5 segundos.

```
1 import os
2 import librosa
3 import matplotlib.pyplot as plt
4 import librosa.display
5
6 # Función para crear espectrogramas a partir de archivos de audio
7 def create_spectrograms(input_path, segment_duration):
8     # Iterar sobre todos los archivos en el directorio de entrada
9     for file in os.listdir(input_path):
10         # Obtener el nombre del directorio más externo (última carpeta en la ruta)
11         last_folder = os.path.basename(os.path.basename(input_path))
12
13         # Ruta completa al archivo de audio
14         audio_file = os.path.join(input_path, file)
15         try:
16             # Cargar el archivo de audio con librosa
17             y, sr = librosa.load(audio_file)
18         except:
19             print(f"Error cargando {audio_file}")
20             continue # Continuar con el siguiente archivo si hay un error al cargar
21
22         # Calcular la duración total del audio en segundos
23         duration = len(y) / sr
24
25         # Iniciar desde el segundo 0
26         start_sec = 0
27
28         # Iterar a través de segmentos de duración especificada
29         for i in range(0, int(duration / segment_duration)):
30             # Crear una figura para el espectrograma
31             fig = plt.figure(figsize=(216/100, 224/100))
32             ax = fig.add_subplot(1, 1, 1)
33             fig.subplots_adjust(left=0, right=1, bottom=0, top=1)
34
35             # Calcular el espectrograma mel
36             mel = librosa.feature.melspectrogram(y=y[int(start_sec*sr):int((start_sec+segment_duration)*sr)], sr=sr, n_mels=224)
37             mel = librosa.power_to_db(mel)
38
39             # Mostrar el espectrograma en la figura
40             librosa.display.specshow(mel, sr=sr, cmap='magma')
41
42             # Definir la ruta donde se guardarán los espectrogramas
43             path = f'C:\Diego\music-genre-classifier\imagenes\gtzan2\{last_folder}\'
44
45             # Verificar si la ruta existe, si no, crearla
46             if not os.path.exists(path):
47                 os.makedirs(path)
48
49             # Guardar la figura como imagen PNG
```

```

50     plt.savefig(f'{path}/{file}_{i}.png')
51
52     # Limpiar la figura para la siguiente iteración
53     plt.clf()
54
55 # Ruta principal donde se encuentran los directorios de géneros musicales
56 path = "C:/Diego/music-genre-classifier/audio/gtzan"
57
58 # Obtener una lista de nombres de directorios (géneros musicales)
59 directorios = [nombre for nombre in os.listdir(path) if os.path.isdir(os.path.join(
60     path, nombre))]
61
62 # Iterar sobre cada directorio y crear espectrogramas para cada archivo de audio
63 for dir in directorios:
64     create_spectrograms(f'{path}/{dir}', 5) # Llamar a la función create_spectrograms
65     # con cada directorio y una duración de segmento de 5 segundos

```

Listado 6.1: Código para crear espectrogramas de Mel

6.1.3. Aumento de datos

Para realizar el aumento de datos se ha utilizado la librería Audiomentations, que de forma sencilla nos permite aplicar diferentes transformaciones de audio a la vez. Las transformaciones seleccionadas han sido:

- **AddGaussianNoise**: Añade una señal de ruido de amplitud aleatoria que se caracteriza por tener una distribución gaussiana. Se añade con el objetivo de aumentar la robustez del modelo, los modelos entrenados con ruido tienden a generalizar con datos no conocidos.
- **Time Stretch**: Estira o comprime el audio en el tiempo, cambiando su duración sin afectar el tono sobre un rango de 0.8 a 1.1 veces su velocidad original.
- **PitchShift**: Cambia el tono sin cambiar la duración en un rango definido de -4 y 4 semitonos.

Cada una de las transformaciones tiene un 50 % de aplicarse en cada canción. El aumento de datos se aplica sobre todo el conjunto de audios de GTZAN, generando un nuevo conjunto de datos con el doble de muestras formado por las canciones originales y las transformadas.

```

1 from audiomentations import Compose, AddGaussianNoise, TimeStretch, PitchShift
2
3 def apply_audio_augmentations(folder_path, dir):
4     # Componer una serie de aumentaciones de audio utilizando la librería
5     # audiomentations
6     augment = Compose([
7         AddGaussianNoise(min_amplitude=0.001, max_amplitude=0.015, p=0.5), # Añadir
8         # ruido gaussiano al audio con una probabilidad de 0.5
9         TimeStretch(min_rate=0.8, max_rate=1.1, p=0.5), # Estirar o comprimir el
10        # tiempo del audio con una probabilidad de 0.5
11        PitchShift(min_semitones=-4, max_semitones=4, p=0.5) # Cambiar el tono del
12        # audio en un rango de -4 a 4 semitonos con una probabilidad de 0.5
13    ])

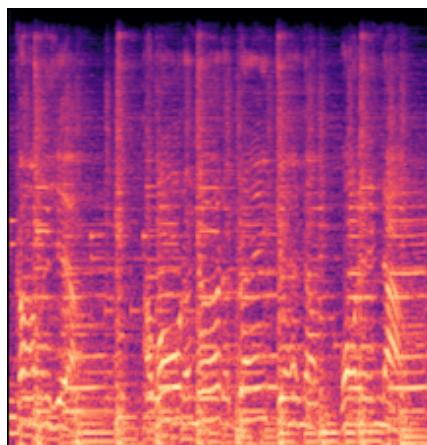
```

```

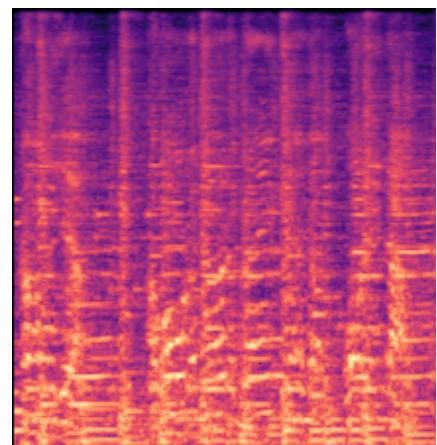
11 # Iterar sobre cada archivo en el directorio especificado por folder_path
12 for filename in os.listdir(folder_path):
13     song_path = os.path.join(folder_path, filename) # Obtener la ruta completa
14     del archivo de audio
15     try:
16         # Cargar el archivo de audio
17         audio, sr = librosa.load(song_path)
18
19         # Aplicar las aumentaciones de audio
20         augmented_audio = augment(samples=audio, sample_rate=sr)
21
22         # Crear espectrogramas para el audio aumentado y guardarlos en la ruta
23         especificada
24         create_spectrograms(augmented_audio, sr, 5, f'C:/Diego/music-genre-
25 classifier/imagenes/gtzan2/{dir}', f'{filename}_augmented')
26     except Exception as e:
27         # Manejar excepciones e imprimir cualquier error que ocurra durante el
28         procesamiento
29         print(f"Error processing {filename}: {str(e)}")

```

Listado 6.2: Código para el aumento de datos

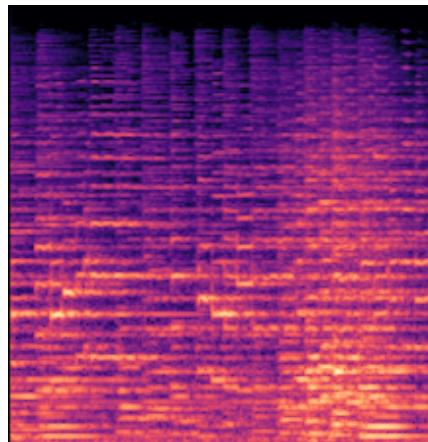


(a)

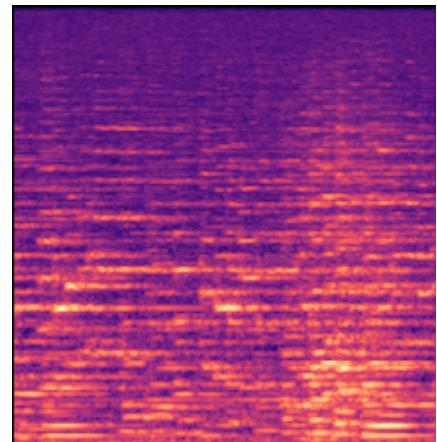


(b)

Figura 6.3: Blues, comparación antes (izquierda) y después (derecha) del aumento de datos

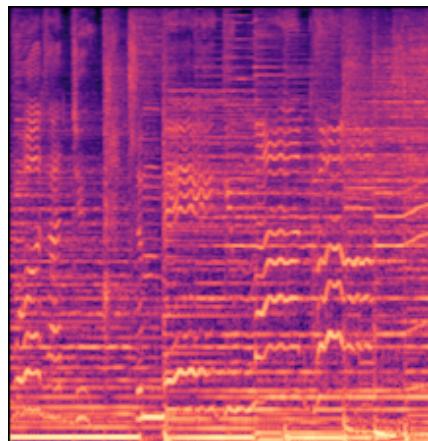


(a)

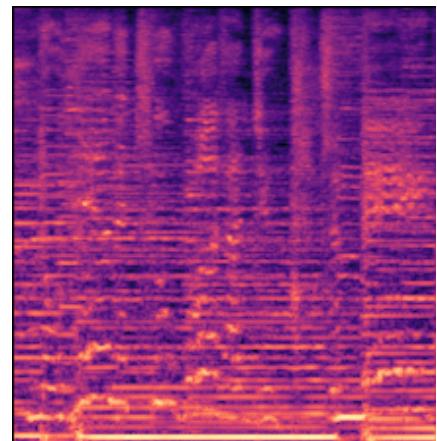


(b)

Figura 6.4: Classical

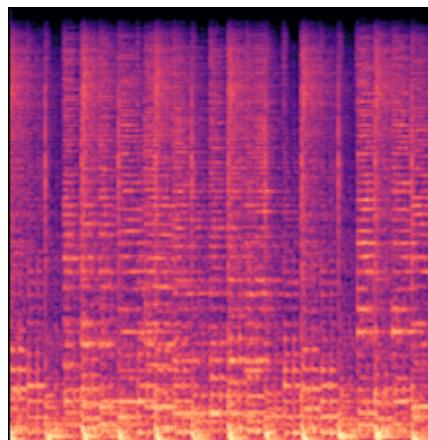


(a)

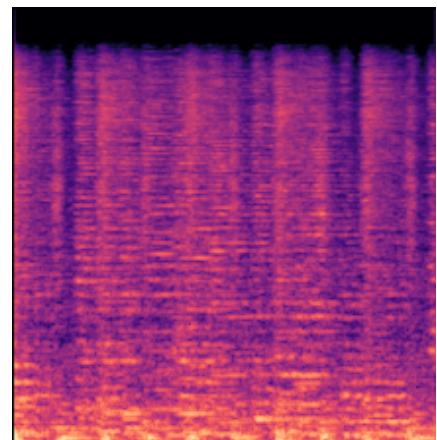


(b)

Figura 6.5: Country

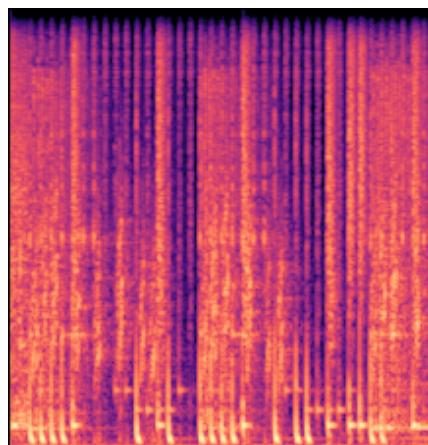


(a)

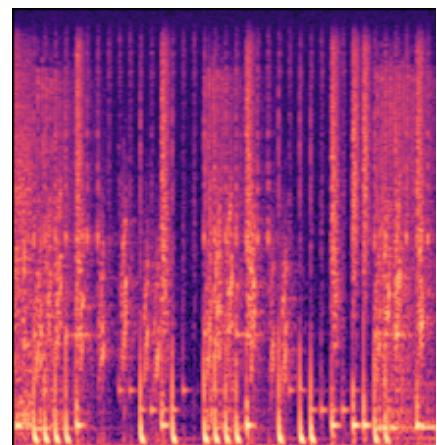


(b)

Figura 6.6: Disco

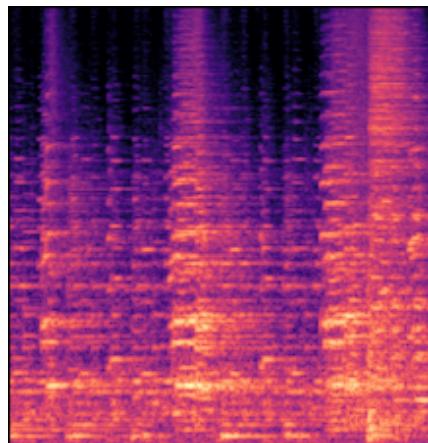


(a)

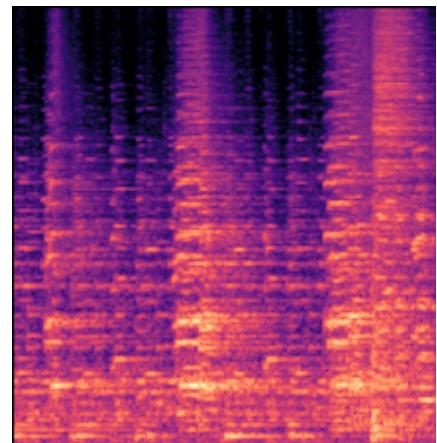


(b)

Figura 6.7: Hip Hop

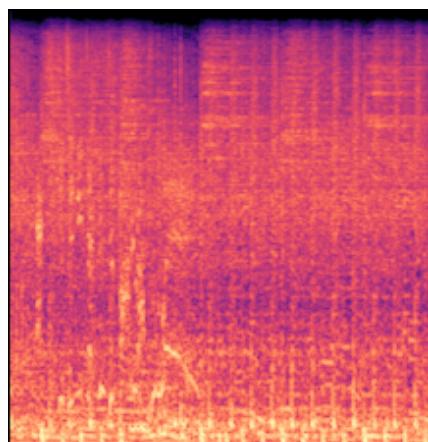


(a)

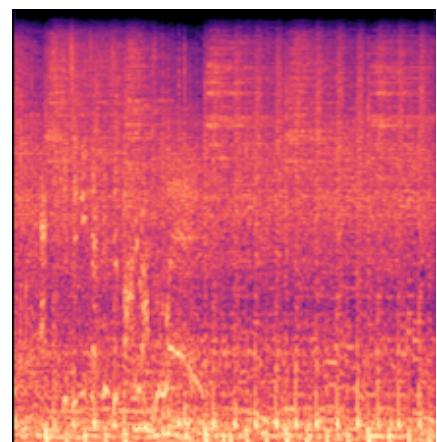


(b)

Figura 6.8: Jazz

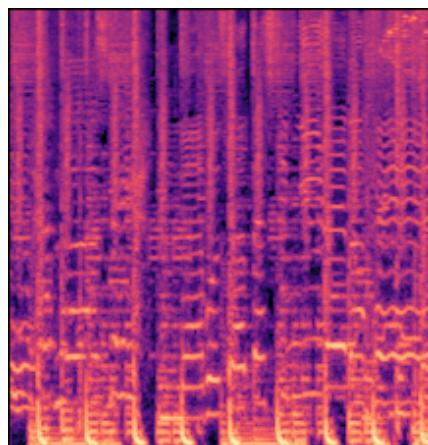


(a)

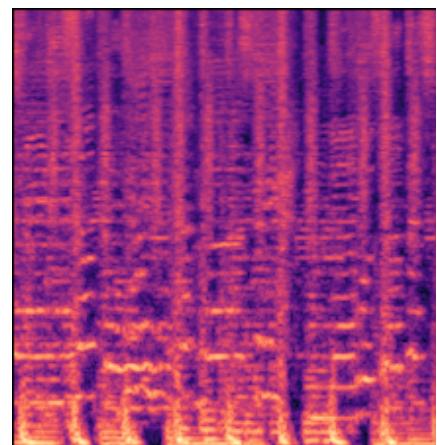


(b)

Figura 6.9: Metal

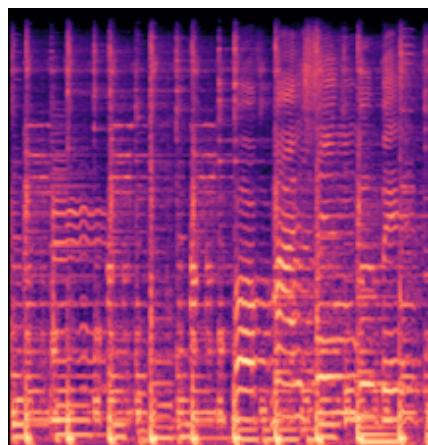


(a)

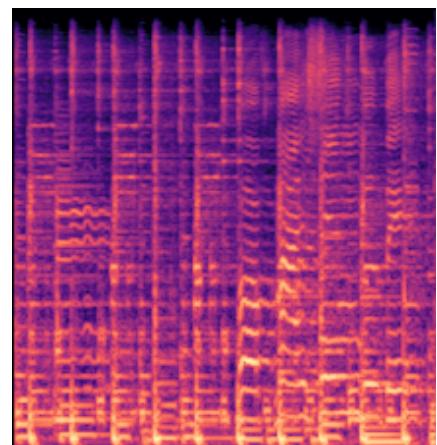


(b)

Figura 6.10: Pop

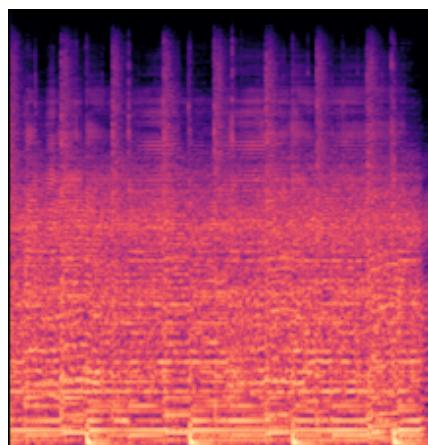


(a)

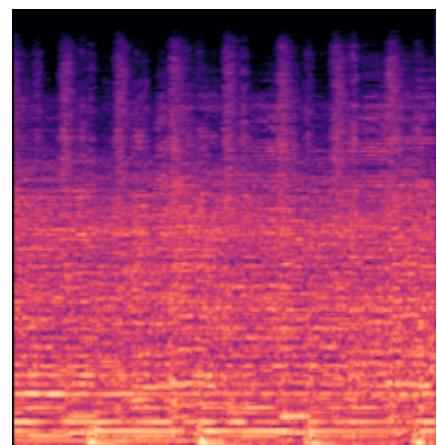


(b)

Figura 6.11: Reggae



(a)



(b)

Figura 6.12: Rock

6.1.4. Arquitectura de la red

Finalmente, la red propuesta para este trabajo tiene 5,677,034 parámetros y está formada por 5 capas convolucionales 2D con un numero de filtros que aumenta progresivamente desde 32 a 128. Por otra parte el tamaño de los filtros es de 5 en todas las capas a excepción de la primera que tiene un filtro de 3x3, debido a que un filtro menor tamaño, capta mejor las características locales de la imagen, y por lo tanto es más adecuado para las primeras capas de una CNN.

La salida del último bloque convolucional es enviada a una red totalmente conectada con 2 capas ocultas, con 1024 y 256 neuronas respectivamente, y una salida Softmax para las 10 clases del conjunto.

El optimizador seleccionado es Adam, el más usado en problemas de este tipo, con una tasa de aprendizaje inicial de 1e-3, que varía gracias al algoritmo de optimización. La función de perdida seleccionada es CategoricalCrossentropy, que alcanza mejores resultados que otras funciones como el error cuadratico medio. Se ha añadido un callback para parar el entrenamiento en caso de que la función de perdido del conjunto de validación no mejore durante 5 épocas. La división en sets se ha hecho mediante la regla de 0.7-0.15-0.15 para entreno, test y validación.

```

1 def conv_block(model, filters, kernel_size, pooling=True):
2     model.add(Conv2D(filters, kernel_size, padding="same", activation="relu"))
3     if pooling:
4         model.add(MaxPooling2D((2, 2), strides=(2, 2)))
5
6 model = Sequential()
7 input_shape=(img_width,img_height,3)
8 model.add(Conv2D(32, 3, input_shape=input_shape,activation="relu"))
9 model.add(Activation('relu'))
10 model.add(MaxPooling2D())
11 conv_block(model,32,5)
12 conv_block(model,64,5)
13 conv_block(model,128,5)
14 conv_block(model,128,5)
15
16 model.add(Dropout(0.4))
17 model.add(Flatten())
18 model.add(Dense(1024))
19 model.add(Activation('relu'))
20
21 model.add(Dense(256))
22 model.add(Activation('relu'))
23
24 model.add(Dense(10))
25 model.add(Activation('softmax'))
26
27 model.summary()
```

Listado 6.3: Código para la hiperparametrización

Hiperparametrización

Es el proceso para seleccionar los mejores parámetros para un modelo de aprendizaje automático. En este caso el procedimiento comienza con la definición de diferentes combinaciones de hiperparámetros, en este caso número de filtros, numero de neuronas

por capa y dropout. El resultado de esta búsqueda devuelve las precisiones de todos las combinaciones posibles según los parámetros dados. Este método es muy demandante temporalmente, ya que una secuencia tan grande de entrenamientos puede requerir horas o días enteros.

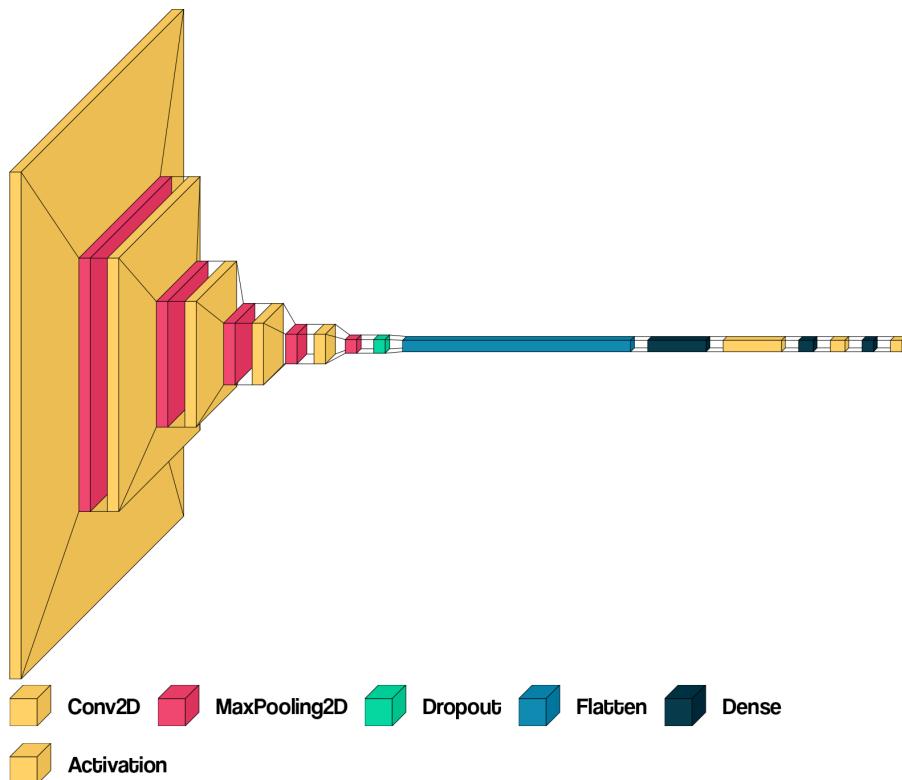
Gracias a la hiperparametrización podemos observar que párametros funcionan mejor con el conjunto de datos y la arquitectura.

```

1 filters_per_layer = [[32, 64, 128, 128], [32, 64, 64, 128], [32, 64, 128, 256]]
2 dropout_rate = [0.1, 0.3]
3 dense_units = [[1024, 256], [512, 256], [128, 64]]
4 best_accuracy = 0
5 for filters in filters_per_layer:
6     for dropout in dropout_rate:
7         for dense in dense_units:
8             model = create_model(filters_per_layer=filters, dropout_rate=dropout,
9 dense_units=dense, input_shape=(img_width, img_height, 3))
10            print('Training model with filters={}, dropout={}, dense={}'.format(
11 filters, dropout, dense))
12            model.fit(train_generator, validation_data=validation_generator, epochs
13 =10, verbose=0)
14            accuracy = model.evaluate(test_generator)[1]
15            print('Accuracy:', accuracy)
16            if accuracy > best_accuracy:
17                best_accuracy = accuracy
18                best_model = model

```

Listado 6.4: Código para la hiperparametrización



6.1.5. Interfaz

Ha sido diseñada con el objetivo de ser lo más sencilla posible, dando prioridad a la correcta funcionalidad de la aplicación en conjunto. Implementa un botón que abrirá el explorador de archivos del sistema operativo para seleccionar un audio en formato MP3, WAV o FLAC. A continuación, el programa realizará y mostrará los resultados en una ventana pop-up junto con el tiempo de ejecución el género o los géneros predichos.



Figura 6.14: Interfaz de la aplicación

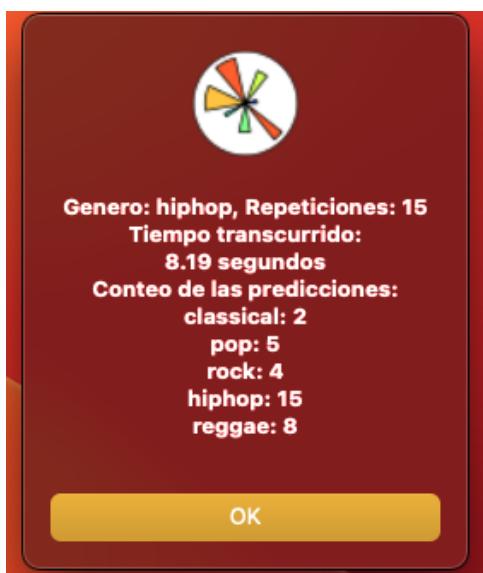


Figura 6.15: Resultados de una predicción

6.2. Resultados

Durante el entrenamiento el callback se activa en la época 21 y devuelve los pesos de la época 16. En ese punto el modelo alcanza una accuracy de 0.9 sobre el conjunto de validación. Se guarda el modelo y procedemos a someterlo a diferentes pruebas con el conjunto de test. El reporte de clasificación de la biblioteca scikit-learn es un método ideal para obtener unas primeras impresiones del desempeño del clasificador. Una buena señal es el balance entre las 3 métricas, precision, recall y f1-score, que de forma global tienen un valor 0.9, esto indica que el funcionamiento del clasificador es excelente, teniendo en cuenta la naturaleza del problema. No solo es preciso, sino que también es capaz de identificar correctamente las instancias positivas y negativas. A partir de los resultados

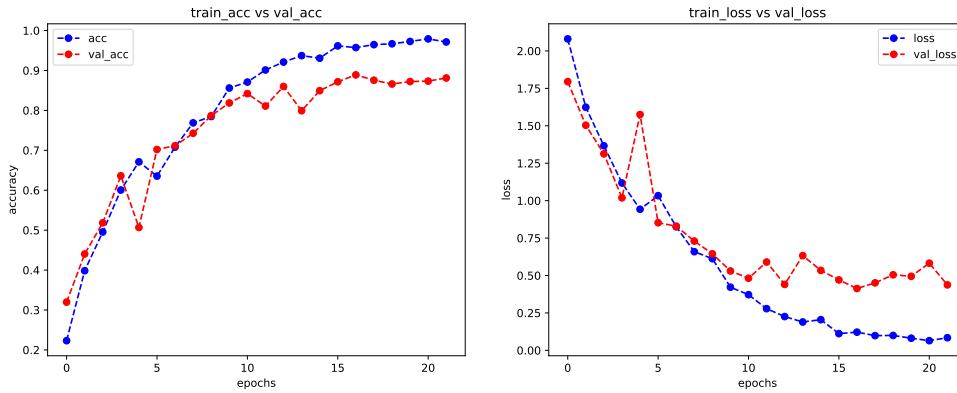


Figura 6.16: Entreno de la red

	precision	recall	f1-score	support
blues	0.83	0.93	0.88	180
classical	0.93	0.94	0.94	180
country	0.86	0.89	0.88	180
disco	0.90	0.87	0.89	181
hiphop	0.91	0.91	0.91	180
jazz	0.89	0.91	0.90	180
metal	0.96	0.93	0.95	180
pop	0.92	0.88	0.90	180
reggae	0.91	0.89	0.90	180
rock	0.87	0.81	0.84	181
accuracy			0.90	1802
macro avg	0.90	0.90	0.90	1802
weighted avg	0.90	0.90	0.90	1802

Figura 6.17: Reporte de clasificación

de la matriz de confusión podemos interpretar que hay un cierto desbalanceo en la clase 9, rock, que es frecuentemente etiquetada como blues o country. Esto puede deberse a que algunas canciones de rock comparten características similares con esos géneros, como el uso de guitarras eléctricas o un ritmo específico.

Por otro lado, la clase classical tiene un comportamiento sobresaliente, al acumular un número de errores menor al resto de clases. Podemos interpretar este resultado como una cualidad intrínseca del género, la música clásica cuenta con una instrumentación, ritmo y melodía muy diferente al resto de géneros que aparecen en GTZAN y es por ello que la CNN puede distinguir 'classical' de forma casi perfecta.

Parece que el clasificador tenga mejores resultados en los géneros que también son más fáciles de distinguir para los humanos. Esto sugiere que el clasificador se beneficia de las mismas características perceptuales que utilizamos las personas. Una curva ROC es una métrica para evaluar el rendimiento de un clasificador que consiste en la visualización de tasa de verdaderos positivos contra la de falsos positivos a diferentes umbrales de decisión. El valor del área bajo la curva es un término clave para la evaluación de un clasificador, valores cercanos a 1 indican un rendimiento excelente.

Tradicionalmente las curvas ROC solo se aplican a clasificadores binarios, pero es posible realizar una adaptación usando la técnica One-vs-Rest, que construye una curva ROC para cada clase, tratando cada clase como positiva y las demás clases como negativas.

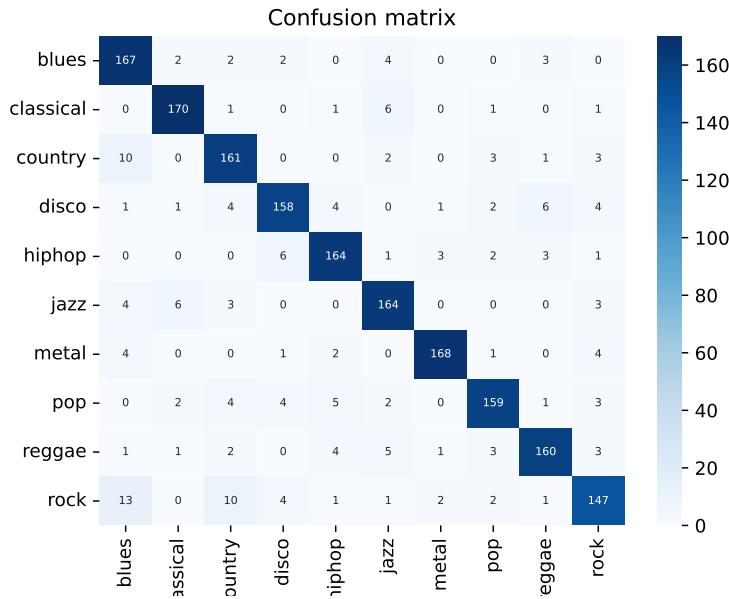


Figura 6.18: Matriz de confusión

Los resultados observables en la gráfica son excepcionales, el área bajo la curva obtiene valores muy altos de 0.99 o 1 a excepción de curva roja que representa 'rock'. La gráfica Precision-Recall funciona de forma similar a la ROC usa un enfoque One-vs-Rest, pero traza la precision respecto al recall.

6.2.1. Filtro de características

Analizar visualmente los filtros de características es una manera excelente de comprender el funcionamiento de la red. En este caso podemos visualizar los filtros de los cinco bloques convolucionales y observar como la red descompone progresivamente el espectrograma.

Primero, introduciremos el espectrograma 6.21 en la red neuronal. Configuraremos la red de tal forma que devuelva la salida de los filtros de características, que luego serán visualizadas con la ayuda del paquete Matplotlib.

El filtro del primer bloque, 6.22, parece captar la forma global del espectrograma. La mayoría de estos filtros se activan predominantemente en las partes superiores del espectrograma y se encargan de distinguir entre las frecuencias graves y agudas.

Por otro lado, los filtros del segundo y tercer bloque, 6.23 y 6.24, se activan de una forma más específica, reconocen patrones que no se ven a simple vista en los filtros de la capa de entrada. Su función es conservar y transmitir está información al mismo tiempo que la comprimen.

El último filtro, 6.25, tiene un tamaño menor, por lo que los píxeles son visibles a simple vista, pero es difícil reconocer el significado de la activación del filtro en ciertos puntos. Algunas activaciones parecen representar patrones, aunque otras parecen arbitrarias. Esto no implica que carezcan de utilidad, posiblemente estén captando relaciones lógicas que no pueden ser percibidas visualmente. Finalmente, la salida del filtro se convierte en la entrada de la red totalmente conectada.

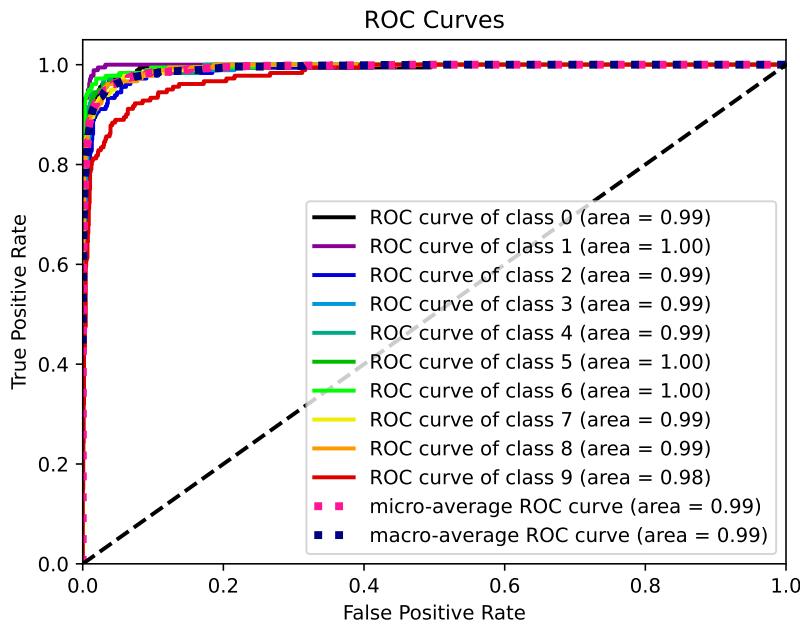


Figura 6.19: Curva ROC

6.2.2. Tiempo de ejecución

Para calcular los tiempos de procesamiento del sistema, se ha utilizado la librería 'time' de Python. Desde la transformación de audio a espectrograma y el cálculo de la predicción una vez cargados el modelo y las librerías. El tiempo medio para un fragmento de 30 segundos es de 0.7 segundos, mientras que para una canción de 5 minutos, este valor aumenta hasta los 7.8 segundos.

El tiempo de procesamiento depende exclusivamente de la duración del audio, por lo que, dependiendo de los requisitos, podemos adaptar el funcionamiento del sistema. El clasificador tendrá un rendimiento más preciso cuando el tiempo de procesamiento es mayor; sin embargo, el desempeño con fragmentos de 30 segundos de audio es suficientemente bueno como para no requerir la canción completa.

El rendimiento de la aplicación en un portátil Macbook Air 2019 M1 con MacOs Ventura es sorprendentemente bueno, mejorando los tiempos de ejecución. Probablemente este suceso es debido a la mejor gestión de procesos y memoria de la arquitectura ARM.

6.2.3. Prueba de funcionalidad

Se ha pedido a tres usuarios diferentes una lista con tres canciones para comprobar su funcionamiento. Los resultados de las predicciones de géneros musicales se presentan en la Tabla 6.1.

Al analizar los resultados, la mayoría de las predicciones coinciden con el género real, por lo que podemos valorar la prueba como positiva. Sin embargo, existen algunas discrepancias interesantes que merecen ser destacadas.

Aunque "Smooth Criminal" es clasificada generalmente como Pop, es comprensible que el sistema lo haya predicho como Disco, la canción contiene elementos y un ritmo característicos presentes en la música Disco, por lo que podemos considerar el error como

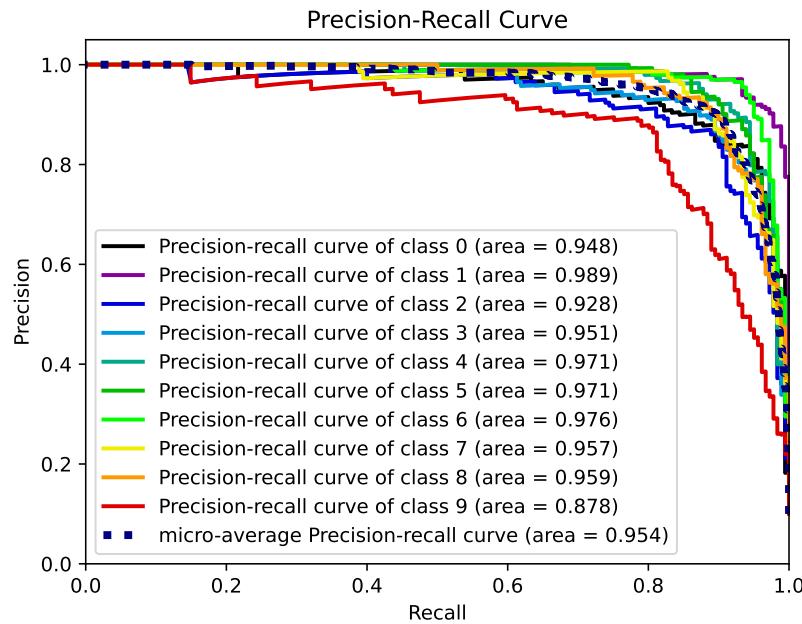


Figura 6.20: Recall

Cuadro 6.1: Resultados de predicción de géneros musicales

Canción	Género Real	Género Predicho
Sweet Home Alabama - Lynyrd Skynyrd	Rock	Rock
Smooth Criminal - Michael Jackson	Pop	Disco
Pura Droga Sin Cortar - Kase.O	Hip-hop	Hip-hop
So What - Miles Davis	Jazz	Jazz
Canon en Re - Johann Pachelbel	Classical	Classical
Is this love - Bob Marley	Reggae	Reggae
Master of Puppets - Metallica	Metal	Metal
Lose Yourself - Eminem	Hip-hop	Reggae
Take Me Home, Country Roads - John Denver	Country	Country
The Thrill is Gone - B.B. King	Blues	Blues

permisible.

La canción 'Lose Yourself' de Eminem es un tema icónico dentro del Hip-Hop. Por lo que categorizarla como reggae es una discrepancia notable. Esto podría deberse a características específicas del instrumental que el sistema ha interpretado de manera incorrecta. Tras una pequeña búsqueda en el conjunto de datos, en las canciones Hip Hop en GTZAN predominan las que tienen un ritmo de bombo caja. 'Lose Yourself' carece de este ritmo, mientras que la canción de Kase.O tiene un ritmo de Hip Hop muy clásico y es predicho correctamente con un alto porcentaje.

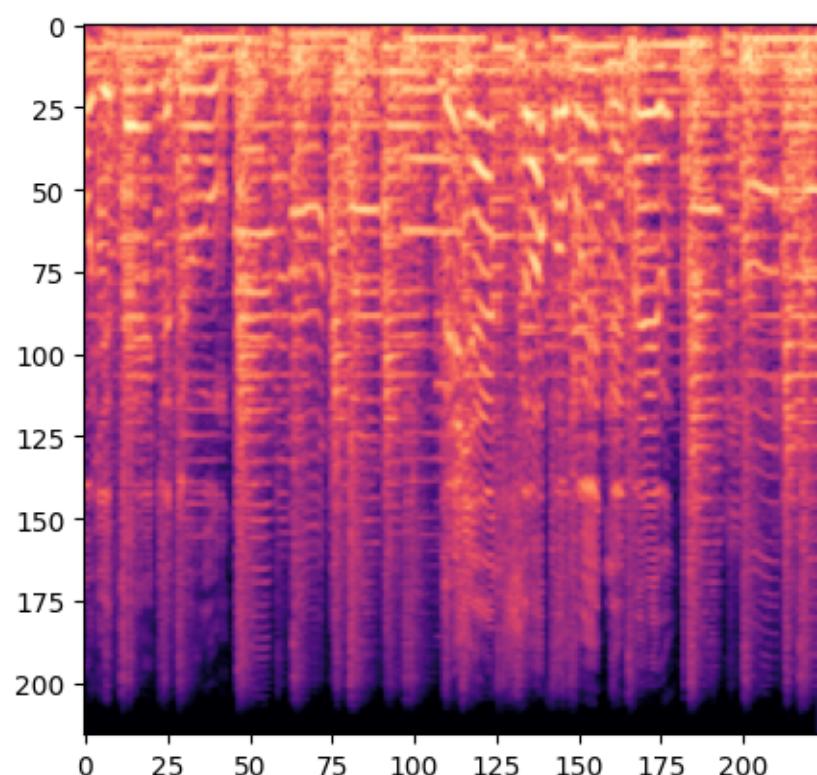


Figura 6.21: Entrada del filtro de características

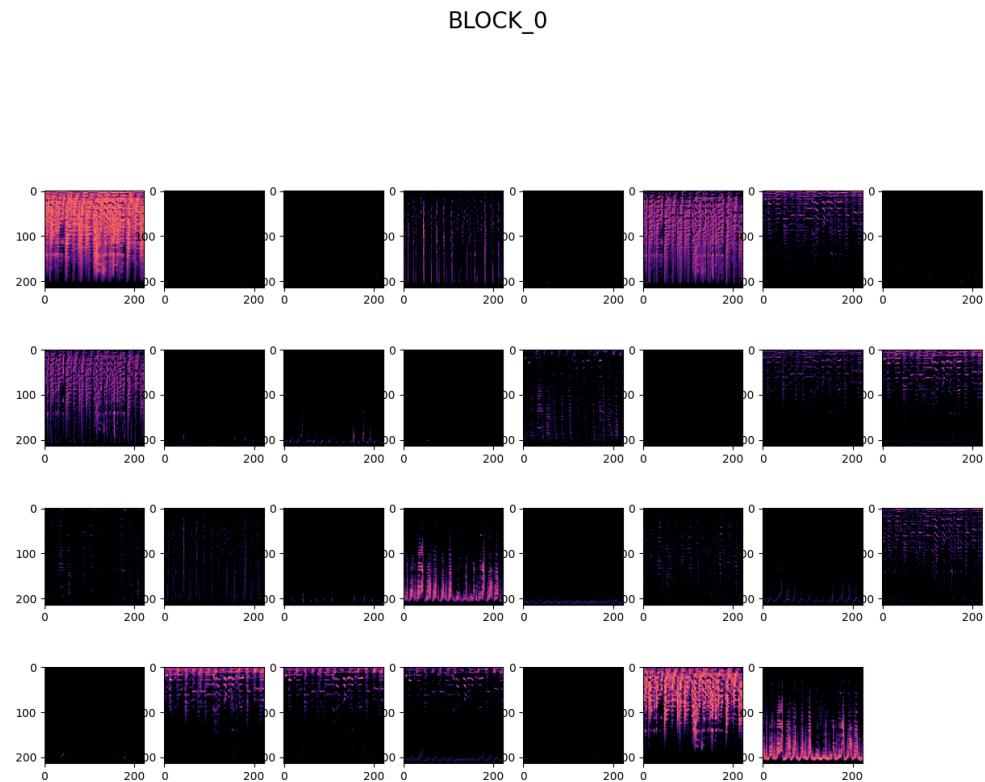


Figura 6.22: Filtro del Bloque 0

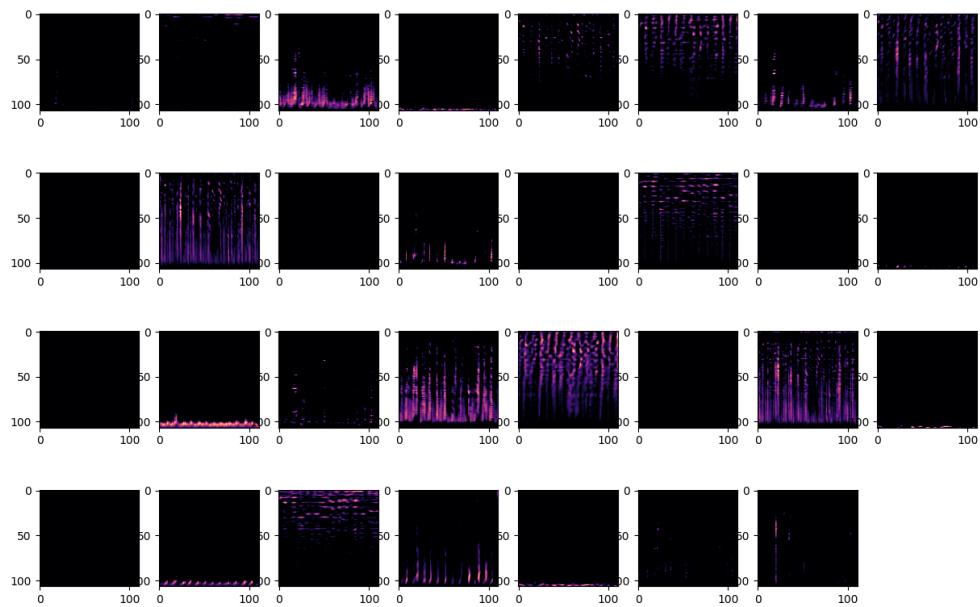
BLOCK_3

Figura 6.23: Filtro del Bloque 3

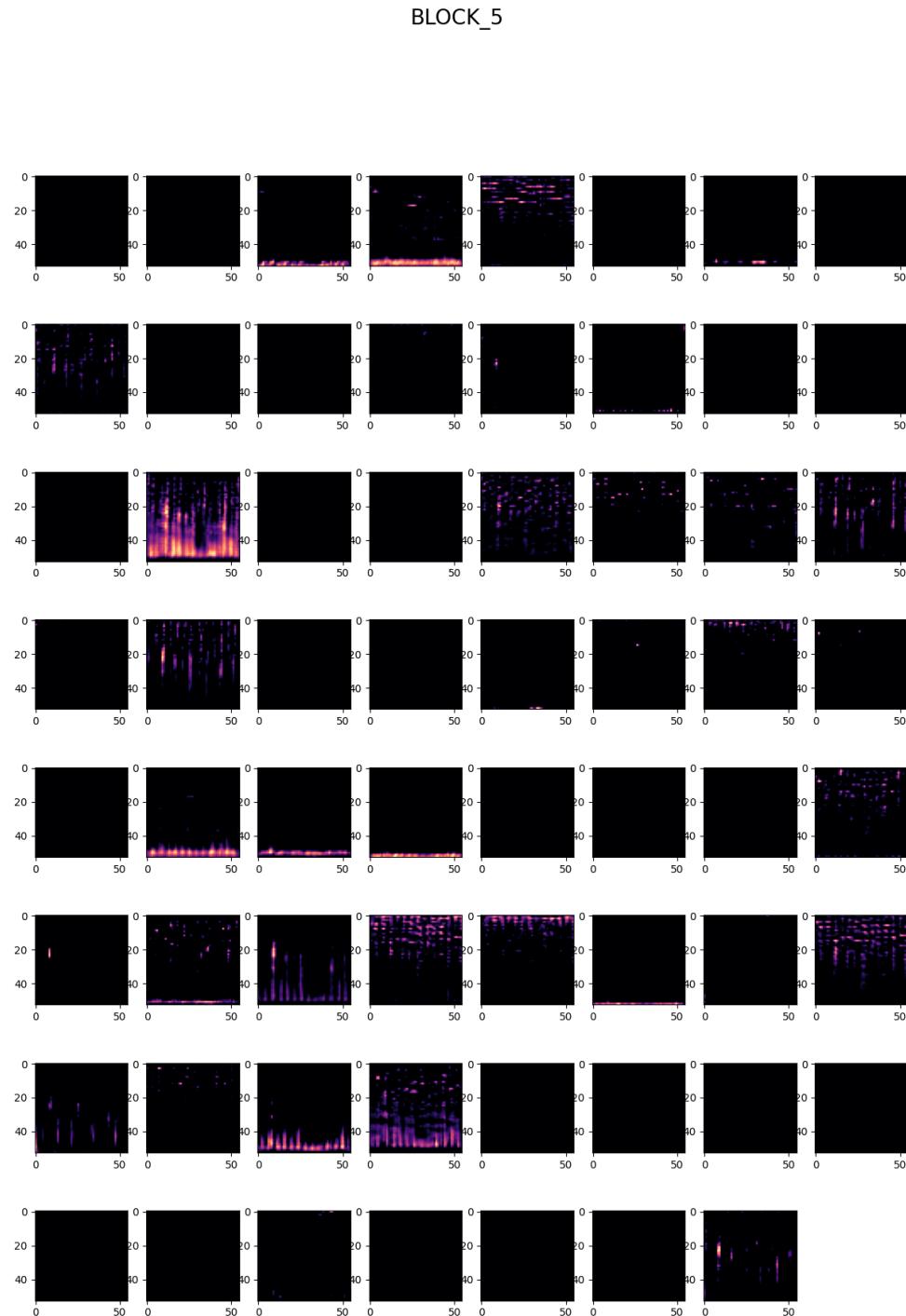


Figura 6.24: Filtro del Bloque 5

BLOCK_9



Figura 6.25: Filtro del Bloque 9

```
Tiempo transcurrido: 0.7255552999999964 segundos
[6, 6, 6, 6, 6, 6]
Genero: metal, Repeticiones: 6
Conteo de las predicciones:
Clase 6: 6 imágenes
```

Figura 6.26: Prueba fragmento

```
Tiempo transcurrido: 7.853250499999831 segundos
[1, 2, 9, 9, 4, 4, 8, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4,
Genero: hiphop, Repeticiones: 43
Conteo de las predicciones:
Clase 1: 1 imágenes
Clase 2: 2 imágenes
Clase 9: 8 imágenes
Clase 4: 43 imágenes
Clase 8: 2 imágenes
Clase 0: 1 imágenes
Clase 6: 4 imágenes
```

Figura 6.27: Prueba canción entera

Capítulo 7

Conclusiones

Como punto final al proyecto, en este capítulo se expondrá un resumen de la funcionalidad de la aplicación elaborada así como su desarrollo, además de un análisis elaborada de las posibles implantaciones o líneas de investigación a realizar en un futuro en caso de querer continuar con el mantenimiento y desarrollo.

7.1. Conclusiones

La aplicación presentada permite la carga de un archivo de audio mediante una interfaz simple y devuelve como salida el género musical de pertenencia. El correcto funcionamiento de la aplicación depende de diferentes módulos: procesamiento del audio, aumento de datos, generación de espectrogramas, diseño de la red neuronal, realización de pruebas y validación del modelo.

Una vez completado el desarrollo y conectado los módulos entre sí, se implementa una interfaz diseñada con el objetivo de ser lo más simple e intuitiva posible, y al ser la primera versión, minimizar el número de errores.

La principal fortaleza de la aplicación es la eficiencia, el modelo de la red neuronal tiene un tamaño de 15 MB.

Las CNNs por sí mismas se caracterizan por su habilidad para extraer y reducir características en modelos con un bajo número de parámetros, pero aún así los modelos generales son pesados y costosos de ejecutar por norma general. Para comparar el rendimiento y eficiencia del modelo neuronal propuesto se ha decidido comparar su rendimiento con otras modelos CNNs relevantes en la industria mediante la técnica de Transfer Learning. Los modelos seleccionados para la comparación son MobileNetV2 y ResNet50. Ambas redes se han importado desde el modulo de aplicaciones de Keras. MobileNetV2 es una red que destaca por su aplicación en dispositivos de bajo consumo, como dispositivos móviles o placas de desarrollo, en esta versión implementada en Keras, MobileNetV2 cuenta con 3,5 millones de parámetros, 2 millones menos que el modelo propuesto, que cuenta con 5,6 millones. ResNet50 tiene un enfoque diferente, a priori el rendimiento de ResNet50 es mayor que MobileNetV2 a cambio de un sacrificio en la eficiencia, el número de parámetros aumenta hasta los 23,9 millones. A continuación se realizará una prueba en la que los dos modelos mencionados, serán entrenados y probados con el mismo conjunto de datos. La métrica seleccionada para la comparación ha sido la Accuracy.

Según los resultados obtenidos, el modelo propuesto alcanza la mayor puntuación con

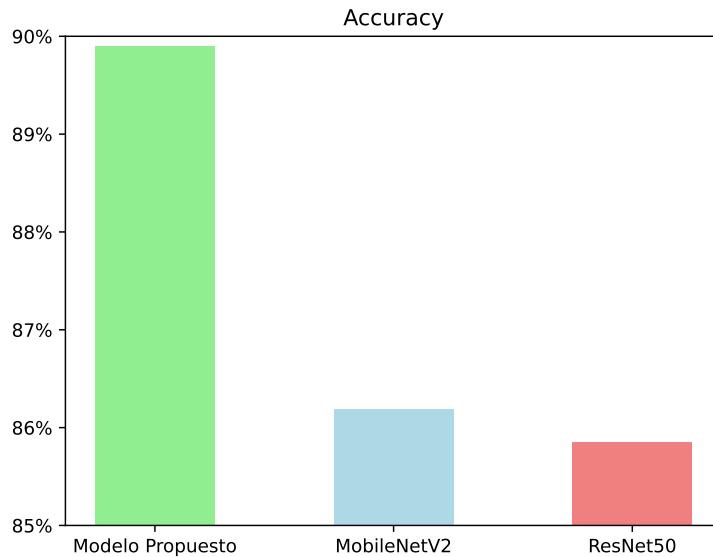


Figura 7.1: Comparación con modelos preentrenados

5 puntos de diferencia sobre el segundo. MobileNetV2 obtiene unos buenos resultados considerando la naturaleza de la red, y con un rendimiento mayor que ResNet50 de la cual descartamos totalmente una posible implementación.

Con estos resultados en la mano, podemos afirmar que el modelo propuesto tienen un excelente relación costo eficiencia, ofreciendo un rendimiento excepcional en la clasificación de géneros musicales a la vez que un costo computacional pequeño respecto a las redes neuronales usadas en la industria. También es importante mencionar que tanto ResNet50 como MobileNetV2 son redes neuronales ampliamente usadas en la industria, y que el modelo propuesto sobrepase el rendimiento de ambas es un resultado significativo y prometedor para su posible implementación en aplicaciones del mundo real.

7.2. Trabajo futuro

Después de analizar los resultados del modelo, podemos concluir que es adecuado para su implementación. Algunas de las aplicaciones en las que podría ser utilizado son:

- Organización de bibliotecas musicales
- Software de producción musical
- Algoritmos de recomendación

El rendimiento de la red podría mejorar si combináramos la red convolucional con otras arquitecturas de aprendizaje profundo.

Las LSTM (Long Short-Term Memory) son un tipo de red recurrente especializadas en manejar secuencias de datos y captar dependencias temporales. Es por ello que son ampliamente usadas en tareas de audio. Podemos usar esta red en compañía de la CNN, que se encargaría de la extracción de características del espectrograma, mientras que las LSTM pueden capturar la dinámica temporal de ellas. La implementación un modelo

mixto CNN-LSTM es una tarea relativamente sencilla que mejoraría el rendimiento del modelo. Sin embargo, el problema de crear modificar el modelo actual con otra arquitectura, sería el aumento de la complejidad del sistema, acompañado por un mayor tiempo de ejecución que puede ser perjudicial para la aplicación, sobre todo si hablamos de software en tiempo real.

Las Temporal Convolutional Networks (TCN) son una arquitectura reciente que se caracteriza por combinar las ventajas que ofrecen CNNs y RNNs. El estudio realizado sobre TCN es mucho menor que el de sus antecesoras, así que sería interesante saber su desempeño en tareas de audio, donde la temporalidad de los datos es de gran relevancia.

Bibliografía

- [1] Lei Wang, Ziyi Zhao, Hanwei Liu, Junwei Pang, Yi Qin, and Qidi Wu. A review of intelligent music generation systems. *Neural Computing and Applications*, 36(12):6381–6401, Apr 2024.
- [2] Sumaiya Dabeer, Maha Mohammed Khan, and Saiful Islam. Cancer diagnosis in histopathological image: Cnn based approach. *Informatics in Medicine Unlocked*, 16:100231, 2019.
- [3] Jaume Segura-Garcia, Sean Sturley, Miguel Arevalillo-Herraez, Jose M. Alcaraz-Calero, Santiago Felici-Castell, and Enrique A. Navarro-Camba. 5g ai-iot system for bird species monitoring and song classification. *Sensors*, 24(11), 2024.
- [4] Abul Abbas Barbhuiya, Ram Kumar Karsh, and Rahul Jain. Cnn based feature extraction and classification for sign language. *Multimedia Tools and Applications*, 80(2):3051–3069, 2021.
- [5] Zoltán Szlávik and Tamás Szirányi. Face analysis using cnn-um. In *Proceedings IEEE International Workshop on Cellular Neural Networks and their Applications (CNNA 2004)*, pages 190–195. Citeseer, 2004.
- [6] Bob L. Sturm. The state of the art ten years after a state of the art: Future research in music information retrieval. *Journal of New Music Research*, 43(2):147–172, April 2014.
- [7] Mingwen Dong. Convolutional neural network achieves human-level accuracy in music genre classification. *CoRR*, abs/1802.09697, 2018.
- [8] Yu Duan. Broadcast swin transformer for music genre classification. *Highlights in Science, Engineering and Technology*, 85:691–699, Mar. 2024.
- [9] Yigang Meng. Music genre classification: A comparative analysis of cnn and xgboost approaches with mel-frequency cepstral coefficients and mel spectrograms, 2024.
- [10] Weibin Zhang, Wenkang Lei, Xiangmin Xu, and Xiaofen Xing. Improved music genre classification with convolutional neural networks. In *Interspeech*, 2016.
- [11] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition, 2015.
- [12] Mohsin Ashraf, Fazeel Abid, Ikram Ud Din, Jawad Rasheed, Mirsat Yesiltepe, Sook Fern Yeo, and Merve Ersoy. A hybrid cnn and rnn variant model for music classification. *Applied Sciences*, 13:1476, 01 2023.

- [13] Jash Mehta, Deep Gandhi, Govind Thakur, and Pratik Kanani. Music genre classification using transfer learning on log-based mel spectrogram. In *2021 5th International Conference on Computing Methodologies and Communication (ICCMC)*, pages 1101–1107, 2021.
- [14] Pei-Chun Chang, Yong-Sheng Chen, and Chang-Hsing Lee. Ms-sincresnet: Joint learning of 1d and 2d kernels using multi-scale sincnet and resnet for music genre classification. In *Proceedings of the 2021 International Conference on Multimedia Retrieval, ICMR '21*, page 29–36, New York, NY, USA, 2021. Association for Computing Machinery.
- [15] Rendra Soekarta, Suhardi Aras, and Ahmad Nur Aswad. Hyperparameter optimization of cnn classifier for music genre classification. *Jurnal RESTI (Rekayasa Sistem dan Teknologi Informasi)*, 7(5):1205 – 1210, Oct. 2023.
- [16] G. Tzanetakis and P. Cook. Musical genre classification of audio signals. *IEEE Transactions on Speech and Audio Processing*, 10(5):293–302, 2002.
- [17] Taras Gorishnyy, Martin Maldovan, Chaitanya Ullal, and Edwin Thomas. Sound ideas. *Physics World*, 18(12):24, dec 2005.
- [18] D.J. Benson. *Music: A Mathematical Offering*. Cambridge University Press, 2007.
- [19] Wikipedia. Amplitud de onda — Wikipedia, the free encyclopedia. <http://es.wikipedia.org/w/index.php?title=Amplitud%20de%20onda&oldid=158272096>, 2024. [Online; accessed 05-April-2024].
- [20] Stephen Mcadams. *Musical Timbre Perception*, pages 35–67. 12 2013.
- [21] Brian C.J. Moore. *Psychoacoustics*, pages 475–517. Springer New York, New York, NY, 2014.
- [22] D. O'Shaughnessy. *Speech Communication: Human and Machine*. Addison-Wesley series in electrical engineering. Addison-Wesley Publishing Company, 1987.
- [23] Santhosh Umesh, Leon Cohen, and Douglas Nelson. Fitting the mel scale. volume 1, pages 217 – 220 vol.1, 04 1999.
- [24] Richard F. Lyon, Andreas G. Katsiamis, and Emmanuel M. Drakakis. History and future of auditory filter models. In *Proceedings of 2010 IEEE International Symposium on Circuits and Systems*, pages 3809–3812, 2010.
- [25] B.C. Moore and B.R. Glasberg. Suggested formulae for calculating auditory-filter bandwidths and excitation patterns. *Journal of the Acoustical Society of America*, 74(3):750–753, 1983.
- [26] Hartmut Traunmüller. Analytical expressions for the tonotopic sensory scale. *The Journal of the Acoustical Society of America*, 88(1):97–100, 07 1990.
- [27] Albertus den Brinker, Jeroen Breebaart, Per Ekstrand, Jonas Engdegård, Fredrik Henn, Kristofer Kjörling, Werner Oomen, and Heiko Purnhagen. An overview of the coding standard mpeg-4 audio amendments 1 and 2: He-aac, ssc, and he-aac v2. *EURASIP J. Audio, Speech and Music Processing*, 2009, 01 2009.

- [28] Siemens Aktiengesellschaft. Verfahren und vorrichtung zur authentifizierung eines benutzers, 2007.
- [29] Md. Afzal Hossan, Sheeraz Memon, and Mark A Gregory. A novel approach for mfcc feature extraction. In *2010 4th International Conference on Signal Processing and Communication Systems*, pages 1–5, 2010.
- [30] Arzo Mahmood and Utku Köse. Speech recognition based on convolutional neural networks and mfcc algorithm. 1:6–12, 01 2021.
- [31] S. Deepak and B.G. Prasad. Music classification based on genre using lstm. In *2020 Second International Conference on Inventive Research in Computing Applications (ICIRCA)*, pages 985–991, 2020.
- [32] Wen-Hsiung Chen, C. Smith, and S. Fralick. A fast computational algorithm for the discrete cosine transform. *IEEE Transactions on Communications*, 25(9):1004–1009, 1977.
- [33] Fang Zheng, Guoliang Zhang, and Zhanjiang Song. Comparison of different implementations of mfcc. *J. Comput. Sci. Technol.*, 16:582–589, 11 2001.
- [34] Matthew Field, Nicholas Hardcastle, Michael Jameson, Noel Aherne, and Lois Holloway. Machine learning applications in radiation oncology. *Physics and Imaging in Radiation Oncology*, 19:13–24, 2021.
- [35] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016.
- [36] Rathindra Nath Mohalder, Md. Alam Hossain, and Nazmul Hossain. Classifying the supervised machine learning and comparing the performances of the algorithms. *International Journal of Advanced Research*, 12:422–438, 01 2024.
- [37] Warren S. McCulloch and Walter Pitts. A logical calculus of the ideas immanent in nervous activity. *Bulletin of Mathematical Biophysics*, 5(1-2):115–133, 1943.
- [38] Rafael Prieto Meléndez, A Herrera, J Pérez, and Alejandro Padrón-Godínez. El modelo neuronal de mcculloch y pitts interpretación comparativa del modelo. 10 2000.
- [39] Marvin Minsky and Seymour Papert. *Perceptrons: An Introduction to Computational Geometry*. 09 2017.
- [40] Hang Li. *Perceptron*, pages 39–53. 12 2023.
- [41] Akshay Hebbar. Mcts guided genetic algorithm for optimization of neural network weights, 2023.
- [42] Akhilesh Waoo and Brijesh Soni. *Performance Analysis of Sigmoid and Relu Activation Functions in Deep Neural Network*, pages 39–52. 07 2021.
- [43] Yumin Pan. Different types of neural networks and applications: Evidence from feedforward, convolutional and recurrent neural networks. *Highlights in Science, Engineering and Technology*, 85:247–255, 03 2024.

- [44] J. S. Espinosa and M. P. Stryker. Development and plasticity of the primary visual cortex. *Neuron*, 75(2):230–249, 2012.
- [45] Astrid A. Zeman, James B. Ritchie, Simone Bracci, Christos Pantelis, Nikolaus Weiskopf, and Colin I. Baker. Orthogonal representations of object shape and category in deep convolutional neural networks and human visual cortex. *Scientific Reports*, 10(1), 2020.
- [46] David H. Hubel and Torsten N. Wiesel. Receptive fields of single neurones in the cat’s striate cortex. *The Journal of Physiology*, 148(3):574–591, 1959.
- [47] Anitha Pasupathy and Charles E Connor. Responses to contour features in macaque area v4. *Journal of neurophysiology*, 82(5):2490–2502, 1999.
- [48] Pulkit Agrawal, Jitendra Malik, Dustin Stansbury, and Jack Gallant. Convolutional neural networks mimic the hierarchy of visual representations in the human brain. 2017.
- [49] Keiron O’Shea and Ryan Nash. An introduction to convolutional neural networks. *CoRR*, abs/1511.08458, 2015.
- [50] Laith Alzubaidi, Jinglan Zhang, Amjad J. Humaidi, Ayad Al-Dujaili, Ye Duan, Omran Al-Shamma, J. Santamaría, Mohammed A. Fadhel, Muthana Al-Amidie, and Laith Farhan. Review of deep learning: concepts, cnn architectures, challenges, applications, future directions. *Journal of Big Data*, 8(1):53, Mar 2021.
- [51] Xiaohan Ding, Xiangyu Zhang, Yizhuang Zhou, Jungong Han, Guiguang Ding, and Jian Sun. Scaling up your kernels to 31x31: Revisiting large kernel design in cnns, 2022.
- [52] Chunyu Xu and Hong Wang. Research on a convolution kernel initialization method for speeding up the convergence of cnn. *Applied Sciences*, 12:633, 01 2022.
- [53] Florentin Bieder, Robin Sandkühler, and Philippe C. Cattin. Comparison of methods generalizing max- and average-pooling, 2021.