TP3 TD6

Ezequiel Grinblat, Luca Mazzarello y Camila Migdal

November 2024

1 Introducción

Para este trabajo se nos planteo como objetivo analizar y evaluar diversos hiperparámetros durante el proceso de entrenamiento de una red neuronal, empleando la base de datos GTZAN.
A lo largo del mismo, estuvimos trabajando con el tunneo de arquitecturas e hiperparametros, funciones de activación, optimizadores y tecnicas de regularización. A partir de este proceso, evaluamos como los mismos afectan la performance del modelo, mediante la metrica *Accuracy*. Pudimos evaluar estos resultados mediante Weight and Biases. Concluímos el trabajo, seleccionando el mejor modelo encontrado. En este informe estaremos desarrollando, el proceso por el cual llegamos a ellos.

2 Configuración inicial

En la configuración inicial de nuestro trabajo, implementamos varias funciones para gestionar el preprocesamiento de datos y la configuración del entrenamiento.

La función '`classes`' se usa para listar todos los géneros musicales presentes, generando así las etiquetas de clasificación que necesitaremos durante el entrenamiento.

Para visualizar los datos, creamos '`plot_waveforms`' y '`plot_genre_waveforms`', que permiten observar los waveforms de audio y analizar la variabilidad entre géneros. La función '`stratify_split`' divide el dataset de manera estratificada en subconjuntos de entrenamiento, validación y testeo, asegurando que las clases estén balanceadas en cada conjunto.

Para el entrenamiento, implementamos funciones como '`count_parameters`' para contar el número de parámetros entrenables en el modelo y '`get_optimizer`' junto con '`get_scheduler`' para seleccionar y configurar el optimizador y el scheduler adecuados. Además, integramos '`l1_regularization`' y '`l2_regularization`' para aplicar regularización, y '`train_model`', que coordina todo el proceso de entrenamiento, utilizando las configuraciones de hiperparámetros, regularización y early stopping según los parámetros especificados.

Cada una de estas funciones, fueron utilizadas a lo largo del trabajo en los puntos pedidos por la consigna.

2.1 Transformaciones de los datos

Antes de comenzar con el análisis, comenzamos por realizar una serie de transformaciones iniciales a nuestros datos las cuales nos permitieron obtener resultados normalizados, con mayor robustez y generalizables. Para así también, poder observar correctamente los resultados tanto en *waveform* como en *espectograma* Las mismas consistieron:

- **NormalizeAudio**: Normaliza las formas de onda para mejorar la estabilidad y prevenir el sobreajuste al estandarizar los datos.
- **TimeStretchWaveform**: Cambia la velocidad de reproducción sin afectar el tono, aumentando la diversidad y robustez del modelo.
- **PadOrTruncate**: Ajusta la longitud de las formas de onda a un valor fijo, asegurando entradas uniformes para la red.
- **PitchShiftWaveform**: Modifica el tono del audio sin alterar la duración, mejorando la capacidad de generalización frente a variaciones en el tono.
- **ResampleWaveform**: Cambia la frecuencia de muestreo de las formas de onda, garantizando consistencia entre audios de diferentes frecuencias.
- **AddNoise**: Añade ruido aleatorio al espectrograma, mejorando la resistencia del modelo a ruidos y variaciones reales.

A continuación, podemos observar el resultado de las transformaciones realizadas.

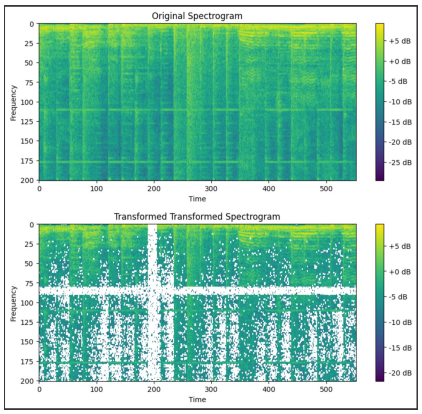

Figure 1: Imagen 1

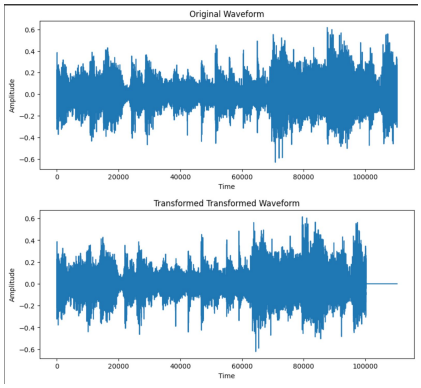

Figure 2: Imagen 2

3 Arquitectura MLP

Para esta estructura planteamos 3 configuraciones de hiperparametros, todas utilizan como función de activación la ReLu y para todas realizamos 100 epochs.

Las mismas fueron pensadas desde una red más simple, hasta una más compleja para analizar este rendimiento.

- La **configuración 1** utiliza una tasa de aprendizaje de 0.001, batch size de 32, cuatro capas densas con 256 nodos cada una, sin dropout.
- La **configuración 2** tiene una tasa de aprendizaje de 0.0005, batch size de 32, dos capas densas con 512 nodos cada una, un dropout de 0.5.
- Finalmente, la **configuración 3** emplea una tasa de aprendizaje de 0.0003, un batch size de 32, tres capas densas con 1024, 512 y 256 nodos, un dropout de 0.3.

Los resultados que obtuvimos los visualizamos unicamente con waveform, ya que al observar este primer análisis y las metricas obtenidas, no consideramos redudante seguir interiorizandonos en esta estrutcura.

Figure 3: Grafico de barras de las tres configuraciones con MLP y waveform

Podemos observar que el mejor valor de accuracy se obtiene con la **configuración 2**. Sin embargo, considerando el trade-off entre valid loss y precisión, el mejor experimento es el de la **configuración 1**. Por el contrario, el peor experimento es el de la **configuración 3**, ya que presentó el valor más bajo de accuracy y el valor más alto en valid loss.

4 CNN

Luego seguimos nuestro análisis, con la arquitectura CNN. En este caso también utilizamos para todo el experimento la función de activación **ReLu** y también realizamos 100 ephocs.

En este caso, también planteamos 3 configuraciones de redes convolucionales. También planteamos desde redes más simples a más complejas y decidimos visualizar los resultados tanto a traves de Waveform, como de espectograma con las mismas tres configuraciones. De esta forma pudimos hacer un análisis más detallado.

Las tres configuraciones consistieron en:

- **Configuración 1**: Utiliza una tasa de aprendizaje de 0.001, un batch size de 8 y una única capa densa con 32 nodos. Tiene un dropout del 0.2. Esta configuración está diseñada con una arquitectura más simple, priorizando un entrenamiento más específico.
- **Configuración 2**: En esta configuración, la tasa de aprendizaje es de 0.0005, un batch de 16, y cuenta con dos capas densas, cada una con 64 nodos. El dropout se incrementa a 0.3 y también se entrena durante 100 épocas. Esta misma busca un equilibrio entre la complejidad de la red y la regularización.
- **Configuración 3**: Esta configuración también tiene una tasa de aprendizaje de 0.0005, pero el tamaño del batch aumenta a 32, con tres capas densas que tienen 128, 128 y 64 nodos respectivamente. Al igual que la configuración 2, tiene un dropout de 0.3. Está configuración es más compleja, con una arquitectura más profunda que busca mejorar el poder representacional del modelo sin perder la regularización adecuada.

Comenzamos nuestro análisis evaluando los reusltados obtenidos a partir de **waveform.**

Figure 4: Grafico de barras de las 3 configuraciones con CNN y waveform

Podríamos considerar que la **configuración 1** es la mejor si nos basamos únicamente en la métrica de accuracy. Sin embargo, al evaluar la valid loss, observamos que este modelo muestra indicios de overfitting, ya que la comparación entre train loss y valid loss revela un alto ajuste a los datos de entrenamiento. Por lo tanto, aunque los resultados de los otros dos modelos son casi idénticos, podemos concluir que **configuración 2 y configuración 3** son las mejores, mientras que la configuración 1 es la peor en esta visualización.

Luego visualizamos las configuraciones de hiperparametro con espectograma.

En este caso, observamos algo similar a lo que ocurrió con waveform en la **configuración 1**. Evaluando el trade-off entre accuracy y valid loss, podemos afirmar que el mejor modelo se obtiene con la **configuración 3**. En cuanto al peor modelo, depende de si nos basamos en la loss o en el accuracy. A nuestro juicio, el peor modelo es la **configuración 1**, ya que muestra resultados indicativos de overfitting (observando la train loss y valid loss).

Figure 5: Grafico de barras de las 3 configuraciones con CNN y espectogramas

Figure 6: Grafico de barras con todos los modelos hechos con CNN

Por ultimo analizando los seis modelos,

En esta visualización, podemos decir que el mejor resultado del experimento se obtiene con la **configuración 3** en el espectrograma, mientras que el peor se obtiene con la **configuración 1**, tanto en el espectrograma como en el waveform, ya que ambas presentan resultados indicativos de overfitting. En cuanto a las otras dos configuraciones, aunque no muestran un valor de accuracy muy alto, los resultados son coherentes con las métricas de valid loss, lo que sugiere un mejor equilibrio entre ajuste y generalización.

5 Funciones de activación

Siguiendo con nuestro análisis, procedimos a experimentar variando la función de activación. Probamos con **ReLU**, como habíamos hecho hasta el momento, y también con **LeakyReLU** y **ELU** para evaluar su impacto en el rendimiento del modelo.

- **ReLU**: Es la función de activación más común, que reemplaza los valores negativos por cero y mantiene los valores positivos. Es simple y eficiente, pero puede sufrir de vanishing gradients en algunas situaciones. En el grafico se va a visualizar como **configuración 1**, ya que es la CNN con

la que venimos trabajando.

- **LeakyReLU**: Es una variante de ReLU que permite un pequeño gradiente para los valores negativos en lugar de hacerlos cero. Esto ayuda a prevenir el problema del dead neuron que puede ocurrir con ReLU.

- **ELU**: Combina características de ReLU y una función exponencial. Para valores negativos, aplica una función exponencial, lo que mejora la propagación de gradientes y puede acelerar el aprendizaje en redes profundas, pero es más costosa computacionalmente.

Comenzamos nuestro análisis visualizando los resultados con waveform. Donde podemos observar que el mejor resultado se obtiene con **LeakyReLU**. Esto nos pareció coherente, ya que la misma tiene en cuenta valores negativos lo cual es util con dataset de audio ya que cuenta con este tipo de valores en su dominio. Cosa que la **ReLU**, no realiza. En cuanto al peor experimento creemos que fue el realizado con **ELU** ya que presenta los valores más bajos en todas las metricas.

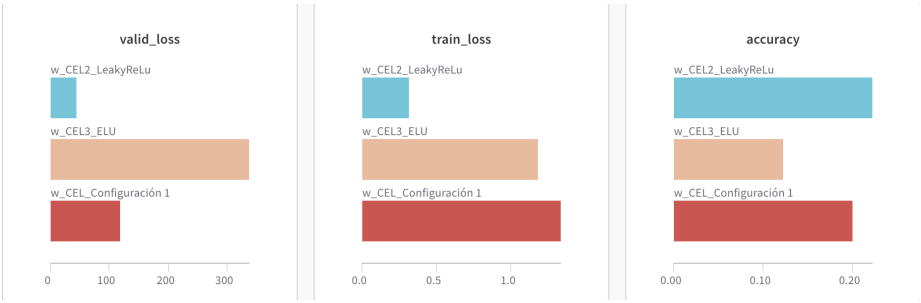

Figure 7: Grafico de barras con modelos con ReLu, LeakyReLU y ELU con waveform

Analizando la visualización para espectograma, pudimos observar que el mejor resultado en cuanto a accuracy se obtiene con **ELU**, pero observando la train loss y valid loss, podríamos inferir que se trata de un modelo con indices de underfitting. Por lo tanto consideramos que el mejor modelo fue el obtenido con **ReLU**. Por esto mismo el peor experimento va a variar dependiendo de la matrica en la que nos fijemos. Si priorizamos precisión optaríamos por un modelo con la **ELU** y en el caso de preferir una validad loss más baja optaríamos por el **LeakyReLU**.

Por ultimo, evaluando los resultados obtenidos para ambas visualizaciones. Consideramos que el mejor modelo se obtiene con la **ReLU** en espectograma (configuración 1) por lo explicado anteriormente. Y el peor consideramos que es el modelo realizado con la función **ELU** para waveform. También por lo explicado anteriormente.

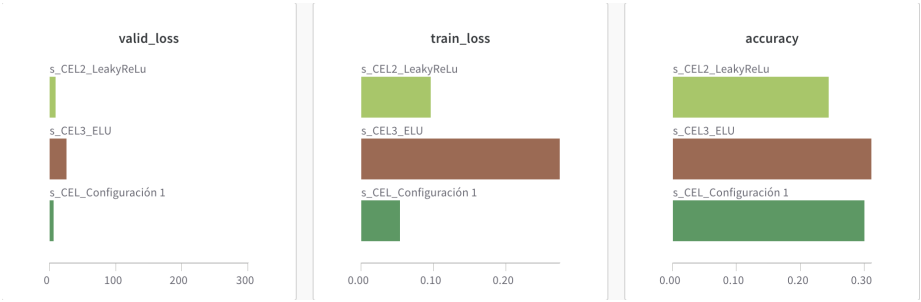

Figure 8: Grafico de barras con modelos con ReLu, LeakyReLU y ELU con espectograma

6 Optimizadores y Schedulers

Para el siguiente experimento, pasamos a hacer las pruebas con 30 epochs ya que notamos, que a partir de este momento los valores tendían a lo mismo.

Utilizamos el optimizador **ADAM** para este experimento y, además, añadimos el optimizador **SGD** para comparar su rendimiento.

En cuanto a los schedulers, probamos con **Plateau** y **Cosine**. El scheduler **Plateau** ajusta la tasa de aprendizaje cuando la métrica de validación deja de mejorar, mientras que el **Cosine** ajusta la tasa de aprendizaje siguiendo una función cosenoidal a medida que avanza el entrenamiento, lo que puede ayudar a alcanzar una convergencia más rápida y estable.

Dado los resultados obtenidos en el experimento anterior: para las visualizaciones de waveform utilizamos como función de activación **LeakyReLU** y por otro lado, para espectograma utilizamos la función de activación **ReLU** que presentaron los mejores resultados.

Figure 9: Grafico de barras con los optimizadores ADAM y SGD y schedueles Plateau y Cosine con espectograma

Al analizar los resultados obtenidos para el espectrograma, observamos que los modelos **sgd-plateau** y **sgd-cosine** son los que muestran el mayor valor de accuracy. Sin embargo, al examinar sus losses, el modelo **sgd-cosine** presenta

el valor más bajo. Por otro lado, el peor modelo es **sgd-plateau**, ya que obtiene el menor accuracy y muestra la peor relación entre los losses de entrenamiento y validación.

Figure 10: Grafico de barras con los optimizadores ADAM y SGD y schedueles Plateau y Cosine con waveform

Por último, al observar los resultados para waveform, el modelo **sgd-cosine** muestra el mayor valor de accuracy y el menor valid loss. En contraste, los peores experimentos corresponden a los representados por la barra gris y la barra violeta. Dada la pequeña diferencia en accuracy, decidimos basarnos en el valid loss. Por lo tanto, consideramos que el modelo **adam-plateau** es el peor.

Figure 11: grafico de barras que compara los graficos de la figure 9 y 10

Observando los seis modelos juntos, podemos observar que los mejores experimentos serán **sgd-cosine** y **adam-plateau** para espectograma. Con una leve diferencia en la relación entre sus losses. En cuanto a los peores experimentos, consideramos que son **adam-cosine** y **adam-plateau** para waveform.

7 Regularización

Por ultimo, realizamos los experimentos con distintos metodos de regularización (L1 y L2).

Para ambas visualizaciones, utilizamos el optimizador **cosine** y scheduler **sgd**, ya que fue la combinación que mejores resultados nos dió. Waveform con **LeakyReLu** y Espectograma con **ReLu**.

Para la visualización de espectograma, no podemos identificar cual es el mejor de los dos. Esto va a depender de que querramos ajustar ya que podríamos decir que el regularizado por **L2** tiende al overfitting y el regularizado por **L1** al underfitting. Aunque el **L1** presenta resultados más balanceados.

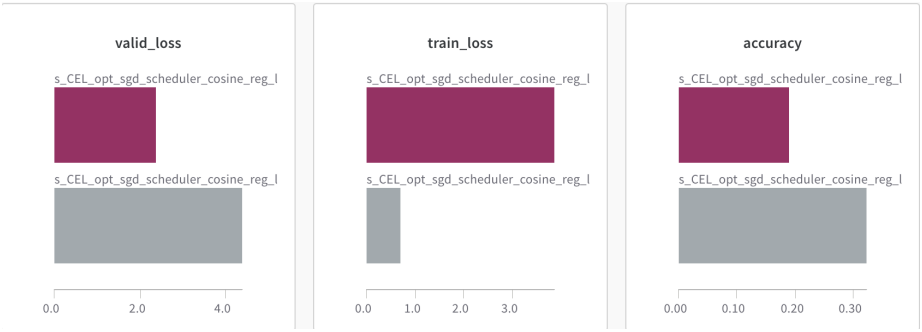

Figure 12: grafico de barras con regularizadores L1 y L2 con espectogramas

Para la visualización de waveform, podemos observar que los resultados son similares a los anteriores. Donde el modelo regularizado por **L1**, tiende al underfitting y el **L2** al overfitting.

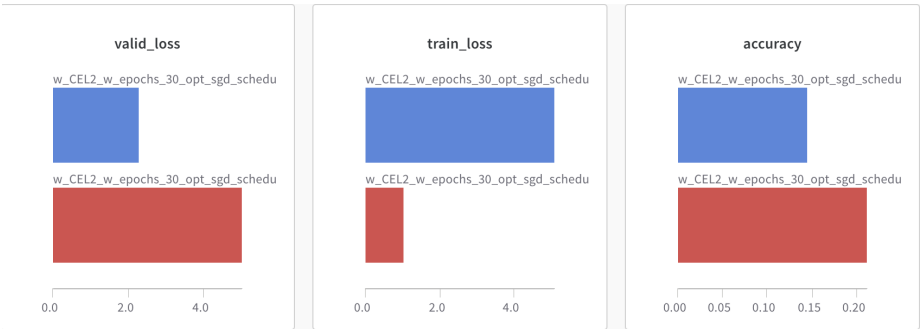

Figure 13: grafico de barras con regularizadores L1 y L2 con waveform

8 Conclusión

Luego de todos los experimentos realizados, decidimos hacer un top5 de los resultados encontrados. Estos mismos rondan un valor de accuracy de 0.3 con una valid loss entre 5 y 6. Creemos que el mejor resultado obtenido fue con el

modelo adam - plateau ya que muestra un mayor valor de accuracy en relación al resto y un modelo balanceado en relación a sus losses.

Figure 14: Top 5: Experimentos realizados

9 Aclaraciones

No hicimos un punto aparte para los diferentes experimentos con epochs y batch sizes, debido a que lo estuvimos haciendo en el trasncurso del trabajo práctico.

Un archivo de la metadata se nos descargaba sin los archivos de codigo. Dejamos los links el notebook en la primera celda para que puedan acceder.