UNIVERSIDAD DE BURGOS
ESCUELA POLITÉCNICA SUPERIOR
Grado en Ingeniería Informática

TFG del Grado en Ingeniería Informática

AI-Music

Presentado por José Ángel López Estrada
en Universidad de Burgos — 23 de junio de 2023
Tutores: César Ignacio García Osorio, Alicia Olivares Gil

UNIVERSIDAD DE BURGOS
ESCUELA POLITÉCNICA SUPERIOR
Grado en Ingeniería Informática

D. nombre tutor, profesor del departamento de nombre departamento, área de nombre área.

Expone:

Que el alumno D. José Ángel López Estrada, con DNI 21071560H, ha realizado el Trabajo final de Grado en Ingeniería Informática titulado título de TFG.

Y que dicho trabajo ha sido realizado por el alumno bajo la dirección del que suscribe, en virtud de lo cual se autoriza su presentación y defensa.

En Burgos, 23 de junio de 2023

Vº. Bº. del Tutor:	Vº. Bº. del co-tutor:
D. nombre tutor	D. nombre co-tutor

Resumen

El rápido crecimiento del consumo de música digital ha generado una gran cantidad de datos de audio, lo que ha priorizado la necesidad de contar con métodos de clasificación y categorización. Plataformas como *Spotify* o *Apple Music* con bibliotecas que superan los 100 millones de canciones y un crecimiento constante de decenas de miles de canciones diariamente, resaltan la importancia de contar con una clasificación musical precisa y eficiente.

En este proyecto, se utilizan métodos de inteligencia artificial (IA) con el fin de clasificar de forma automática diferentes géneros musicales a partir de un análisis de las características del audio.

La aplicación desarrollada en el proyecto se presenta como una plataforma web que permite a los usuarios cargar y analizar sus canciones.

Descriptores

machine learning, big data, inteligencia artificial, clasificación, música, python, flask, servidor web ...

Abstract

The rapid growth of digital music consumption has generated a large amount of audio data, which has prioritized the need for classification and categorization methods. Platforms like *Spotify* or *Apple Music*, with libraries exceeding 100 million songs and a constant growth of tens of thousands of songs daily, highlight the importance of having accurate and efficient music classification.

In this project, artificial intelligence (AI) methods are used to automatically classify different music genres based on an analysis of audio features.

The application developed in the project is presented as a web platform that allows users to upload and analyze their songs.

Keywords

machine learning, big data, artificial intelligence, classification, music, python, flask, web server

Índice general

Índice de figuras

Índice de tablas

Introducción

1.1. Contexto

La música es un importante elemento cultural con una gran variedad de estilos y géneros. La identificación automática de estilos musicales es un campo de estudio emergente, cuyo objetivo es crear sistemas inteligentes capaces de reconocer y clasificar automáticamente el estilo musical de una canción o pieza musical sin intervención humana.

Este ámbito de estudio tiene importantes aplicaciones en áreas como la recomendación musical o la organización del audio digital.

La aplicación de algoritmos de inteligencia artificial ha mostrado un éxito considerable en este ámbito.

El presente trabajo tiene como objetivo desarrollar un sistema de reconocimiento automático de estilos musicales utilizando procesamiento de audio y algoritmos de inteligencia artificial.

1.2. Estructura del proyecto

El presente documento contiene la siguiente estructura.

- **Introducción**: contexto, descripción del problema, estructura del proyecto y materiales relevantes.
- **Objetivos del proyecto**: explicación y exposición de los objetivos del proyecto, tanto generales como técnicos.
- **Conceptos teóricos**: breve explicación de los conceptos teóricos principales que son necesarios para entender el problema a resolver.

- **Técnicas y herramientas**: presentación de la metodología, técnicas y herramientas utilizadas en el desaroollo del proyecto.
- **Aspectos relevantes del desarrollo del proyecto**: aspectos importantes en el desarrollo del proyecto.
- **Trabajos relacionados**: estado del arte y trabajos relacionados en el área de estudio.
- **Conclusiones y líneas de trabajo futuras**: conclusiones obtenidas tras la finalización del proyecto y posibles líneas de trabajo futuras.

Además, el documento*anexos* contiene la siguiente estructura:

- **Apéndice A. Plan de Proyecto Software**:
- **Apéndice B. Especificación de requisitos**:
- **Apéndice C. Especificación de diseño**:
- **Apéndice D. Documentación técnica de programación**:
- **Apéndice E. Documentación de usuario**:

1.3. Materiales

- **Repositorio**: https://github.com/jle1001/AI-Music
- **Aplicación**:

Objetivos del proyecto

2.1. Objetivos principales

- Desarrollar un sistema de reconocimiento de estilos musicales utilizando inteligencia artificial.
- Diseñar e implementar una aplicación web que permita usar el modelo de una forma sencilla.
- Obtener conclusiones y conocimiento a partir de los datos.

2.2. Objetivos técnicos

- Desarrollar un sistema de *aprendizaje automático* que, utilizando pistas de audio, detecte de forma automática el estilo musical basándose en diversas características.
- Evaluar el modelo con diversas métricas como *accuracy*, *precision* o *F1*.
- Implementar tests unitarios y de integración.
- Desarrollar un aplicación web utilizando librerías como Flask.
- Desplegar la aplicación con el modelo entrenado a una plataforma online como Heroku.
- Utilizar un sistema de control de versiones como Git para poder seguir los cambios en el código.
- Seguir una metodología ágil como SCRUM para la gestión del desarrollo del proyecto.

Conceptos teóricos

Antes de empezar con el desarrollo del proyecto, es necesario explicar una serie de conceptos teóricos.

3.1. Sonido

El sonido es una vibración mecánica que se propaga a través de un medio elástico, como el aire, el agua o cualquier otro material. Estas vibraciones generan diferencias de presión en el medio, que son captadas por nuestros oídos y percibidas como sonido.

Matemáticamente, el sonido se puede representar mediante una función matemática $f(t)$, donde t representa el tiempo. Esta función describe cómo varía la presión o desplazamiento de partículas en el medio a medida que pasa el tiempo.

Representación Matemática del Sonido

La representación matemática más común del sonido es la onda sinusoidal. Una onda sinusoidal se puede describir mediante la siguiente ecuación:

$$f(t) = A\sin(2\pi ft + \phi) \tag{3.1}$$

Donde:

- A es la amplitud de la onda, que representa la máxima desviación de la onda desde su posición de equilibrio.

- f es la frecuencia de la onda, que determina la cantidad de ciclos completos que la onda realiza en un segundo.
- t es el tiempo.
- ϕ es la fase inicial de la onda, que determina el desplazamiento horizontal de la onda en el tiempo.

3.2. Reconocimiento musical

El reconocimiento musical es un área que se centra en el análisis de las características del audio, para así, extraer información relevante. Por ejemplo la identificación de canciones, géneros musicales, o artistas.

Características del Sonido

El reconocimiento musical se basa en el análisis de diversas características del sonido. Algunas de las características más comunes son:

- **Ritmo**: El ritmo es una de las propiedades fundamentales de la música y se refiere a la organización temporal de los eventos sonoros. Su complejidad abarca desde un ritmo repetitivo a un ritmo con notas y silencios complejos. El ritmo se puede analizar en términos de su complejidad, regularidad, velocidad y otros aspectos. En algunos géneros de música, como el techno, el ritmo suele ser el aspecto más dominante. La importancia del ritmo en el reconocimiento musical puede involucrar la detección del tempo y el análisis de los patrones rítmicos para obtener conclusiones.
- **Frecuencia**: La frecuencia musical es el número de vibraciones u oscilaciones por segundo en el sonido. La frecuencia de estas vibraciones se mide en ciclos por segundo, o Hertz (Hz). Mientras mayor sea la frecuencia más agudo es el tono, y cuando menor más grave. La frecuencia musical se puede utilizar para identificar notas específicas o para determinar la armonía.
- **Timbre**: El timbre se refiere a las características tonales y armónicas que distinguen diferentes instrumentos y voces. Dos instrumentos musicales pueden reproducir la misma nota pero sonar de una manera completamente diferente debido a su timbre. El análisis del timbre puede ayudar a reconocer estilos musicales debido a que permite identificar los diferentes instrumentos implicados en una canción.

- **Estructura Musical**: La estructura musical se refiere a la organización global de una composición musical. Estudiando la estructura musical se pueden distinguir las diferentes partes de una canción como el estribillo, versos o secciones instrumentales. De esta manera se pueden encontrar patrones repetitivos y que se ajusten a estilos musicales concretos.

Estilos Musicales y Reconocimiento

Como se ha visto en el apartado anterior, los estilos musicales poseen características que se pueden utilizar para realizar una clasificación. Algunos géneros musicales tienen una estructura musical ampliamente establecida. Por ejemplo una gran cantidad de la música pop tiene una estructura definida como:

Verso - Estribillo - Verso - Estribillo - Puente - Estribillo

Este tipo de características o patrones comunes pueden ser utilizados por modelos de aprendizaje automático para aprender de los datos y extraer conclusiones.

Aplicaciones del Reconocimiento Musical

El reconocimiento musical tiene diversas aplicaciones prácticas y de gran uso en la actualidad, algunas de las cuales son:

- **Recomendación de música**: Los algoritmos de reconocimiento musical se utilizan para recomendar música a los usuarios en función de sus preferencias y patrones de escucha. Estos sistemas analizan las características de las canciones más escuchadas por el usuario y crean un modelo de recomendación basado en sus gustos.

- **Clasificación de géneros musicales**: El reconocimiento musical se utiliza para clasificar automáticamente las canciones en diferentes géneros musicales, lo que facilita la organización y la búsqueda de música en grandes bibliotecas digitales.

3.3. Ejemplo teórico de extracción de características musicales

Espectrograma

Un espectrograma es una representación visual del espectro de frecuencia de una señal de audio en función del tiempo. Proporciona información detallada sobre cómo se distribuye la energía del sonido en diferentes frecuencias a lo largo del tiempo.

A continuación se explica el proceso para obtener un espectrograma de una canción.

- **Preprocesamiento de la señal de audio**: La señal de audio de la canción se divide en segmentos. De esta manera es posible analizar la variación espectral en diferentes puntos de la señal a lo largo del tiempo.
- **Transformada de Fourier de tiempo corto (STFT)**: Cada segmento de la señal se somete a una transformada de Fourier de tiempo corto (STFT). La STFT divide la señal en múltiples segmentos de tiempo y calcula la suma de diferentes frecuencias en cada segmento. Esto se logra mediante la aplicación de una ventana temporal a cada segmento y luego calculando la transformada de Fourier de cada ventana.
- **Cálculo de la magnitud del espectro**: La STFT proporciona diversa información sobre las fases y amplitudes de las frecuencias en cada segmento de tiempo. Sin embargo, para construir un espectrograma, generalmente se toma la magnitud del espectro (amplitud absoluta de las frecuencias).
- **Representación visual**: La magnitud del espectro se representa visualmente en un gráfico 2D, donde el eje horizontal representa el tiempo y el eje vertical representa las frecuencias. La intensidad del color o brillo en cada punto del gráfico indica la energía o amplitud de la frecuencia correspondiente.

Los espectrogramas tienen una gran utilidad no solo en el estudio de la música. Por ejemplo, se pueden utilizar para visualizar las señales de un radar y detectar objetos o en lingüistica estudiando los patrones de frecuencia de los sonidos del habla y así estudiar los diferentes fonemas o acentos.

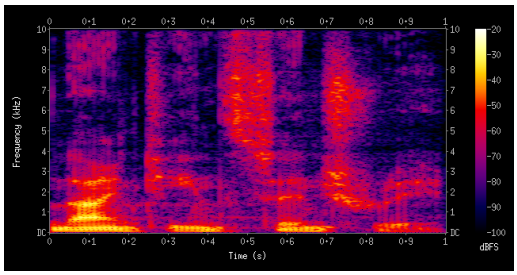

Figura 3.1: Espectrograma de una pista de audio.

Coeficientes Cepstrales de Frecuencias de Mel (MFCC)

Los coeficientes cepstrales de frecuencias de Mel (MFCC) son características ampliamente utilizadas en el procesamiento de señales de audio y el reconocimiento de voz. Estos coeficientes representan las características espectrales de una señal de audio en función de la escala de Mel, que es una escala perceptual de frecuencia basada en la respuesta del oído humano.

A continuación se explica el proceso para obtener los coeficientes MFCC de una pista de audio.

1. **Preénfasis**: La señal de audio se normaliza con un filtro de preénfasis, que resalta las frecuencias de alta frecuencia y compensa la atenuación de las frecuencias más bajas. Esto ayuda a mejorar la relación señal-ruido y realzar las características relevantes en el espectro.
2. **División en tramas**: La señal preénfasis se divide en tramas o segmentos cortos y superpuestos en el tiempo. Esto se hace para capturar la variación espectral en diferentes puntos de la señal a lo largo del tiempo.
3. **Cálculo de la Transformada de Fourier de tiempo corto (STFT)**: A cada trama de la señal se le aplica la Transformada de Fourier de tiempo corto (STFT), que calcula la contribución de diferentes frecuencias en cada trama. La STFT proporciona diversa información sobre las fases y amplitudes de las frecuencias en cada segmento de tiempo.
4. **Banco de filtros de Mel**: Se aplica un banco de filtros de Mel, que consiste en una serie de filtros triangularmente espaciados en la escala de Mel. Estos filtros se utilizan para representar el espectro en términos de bandas de frecuencia de Mel.

5. **Logaritmo de la energía**: Se calcula el logaritmo de la después de aplicar el banco de filtros de Mel. Esto se hace para tener en cuenta la respuesta no lineal del oído humano a las frecuencias.
6. **Transformada de Coseno Discreta**: Se aplica la Transformada de Coseno Discreta (DCT) a los valores obtenidos anteriormente.
7. **Extracción de los coeficientes MFCC**: Finalmente, se seleccionan los coeficientes cepstrales más significativos para representar la información espectral de la señal de audio. Estos coeficientes son los utilizados como características para aplicaciones de procesamiento y reconocimiento de audio.

Los coeficientes MFCC son ampliamente utilizados en aplicaciones como:

- Reconocimiento de voz: Como los MFCC son una forma de reducir la complejidad de la voz a estructuras más simples y facilmente procesables, son utilizados para el reconocimiento de voz. Por ejemplo asistentes de audio como Siri o Alexa utilizan (junto a otros sistemas) MFCCs.
- Identificación de hablantes: Los MFCC se pueden utilizar para identificar voces de diferentes hablantes, debido a que cada persona tiene una forma característica de hablar.
- Clasificación de audio: Se pueden utilizar para identificar diferentes sonidos o instrumentos músicales en una pista de audio y de esta manera obtener información relevante que permita realizar una clasificación.

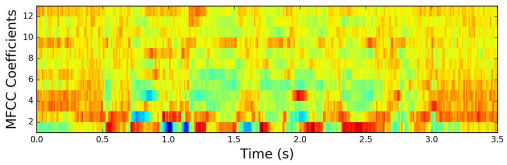

Figura 3.2: MFCC de una pista de audio.

3.4. Inteligencia Artificial

La inteligencia artificial (IA) es la capacidad de un sistema informático de imitar funciones cognitivas humanas, como el aprendizaje y la solución de problemas

Los sistemas de inteligencia artificial pueden analizar grandes cantidades de datos, reconocer patrones y tomar decisiones basadas en esa información. Pueden aprender de la experiencia y mejorar su rendimiento con el tiempo.

Hay diferentes enfoques en la IA, incluyendo el aprendizaje automático (*machine learning*), el procesamiento del lenguaje natural (*natural language processing*), la visión por computador (*computer vision*) y la robótica, entre otros.

La IA se utiliza en una amplia variedad de aplicaciones cómo en sistemas de recomendación, análisis de datos o diagnósticos médicos por ejemplo.

Aprendizaje automático (Machine Learning)

El aprendizaje automático (*machine learning*) es un subcampo de la inteligencia artificial que se centra en el desarrollo de algoritmos y modelos que permiten a los sistemas aprender y extraer información a partir de datos, sin ser explícitamente programados para ello. Existen diversos tipos de aprendizaje automático:

- **Aprendizaje supervisado**: Se proporciona como entrada a los algoritmos un conjunto de datos de entrenamiento etiquetados. El modelo aprende a realizar predicciones o tomar decisiones basadas en estos ejemplos etiquetados. El aprendizaje supervisado se utiliza en tareas de clasificación o regresión.
- **Aprendizaje no supervisado**: Los algoritmos trabajan con conjuntos de datos no etiquetados, es decir, sin clase conocida. El objetivo es encontrar patrones u estructuras ocultas en los datos. El aprendizaje no supervisado se utiliza en tareas como el (*clustering*).
- **Aprendizaje por refuerzo**: En este tipo de aprendizaje, un agente inteligente interactúa con su entorno y aprende a tomar decisiones según una serie de recompensas o penalizaciones. El objetivo es encontrar una política de actuación que maximize las recompensas recibidas. El aprendizaje por refuerzo es utilizado para entrenar agentes en videojuegos o en robótica.

- **Aprendizaje semi-supervisado**: En este tipo de aprendizaje el conjunto de datos no está completamente etiquetado, por lo tanto, el objetivo es maximizar el rendimiento del modelo a partir de los datos con clase conocida.

3.5. Ejemplo de aprendizaje supervisado

En este proyecto se va a utilizar un enfoque de IA utilizando aprendizaje supervisado. Por lo que se va a detallar un ejemplo simple para entender su funcionamiento:

Objetivo

Construir un modelo de aprendizaje automático para clasificar correos electrónicos como "spam.º "no spam".

Conjunto de datos

Conjunto de datos etiquetados que contiene 1000 correos electrónicos, donde cada correo tiene características como la frecuencia de ciertas palabras sospechosas de pertenecer a spam, la longitud del mensaje, la presencia de enlaces o imágenes, entre otros. **Además de incluir la clase a la que pertenece: "Spam.º "No Spam"**.

Preparación de los datos

Se deben preparar los datos para el entrenamiento del modelo. Una opción adecuada es representar cada correo electrónico como un vector de características.

Correo	**Palabras sospechosas**	**Longitud**	**Enlaces**	**Etiqueta**
1	1	120	0	No spam
2	4	56	1	Spam
3	1	352	2	No spam
4	9	174	0	Spam

Tabla 3.1: Ejemplo de datos de entrenamiento en un problema de aprendizaje supervisado

Entrenamiento del modelo

Una vez preparados los datos en un formato adecuado, estos son introducidos en un algoritmo de aprendizaje supervisado formando un modelo de aprendizaje automático. Por ejemplo Redes Neuronales Artificiales, Máquinas de Soporte Vectorial o Árboles de decisión.

Predicción

Una vez entrenado el modelo, se alimenta con datos externos y se realizan predicciones.

El algoritmo utilizado en el proyecto es más complejo y será explicado en detalle en las siguientes secciones.

3.6. Redes neuronales

Para realizar el entrenamiento del modelo de aprendizaje automático se han utilizado redes neuronales. Las redes neuronales son una categoría de modelos de aprendizaje automático inspirados en la estructura biológica del cerebro humano. Se componen de nodos interconectados, o "neuronas", organizadas en capas, que trabajan juntas para procesar información y hacer predicciones.

Las redes neuronales se componen de los siguientes elementos:

- **Neuronas**: entidad matemática inspirada en la neurona biológica. Es el componente básico de una red neuronal.
- **Capas**: las neuronas están organizadas por capas. Capa se refiere a un conjunto de neuronas que procesan información al mismo tiempo. Existen tres tipos de capas: capa de entrada, capas ocultas y capa de salida. Cada neurona en una capa está conectada a todas las neuronas en la capa anterior y a todas las neuronas en la capa siguiente.
- **Proceso de entrenamiento**: el proceso de entrenamiento de una red neuronal consiste en ajustar los pesos y sesgos de las neuronas de tal manera que el error entre las predicciones y los valores reales sea lo más pequeño posible. Existen diversos algoritmos para realizar este cálculo iterativo de los valores de cada neurona.

Neuronas artificiales

Una neurona artificial es una entidad matemática inspirada en una neurona biológica. Se utilizan para modelar la información que una red recibe, procesa y luego utiliza para tomar decisiones. Al igual que las neuronas en el cerebro humano, una neurona artificial toma una serie de entradas y produce una salida. Cada entrada tiene un peso asociado, que ajusta el valor de esa entrada para la salida final. Los pesos y valores se ajustan durante el proceso de entrenamiento para mejorar el rendimiento de la red.

La neurona artificial suma las entradas ponderadas y luego aplica una función de activación para producir la salida. Las funciones de activación son necesarias para introducir no linealidad en la red, lo que permite que la red aprenda a partir de datos más complejos.

Hay varios tipos de neuronas diferenciadas por su función de activación. Algunos ejemplos podrían ser:

Neurona umbral: esta neurona aplica una función de umbral a las entradas. Si la suma ponderada de las entradas supera un cierto umbral, la neurona se activa.

La función de activación umbral se define de la siguiente manera:

$$f(x) = \begin{cases} 0, & \text{si } x < \text{umbral} \\ 1, & \text{si } x \geq \text{umbral} \end{cases}$$

Donde umbral es el valor de umbral que determina el punto de corte para la activación de la neurona.

Neurona sigmoidal: la neurona sigmoidal aplica una función sigmoide a las entradas

La función de activación sigmoidal se define de la siguiente manera:

$$f(x) = \frac{1}{1 + e^{-x}}$$

Donde x es el valor de entrada de la neurona.

Neurona ReLU *(Rectified Linear Unit)*: este tipo de neurona utiliza la función ReLU como su función de activación. Este tipo de función de activación introduce no linealidad sin los problemas que pueden surgir con otras funciones de activación.

La función de activación ReLU se define de la siguiente manera:

$$f(x) = \begin{cases} 0, & \text{si } x < 0 \\ x, & \text{si } x \geq 0 \end{cases}$$

Donde x es el valor de entrada de la neurona.

Neurona softmax: esta neurona se utiliza en la capa de salida para problemas de clasificación multiclase. La función Softmax toma un vector de entradas y produce un vector de salidas, cada una de las cuales está entre 0 y 1 y la suma total es 1. Cada salida puede interpretarse como la probabilidad de que la entrada pertenezca a una clase particular.

$$f(x_i) = \frac{e^{x_i}}{\sum_{j=1}^{N} e^{x_j}}$$

Donde x_i es el valor de entrada de la neurona y N es el número total de entradas.

Capas

Una red neuronal artificial está compuesta por un conjunto de capas neuronales interconectadas. Cada capa consiste en un conjunto de neuronas las cuales reciben información de la capa anterior y envía su salida a las neuronas en la capa siguiente.

Capa de entrada: es la primera capa de la red. Cada neurona en la capa de entrada corresponde a una característica en el conjunto de datos de entrada.

Capas ocultas: capas que se encuentran entre la capa de entrada y la capa de salida. Su número varía dependiendo de la profundidad de la red. En estas capas es donde se produce la mayor parte del cálculo de la red. Los pesos de las conexiones entre las neuronas en las capas ocultas se ajustan durante el entrenamiento.

Capa de salida: última capa de la red. Es la capa donde se devuelve la información. El número de neuronas en la capa de salida suele estar determinado por el tipo de problema que se está resolviendo. Por ejemplo, en un problema de clasificación binaria, solo se necesitaría una neurona de salida. En un problema de clasificación de varias clases, el número de neuronas de salida sería igual al número de clases.

Tipos de capas

Capas densas o totalmente conectadas: cada neurona está conectada a todas las neuronas en la capa anterior y a todas las neuronas en la capa siguiente.

Capas convolucionales: capas formadas por redes neuronales convolucionales. En lugar de conectarse a todas las neuronas de la capa anterior, una neurona en una capa convolucional está conectada a un pequeño subconjunto de las neuronas en la capa anterior.

Capas recurrentes: capas formadas por redes neuronales recurrentes. En una capa recurrente, las neuronas tienen conexiones de retroalimentación con ellas mismas a lo largo del tiempo.

Capas de pooling: capas utilizadas para reducir la dimensionalidad de los datos. Una capa de pooling toma un subconjunto de las entradas

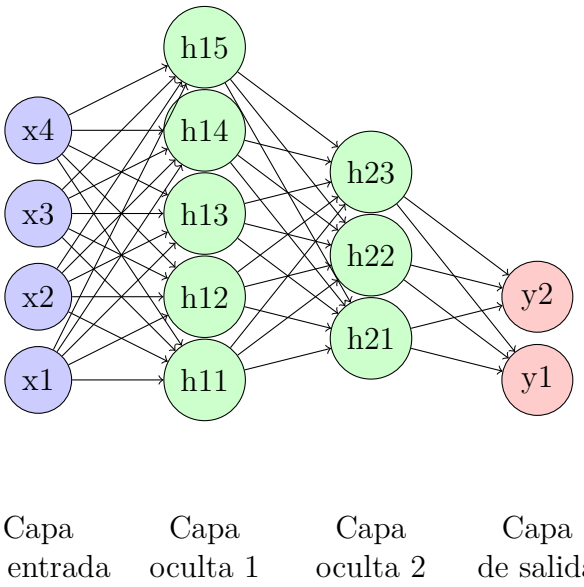

Figura 3.3: Red neuronal con dos capas ocultas

y produce una sola salida, que es una medida estadística resumida de sus entradas (por ejemplo, el máximo o el promedio).

Capas de normalización: capas utilizadas para normalizar las entradas a una red o las salidas de una capa anterior. La normalización ayuda con el proceso de entrenamiento de la red.

3.7. Proceso de entrenamiento

El entrenamiento de una red neuronal implica un proceso que, de forma iterativa, ajusta los pesos de cada una de las neuronas para minimizar la diferencia entre las salidas predichas y las salidas reales del conjunto de datos de entrenamiento. Este proceso se puede clasificar en varios pasos:

Inicialización de pesos: el primer paso en el entrenamiento es inicializar los pesos de las neuronas de la red. Normalmente se inicializan de manera aleatoria.

Propagación hacia adelante: la propagación hacia adelante es el proceso por el cual la red neuronal transmite la información desde la capa de entrada hasta la capa de salida mediante la información calculada en cada función de activación neuronal.

Función de perdida: proceso de medición de la calidad de las predicciones. La función de perdida (o función de coste) genera un valor indicando cuanta distancia existe entre las predicciones y las clases verdaderas. Existen distintos tipos de funciones de perdida, en el caso de este proyecto la función utilizada es *Sparse Categorical Crossentropy*.

Backpropagation: proceso de ajuste de los pesos de la red neuronal en función al error calculado en la función de perdida. El objetivo es minimizar iterativamente la función de perdida propagando los errores desde la capa de salida hacia las capas ocultas por medio de la regla de la cadena. De esta manera se puede descubrir que número de neuronas deben actualizar sus pesos.

Iteración: todo este proceso se repite N veces. Cada proceso completo de inicialización, propagación hacia adelante, función de coste y backpropagation se llama época (*epoch*). Normalmente se ejecutan varias epocas hasta que la función de perdida se estabiliza.

3.8. Evaluación del modelo

La evaluación de un modelo de red neuronal es el proceso donde se determina la efectividad del modelo. Este proceso debe realizarse durante y después del entrenamiento.

Evaluación con el conjunto de prueba: tras entrenar el modelo es imprescindible realizar una evaluación final con el conjunto de prueba. Esta evaluación da como resultado una evaluación más precisa de como se comportará el modelo con datos externos.

Cálculo de métricas de rendimiento

Existen diversas métricas de rendimiento que indican como de bueno es el entrenamiento del modelo. Algunas de las más usadas son:

Accuracy: proporción de predicciones correctas que hace el modelo en relación con todas las predicciones que hace.

$$\text{Accuracy} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{FP} + \text{FN} + \text{TN}} \tag{3.2}$$

Recall: recall es la proporción de instancias positivas que el modelo predice correctamente en relación con todas las instancias positivas reales.

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}} \tag{3.3}$$

F1-Score: media armónica de precisión y recall.

$$\text{F1-Score} = 2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}} \tag{3.4}$$

MSE: es una métrica comúnmente usada para problemas de regresión. Es el promedio de los cuadrados de las diferencias entre las predicciones del modelo y las clases verdaderas.

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2 \tag{3.5}$$

Técnicas y herramientas

Esta parte de la memoria tiene como objetivo presentar las técnicas metodológicas y las herramientas de desarrollo que se han utilizado para llevar a cabo el proyecto.

4.1. Herramientas utilizadas

- **Python**: Python es un lenguaje de programación de alto nivel ampliamente utilizado en el campo del aprendizaje automático y la inteligencia artificial. Esto es debido, en parte, a que posee una gran cantidad de bibliotecas y frameworks que facilitan el procesamiento de datos y la implementación de algoritmos de inteligencia artificial. En este proyecto, Python se ha utilizado para la implementación del back-end. Se han aprovechado su conjunto de bibliotecas para realizar el desarrollo Web (Flask), procesamiento de datos (Librosa, NumPy, Pandas) y métodos de aprendizaje automático (TensorFlow y Scikit-learn).
- **librosa**: librosa es una biblioteca de Python utilizada para el análisis y procesamiento de audio. Proporciona de una manera sencilla el acceso a métodos y funciones para realizar análisis, extracción y visualización de características de audio.
- **TensorFlow**: TensorFlow es una biblioteca de código abierto utilizada en el campo del aprendizaje automático. Proporciona la posiblidad de implementar redes neuronales complejas de una forma sencilla con grandes volúmenes de datos.
- **scikit-learn**: scikit-learn es una biblioteca de aprendizaje automático de Python que proporciona una amplia gama de algoritmos y herramientas para el análisis de datos y la construcción de modelos.

Incluye funciones para la división de conjuntos de datos en conjuntos de entrenamiento (*train* y *prueba*) que han sido utilizadas en este proyecto.
- **NumPy**: NumPy es una biblioteca de Python utilizada para realizar cálculos numéricos en matrices y matrices multidimensionales.
- **Pandas**: bibliteca de código abierto en Python que proporciona herramientas de análisis de datos.
- **Flask**:

4.2. Justificación de las herramientas utilizadas

Lenguaje de programación

Lenguajes de programación planteados: Python, Go, C

Lenguajes de programación elegido: Python

A continuación se presentan algunas justificaciones que han decantado el desarrollo del proyecto en lenguaje Python en comparación con otros lenguajes de programación.

Integraciones: Python cuenta con una completa integración con la mayoría de bibliotecas y frameworks especializados en aprendizaje automático, como TensorFlow, scikit-learn, PyTorch, Keras entre otros. En el caso de C, aunque existen algunas bibliotecas como *mlpack*, *libsvm* o una API para controlar TensorFlow (*libtensorflow*) su comunidad de aprendizaje automático es bastante más pequeña que la de Python, por lo que hay menos soporte y recursos disponibles. En cuanto a Go, su implementación en este proyecto hubiera sido muy interesante debido a ser un lenguaje moderno, rápido y muy popular, pero ocurre lo mismo que con C. Su soporte para aprendizaje automático y procesamiento de datos es pequeña por lo que su suporte y recursos son escasos. Esto ha sido vital para elegir Python por encima del resto de lenguajes planteados.

Rendimiento: En este apartado los ganadores claros son C y Go, al tratarse de lenguajes compilados, pero en este proyecto el rendimiento bruto no es la prioridad como pueden ser la implementación de algoritmos de aprendizaje automático o el tratamiento de datos. Por eso mismo se ha elegido Python teniéndo en cuenta que aunque es el que menor rendimiento tiene, es el que otorga un mayor valor al proyecto.

Biblioteca de aprendizaje automático

Bibliotecas planteadas: TensorFlow, Pytorch, Scikit-learn

Bibliotecas elegidas: TensorFlow y Scikit-learn (como herramienta de partición de datos).

Integraciones: tanto TensorFlow, Pytorch como Scikit-learn ofrecen excelentes integraciones con Python. Por lo que este apartado no ha sido importante para valorar la elección entre uno u otro.

Enfoque y alcance: Tanto TensorFlow como Pytorch son bibliotecas de aprendizaje automático de propósito general con soporte para aprendizaje profundo y grandes volumenes de datos. En cambio Scikit-learn es una biblioteca de aprendizaje automático centrada en algoritmos tradicionales y no es adecuado para grandes volúmenes de datos. Por estas razones se descarta Scikit-learn como una opción viable para entrenar el modelo planteado en este proyecto, aunque se ha utilizado como biblioteca de métodos de procesamiento del conjunto de datos mediante sus herramientas de partición de conjunto de datos (*train_test_split*).

Rendimiento: tanto TensorFlow como Pytorch poseen soporte para entrenamiento con GPU o TPU, por lo que su rendimiento es muy similar. En este caso la elección ha sido utilizar TensorFlow debido a su mayor documentación y posibilidad de configuración.

Framework web

Frameworks web planteados: Flask, Django

Framework web elegido: Flask

Rendimiento: Flask y Django poseen un rendimiento similar en la mayoría de ocasiones aunque es posible, que en ciertas ocasiones, Flask sea algo más rápido debido a su naturaleza.

Enfoque: Flask sigue una filosofía más simple y minimalista en su desarrollo. Esto ayuda a que su implementación en el proyecto sea más sencilla y *directa*. En cambio Django sigue un patrón de diseño (MVC) más estricto y con muchas funciones que en este proyecto no han sido necesarias. Por estas razones se ha escogido Flask como la opción para implementar la aplicación web.

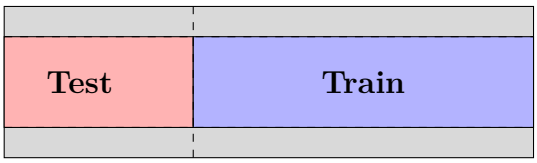

Figura 4.1: División del conjunto de datos

4.3. Técnicas utilizadas

Se han utilizado diversas técnicas y metodologías para llevar a cabo el desarrollo y entrenamiento de los modelos de aprendizaje automático.

(En el documento *Anexos* se describe de una forma mucho más profunda el desarrollo metodológico del proyecto.)

4.4. Procesamiento y extracción de características de audio

Para el procesamiento y extracción de características de audio, se ha utilizado la biblioteca librosa en Python. Cómo se ha mencionado anteriormente, librosa permite el análisis y procesamiento de audio de una manera sencilla y se ha utilizado para extraer las características de audio para después alimentar al algoritmo de aprendizaje automático.

4.5. División del conjunto de datos

Para dividir el conjunto de datos en conjuntos de entrenamiento, prueba y validación, se ha utilizado la biblioteca scikit-learn. Scikit-learn proporciona una función llamada *train_test_split* que permite dividir el conjunto de datos en partes destinadas para el entrenamiento y la evaluación del modelo. Esta técnica de división del conjunto de datos es fundamental para evaluar la capacidad de **generalización** del modelo y evitar el **sobreajuste**.

4.6. Redes neuronales con TensorFlow

Para realizar el entrenamiento de los datos, se han utilizado redes neuronales implementadas con la biblioteca TensorFlow.

En este proyecto, se han utilizado diferentes arquitecturas de redes neuronales, como redes neuronales convolucionales (CNN) y redes neuronales multicapa. Estas arquitecturas son adecuadas para tareas de procesamiento de audio y han demostrado ser efectivas en la clasificación y reconocimiento de patrones en audio.

Las redes neuronales se entrenan actualizando los pesos y los sesgos de la red iterativamente para minimizar la pérdida (*loss*). Una vez entrenadas, las redes neuronales realizan predicciones sobre nuevos datos de audio, clasificándolos en categorías o estilos musicales.

Aspectos relevantes del desarrollo del proyecto

5.1. Desarrollo del Proyecto

El desarrollo de este proyecto se ha llevado a cabo utilizando una metodología ágil basada en Scrum. Este enfoque proporciona flexibilidad necesaria para adaptar cambios en los requerimientos y para mejorar iterativamente el proyecto durante su desarrollo. Las metodología ágiles se basan en la idea de dividir el proyecto en ciclos iterativos llamados "sprints", donde se llevan a cabo actividades de planificación, desarrollo, pruebas y revisión. Estos sprints suelen tener una duración fija, en este caso han sido de 1 semana con alguna excepción justificada.

Metodología Scrum

En el documento *Anexos* se explica en detalle el proceso de desarrollo del proyecto.

En resumen el desarrollo ha consistido en los siguientes pasos iterativos:

1. **Planificación inicial**: Etapa de definición de objetivos generales del proyecto. Creación del Product Backlog.
2. **Reuniones de Sprint**: Al comienzo de cada sprint, se realiza una pequeña reunión para revisar las tareas realizadas y seleccionar las tareas a desarrollar.
3. **Desarrollo del sprint**: Durante el sprint, se trabaja en cada una de las tareas asignadas.

4. **Mejora continua**: La metodología ágil promueve la mejora continua y el aprendizaje a lo largo del desarrollo del proyecto.

Análisis

El objetivo del proyecto es aplicar técnicas de Inteligencia Artificial (IA) y Aprendizaje Automático (ML) para la clasificación de estilos musicales. Durante la fase de análisis, se estudiaron varios modelos de ML, eligiendo finalmente la Red Neuronal Convolucional (CNN) debido a su mayor eficacia.

Otro factor a tener en cuenta en un proyecto de IA es el conjunto de datos. Se han pensado diversos conjuntos de datos para entrenar el modelo como:

- **GTZAN Dataset**: conjunto de datos muy utilizado en la clasificación musical. Recopilado por George Tzanetakis en 2002, consta de **1000 fragmentos** de audio de **30 segundos**, distribuidos en **10 géneros musicales**.
- **Million Song Dataset**: conjunto de datos formado por las *características musicales y metadatos* de un millón de canciones populares. No incluye pistas de audio.
- **MagnaTagATune**: conjunto de datos formado por **25863 fragmentos** de audio de **29 segundos**.
- **FMA (Free Music Archive)**: conjunto de datos musicales, libre de derechos, constituido por **106574 fragmentos** de audio de **30 segundos**.

Finalmente se ha elegido el conjunto de datos **FMA (Free Music Archive) de 7.2 GiB**, con 8000 fragmentos de audio para realizar el entrenamiento del modelo. Esto es debido a que ofrece una gran cantidad de pistas musicales libres de derechos. Además, al incluir las pistas musicales (a diferencia de Million Song Dataset por ejemplo) es posible extraer las características de forma manual utilizando librosa u otras herramientas.

Diseño

En cuanto al diseño de la aplicación, en un primer momento se pensó en el desarrollo de una aplicación de escritorio. Sin embargo, esta idea fue descartada y se optó por desarrollar una aplicación web debido a las siguientes razones:

- **Evolución tecnológica**: Es una realidad que las aplicaciones web están dominando el momento tecnológico actual, por lo que realizar el proyecto en web es una buena manera de estar al día de esta tecnología.
- **Accesibilidad**: La aplicación web es accesible desde cualquier dispositivo con una conexión a internet, independientemente del sistema operativo. De esta manera se amplía el alcance de la aplicación.
- **Actualizaciones**: Cuando se actualiza una aplicación web, automaticamente los usuarios reciben la última versión disponible en producción, eliminando la necesidad de descargar e instalar actualizaciones manualmente.
- **Mantenimiento**: Generalmente es más sencillo mantener una aplicación web que una aplicación de escritorio, ya que factores como hardware o sistema operativo del usuario desaparecen.

El marco de trabajo escogido para realizar la implementación de la aplicación web ha sido:

- **Backend**: Para el backend se ha utilizado Flask, un marco de trabajo de Python, pensado para realizar aplicaciones web y APIs.
- **Frontend**: Para el frontend, se ha optado por HTML y CSS. HTML es un lenguaje de marcado que define la estructura y contenidos de las páginas web que conforman la aplicación. CSS es utilizado para definir los estilos de los documentos HTML y añadir elementos atractivos para el usuario.

5.2. Extracción de características de audio

En este sección se va a analizar el proceso de extracción de características de audio.

Extracción de MFCC usando Python y librosa

El método de extracción de MFCC está diseñado para recorrer todos los archivos en un directorio especificado y extraer las características MFCC de cada archivo de audio que encuentre. La explicación de los detalles relevantes es la siguiente:

```
y, sr = librosa.load(item)

if librosa.get_duration(y=y, sr=sr) < 25:
continue

y = y[:(25 * sr)]
mfcc = librosa.feature.mfcc(y=y, sr=sr, n_mfcc=10)
mfcc_normalized = (mfcc - np.mean(mfcc)) / np.std(mfcc)

mfccs[int(item.stem)] = mfcc_normalized.ravel()
```

1. **y, sr = librosa.load(item)**: Carga del archivo de audio, devolviendo tanto la señal de audio (y) como la frecuencia de muestreo (sr).
2. **if librosa.get_duration (y=y, sr=sr) < 25**: Comprobación de la duración del audio. Si el audio es inferior a 25 segundos se omite el archivo y se continúa con el siguiente.
3. **y = y[:(25 * sr)]**: Recorte de los primeros 25 segundos de la señal de audio.
4. **mfcc = librosa.feature.mfcc(y=y, sr=sr, n_mfcc =10)**: Extracción de 10 coeficientes MFCC.
5. **mfcc_normalized = (mfcc - np.mean(mfcc)) / np.std(mfcc)**: Normalización de los coeficientes MFCC restando la media y dividiendo por la desviación estándar.
6. **mfccs[int(item.stem)] = mfcc_normalized.ravel()**: Los coeficientes MFCC normalizados se aplanan en una matriz 1D y se almacenan en un diccionario.

Extracción del resto de características

Para comparar el rendimiento y elegir el mejor conjunto de características de audio posibles para realizar la clasificación, se ha repetido el proceso con:

1. **Espectrogramas**: librosa.feature.melspectrogram(y=y, sr=sr)
2. **Cromagramas**: librosa.feature.chroma_stft(y=y, sr=sr)

5.3. Implementación de las redes neuronales

En esta sección se van a estudiar las redes neuronales utilizadas en el proyecto y su eficacia.

Modelo de Red Neuronal inicial

```
initial_model = tf.keras.Sequential([
        tf.keras.layers.Dense(512, activation='relu'),
        tf.keras.layers.Dropout(0.2),
        tf.keras.layers.Dense(512, activation='relu'),
        tf.keras.layers.Dropout(0.2),
        tf.keras.layers.Dense(512, activation='relu'),
        tf.keras.layers.Dense(164, activation='softmax')
])
```

Primer modelo genérico que se utilizó para estudiar la calidad del entrenamiento y sus predicciones. Este modelo específico es un ejemplo de una red neuronal totalmente conectada, que se configura como una secuencia de capas.

`tf.keras.layers.Dense(512, activation='relu')`

Primera capa del modelo. Cada neurona en esta capa está conectada a todas las neuronas en la capa anterior. Tiene 512 neuronas y usa la función de activación ReLU (Rectified Linear Unit). La función **ReLU** es una función de activación no líneal por lo que el modelo puede aprender patrones complejos.

`tf.keras.layers.Dropout(0.2)`

Capa de Dropout. Dropout es una técnica de regularización utilizada para evitar el sobreajuste. Durante el entrenamiento, aleatoriamente se desactivan algunas neuronas para evitar un ajuste excesivo a los datos de entrenamiento. En este caso se desactivan el 20 % de las neuronas.

`tf.keras.layers.Dense(512, activation='relu')`

`tf.keras.layers.Dropout(0.2)`:

Estas son la segunda capa oculta y la segunda capa de Dropout, respectivamente.

`tf.keras.layers.Dense(512, activation='relu')`

Esta es la tercera capa oculta del modelo. Al igual que las dos primeras capas ocultas, tiene 512 neuronas y utiliza la función de activación ReLU.

```
tf.keras.layers.Dense(164, activation='softmax')
```

Capa de salida del modelo. Tiene 164 neuronas, que corresponden al número de estilos musicales presentes en el dataset. La función de activación **Softmax** produce una distribución de la probabilidad entre las diferentes clases.

TODO: INTRODUCIR GRÁFICOS CON LOS RESULTADOS DEL ENTRENAMIENTO Y REPRESENTACIÓN DE LAS CAPAS

Modelo de Red Neuronal Convolucional 1D (CNN 1D)

```
conv_model = tf.keras.Sequential([
    tf.keras.layers.Reshape((10770, 1), input_shape=(None, 10770)),
    tf.keras.layers.Conv1D(64, 3, padding='same', activation='relu'),
    tf.keras.layers.Conv1D(64, 3, padding='same', activation='relu'),
    tf.keras.layers.BatchNormalization(),
    tf.keras.layers.MaxPooling1D(pool_size=2),

    tf.keras.layers.Conv1D(128, 3, padding='same', activation='relu'),
    tf.keras.layers.Conv1D(128, 3, padding='same', activation='relu'),
    tf.keras.layers.BatchNormalization(),
    tf.keras.layers.MaxPooling1D(pool_size=2),

    tf.keras.layers.Conv1D(256, 3, padding='same', activation='relu'),
    tf.keras.layers.Conv1D(256, 3, padding='same', activation='relu'),
    tf.keras.layers.BatchNormalization(),
    tf.keras.layers.MaxPooling1D(pool_size=2),

    tf.keras.layers.Conv1D(128, 3, padding='same', activation='relu'),
    tf.keras.layers.Conv1D(128, 3, padding='same', activation='relu'),
    tf.keras.layers.BatchNormalization(),
    tf.keras.layers.MaxPooling1D(pool_size=2),

    tf.keras.layers.Flatten(),
    tf.keras.layers.Dense(512, activation='relu'),
    tf.keras.layers.Dropout(0.5),
    tf.keras.layers.Dense(512, activation='relu'),
    tf.keras.layers.Dense(164, activation="softmax")
])
```

Las redes neuronales convolucionales son especialmente eficaces para la clasificación musical debido al reconocimiento de patrones locales, en el contexto de la música es especialmente relevante ya que puede traducirse en la capacidad de identificar patrones como ritmo o tonos de forma eficaz.

`tf.keras.layers.Reshape((10770, 1), input_shape=(None, 10770))`

Primera capa del modelo. Se cambia la forma de los datos de entrada a un formato aceptado por las capas convolucionales propuestas. En este caso, los datos de entrada se reforman a una matriz 2D de 10770 filas y 1 columna.

```
    tf.keras.layers.Conv1D(64, 3, padding='same', activation='relu')
tf.keras.layers.Conv1D(64, 3, padding='same', activation='relu')
```

Primeras capa convolucionales. La convolución se realiza con **64 filtros** y un **tamaño de kernel de 3**. La función de activación es la ReLU (Rectified Linear Unit).

```
tf.keras.layers.BatchNormalization()
```

Capa de normalización por lotes. Técnica necesaria para estabilizar y acelerar el proceso de entrenamiento. El funcionamiento consiste en aplicar una transformación que mantenga la salida media cercana a 0 y la desviación típica cercana a 1.

```
tf.keras.layers.MaxPooling1D(pool_size=2)
```

Capa de pooling. Reduce la dimensión espacial de la entrada extrayendo las características más importantes y así prevenir el sobreajuste.

Posteriormente se realizan los mismos pasos con diferentes capas de 128 neuronas, 256 neuronas y finalmente otras 128 neuronas. Hasta llegar a las últimas capas.

```
tf.keras.layers.Flatten()
```

Esta capa, reduce la dimensionalidad de la entrada a una entrada 1D. **Es el puente entre las capas convolucionales y las capas densas.**

```
tf.keras.layers.Dense(512, activation='relu')
```

Capa completamente conectada de 512 neuronas con una función de activación ReLU.

```
tf.keras.layers.Dropout(0.5)
```

Capa de Dropout que desactiva aleatoriamente el 50 % de las neuronas para evitar el sobreajuste.

```
tf.keras.layers.Dense(512, activation='relu')
```

Capa completamente conectada de 512 neuronas con una función de activación ReLU.

```
tf.keras.layers.Dense(164, activation="softmax")
```

Capa de salida del modelo. Tiene 164 neuronas, que corresponden al número de estilos musicales presentes en el dataset. La función de activación **Softmax** produce una distribución de la probabilidad entre las diferentes clases.

Este modelo es una CNN 1D con una estructura formada por una serie de bloques de capas convolucionales seguidas de normalización y pooling, seguidos por una serie de capas densas.

La primera parte del modelo (las capas convolucionales) se encarga de aprender características locales en pequeñas ventanas de los datos de entrada, mientras que la segunda parte del modelo, formado por capas completamente conectadas, aprende a combinar estas características para hacer predicciones.

TODO: INTRODUCIR GRÁFICOS CON LOS RESULTADOS DEL ENTRENAMIENTO

Trabajos relacionados

Este apartado sería parecido a un estado del arte de una tesis o tesina. En un trabajo final grado no parece obligada su presencia, aunque se puede dejar a juicio del tutor el incluir un pequeño resumen comentado de los trabajos y proyectos ya realizados en el campo del proyecto en curso.

Conclusiones y Líneas de trabajo futuras

7.1. Conclusiones

A través de este proyecto, se ha demostrado la eficacia pero también la complejidad del uso de la Inteligencia Artificial (IA) para la clasificación automática de estilos musicales.

En líneas generales, el sistema ha demostrado tener una eficacia razonable en la tarea de clasificar estilos musicales. No obstante, en el proceso también se han identificado limitaciones.

En primer lugar, aunque el proceso de extracción de características de audio juega un papel crucial en el desarrollo y entrenamiento de los modelos de clasificación de estilos musicales, podría ser conveniente considerar la exploración de otros métodos o la implementación de técnicas más sofisticadas con el objetivo de enriquecer la calidad de los datos que alimentan a la red neuronal.

Podría ser beneficioso explorar técnicas de *deep learning* como el aprendizaje por transferencia (*transfer learning*), los modelos *pre-entrenados* podrían usarse como un punto de partida, lo que podría mejorar la eficiencia del aprendizaje y potencialmente mejorar la precisión del modelo.

La cantidad del conjunto de datos tiene un impacto directo en la eficacia del sistema. Utilizar un conjunto de datos más extenso y variado podría mejorar la capacidad del sistema para generalizar.

En conclusión, este proyecto ha servido para demostrar que la clasificación de estilos musicales utilizando IA es factible y que la inteligencia

artificial aplicada a la música ofrece un gran potencial y motivación para investigaciones futuras.

7.2. Líneas de trabajo futuras

1. **Experimentación con diferentes modelos**: Se han utilizado principalmente redes neuronales convolucionales en este proyecto, pero hay muchas otras arquitecturas de red. Por ejemplo, las redes neuronales recurrentes (RNN) o modelos generativos adversarios (GANs) podrían ser buenas opciones para experimentar.

2. **Explorar enfoques de aprendizaje semi-supervisado y no supervisado**: Dada la dificultad de etiquetar grandes conjuntos de datos musicales, también se podría considerar la utilización de técnicas de aprendizaje semi-supervisado o no supervisado. Estos enfoques podrían ser útiles para aprovechar datos no etiquetados y mejorar el rendimiento del sistema.

3. **Incorporación de feedback del usuario**: Permitir a los usuarios corregir las clasificaciones incorrectas del modelo y utilizar esa información para mejorar el modelo.

4. **Construcción de un sistema de recomendación**: Teniendo el modelo de clasificación de estilos musicales, podría ser interesante construir un sistema de recomendación musical que sugiera canciones a los usuarios basándose en sus preferencias y hábitos de escucha.

TODO: ADD MORE FUTURE ENHANCEMENTS AND IMPROVE THIS SECTION!