TP4 TDVI

Ezequiel Grinblat, Luca Mazzarello y Camila Migdal

November 2024

1 Introducción

Para este proyecto se nos planteo el desafío, de implementar un codificador musical basado en redes neuronales con el fin de generar vectores representativos en un espacio latente para cada canción. Estos vectores, una vez obtenidos, podrán ser decodificados para reproducir el sonido original. A lo largo de este informe, desarrollaremos el proceso realizado y las distintas pruebas utilizadas para obtener lo más cercano a los sonidos originales.

2 Diseño del Autoencoder: CELAutoencoder

En este trabajo desarrollamos y evaluamos dos versiones iniciales de un autoencoder para procesar señales de audio unidimensionales. Ambas versiones compartieron una arquitectura base, pero diferían en la función de activación utilizada: `LeakyReLU` en la primera versión y `SiLU` en la segunda. El objetivo de esta comparación fue determinar cuál función de activación ofrecía un mejor desempeño en la reconstrucción de las señales de audio. Tras las pruebas realizadas, observamos que la activación `SiLU` presentó un rendimiento superior. A continuación, describimos en detalle la estructura base mencionada, separando la explicación en dos componentes: el encoder y el decoder.

3 Encodear la canción en un vector latente

3.1 Estructura del Encoder

El encoder está diseñado para comprimir la señal de entrada en una representación de menor dimensionalidad, extrayendo características relevantes mediante una serie de capas convolucionales, normalizaciones batch y funciones de activación. Inicialmente, la señal de audio unidimensional con un canal y una longitud de 110250 muestras es transformada por una primera capa convolucional que incrementa los canales a 16, manteniendo la longitud original gracias al uso de un padding adecuado. A continuación, se aplica una capa de max pooling que reduce la longitud de la señal a la mitad, pasando de 110250 a 55125

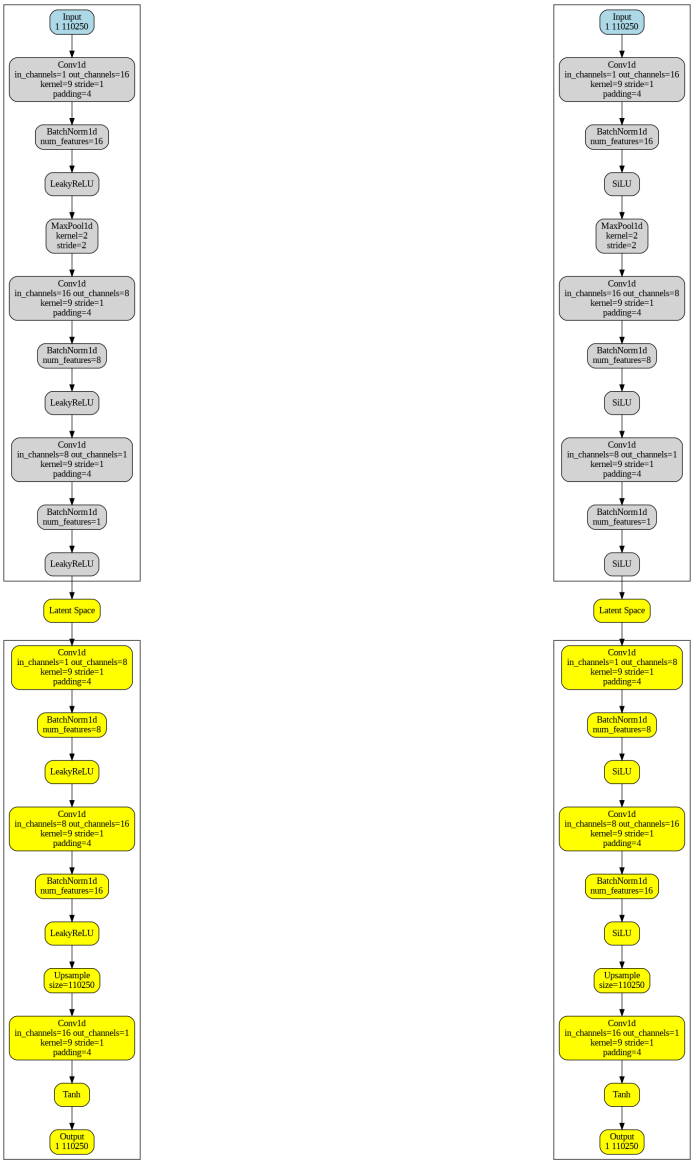
muestras, lo que permite una compresión significativa sin perder información clave.

Posteriormente, una segunda capa convolucional reduce los canales de 16 a 8, manteniendo la longitud de la señal. Esta etapa también incluye una normalización batch para estabilizar el aprendizaje y una función de activación (`LeakyReLU` o `SiLU`, dependiendo de la versión del autoencoder). Finalmente, una tercera capa convolucional comprime aún más la representación, reduciendo los canales a uno y manteniendo la longitud de 55125 muestras. La salida del encoder es, por lo tanto, un tensor con un canal y una longitud reducida, que representa una codificación comprimida de la señal original.

3.2 Estructura del Decoder

El decoder realizó la operación inversa al encoder, con el objetivo de reconstruir la señal original a partir de la representación comprimida. Comenzó con una capa convolucional que incrementó los canales de uno a ocho, seguida de una normalización batch y la función de activación correspondiente. Luego, una segunda capa convolucional incrementó los canales de ocho a dieciséis, manteniendo la longitud de la señal en 55125 muestras.

Para restaurar la longitud original de la señal, se utilizó una capa de `Upsample` que duplicó la longitud de la señal de 55125 a 110250 muestras. Finalmente, una capa convolucional redujo los canales de dieciséis a uno, generando la señal reconstruida. Se aplicó una activación `Tanh` en esta última capa para normalizar los valores de salida en un rango entre -1 y 1, adecuado para trabajar con audios. De esta manera, el decoder transformó la representación comprimida de vuelta a la forma original de la señal de audio.



(a) Estructura del CELAutoencoderLeakyReLU

(b) Estructura del CELAutoencoderSiLU

Figure 1: Estructura de ambos Autoencoders

3.3 Primeros resultados

En la Figura 2 (correspondiente al CELAutoencoder con LeakyReLU), se observa que la señal reconstruida mantiene una estructura general similar a la señal

original. Sin embargo, la reconstrucción presenta una mayor suavización en la amplitud, lo que evidencia una pérdida de detalles finos. Este comportamiento se ve reflejado en los resultados obtenidos durante el entrenamiento, donde se alcanzó una train loss de 0.0031 y una valid loss de 0.0036.

Por otro lado, en la Figura 3 (correspondiente al CELAutoencoder con SiLU), la reconstrucción de la señal muestra una mayor precisión, conservando mejor los detalles finos y alineándose de manera más cercana con la señal original. Este resultado se corresponde con las métricas obtenidas, donde se logró una train loss de 0.0018 y una valid loss de 0.0020, demostrando no solo una reconstrucción más fiel, sino también una convergencia más rápida del modelo.

Ambos autoencoders lograron reducir el tamaño de la señal de entrada, de dimensiones 1x110250, a un vector latente comprimido de 1x55125, lo que evidencia la capacidad de codificación de ambas arquitecturas. Sin embargo, los resultados muestran que el CELAutoencoder con SiLU es más efectivo en la reconstrucción de señales de audio, tanto en términos de calidad como de eficiencia en el entrenamiento.

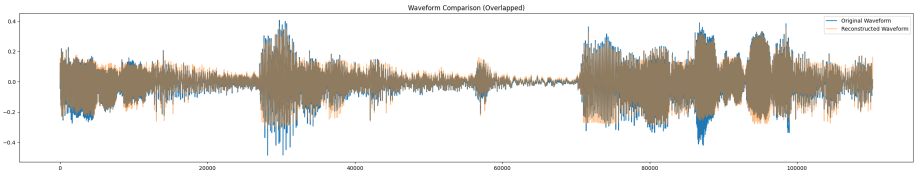


Figure 2: Reconstrucción con CELAutoencoderLeakyReLU

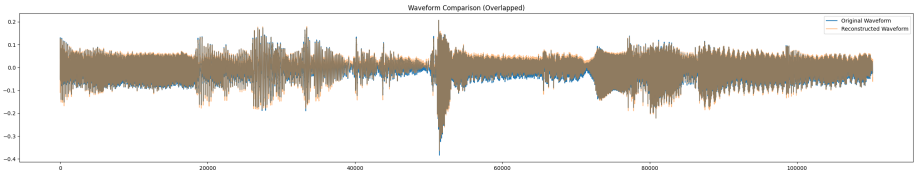


Figure 3: Reconstrucción con CELAutoencoderSiLU

4 Análisis exploratorio de vectores latentes

Para evaluar el impacto del tamaño del espacio latente en la reconstrucción de las señales, realizamos dos pruebas adicionales utilizando variaciones en el diseño del autoencoder base. Exploramos un espacio latente más chico, representado un espacio latente equivalente a un cuarto del tamaño original, y otro más pequeño, cuyo espacio es de un decimo con respecto al tamaño original, con los modelos CELAutoencoder2 y CELAutoencoder3, respectivamente.

El CELAutoencoder2 fue diseñado con un espacio latente más pequeño, de dimensiones 1x27563, en comparación con el modelo base (CELAutoencoderSiLU) cuyo espacio latente era de 1x55125. El objetivo de esta variación fue

evaluar si una compresión mayor podía mantener la calidad de reconstrucción al reducir la cantidad de información almacenada.

El CELAutoencoder2, con un espacio latente más pequeño de 1x27563, logró una train loss de 0.0032 y una valid loss de 0.0035, demostrando un buen desempeño en la reconstrucción de la señal. No obstante, la reducción del espacio latente resultó en una ligera pérdida de precisión en comparación con el modelo base (1x55125), lo que sugiere que un tamaño intermedio podría ser óptimo para equilibrar compresión y calidad al reconstruir.

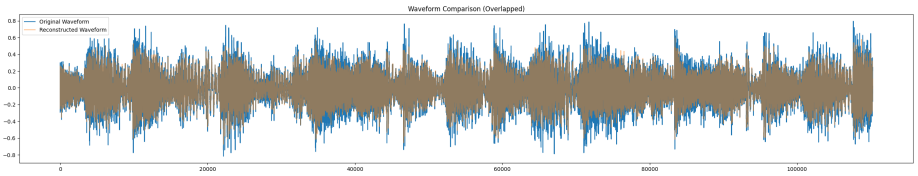


Figure 4: Resultados del CELAutoencoder2

Luego, el CELAutoencoder3 se diseño con un espacio latente más pequeño, de dimensiones 1x11025, con el objetivo de evaluar el efecto de una compresión extrema en la calidad de la reconstrucción. Este modelo reduce significativamente el tamaño del espacio latente en comparación con el CELAutoencoder-SiLU (1x55125) y el CELAutoencoder2 (1x27563).

El modelo presentó una valid loss más alta, con un valor de 0.0068, en comparación con las versiones anteriores, lo que sugiere que la reducción extrema del espacio latente afecta negativamente la capacidad del autoencoder para reconstruir la señal original con precisión.

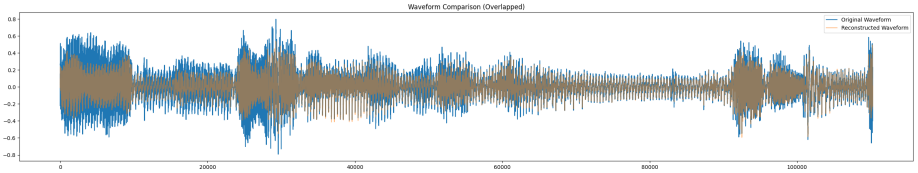
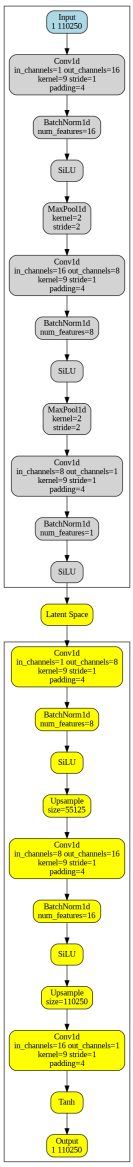
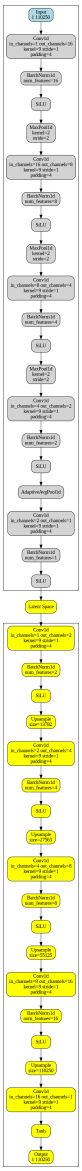


Figure 5: Resultados del CELAutoencoder3

En la figura 5, se observa que la señal reconstruida (en marrón) pierde precisión con respecto a la señal original (en azul). Aunque la estructura general de la señal se mantiene, hay una suavización considerable y una pérdida notable de los detalles más finos, especialmente en las regiones de alta frecuencia y mayor complejidad. Esta pérdida de información es esperable debido a la limitación del espacio latente, que no logra capturar de manera efectiva todas las características esenciales de la señal original.



(a) Estructura del CELAutoencoderCEL2 (EL = 1x2756)



(b) Estructura del CELAutoencoderCEL3 (EL = 1x11025)

Figure 6: Estructura de nuevos Autoencoders

5 Modelos de Clasificación

5.1 Supervisado

En esta sección, implementamos un clasificador de géneros musicales utilizando aprendizaje supervisado. Aprovechamos los autoencoders previamente entrenados como extractores de características para reducir la dimensionalidad de las señales de audio y obtener representaciones más compactas y útiles de las mismas. Estas representaciones del espacio latente son utilizadas como entrada para un clasificador basado en redes neuronales, cuya tarea principal es predecir el género al que pertenece cada señal.

Para lograr esto, utilizamos un MusicGenreClassifier, un modelo que incorpora el encoder preentrenado de cada autoencoder (CELAutoencoderSiLU, CELAutoencoder2 y CELAutoencoder3). El encoder se congela durante el entrenamiento, lo que significa que sus parámetros no se actualizan, permitiendo que el modelo clasificador utilice las características ya aprendidas. Posteriormente, añadimos capas completamente conectadas (fully connected) con funciones de activación ReLU, dropout para regularización y una capa final que genera la predicción de los géneros musicales.

El proceso de entrenamiento y validación del clasificador se lleva a cabo de manera supervisada utilizando CrossEntropyLoss como función de costo y Adam como optimizador. Para evitar el sobreajuste, implementamos un scheduler de reducción de tasa de aprendizaje basado en la pérdida de validación. Además, evaluamos el desempeño del clasificador utilizando métricas como la exactitud (accuracy), la matriz de confusión y un reporte detallado de clasificación.

En las siguientes subsecciones, detallaremos los resultados obtenidos para cada modelo.

5.1.1 Resultados con CELAutoencoderSiLU

Para el **CELAutoencoderSiLU**, utilizamos su encoder preentrenado con un espacio latente de **1x55125** como extractor de características. La representación comprimida de la señal se alimentó a nuestro clasificador supervisado, diseñado con capas completamente conectadas y activación **ReLU**.

Métricas de desempeño

- **Pérdida de validación más baja:** 1.9677
- **Exactitud en Validación:** 23.00%

A continuación se presenta el reporte detallado de clasificación para cada género musical:

Género	**Precisión**	**Recall**	**F1-Score**	**Soporte**
reggae	0.17	0.14	0.15	7
pop	0.14	1.00	0.24	5
disco	0.00	0.00	0.00	13
hiphop	0.00	0.00	0.00	15
jazz	0.00	0.00	0.00	12
classical	0.52	1.00	0.68	16
country	0.00	0.00	0.00	7
rock	0.00	0.00	0.00	10
metal	0.00	0.00	0.00	7
blues	0.08	0.12	0.10	8
Exactitud		23.00%		
Macro avg	0.09	0.23	0.12	100
Weighted avg	0.11	0.23	0.14	100

Table 1: Reporte de clasificación para CELAutoencoderSiLU

La matriz de confusión en la Figura 7 muestra el desempeño del clasificador para los diferentes géneros musicales:

- El modelo logró una clasificación perfecta para el género **classical** con 16 predicciones correctas.
- Para el género **pop**, el recall fue de **1.00**, indicando que todas las muestras de este género fueron clasificadas correctamente.
- Géneros como **disco, hiphop, jazz, rock, metal** y **country** muestran un desempeño bajo, con precisión y recall cercanos a cero.

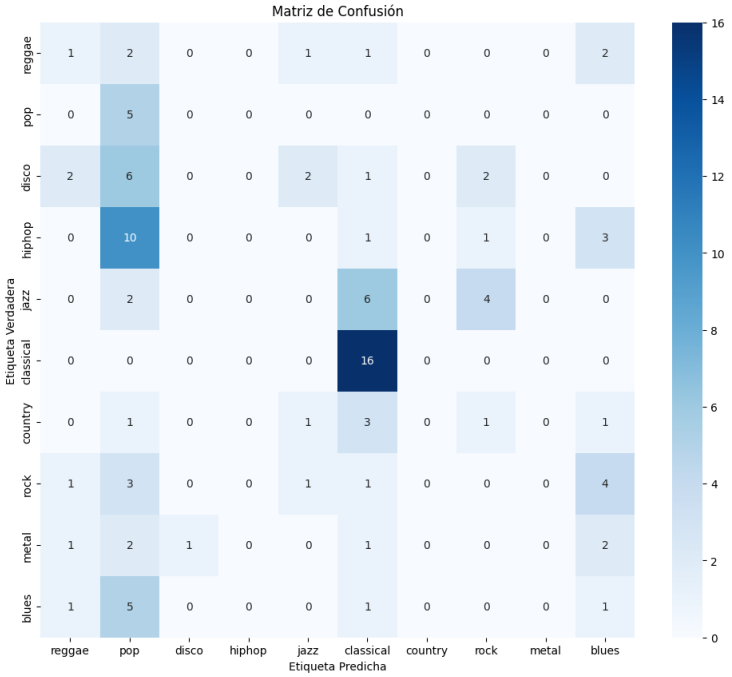


Figure 7: Matriz de confusión para CELAutoencoderSiLU

El desempeño general del clasificador con el **CELAutoencoderSiLU** evidencia una exactitud global del **23.00%**, con un claro sesgo hacia géneros como **classical** y **pop**. Esto sugiere lo siguiente:

- El modelo identifica patrones distintivos en géneros bien definidos como **classical**.
- La pérdida de información en el espacio latente afecta más a géneros con señales complejas o menos representadas, como **rock, metal** o **disco**.

5.1.2 Resultados con CELAutoencoder2

Para el **CELAutoencoder2**, utilizamos su encoder preentrenado con un espacio latente de **1x27563** como extractor de características. La representación comprimida de la señal se utilizó como entrada al clasificador supervisado.

Métricas de desempeño

- **Pérdida de validación más baja:** 1.9776
- **Exactitud en Validación:** 29.00%

A continuación se presenta el reporte detallado de clasificación para cada género musical:

Género	Precisión	Recall	F1-Score	Soporte
reggae	0.00	0.00	0.00	7
pop	0.14	1.00	0.24	5
disco	0.14	0.15	0.15	13
hiphop	0.00	0.00	0.00	15
jazz	0.29	0.17	0.21	12
classical	0.52	1.00	0.68	16
country	0.00	0.00	0.00	7
rock	0.33	0.40	0.36	10
metal	0.00	0.00	0.00	7
blues	0.00	0.00	0.00	8
Exactitud		29.00%		
Macro avg	0.14	0.27	0.16	100
Weighted avg	0.18	0.29	0.20	100

Table 2: Reporte de clasificación para CELAutoencoder2

La Figura 8 muestra la matriz de confusión para el clasificador basado en el **CELAutoencoder2**:

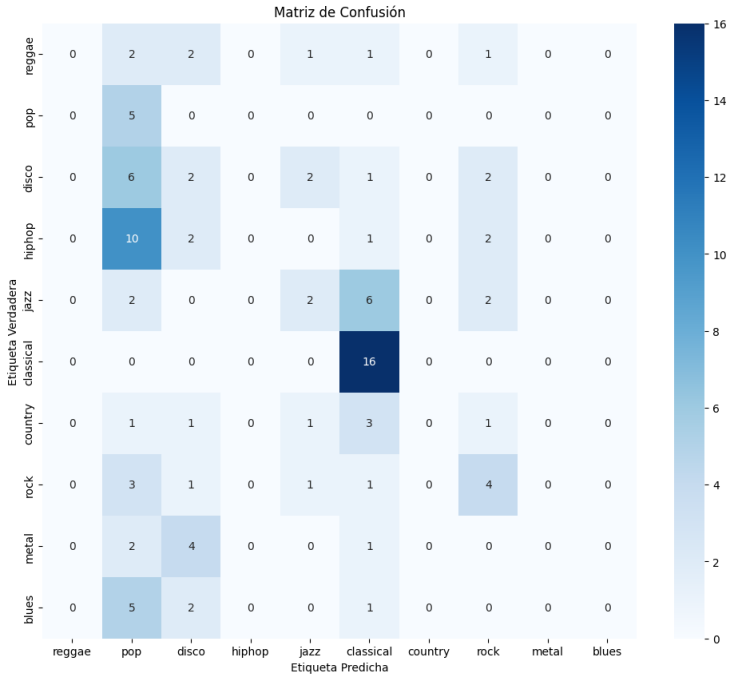


Figure 8: Matriz de confusión para CELAutoencoder2

El desempeño general del clasificador con el **CELAutoencoder2** muestra

una mejora respecto al modelo anterior, alcanzando una exactitud global de **29.00%**. Las principales observaciones son las siguientes:

- El modelo clasifica correctamente todas las muestras del género **classical**, con un **recall** de 1.00 y un **f1-score** de 0.68.
- En el género **pop**, se observa nuevamente un **recall** de 1.00, aunque con una precisión baja (0.14), lo que indica confusiones con otros géneros.
- Se aprecia un rendimiento ligeramente mejor en **rock** y **jazz**, con f1-scores de 0.36 y 0.21, respectivamente.
- Géneros como **hiphop, reggae, metal** y **blues** mantienen un desempeño cercano a cero, lo que evidencia dificultades para identificar características distintivas en estos casos.

5.1.3 Resultados con CELAutoencoder3

Para el **CELAutoencoder3**, utilizamos su encoder preentrenado con un espacio latente de **1x11025**, el más reducido entre los modelos evaluados. La señal comprimida fue utilizada como entrada al clasificador supervisado para predecir el género musical correspondiente.

Métricas de desempeño

- **Pérdida de validación más baja:** 1.9530
- **Exactitud en Validación:** 30.00%

A continuación, se presenta el reporte detallado de clasificación para cada género musical:

Género	**Precisión**	**Recall**	**F1-Score**	**Soporte**
disco	0.20	0.31	0.24	13
rock	0.00	0.00	0.00	10
blues	0.00	0.00	0.00	8
jazz	0.25	0.08	0.12	12
hiphop	0.32	0.53	0.40	15
country	0.14	0.14	0.14	7
classical	0.47	1.00	0.64	16
pop	0.00	0.00	0.00	5
reggae	0.00	0.00	0.00	7
metal	0.00	0.00	0.00	7
Exactitud		30.00%		
Macro avg	0.14	0.21	0.16	100
Weighted avg	0.19	0.30	0.22	100

Table 3: Reporte de clasificación para CELAutoencoder3

La Figura 9 muestra la matriz de confusión para el clasificador basado en el **CELAutoencoder3**:

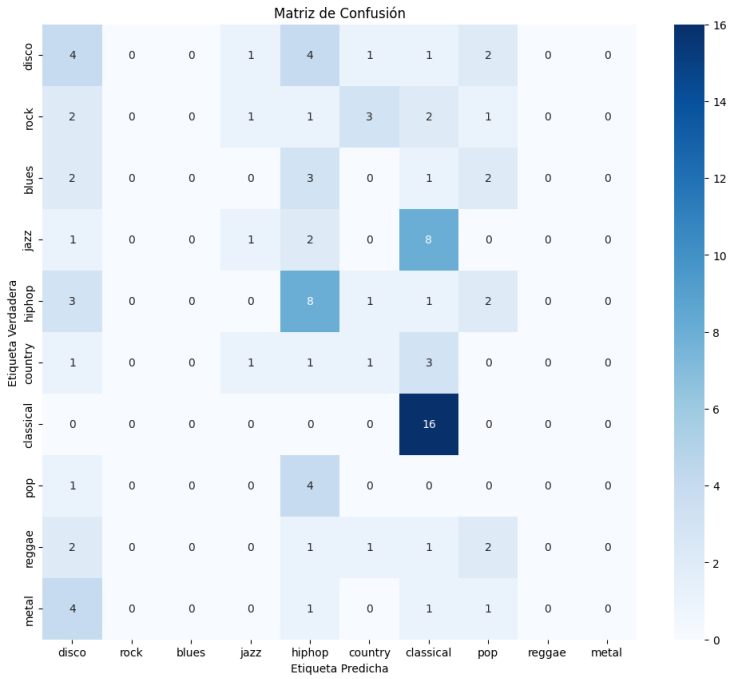


Figure 9: Matriz de confusión para CELAutoencoder3

El desempeño general del clasificador con el **CELAutoencoder3** alcanzó una exactitud global de **30.00%**, la más alta entre los modelos evaluados. Las observaciones principales son las siguientes:

- El género **classical** vuelve a ser el mejor clasificado, con un **recall** de 1.00 y un **f1-score** de 0.64.
- El modelo mostró mejoras en la clasificación de **hiphop**, alcanzando un **f1-score** de 0.40 y un **recall** de 0.53.
- **Disco** y **country** lograron resultados moderados con **f1-scores** de 0.24 y 0.14, respectivamente.
- Géneros como **rock, blues, reggae** y **metal** continúan presentando un rendimiento bajo, con precisiones y recalls de 0.00.

En conclusión, los resultados obtenidos muestran que cada uno de los autoencoders evaluados presenta ventajas y desafíos específicos en la clasificación de géneros musicales. El CELAutoencoderSiLU proporciona un buen punto de partida para el clasificador, aunque enfrenta dificultades al identificar géneros

con características menos dominantes. Por su parte, el CELAutoencoder2 logra mejorar el desempeño global del clasificador en comparación con el modelo base, pero aún tiene dificultades con géneros que presentan señales menos definidas o más complejas. Finalmente, el CELAutoencoder3, a pesar de contar con el espacio latente más reducido, alcanzó la mayor exactitud general entre los modelos evaluados. Esto sugiere que, aunque la compresión extrema puede resultar en pérdida de información, el clasificador logra identificar características relevantes para ciertos géneros específicos.

5.2 No supervisado

En esta sección realizamos un análisis no supervisado utilizando el algoritmo de **K-means** aplicado al espacio latente de cada autoencoder entrenado. A diferencia del enfoque supervisado, aquí no se emplean etiquetas para el entrenamiento, permitiendo descubrir agrupamientos naturales basados únicamente en las características comprimidas en el espacio latente.

Para facilitar la visualización y análisis, aplicamos una reducción de dimensionalidad utilizando **PCA** (Análisis de Componentes Principales). Originalmente, estábamos utilizando las **waveforms** de los audios como datos de entrada, que tienen una naturaleza unidimensional. Sin embargo, al pasar por el encoder del autoencoder, estas representaciones se convierten en espacios latentes de mayor dimensionalidad. La reducción a 2 dimensiones mediante PCA resulta conveniente porque nos permite **graficar las representaciones** en un espacio bidimensional, facilitando la interpretación visual y la detección de patrones en los datos. Sin esta transformación, sería imposible visualizar adecuadamente los clusters generados en dimensiones superiores.

5.2.1 Clustering de CELAutoencoderSiLU

Para este experimento, utilizamos el **CELAutoencoderSiLU** como extractor de características. Inicialmente, se extrajeron las representaciones comprimidas de las señales de audio utilizando el **encoder** del autoencoder. Posteriormente, para reducir la dimensionalidad y facilitar la visualización, aplicamos **PCA** (Análisis de Componentes Principales), proyectando las características en un espacio bidimensional. La varianza explicada por las dos primeras componentes principales fue de un total del 8.75%, distribuyéndose como 7.12% para la primera componente y 1.63% para la segunda.

El siguiente paso consistió en aplicar el algoritmo **K-means** sobre estas representaciones comprimidas en 2D. El número de clusters, k, se exploró en un rango de 2 a 20, utilizando el **método del codo** para identificar el valor óptimo. Este método evalúa la **inercia**, es decir, la suma de las distancias cuadradas entre los puntos y sus centroides. Como se observa en la figura 10, la inercia disminuye rápidamente hasta estabilizarse alrededor de $k = 17$, sugiriendo que este valor proporciona un buen equilibrio entre simplicidad y calidad del agrupamiento.

Además del método del codo, evaluamos las métricas de **homogeneidad**, **completitud** y **V-measure** para cada valor de k. La homogeneidad mide si cada cluster contiene únicamente elementos de una misma clase, mientras que la completitud evalúa si todos los elementos de una misma clase se encuentran dentro del mismo cluster. La V-measure es el promedio armónico entre estas dos métricas. Los resultados muestran un aumento progresivo en estas métricas hasta alcanzar su máximo cerca de $k = 17$, lo cual coincide con la selección del método del codo.

Por último, calculamos un **balance score** como el promedio entre la homogeneidad y la completitud, obteniendo nuevamente un valor máximo en $k = 17$. Esto refuerza la selección de este número de clusters como el óptimo para la agrupación de las señales comprimidas.

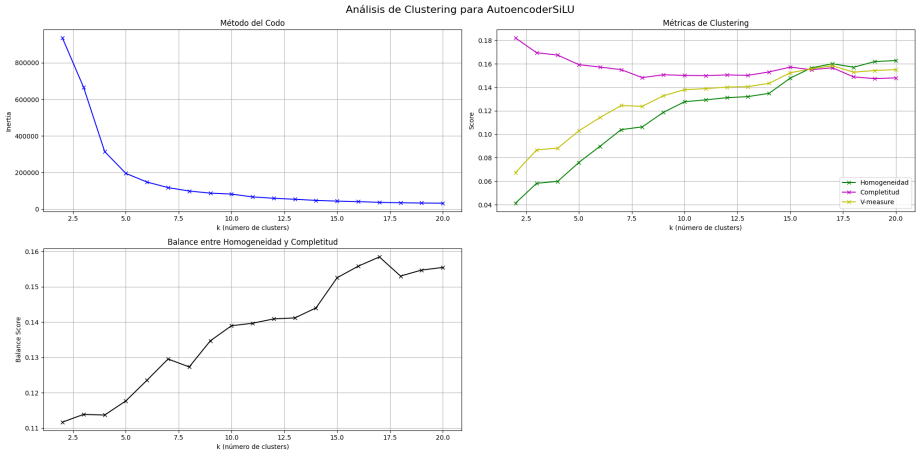


Figure 10: K-means Clustering en 2D para CELAutoencoderSiLU (k=17)

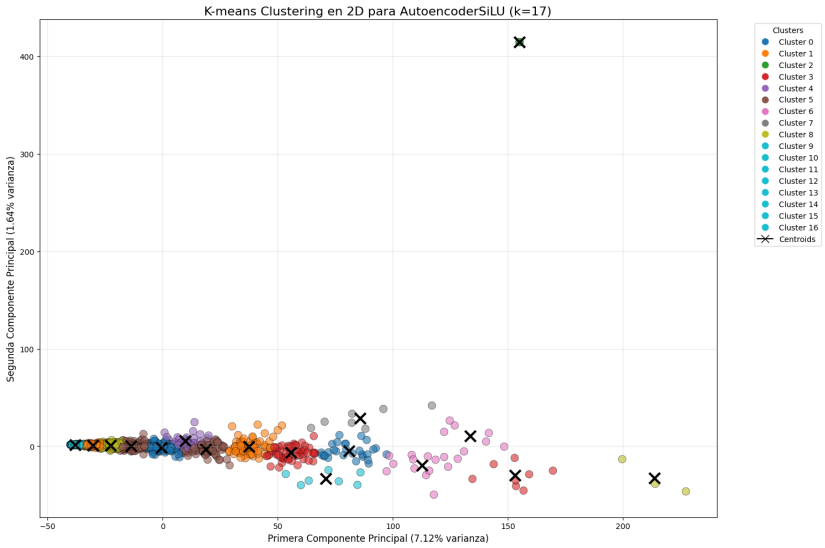


Figure 11: Visualización de los clusters y centroides identificados para el CELAutoencoderSiLU

5.2.2 Clustering de CELAutoencoder2

En este análisis, utilizamos el **CELAutoencoder2** como extractor de características para el clustering no supervisado. Similar al experimento anterior, aplicamos **PCA** (Análisis de Componentes Principales) para reducir la dimensionalidad del espacio latente a 2 componentes principales. Esta reducción es esencial debido a que las señales originales, al ser comprimidas en el espacio latente, se transforman en datos de mayor dimensión, dificultando su interpretación visual. La proyección a un espacio bidimensional permite visualizar los agrupamientos resultantes de manera intuitiva y facilita la comparación de los clusters generados por el algoritmo **K-means**.

La varianza explicada por las dos primeras componentes principales alcanzó un total del **8.65%**, donde la primera componente explicó un **7.03%** y la segunda un **1.63%** de la varianza. Aunque este valor es bajo, es suficiente para capturar patrones globales en las representaciones comprimidas del autoencoder.

A continuación, aplicamos el algoritmo **K-means** para encontrar agrupamientos naturales en los datos proyectados. Exploramos un rango de valores para k, desde 2 hasta 20, evaluando la inercia mediante el **método del codo**. Este método permitió identificar un punto de inflexión en la curva de inercia, sugiriendo que el número óptimo de clusters es $k = 17$, como se observa en la figura.

Además, se evaluaron las métricas de **homogeneidad**, **completitud** y **V-measure**. Estas métricas aumentaron progresivamente hasta alcanzar valores máximos cercanos al k óptimo. Finalmente, calculamos el **balance score**, definido como el promedio entre homogeneidad y completitud. Este balance

también confirmó que $k = 17$ es el valor que mejor organiza las representaciones latentes.

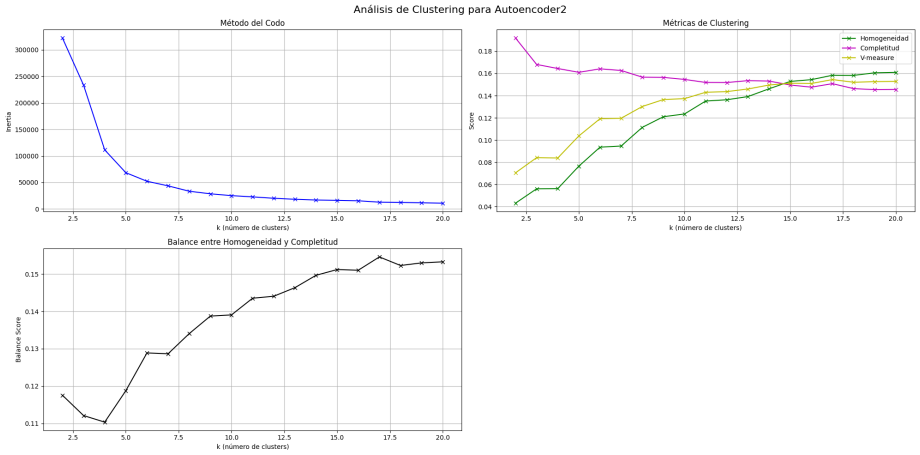


Figure 12: Análisis de Clustering para CELAutoencoder2

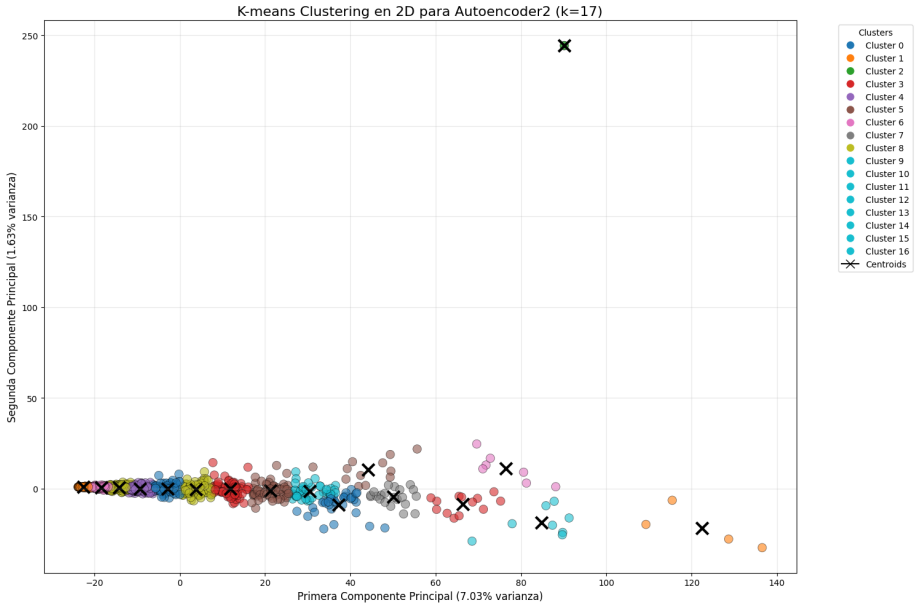


Figure 13: K-means Clustering en 2D para CELAutoencoder2 (k=17)

Para el **CELAutoencoder3**, realizamos el mismo procedimiento de análisis. El espacio latente comprimido generado por su **encoder** fue reducido a 2 dimensiones utilizando **PCA**. En este caso, la varianza explicada por las dos primeras

componentes principales fue ligeramente mayor, alcanzando un total de 11.58, con un 8.91 explicado por la primera componente y un 2.67 por la segunda.

El análisis del **método del codo** reveló que el valor óptimo del número de clusters fue $k = 20$. A diferencia de los dos modelos anteriores, este resultado sugiere una mayor fragmentación de los datos en agrupamientos más específicos. Las métricas de **homogeneidad**, **completitud** y **V-measure** mostraron un comportamiento similar, con un incremento progresivo hasta estabilizarse en $k = 20$. El **balance score** también confirmó este valor como el óptimo.

La mayor cantidad de clusters identificada para el **CELAutoencoder3** podría explicarse por su espacio latente más reducido, que obliga al modelo a conservar solo las características más esenciales de las señales. Como resultado, el algoritmo **K-means** detecta agrupamientos más finos y específicos en comparación con los modelos anteriores.

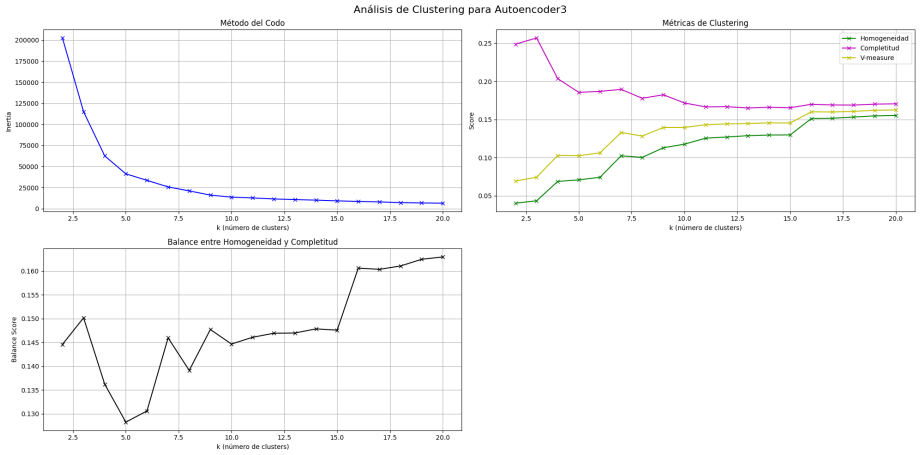


Figure 14: Análisis de Clustering para CELAutoencoder3

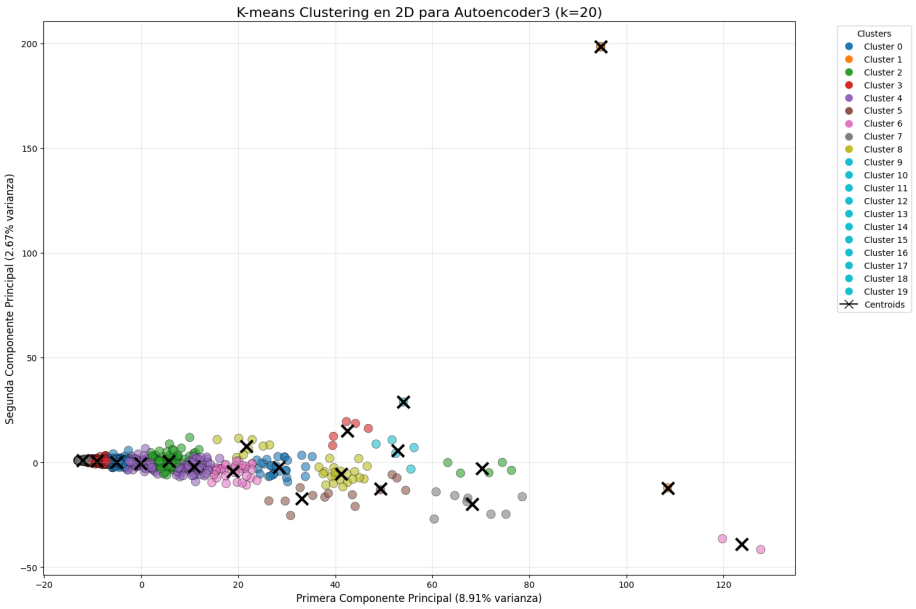


Figure 15: Distribución de Clusters para CELAutoencoder3 en 2D

5.2.3 Conclusión General del Clustering

En los tres experimentos realizados con **CELAutoencoderSiLU**, **CELAutoencoder2** y **CELAutoencoder3**, observamos que el algoritmo **K-means** aplicado a los espacios latentes logró identificar agrupamientos naturales en los datos. A pesar de que la varianza explicada por PCA fue baja en todos los casos, los resultados del clustering fueron coherentes y estructurados.

El **CELAutoencoderSiLU** y el **CELAutoencoder2** produjeron un número óptimo de **17 clusters**, lo que sugiere que ambos modelos capturan características similares en sus espacios latentes. Por otro lado, el **CELAutoencoder3**, con su espacio latente más reducido, identificó un mayor número de clusters (**20**), reflejando una fragmentación más fina y específica de los datos.

En general, estos resultados demuestran que los espacios latentes generados por los autoencoders preservan información suficiente para realizar tareas no supervisadas, como el **clustering**, permitiendo descubrir patrones ocultos en los datos sin la necesidad de etiquetas.

Table 4: Comparación de Resultados de Clustering para los Autoencoders

Modelo	Var. PCA Total	K Óptimo	Homogeneidad	Completitud	V-measure	ARI	AMI
AutoencoderSiLU	8.75%	17	0.160	0.157	0.158	0.086	0.130
Autoencoder2	8.65%	17	0.158	0.151	0.154	0.083	0.126
Autoencoder3	11.58%	20	0.155	0.171	0.163	0.086	0.129

5.3 Supervisado vs No Supervisado

Los resultados obtenidos de los modelos supervisados y no supervisados muestran las capacidades y limitaciones de los autoencoders evaluados (**CELAutoencoderSiLU**, **CELAutoencoder2** y **CELAutoencoder3**) en la clasificación y el análisis de las señales de audio correspondientes a diferentes géneros musicales.

En el enfoque **supervisado**, el **CELAutoencoder3** destacó al alcanzar la mayor exactitud general, a pesar de contar con el espacio latente más reducido. Esto sugiere que una mayor compresión de las características latentes no necesariamente implica pérdida de información crítica, sino que en algunos casos puede facilitar la identificación de patrones relevantes para la clasificación. Sin embargo, todos los modelos enfrentaron dificultades al clasificar géneros con señales menos definidas o características dominantes menos evidentes, como es común en tareas de audio complejas.

Por otro lado, en el enfoque **no supervisado**, los resultados del clustering con **K-means** aplicados a los espacios latentes reflejan una mayor fragmentación de los datos. En particular, el **CELAutoencoder3** identificó un mayor número de clusters (**20**) en comparación con los otros dos modelos (**17 clusters**). Esta diferencia podría deberse a la capacidad del algoritmo para descubrir **sub-géneros** o variaciones sutiles dentro de los géneros musicales utilizados, lo cual no era evidente en el análisis supervisado. La mayor cantidad de clusters en el modelo **CELAutoencoder3** también sugiere que la compresión extrema podría estar capturando detalles más específicos de las representaciones latentes.

En términos de comparación entre los enfoques, mientras que el análisis supervisado se enfoca en identificar patrones específicos asociados a las etiquetas de los géneros, el enfoque no supervisado permite una exploración más libre de las representaciones latentes, revelando estructuras ocultas como la posible existencia de sub-géneros y agrupamientos naturales de las señales. Esta diferencia es especialmente relevante en tareas donde la clasificación estricta no es suficiente para capturar toda la variabilidad presente en los datos.

Tanto los modelos supervisados como los no supervisados demuestran que los autoencoders pueden generar representaciones latentes útiles para analizar datos complejos como las señales de audio. La combinación de ambos enfoques ofrece un análisis más completo: el aprendizaje supervisado proporciona una medida de desempeño en tareas de clasificación específicas, mientras que el clustering no supervisado revela patrones adicionales y agrupamientos naturales que podrían enriquecer la interpretación de los resultados, como la identificación de subgéneros musicales o características sutiles entre los géneros.

6 Encodear música nueva

En esta sección, se evaluó la capacidad del autoencoder para generar una representación latente de una canción que no forma parte del conjunto de datos

utilizado para su entrenamiento, con el objetivo de verificar su desempeño al trabajar con música nueva. El proceso comenzó con la carga de una canción MP3, la cual fue convertida a formato WAV y luego preprocesada para ajustarse a los requisitos del modelo. Esto incluyó la conversión a mono, el re-muestreo al sample rate objetivo y el padding de la canción para que tuviera una duración uniforme de 5 segundos. Posteriormente, se normalizó la señal de audio y se ajustó su forma para ser compatible con el modelo de autoencoder.

Este procedimiento fue realizado utilizando tres modelos de autoencoder diferentes: CELAutoencoderSiLU, CELAutoeconder2 y CELAutoencoder3. Con los modelos CELAutoencoderSiLU y CELAutoencoder2, los resultados fueron satisfactorios, ya que las canciones reconstruidas mostraron una buena calidad y fidelidad respecto a las originales. Sin embargo, con el CELAutoencoder3, que posee un espacio latente más pequeño, los resultados fueron significativamente peores, ya que la capacidad de este modelo para capturar la información latente necesaria para la reconstrucción de la canción fue limitada. Esto sugiere que el tamaño del espacio latente influye directamente en la capacidad del modelo para generalizar y reconstruir adecuadamente música fuera del conjunto de entrenamiento.

Una vez preprocesado el audio, el modelo fue utilizado para generar una versión reconstruida de la canción a partir de su representación latente. Se compararon la forma de onda original y la reconstruida, tanto visualmente como mediante la reproducción de ambos audios. La calidad de la reconstrucción puede ser evaluada observando las diferencias en las formas de onda y la fidelidad del audio reconstruido respecto al original. Este procedimiento permite evaluar si el autoencoder puede generalizar su capacidad de codificación y decodificación a nuevas muestras de audio, fuera del conjunto de entrenamiento utilizado.

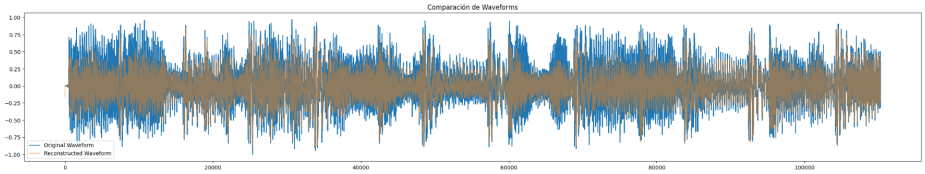


Figure 16: Autoencoder aplicado a la canción "Me quiere mucho, poquito o nada"

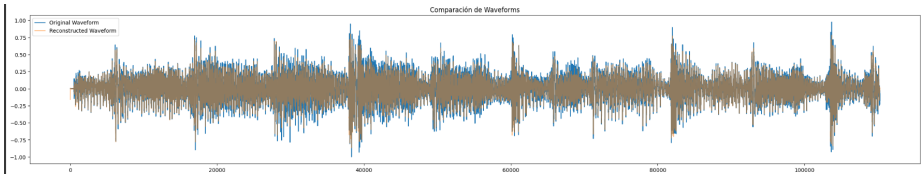


Figure 17: Autoencoder aplicado a la canción "Flores Amarillas"

7 Generación de música

Durante el procesamiento del audio con el autoencoder, hemos realizado diversas modificaciones en el espacio latente con el objetivo de observar cómo estas alteraciones impactan las características del audio reconstruido. Al manipular esta representación compacta, es posible generar nuevas variantes del audio que reflejan los cambios aplicados.

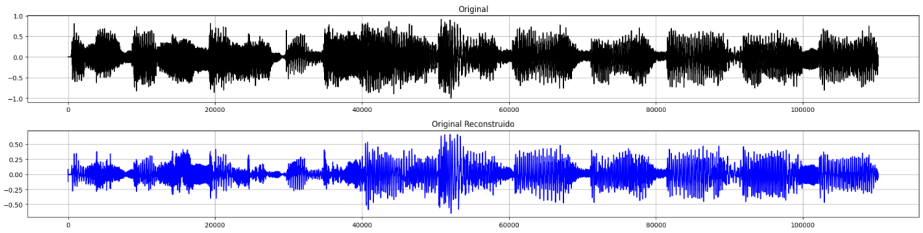
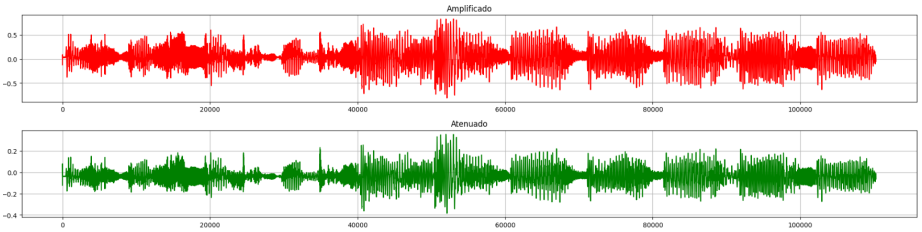


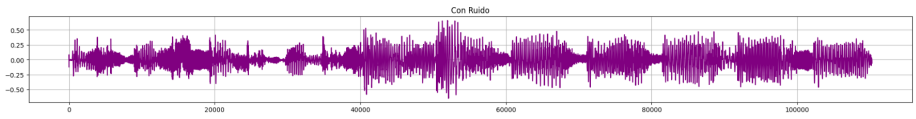
Figure 18: grafico original y autoencoder de la canción Osito Gominola

Las modificaciones realizadas incluyen:

- **Amplificación y Atenuación**: Multiplicamos el vector latente por factores como 1.5 (amplificación) o 0.5 (atenuación). Esto simula un cambio en la intensidad de las características del audio, parecido a subir o bajar el volumen.



- **Adición de Ruido**: Incorporamos un pequeño nivel de ruido gaussiano al espacio latente. Esta operación permite analizar la robustez del modelo y cómo responde a perturbaciones menores, generando una situación con interferencias.



- **Suavizado**: Aplicamos un suavizado mediante un *pooling* en el espacio latente para ver cómo se ven afectadas las transiciones bruscas entre

componentes del audio, generando versiones más “planas” del sonido reconstruido.

Estas transformaciones son posibles gracias a la naturaleza vectorial y continua del espacio latente, lo que facilita operaciones directas. Además, el autoencoder puede decodificar estas variaciones para generar salidas plausibles, ya que ha sido entrenado para mapear representaciones latentes a canciones o sonidos. Esto nos deja ver lo util y adaptable que son los autoencoders tanto para la manipulación de audio como para la experimentación para comprender cómo distintas modificaciones afectan las características percibidas de un sonido.

8 Conclusión

En este trabajo práctico, tuvimos que diseñar e implementar un autoencoder para la codificación y reconstrucción de canciones, lo que nos permitió explorar el uso de distintos modelos e hiperparámetros para el procesamiento de señales de audio. A lo largo del proyecto, aprendimos cómo modifica el tamaño del espacio latente la calidad de la reconstrucción y la importancia de las funciones de activación para capturar patrones complejos.

Además, evaluamos cómo diferentes configuraciones del modelo afectan el desempeño tanto en la reconstrucción del audio como en la clasificación de los distintos géneros de las canciones, lo que nos sirvió para experimentar con técnicas como clustering y reducción de dimensionalidad para el análisis de datos.

Finalmente, al trabajar con datos nuevos y manipular representaciones latentes para generar variaciones de audio, descubrimos la gran versatilidad de estas herramientas. Este trabajo nos ayudó a entender mejor el uso de redes neuronales, a analizar con mayor precisión los resultados obtenidos y a trabajar en torno a estos para mejorar nuestros modelos.