DESIGN OF AN IMPROVED MODEL FOR MUSIC SEQUENCE GENERATION USING CONDITIONAL VARIATIONAL AUTOENCODER AND CONDITIONAL GAN

PROYECTO 1 DE MODELOS GENERATIVOS PROFUNDOS

Stefanny Arboleda Ceferino
stefanny.arboleda@cimat.mx

Emmanuel A. Larralde Ortiz
emmanuel.larralde@cimat.mx

ABSTRACT

In this technical report, we present the results and challenges of trying to replicate [1]. In the original work, they claim they present a hybrid architecture capable of generating realistic musical samples with rich structures and subtle details while taking into account attributes such as genre, mood, or composer identity.

The hybrid architecture incorporates a Conditional Variational Autoencoder (CVAE) and a conditional Wassesrstein Generative Adversarial Network with Gradient Penalty (WGAN-GP) for musical samples generation from MIDI files. Meanwhile, the CVAE part (in a Seq2Seq architecture) is included to model conditioned latent distributions over music sequences and the genre attribute.

Their results show that this hybrid CVAE-GAN framework could not only significantly improve the quality of generated samples (in terms like FAD) but also allow for a better listening experience. However, while trying to build and train their CVAE-GAN, we faced several challenges because of the lack of details of their approach.

Link to our GitHub repository: https://github.com/StefannyAC/DGM-Project1

***K*eywords** Generación de Música · Wasserstein GAN · CVAE · Piano Roll

1. Introducción

Mientras que las imágenes digitales se representan con matrices de píxeles con valores en el rango de 8 bits, una pieza musical puede ser almacenada en un archivo MIDI (Musical Instrument Digital Interface). Un archivo MIDI almacena instrucciones para reproducir música, en lugar de audio real, como si fuera una partitura digital. Contiene información sobre notas, instrumentos, tempo, volumen, entre otras cosas.

Una representación musical similar a MIDI es la cinta de piano (*piano roll*). En el pasado, estas cintas se usaban para tocar un piano de forma automática, usando perforaciones que coincidían con las notas de los pianos. En el contexto de los archivos MIDI, un piano-roll es una matriz de tamaño $128 \times T$ que muestra una secuencia de notas musicales, donde los 128 renglones corresponden a los 128 tonos existentes en MIDI, las columnas corresponden al tiempo y el valor numérico a la velocidad. De esta forma, el elemento $pr_{i,j} = v$ indica que la nota i se tocará en el paso j y con una velocidad v (número entero entre 0 y 127).

Con una representación numérica como *piano roll*, es posible utilizar modelos de IA generativa para generar muestras a partir de un espacio latente de "baja dimensión"$Z \sim \mathcal{N}(0, I_d)$ a una secuencia de 128 tonos de una duración T. En este trabajo, una CVAE es utilizada para crear el espacio latente. Las representaciones en este son utilizadas como entradas de una cGAN (GAN condicionada). El generador del cGAN busca generar secuencias consistentes, mientras que el discriminador (o crítico) trata de no solo juzgar si las secuencias de sonidos son reales o no, sino también juzga si la secuencia generada es coherente con las condiciones.

No obstante, una GAN condicionada simple tiene problemas de convergencia. Puede que el gradiente se desvanezca y el modelo deje de mejorar rápidamente, o al contrario, explote y época tras época el modelo se vuelva errático o incluso produzca predicciones numéricamente inestables. También, es posible que el generador converja a una clase de muestras reales y pierda la habilidad de diversificar sus predicciones.

Para mitigar las complicaciones mencionadas, una *Wasserstein GAN with Gradient Penalty* (WGAN-CP) es incorporada para mejorar el rendimiento de convergencia. Además, se utiliza un enfoque de *Curriculum Learning* para facilitar el aprendizaje de secuencias más largas. El término *Curriculum Learning* quiere decir que gradualmente se escala la dificultad de la tarea en el proceso de entrenamiento, donde el proceso de entrenamiento inicia con patrones más sencillos y continúa progresivamente con patrones más complicados o largos.

2. Detalles de implementación

Para garantizar la reproducibilidad de los resultados presentados, en esta sección se detallan los detalles fundamentales del desarrollo. Primero, se describe la composición y el pre-procesamiento del conjunto de datos utilizado. Segundo, se expone el flujo de trabajo y la metodología de entrenamiento secuencial de tres etapas (CVAE, C-GAN acoplada e Híbrida). Finalmente, se presentan los hiperparámetros y configuraciones específicas que definieron el comportamiento de las arquitecturas y los optimizadores en cada etapa del proceso.

2.1. Conjunto de datos

Se utilizó el conjunto de datos *Lakh MIDI Dataset* (LMD), específicamente el subconjunto *Lakh MIDI Dataset Clean* ubicado en Kaggle. El subconjunto limpio contiene 176,581 archivos MIDI únicos, de los cuales 45,129 (LMD-matched) han sido emparejados con el conjunto de datos *The Million Song Dataset* (MSongsDB).

Mientras que LMD solo incluye archivos MIDI por artista y título de la canción, MSongsDB contiene metadatos que fueron copiados en LMD-matched. MSongsDB viene con código para leer los metadatos. Así, fue posible recuperar el género de algunas de las canciones de *Lakh MIDI Dataset Clean*. De hecho, se utilizó el atributo *mbtags* (*MusicBrainz tags*) para clasificar los archivos en uno de los 4 géneros.

No obstante, con el método descrito anteriormente, sólo fue posible extraer el género de aproximadamente el 10 % de los archivos MIDI, pues la mayoría de las etiquetas *mbtags* venían vacías. Las etiquetas faltantes fueron rellenadas por *Chat-GPT* (gpt-3.5-turbo, gpt-4.1-mini y gpt-4o-mini) utilizando la API oficial del *OpenAI*.

Finalmente, los archivos fueron pre-procesados utilizando el paquete de python pretty_midi, pero aproximadamente el 5 % de los archivos no pudieron ser leídos.

2.2. Flujo de Trabajo y Metodología de Entrenamiento

El entrenamiento del modelo híbrido propuesto, basado en el artículo de referencia, no se realiza de forma monolítica. En su lugar, se implementa un flujo de trabajo de tres etapas secuenciales diseñado para estabilizar el entrenamiento y guiar a los componentes del modelo hacia un espacio de características robusto.

Cada etapa utiliza los pesos generados por la anterior como punto de partida, permitiendo que cada componente (CVAE, Generador y Crítico) se especialice antes de ser ensamblado en la arquitectura final. A continuación, se detalla el propósito y la implementación de cada etapa.

2.2.1. Stage-1: Pre-entrenamiento de la CVAE

La primera etapa, implementada en el script `train_cvae.py`, se centra exclusivamente en entrenar el Autoencoder Variacional Condicional (CVAE).

- **Objetivo:** Entrenar el *encoder* para comprimir eficientemente la información del piano-roll (X), los eventos (E) y la condición (y) en un espacio latente z; y entrenar el *decoder* para reconstruir X a partir de z e y.
- **Función de Pérdida:** El modelo se optimiza minimizando la Evidencia del Límite Inferior (ELBO), definida como una combinación de la pérdida de reconstrucción y la divergencia de Kullback-Leibler (KL):
  $$L_{ELBO} = L_{recon} + \beta \cdot D_{KL}(q(z|X, E, y)||p(z))$$
  Donde L_{recon} se calcula mediante Entropía Cruzada Binaria (BCE) entre el X real y el X reconstruido, y β es un hiperparámetro que pondera el término de regularización KL.
- **Manejo de Desbalance:** Para mitigar el desbalance de clases (géneros musicales) en el dataset, la pérdida ELBO de cada muestra es ponderada utilizando el inverso de la frecuencia de su clase (calculado en `main` y aplicado en `train_cvae_epoch`).
- **Salida:** Esta etapa genera el archivo `checkpoints/cvae_pretrained_best.pth`, que contiene los pesos de la CVAE optimizada para la tarea de reconstrucción.

2.2.2. Stage-2: Pre-entrenamiento de la C-GAN Acoplada

La segunda etapa, implementada en `train_cgan.py`, introduce la Red Generativa Antagónica Condicional (C-GAN) y la acopla a la CVAE entrenada en la etapa anterior.

- **Objetivo:** Pre-entrenar el Generador (G) y el Crítico (D) de la C-GAN. A diferencia de una GAN estándar que tomaría ruido aleatorio como entrada, el Generador aquí aprende a generar secuencias a partir de los vectores latentes z producidos por la CVAE.
- **Arquitectura Acoplada:**
  1. Se carga la CVAE pre-entrenada (`cvae_pretrained_best.pth`).
  2. La CVAE se pone en modo de evaluación (`cvae.eval()`) y sus parámetros se **congelan** (`p.requires_grad_(False)`). No se actualiza en esta etapa.
  3. Para un lote de datos reales (X, E, y), se obtiene el vector latente $z = \text{CVAE}_{enc}(X, E, y)$.
  4. El Generador produce una muestra falsa: $X_{fake} = G(z, y)$
  5. El Crítico D aprende a diferenciar entre las tuplas (X, y) [reales] y (X_{fake}, y) [falsas].
- **Estabilización (WGAN-GP):** El entrenamiento utiliza la metodología de Wasserstein GAN con Penalización de Gradiente (WGAN-GP) para mayor estabilidad.
  - **Pérdida del Crítico (L_D):** Se optimiza para maximizar la distancia de Wasserstein, actualizándose 5 veces (`critic_iters`) por cada actualización del generador.
    $$L_D = \mathbb{E}[D(X_{fake}, y)] - \mathbb{E}[D(X, y)] + \lambda_{GP} \cdot \mathbb{E}[(||\nabla_{\hat{X}} D(\hat{X}, y)||_2 - 1)^2]$$
  - **Pérdida del Generador (L_G):** Se optimiza para minimizar la puntuación del crítico sobre muestras falsas.
    $$L_G = -\mathbb{E}[D(X_{fake}, y)]$$
- **Salida:** Se guardan los pesos pre-entrenados del generador y el crítico en `checkpoints/generator_pretrained_best.pth` y `checkpoints/critic_pretrained_best.pth`.

2.2.3. Stage-3: Entrenamiento Híbrido y *Curriculum Learning*

La etapa final, implementada en `train_hybrid.py`, une todos los componentes para el entrenamiento conjunto y definitivo del sistema híbrido.

- **Objetivo:** Afinar (fine-tune) la CVAE y el Generador simultáneamente, aprovechando tanto la capacidad reconstructiva de la CVAE como la capacidad generativa de la GAN.
- **Arquitectura Completa:**
  1. Se cargan los pesos pre-entrenados de CVAE (Stage-1) y, opcionalmente, de G y D (Stage-2).
  2. A diferencia de la Stage-2, la CVAE ahora se pone en modo de entrenamiento (`cvae.train()`) y sus pesos **se descongelan**.
  3. El entrenamiento del Crítico (D) se mantiene idéntico al de la Stage-2 (WGAN-GP).
  4. El entrenamiento de la CVAE y el Generador (G) se realiza de forma conjunta con una **pérdida híbrida**:

     $$L_{Hyb} = \alpha \cdot L_{ELBO} + \gamma \cdot L_G$$

     Donde L_{ELBO} es la pérdida de la CVAE (calculada ahora con MSE en lugar de BCE) y L_G es la pérdida adversaria del generador. Los parámetros α y γ balancean la importancia de la reconstrucción (fidelidad) vs. la generación (realismo).
- **Curriculum Learning:** Una característica de implementación clave en esta etapa es el uso de *Curriculum Learning* sobre la longitud de la secuencia (T). El entrenamiento no se realiza directamente sobre la longitud final, sino en fases:
  1. El modelo se entrena inicialmente con secuencias cortas ($T = 32$).
  2. Los pesos aprendidos se transfieren a un nuevo modelo reconfigurado para secuencias más largas ($T = 64$).
  3. El proceso se repite, transfiriendo los pesos del modelo de $T = 64$ a la configuración final de $T = 128$.

  Esta técnica permite al modelo aprender primero patrones locales simples antes de abordar la complejidad de dependencias temporales largas, pretendiendo así mejorar significativamente la estabilidad y el resultado final.

2.3. Configuración Experimental

Para asegurar la reproducibilidad del estudio, a continuación se detallan los hiperparámetros y configuraciones específicas utilizadas en cada etapa del entrenamiento.

2.3.1. Parámetros de la Arquitectura

Las dimensiones clave de los modelos fueron:

- **Dimensión Latente (z_{dim}):** 32
- **Número de Condiciones ($cond\_dim$):** 4 (correspondientes a los géneros musicales)
- **Embedding de Eventos (CVAE):** 16
- **Embedding de Condición (CVAE):** 4
- **Dimensiones Ocultas (CVAE):** 16 (enc_hid) y 16 (dec_hid)
- **Dimensiones Ocultas (C-GAN):** 32 (hidden_dim)

2.3.2. Parámetros de Entrenamiento

Entrenamiento General:

- **Optimizadores (CVAE):** Adam con tasa de aprendizaje de 5×10^{-5}.
- **Optimizadores (C-GAN y G/D Híbrido):** Adam con tasa de aprendizaje de 1×10^{-4} y $\beta = (0{,}5, 0{,}999)$.
- **Tamaño de Lote (Batch Size):** 512

Parámetros Específicos por Etapa:

- **Stage-1 (CVAE):**
  - β (Peso de KL): 0.05, *Se utiliza un valor bajo para priorizar la calidad de la reconstrucción sobre la regularización del espacio latente en esta etapa inicial.*
  - Épocas: 10
  - Longitud de Secuencia (fija): 32
- **Stage-2 (C-GAN Acoplada):**
  - λ_{GP} (Penalización de Gradiente): 10.0
  - $critic\_iters$ (Iteraciones del Crítico): 5
  - Épocas: 20
  - Longitud de Secuencia (fija): 32

  Se adoptan estos valores siguiendo las recomendaciones del artículo original de WGAN-GP [2], donde demostraron ser robustos para estabilizar el entrenamiento antagónico.
- **Stage-3 (Híbrido con Curriculum Learning):**
  - α (Peso de ELBO): 1.0
  - γ (Peso de L_G): 0.5
  - β (Peso de KL en ELBO): 0.01, *Se reduce aún más para minimizar la restricción sobre el encoder, permitiendo que la L_G guíe la estructura del espacio latente.*
  - $critic\_iters$: 5
  - λ_{GP}: 10.0
  - Currículum (T y Épocas): (T=32, 10 épocas) $\rightarrow$ (T=64, 10 épocas) $\rightarrow$ (T=128, 15 épocas)

  Los pesos $\alpha = 1{,}0$ y $\gamma = 0{,}5$ se eligieron para balancear la pérdida de reconstrucción (fidelidad) con la pérdida adversaria (realismo). Se da prioridad a la ELBO para asegurar que el generador no se desvíe de la información de entrada (colapso de modo).

3. Métricas de Evaluación

Para evaluar de forma integral el rendimiento del modelo híbrido, se utilizaron tres métricas distintas. El Error Cuadrático Medio (MSE) y la Evidencia del Límite Inferior (ELBO) se emplearon para cuantificar la calidad de la reconstrucción y el ajuste del espacio latente de la CVAE. Para evaluar la calidad perceptual y el realismo de las muestras generadas por el sistema completo, se utilizó la Distancia Fréchet de Audio (FAD).

3.1. Error Cuadrático Medio (MSE)

Descripción Conceptual. El Error Cuadrático Medio es una métrica de fidelidad que mide la diferencia promedio, píxel a píxel (o, en este caso, nota a nota), entre la secuencia de entrada X y la secuencia reconstruida $\hat{X}$. Es útil para cuantificar la capacidad del autoencoder para preservar la información original. Un MSE más bajo indica una reconstrucción más precisa.

Implementación. La evaluación del MSE se implementa en `eval_mse.py`.

- Se utiliza el conjunto de datos de `test`.
- Se carga el componente CVAE (ej. `hybrid_cvae_best_T128.pth`) entrenado durante la Stage-3 (híbrida).
- **Detalle clave:** Para aislar la calidad del *encoder* y el *decoder* sin la influencia de errores acumulados en la generación, la reconstrucción se realiza con **teacher forcing**. Es decir, el decoder recibe la secuencia real X como entrada en cada paso de tiempo (teacher=X), en lugar de su propia salida anterior.
- El MSE se calcula por muestra y luego se agregan la media y la desviación estándar por género y de forma global (genre_id = -1), guardándose en `metrics/eval_mse_T{T}.csv`, donde el último T corresponde a la `seq_len` que se esté evaluando, por defecto será 128.

3.2. Evidencia del Límite Inferior (ELBO)

Descripción Conceptual. La ELBO es la función objetivo fundamental de una VAE. Consta de dos términos: (1) la **pérdida de reconstrucción**, que fomenta la fidelidad (como el MSE), y (2) la **divergencia Kullback-Leibler (KL)**, un término de regularización que fuerza al espacio latente a seguir una distribución conocida (típicamente una Gaussiana estándar). Evaluar la ELBO en el conjunto de test nos indica qué tan bien el modelo generaliza, balanceando la precisión de la reconstrucción con la estructura de su espacio latente. Un valor de ELBO más alto (o un negativo de ELBO más bajo) es mejor.

Implementación. El script `eval_elbo.py` calcula la ELBO sobre el conjunto de `test`.

- Utiliza la CVAE entrenada en la Stage-3 (híbrida).
- A diferencia del script de MSE, este script **no** utiliza teacher forcing en el decoder para el cálculo de la reconstrucción.
- La pérdida de reconstrucción se calcula usando `F.mse_loss` (alineado con la Stage-3) y la divergencia KL se calcula entre la salida del encoder y la prior $N(0, I)$.
- La métrica final se reporta como $ELBO = -(L_{recon} + \beta \cdot D_{KL})$.
- Los resultados ('Recon_Loss_mean', 'KL_Div_mean', 'ELBO_mean') se desglosan por género y de forma global, guardándose en `metrics/eval_elbo_T{T}.csv`, donde el último T corresponde a la `seq_len` que se esté evaluando, por defecto será 128.

3.3. Distancia Fréchet de Audio (FAD)

Descripción Conceptual. La FAD es una métrica de calidad perceptual diseñada para evaluar modelos generativos de audio. Es análoga a la Distancia Fréchet de Inception (FID) usada en imágenes. La FAD mide la similitud entre dos distribuciones de audio (la de los datos reales y la de los datos generados) en un espacio de características (embeddings) profundo.

En lugar de comparar muestras 1-a-1, la FAD compara la media y la covarianza de las dos distribuciones. Un valor de FAD más bajo indica que la distribución de las muestras generadas es perceptualmente más similar a la distribución de las muestras reales.

Implementación. La evaluación de la FAD se detalla en `eval_fad_samples.py` y representa la prueba más completa del sistema híbrido.

- Se cargan tanto la **CVAE** como el **Generador** (G) de los checkpoints de la Stage-3 (híbrida).

- **Generación de Muestras Reales:** Se toman piano-rolls del conjunto de `test`, se convierten a objetos PrettyMIDI ('pianoroll_to_pretty_midi') y se renderizan a archivos WAV usando un sintetizador ('render_midi_to_wav').
- **Generación de Muestras Falsas:** Este es el paso crucial. Para probar la capacidad generativa real del modelo, se muestrea un vector latente z directamente de una distribución Gaussiana estándar ('z = torch.randn(...)'). Este z aleatorio (junto con una condición y) se alimenta al Generador G para crear el piano-roll falso ('Xg = gen(z, y)'), que luego se convierte a WAV.
- **Extracción de Embeddings:** Se utiliza un modelo **Audio Spectrogram Transformer (AST)** pre-entrenado para calcular los embeddings de todos los archivos WAV (reales y falsos).
- **Cálculo de FAD:** Finalmente, se calcula la distancia Fréchet ('frechet_distance') entre las estadísticas (media y covarianza) de los embeddings reales y falsos. Este cálculo se realiza tanto por género como de forma global, guardándose en `metrics/fad_results.csv`, el cual contiene los resultados para todas las etapas del curriculum learning.

4. Resultados

En esta sección se presentan y analizan los resultados cuantitativos obtenidos tras aplicar el flujo de entrenamiento de tres etapas. La evaluación se basa en las métricas definidas en la sección anterior, con el objetivo de medir tanto la calidad de la generación perceptual como la fidelidad de la reconstrucción del modelo.

El análisis se estructura de la siguiente manera:

- Primero, se presentan los resultados de la Distancia Fréchet de Audio (FAD). Esta métrica se utiliza para comparar el realismo de las muestras generadas a través de las diferentes etapas del *Curriculum Learning* (longitud de secuencia 32, 64 y 128), permitiendo identificar el impacto de la longitud de la secuencia en la calidad generativa.
- Segundo, se evalúa el rendimiento del componente CVAE del modelo híbrido final (T=128) mediante el Error Cuadrático Medio (MSE) y la Evidencia del Límite Inferior (ELBO). Estas métricas nos permiten cuantificar la precisión de la reconstrucción y la calidad del espacio latente aprendido.

Este doble enfoque permite un diagnóstico completo, contrastando la capacidad de generación de muestras nuevas (FAD) con la capacidad de codificación y reconstrucción de datos existentes (MSE/ELBO). Adicionalmente, se enfatiza que los resultados a mostrarse fueron ejecutados en una GPU `NVIDIA GeForce RTX 5060 Ti`, en un entorno de PyTorch con soporte para CUDA; y se pueden encontrar los códigos utilizados para cada etapa en de nuestro repositorio de GitHub.

4.1. Resultados cuantitativos

La Tabla 1 y la Figura 1 resume los resultados por género y longitud de secuencia. Se observa:

- **Tendencia con la longitud**: el mejor *Overall* para nuestro modelo se alcanza en **Seq_len=64** (6.58), superando a 32 (6.79) y 128 (11.02). Esto sugiere que 64 equilibra bien la coherencia temporal y la estabilidad adversaria; con 128 aparece degradación sugiriendo quizás que el modelo necesite entrenarse por más épocas.
- **Por género**: con Seq_len=64 se obtienen los mejores valores en *Classical* (8.11), *Jazz* (7.72) y *Pop* (7.93), mientras que *Rock* favorece **Seq_len=32** (7.22). Esto indica distinta sensibilidad de los géneros a la ventana temporal efectiva.
- Se reportan también los valores de [1]. Dado que el pre-procesado, los extractores de embedding y los *splits* pueden diferir, tomamos esa fila como referencia y no como comparación directa uno-a-uno.

Tabla 1: Resultados por género y longitud de secuencia

Modelo	Seq_len	Overall	Classical	Jazz	Pop	Rock
Nuestro	32	6.79	10.34	9.03	10.06	7.22
	64	6.58	8.11	7.72	7.93	9.78
	128	11.02	14.54	12.95	11.20	12.90
[1]	–	1.85	1.8	1.9	1.7	2.0

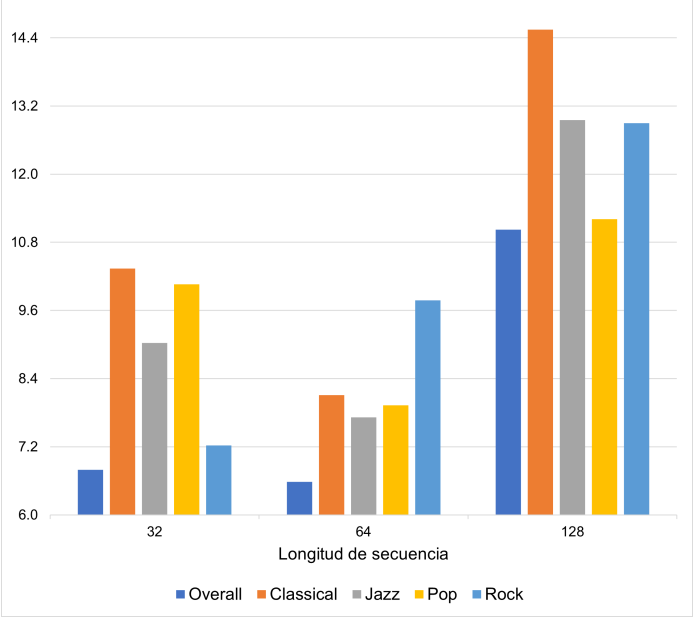

Figura 1: Comparaciones Fréchet Audio Distance (FAD)

Además del realismo generativo (FAD), se evaluó la fidelidad de la reconstrucción del componente CVAE (entrenado en la Stage-3) usando el Error Cuadrático Medio (MSE). La Tabla 2 resume estos hallazgos para la longitud de secuencia final ($T = 128$), medidos con *teacher forcing* como se describe en la sección de métricas.

Se observa un MSE global (Overall) muy bajo, de **0.018**, lo que indica que el autoencoder aprendió a codificar y decodificar las secuencias con alta fidelidad. A nivel de género, 'Jazz' (**0.016**) y 'Rock' (**0.017**) presentan la reconstrucción más fiel (el menor error). 'Classical' (**0.020**), en contraste, muestra el error más alto, lo que sugiere que sus estructuras armónicas o temporales fueron marginalmente más difíciles de comprimir y reconstruir sin pérdidas.

Finalmente, la Tabla 3 desglosa los componentes de la Evidencia del Límite Inferior (ELBO) en el conjunto de prueba para $T = 128$. Esta métrica nos permite inspeccionar el balance entre la reconstrucción y la regularización del espacio latente.

Como se observa en la fila 'Overall', el valor de la ELBO (Media) de **-295.58** está dominado casi en su totalidad por el término de 'Recon. Loss' (**294.65**). La contribución del término de 'KL Div' (92.27) es significativamente atenuada por el hiperparámetro $\beta = 0{,}01$ (utilizado en la evaluación para coincidir con el entrenamiento). En este punto, la conclusión es clara: la pérdida está gobernada por la reconstrucción.

Tabla 2: Resultados de MSE por género (T=128) usando la CVAE Híbrida

Género	MSE (Media)	MSE (Std. Dev.)
Classical	0.020	0.009
Jazz	0.016	0.009
Rock	0.017	0.008
Pop	0.019	0.008
Overall	0.018	0.008

Tabla 3: Componentes de la ELBO por género (T=128)

Género (ID)	Recon. Loss (Media)	KL Div (Media)	ELBO (Media)
Classical (0)	329.58	93.60	-330.52
Jazz (1)	256.27	83.61	-257.11
Rock (2)	284.69	93.36	-285.62
Pop (3)	305.12	91.96	-306.04
Overall (-1)	294.65	92.27	-295.58

Este resultado es consistente con la configuración del entrenamiento híbrido (Stage-3), donde se utilizó un β pequeño precisamente para relajar la restricción sobre el espacio latente y permitir que la pérdida adversaria L_G ayudara a estructurarlo, priorizando la calidad de la reconstrucción.

4.1.1. Análisis cuantitativo general

En conjunto, las tres métricas reportadas ofrecen un panorama completo. Los resultados de MSE y ELBO demuestran que el componente CVAE del modelo híbrido es funcional y robusto: aprendió a codificar y reconstruir las secuencias de manera efectiva, logrando una alta fidelidad (bajo MSE de 0.018) al tiempo que mantiene un espacio latente estructurado (KL baja y controlada).

Sin embargo, la métrica FAD revela una historia más compleja. El mejor resultado generativo (FAD de 6.58 en $T = 64$) no coincide con la longitud final del currículum ($T = 128$, FAD de 11.02). Esto sugiere que si bien el modelo puede **reconstruir** secuencias largas (como lo demuestra el excelente MSE en $T = 128$), tiene dificultades para **generar** secuencias nuevas y coherentes a esa misma longitud cuando se muestrea un vector z aleatorio. Este desacoplamiento entre la capacidad de reconstrucción y la de generación es un desafío conocido en los modelos híbridos, y apunta a que la componente adversaria del entrenamiento en $T = 128$ pudo haber sido insuficiente o inestable, como se sugirió anteriormente.

4.2. Análisis de las representaciones

En esta sección se explora de forma cualitativa las representaciones en estado latente generadas por los modelos entrenados. Principalmente, se espera que los grupos, dados por el género musical, no estén muy condensados en una sola región, que exista una transición suave entre muestras y, preferentemente, que existan estructuras separables por clase.

Al aplicar la técnica de reducción de dimensione T-SNE (con perplexity = 5) a un total de 10,000 muestras del conjunto de entrenamiento, se observa (Figura 2) que ni el CVAE entrenado de forma individual ni el entrenado de forma conjunta con *curriculum learning* generan variables en el espacio latente Z con estructuras separables en el espacio reducido con T-SNE. Esto indica que probablemente el modelo no es capaz de generar secuencias de tonos con ritmos diferentes. No obstante, por lo menos se nota que no hay un colapso de las clases, aunque pareciera que el modelo entrenado con más épocas sigue una tendencia a colapsar las clases.

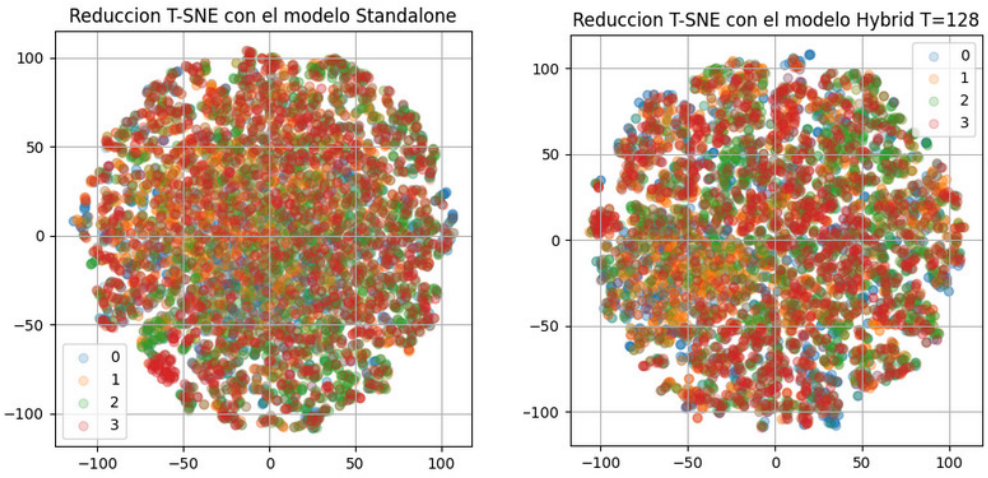

Figura 2: Reducción con T-SNE de los espacios latentes generados por la CVAE con muestras del conjunto de entrenamiento.

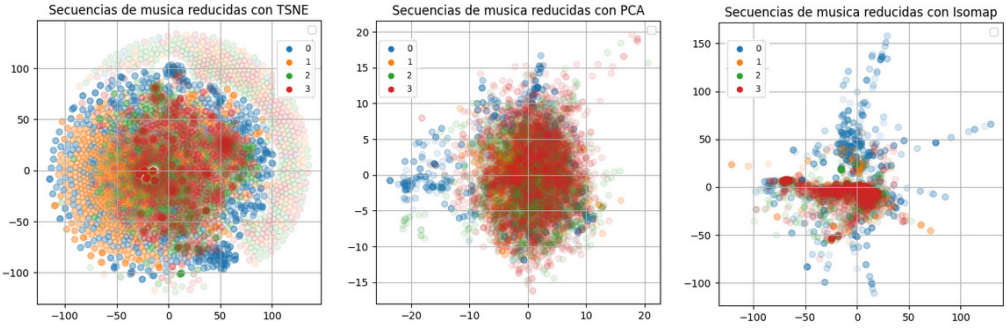

Figura 3: Resultados de aplicar PCA, T-SNE e Isomap (usando los valores pre-establecidos de *scikit-learn*) a las secuencias de tonos de 10,000 muestras del conjunto de entrenamiento. Ninguno de los dos métodos genera representaciones con estructuras separables por género.

Otra pregunta que puede hacerse sobre las representaciones del modelo híbrido es sobre la capacidad del generador de representar las diferentes clases. Sin embargo, es importante preguntarse si las secuencias muestran estructuras fácilmente separables. Encontramos que, con los métodos de reducción de dimensión populares, no es posible encontrar dichas estructuras (véase Figura 3). Entonces, no se esperaría que el modelo sea capaz de generar secuencias con tal estructura si los datos no lo tienen.

Con los resultados cualitativos presentados en esta sección, parece que la secuencia de tonos de MIDI podría no ser una buena representación del ritmo ni el género musical de las muestras, pues los instrumentos, las distorsiones, la producción y (sobre todo) la letra, son fundamentales para producir música de un género. Así, el CVAE tampoco genera un espacio latente con claras distinciones de las condiciones, ni el generador puede hacerlo. Otra prueba que puede hacerse es reemplazar las etiquetas de género por una lista aleatoria y observar si existe un impacto significativo en el rendimiento de los modelos.

5. Conclusiones

En este trabajo se implementó y evaluó un modelo generativo híbrido, combinando un Autoencoder Variacional Condicional (CVAE) con una Red Generativa Antagónica Condicional (C-GAN) estabilizada con WGAN-GP, para la generación de secuencias musicales de piano-roll condicionadas por género. El entrenamiento se abordó mediante una estrategia de tres etapas, culminando en un entrenamiento conjunto con *Curriculum Learning* para manejar secuencias de longitud creciente (T=32 a T=128).

El hallazgo principal de la evaluación cuantitativa es un claro desacoplamiento entre la capacidad de reconstrucción y la de generación del modelo. Por un lado, el componente CVAE del modelo final (T=128) demostró una excelente fidelidad de reconstrucción, alcanzando un bajo Error Cuadrático Medio global de **0.018** (Tabla 2). Los resultados de la ELBO (Tabla 3) confirmaron que el modelo funciona según lo diseñado, con una pérdida dominada por la reconstrucción y una contribución de KL controlada por el hiperparámetro β.

Por otro lado, la métrica de calidad perceptual (FAD) reveló que la capacidad de *generación* (muestreando z de forma aleatoria) no mejoró con la longitud final del currículum. El modelo alcanzó su FAD óptimo en **T=64** (FAD de 6.58), pero esta calidad se degradó significativamente en T=128 (FAD de 11.02), como se muestra en la Tabla 1.

La explicación a esta discrepancia se encuentra en el análisis cualitativo de las representaciones. El análisis T-SNE del espacio latente Z (Figura 2) no mostró una separación de clases clara, lo cual indica que el encoder no aprendió a agrupar los géneros. Más importante aún, el análisis T-SNE sobre los datos de entrada (Figura 3) demostró que los propios piano-rolls de entrada no presentan una estructura claramente separable por género.

De esto se concluye que la limitación fundamental no reside únicamente en la arquitectura del modelo, sino en la insuficiencia de la representación de piano-roll para la tarea de clasificación y generación condicionada por género. Atributos cruciales como la instrumentación, el timbre, la producción y la letra, que definen un género, están ausentes. El modelo no puede aprender a generar distinciones que no son evidentes en los datos de entrada.

5.1. Trabajo Futuro

Basado en estas conclusiones, se proponen varias líneas de trabajo futuro:

- **Validar la Condición:** Realizar el experimento sugerido en la sección anterior, re-entrenando el modelo con etiquetas de género aleatorias. Si las métricas (FAD, ELBO) no se degradan significativamente, se confirmaría que el modelo ignora la condición de género, ya que esta no aporta información relevante desde la representación de piano-roll.
- **Optimizar el Currículum:** Investigar la degradación de FAD en T=128. Es posible que esta etapa final requiera un mayor número de épocas o un ajuste de los pesos de la pérdida híbrida (α y γ) para estabilizar el entrenamiento antagónico en secuencias largas.

Referencias

[1] Pallavi Ganorkar and Anagha Rathkanthiwar. Design of an improved model for music sequence generation using conditional variational autoencoder and conditional gan. In *2024 2nd DMIHER International Conference on Artificial Intelligence in Healthcare, Education and Industry (IDICAIEI)*, pages 1–4, 2024.

[2] Ishaan Gulrajani, Faruk Ahmed, Martin Arjovsky, Vincent Dumoulin, and Aaron C Courville. Improved training of wasserstein gans. In I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 30, pages 5767–5777. Curran Associates, Inc., 2017.

Appendices

A. Modelos entrenados

Puedes cargar cualquiera de nuestros modelos pre-entrenados. La dimensión latente es de tamaño 32.

```
## CVAE
#cvae pre-entrenado por si solo. Se busca minimizar ELBO
cvae = torch.hub.load('StefannyAC/DGM-Project1', 'cvae_standalone')

#cvae entrenado conjuntamente en el modelo híbrido.
cvae = torch.hub.load('StefannyAC/DGM-Project1', 'cvae_hybrid', seq_len=seq_len)


## Generador
# Generador del WGAN entrenado usando un CVAE congelado.
generator = torch.hub.load('StefannyAC/DGM-Project1', 'generator_standalone')

# Generador del WGAN entrenado conjuntamente en el modelo híbrido
generator = torch.hub.load(
    'StefannyAC/DGM-Project1', 'generator_hybrid', seq_len=seq_len
)


## Crítico
# Crítico del WGAN entrenado usando un CVAE congelado.
critic = torch.hub.load('StefannyAC/DGM-Project1', 'critic_standalone')

# Crítico del WGAN entrenado conjuntamente en el modelo híbrido
critic = torch.hub.load(
    'StefannyAC/DGM-Project1', 'critic_hybrid', seq_len=seq_len
)
```