Proyecto Final de Aprendizaje de Máquina

Music Genre Classifier

Integrantes:

Marcos Tirador Del Riego C-411

Leandro Rodríquez Llosa C-411

Victor Manuel Amador Sosa C-412

Francisco Ayra Caceres C-412

Raúl Beltrán Gómez C-412

Niley González Ferrales C-411

Arian Pazo Valido C-311

Cuarto año. Ciencias de la Computación.

Facultad de Matemática y Computación, Universidad de La Habana, Cuba

Junio 2023

Resumen

La clasificación de géneros musicales juega un papel crucial en las aplicaciones modernas de procesamiento de señales de audio digital. En este estudio, proponemos varios enfoques de aprendizaje automático para categorizar con precisión pistas de música en géneros predefinidos. Cada enfoque utiliza diversos conjuntos de características que se pueden extraer de las canciones; desde los comunes: MFCC, señal de audio directa; hasta características poco exploradas en este problema: letra de la canción, Transformada de Wavelet. Para evaluar los modelos utilizamos el dataset de referencia en el campo, GTZAN. En los resultados obtenidos destaca que la letra de la canción no aporta mucho a la clasificación, al menos en el dataset utilizado. Los otros modelos muestran resultados consistentes con el estado del arte, con una precisión entre 75 % y 80 %

***Palabras clave* —** aprendizaje automático · clasificación · géneros musicales · MFCC · CNN · Conv1D · encoder · letra de canciones

I. Introducción

La clasificación automática de géneros musicales se ha vuelto cada vez más importante debido a la gran cantidad de música digitalizada disponible en la actualidad. El etiquetado de géneros preciso permite una organización, recuperación y recomendación eficientes de canciones, allanando el camino para experiencias personalizadas en servicios multimedia. La identificación de géneros es útil para crear listas de reproducción personalizadas, analizar las preferencias de los oyentes, recomendar artistas/canciones similares, detectar material con derechos de autor e identificar estados de ánimo asociados con ciertos géneros. Sin embargo, a pesar de la gran cantidad de soluciones y avances propuestos en el procesamiento de señales musicales, persisten varios problemas para lograr sistemas de reconocimiento de género robustos y eficientes.

En primer lugar, la definición de géneros musicales sigue siendo controvertida, ya que la percepción humana varía mucho. Estas interpretaciones subjetivas conducen a anotaciones inconsistentes y límites ambiguos entre géneros. Esta falta de consenso impide el desarrollo de modelos confiables capaces de generalizar a través de diferentes conjuntos de datos.

Aunque los géneros se derivan de influencias culturales, históricas, sociales, geográficas, tecnológicas y creativas, por lo general se reducen a etiquetas binarias. Sin embargo, muchos estilos comparten atributos y se superponen, lo que los hace difíciles de distinguir. Además, los artistas mezclan con frecuencia diferentes géneros, lo que agrega complejidad al proceso de clasificación.

En segundo lugar, la disponibilidad limitada de datos etiquetados de alta calidad plantea un gran desafío cuando se entrenan algoritmos de aprendizaje automático para la clasificación de géneros musicales. Debido al extenso tiempo que se requiere para etiquetar manualmente las pistas con géneros específicos, muchos investigadores recurren a conjuntos de datos pequeños o sintéticos, lo que lleva a modelos sobreajustados o inadecuados. Además, la obtención de colecciones más importantes a menudo implica derechos de licencia o barreras técnicas, lo que dificulta la accesibilidad y la reproducibilidad.

Por último, los enfoques actuales de aprendizaje profundo se basan en gran medida en procedimientos de preentrenamiento computacionalmente costosos, lo que los hace intensivos en recursos. Los trabajos existentes

suelen utilizar servicios en la nube o potentes GPU, lo que dificulta la implementación práctica sin hardware especializado o recursos financieros.

II. Análisis de features

Durante mucho tiempo, la clasificación de géneros musicales se basó principalmente en la información extraída de etiquetas que describían dísimiles características de los audios. Sin embargo, en los últimos años, los investigadores han explorado el uso de varios features para mejorar la precisión de la clasificación de géneros musicales. El surgimiento de nuevos modelos de aprendizaje automático también ha influido en la inclusión de features que no eran muy utilizados.

Por ejemplo, en lugar de usar características de audio pre-extraídas, se ha utilizado el audio puro como cracterística, con la intención de crear modelos más flexibles [2]. La ventaja de usar señales de audio sin procesar es que los modelos pueden aprender a reconocer patrones sin depender de características precalculadas que podrían no capturar toda la información relevante. Para ello la señal de audio se preprocesa llevandola a una frecuencia de muestreo fija de 22050 . Se recorta o rellena para tener una longitud fija de 660000 muestras, lo que corresponde a 15 segundos de audio. El uso de audio sin procesar como característica para la clasificación de géneros musicales es un enfoque simple y directo que no requiere pasos adicionales de extracción de características. Sin embargo, es importante tener en cuenta que puede requerir más recursos computacionales.

Otras técnicas de extracción de características para la música y las señales de audio se han basado en gran medida en la transformada de Fourier, como los espectogramas, en particular los MFCC. Los Mel Frequency Cepstral Coefficients (MFCC) han ganado gran popularidad en el campo de clasificación de audio. Constituyen una representación compacta del espectro de la señal de audio, lo que es particularmente útil cuando se trabaja con grandes cantidades de datos. Para calcular los MFCC, primero se segmenta la señal de audio en marcos cortos y se aplican una serie de operaciones matemáticas a cada marco, incluyendo preénfasis, enventanamiento, transformada de Fourier y compresión logarítmica, que juntas crean un conjunto de características que capturan muchos aspectos importantes de la señal de audio[13].

Los MFCC se pueden representar como una matriz donde cada fila corresponde a un marco particular de la señal de audio, y cada columna representa un MFCC particular.

Teniendo en cuenta la popularidad de los MFCC anteriormente mencionados, se puede genera un encoder.

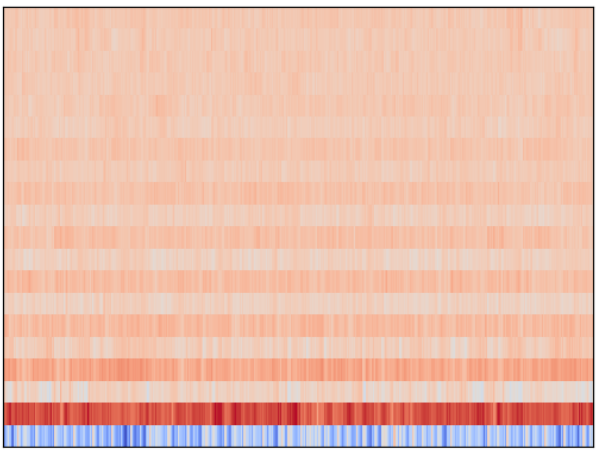

Figura 1: *Ejemplo de MFCC*

Esto asociaría a cada canción, a partir de su MFCC, una representación compacta de solo 500 dimensiones; intentando guardar la mayor cantidad de información posible de la misma. Sería muy interesante comparar el comportamiento de ambas versiones, los MFCC y el encoder reducido, para ver si se pierde información, o se reduce el ruido al hacer esta reducción de dimensiones.

De la música también se puede analizar la letra, . Recientemente surgió un modelo de Inteligencia Artificial de extracción de letras, Whisper [1], lo que permite utilizar un dataset que no tenga este feature directamente. La letra es una característica ligera y que intuitivamente puede aportar bastante información sobre el género de la canción. Los métodos actuales de clasificación utilizando la letra utilizan técnicas de embedding de palabras como GloVe, Word2Vec[5] y BERT [4]. Este último es un modelo basado en redes neuronales para el procesamiento de lenguaje natural, desarrollado por Google.

La clasificación de géneros musicales basada en la Transformada de Fourier, utilizando MFCC y espectrogramas, se ha explorado extensivamente en los últimos años. La Transformada de Fourier tiene una alta resolución en el dominio de la frecuencia, pero tiene una resolución cero en el dominio del tiempo. Esto significa que puede decirnos exactamente qué frecuencias están presentes en una señal, pero no en qué lugar en el tiempo se han producido [8]. Un mejor enfoque para analizar señales con un espectro de frecuencias dinámico es la Transformada Wavelet. Esta tiene una alta resolución tanto en el dominio de la frecuencia como en el del tiempo. Además, puede proporcionar una resolución de frecuencia variable, lo que significa que puede adaptarse a diferentes escalas de tiempo y frecuencia.

Las características mencionadas hacen de las wavelets una alternativa interesante en la clasificación de géneros musicales, donde ciertos géneros pueden tener patrones

rítmicos más rápidos o lentos que otros [11]. Por último, ya que utiliza pocos datos para representar una señal tiene gran potencial en el procesamiento de grandes cantidades de datos, como los datasets de música.

III. Metodología y experimentación

Al enfrentar el problema de clasificación de géneros fue realizada una investigación exhaustiva acerca de que modelos eran recomendables para cada una de las características. Entre las conclusiones obtenidas resalta el hecho de que la letra, por sí sola no supera el 70 % de precisión. Por ello fue decidido relizar una combinación innovadora entre el encoder de los MFCC y el embedding de la letra, para comprobar si la unión de ambas mejora los resultados respecto a la letra sola.

1. Encoder de MFCC y embedding de la letra

La idea de combinar el análisis de la letra y el encoder es concatenar los vectores resultantes de estos procesamientos y clasificar en base a esta combinación de features. Este modelo fue llamado VisionLang.

La arquitectura del encoder fue diseñada utilizando ideas, principalmente de *An Introduction to Autoencoders* [14] y *Autoencoders* [15].

Para construir el encoder fue programado un autoencoder y se tomó el modelo hasta el bottleneck. La arquitectura del autoencoder combina capas maxpooling y upsampling al inicio y al final respectivamente para moderar el tamaño la imagen, intercaladas con capas convolucionales y en el medio, un par de capas densas para aprovechar que ya el número de dimensiones era relativamente pequeño y realizar un poco más de aprendizaje.

La arquitectura en detalle es la siguente:

Como se puede observar la arquitectura es casi simétrica. Fue tomada como función de pérdida y métrica el error cuadrático medio (*mean squared error*). La entrada del autoencoder y la salida esperada fueron las imágenes del feature MFCC del conjunto de entrenamiento luego de haber sido normalizadas, es decir, que en vez de estar en el rango $[0, 255]$ cada valor de la imagen de entrada, los representamos en el rango $[0, 1]$.

Se probaron otras arquitecturas, desde algunas que no tenían capas densas hasta otras que principalmente consistían en capas densas. El problema en las arquitecturas carentes de capas densas era que sus resultados no eran lo suficientemente buenos, es decir, debido a su relativamente baja cantidad de parámetros el nivel de aprendizaje que podían lograr era inferior al que se logró luego con arquitecturas con capas densas. Por otro lado, las arquitecturas que consistían principalmente en capas densas tenían problemas como que los modelos eran muy grandes, algunos pasando de los GiB de almacenamiento y presentaban un problema para nosotros a la hora del entrenamiento. Otro problema que tienen las arquitecturas más basadas en capas densas es su tendencia al overfitting. En las prueba realizadas, las arquitecturas carentes de capas densas no presentaban este tipo de problema ya que los resultados en los conjuntos de entrenamiento, test y validación tenían poca diferencia entre ellos, sin embargo en las arquitecturas que tenían capas densas, por el gran número de parámetros si se evidencia una diferencia sustancial entre los resultados en los conjuntos de entrenamiento, y los obtenidos en los de prueba y validación.

Nombre de la capa	**Tipo de la capa**	**Shape de salida**
$input_l$	Input	(192, 256, 3)
$maxp_{ini}$	MaxPooling2D	(96, 128, 3)
$encoding_1$	Conv2D	(96, 128, 12)
$maxp_1$	MaxPooling2D	(48, 64, 12)
$encoding_2$	Conv2D	(48, 64, 6)
$maxp_2$	MaxPooling2D	(24, 32, 6)
$encoding_3$	Conv2D	(24, 32, 3)
$flat_1$	Flatten	(2304)
$bottleneck$	Dense	(500)
$decoding_1$	Dense	(2304)
$resh_1$	Reshape	(24, 32, 3)
$decoding_2$	Conv2D	(24, 32, 6)
Up_1	UpSampling2D	(48, 64, 6)
$decoding_3$	Conv2D	(48, 64, 12)
Up_2	UpSampling2D	(96, 128, 12)
$decoding_4$	Conv2D	(96, 128, 3)
$output_l$	UpSampling2D	(192, 256, 3)

Cuadro 1: *Arquitectura del autoencoder.*

Por cada audio del dataset se extrae la letra haciendo uso del modelo, Whisper[1]. Luego, se procesa la letra para obtener el embedding utilizando BERT [4].

Una vez obtenidos los dos embedding correspondientes a la letra de la canción y la música (MFCC), se conforma la capa de entrada de la red neuronal de VisionLang, o sea, la concatenación de los dos vectores obtenidos. Luego la red tiene dos capas ocultas con función de activación RELU y cantidad de neuronas 128 y 64 respectivamente. Finalmente la capa de salida consiste de 10 neuronas que representan a cada uno de los géneros musicales que analizamos, y cuenta con softmax como función de activación. Siendo coherente con esto último usamos como función de pérdida la Categorical Cross Entropy. Además, el modelo fue entrenado durante 500 epochs.

II. CNN-MFCC

Las redes neuronales convolucionales (CNN) se utilizan ambpliamente en el campo para entrenar con MFCC, debido a que estas redes están diseñadas específicamente para detectar patrones y características en imágenes, lo que las hace muy útiles en problemas de clasificación y reconocimiento de objetos. Esto significa que no se necesita un conocimiento experto para diseñar los filtros o características que se utilizan para procesar las imágenes, ya que la red es capaz de aprenderlos a partir de los datos de entrenamiento.

La arquitectura de una CNN se compone de varias capas de procesamiento, incluyendo capas de convolución y de pooling que permiten extraer características relevantes de las imágenes de entrada. La capa de convolución es la que aplica un filtro o kernel a la imagen de entrada para detectar patrones específicos, como bordes, líneas o texturas. La capa de pooling reduce la resolución de la imagen de salida, lo que ayuda a reducir el número de parámetros que deben entrenarse y a evitar el sobreajuste. También se usan capas independientemente de la necesidad del modelo, como capas para aplanar las imágenes a un vector, capas densas para pasarle lo que se detecto en las anteriores y ajustar pesos, y capas de activación.

Arquitectura de capas del modelo

- Capa de entrada input: recibe la la imagen con tamaño 256x192 y 3 filtros RGB.(256,192,3).
- Capa Conv2D: recibe la capa input anterior y devuelve datos de tamaño (256,192,64), aplica 64 filtros a la imagen.
- Capa AveragePooling2D: recibe datos de tamaño (256,192,64) y devuelve la imagen reducida en (2,2), devuelve datos de tamaño (128,96,64).
- Capa Conv2D: recibe datos de tamaño (128,96,64) y devuelve datos de tamaño (128,96,128), aplica 128 filtros a la imagen.
- Capa AveragePooling2D recibe datos de tamaño (128,96,128) y devuelve la imagen reducida en (2,2), devuelve datos de tamaño (64,48,128).
- Capa Conv2D: recibe datos de tamaño (64,48,128) y devuelve datos de tamaño (64,48,256), aplicando 256 filtros a la imagen.
- Capa GlobalAveragePooling2D: recibe datos de tamaño (128,96,128) y devuelve un vector 1D [12] de tamaño 256, con una salida por cada filtro.
- Capa Dense de 256 neuronas: recibe los datos de la salida de la capa anterior y los procesa con la función de activación RELU.
- Capa Dense de 128 neuronas: recibe los datos de la salida de la capa anterior y los procesa con la función de activación RELU.
- Capa Dense de 64 neuronas: recibe los datos de la salida de la capa anterior y los procesa con la función de activación RELU.
- Capa Dense de 32 neuronas: recibe los datos de la salida de la capa anterior y los procesa con la función de activación RELU.
- Capa Dense de 10 neuronas: recibe los datos de la salida de la capa anterior y los procesa con la función de activación softmax para determinar la salida de tipo clasificación.

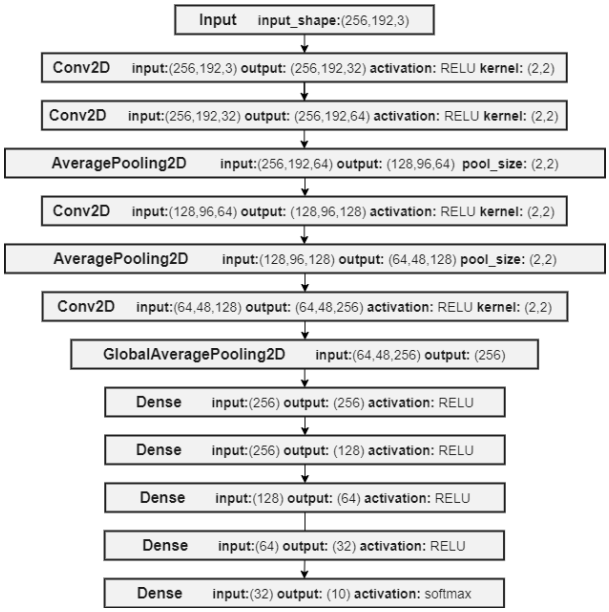

Figura 2: *CNN Model*

III. Audio puro con CNN 1D

Uno de los acercamientos más interesantes a la clasificación utilizando el audio puro es utilizar redes convolucionales unidimensionales (CNN 1D). La arquitectura de ResNet [3] es una popular red neuronal convolucional utiliza bloques residuales que permiten que la información fluya a través de la red sin perder información en el camino. Estos bloques residuales están compuestos por varias capas convolucionales y de normalización, y se utilizan para crear una red neuronal profunda con un número de capas mucho mayor que el de las redes neuronales convencionales

Cabe mencionar que, aunque el modelo está inspirado en la arquitectura ResNet, la implementación en el código no incluye las conexiones de atajo (shortcut connections) que caracterizan a ResNet. Estas conexiones permiten que las señales pasen directamente a través de la red, ayudando a combatir el problema del desvanecimiento del

gradiente durante el entrenamiento de redes profundas. A pesar de la ausencia de estas conexiones, el modelo aún puede ser efectivo para la tarea de clasificación de géneros de música. Los trabajos futuros podrían considerar la implementación de las conexiones de atajo para recrear más fielmente la arquitectura ResNet y potencialmente mejorar el rendimiento.

La idea principal de dicho modelo es tomar una señal de audio, pasarla a través de una serie de capas convolucionales para extraer características útiles, y luego usar esas características para clasificar la señal en uno de los géneros musicales.

La red neuronal incluye capas convolucionales, una capa de agrupación (pooling), y una capa densa (fully connected). El modelo se entrena utilizando una función de pérdida de entropía cruzada categórica y el optimizador Adam.

IV. Transformada Wavelet

La clasificación de audio utilizando la Transformada Wavelet ha ganado atención en los últimos años debido a su eficacia en la captura de características locales de las señales en varias escalas. Sin embargo, la investigaciones aún son escasas en comparación con otras técnicas de extracción de características. Esto puede adjudicarse a la complejidad involucrada en la selección de wavelets apropiadas y la dificultad que se le atribuye al trabajo con ellas[8].

V. Wavelet Discretas y Complejas

Existen tres tipos principales de wavelets: discretas, continuas y complejas. Cada tipo ofrece numerosas opciones para elegir en función de requisitos específicos, por lo que es esencial realizar investigaciones y experimentos exhaustivos antes de aplicarlos. Debido a que las wavelets continuas se suelen analizar convertidas a imágenes, y a que sus resultados no son especialmente superiores a MFCC [7], la investigación fue enfocada en wavelets discretas y complejas.

La Transformada Wavelet Discreta (DWT) [8] es un caso especial de Transformada Wavelet que proporciona una representación compacta de la señal en el tiempo y la frecuencia, que se puede calcular de manera eficiente[11]. Por otro lado la Transformada Wavelet Compleja de Doble árbol (DT-CWT) [9] es una mejora relativamente reciente a DWT; ya que para señales moduladas complejas como el audio, DWT encuentra algunas pocas deficiencias: oscilaciones, varianza de desplazamiento, aliasing y falta de direccionalidad [7].

Respecto a la Transformada Wavelet Discreta, Daubechies wavelet empíricamente muestra el mejor comportamiento en muchas aplicaciones[7]. En las pruebas con distintos órdenes de Daubechies wavelet, db12 fue la que mejor se desempeñó. Por otro lado la Transformada Wavelet Compleja de Doble árbol, mostró los mejores resultados con 17 niveles de descomposición.

Teniendo en cuenta las recomendaciones de [8], varios de los modelos de machine learning tradicional son muy prometedores en aprendizaje con wavelets. Para la experimentación preliminar fue analizado el comportamiento de dichos modelos en GTZAN (como se puede ver en la tabla), destacándose Random Forest Classifier y Gradient Boosting Classifier.

Modelo Feature	DWT	DT-CWT
Random Forest	0.7487	0.7527
Gradient Boosting	0.7497	0.7467
Logistic Regression	0.6857	0.7297
Linear SVC	.6637	0.7227
SVC	0.6486	0.6847

Cuadro 2: *Precisión promedio en 5-Fold*

IV. Resultados

El dataset fue dividido en 80 % para entrenar, 10 % para validar y 10 % para pruebas, manteniendo el balance de cantidad de elementos por cada clase, para todos los conjuntos.

El modelo de CNN utilizando MFCC, tras 300 épocas de entrenamiento, mostró una precisión de 0.7879 en los datos de prueba, 0.7699 en los de validación y 0.9786 en los de entrenamiento.

Se muestran en los siguientes gráficos los resultados del entrenamiento en el transcurso de las épocas. Se puede apreciar que cuando los datos de entrenamiento pierden el sobreajuste, los datos de validación por lo general mejoran el resultado. En esos picos se nota cómo el modelo generaliza mejor.

También se reporta una matriz de confusión para comprobar el comportamiento en cada género. Se observa que no se desempeña de la misma forma en todos los géneros, ya que hay algunos (como el country) donde se observan mejores resultados que en otros (reggae).

En la siguiente tabla se muestran los resultados de test por cada género:

Al entrenar el autoencoder se realizó un *save* cada 10 epochs. En la siguiente imagen se observan los resultados sobre el número del *save* en el modelo de autoencoder presentado:

A partir del análisis de la gráfica anterior se tomó como cantidad de epochs a ejecutar la cantidad de 150, ya que se conjeturó (y luego validó) que el comportamiento

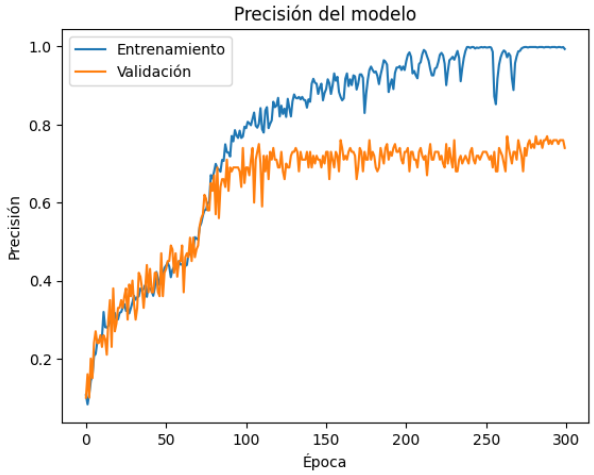

Figura 3: *CNN Accuracy*

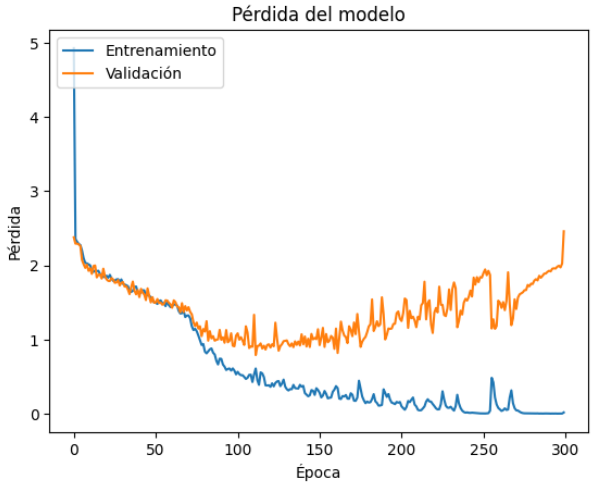

Figura 4: *CNN Loss*

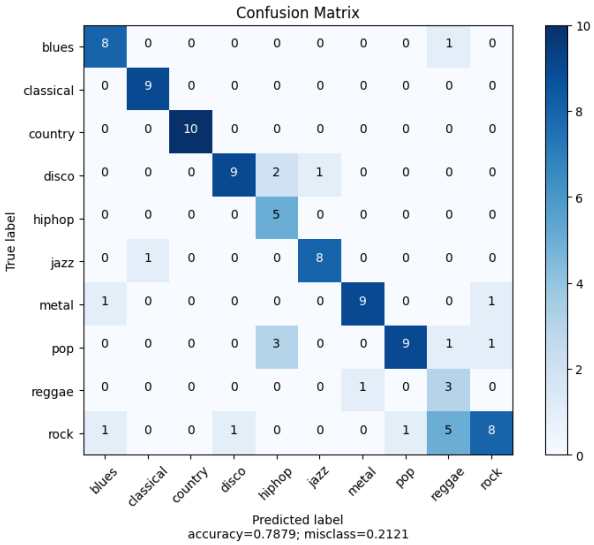

Figura 5: *Matriz de confusión*

Genre	Accuracy
blues	0.551
classic	0.851
country	1.000
disco	0.852
hip hop	0.749
jazz	0.827
metal	0.949
pop	0.651
reggae	0.759
rock	0.551
average	**0.787**

Cuadro 3: *Precisión de clasificación por cada género de GTZAN.*

del modelo en el conjunto de prueba sería muy similar al comportamiento en el conjunto de validación y en este punto es que se obtiene un mejor performance en el conjunto de prueba.

Volviendo atrás, los resultados obtenidos para cada tipo de modelo:

- los modelos que no tenían capas densas lograron primeramente un error (MSE) de 0,0025 y luego de ampliar la cantidad de dimensiones que salen del bottleneck, es decir la cantidad de dimensiones de la representación se logró 0,0021. Estos modelos tenían muy poco overfitting luego de 500 epochs.
- los modelos que presentan capas densas, por su parte, comenzaron con resultados en el conjunto de entrenamiento de hasta 0,0011 con 500 epochs, lo cual era muy bueno pero podía implicar overfitting. A medida que se redujeron la cantidad de parámetros (de 288 millones a los 2,3 millones del modelo propuesto) los resultados en el conjunto de entrenamiento fueron peores pero nunca sobrepasaron el valor de 0,0013 en 500 epochs. Luego de correr el modelo propuesto solo 150 epochs se obtuvieron los mejores resultados tanto en el conjunto de prueba como en el validación, oscilando alrededor de 0,00185, por su lado en el conjunto de entrenamiento se obtuvieron resultados alrededor de 0,0016, lo que evidencia la presencia de overfitting.

Se realizó también cross validation con Kfold dividiendo todo el conjunto de entrenamiento en 10 subconjuntos. Los resultados del cross validation coincidieron con los resultados anteriormente descritos. Este test se hizo luego de haber fijado la cantidad de epochs en 150.

VisionLang, que fue entrenado durante 500 epochs, presenta una precisión, para el conjunto de entrenamiento,

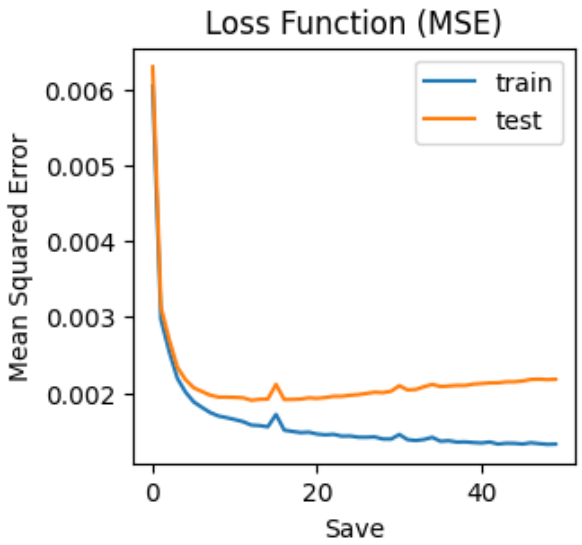

Figura 6: *Gráfica de error del modelo contra número de épocas del entrenamiento.*

casi perfecta llegados al epoch número 300. En el caso del conjunto de datos de validación, que es el que nos da la efectividad de nuestro modelo, vemos que el accuracy crece rápidamente hasta llegar al valor 0,57 alrededor del epoch 150.

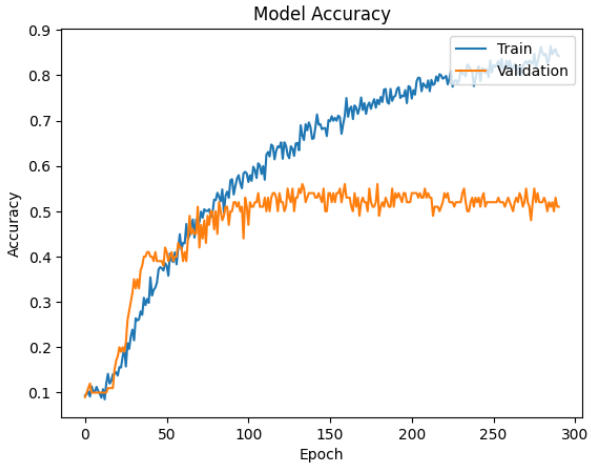

Figura 7: *Gráfico de precisión*

Con más capas y neuronas el modelo hacía rápidamente overfitting, o sea en pocos epochs. Esto nos llevó a reducirlo a la arquitectura actual. Como podemos observar en la próxima gráfica la función de pérdida para los datos entrenantes decrece rápidamente acercándose mucho a cero en el epoch 300. En el caso de los datos de validación, vemos como a partir del epoch 100 la red comienza a hacer overfitting, pero no es representativo.

Presentamos aquí además la matriz de confusión.

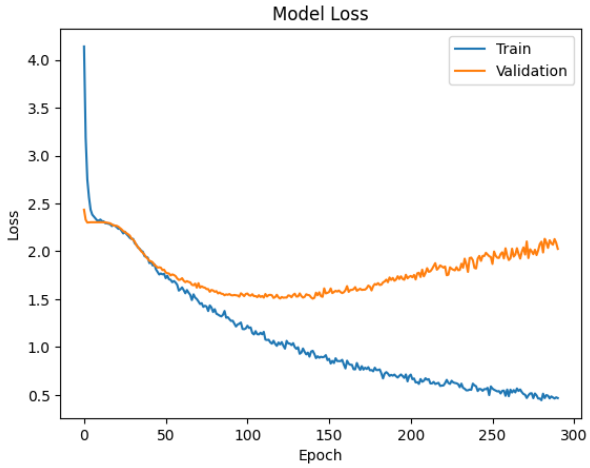

Figura 8: *Gráfico de función de pérdida*

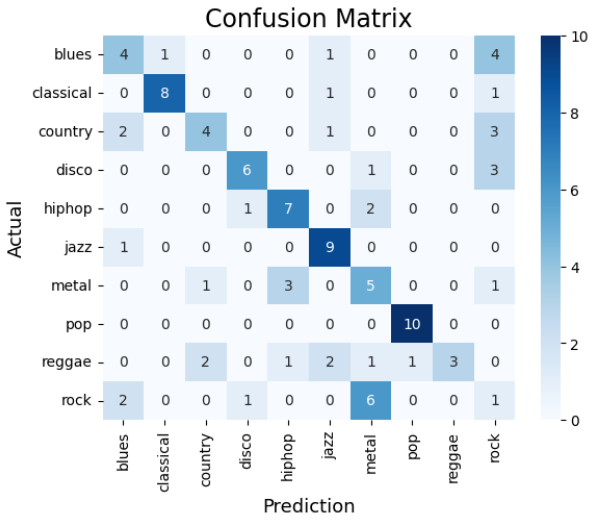

Figura 9: *Matriz de Confusión*

Finalmente con un accuracy de 0,57 y con valor para la función de pérdida de 2,9323 llegamos a la conclusión de que este modelo no arroja resultados que mejoren los obtenidos en los modelos ya vistos anteriormente.

En la experimentación con wavelets, después de un análisi más exhaustivo, se pudo deducir que los mejores resultados provienen del uso de DT-CWT con Random Forest, que al probarlo en el 20 % de GTZAN muestra 78 % de precisión.

Para corroborar los resultados anteriores fueron realizadas 30 iteraciones de Cross Validation donde la precisión promedio fue de 0.7797, y la desviación estándar de 0.0819. Sin embargo hubo muchas instancias con valores superiores a 0.90, la mayor alcanzando 0.97 de precisión.

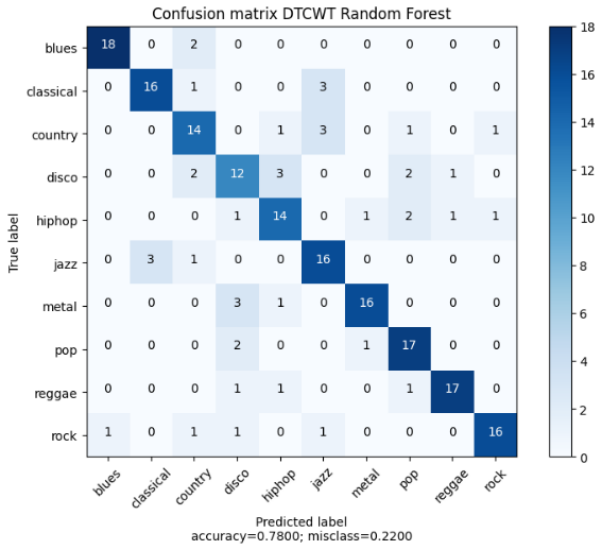

Figura 10: *Matriz de confusión de Random Forest con DT-CWT*

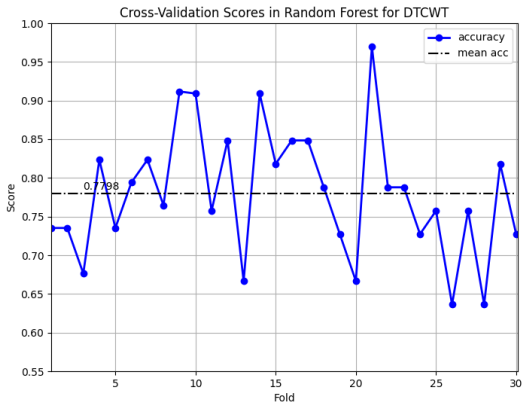

Figura 11: *DT-CWT Cross-Validation*

V. Ensemble

Como modelo final del proyecto se propuso crear un ensemble para mezclar resultados de distintos modelos preentrenados por separado y asi aprovechar la fortaleza de cada uno de estos. Se decidio usar el tipo de ensemble staking. Este consiste en entrenar los modelos base y el modelo final(el ensemble) con conjuntos de datos distintos. Para el entrenamiento se dividió la base de datos en 56 % para entrenar los modelos base, 12 % para validar, 12 % para probar(test) y 20 % para entrenar el modelo final.

Arquitectura

La arquitectura de este modelo es simple, es un modelo secuencial que consta de una capa densa de entrada con 100 neuronas con función de activacion relu, a la cual se le pasa el resultado de las predicciones de los modelos preentrenados, ergo la capa tendrá como entrada una lista de vectores, donde la posición i del vector indica el resultado predicho por el i-ésimo modelo que que pertenece a los modelos que se utilicen en una iteración dada(los modelos a usar en el ensemble son maleables). Seguido de esta capa le siguen dos capas densas más ambas de 100 neuronas y activacion relu, cierra con una capa densa de 10 neuronas con funcion de activacion softmax y con ello la salida del modelo sera un numero entre [0,9], el cual representara un genero dado.

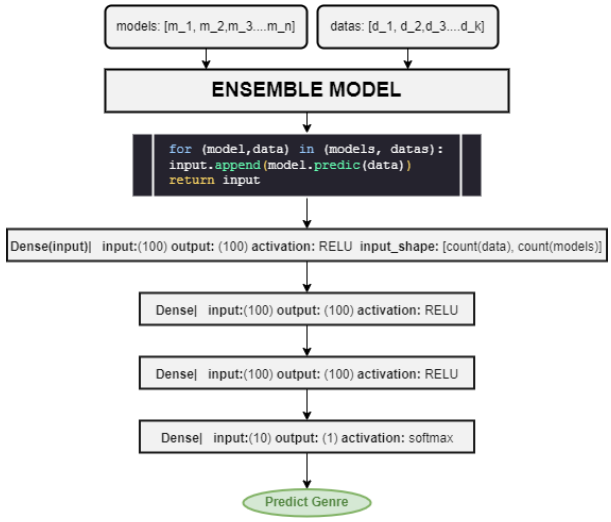

Figura 12: *Modelo del ensemble*

Resultados

Dado que la base de datos utilizada contiene una cantidad relativamente pequeña de datos al entrenar los modelos base con solo un 56 % de los datos se obtuvieron los resultados:

- mfcc-cnn: 0.61 test accuracy
- wavelets: 0.60 test accuracy

Al entrenar el modelo final con el 20 % de los datos restantes se obtuvo una precisión de 0.65, lo cual si bien no es mejor resultado que los modelos entrenados individualmente con un conjunto de datos más amplio, es un resultado que nos indica que en trabajos futuros con datasets más abarcadoras, o con una dataset ampliada que tenga suficientes datos para preentrenar bien los modelos base, el ensemble mejora los resultados de los modelos previamente entrenados aprendiendo de las fortalezas de cada uno de estos.

VI. Recomendaciones

Un aspecto crítico que afecta el rendimiento de los modelos de clasificación radica en la precisión y consistencia

de las anotaciones asignadas manualmente. Los esfuerzos futuros deberían priorizar la recopilación de datos de mayor calidad a través de pautas más estrictas, anotaciones de expertos o procedimientos de ratificación de colaboración colectiva.

Otro enfoque para impulsar el rendimiento de los modelos de clasificación de género es combinar múltiples modelos a través de ensembles. Al integrar predicciones de distintos algoritmos y representaciones de características, los ensembles aprovechan las fortalezas de los clasificadores individuales mientras mitigan sus debilidades.

Al seguir estas direcciones, los investigadores pueden ampliar nuestra comprensión de la clasificación de géneros musicales y sentar bases sólidas para soluciones de próxima generación en áreas relacionadas, incluido el análisis de audio, la recuperación de información o las interfaces interactivas centradas en el ser humano.

Referencias

[1] A. Radford, J. Wook Kim, T. Xu, G. Brockman, C. McLeavey y I. Sutskever: *Robust Speech Recognition via Large-Scale Weak Supervision*. Preprint on arXiv: 2212.04356, 2022.

[2] Safaa Allamy, Alessandro Lameiras Koerich: *1D CNN Architectures for Music Genre Classification*. Preprint on arXiv.2105.07302, 2021.

[3] Kaiming He, Xiangyu Zhang, Shaoqing Ren, Jian Sun: Deep Residual Learning for Image Recognition, 2015

[4] J. Devlin, M. Chang, K. Lee y K. Toutanova: *BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding*. Preprint on arXiv: 1810.04805, 2018.

[5] Megan Leszczynski, Anna Boonyanit: *Music Genre Classification using Song Lyrics*, 2021.

[6] Kevin P. Murphy: *Machine Learning: A Probabilistic Perspective*. MIT Press, 2012.

[7] Pranav Vijaya Kumar Rao, Vishwas Nagesh Moolimani: *ECG Analysis based feature extraction using Wavelet Transform for Music Genre Classification*, 2020.

[8] Ahmet Taspinar: *A guide for using the wavelet transform in machine learning*, unpublished. [Online]. Available: http://ataspinar.com/2018/12/21/a-guide-for-using-the-wavelet-transform-in-machine-learning/

[9] Selesnick, I.W. and Baraniuk, R.G. and Kingsbury, N.C., *The dual-tree complex wavelet transform*, 2005, IEEE Signal Processing Magazine, pp. 123-151.

[10] Liliana R. Castro, Silvia M. Castro: *Wavelets y sus Aplicaciones*, ler. Congreso Argentino de Ciencias de la Computación, pp. 195-204.

[11] George Tzanetakis, Georg Essl, Perry Cook: *Automatic Musical Genre Classification Of Audio Signals*

[12] Rusian Gohkman: *Machine Learning and Deep Learning methods for music genre Classification*

[13] Mel-frequency cepstrum (s.f.). Wikipedia. [Online]. https://en.m.wikipedia.org/wiki/Mel-frequency_cepstrum, junio 2023.

[14] Umberto Michelucci: *An Introduction to Autoencoders*. Preprint on arXiv: 2201.03898, 2022.

[15] Dor Bank and Noam Koenigstein and Raja Giryes: *Autoencoders*. Preprint on arXiv: 2003.05991, 2021.