Grado en Ingeniería Informática

2018- 2019

Trabajo de Fin de Grado

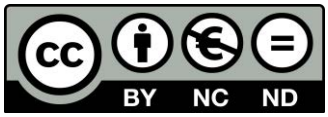
"Procesamiento de Audio con Técnicas de Inteligencia Artificial"

Jesús Iriz González

Tutor:

Miguel Ángel Patricio Guisado

8 de octubre de 2019 a las 10:00 en Sala de Juntas, Colmenarejo



Esta obra se encuentra sujeta a la licencia Creative Commons **Reconocimiento – No Comercial – Sin Obra Derivada**

Contenido

1. Introducción

1.1. Introducción a la problemática

A inicios del curso 2018-2019, se nos presentó a mí y a 9 compañeros más la posibilidad de participar en una serie de cátedras ofrecidas por las empresas BQ y MasMovil. El objetivo de estas cátedras era fomentar investigaciones relacionadas con la Inteligencia Artificial y el Aprendizaje Automático, formando a 10 alumnos de la Universidad Carlos III en estos campos. En mi caso, yo fui uno de los afortunados alumnos seleccionados, y de entre el rango de posibles estudios que ofertaban, decidí adentrarme en el mundo del audio con el interés de explotar las capacidades de la Inteligencia Artificial en este proceso.

El procesamiento de audio es un subcampo del procesamiento de señales digitales, el cual se especializa en el estudio de las propiedades matemáticas de las señales digitales de audio para extraer sus propiedades, analizarlas, y finalmente aplicar modificaciones a la fuente de audio original.

Este campo es uno de los más amplios dentro del estudio de las señales digitales, ya que involucra una gran cantidad de fuentes de audio a estudiar, tales como la música, sonidos ambientales, o incluso el habla humana. Últimamente el campo del procesamiento de audio ha experimentado un gran desarrollo dado al reciente crecimiento en el uso de asistentes personales como Siri (Apple), Alexa (Amazon) o Cortana (Microsoft). La mayoría de los nuevos dispositivos móviles que entran al mercado hoy en día ya incluyen un asistente personal capaz de reconocer tu voz y ejecutar una serie de instrucciones dependiendo de lo que pidas, lo que ha aumentado la cantidad de recursos y dinero que grandes empresas como las mencionadas anteriormente invierten en el procesado de habla.

A pesar de ello, otras aplicaciones del procesamiento de audio como el procesado de música han caído más en el olvido, viéndose menos estudiadas tanto en tecnologías hardware como software.

Durante los últimos años, el campo de la inteligencia artificial ha experimentado un gran crecimiento y difusión en la vida cotidiana de la mayoría de las personas. Ya no es raro

de vez en cuando oír hablar de redes de neuronas, inteligencia artificial o machine learning, ya sea por los avances que pueden traer a la sociedad, o por los potenciales peligros que pudieran suponer.

Algunos de los motores de búsqueda de internet están empezando a intentar aplicar técnicas de inteligencia artificial a sus búsquedas para ofrecer unos mejores resultados a sus usuarios. Por otro lado, cada día el mundo de la conducción autónoma de vehículos crece más, arrastrando a cada vez más compañías de automóviles a investigar en este campo. Incluso redes sociales como Instagram o Snapchat dan uso a redes neuronales para reconocer rostros humanos y aplicar modificaciones en ellos en forma de filtros.

Dentro del campo de la inteligencia artificial, el estudio y construcción de redes neuronales se encuentra a la vanguardia de la investigación de nuevas y más potentes técnicas de inteligencia artificial, capaces de implementarse en gran variedad de sistemas de campos tan dispares como el reconocimiento facial o el procesamiento de habla.

Parece innegable que nos acercamos a un mundo en el que la inteligencia artificial se va a encontrar en todo ámbito y lugar de nuestras vidas, incluyendo nuestros bolsillos, casas o incluso nuestras ciudades enteras.

Por ahora, la mayor parte de las técnicas aplicadas al procesamiento de audio (Concretamente, al área de edición de audio) dan soluciones no deterministas y subjetivas, es decir, que obtenemos un resultado distinto dependiendo de la persona encargada de la tarea de alterar el sonido. Esto puede llegar a ser un problema, porque cuanto más subjetiva sea una solución, es más probable que para otra persona esta no sea la mejor solución (o ni siquiera una buena solución)

El problema que he decidido tratar de solventar, es intentar mezclar los dos conceptos de procesamiento de audio e inteligencia artificial (Redes neuronales en este caso) y buscar lo que podría ser una aplicación interesante de ambas. La mejora de la calidad de sonido de piezas de música, también conocida como ecualización musical, es un factor muy importante a la hora de disfrutar de una canción, pero no es un proceso generalmente automatizado, ya que es el propio usuario el que tiene que cambiar manualmente los parámetros de ecualización con este fin. Hoy en día, cuando tomas una foto con la cámara de tu móvil, esperas que este automáticamente se encargue de modificar parámetros como contraste, color y brillo dependiendo de la imagen, de manera que esta tenga una mayor calidad, pero la mejora de sonido sigue siendo un proceso manual.

Dado esto, voy a estudiar el campo de la ecualización musical y a investigar la factibilidad de una solución basada en redes neuronales, intentando solucionar cualquier problema que pueda surgir por el camino.

1.2. Objetivos

Objetivo principal

El objetivo principal de este trabajo es investigar la viabilidad de aplicar la estructura de una red neuronal para procesar una pista musical, con el objetivo de mejorar su calidad. La investigación consistirá en estudiar distintos métodos que puedan acabar llevando a la solución, comparándolos en calidad de resultados y determinando cuál es la mejor solución a este problema concreto. El resultado más interesante sería llegar a una solución capaz de funcionar en dispositivos móviles a tiempo real, es decir, que sea lo suficientemente eficiente tanto en consumo de recursos de hardware, como en consumo de tiempo.

Con el fin de mejorar la eficiencia, voy a estudiar romper el esquema que se utiliza en la mejora de imágenes en el que la entrada de una imagen produce directamente como salida una imagen mejorada. En mi caso, para romper con este proceso, propongo primeramente clasificar una canción por géneros, dividirla en intervalos de tiempo, y clasificar cada intervalo con ayuda de una red neuronal. De esta manera, se genera una biblioteca con información de a qué género se aproxima más la canción en cada intervalo, consiguiendo que finalmente no sea necesario realizar todo el proceso en un único caso en un dispositivo móvil, si no que se puedan precalcular los parámetros y posteriormente aplicar una ecualización. El proceso de fragmentación y ecualización se comentará más a fondo en los puntos 2.3, 2.4, y sobre todo en el punto 3.

Objetivos secundarios

Así mismo, existe una serie de objetivos secundarios que se busca que sean cumplidos o al menos analizados durante el estudio de la problemática:

- Estudiar cómo funciona y cómo se representa digitalmente una onda de sonido, comprendiendo conceptos como amplitud de onda, frecuencia o timbre.
- Estudiar qué características tiene una onda de sonido, y cómo pueden ser extraídas de esta con métodos matemáticos tales como descomposiciones en frecuencias.
- Estudiar distintas arquitecturas de redes neuronales, comprendiendo su funcionamiento, y comparando sus capacidades y fortalezas para los distintos problemas de aprendizaje automático en los que se puedan aplicar.

- Comparar esta área con campos similares (Como la edición y mejora de fotografías) tanto en los procesos que llevan a la mejora de la fuente original, ya sea imagen o sonido, como en los resultados obtenidos.
- Investigar si es posible eliminar el factor subjetivo de la edición musical, y en caso de que lo sea, cómo alcanzar una solución generalmente mejor para la mayoría de los usuarios.
- Estudiar varias alternativas de procesamiento de los datos de entrada y de salida del sistema, y determinar cuál es la más eficaz y eficiente para este caso.
- Estudiar las posibles mejoras que pueden añadirse a este estudio a futuro, añadiéndole más complejidad y alcanzando unos mejores resultados.
- Comparar distintas bases de datos de música y elegir aquella que mejor se adecue al problema, estudiando los motivos de elección para cada distinto caso.

1.3. Marco regulador

Para la realización de este trabajo, se deberán tener en cuenta dos reglamentos de gran importancia:

Por un lado, se tendrá que contemplar las posibilidades de utilizar un conjunto de datos (O dataset) propio, específico, y generado personalmente, o un dataset ya construido y de propósito más general. Para este último caso, es fundamental asegurarse de que las pistas de música utilizadas en el dataset son recopiladas contando con la autorización de su autor original, además de que el propio dataset también tiene que estar autorizado para al menos su uso académico y/o de investigación.

De igual manera, ya sea para su recopilación en un nuevo dataset o para su utilización en el propio proyecto o bien para analizar sus características o bien para analizar los resultados, las canciones que se utilicen también tienen que permitir esta clase de uso sobre ellas.

Como se establece en el capítulo II, artículo 34 de la ley de propiedad intelectual [1], quedan exentos de solicitar autorización del autor de una obra para el uso de esta, “Cuando la utilización se realice con fines de ilustración de la enseñanza o de investigación científica siempre que se lleve a efecto en la medida justificada por el objetivo no comercial que se persiga e indicando en cualquier caso su fuente”. Dado que los fines de este trabajo son puramente de investigación, y que no se persigue un objetivo comercial, puedo libremente dar uso de estas bases de datos.

1.4. Impacto socioeconómico

El campo del análisis de audio, como ya se ha comentado anteriormente, es un campo muy amplio que alcanza muchos ámbitos muy distintos, pero a la hora de analizar el impacto socioeconómico de esta investigación voy a centrarme únicamente en el sector de la música y la edición musical, dejando de lado campos en los que se investigará mucho menos como el reconocimiento de habla. De esta manera, los sectores más involucrados son los siguientes:

Público general

Este proyecto no deja de ser una investigación enfocada en una posible mejora para las personas a la hora de escuchar música, por lo que estos, los usuarios, son el principal grupo de interés. Por lo general, la gente no dedica tiempo a entrar en la configuración de sus reproductores de música, ya sean móviles, ordenadores, u otro tipo de dispositivos, por lo que, si finalmente se alcanzara una solución factible, la potencial mejora en la calidad del sonido que escuchan las personas puede ser considerable.

Compañías de fabricación de dispositivos móviles

Al comienzo de esta investigación, este era un campo muy poco desarrollado. Lo máximo que un dispositivo móvil llegaba a implementar en términos de ecualización eran una serie de perfiles seleccionables según el tipo de música que fueras a escuchar, o incluso un conjunto de barras para modificar los parámetros de ecualización manualmente, pero esto no dejaba de ser una entrada manual y no automática, ya que permanecían constantes hasta que el usuario volvía a decidir modificarlos. A día de hoy, algunos de los dispositivos móviles más sofisticados ya implementan prototipos de ecualizadores inteligentes que no requieren una interacción del usuario para funcionar y adaptarse, lo que deja patente el interés que tienen estas empresas en el campo.

Plataformas de retransmisión de música

Otro sector que se puede ver beneficiado es el de las plataformas de retransmisión de música, tales como Spotify o YouTube. Pequeñas mejoras y diferencias respecto a otras plataformas pueden hacer que al final un usuario se decante por usar una u otra. Es por ello, que este estudio también puede resultar de interés para ellas.

Producción musical

El campo de la producción musical es el encargado de utilizar una serie de pistas de sonido individuales en crudo (Como por ejemplo la voz o la guitarra de una canción por separado) y unirlas en una canción aplicando las mejoras que puedan considerar necesarias a cada pista. Como ya he comentado, esta es una tarea subjetiva en la que influyen las opiniones de los productores musicales. Comparando este caso con el mundo de la fotografía, la producción musical sería el proceso del fotógrafo desde que toma la foto hasta que la acaba de retocar digitalmente en un ordenador y la entrega al cliente. Aun así, la edición mediante software en un dispositivo móvil y la efectuada por un fotógrafo profesional coexisten y se llegan a complementar, no competir entre ellas. Por lo tanto, aunque a priori pudiera parecer que la ecualización inteligente de música pudiera sustituir la labor de un profesional de la música, considero que no solo no son rivales, si no que pueden llegar a ser complementarias e incluso beneficiosas.

1.5. Terminología y definiciones

En este punto se incluirá una colección de términos específicos del problema con el objetivo de facilitar la lectura de los puntos posteriores.

- **Inteligencia artificial:** Es, en contraste a la inteligencia natural, una combinación de cálculos y algoritmos que simulan una inteligencia como podría ser la de un ser humano, pero dentro de un entorno informático como un ordenador.
- **Machine learning:** Rama de la inteligencia artificial que estudia técnicas que permitan que un sistema de inteligencia artificial aprenda, es decir, que su desempeño mejore continuamente con el uso, experiencia, o tiempo.
- **Red neuronal:** Técnica de inteligencia artificial derivada del campo del machine learning, que busca imitar las conexiones entre neuronas del cerebro de un ser

vivo, capaz de realizar algunas tareas que pueden considerarse inteligentes, apoyándose en un aprendizaje previo.

- **CPU:** Central processing unit, unidad de procesamiento central, es la unidad hardware que dentro de un ordenador realiza todos los cálculos principales de este.
- **GPU**: Graphics processing unit, unidad de procesamiento gráfico, es una unidad hardware especializada en cálculos gráficos (Cálculos vectoriales). Al realizar operaciones vectoriales de manera más eficiente que una CPU, hace a estos dispositivos hardware mucho más eficientes para trabajos con redes neuronales que los procesadores principales.
- **T-SNE**: T-distributed Stochastic Neighbor Embedding, es un algoritmo de machine learning para visualizer espacios vectoriales de muchas dimensiones en una representación aproximada de dos o tres dimensiones.
- **Procesamiento de habla:** Análisis de las señales digitales de audio producidas por el habla humana con el objetivo de, o bien reproducirla (Transformación de texto a voz) o bien reconocerla (Reconocimiento de habla o de voz)
- **TTS**: Por sus siglas en inglés, Text To Speech, literalmente "Texto a habla", consiste en la generación de sonido simulando la voz de una persona a partir de un texto escrito.
- **Onda:** Propagación de una alteración en el espacio, que suele ser representada en una gráfica como una relación de su amplitud respecto al tiempo.
- **Amplitud de onda:** Valor en un instante de tiempo que toma una onda, medido desde un punto central o de reposo que es considerado el punto de equilibrio o de amplitud 0.
- **Frecuencia de onda:** Cantidad de oscilaciones que realiza una onda en un intervalo de tiempo, generalmente representada en Hercios (Hz u oscilaciones por segundo)
- **Nota musical:** Onda emitida por un instrumento musical o una voz humana con una frecuencia concreta.
- **BPM:** Por sus siglas en inglés, Beats per minute, en español pulsos por minuto. Es una medida del ritmo de una canción, a más pulsos por minuto, más rápido suena la canción.
- **Banda de frecuencias:** Rango continuo de frecuencias que va desde una frecuencia mínima y una máxima, comprendiendo todas las frecuencias entre ellas.
- **Decibelios:** Coloquialmente el volumen de un sonido. Se encuentra relacionado con la amplitud, ya que a mayor amplitud por lo general se percibe un mayor volumen, pero el volumen necesita un intervalo de tiempo para cobrar sentido, al contrario que la amplitud que se define como el valor instantáneo de una onda. Por esto mismo, que la amplitud de una onda en un instante sea 0, no implica que el volumen del sonido sea también 0.

- **Timbre:** Percepción de un sonido que permite diferenciar la fuente de un sonido de otro de la misma frecuencia. Por ejemplo, la nota Do en un piano y en una guitarra tiene misma frecuencia, pero distinto timbre, por ello somos capaces de distinguir si la nota es tocada en un piano o en una guitarra
- **Ecualización:** Proceso en el cual se modifica el volumen de una banda de frecuencias, atenuando unos rangos y reforzando otros, buscando como objetivo mejorar la calidad del sonido.
- **Dataset:** Colección de datos preparado para ser utilizado por un agente (Humano o máquina) para realizar un análisis sobre este o extraer información.
- **Features/Características:** Datos numéricos obtenidos de una fuente no directamente matemática. La extracción de características consiste en transformar una entrada en crudo como puede ser una imagen o una pista de sonido en una serie de parámetros numéricos que puedan ser manejados por un sistema de computación como una red neuronal.

1.6. Estructura de la memoria

En este punto se describirán brevemente los puntos por los que está compuesta la memoria, con el fin de tener un punto de referencia para encontrar la información que se pueda necesitar, así como facilitar la lectura del documento.

El punto **1. Introducción**, incluye una pequeña introducción a la problemática escogida para este documento, así como los objetivos que se buscan durante su desarrollo, el marco regulador con la legislación a tener en cuenta, el impacto socioeconómico, y una colección de términos técnicos con sus explicaciones.

En el punto **2. Estado del arte**, se detalla la investigación llevada a cabo en relación al problema de la ecualización inteligente de pistas de música, recopilando los avances actuales en la cuestión, los conocimientos previos y adquiridos durante el desarrollo, y algunas de las decisiones tomadas a la hora de por ejemplo escoger una arquitectura de red, o un dataset con los datos necesarios para el proyecto.

En el punto **3. Desarrollo y funcionamiento final** se discuten y asientan todas las tomas de decisión que involucran al entorno de desarrollo escogido, además los distintos módulos que componen el proyecto y su funcionamiento.

En el punto **4. Experimentos** se incluyen los experimentos realizados, acompañados de gráficas explicativas de los resultados, con el fin de determinar qué solución es la mejor para el problema.

En el punto **5. Planificación y metodología** se recoge la planificación original que se estimaba para el proyecto, así como la planificación final que ha resultado tras su finalización. Por último, incluye una estimación de presupuesto para el proyecto.

En el punto **6. Conclusiones y líneas futuras** se concluyen finalmente los objetivos del proyecto, decidiendo sobre una solución final, y comentando cuales pueden ser las futuras investigaciones y mejoras a realizar sobre la base investigada en este proyecto.

2. Estado del arte

En este punto se va a mostrar el estado actual de desarrollo de las tecnologías y los campos involucrados en el proyecto. Comenzaré comentando conceptos más matemáticos y relacionados con la música para comprender la base del proyecto, para seguir hablando de los géneros musicales y de qué manera se puede clasificar una canción por géneros. A continuación, hablaré del proceso de ecualización, de su contexto actual a la hora de elaborar el documento, y de cómo aplicarlo a la división por géneros escogida. Por último, hablaré sobre los últimos avances en procesamiento de sonido en el campo de machine learning, y analizaré distintas arquitecturas de redes neuronales, así como qué parámetros pueden funcionar como entrada para la red.

2.1. Ondas y música

Para entender cómo puede operar una red neuronal que trabaje con música, es fundamental primero entender los conceptos tras una pista de audio y qué es lo que hace que una canción suene de una manera o de otra.

El objetivo de la red neuronal que voy a emplear es primeramente conseguir determinar a qué genero corresponde una canción. Una persona es capaz de identificar si una escena corresponde a una ciudad, a un bosque, o a una habitación cerrada, pero ¿Cómo lo consigue? Identificando objetos individuales como edificios, árboles o muebles y componiéndolos, comprendemos que la estructura de una escena se corresponde con una clase en concreto.

Con la música ocurre lo mismo. Si conseguimos separar una canción en los elementos individuales que la componen, conseguiremos identificar exactamente a qué genero corresponde. Por ejemplo, si diferenciamos guitarras eléctricas, una batería, un bajo eléctrico, y una voz, podremos concluir con relativa exactitud que aquello que estamos escuchando es probablemente una canción de rock, sin embargo, si la canción está compuesta por varios violines, violas, clarinetes, etc, probablemente sea una pieza de música clásica. Siguiendo con la analogía de las imágenes, estos instrumentos corresponderían con elementos como árboles o coches en la escena de una fotografía.

Podemos seguir ahondando de esta manera en las características de la música. Por ejemplo, las guitarras eléctricas y las baterías son comunes en géneros distintos como el

rock y el metal, ¿Cómo se pueden diferenciar ambos? Los distintos géneros musicales no se limitan a diferenciarse únicamente en base a sus instrumentos, si no que existen distintas técnicas y sonoridades empleadas en cada uno. Por ejemplo, la música rock tiende a utilizar acordes mayores, mientras que el metal tiende a utilizar acordes menores. Los acordes son agrupaciones de tres o más notas que tienen una sonoridad concreta según la cantidad de notas y la distancia que existe entre ellas. Por lo general, los acordes mayores se caracterizan por una sonoridad alegre y viva, mientras que los acordes menores suenan más tristes y nostálgicos.

Otra manera de diferenciar dos géneros es mediante las técnicas empleadas en la canción que no necesariamente tengan que ver con las notas musicales empleadas. Por ejemplo, el vibrato de la voz. El vibrato consiste en hacer que el tono de la voz oscile entorno a la nota principal que se está representando. Esta es una técnica muy común en por ejemplo óperas, donde en el momento en que el o la cantante va a sostener la voz en una nota durante un largo periodo de tiempo, la mantiene con un vibrato. Sin embargo, en otros tipos de música como en el pop no son tan comunes.

Pero ¿Cómo podemos reconocer todos estos tonos si tan solo tenemos acceso a los datos de una única onda de sonido, sin tener información por separado de cada intérprete? Una onda de sonido no es más que una composición de ondas fundamentales, es decir, de ondas de una frecuencia constante, pero de amplitud variable en el tiempo. De esta manera, una onda compleja como la que puede emitir un instrumento siempre puede ser descompuesta en múltiples ondas simples, como se muestra en la siguiente figura:

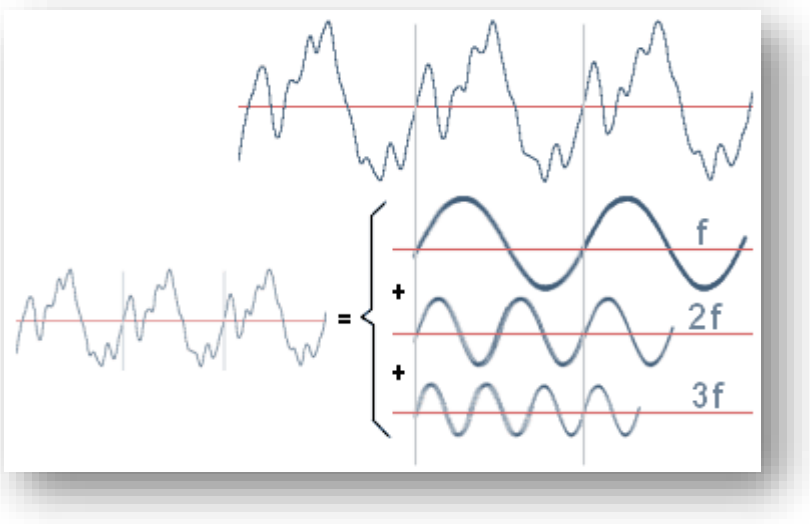


Fig 1. Descomposición de una onda compleja en ondas más simples [2]

Por lo tanto, si un instrumento puede descomponerse en frecuencias simples, una pieza de música con múltiples instrumentos también puede descomponerse en un conjunto de frecuencias simples que esta vez sí ya pueden ser comprendidas por un ordenador.

2.2. Géneros musicales

Como ya se ha concluido en el anterior punto, los géneros musicales son agrupaciones de sonoridades y técnicas que dan como producto un conjunto reconocible y distinguible por sus características de otros géneros.

La división de la música por géneros musicales puede ser tan exhaustiva como se desee, pueden diferenciarse por ejemplo dos géneros, siendo música instrumental aquella que incluye instrumentos, y música vocal aquella que además de instrumentos incluye voz. Incluso si queremos, podemos llegar a diferenciar géneros según las peculiaridades que tengan según su procedencia, separando entre pop occidental y pop oriental si queremos.

Los géneros musicales también pueden ser entendidos como un diagrama de árbol, en el que existe una jerarquía según si unos géneros son derivados de otros. Por ejemplo, el jazz es un género derivado de la mezcla del rock & roll y de la música afroamericana.

La decisión por tanto recae en qué y cuántos géneros se quieren diferenciar.

2.3. Clasificación por géneros

Llegando a la cuestión de qué clasificación de géneros escoger comienza a surgir la duda de si será mejor utilizar una división muy amplia o una más libre que agrupe más canciones en cada género. Si se utilizan muchos géneros, se alcanzará una gran precisión a la hora de definir las características de una canción, sin embargo, si se utilizan pocos se puede eliminar cierta confusión que pueda surgir al no saber diferenciar dos géneros que distan en detalles muy pequeños. Para llegar a una solución, voy a estudiar los géneros presentes en los datasets más habituales para el uso en sistemas de machine learning. La siguiente figura muestra los datasets de música más utilizados en los trabajos publicados de machine learning:

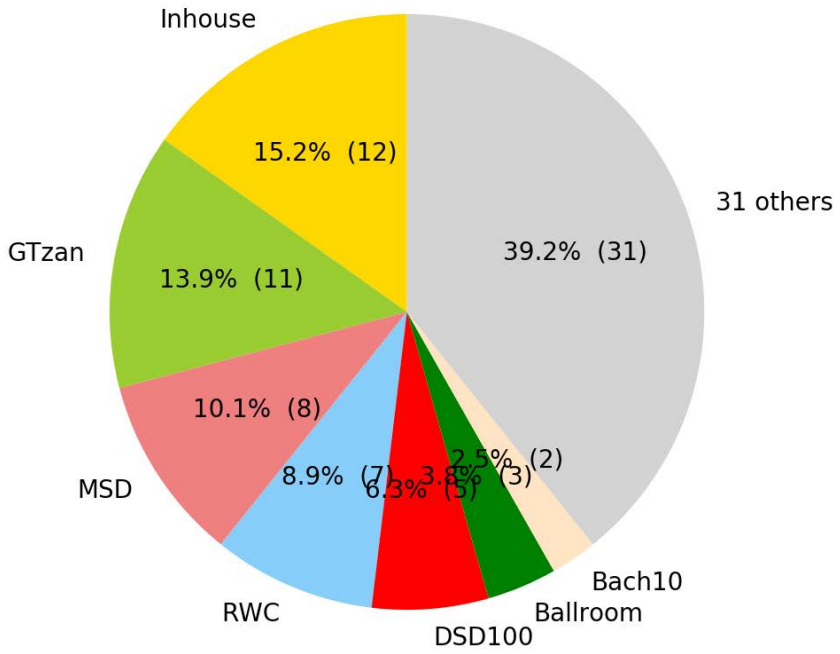


Fig 2. Proporción de datasets usados en estudios de machine learning con música [3]

Como se puede ver, existe un gran número de datasets distintos y ninguno parece ser realmente dominante, así que yo voy a centrarme en los 5 primeros que ya suponen una mayoría sobre el total.

Inhouse

Inhouse corresponde a todos aquellos datasets generados por el propio usuario de este y para el problema personal al que lo vaya a dedicar. Por tanto, aunque sea la técnica más empleada para esta cuestión, no es de gran interés para comprender la dinámica general de clasificación por géneros, ya que cada dataset empleará una división distinta.

GTzan

No considerando los datasets Inhouse, GTzan [4] es el dataset más utilizado para este tipo de trabajos. Originalmente, es el dataset que crearon George Tzanetakis y Perry Cook en su trabajo "Music genre classification of audio signals", uno de los trabajos pioneros de clasificación de música en géneros, por lo que resulta de especial interés. El dataset consiste en 1.000 fragmentos de canciones de 30 segundos cada uno, divididos en 10 géneros: Blues, Clásica, Country, Disco, Hip-hop, Jazz, Metal, Pop, Reggae y Rock. Además, los fragmentos fueron recogidos de distintas fuentes, como la radio o CDs de música, haciendo que además exista una variación en las condiciones de grabación de cada una.

MSD (Million Song Dataset)

Como indica su nombre, MSD [5] es un dataset muy amplio que cuenta con un total de un millón de canciones. Este dataset tan solo contiene las etiquetas de cada canción con una serie de características ya extraídas. El dataset en si no contiene ni las canciones completas, ni las etiquetas de géneros de cada una, pero ambos pueden ser descargados mediante trabajos contribuidos por la comunidad. Curiosamente, los géneros empleados por este dataset son muy similares a los de GTzan, tan solo fusionando Pop y Rock en uno, sustituyendo música Disco por música Electrónica, y añadiendo géneros como música Latina, música Vocal, y New Age. El volumen completo de datos suma un total de 280GB, pero existe una versión comprimida con 10.000 canciones que pesa 1.8GB y es más manejable.

RWC (Real World Computing)

El dataset RWC [6] contiene varios conjuntos de datos, entre los que se encuentra un dataset de sonidos de instrumentos, y una base de datos específicamente de música clásica. Para este trabajo, el subconjunto que es de interés es la base de datos de géneros de música, que coincide en el uso de Pop, Rock, Disco, Jazz y música Clásica con los demás datasets, pero añade música Latina, Marchas, música del Mundo, música Vocal y música tradicional japonesa, siendo estos 11 géneros a su vez divididos en 3 subgéneros. Por su lado, este es un dataset pequeño, contando con tan solo 3 piezas por subcategoría más una correspondiente a una canción A Capella, sumando un total de solo 100 canciones.

DSD100

Por último, DSD100 es un dataset con un total de 100 canciones completas y etiquetadas por géneros. Observando las etiquetas de géneros, se puede ver que cuenta con un total de 66 géneros distintos, llegando a diferenciar dentro del Rock (Que incluso fue agrupado con el Pop por MSD) Rock emo, acústico, de cine, vasco, country rock, e incluso rock indie ucraniano.

Los datasets Ballroom [8] y Bach10 [9] consisten en datasets de canciones de "salones de baile" como el Tango o el Vals, y de piezas de composiciones de Bach respectivamente. Por este motivo, ninguno de los dos es relevante para este trabajo.

Por lo que se puede observar tras analizar todos estos datasets, parece ser que la tendencia general es utilizar un conjunto pequeño de unos 10 géneros, y que existen algunos que son muy habituales encontrarse en todos los datasets, como la música Clásica, el Jazz o el Rock. Por este motivo, tomo la decisión de usar el conjunto de géneros de GTzan como base para el resto de la investigación.

2.4. Ecualización musical

La ecualización consiste en el proceso de alterar la amplitud de cada una de las frecuencias de un audio con el objetivo de hacer más notables unos rangos de frecuencias, y difuminar otros.

Para ello, lo que se hace es separar la onda utilizando la transformada de Fourier, que permite transformar la función de la onda desde el dominio del tiempo al dominio de las frecuencias, es decir, que descompone la onda en sus frecuencias que originalmente se encontraban acopladas. Para cada una de estas frecuencias, se aplica una modificación sobre su amplitud, traduciéndose en una variación del volumen de cada una.

La norma ISO establece que al menos, las bandas de frecuencias tienen que ser al menos de 31, 63, 125, 250, 500, 1000, 2000, 4000, 8000 y 16000 Hercios [10]. A cada una de estas bandas se le aplica una variación de volumen de entre -12 decibelios, y +12 decibelios. Una buena ecualización, aunque sea ligeramente subjetiva, busca amplificar los rangos de frecuencias más habituales de la canción, de manera que los detalles más importantes resalten más sobre los menos importantes.

Para mi caso de estudio, necesito encontrar un conjunto de 10 parámetros preestablecidos de ecualización para cada uno de los 10 géneros que he escogido en el punto anterior. La manera de obtenerlos será buscar algún sistema que incluya algún ecualizador parametrizable, y estudiar los perfiles que ofrezca para cada uno de estos géneros. Una plataforma que ofrece perfiles de ecualización ya creados es iTunes [11], desarrollada por Apple.

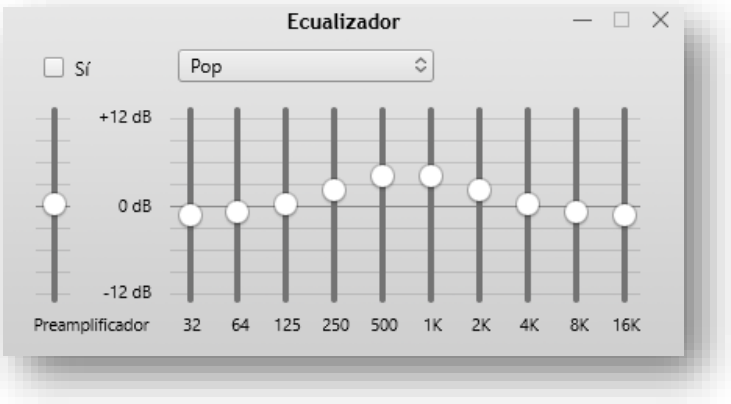


Fig 3. Perfil de ecualización para Pop de iTunes

En la anterior imagen, se puede ver que se busca potenciar las frecuencias cercanas a 500Hz y a 1000Hz. Esto se debe a que los sonidos predominantes en la música pop, como las voces, las guitarras, los teclados y los sintetizadores rondan esas frecuencias. Las voces habituales femeninas del pop rondan entre el La 3 y el Do 6 [12], de frecuencias 220Hz y 1046Hz respectivamente, por lo que es lógico que se amplifiquen estos rangos. Sin embargo, los sonidos muy agudos y muy graves no suelen utilizarse en este género musical, ya que no coinciden con el rango de frecuencias de ningún instrumento habitual del pop. Otro ejemplo es el perfil de ecualización que iTunes ofrece para la música rock:

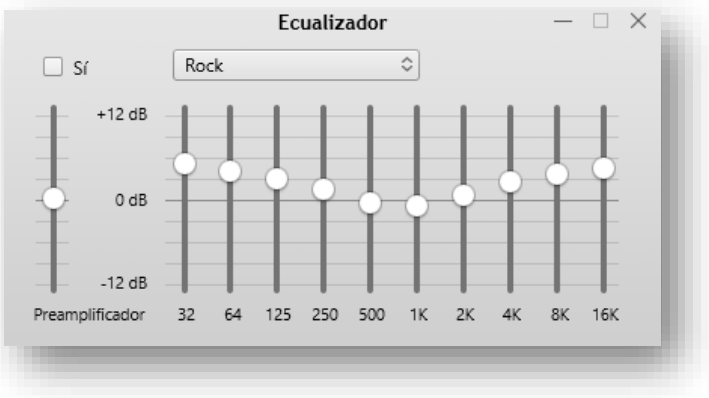


Fig 4. Perfil de ecualización para Rock de iTunes

En este caso, se puede observar que el perfil es completamente inverso al perfil del pop, ya que las frecuencias que se intentan potenciar son las más agudas y las más graves. En el rock, se da uso a instrumentos como la batería y el bajo eléctrico, que emiten frecuencias muy bajas, de entre 20Hz y 500Hz, pero generalmente rondando entre los 20Hz y los 261Hz del Do 4. Por otro lado, las voces y las guitarras eléctricas tienden a sonidos muy agudos.

Revisando la biblioteca de ecualización que ofrece iTunes, encuentro que hay disponibles ajustes para 6 de los 10 géneros de GTzan, faltando un perfil para Country, Reggae, y Disco, así que los tres géneros tienen que ser analizados en instrumentación y en base a ello diseñar unos nuevos perfiles, usando como referencia una tabla de conversión de notas a frecuencias y una tabla de rangos de frecuencias para cada instrumento de la referencia [13]

Country

El country es un género que, aun siendo un derivado cultural del Rock & Roll, cuenta con una instrumentación muy parecida al Pop, así que tomaré este último como punto de partida. Algunos de los instrumentos más notables de este género que no suelen estar presentes en el Pop son los siguientes:

- **Violín**: Su rango de frecuencias es desde el Sol 3 hasta el Sol 7, que corresponde con las frecuencias de 196Hz a 3136Hz.

- **Banjo**: Su afinación más habitual corresponde con las frecuencias de 110Hz a 800Hz.
- **Harmónica**: Frecuencias de entre 180Hz y 3100Hz.

De esta manera, en base a esto y al perfil de ecualización del Pop, potenciaré ligeramente las bandas de 125Hz, 1kHz, 2kHz y 4kHz, y algo más las bandas de 250Hz y 500Hz.

Reggae

El reggae es un género desarrollado originalmente en Jamaica, y derivado de géneros como el Rock o el Ska. Este género tiene en común con el Rock que mantiene los instrumentos graves, como los bajos eléctricos y la percusión, sin embargo, las voces y las guitarras que se emplean tienden a ser ligeramente más agudas que en el Rock. Para desplazar un poco las frecuencias de estos timbres hacia tonos más agudos, difuminaré las frecuencias más graves de estos y potenciaré las más agudas.

Como resultado, tomo el perfil de ecualización del Rock, mantendré las frecuencias de hasta 64Hz, bajaré las frecuencias de 125Hz y 250Hz, y subiré las de 250Hz y 500Hz.

Disco

El disco, como su nombre indica, es un género que se popularizó en las salas de fiesta (discotecas) a partir de la década de los setentas. Debido a las limitaciones de sonido de los aparatos de reproducción de la época, era complicado emitir sonidos muy agudos en dispositivos a alto volumen, así que las frecuencias muy altas no son muy predominantes en este género.

Por otro lado, siendo un derivado del Pop y el Blues, mantiene algunos de los sonidos graves del blues con la percusión y el bajo, y los sonidos centrales y no muy agudos del Pop añadiendo instrumentos como los sintetizadores.

Por esto, partiré del Blues e incrementaré las frecuencias de 250Hz y 500Hz, bajando las frecuencias de 4kHz, 8kHz, y 16kHz.

Finalmente, ya dispongo de perfiles para cada uno de los 10 géneros que he escogido, incluidos en los anexos para su consulta.

2.5. Machine learning

El campo del machine learning está constituido por gran variedad de sistemas capaces de aprender a realizar de una manera de la mejor manera posible, como, por ejemplo, aprender a clasificar géneros de música, detectar fallos en un servidor antes de que se produzcan, o simular una conversación.

De una manera muy general y poco específica, la mayoría de los sistemas de machine learning recaen en dos grandes categorías de algoritmos:

- **Aprendizaje supervisado**: Estos algoritmos tratan de buscar una relación entre una serie de datos de entrada y unas salidas etiquetadas para cada caso. Uno de los problemas más comunes para los que se utilizan este tipo de algoritmos, es el de clasificación, mediante el cual, a base de recibir ejemplos, el sistema aprende a diferenciar unas entradas de otras.
- **Aprendizaje no supervisado**: En este caso, el algoritmo no recibe etiquetas para cada caso, por lo que se encarga de reconocer patrones dentro de los datos de entrada y buscar de qué manera puede etiquetar nuevas entradas.

Para mi problema, he optado por usar un sistema de aprendizaje supervisado, ya que voy a realizar una traducción de la etiqueta que asigne el sistema a una ecualización adecuada. Otra manera de trabajar puede ser utilizar un sistema de aprendizaje no supervisado que busque patrones dentro de los géneros musicales y los clasifique según las características comunes que esta encuentre, pero ya disponer de las etiquetas aporta más capacidad de decisión sobre de qué manera quiero que funcione el sistema, y además facilita el trabajo de aprendizaje.

Una de las técnicas de aprendizaje automático, quizás la que está cobrando más relevancia estos últimos años, y la que voy a utilizar yo en este proyecto, son las redes de neuronas artificiales.

Este paradigma imita el funcionamiento de una red de neuronas natural, en la que unas neuronas se comunican con otras recibiendo estímulos, que son procesados y reenviados a otras neuronas, buscando algún fin. A cada conexión entre neuronas se la asigna un peso, de manera que, si una conexión se encuentra entre una neurona que emite un estímulo de un determinado valor, la neurona que recibe ese estímulo lo hará alterado por el peso de la conexión. Mediante el aprendizaje, todos los pesos de una red se van

ajustando hasta alcanzar valores cada vez más precisos, comparando la salida de la red con la salida esperada por esta.

2.5.1. Ventajas y desventajas

A continuación, paso a comentar las ventajas y desventajas que puede tener el uso de una red neuronal en vez de otro tipos de métodos, tanto en el caso general, como en el caso concreto de la ecualización musical.

Ventajas:

- **Aprendizaje automático**: El sistema solo necesita una serie de datos etiquetados que serán los que utilice para mejorar su funcionamiento durante una etapa de aprendizaje. Una vez la red está entrenada, los resultados que esta obtiene son mucho más acertados, es decir, las etiquetas predichas coinciden en mayor número de casos con las etiquetas preasignadas.
- **Abstracción**: El proceso interno es prácticamente ajeno al usuario, es la red la que aprende esquemas y patrones, librando al usuario de tener que identificarlos previamente.
- **Velocidad**: El entrenamiento es el proceso más largo de la etapa de vida de una red neuronal, pero una vez la red está completamente entrenada, la obtención de salidas es considerablemente rápida, ya que se tratan de cálculos matemáticos sencillos.
- **Precisión:** Los resultados obtenidos por la red son muy acertados, ya que se limita a aprender una tarea y a perfeccionarla durante el entrenamiento.
- **Resultados objetivos**: Al tratarse de un proceso matemático similar al de aproximación de una función mediante regresión, los resultados obtenidos son, por lo general, objetivamente mejores, siempre y cuando los parámetros de entrada escogidos sean los adecuados para obtener la salida esperada.
- **Similar al cerebro**: Al buscar imitar el funcionamiento de las neuronas de un cerebro, los resultados se obtienen de una manera similar pero más simplificada de cómo lo haría un ser humano**.**
- **Elimina la necesidad de ser profesional**: Utilizando redes neuronales que ya hayan aprendido, cualquier usuario puede disfrutar de una fotografía con los parámetros bien ajustados o de una canción correctamente ecualizada, sin necesidad de ser un profesional del campo o de acudir a uno.

Desventajas

- **Tiempo extenso de entrenamiento**: Sobre todo con volúmenes grandes de datos y de categorías para clasificar, el tiempo de entrenamiento de una red neuronal puede ser realmente extenso, llegando a horas e incluso días.
- **Complejidad de la tarea**: En algunos casos, si la tarea es compleja, es posible que se necesite una cantidad enorme de parámetros de entrada, por lo que se requiere un considerable tiempo de análisis para determinar cuáles son las mejores entradas para una determinada tarea.
- **Gran cantidad de datos**: Para conseguir un aprendizaje efectivo, es necesario utilizar un gran volumen de ejemplos iniciales, que mantengan variedad de casos y de etiquetas, para conseguir que la red clasifique de manera eficaz.
- **Oscuridad**: Debido a que el proceso es hermético, solo podemos analizar si los resultados obtenidos son buenos o malos, pero no qué patrones ha aprendido el sistema ni qué significan los valores intermedios.
- **Etiquetado manual**: Para poder conseguir una biblioteca o dataset de entradas ya etiquetadas, es necesario que estas ya hayan sido analizadas y clasificadas previamente por una persona, es decir, aunque las redes neuronales permitan automatizar el proceso de clasificación, siempre será necesaria una persona que clasifique manualmente de manera previa.

2.5.2. Cómo funciona una red neuronal

Una red neuronal consiste en un conjunto de neuronas separadas por capas y comunicadas entre ellas mediante conexiones. Cada neurona recibe un valor inicial, y esta lo transforma y reenvía a todas las neuronas con las que esté conectada en adelante. Estos valores iniciales pueden proceder o de la entrada inicial a la red, o de incluso otra neurona. Cada neurona cuenta con una función de activación y cada conexión con un peso. Las funciones de activación son funciones limitadoras, que o bien modifican el valor que emite la neurona o impone un mínimo para que este valor sea propagado. En cada conexión que comunica una neurona A con una neurona B, se multiplica el valor de salida de la neurona A por el valor del peso de la conexión, quedando como resultado que la entrada que recibe la neurona B es igual a la suma de todas las salidas multiplicadas por todos los pesos conectados con ella.

Así, la entrada se va procesando hasta que alcanza la última capa o capa de salida, de la que se extraen los resultados del cómputo. La fortaleza de las redes neuronales reside en que los pesos de las conexiones pueden ajustarse según si los resultados obtenidos en la

capa de salida son correctos o no, en un proceso conocido como entrenamiento. De esta manera, si el resultado obtenido es acertado, se refuerza el comportamiento de la red ante ese tipo de entradas, mientras que, si el resultado es erróneo, se ajustan los pesos para tratar de "acercar" los futuros resultados a una solución satisfactoria. A cada uno de los pasos en el entrenamiento de una red neuronal se le conoce como "época" (o Epoch en inglés)

2.5.3. Arquitecturas de redes

Para alcanzar a hablar sobre los tipos de estructuras de redes de neuronas que mejor pueden funcionar para este problema, primero es fundamental entender los modelos más básicos.

Redes monocapa

Esta es la arquitectura de red neuronal más simple. En ella, sólo existe una capa de neuronas que recibe las entradas y directamente las emite como una o varias salidas calculadas. Debido a su simpleza, este es un tipo de red neuronal muy adecuado para problemas simples de clasificación, con pocas entradas y pocas clases o etiquetas como salidas, ya que son también redes que obtienen resultados muy rápidos.

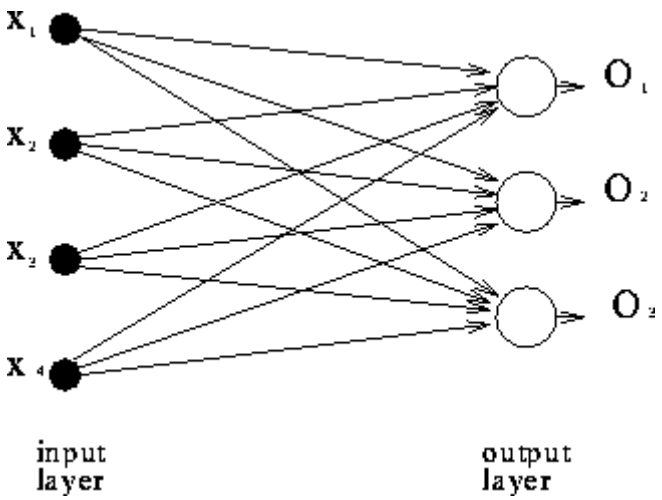


Fig 5. Red monocapa [14]

Incluso en su simpleza, contando con solo una capa se pueden alcanzar sistemas ligeramente más sofisticados, como una red de Hopfield, en la cual las neuronas tienen a su vez conexiones con neuronas de la misma capa. Este tipo de redes también se conoce como redes de neuronas recurrentes.

Redes multicapa

Añadiendo capas de neuronas, se alcanza una mayor capacidad de cómputo y unos resultados más precisos. La desventaja de estos tipos de redes es que, debido a su mayor tamaño, también acarrean más necesidad de tiempo y recursos para llegar a la solución y ser entrenadas.

La diferencia de estas redes con las redes monocapa es que estas incluyen una o más capas intermedias llamadas capas ocultas, que continúan el proceso de emitir resultados a otras neuronas.

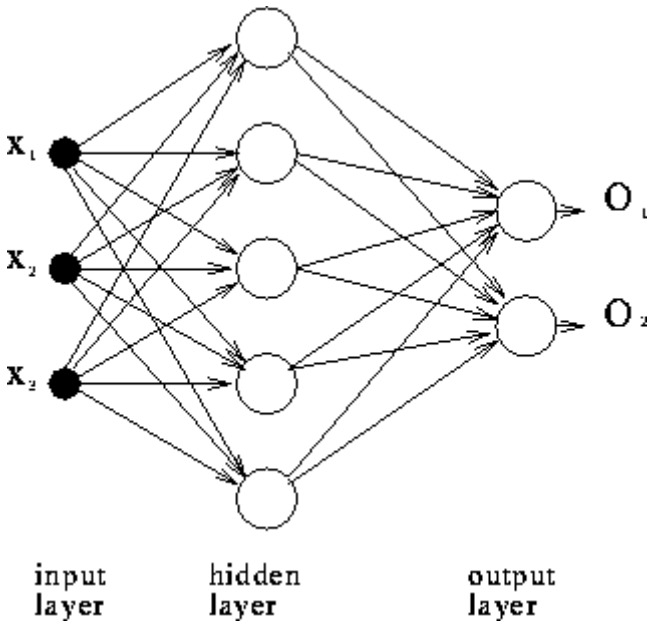


Fig 6. Red multicapa [15]

De igual manera, combinando redes distintas, y aplicando modificaciones sobre estos modelos, se generan nuevas arquitecturas más eficientes en ciertos casos [16]. A

continuación, se comparan los dos modelos de redes de neuronas avanzados que más uso han recibido en problemas relacionados con el tratamiento de música o sonido según [3].

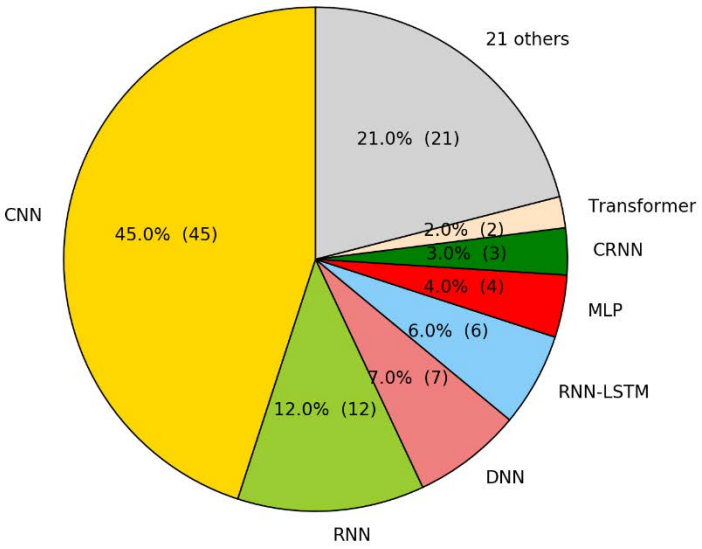


Fig 7. Proporción de tipos de redes en estudios de machine learning con música [3]

El estudio anteriormente citado, incluye una recopilación no exhaustiva de todos los artículos publicados sobre música y redes neuronales. Como se ve en la gráfica anterior, Los tipos más usados son las redes de neuronas convolucionales (CNN, Convolutional Neural Network), las redes de neuronas multicapa o profundas (DNN, Deep Neural Network) y las redes de neuronas recurrentes (RNN, recurrent neural network), incluyendo su variante LSTM (Long Short-Term Memory). Incluso aparece en un sexto puesto una combinación de redes recurrentes con redes convolucionales, CRNN. En el siguiente punto, vamos a analizar qué es lo que hace que estas dos arquitecturas de redes sean las más usadas, y a discutir cuál se adecua más al problema en cuestión.

Primero, cabe destacar que parece haber una tendencia desde el año 2009 a utilizar casi en exclusiva redes de neuronas convolucionales.

Redes de neuronas recurrentes

Como ya se ha comentado brevemente, estas redes utilizan la técnica de realimentar sus neuronas mediante bucles, permitiendo que la información persista a lo largo del tiempo.

Debido a esto, se convierten en un tipo de redes muy útiles para tareas de reconocimiento de procesos que van "avanzando" en el tiempo, como por ejemplo la escritura manual o el reconocimiento de voz. De esta manera, se consigue mantener algo de información que pueda dar sentido a los siguientes datos que entren en el sistema en la siguiente capa o época de entrenamiento.

Un subtipo de redes de neuronas recurrentes que se encuentra en auge y que fue descubierto a finales del siglo XX, son las redes LSTM.

Redes de neuronas LSTM

El mayor problema de las redes de neuronas recurrentes es que la información retroalimentada se almacena en una memoria virtual que se puede considerar no real, es decir, esta información se difumina con el resto de los datos ya que al mezclarse con otras entradas y modificarse por las conexiones, con el tiempo la información tiende a difuminarse. Las redes LSTM, por sus siglas en inglés, de memoria de corto y largo plazo, solucionan este problema añadiendo unas nuevas celdas de memoria que pueden almacenar información durante largos intervalos de tiempo, haciendo así que sean más efectivas a la hora de analizar dependencias temporales muy extensas, como, por ejemplo, en una canción.

Aun así, aunque cada vez sean más eficientes para tareas relacionadas con el habla como reconocimiento de voz o TTS, y destacando la importancia que tiene en la música la progresión seguida en el tiempo, se van poco a poco abandonando a favor de redes de neuronas convolucionales.

Redes de neuronas convolucionales

Uno de los objetivos de este trabajo, es analizar la cuestión comparándola con problemas similares, como la clasificación y segmentación de imágenes. Las redes de neuronas convolucionales utilizan como entradas una tabla de datos, lo que las hace idóneas para trabajos con imágenes por su naturaleza bidimensional. El aporte que hace a la arquitectura la entrada de dos dimensiones es que de cierta manera se informa previamente de que dos entradas situadas cerca en el mapa de entrada tienen o pueden tener cierta relación.

Una manera de visualizar un audio es mediante un espectrograma, una representación del sonido utilizando ventanas temporales y descomponiendo cada ventana en un espectro de

frecuencias. Los valores dibujados muestran los coeficientes para cada frecuencia en un instante concreto, siendo más rojo cuanto más intensa sea la frecuencia, y más azul cuanto menos lo sea.

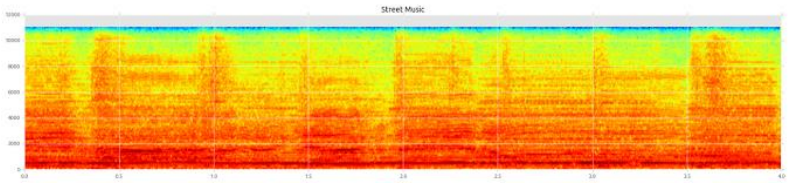


Fig 8. Muestra de un espectrograma de música callejera [17]

Con esta idea, existen algunos artículos como en el que aparece el espectrograma anterior, en los que se intentan clasificar sonidos usando como entrada una imagen, el espectrograma del sonido. Esta técnica realmente puede simplificarse eliminando el paso intermedio de convertir el espectrograma en una imagen, y directamente usar sus coeficientes como parámetros de entrada.

De esta manera, lo que en el análisis de imágenes eran las dos dimensiones espaciales, en el análisis de música las dos dimensiones que se usarán son la temporal, y la frecuencia.

2.5.4. Extracción de características

A la hora de obtener una red neuronal que funcione adecuadamente, es más importante escoger bien los datos con los que se va a alimentar a la red incluso que la propia arquitectura de esta. Vamos a comparar distintas posibilidades de datos o características que podemos emplear en este trabajo.

Amplitud de onda

El caso más simple de información es extraer la amplitud de onda de la canción para cada instante de tiempo, y alimentar a la red con una serie temporal de amplitudes. Los ficheros de sonido se codifican en su mayoría de esta manera, como una representación temporal de la onda, usando una frecuencia de muestreo de 44,1kHz [18], es decir, tomando el valor de la amplitud un total de 44.100 veces en un segundo. Tradicionalmente, las canciones oscilaban una longitud de unos 2:48 min. La razón de esto es que ese era el

tamaño que cabía en los discos de vinilo. Hoy en día, habiendo perdido esa limitación de grabación de la música, la canción media oscila entre los 3 y los 5 minutos. Tomando una media de 4 minutos a 44.100 muestras por segundo, acabaríamos con un total de más de 10 millones de datos de entrada para la red, lo cual es poco manejable.

Una solución consiste en utilizar una frecuencia de muestreo mucho menor, el problema es que no todas las conversiones entre frecuencias de muestreo son precisas, de manera que se podría perder información. Esto es debido a que, si la nueva frecuencia de muestreo no es un divisor exacto de 44.100, los valores que no recaerían en un punto exacto son aproximados mediante interpolación, que podría no ser del todo correcta.

Otra solución, consiste en utilizar solo un fragmento de la canción para determinar su género, pero en este caso el problema sería escoger ese fragmento. Actualmente, gran parte de las canciones de Pop contienen un fragmento en el que se introduce una pequeña sección de Rap, así que, si queremos evitar confundir la parte por el todo automatizando el proceso, tendremos de igual manera que analizar la canción completa. De este modo, esta métrica queda descartada para el problema.

Frecuencias

Una alternativa a la amplitud de onda sin entrar a parámetros muy complejos es utilizar las frecuencias. Como se describe en el punto **2.1. Ondas y música**, cualquier onda de sonido puede transformarse del dominio del tiempo al dominio de las frecuencias, es decir se puede definir una canción en vez de según su amplitud en cada instante, según su suma de frecuencias.

Esto elimina el problema de la temporalidad, ya que toda la información temporal se codifica en frecuencias, sin perder información relevante.

El problema que surge es que cuanta más precisión queramos conseguir, necesitaremos más frecuencias.

Escala de Mel

La escala de Mel [19] es una transformación aplicada sobre las frecuencias, transformándola de una escala lineal a una escala perceptual. Por ejemplo, un ser humano no percibe igual la diferencia entre 100Hz y 200Hz que entre 2.000Hz y 2.100Hz, aunque la distancia numérica entre ambos pares es igual.

Lo que se consigue con esta transformación, es que, por un lado, los valores introducidos a la red sean más "naturales", siendo estos más representativos de cómo los percibe una persona, y, por otro lado, consigue eliminar la cantidad de frecuencias necesarias para obtener la misma precisión, ya que las frecuencias más altas aportan menos información al oído.

El proceso de transformación de **h** Hercios a **m** Mels es el siguiente:

$$\boldsymbol{m} = 1127{,}01048 \ln\left(1 + \frac{\boldsymbol{h}}{700}\right)$$

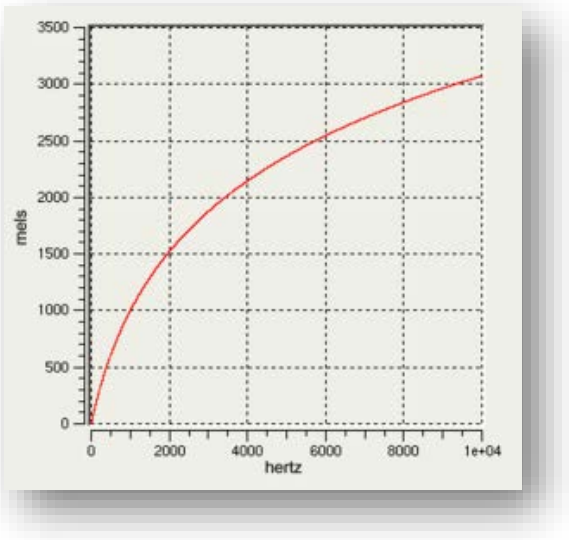


Fig 9. Gráfica de conversión de Hercios a Mels

MFCC

Los Coeficientes Cepstrales de las Frecuencias de Mel (Mel Frequency Cepstral Coeficients) [19] son unos coeficientes de representación de una onda de sonido derivados de los coeficientes de la escala de Mel. Para obtener estos coeficientes, el proceso es el siguiente:

1. Segmentar el sonido en tramos de longitud fija.

2. A cada tramo, aplicarle la transformada de Fourier discreta para separarlo en frecuencias y obtener la potencia espectral (o energía relativa) de cada frecuencia en el segmento
3. Aplicar la escala de Mel a los espectros obtenidos en el punto anterior.
4. Tomar el logaritmo neperiano de cada coeficiente de Mel obtenido.
5. Aplicar la transformada de coseno discreta a cada uno de estos logaritmos

Así obtenemos para cada canción información temporal con cada segmento, e información de las frecuencias con cada coeficiente obtenido en cada segmento. De esta manera, aunque manejando más datos, la información temporal se difumina menos que utilizando simplemente los coeficientes de Mel. Como el resultado de calcular los MFCCs es una matriz bidimensional, esto los hace idóneos para trabajar con una red de neuronas convolucional.

Parámetros "naturales"

La última opción es utilizar otra serie de parámetros, más comprensibles de manera directa por una persona, como puede ser tomar para cada instante el volumen, la energía del sonido, el tono, o parámetros más complejos como el XCR (Zero Crossing Rate) que contabiliza cuántas veces cruza una onda el punto de amplitud cero en un tramo.

El problema que presenta la utilización de estos parámetros es que requieren mucho más tiempo de cálculo que por ejemplo los MFCCs, y de igual manera son deducibles a partir de ellos al menos utilizando una máquina. Por ejemplo, el ZCR y el tono se encuentran codificados en la frecuencia, y la energía y el volumen se codifican en la amplitud, ambos resumidos y condensados en los MFCCs.

Después de analizar los tipos de redes y los parámetros que se pueden utilizar para análisis musical, y respaldado por numerosos estudios como [20], puedo concluir que el método más eficiente a seguir es codificar la música usando los MFCCs, y trabajar con una red neuronal convolucional que entienda la representación en dos dimensiones de estos.

2.6. Trabajos similares

En este punto voy a comentar brevemente los estudios que más han inspirado el presente trabajo, comparando sus conclusiones con las alcanzadas en este documento.

Music Genre Classification using Deep Neural Networks – J.S. Albert & J.T. Ferran [21]

En este trabajo, los autores Ferrán José Torra y Albert Jiménez Sanfiz estudian un sistema que clasifique por géneros una canción introducida. Para ello proponen segmentar las canciones y usar todos los fragmentos para el proceso. Otros estudios utilizan tan solo la parte principal de una canción para clasificarla, pero esto supone una importante pérdida de información del resto de la canción. Otras de las ventajas que ofrece su propuesta de segmentación, es que, en entrenamiento se dispone de más datos, y que en la fase de pruebas los resultados serán más acertados al poder calcular una media y determinar una clasificación acertada para el total de la canción y no solo para un fragmento.

Para programar la red neuronal, utilizan Keras y Theano, descritos más adelante en el punto **3.1. Frameworks de desarrollo**. La red empleada es una red convolucional recurrente, CRNN. Su dataset es un dataset casero consistente de 30 canciones de diez géneros distintos. Estos diez géneros corresponden con los diez géneros de GTzan.

Para la fragmentación, dividen la canción en trozos de igual longitud, deshaciéndose de los extremos sobrantes al principio y al final de la canción que no contienen información relevante.

Tras el estudio, concluyen que, para canciones de larga duración, el sistema de fragmentación produce resultados considerablemente mejores que utilizando un único recorte. Además, incluyen una matriz de confusión de géneros que es interesante analizar:

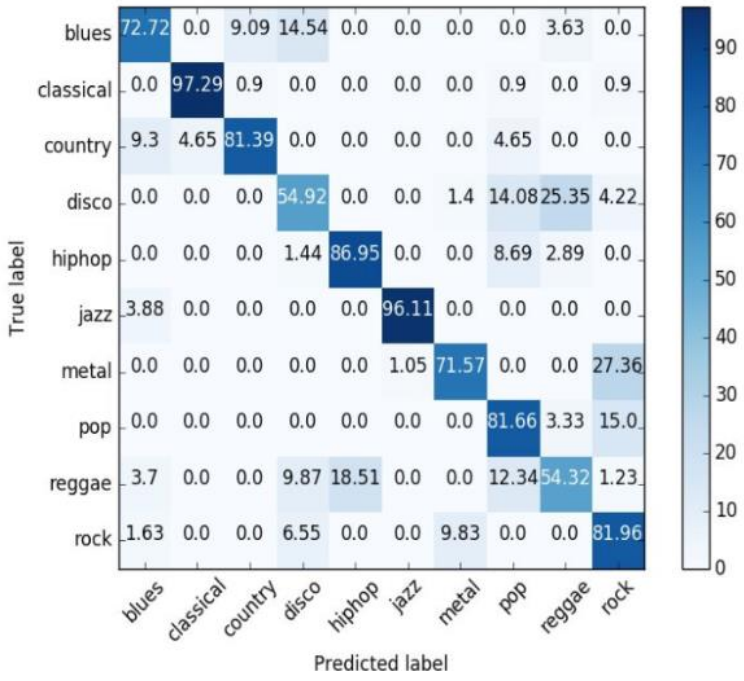


Fig 10. Tabla de confusión entre géneros en el estudio [21]

En la imagen, se observa que existen géneros que son fácilmente confundidos entre ellos. Ejemplos notables de esto son Rock con Metal con una precisión de entre el 73% y el 90%, y disco con reggae, con una precisión de entre 75% y 90%. Esta confusión parece ser debida a la instrumentación similar entre estos géneros, y a que en ocasiones es complicado incluso para una persona dibujar una línea separadora entre los elementos de ambos pares.

Genre Classification of Songs Using Neural Network – Masood S. [22]

En este trabajo, también con el objetivo de clasificar música por géneros, los autores utilizan un dataset de 400 canciones de dos géneros, música india y música Clásica, con 200 canciones cada uno. Los datos que extraen de cada canción son los coeficientes de MFCC de la canción completa.

El aporte de este estudio es que implementan una capa intermedia que aprende a transformar los MFCCs en otra serie de parámetros de la canción, entre los que se encuentran los pulsos por minuto (BPM), la cantidad de voz, la cantidad de instrumentos,

e incluso un parámetro muy curioso al que llaman “Bailabilidad”, y que definen como la capacidad que tiene una canción de ser bailada.

La red que utilizan es un perceptrón multicapa, uno de los sistemas básicos de red neuronal multicapa.

Los resultados obtenidos no son muy precisos (Un 87% para identificar una canción de música clásica, y un 82% para una de música india), sobre todo considerando que solo diferencia entre 2 géneros, lo cual debería ser una tarea mucho más sencilla que con más categorías.

Este estudio sirve como muestra de que existen métodos más efectivos que una red neuronal multicapa simple, y transformar los MFCCs en parámetros distintos como los comentados. De esta manera y comparando con el estudio anterior, es fácil concluir que usar los MFCCs como parámetros de entrada de la red es la decisión más acertada.

Recommending Music on Spotify with Deep Learning – Sander Dieleman [23]

Este estudio utiliza una red convolucional con el objetivo de analizar si es posible utilizar una red neuronal para crear listas de reproducción de recomendaciones en plataformas como Spotify en base a la música que escuchen habitualmente. La idea consiste en entrenar una red con las canciones más escuchadas de un usuario, observar su funcionamiento, y considerar cómo ofrecer listas de recomendaciones similares.

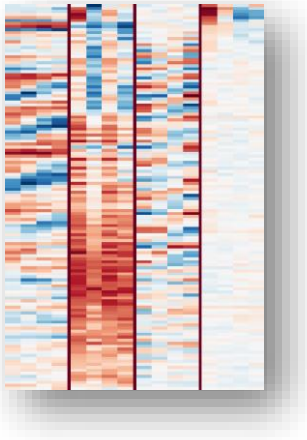
De nuevo, emplea los MFCCs como parámetros de entrada, pero esta vez extrayéndolos de solo un fragmento de cada canción. La salida consiste en una clasificación por géneros.

Lo interesante de este estudio se encuentra en el análisis sobre qué está aprendiendo la red. Para comprobarlo, analiza las salidas de las neuronas de cada capa individualmente, introduce una serie de canciones, y genera listas de reproducción que tengan en común que activan una neurona concreta de esa capa. De esta manera, observa cómo según avanza en las capas, las capas más profundas destacan características más específicas, como géneros, y las capas iniciales encuentran patrones en la sonoridad de la canción.

Por ejemplo, en la primera capa, descubre que las neuronas 14, 250, y 253 identifican vibrato, terceras vocales (Voces cantando una sobre otra a una distancia específica de 3 tonos) y percusión grave. Así, en la siguiente capa identifica afinaciones completas de la canción e incluso acordes. Por último, en la capa anterior a la capa de salida consigue identificar unos prototipos de géneros finales realmente específicos, como Góspel, Rock cristiano, Chiptune o Pop chino.

Lo que se puede concluir a partir de este estudio, es que la manera de funcionamiento de la red consiste en identificar las técnicas y sonidos presentes en esta, y mediante su combinación poco a poco acabar obteniendo una clasificación por géneros, de manera parecida al planteamiento que haría una persona de identificar instrumentos, identificar la manera en que se canta, y acabar concluyendo el género.

Una muestra de los coeficientes de MFCC para algunas de estas salidas es la siguiente:



En el diagrama se muestran las frecuencias en el eje 'y', el tiempo en el eje 'x', y los coeficientes en un gradiente entre rojo y azul, siendo rojo el mayor y azul el menor.

Las dos muestras más sencillas de entender son la tercera y la cuarta columna. La tercera, captura terceras vocales, y por tanto se puede observar cierta periodicidad entre las líneas paralelas de colores que representan frecuencias a igual distancia. La cuarta, detecta sonidos graves, así que activa las frecuencias más bajas (Situadas en la parte superior)

Fig 11. Muestra de MFCCs en [23]

3. Desarrollo y funcionamiento final

En este punto describiré las conclusiones alcanzadas para el desarrollo de la prueba del proyecto, las decisiones finales sobre qué framework, qué metodología de proceso, y que manera de funcionamiento he tomado, y cuál es el proceso desde que se escoge una canción hasta que se obtiene una salida de la red.

3.1. Frameworks de desarrollo

Tomando otra vez como referencia a [3], tenemos la siguiente gráfica:

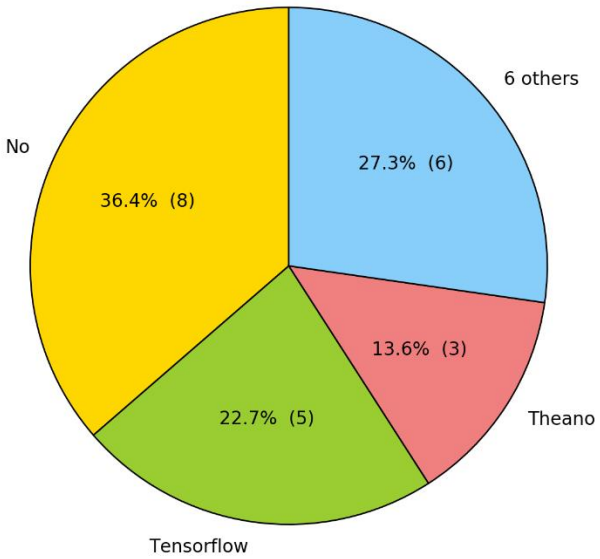


Fig 12. Proporción de frameworks en proyectos de machine learning con música

En ella se ve que la decisión más habitual es no usar ningún framework específico, es decir programar todas las funcionalidades de la red manualmente. Ya que mi diseño de red es uno general y que responde a una arquitectura común, lo más sensato es que existiendo librerías como TensorFlow que aceleran el proceso de construcción de la red, mejor me centre en este tipo de frameworks. A continuación, voy a discutir sobre distintos frameworks para decidir el mejor:

TensorFlow

Tensorflow es la librería de aprendizaje automático de uso más extendido. Desarrollada por Google y de código abierto, está disponible para múltiples lenguajes de programación, pero el más habitual es Python. Esta librería se desarrolló originalmente para completar los proyectos del propio Google, pero actualmente también se usa en proyectos de investigación y desarrollo en todo el mundo. Su nombre proviene de los tensores, un objeto matemático que es utilizado por las redes neuronales que se construyen con esta herramienta para sus operaciones sobre los conjuntos de datos.

A pesar de estar tan extendido, TensorFlow es complicado de entender, ya que usa términos y conceptos (Como las sesiones y los propios tensores) que no son habituales. Siendo la librería más usada de machine learning, también existe una enorme comunidad a su espalda, lo que facilita encontrar recursos o soluciones a los problemas que puedan surgir.

TensorFlow está preparado para poder ejecutarse sobre gran variedad de dispositivos y arquitecturas, como CPUs, GPUs, o clústeres de servidores.

Theano

Theano es una librería de Python que pone a disposición del programador una serie de funciones que permiten manipular y evaluar expresiones matemáticas, en especial expresiones matriciales. Al igual que TensorFlow, Theano está diseñado para poder ejecutarse tanto sobre CPU como sobre GPU.

Aunque su foco son los cálculos matemáticos, Theano también puede utilizarse para la construcción de sistemas de machine learning, aunque desde bajo nivel.

PyTorch

PyTorch es una librería de machine learning de código abierto desarrollada por Facebook. PyTorch está diseñado para solucionar principalmente problemas de procesamiento de lenguaje, y tareas de visión artificial, por lo que para el problema de clasificación musical por géneros no es suficientemente potente.

Al igual que TensorFlow, PyTorch opera utilizando tensores. PyTorch está construido sobre Torch, otra librería basada en Lua que provee una amplia variedad de algoritmos

de machine learning, por tanto, PyTorch actúa como un acercamiento de las expresiones de Torch, más complejas y de bajo nivel, a un lenguaje más comprensible y fácil de manejar.

Keras

Keras es una librería también para el lenguaje de programación Python que se ejecuta sobre otras librerías de aprendizaje automático como Tensorflow y Theano, y que aporta una visión simplificada para construir redes de neuronas en menos tiempo que partiendo de la base de los sistemas sobre los que se ejecuta.

Keras aporta las funcionalidades básicas de creación de capas, ya sean simples, capas convolucionales, o capas recurrentes, pero no permite una experimentación más profunda como definir funciones de activación concretas o conexiones entre neuronas que salgan de lo habitual.

Para este trabajo, la arquitectura de la red ya está definida, y no es necesario experimentar con nuevos métodos, funciones de activación, o estructuras de red. TensorFlow proporciona una base robusta para cálculos de machine learning, y Keras simplifica el prototipado de redes, y permite la suficiente personalización de la red necesaria para este proyecto sin alcanzar complejidades demasiado altas para un problema simple, por tanto, decido utilizar como framework el lenguaje de programación Python, y para cálculos de machine learning Keras montado sobre TensorFlow.

Por último, habiendo escogido ya el lenguaje de programación que utilizaré, necesito decidir sobre qué librería o librerías para trabajar con ficheros de audio utilizaré. Las librerías más potentes disponibles para Python, y además dos de las más utilizadas, son PyDub y LibRosa. Para mi proyecto, utilizaré LibRosa para procesamiento de audio, como por ejemplo extraer las características, y PyDub.

3.2. Decisiones finales para el desarrollo

- **Fragmentación:** Una vez la red ya ha sido entrenada, para obtener una salida de género para la canción, segmentaré esta en fragmentos de igual longitud y cada uno de ellos será analizado y ecualizado individualmente, manteniendo el máximo

de información posible de la canción, y pudiendo aplicar distintas ecualizaciones a cada segmento con sonido diferente de esta.

- **MFCCs:** Los parámetros de entrada que recibirá la red serán los Coeficientes Cepstrales de Frecuencia de Mel. Esta decisión se fundamenta en que son unos coeficientes rápidos de calcular, eficientes, representativos de la percepción humana, y que representan información tanto de las frecuencias como de la sucesión temporal de estas en el fragmento analizado.
- **Red convolucional**: La red escogida para este problema es una red de neuronas convolucional. Este tipo de red funciona bien para trabajos del ámbito de la música ya que mantiene las relaciones entre frecuencias e instantes de tiempo próximos, que a su vez se encuentran representados en los MFCCs.
- **Clasificación:** El problema se abordará añadiendo un paso intermedio de clasificación, que permitirá que los cálculos iniciales se realicen previamente y que a su vez facilitará que la propia ecualización se produzca más rápidamente llegando incluso a ejecutarse en tiempo real.
- **Ecualización:** La ecualización se realizará asignando unos perfiles de ecualización a cada uno de los géneros de clasificación, de manera que la canción (O fragmento de esta) se ecualizarán en base al perfil correspondiente.
- **Mezclar géneros:** Se considerará la mezcla de géneros, es decir, al existir algunos casos de confusión, se añadirá entre los cálculos la posibilidad de que, en vez de ecualizarse en base al género más probable, se ecualice proporcionalmente en base a los n géneros más probables.
- **Dataset GTzan:** El dataset que se utilizará para el entrenamiento inicial de la red, será el dataset de GTzan, ya que contiene fragmentos de canciones completos y no solo algunas características ya extraídas, permitiendo así experimentar un poco más con los datos que se extraen y utilizan de cada canción.
- **Géneros GTzan:** El conjunto de géneros escogido es el de los diez ofrecidos por el dataset de GTzan, no solo por ser los disponibles en el dataset que se va a usar, si no por su extendido uso en gran parte de los trabajos sobre música y aprendizaje automático, y por los buenos resultados que muestran en estos.
- **Python:** El lenguaje de programación escogido es Python, por un lado, por su potencia para operaciones matemáticas, sobre todo vectoriales, y por las potentes librerías de machine learning que ofrece.
- **TensorFlow:** Tensorflow es escogido como base para los procesos de machine learning en Python por su variedad de recursos disponibles, por su potencia, y por la posibilidad de utilizar otras librerías como Keras sobre esta para simplificar la construcción de la red.
- **Keras:** Se escoge Keras como librería de alto nivel montada sobre TensorFlow por su facilidad de uso y por disponer de todos los requisitos necesarios para poder construir una red neuronal convolucional a medida de este problema.

3.3. Arquitectura de la red empleada

Como ya se ha comentado anteriormente, la arquitectura de la red escogida es una red de neuronas convolucional. En este apartado, describiré más a fondo la arquitectura y funcionamiento de la red.

La red tomará como entrada un conjunto de MFCCs. Acorde a múltiples estudios como [26] el número óptimo de frecuencias a tomar para calcular los MFCCs se encuentra entre 10 y 20. En mi caso, utilizaré 13 respaldado por otros estudios como [23]. El consenso sobre cuántas ventanas tomar en cada fragmento es de cerca de una ventana por cada 0,05 s de audio, así que usando muestras de 30 s de duración usaré también 600 tomas de ventanas por canción. De esta manera, la red tomará como entrada un total de 600 *13 parámetros, es decir.

El resultado al que se deberá llegar es de 10 salidas, una representando cada uno de los 10 géneros por los que se va a clasificar cada canción.

Tanto en [24] como en [23] se utiliza una red convolucional con 3 capas intermedias, pero tal y como parecen indicar los resultados de este último en el que la tercera capa parece ya llegar a clasificar por géneros, optaré por experimentar con una red con 2 capas ocultas intermedias.

De esta manera, cada una de estas capas aplicará un max pooling y un dropout. El max pooling, consiste en aplicar una reducción de parámetros, con el objetivo de que según se adentra en la red, cada vez se obtengan menos datos y la red empiece a concretarse en una salida determinada (Por ejemplo, clasificar 7.800 parámetros en solo 10 géneros). Por su lado, el dropout consiste en eliminar cierto porcentaje de conexiones entre capas en cada iteración, de manera que la red empieza a adaptarse no solo a su función de clasificar, si no que cada capa empieza a aprender a solucionar errores de capas anteriores si los hubiera. Al manipular ambos datos hay que tener cuidado, ya que un valor muy extremo de ellos podría llevar a resultados malos de la red.

Por último y justo antes de la salida final, se utilizará una capa densa (O completamente conectada) en la que no se aplicará ningún max pooling, es decir, mantendrá todas las conexiones con la capa anterior.

Finalmente, esta capa aplica una función ReLU (Rectified Linear Unit), la cual es igual a 0 cuando su entrada es menor o igual a 0, y es lineal cuando la entrada es positiva.

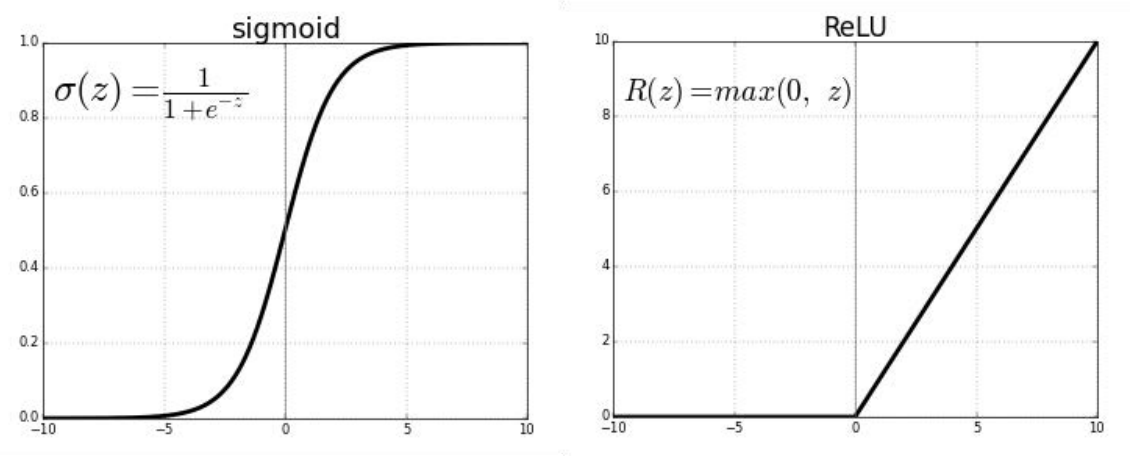


Fig 13. Representación de las funciones Sigmoide y ReLU

Tras estas capas, ya se obtiene la salida final de la red, un valor numérico para cada uno de los 10 géneros a la que se le podrá aplicar una función Softmax para "exagerar" el género más probable, o una función probabilística menos abrupta para obtener una probabilidad variada para cada género.

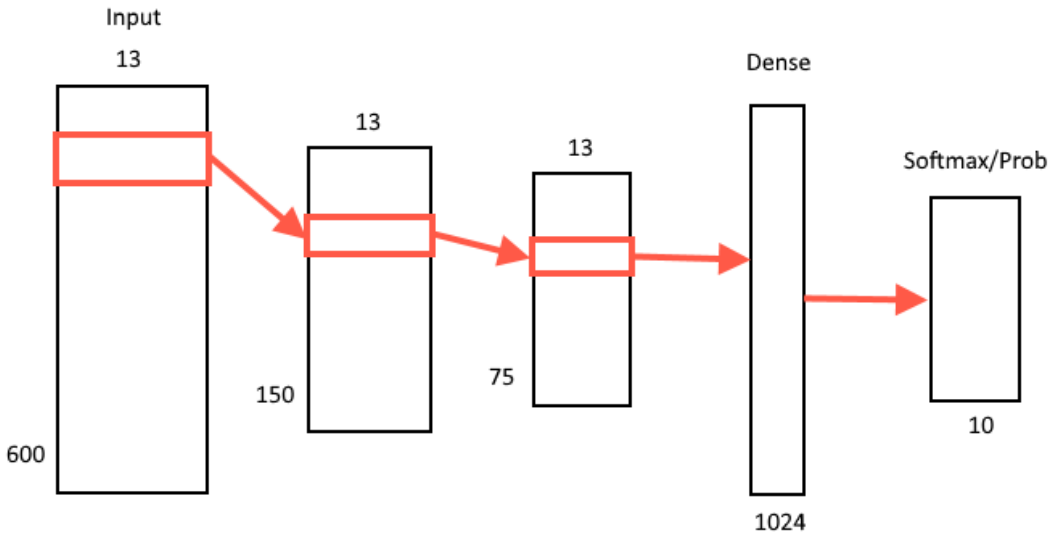


Fig 14. Esquema de la red neuronal escogida

3.4. Métodos, proceso de ejecución, clases, entradas y productos

El proceso de ejecución se va a subdividir en 4 procesos. El objetivo de esta subdivisión es independizar los cálculos, de manera que se puedan ejecutar en momentos distintos o incluso en sistemas separados, pudiendo optimizar el tiempo de espera. Como ya se ha comentado en el punto anterior, la red se basará en el proyecto de GitHub del estudio referenciado en [24], pero modificando los parámetros de la red, añadiendo los procesos de ecualización, prueba y experimentación, y fragmentando el sistema en módulos más independientes. Los procesos escogidos son los siguientes:

Preprocesado

En este proceso, el objetivo es extraer las características del dataset de canciones para su manejo por la red.

El dataset de GTzan, lo organizaré en 10 carpetas, nombradas cada una por un género, y conteniendo las 100 canciones de cada uno de estos. El proceso iterará por cada una de estas carpetas generando un fichero de información para cada una de las canciones. En el caso de las canciones de entrenamiento, usará la longitud completa de cada fichero (30 segundos) para generar los coeficientes MFCC, que se almacenarán en ficheros con extensión .pp con el mismo nombre. Para recopilar todos los datos, luego se iterará sobre cada uno de estos ficheros .pp, acumulándolos en un único fichero que representa una tabla de datos, donde cada fila es una entrada de información (Una canción) y cada columna representa cada uno de los coeficientes de los MFCCs.

Para los ficheros de prueba que se usan para obtener los resultados finales, primero se fragmentarán en ventanas de 30 segundos también, obteniendo un set de datos para cada fragmento de la canción. Incluyo este proceso en este punto por el uso de las mismas clases y métodos en el código, aunque temporalmente se ejecutaría en el proceso de "Prueba". Cuando la canción se divide en fragmentos, lo habitual es que el extremo final dure menos de 30 segundos de duración, así que siempre que alcance una longitud mínima de 20 segundos, se seguirá considerando. Si no supera la longitud mínima, se descarta y se le aplica la misma clasificación que al fragmento que lo precede.

Para medir el tiempo medio de ejecución de esta parte, tomaré medidas para la fragmentación y lectura de una canción, y luego para el procesado. Para determinar si este

tiempo depende de la longitud del fragmento, haré pruebas con fragmentos de 30 y de 120 segundos.

30 segundos carga	120 segundos carga	30 segundos procesado	120 segundos procesado
9,24 s	8,36 s	20,44 s	19,70 s
9,53 s	10,35 s	18,26 s	19,76 s
8,88 s	9,29 s	18,90 s	18,37 s

Por lo que parece, la longitud de la ventana no afecta al tiempo individual, por lo menos no de manera significativa. Para la carga de 30 segundos, se tarda de media 9,22 s, mientras que, para ventanas de 120 segundos, la media es de 9,33 s. Para el procesado, en ventanas de 30 segundos la media son 19,2 s, y para 120 segundos la media es de 19,27. Por tanto, el tiempo medio de carga y procesado para ventanas de 30 segundos son unos 28,42 s, y para ventanas de 120 segundos son 28,5 s, una diferencia mínima que se puede considerar despreciable. Por ello, podemos concluir que el tiempo medio de procesado para un fragmento es de aproximadamente 28,5 s.

Entrenamiento

El entrenamiento es sin duda el proceso más lento y costoso del proyecto. Como se ha visto en el punto anterior, para procesar un fragmento de 30 segundos se tarda de media aproximadamente 28,5 s, así que, contando con un dataset de 1000 canciones, el tiempo de procesado será de 7 h 55 min, lo que coincide aproximadamente con el tiempo real medido de 8 h 3 min.

El entrenamiento consiste en iniciar la red con unos parámetros aleatorios, y entrenarla con el dataset completo durante 100.000 iteraciones.

Para ello, en cada iteración primero se entrenará utilizando aleatoriamente los datos de 800 de las 1.000 canciones, y luego se calculará la tasa de acierto pasando la red entrenada a las 200 canciones restantes. Al acabar con las 100.000 iteraciones, se almacenará el estado de la red en el que se consiguió un mejor resultado de entre las 100.000, que no necesariamente será el estado final.

Sin contar el tiempo de Preprocesado, el entrenamiento tardó 53 h 23 min en el primer entrenamiento, y 52 h 48 min en el segundo. Afortunadamente, solo es necesario entrenar la red una única vez, porque ya se consiguen unos resultados aceptables. Los resultados

obtenidos no son del todo satisfactorios, ya que produce una tasa de acierto de cerca del 60%. Para algunas canciones, como por ejemplo piezas de música clásica, la tasa de acierto llega incluso a 93,2% de acierto, pero existen géneros y grupos de géneros que reducen enormemente la tasa hasta medias de hasta 40,31% para canciones de rock y metal. Los datos utilizados para comprobar la efectividad de la red corresponden a un subconjunto del dataset MSD. Algunos géneros son compartidos tanto por GTzan como por MSD, pero para los géneros distintos he utilizado canciones obtenidas mediante su búsqueda en la sección de canciones más populares de Spotify. Para esta comprobación, he usado no solo uno si no los dos modelos entrenados y he obtenido una media de ambos. La tabla con la tasa de aciertos para cada género con el conjunto de pruebas con la red ya entrenada se encuentra en el Anexo II.

La duda que surge con este resultado es si realmente toda esta confusión supone un problema. Si aplicamos mezcla de géneros y ponderamos equitativamente entre aquellos que la red determina que son los más probables, es posible que obtengamos mejores resultados de ecualización. También es posible que realmente los géneros que producen confusión sean lo suficientemente similares entre ellos como para que igualmente aplicar solo la ecualización del más probable ya consiga resultados suficientemente buenos.

Procesado de la prueba y predicción

Este módulo consiste en preprocesar una canción individual, pasar los datos obtenidos por la red ya entrenada, y obtener unos resultados de clasificación. En este punto, solo se comenta el proceso que siguen los datos y los cálculos internos que se realizan, los resultados de estas pruebas se detallan en el punto **4. Experimentos**.

La ejecución consiste en tres pasos: Fragmentar, preprocesar, y obtener resultados. Para decidir el tamaño de fragmentación, se utilizan dos umbrales, un tamaño máximo de fragmento, y un tamaño mínimo. El tamaño máximo se corresponderá con 30 s, ya que es la misma longitud con la que está entrenada la red, y de la que ha aprendido a clasificar por géneros. También se define un tamaño mínimo de 20 segundos, ya que una medida inferior no contiene datos suficientes para los cálculos de los MFCCs. De esta manera, la canción se fragmentará en trozos de 30 segundos hasta llegar al último, de menor longitud. En este caso, existen dos opciones:

- El tamaño del ultimo fragmento es menor que el mínimo. En este caso, el último fragmento tomará como clasificación la misma que se haya obtenido para el anterior fragmento.

- El tamaño del ultimo fragmento es mayor que el mínimo. En este otro caso, este fragmento se procesará de igual manera y se obtendrá una clasificación específica para él.

El siguiente paso consiste en preprocesar los fragmentos obtenidos. Como el tiempo medio de carga y procesado es de 28,5 s por fragmento, la fórmula para obtener el tiempo total de procesado para una canción es la siguiente:

$$t = 28{,}5 \times \left\lceil \frac{duraciónTotalDeLaCancion}{30} \right\rceil$$

Si el resto de dividir la duración de la canción entre 30 es mayor que 20 y

$$t = 28{,}5 \times \left(\left\lceil \frac{duraciónTotalDeLaCancion}{30} \right\rceil - 1 \right)$$

Si el resto de dividir la duración de la canción entre 30 es menor que 20

Observando las dos fórmulas, se deduce que el tiempo de procesado siempre será mayor en la primera que en la segunda. Para cada fragmento de 30 s, se tardan 28,5 s de cálculo, la duda existe en el último fragmento. Si su longitud se encuentra entre 0 s y 20 s, este no se procesará y el tiempo de ejecución será siempre menor al tiempo de reproducción de la canción. Sin embargo, si su longitud es de entre 20 s y 30 s, su tiempo de procesado será igualmente de 28,5 s, así que existe la posibilidad de que el tiempo de procesado completo sea mayor al tiempo de la canción.

Por ejemplo, en una canción de 2 min y 21 s, contaremos con 4 fragmentos de 30 s y un último fragmento de 21. Como son 5 fragmentos, el tiempo total de cálculo son 142,5 s, o lo que es lo mismo, 2 minutos con 22,5 s. A continuación, una muestra visual de esta idea usando la calculadora online de desmos, que permite dibujar funciones:

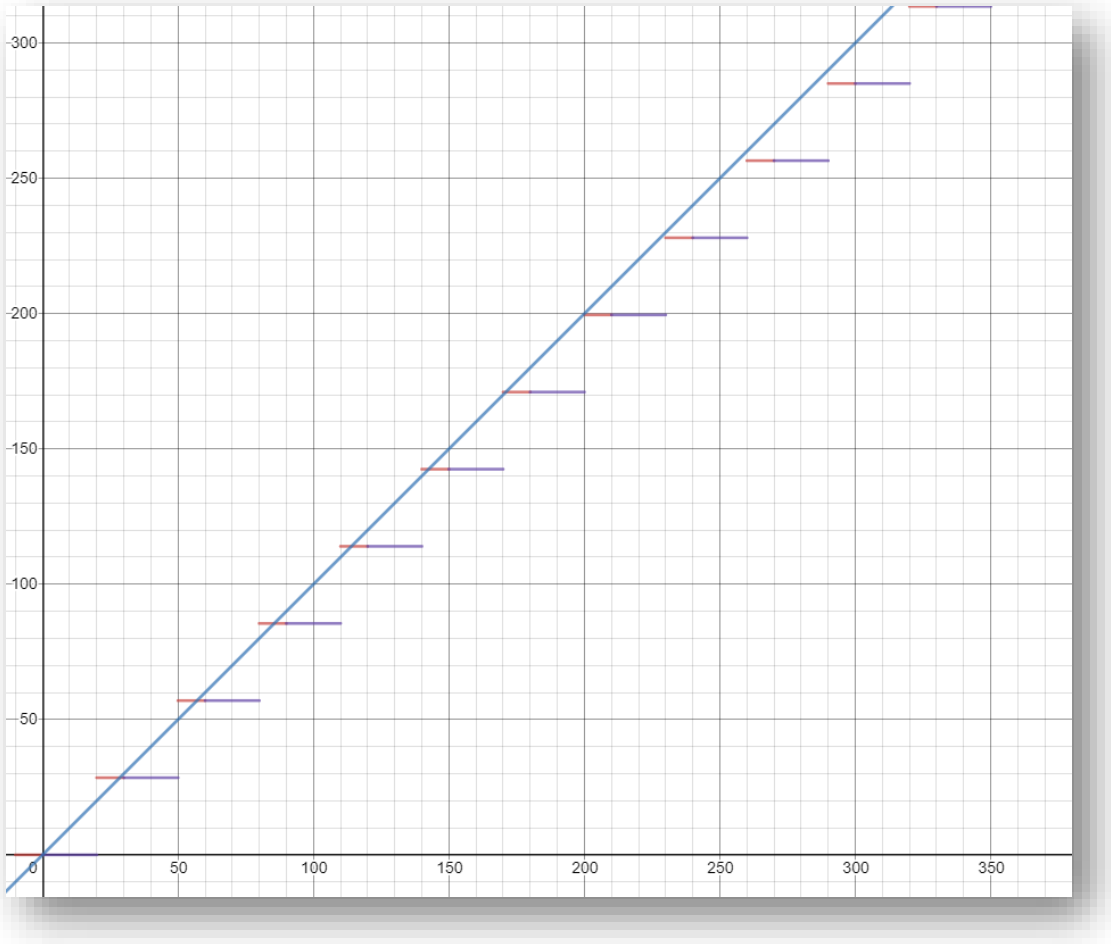


Fig 15. Gráfica de representación de tiempo de ejecución junto a tiempo de reproducción

La línea azul muestra la longitud de la canción, mientras que los segmentos morados y rojos muestran el tiempo de procesado para el caso de menos de 20 s y para el de más de 20 s respectivamente. En la gráfica se observa que en los casos de la línea morada (ultimo trozo menor de 20 s) siempre será menor el tiempo de ejecución que el de procesado. Si pintamos sobre la gráfica los segmentos en los que el tiempo de ejecución es menor al de procesado, obtenemos lo siguiente:

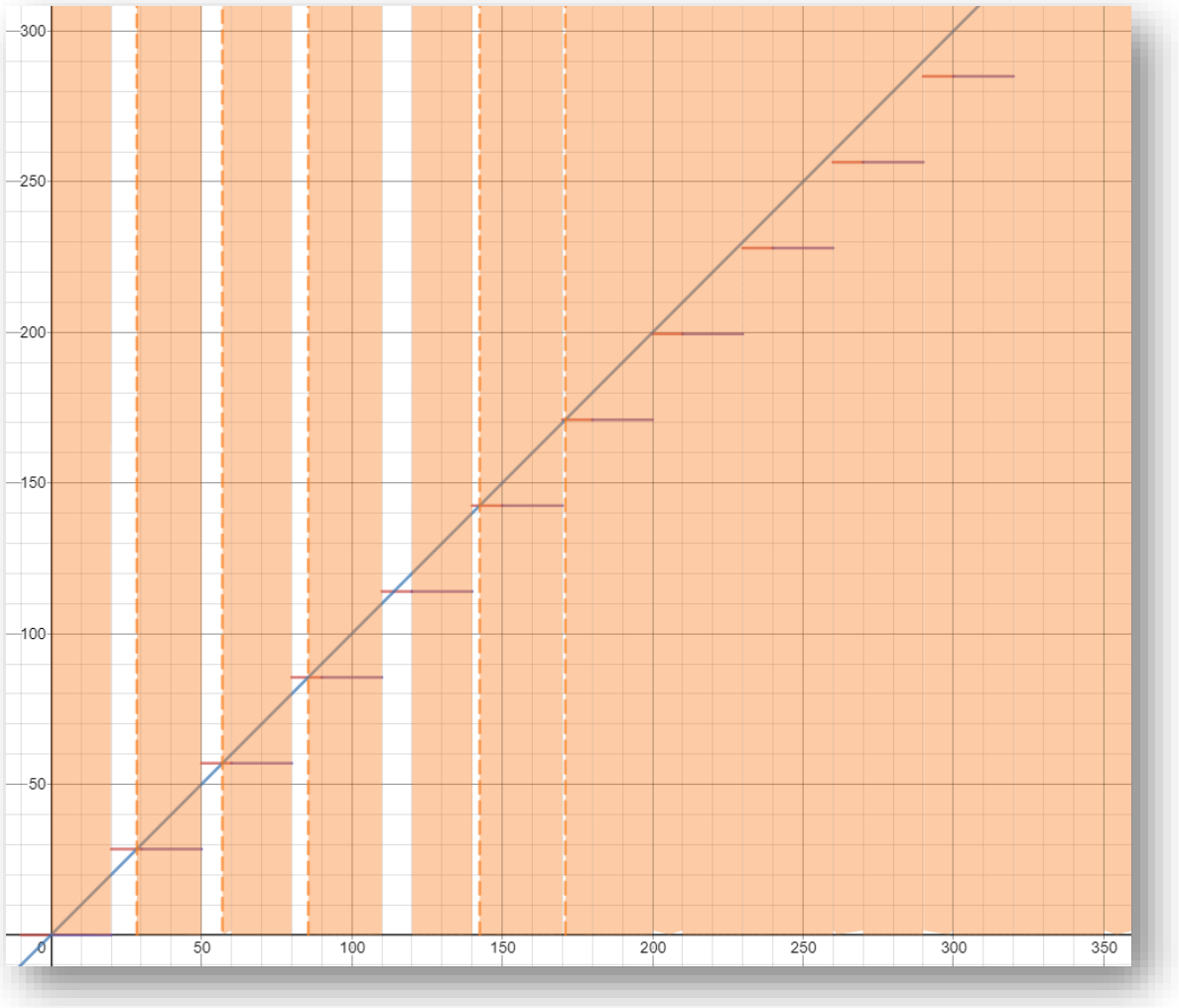


Fig 16. Gráfica de intervalos en los que el tiempo de ejecución es menor al tiempo de reproducción

En esta otra gráfica, se observa que, a partir de 171 s de duración, siempre el tiempo de reproducción será mayor al de procesado.

Finalmente, una vez obtenidos los datos de los fragmentos de la canción, cada uno pasará individualmente por la red emitiendo unos resultados. Con el fin de analizar distintas soluciones, primeramente, probaré a obtener la salida con un único género, y luego con los 3 géneros más probables según la red. La elección de un único género se realizará mezclando ecualizaciones, pero aplicando una función softmax a la salida de la red. Esta función “exagera” las diferencias, consiguiendo que la distribución de géneros alcance

casos de casi 100% de probabilidad, dejando el resto de los géneros con valores muy bajos. Para el método de top 3 géneros, se aplicará una función probabilística sobre los 3 con una salida más probable de la red.

Además, también probaré a utilizar un método de interpolación para conseguir ecualizaciones que vayan suavemente transicionando unas entre otras.

Interpretación de resultados

Este último módulo obtiene los resultados generados por los anteriores módulos, los interpretará, y representará gráficas y diagramas con el fin de comprender el funcionamiento y determinar qué estrategias han sido válidas y cuáles han fracasado.

Primero, se genera una gráfica mediante el algoritmo t-SNE para representar cómo afectan los MFCCs a la clasificación de cada género:

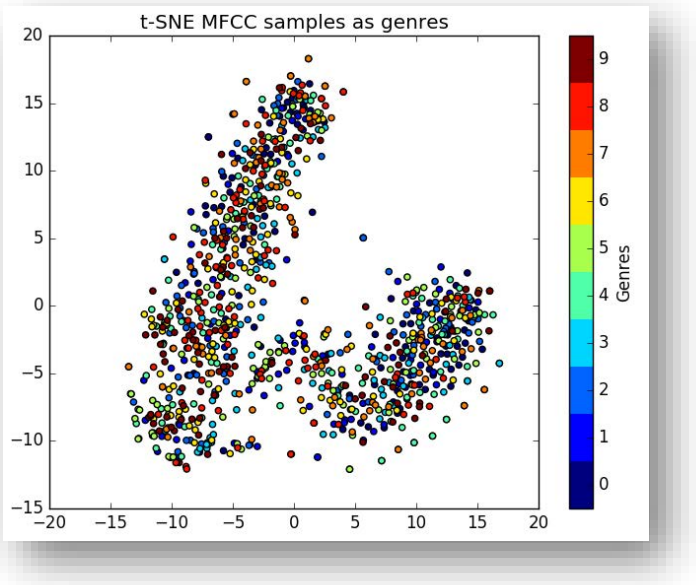


Fig 17. Gráfica T-SNE de relevancia de los MFCCs para los géneros

El algoritmo t-SNE reduce un espacio vectorial de n dimensiones a una representación gráfica de 2 dimensiones, tratando de mantener la muestra de cúmulos y la distancia entre puntos. Para este caso, la cantidad de dimensiones es extremadamente elevada, así que cabe esperar que la representación no sea muy acertada. Además, los coeficientes de

MFCC por sí mismos no aportan mucha información, es su combinación la que determina el género. Estas sospechas se confirman observando la gráfica, en la que apenas se distinguen cúmulos y todos los colores aparecen mezclados.

Además, también se obtienen gráficas para comparar los métodos de mezcla de géneros y de interpolación. Estas gráficas están descritas y discutidas en el punto **4. Experimentos**.

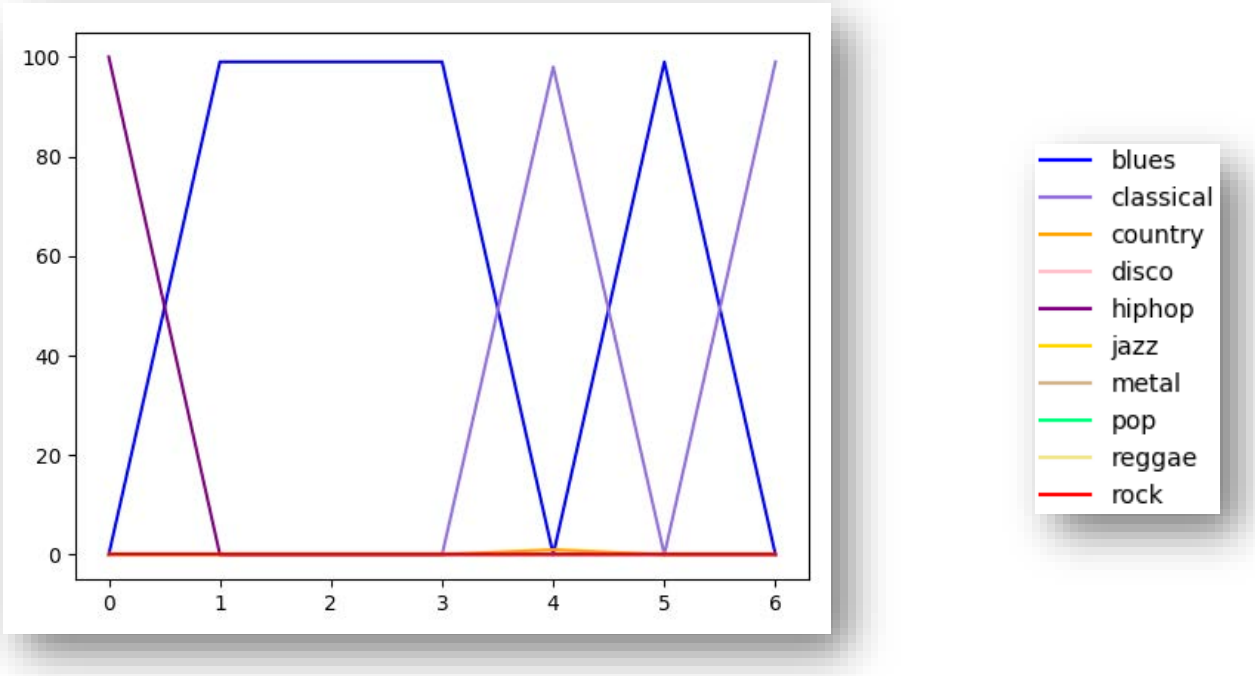
4. Experimentos

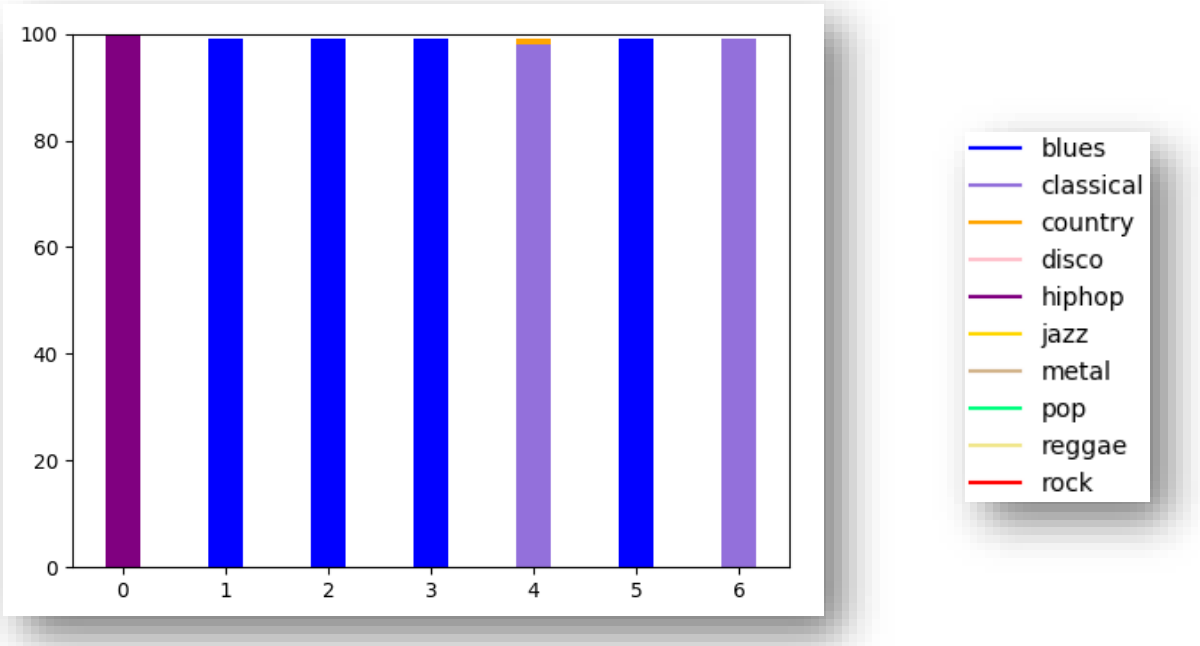
Para la prueba de funcionamiento, he decidido utilizar una canción que en mis pruebas personales durante el desarrollo ha producido mucha confusión entre géneros, sobre todo entre Country, Blues y Clásica. La canción es Gravity, del grupo Against the current. La música de este grupo, y en concreto esta canción, suele ser Pop-Rock, lo cual hace más curioso que la confusión se produzca entre géneros que no sean ni Pop ni Rock.

Primero, compararé las salidas de clasificación para el caso de Softmax frente al modelo Probabilístico, y luego compararé los resultados de aplicar o no interpolación

4.1. Clasificación

Resultados Softmax

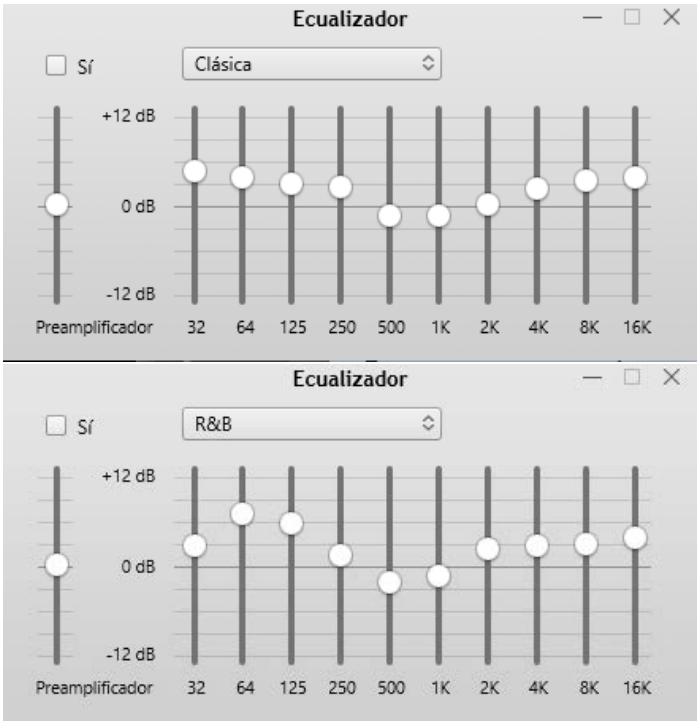




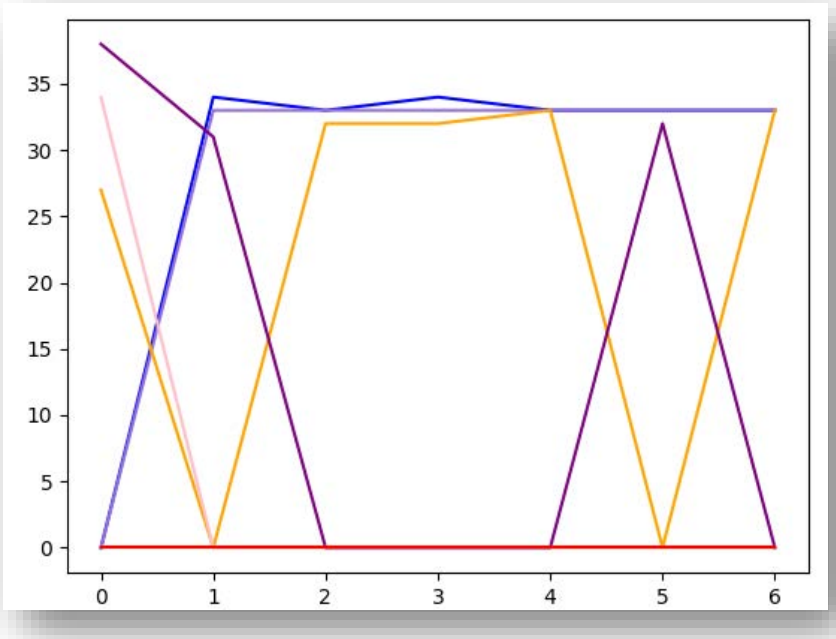
Ambas gráficas representan la probabilidad (eje y) para cada segmento de tiempo (eje x) de corresponder a un género o a otro. Cada color corresponde a un género según su color. La primera gráfica incluye una representación de la probabilidad individual de cada género en cada instante, mientras que la segunda representa las probabilidades en un diagrama de barras acumulativo, donde cada género aparece junto a los demás en cada barra.

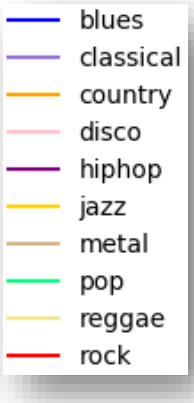
La duración de la canción usada es de 3 minutos y 42 segundos, por lo que se generan 7 fragmentos de 30 segundos cada uno. Al quedar 12 segundos restantes, es decir, menos de 20, esta información no se utiliza. En este experimento, se utiliza la función softmax para obtener los géneros, y se observa que los géneros obtenidos para cada segmento obtienen cada uno casi un 100% de probabilidad. Para esta canción, se observa claramente una estructura en tres partes, la primera detectada como Hip-hop, la segunda como Blues, y el final de la canción entre Clásica y Blues. En el quinto segmento (numero 4) se aprecia un pequeño segmento que corresponde con country. Esto indica que, si no utilizáramos la función softmax, en el método probabilístico los resultados para Clásica y Country serían muy similares.

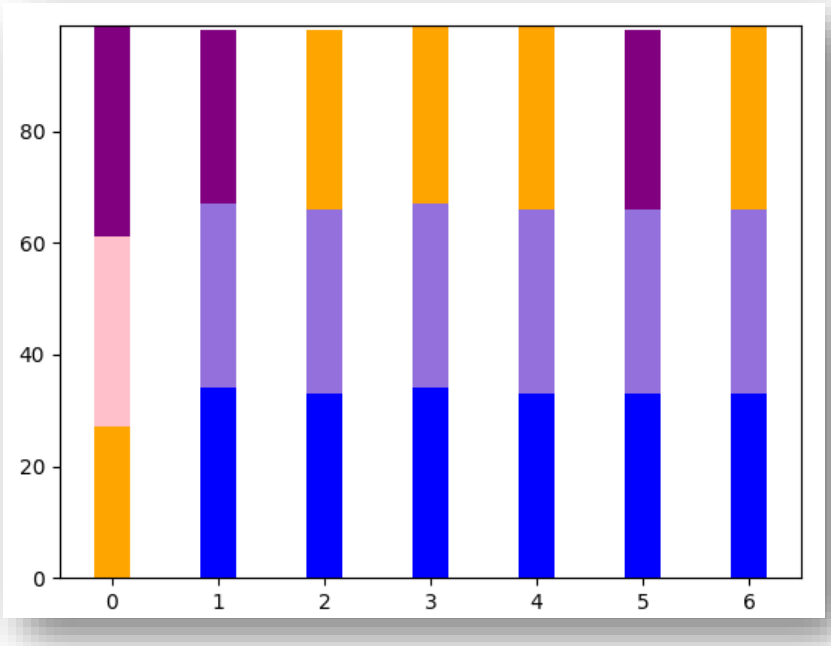
Si observamos los perfiles de ecualización para música Clásica y Blues, observamos que, aunque sean confundidos, realmente responden a unas necesidades de ecualización muy similares, así que, si el objetivo no es clasificar, si no ecualizar en base a la clasificación, este no es un mal resultado.



Resultados Probabilístico

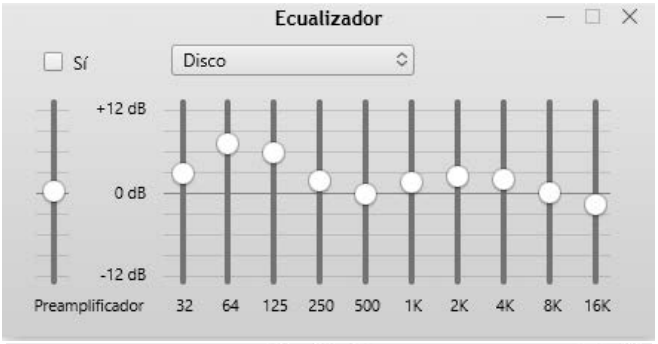


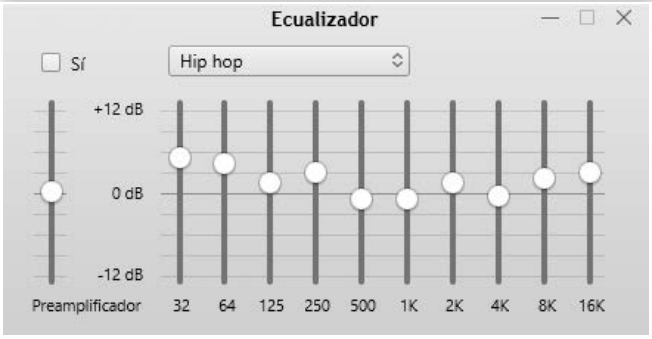




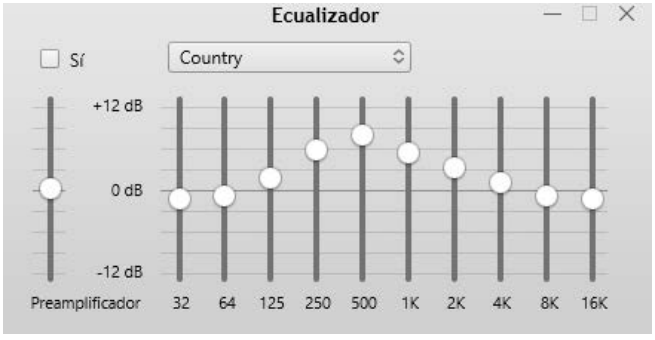
En este caso, se han tomado los tres géneros más probables para cada segmento. Parece confirmarse que la canción sigue la estructura descrita anteriormente de tres partes.

Para el primer segmento, aparece una gran confusión entre Hip-Hop y Disco, acumulando entre las dos el 75% de la probabilidad. Al igual que en el caso de Clásica con Blues, ambos géneros usan perfiles de ecualización relativamente parecidos, sobre todo en las bandas intermedias y agudas.



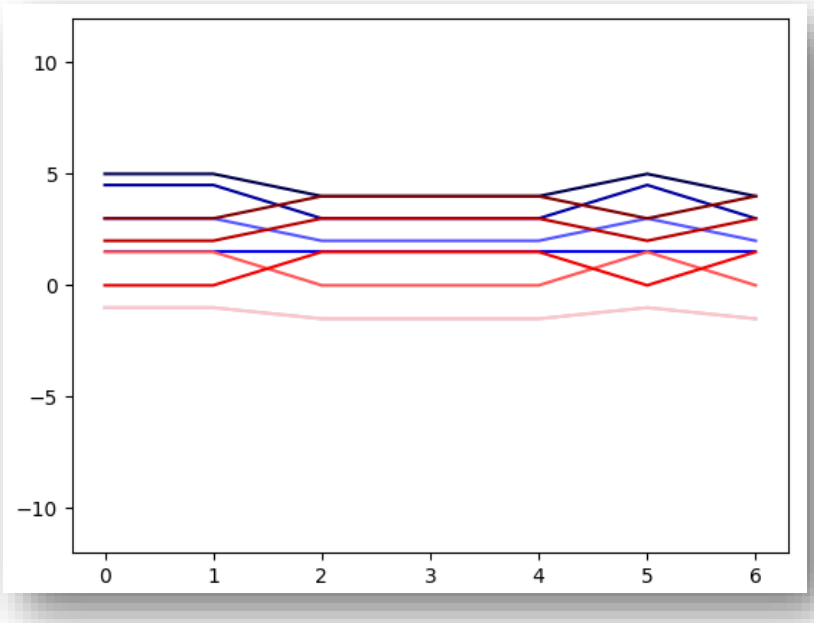


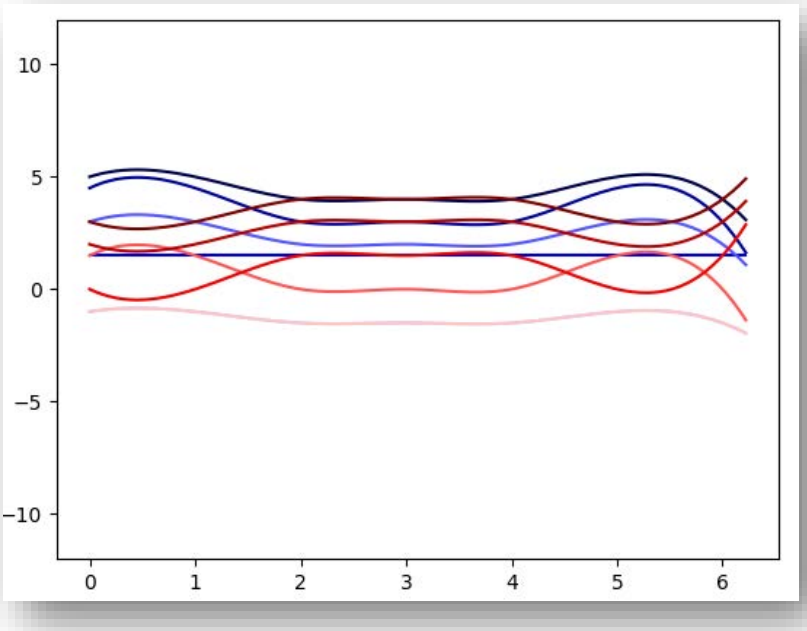
Para los segmentos intermedios y finales, la confusión importante se produce entre Blues y Clásica, igual que en el caso del Softmax. Incluso, si comparamos los perfiles de Blues, Clásica, Hip-hop y Disco, los cuatro siguen una estructura similar en forma de 'V' que acentúa graves y agudos, pero mantiene o atenúa las frecuencias intermedias. Curiosamente, también existe confusión con el Country, el cuál presenta un perfil completamente opuesto.

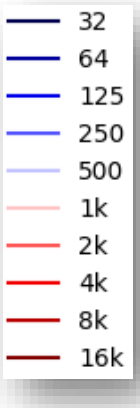


4.2. Interpolación vs no interpolación

Resultados softmax





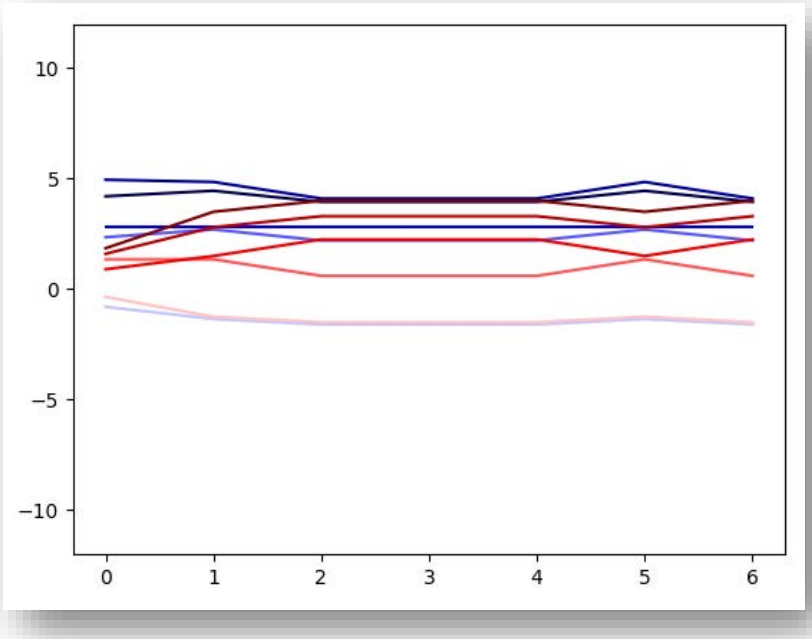


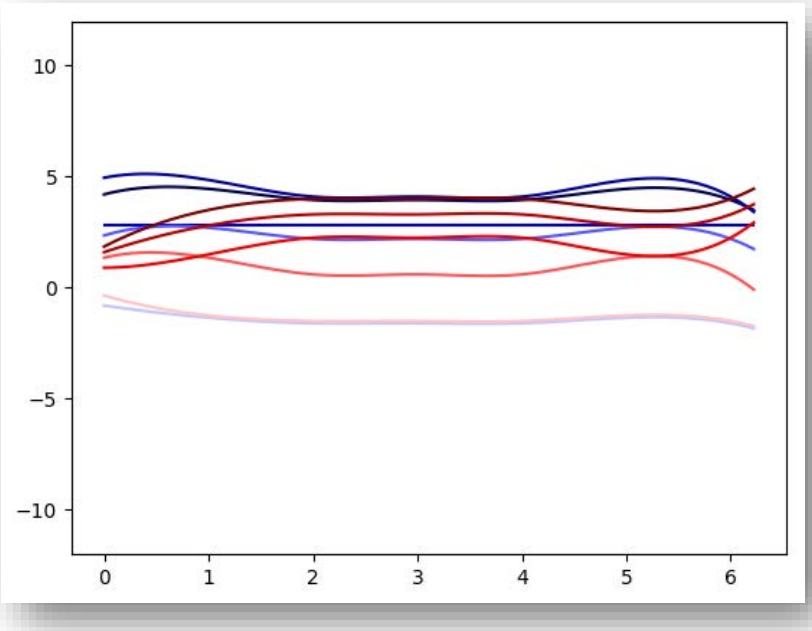
Estas gráficas representan la ganancia de cada banda de frecuencias (eje y) respecto a cada segmento de tiempo (eje x). Las bandas se representan con líneas de colores, granates y rojas cuando se trata de frecuencias altas/agudas, y azules cuando se trata de frecuencias bajas/graves.

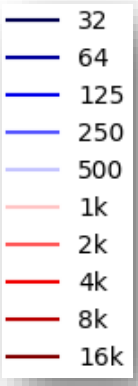
Con este experimento se confirma la sospecha de la estructura de la canción en tres partes. Comparando el sistema de interpolación, en algunos instantes de tiempo parece que se pierde precisión, por ejemplo, entre los segmentos 0 y 1 la frecuencia de 2k aparece por encima de la de 8k (mayor ganancia) en la segunda gráfica, pero aparece por debajo en la primera.

Aun así, considero que aplicar interpolación es beneficioso porque requiere poco tiempo de ejecución (Algo menos de 0,5 s) y aporta suavidad a las transiciones entre ecualizaciones entre segmentos.

Resultados probabilístico







Por último, los resultados de ecualización para el caso de la clasificación probabilística. Estas gráficas representan unos resultados muy similares a los obtenidos mediante softmax, pero algo más suaves y menos abruptas. Esto es la confirmación definitiva de que la ponderación de géneros no solo funciona, si no que además es beneficiosa, produciendo unos resultados mucho más limpios y efectivos.

5. Planificación y metodología

El Proyecto ha sido desarrollado como parte del convenio entre la Universidad Carlos III y las empresas BQ y MasMovil, las cuales ofrecieron una serie de cátedras remuneradas durante las que realizaríamos una investigación que además nos podría ser útil para utilizar tanto los conocimientos como el proyecto desarrollado como nuestro trabajo de fin de grado. Es por ello que la planificación tanto inicial como real se encuentra fuertemente marcada y definida por el progreso de la cátedra.

La duración total de la cátedra fue de 8 meses, comenzando el día 15 de noviembre de 2018, y finalizando el día 15 de Julio. A partir de esta fecha, se produjo un refinamiento y revisión del trabajo para completarlo y prepararlo para la entrega final.

5.1. Planificación inicial

La planificación inicial consideraba que todo el desarrollo se produciría y completaría dentro del periodo de 8 meses dado. Siendo así, se estimaba que los tiempos fueran los siguientes:

Tarea	Inicio	Fin	Días
Planificación	15-nov	29-nov	15
Estado del arte	30-nov	10-ene	42
Análisis de alternativas	11-ene	24-ene	14
Desarrollo	25-ene	16-may	112
Estudio de resultados	17-may	27-jun	42
Desarrollo del documento	28-jun	15-jul	18

Aunque con plazos extensos, esta planificación fue diseñada con el objetivo de adecuarse a los 8 meses de cátedra y a las pausas que esta tendría durante épocas de exámenes. La duración total estimada era de 243 días a la fecha de creación.

La fase de Desarrollo seguiría una estrategia iterativa e incremental, en la que se alternarían entre fases de diseño, implementación y pruebas cada aproximadamente 2 semanas. Esta decisión, se toma con el objetivo de facilitar corregir errores de

planteamiento que puedan surgir durante el desarrollo, y para permitir que el proyecto sea mucho más adaptable a los cambios que se propongan. Además, siendo una cátedra de investigación, el foco es la investigación, lo que propicia que se decida cambiar el planteamiento del proyecto una vez ya se ha iniciado.

Como producto de esta planificación inicial, se obtiene el siguiente diagrama de Gantt:

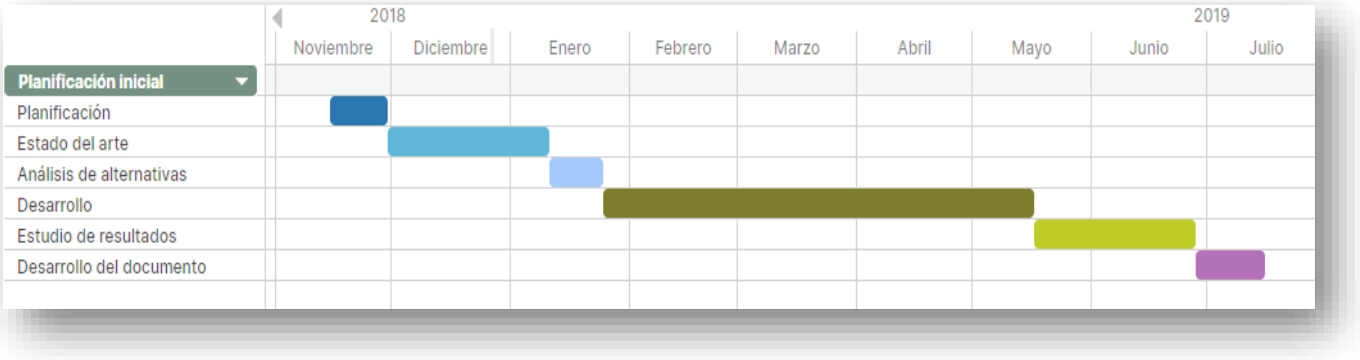


Fig 18. Diagrama de Gantt De la planificación inicial

5.2. Planificación real

Debido a los amplios plazos de la planificación inicial, en la planificación real no surge ningún retraso considerable. Para esta nueva planificación, simplemente se extiende ligeramente la fase de Desarrollo solapándose con el estudio de resultados, y aparecen cuatro nuevas etapas de revisión del estado del arte, del desarrollo, de los resultados, y del documento:

Tarea	Inicio	Fin	Días
Planificación	15-nov	29-nov	15
Estado del arte	30-nov	10-ene	42
Análisis de alternativas	11-ene	24-ene	14
Desarrollo	25-ene	30-may	126
Estudio de resultados	24-may	04-jul	42
Desarrollo del documento	05-jul	22-jul	18
Vacaciones	23-jul	18-ago	27

Revisión del estado del arte	19-ago	27-ago	9
Revisión del desarrollo	28-ago	03-sep	7
Revisión de los resultados	04-sep	07-sep	4
Revisión del documento	08-sep	20-sep	13

En esta tabla se reflejan tanto los retrasos en las etapas de desarrollo, estudio de resultados, y de desarrollo del documento, como el periodo de vacaciones, y las 4 revisiones posteriores. En total, suman 310 días desde el inicio hasta el fin del proyecto. A continuación, un nuevo diagrama de Gantt reflejando estos cambios y comparado con el diagrama inicial:

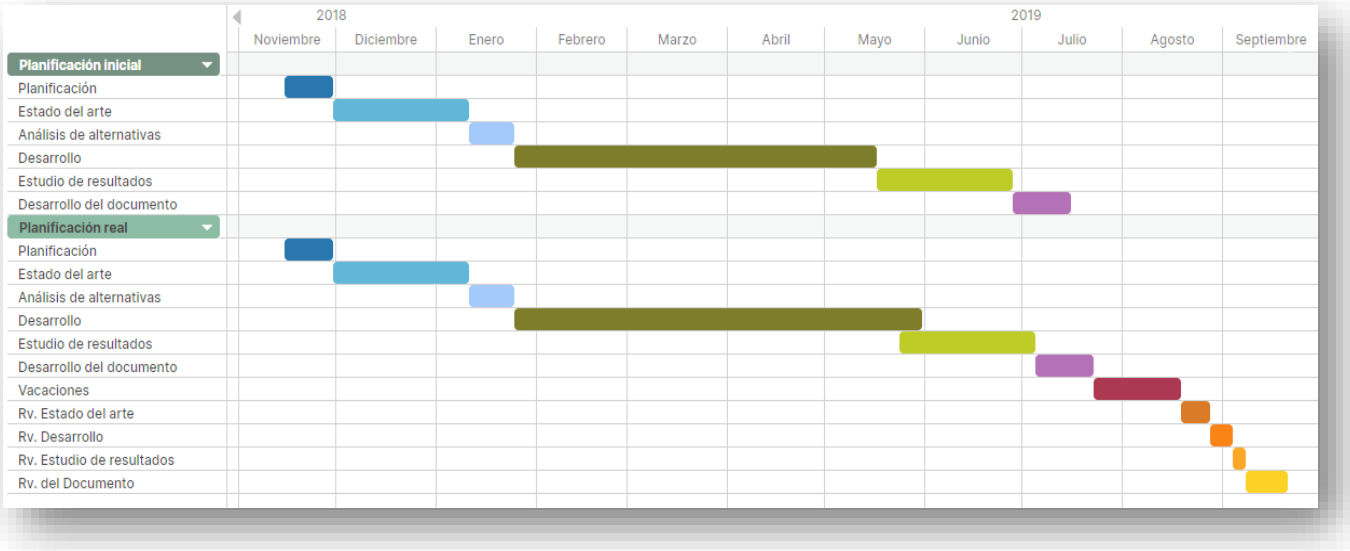


Fig 19. Diagrama de Gantt De la planificación Real frente a la inicial

5.3. Presupuesto

Para estimar el coste del proyecto, se considerarán 3 sectores: Costes humanos, costes hardware, y costes software. A continuación, presento el análisis de costes individual de cada uno:

Costes humanos

En este sector se estiman costes por el esfuerzo humano de las personas involucradas en el proyecto, en este caso, yo. Como ya se ha presentado en la planificación, la duración total del proyecto ha sido de 310 días. Excluyendo fines de semana y el periodo de vacaciones, resulta en 203 días de trabajo. Estimo una media de 2 horas de trabajo diarias, sumando un total de unas 406 h. El salario por horas correspondería a 15€ por hora de trabajo. Siendo así, el salario final sumaría un total de:

$$406\ horas\ \times 15\ \frac{€}{hora} =\ 6090\ €$$

Sobre este salario básico habría que añadir el importe de la seguridad social (23,60%) y el importe del IVA (21%). Este último se aplicará al obtener el coste total sobre todos los costes acumulados. Aplicando el coste de la seguridad social, los gastos en salario ascienden a **7527,24 €.**

Costes hardware

En este apartado se incluyen todos los costes de los dispositivos hardware que han sido requeridos durante la realización del proyecto. A los costes totales de cada dispositivo, se le aplicará una reducción a su precio original acorde a la proporción entre los meses esperados de utilidad del dispositivo, y los meses que ha sido utilizado para el proyecto. Los gastos en hardware son producidos por tres dispositivos: Un ordenador portátil Lenovo Thinkpad x270, un monitor secundario al del portátil LG 24MP58VQ-P, y un ratón Logitech G203 Prodigy. Los gastos son los siguientes:

Nombre del dispositivo	Meses de uso	Meses de duración	Precio inicial	Precio final
Lenovo Thinkpad x270	8	72	1.529,51€	169,95€
LG 24MP58VQ-P	8	36	200,95€	44,66€
Logitech G203 Prodigy	8	12	41,50€	27,67€
			Total:	**242,28€**

Costes software

En este apartado se estiman los gastos de todos los programas software que han sido necesarios para la realización del proyecto. Estos programas serán meramente enumerados, ya que todos han sido gratuitos o bien con licencia gratuita ofrecida por la universidad. Los programas son los siguientes:

Software	Tipo de licencia
Sistema operativo Ubuntu	Gratuita
Jetbrains PyCharm	Licencia universitaria
Microsoft Office: Word	Licencia universitaria
Microsoft Office: Excel	Licencia universitaria
GanttPRO	Gratuita
iTunes	Gratuita (Sin compra de canciones)

Costes finales

Sumando los tres puntos, y aplicando un 10% de aumento por riesgos, un 15% de beneficios, y un 21% para pagar el IVA. Con todo esto, queda un coste total y final de **11.892,41 €.**

Costes totales	7.527,24 € + 242,28 € = 7.769,52 €
Costes con riesgo (10%)	7.769,52 € + 776,95 € = 8.546,47 €
Costes con beneficios (15%)	8.546,47 € + 1.281,97 € = 9.828,44 €
Costes con IVA (21%)	**9.828,44 € + 2.063,97 € = 11.892,41 €**

6. Conclusiones y líneas futuras

6.1. Conclusiones

En este punto se discuten qué conclusiones se han obtenido al final de todo el proceso de estudio del trabajo.

- La decisión de separar la canción en fragmentos de longitud fija es la adecuada, ya que de otra manera se pierde información en algunos tramos sobre cuál es la mejor manera para ecualizar una canción.
- La confusión de géneros no es un problema, ya que esta no solo ocurre entre géneros similares en sonoridad, si no entre géneros que responden a unas necesidades de ecualización parecidas, es decir, con perfiles similares.
- El método probabilístico de géneros, frente al acercamiento del género más probable, parece ser más preciso.
- El método de interpolación de valores pierde información de la canción, pero suaviza las transiciones y es beneficioso para el resultado final de ecualización.
- El proceso de entrenamiento y de cálculos previos para canción individual es lento y costoso.
- La ecualización de sonido tiene potencial en el mercado, como muestra la tendencia de grandes empresas a empezar a implementar ecualizadores inteligentes en los nuevos dispositivos que salen al mercado.

6.2. Líneas futuras.

En este punto, se recopilan algunas de las líneas de investigación que se pueden seguir a partir de este trabajo buscando obtener unos resultados mejores o más novedosos en este o campos paralelos de estudio.

- **Optimización de velocidad:** Si se consigue que el sistema al completo, desde el Preprocesado de la canción hasta su ecualización sea más rápido, se podría

estudiar la posibilidad de implementar un sistema que funcione de esta manera en dispositivos móviles.

- **Cálculos en la nube:** Se puede estudiar la posibilidad de delegar algunos cálculos a la nube, disminuyendo la carga del dispositivo desde el que se solicita una ecualización, y posiblemente incluso optimizando la velocidad al poder contar con hardware más rápido.
- Feedback de los usuarios: Otra línea de investigación es solicitar feedback a los usuarios que escuchan una canción ecualizada con este sistema, con el objetivo de afinar aún más los parámetros de la red para contentar a más personas, e incluso personalizando la experiencia para cada uno.
- **Detección de elementos significativos:** En ocasiones, un instrumento concreto o incluso una nota individual necesitan ser más resaltados en una canción que el resto de los sonidos, independientemente del género. Otra línea de desarrollo sería enseñar a la red a detectar estas situaciones y a ecualizar utilizando esa información y no solo la distribución de géneros.
- **Otros parámetros de optimización:** Por último, se pueden estudiar qué otros parámetros pueden ser relevantes para obtener una ecualización, desde preferencias personales del usuario o cuál es el dispositivo en el que se reproduce, hasta aplicar detección de entornos realimentando la red con el sonido de la canción captado por el micrófono.

Bibliografía

[1] Real Decreto Legislativo 1/1996, de 12 de abril, por el que se aprueba el texto refundido de la Ley de Propiedad Intelectual, regularizando, aclarando y armonizando las disposiciones legales vigentes sobre la materia http://noticias.juridicas.com/base_datos/Admin/rdleg1-1996.html

[2] Quintanilla M. (2009). *Análisis armónico: el Teorema de Fourier*. Cpms-acusticamusical.blogspot.com. Available at: http://cpms-acusticamusical.blogspot.com/2009/10/analisis-armonico-el-teorema-de-fourier.html

[3] Bayle, Y. (2019). *ybayle/awesome-deep-learning-music*. GitHub. Available at: https://github.com/ybayle/awesome-deep-learning-music/blob/master/README.md

[4] G. Tzanetakis and P. Cook, "Musical genre classification of audio signals," in IEEE Transactions on Speech and Audio Processing, vol. 10, no. 5, pp. 293-302, July 2002. doi: 10.1109/TSA.2002.800560

[5] Thierry Bertin-Mahieux, Daniel P.W. Ellis, Brian Whitman, and Paul Lamere, The million song dataset, Proceedings of the 12th International Conference on Music Information Retrieval (ISMIR 2011), 2011

[6] Staff.aist.go.jp. (2012). *RWC Music Database (in English)*. Available at: https://staff.aist.go.jp/m.goto/RWC-MDB/

[7] Liuktus, A., Stöter, F., Rafii, Z., Kitamura, D., Rivet, B., Ito, N., Ono, N. and Fontecave, J. (2017). *The 2016 Signal Separation Evaluation Campaign*. Sigsep.github.io. Available at: https://sigsep.github.io/datasets/dsd100.html

[8] Gouyon, F. (2006). *Ballroom*. Mtg.upf.edu. Available at: http://mtg.upf.edu/ismir2004/contest/tempoContest/node5.html

[9] Duan, Z. and Pardo, B. (2011). Music.cs.northwestern.edu. Available at: http://music.cs.northwestern.edu/data/Bach10_Dataset_Description.pdf

[10] Zytrax.com. (2018). *Tech Stuff - Equalization (EQ), Metering and the FFT*. Available at: http://www.zytrax.com/tech/audio/equalization.html

[11] Apple (España). (2019). *iTunes*. Available at: https://www.apple.com/es/itunes/

[12] Magazine, L. (2019). *Registros de voz, tipos y clasificación - LaCarne Magazine*. LaCarne Magazine. Available at: https://lacarnemagazine.com/la-voz-registros-clasificacion/

[13] Zytrax.com. (2019). *Tech Stuff - Frequency Ranges*. Available at: http://www.zytrax.com/tech/audio/audio.html

[14] Sahin, E. *Simple Perceptron*. Cns-web.bu.edu. Available at: http://cns-web.bu.edu/~erol/html/msthesis/l2-node4.html#SECTION00121000000000000000

[15] Sahin, E. *Multi-layer Networks*. Cns-web.bu.edu. Available at: http://cns-web.bu.edu/~erol/html/msthesis/l2-node5.html#SECTION00122000000000000000

[16] Le, J. (2018). *The 10 Neural Network Architectures Machine Learning Researchers Need To Learn*. Medium. Available at: https://medium.com/cracking-the-data-science-interview/a-gentle-introduction-to-neural-networks-for-machine-learning-d5f3f8987786

[17] Ks, V. (2018). *Audio classification using Image classification techniques | Codementor*. Codementor.io. Available at: https://www.codementor.io/vishnu_ks/audio-classification-using-image-classification-techniques-hx63anbx1

[18] Gallagher, M. (2015). *7 Questions About Sample Rate*. 7 Questions About Sample Rate. Available at: https://www.sweetwater.com/insync/7-things-about-sample-rate/

[19] Kom.aau.dk. (2019). Available at: http://kom.aau.dk/group/04gr742/pdf/MFCC_worksheet.pdf

[20] Logan, B. (2000). *Mel Frequency Cepstral Coefficients for Music Modeling*. Semanticscholar.org. Available at: https://www.semanticscholar.org/paper/Mel-Frequency-Cepstral-Coefficients-for-Music-Logan/55afc2d63fd410a719c3bbe9772d9bbc6bc565a6

[21] Jiménez Sanfiz, A. and José Torra, F. (2018). *jsalbert/Music-Genre-Classification-with-Deep-Learning*. GitHub. Available at: https://github.com/jsalbert/Music-Genre-Classification-with-Deep-Learning

[22] Masood, S. (2014). *Genre classification of songs using neural network*. Available at: https://www.researchgate.net/publication/280565926_Genre_classification_of_songs_using_neural_network

[23] Dieleman, S. (2014). *Recommending music on Spotify with deep learning*. Available at: http://benanne.github.io/2014/08/05/spotify-cnns.html

[24] Lachmish, M. (2018). *mlachmish/MusicGenreClassification*. GitHub. Available at: https://github.com/mlachmish/MusicGenreClassification

[25] Desmos Graphing Calculator. (2019). *Desmos graph*. Available at: https://www.desmos.com/calculator/89ra57raqc

[26] Tjoa, S. (2018). *mfcc*. Musicinformationretrieval.com. Available at: https://musicinformationretrieval.com/mfcc.html

English state of the art and summary

Introduction

Audio processing is a subfield of digital signal processing, which studies the mathematical properties of a digital audio signal in order to extract Its properties/characteristics, analyze It, or apply modifications to an audio source.

Audio processing is one of the most extensive fields within signal processing, due to it involving a large amount of different audio sources such as music, speech, environmental sound, etc. Lately, the audio processing field has been widely developed given the recent rise in personal assistants such as Siri, Alexa or Cortana. Most mobile devices that enter the market today include a personal assistant capable of recognizing your voice and executing a series of instructions depending on what you say to the device, which has significantly increased the investment in time and resources that large companies such as Amazon, Apple and Google dedicate to the development and improvement of speech processing.

Despite this, other applications of audio processing like song processing have not experienced a similar improvement in both software and hardware technologies applied to it.

On the other hand, artificial intelligence is also being applied to more and more problems of our daily life. For example, some search engines use artificial intelligence techniques to offer better results to our queries. Also, the world of autonomous driving is becoming more important every day. Even social networks such as Instagram or Snapchat, use artificial intelligence to recognize our faces and apply filters on them. Inside the artificial intelligence field, neural networks are experimenting a considerable growth and are being implemented in a large amount of systems, like face recognition and speech processing. It is undeniable that we are approaching a world in which artificial intelligence will be present in every home (and even cities) we live in.

For now, most of the techniques applied to audio processing (more precisely, audio editing) are almost always non-deterministic solutions. By that I mean that we always get a subjective output depending on the person modifying the audio source. This can be a

problem, since the solution someone gets may not be the best (Or even a good solution) for another one.

The problem that I've decided to solve is trying to merge both audio processing and artificial intelligence (more exactly, neural networks) techniques and develop an interesting application to them. Song equalization is a very important factor to enjoy a piece of music, yet not as studied and developed as, for example, image processing. Nowadays, everyone expects their mobile phone to make their photos look nice and clean, but no one switches their equalization profile when listening to different music genres. The most sophisticated mobile phones even modify your photo's contrast, color and brightness depending on the scene, yet we must manually switch every frequency's volume in our phones to reach the best quality while listening to music.

Given this, I'm going to study the field of music equalization and try to offer a feasible solution based on neural networks, trying to solve any problem along the way.

Problems found

Just as discussed in the introduction, audio equalization (Moreover, using neural networks) is not even close to the amount of research in other fields like image processing. Given that, I'll try to compare my problem's domain to image enhancement's one, to look for problems and solutions that may work just as well or even better.

Determining a good equalization is not easy, owing to it not being an objective decision, but there is just the same problem in image enhancement. The solution to it, is using datasets composed of edited and original photos, and user ratings to them from different social networks like Instagram or Pinterest; If an edited image has a high rating, that means that it is well edited. The conclusion that we can take from this, is that we need to know what is commonly considered "well equalized" and what is not. Given that I have no access to a wide enough amount of people (like a social network) to get the information from, the reference that I'm going to take is a set of equalization presets, like those in a mobile phone.

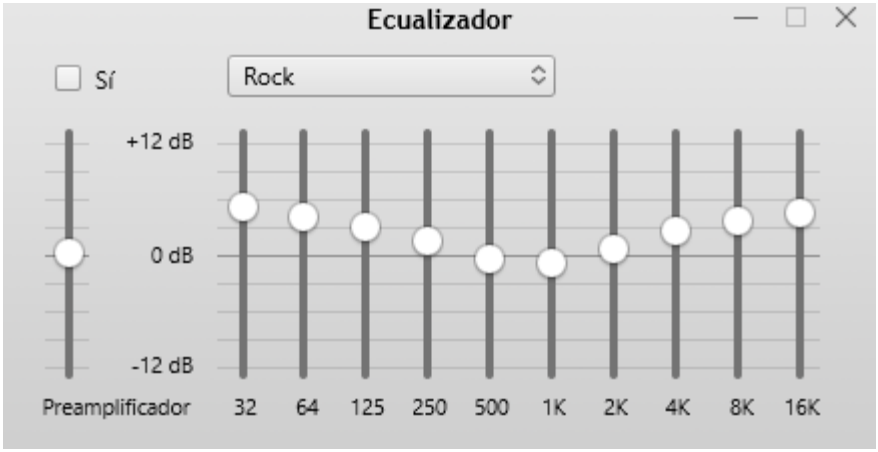


Fig 16. Equalization profile for Rock on iTunes

When using this solution, a new problem arises, what about a song that is not easily encased in a single genre? For image enhancement, there is not such a thing as “image genre” so we can’t find a similar problem in there. Given this, the solution I propose is mixing a certain number of equalization profiles depending on the probabilistic output from the neural network. For example, if the network determines that a song is 50% rock and 50% pop music, we can weigh each equalization profile to match the results from the classification and creating a “Custom” profile for the specific song.

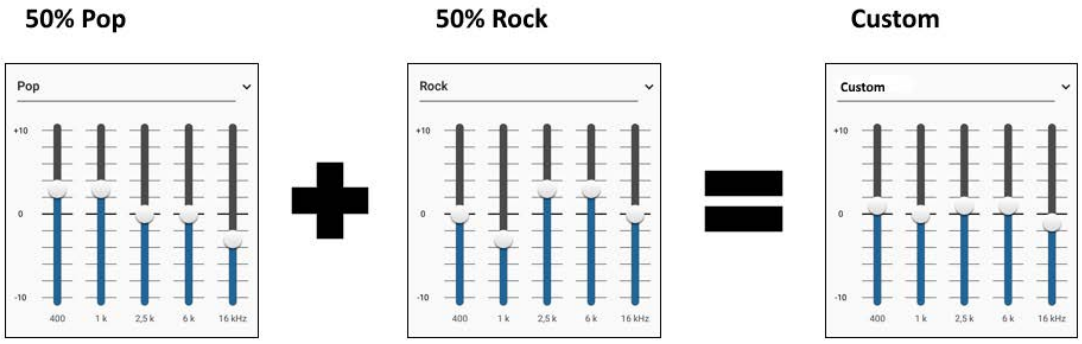


Fig 17. Sample mixing of 2 equalization profiles

Another important problem is the file size. Usually, song files are much heavier than image files and even the latter are reduced in order to make it easier to compute an output. For music, the average song has been increasing in length through the years, sitting at

around 230 seconds (3 minutes and 50 seconds) the year 2010. Also, MPEG-1 audio (The one used in .mp3 format) uses a 44,1KHz Sampling rate, that means 44.100 samples each second, adding up for more than 10 million samples for each song. Nonetheless there is a lot of important information in a song that can't be lost by reducing its size. The solution to this issue is feature extraction, more precisely using Mel Frequency Cepstral Coefficients (Or MFCC) of an audio file and using them as the inputs to the neural network. The MFCCs are proven to be a meaningful way of obtaining information from a song for similar classification problems like speech recognition or ambient sound classification.

At last, a song doesn't always sound the same throughout the whole duration of the file, but by obtaining the MFCCs that information is lost. One way we can solve this, is not obtaining the coefficients of the whole song, but obtaining them from a fixed length of time. Like so, not that much information is lost and we get a more detailed description of a song through time.

Related work

Despite intelligent music equalization is not a widely studied topic, there is a lot of research being done in the field of music genre classification, given It's an arduous task to manually sort songs by their genre.

Works like the one conducted by Masood S. try to eliminate the manual work of classifying audio files by using a Parallel Multi-Layered Perceptron architecture. The network outputs the most likely genre of the song instead of a probabilistic distribution of likeliness to belong to a set of genres, due to the goal of his research being just to classify in a single genre.

The inputs used for this network range from more abstract and mathematical ones like Discrete Wavelet Transform (DWT) and Mel Frequency Cepstral coefficients (MFCC) to more down to earth features, like Danceability, Loudness and Energy.

The results of his work are tested using a 2-genre 200-songs database, involving Classical and Sufi songs, obtaining an overall accuracy of around 85%.

Sarfaraz Masood claims that this same network can be used to classify other genres like Pop, Rock or Hip-Hop, and even songs that are a mixture of different genres, like Jazz-Rock, but the output of the network would still be 1 genre (Considering Jazz-Rock as a completely separate genre to Rock and Jazz)

Other studies like the one from Albert Jiménez Sanfiz and Ferran José Torra [21] simply use the MFCCs as the extracted features from a song, outputting an array of probability of a song belonging to a certain genre. Their study shows that some genres tend to be confused with other ones, but as discussed earlier, there is no need of getting a single genre from a song, but a set of them to equalize it proportionally. The genre confusion matrix is as follows:

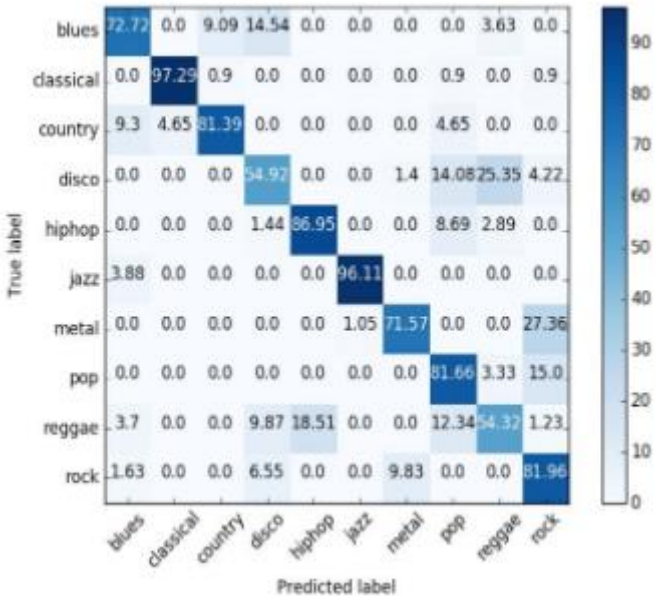


Fig 18. Genre confusion matrix from [21]

From this image, it seems like rock music shares a lot of similarities to metal music leading to some metal songs being labeled as rock. Other genres like classical or jazz are usually well labeled given to them sounding much different to other kind of songs, opposite to rock and metal songs.

Even though some genres have a really low accuracy rate (54.32% from reggae is really low for a classifying neural network), it seems like the confusion is shared between similar-sounding genres. From this I can conclude that the approach of mixing different equalization profiles to reach the best solution may work the best for the problem that is being tackled in this document.

Datasets

A dataset is a collection of data structured like a table where every column represents a variable, and every row is a different entry. A wide variety of datasets are used in deep learning for music. The following chart shows the most common ones:

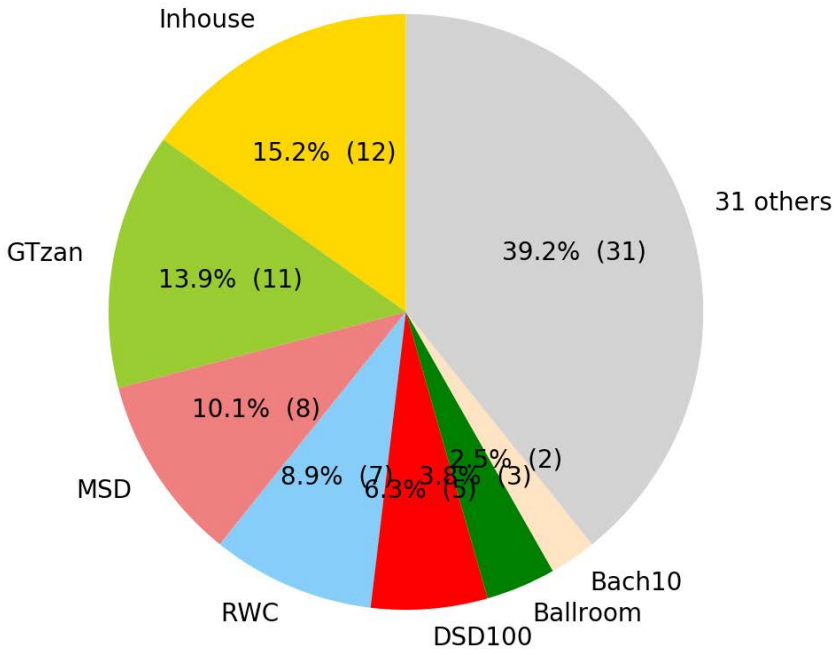


Fig 19. Dataset distribution for music related machine learning projects

As Inhouse refers to a dataset built by the programmer of the network himself, we are left with GTzan and MSD (Million song database) as the most used datasets.

GTZAN

GTzan is the dataset used by George Tzanetakis and Perry Cook in their work "Music genre classification of audio signals" [4]. It consists of 1000 songs from 10 different genres (Blues, Classical, Country, Disco, HipHop, Jazz, Metal, Pop, Reggae and Rock), 30 seconds each. All the audio files where collected from different sources like form the radio or from CDs making it so there is a variation in the song's recording conditions. This dataset is widely used from the success achieved in the study.

Million song dataset

The "million song dataset" (MSD [5]) contains just the features and metadata for one million songs, considerably reducing the disk size needed to hold that much information as music files. The dataset only contains the extracted features from the original audio files, but they can be downloaded if needed from different sources.

Final decisions

- **Fragmentation:** Once the network is already trained, in order to obtain the song's genre, I'll split it in fragments of equal length, and each of them will be analyzed and equalized individually, maintaining the maximum amount of information and being able to apply different equalizations for each different sounding segment.
- **MFCCs:** The input parameters that the network will be receiving are the Mel's Frequency Cepstral Coefficients. This decision is because they are fast to calculate, efficient, representative of human perception, and they represent information both from frequencies and from time.
- **Convolutional network**: The chosen network for this problem is a convolutional neural network. This kind of network works well for music related works, given that it holds relations between close frequencies and time instants, that are also represented in MFCCs.
- **Classification:** The problem will be solved by adding an intermediate classification step, that will allow the initial calculations to be finished early, so the equalization process will be made faster, maybe even executing in actual real time.

- **Equalization:** The equalization will be made by assigning a series of equalization profiles to each music genre, so the song (or the song fragments) will be equalized as their corresponding profile.
- **Genre mixing:** As there are some genre confusion cases, the system will be able to, instead of applying the equalization profile of the most likely genre, apply a mix of the n most likely genres distributed proportionally by their probabilities.
- **GTzan Dataset:** The chosen dataset for the network's training will be the GTzan dataset, given that it contains the whole song tracks and not only some of their precalculated parameters, allowing for some more experimentation with the data and the features extracted from it.
- **GTzan Genres:** The set of genres chosen is the one from GTzan dataset, not only because it's the one from the dataset that will be used in training, but because it's extended use in a big part of the music and machine learning studies, and for the good results shown in them.
- **Python:** The chosen programming language is Python, first because of its power for mathematical operations (Specially vectorial operations) and for its powerful machine learning libraries.
- **TensorFlow:** TensorFlow is chosen as the base for the Python's machine learning process for its wide variety of available resources on the internet, for its power, and for its ability to use other libraries such as Keras over it to simplify the networks modeling and constructing.
- **Keras:** Keras is chosen as the high-level library that works over TensorFlow because of its ease of use, and for putting at my disposal all the required elements to build a convolutional neural network designed for this problem.

Network description

As said earlier, the chosen networks architecture will be a convolutional neural network.

The structure of my network will consist of 2 hidden layers contrary to other works like [23] and [24]. The reason for this decision is that, as shown in the latter one, a third convolutional layer already starts distributing songs in genres.

Each of these two layers, will apply a max pooling and a dropout. Max pooling consists of applying a reduction to the number of parameters in the network, so each layer has less and less data to work with than the last one, concluding in a small classifying output. For example, as my networks takes 600 * 13 data values as input, I need to apply some max pooling to reduce it to the 10-genre output.

Dropout consists on breaking a certain proportion of connections between two given layers. This is made so the network not only learns to classified, also each layer learns to try and fix some of the errors that may have occurred on previous layers.

Lastly and just before the final output, I will be using a dense layer (Or fully connected layer) where there will be no max pooling.

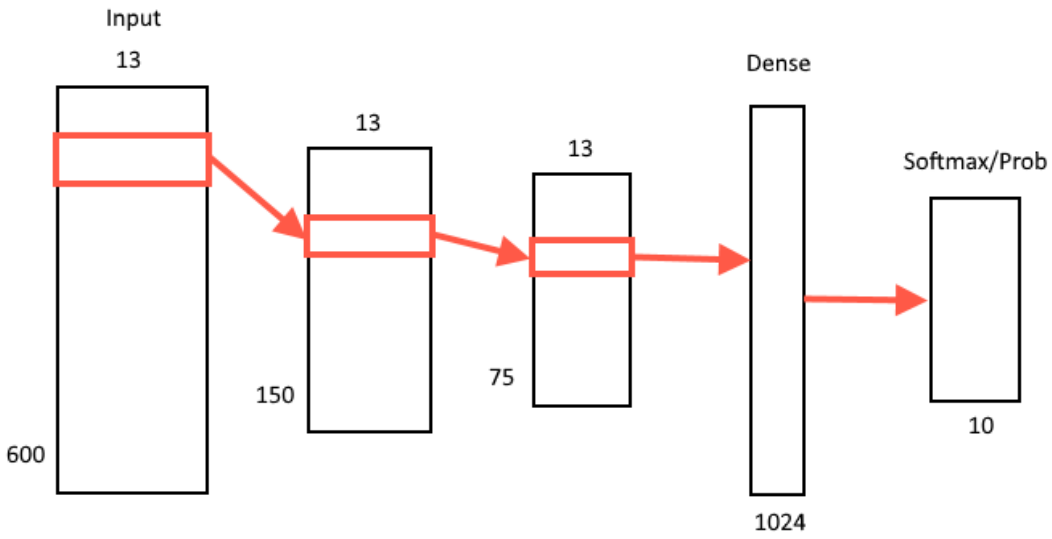


Fig 14. Chosen neural network scheme

Final results

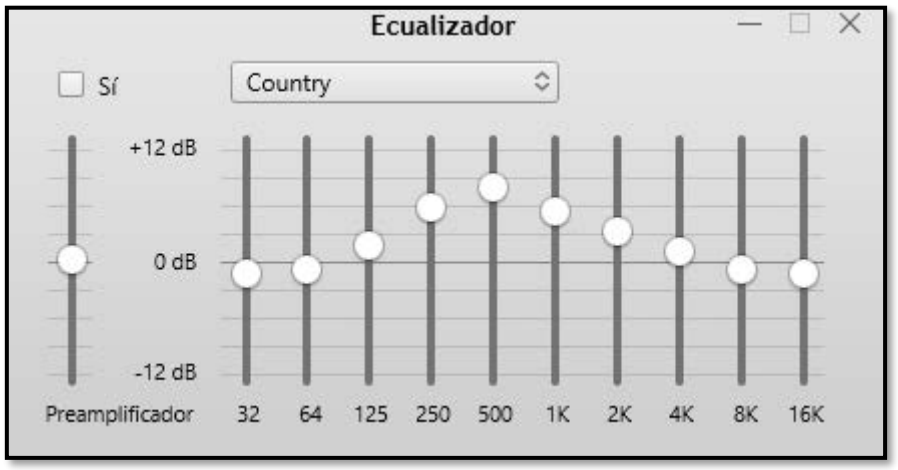
Once the network is trained, we can start to see some of the results. The final results are not completely satisfactory, because the network gets around a 60% accuracy on classifying songs. Some genres such as classical music seem to be really well split reaching even around 93.2% accuracy rate, but other genres such as Rock and Metal are easily mistaken and average around 40.31% accuracy.

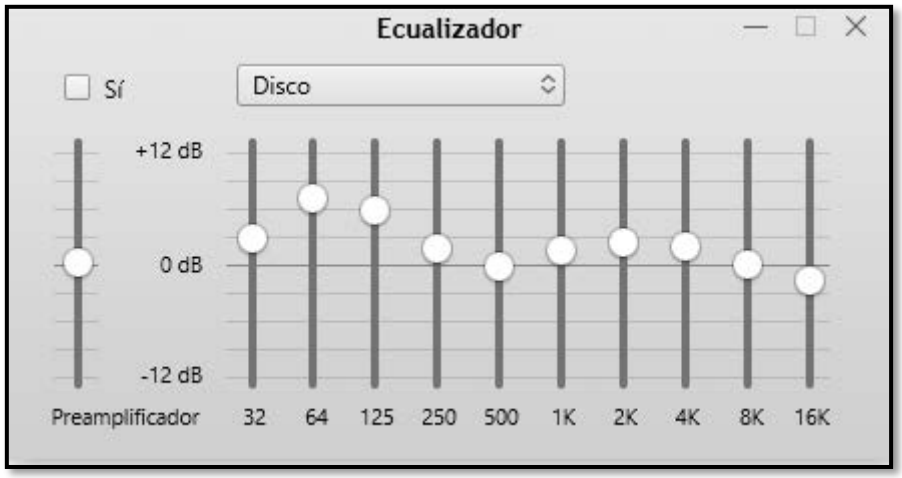
The used dataset to get the results from the network output, is the MSD. For the genres that are not shared between MSD and GTzan I used some songs from Spotify's "most popular songs" playlist.

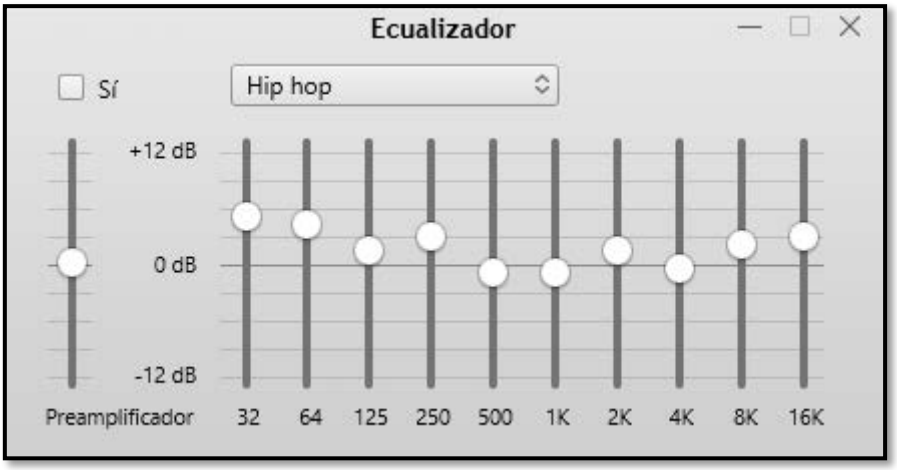
This results can be seen in chapter **ANEXO II: Tasas de aciertos para cada género en la red entrenada.**

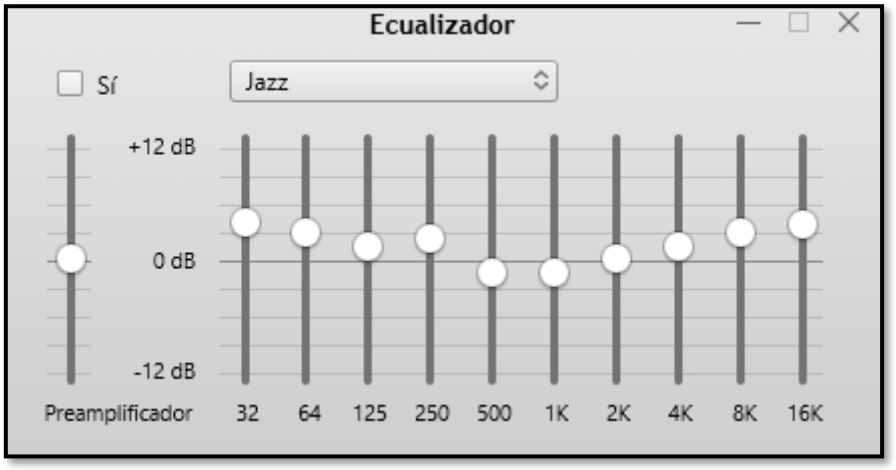
Anexos

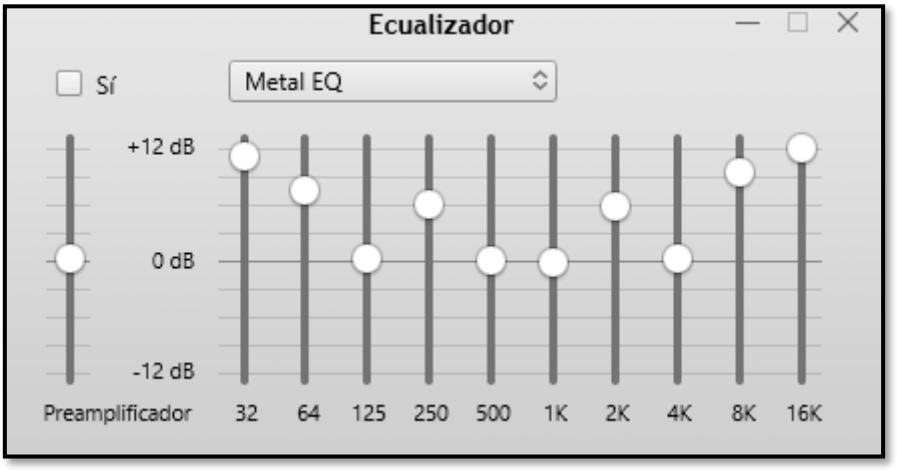
ANEXO I: Perfiles de ecualización

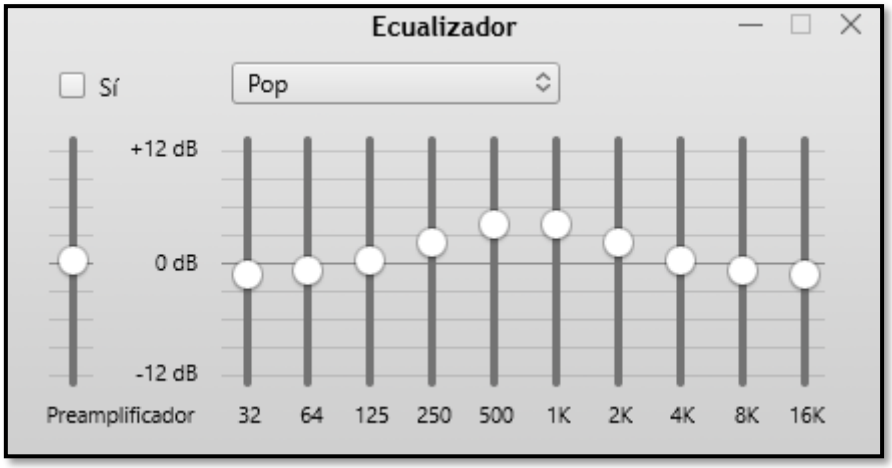


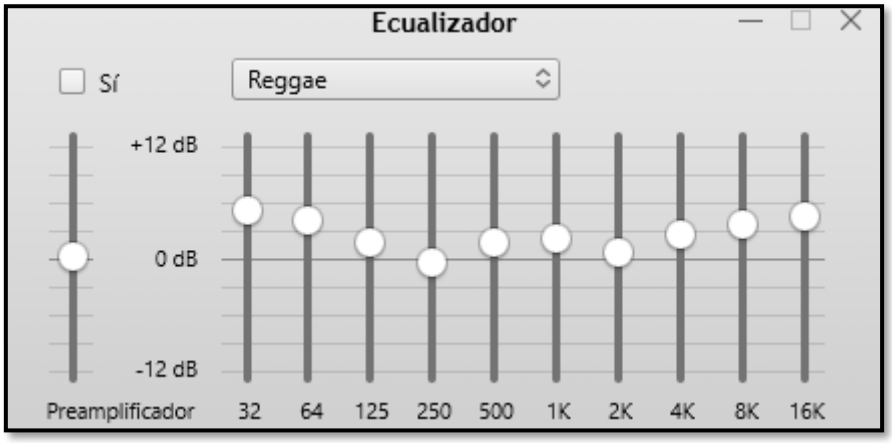


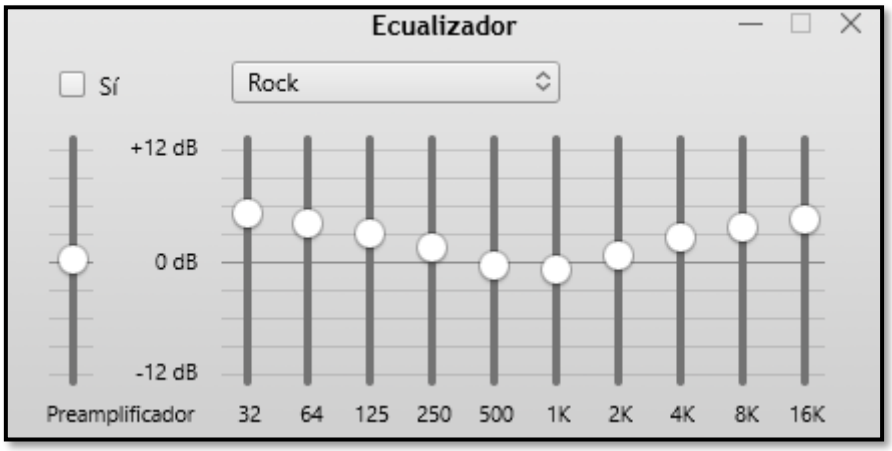

Ecualizador
Sí
Hip hop
+12 dB
0 dB
-12 dB
Preamplificador
32
64
125
250
500
1K
2K
4K
8K
16K

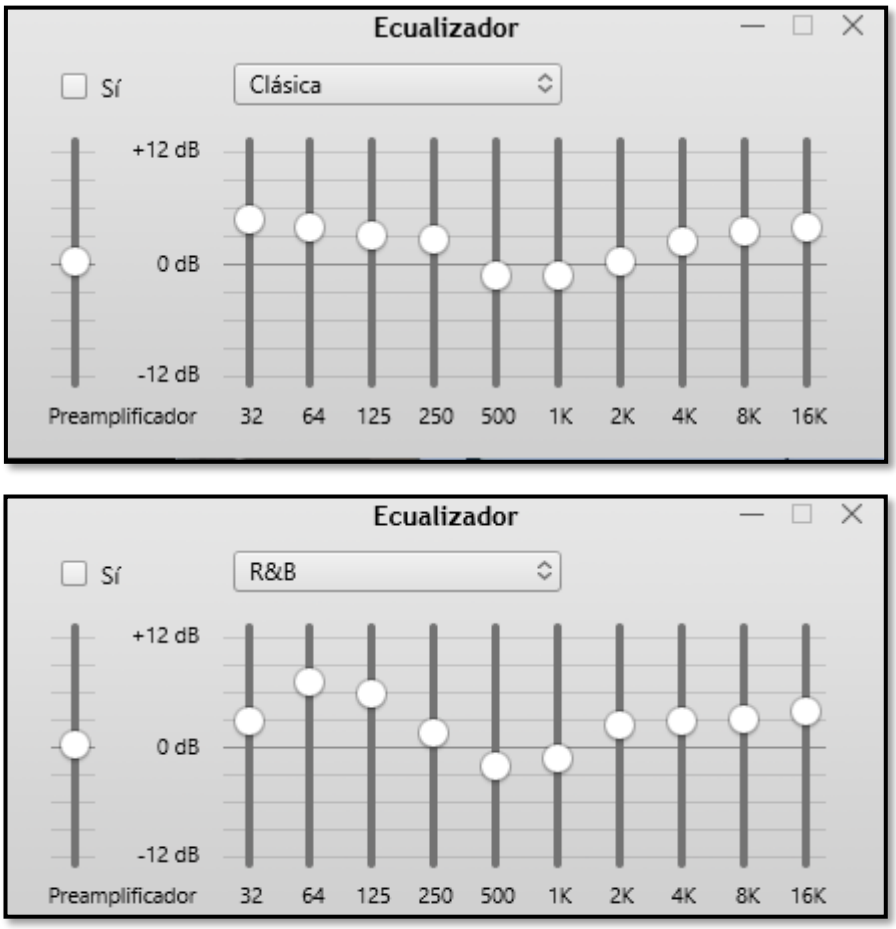

Ecualizador
Sí
Jazz
+12 dB
0 dB
-12 dB
Preamplificador
32
64
125
250
500
1K
2K
4K
8K
16K


Ecualizador
Sí
Metal EQ
+12 dB
0 dB
-12 dB
Preamplificador
32
64
125
250
500
1K
2K
4K
8K
16K


Ecualizador
Sí
Pop
+12 dB
0 dB
-12 dB
Preamplificador
32
64
125
250
500
1K
2K
4K
8K
16K


Ecualizador
Sí
Reggae
+12 dB
0 dB
-12 dB
Preamplificador
32
64
125
250
500
1K
2K
4K
8K
16K


Ecualizador
Sí
Rock
+12 dB
0 dB
-12 dB
Preamplificador
32
64
125
250
500
1K
2K
4K
8K
16K


Ecualizador
Sí
Clásica
+12 dB
0 dB
-12 dB
Preamplificador
32
64
125
250
500
1K
2K
4K
8K
16K

Ecualizador
Sí
R&B
+12 dB
0 dB
-12 dB
Preamplificador
32
64
125
250
500
1K
2K
4K
8K
16K

	32	64	125	250	500	1000	2000	4000	8000	16000
blues	3	7	6	1,5	-2	-1,5	3	3	3	4
classical	4,5	4	3	3	-1,5	-1,5	0	3	4	4
country	-1,5	-1	1,5	6	8	6	3	1	-1	-1,5
disco	3	7	6	1,5	0	1,5	2	2	0	-1,5
hiphop	5	4,5	1,5	3	-1	-1	1,5	0	2	3
jazz	4	3	1,5	2	-1,5	-1,5	0	1,5	3	4
metal	11	7,5	0	6	0	0	6	0	9	11
pop	-1,5	-1	0	2	4,5	4,5	2	0	-1	-1,5
reggae	5	4	1,5	0	1,5	2	1	3	4	4,5
rock	5	4	3	1,5	0	-1	1	3	4	4,5

ANEXO II: Tasas de aciertos para cada género en la red entrenada

	blues	classical	country	disco	hiphop
Tasa de acierto	54.46%	93.20%	53.30%	57.22%	50.16%

	jazz	metal	pop	reggae	rock	Promedio
Tasa de acierto	87.34%	34.90%	85.42%	73.88%	45.72%	61.67%