Universidad Internacional de Valencia

Generación de Música Simbólica Mediante Emociones

Titulación:
Máster en Inteligencia Artificial
Curso académico
2024 – 2025

Alumno/a:
Soto Medina, Luis
D.N.I: 77139606K

Director/a de TFM: César Augusto Guzmán Álvarez

Convocatoria:
Tercera

7 de abril de 2025

De:
Planeta Formación y Universidades

Agradecimientos

Con este proyecto termina un periodo intenso, de continuo aprendizaje y renuncia. Por ello, no puedo finalizar sin agradecer.

En primer lugar, a mi familia, Juan Ignacio, Rosa, Ignacio y Elena, por ser mi apoyo incondicional, por soportarme y escucharme, por el tiempo que no hemos podido estar juntos. Sois mi acimut.

A mi tutor, César Augusto Guzmán, por haberme dado la oportunidad, escucharme y acompañado.

A mis amigos, Carlos, Adriana, Pablo, Javier, Luis, Rafa, ... por estar siempre que os necesitaba. Como dice la canción: "Mis amigos suman más que mis demonios".

Por último, a los profesores de este máster, por dedicarnos vuestro tiempo y conocimiento.

Gracias

Índice general

Índice de figuras

Índice de tablas

Lista de Acrónimos

CNN *Convolutional Neural Network*.

CPWord *Compound Word Tokenize*.

GAN Red Generativa Antagónica o *Generative Adversarial Network*.

LDM *Latent Diffusion Model*.

LMD-full *The Lakh MIDI Dataset*.

LSTM *Long Short-Term Memory*.

LLM *Large Language Model*.

MFCC *Mel-Frequency Cepstral Co-efficient*.

MIDI Musical Instrument Digital Interface.

MMD MetaMIDI.

NLP *Natural Language Processing*.

RAG *Retrieval-Augmented Generation*.

REMI *Revamped MIDI*.

UPF Universidad Pompeu Fabra.

VAE *Autoencoder* variacionales o *Variational AutoEncoders*.

Resumen

La música generativa se define como aquella creada mediante algoritmos capaces de producir música de manera autónoma. En los últimos años, esta tecnología se ha convertido en una herramienta para músicos y compositores en su proceso creativo. En este contexto, el presente trabajo final de máster desarrolla un modelo de aprendizaje profundo capaz de generar música simbólica, adaptando la creación según los cuadrantes del modelo de las emociones de Russell. Este proyecto utiliza un flujo de trabajo estructurado basado en *machine learning pipeline*, mediante con el que se realiza un proceso que abarca, desde la creación de un conjunto de datos MIDI etiquetados con emociones hasta su posterior análisis y procesamiento, lo que incluye la extracción de las principales características musicales necesarias para alimentar el modelo. De este modo, se ha diseñado un modelo basado en un *transformer* autoregresivo basado en *decoder*, entrenando y evaluando su efectividad mediante métricas tanto objetivas como subjetivas, con el objetivo de generar música condicionada por emociones.
Además de todo esto, este trabajo profundiza en las implicaciones éticas y legales de la música generativa mediante inteligencia artificial, abordando aspectos como la autoría, la creatividad y originalidad, los derechos de autor y la apropiación cultural.

Abstract

Generative music is defined as music created through algorithms capable of producing compositions autonomously. In recent years, this technology has become a valuable tool for musicians and composers in their creative process. In this context, the present Master's Thesis develops a deep learning model capable of generating symbolic music, adapting its creation according to the quadrants of Russell's model of emotions.

In this project, following a workflow based on a machine learning pipeline, the process spans from the creation of a MIDI dataset labeled with emotions to its analysis and processing, extracting the key musical features needed to train the model. Additionally, an autoregressive transformer model based on a decoder is designed, trained, and evaluated using both objective and subjective metrics to generate music conditioned by emotions.

Additionally, this study explores the ethical and legal implications of AI-generated music, addressing issues such as authorship, creativity and originality, copyright, and cultural appropriation, among others.

Introducción 1

La palabra música, proviene del griego y significa "arte de las musas". Se define como el arte de crear y organizar sonidos y silencios a través de principios como la melodía, la armonía y el ritmo. A lo largo de la historia este concepto ha ido evolucionando, adaptándose a cambios culturales y tecnológicos, pero siempre se ha mantenido como una de las manifestaciones humanas más universales y poderosas. Como indica Byrne (2012), la música no solo es un producto artístico, sino un reflejo del contexto en el que se crea y se percibe. Su impacto va más allá de lo percibido auditivamente, ya que influye en nuestras emociones, estado de ánimo e incluso en nuestra percepción del tiempo y el espacio.

Las emociones siempre han estado relacionadas con la música. En los primeros pasos de la educación musical se enseña a identificar las emociones que transmite la música, desde la alegría de una tonalidad mayor hasta la tristeza o melancolía de un modo menor. Sin embargo, la tonalidad no es el único elemento que influye en la percepción emocional de una pieza musical. Otros elementos musicales como el *tempo*, la dinámica, la textura y la armonía tienen un papel fundamental en la expresión emocional de una pieza. Un *tempo* rápido suele estar asociado con sensaciones de energía y entusiasmo, mientras que un *tempo* lento suele provocar calma o tristeza. La concentración de notas y su disposición en el registro melódico también afecta en la percepción emocional. Las melodías con frases largas y notas sostenidas tienden a generar sensaciones de tranquilidad, mientras que aquellas con notas breves y articulaciones marcadas suelen provocar excitación.

Esta relación entre música y emoción no sólo se ha explorado en el ámbito artístico, sino que también se han encontrado aplicaciones en campos como la psicología, la terapia y más recientemente, en la inteligencia artificial.

En el campo de la psicología, por ejemplo, la clasificación e identificación de emociones es un área clave de estudio. Desde los primeros estudios se han distinguido entre emociones agradables, como el amor (asociado con la felicidad o la alegría) y emociones desagradables, como la ira (relacionada con la furia o el enfado). Sin embargo, estas emociones básicas engloban otras tantas emociones menores. Por ello, surge la necesidad de establecer modelos que expliquen la diversidad de expresiones emocionales como combinación de emociones básicas. Entre estos modelos, los enfoques "circumplejos" han ocupado la mayor atención, organizando las emociones en un círculo polar con extremos opuestos.

El modelo más extendido es la teoría cincumpleja de las emociones de Russell (1980), que distribuye las emociones en un espacio bidimensional formado por los ejes *valence* y *arousal*. El

eje de *valence* representa el placer o desagrado que una emoción genera. Por el contrario, el eje *arousal* muestra el nivel de activación fisiológica, desde la calma hasta la excitación. Gracias a este modelo de representación, se pueden ubicar las emociones en un espacio continuo, como la felicidad (alto *valence* y *arousal*) o la tristeza (bajo *valence* y *arousal*) (Figura 1.1).

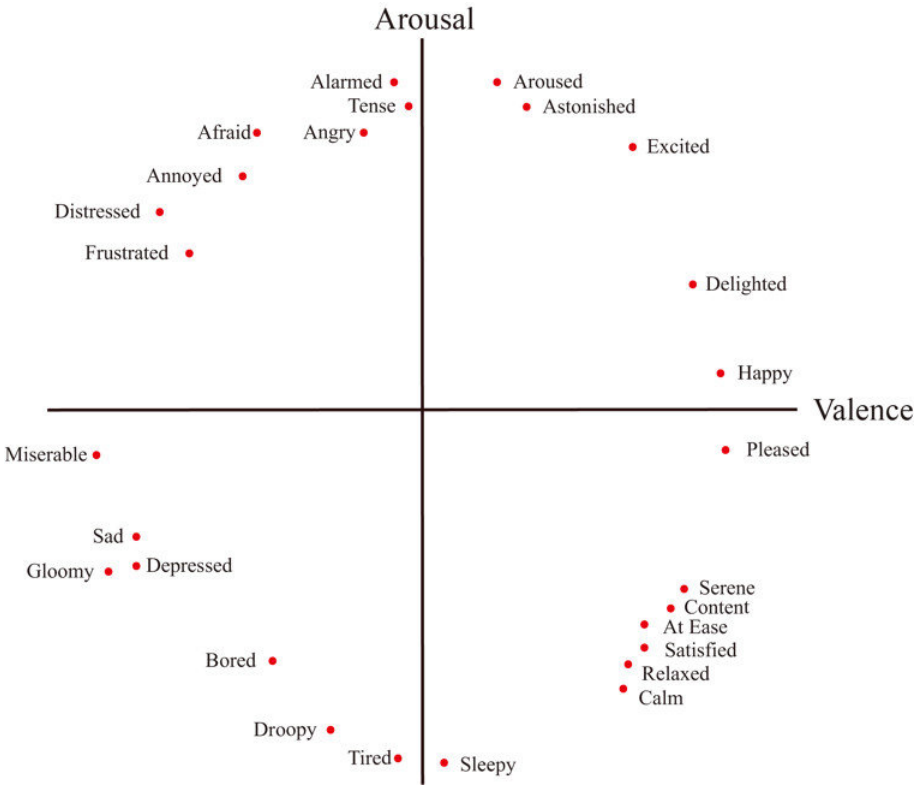

Figura 1.1: *Modelo circumplejo de emociones de Russell.*

La capacidad de la música para generar emociones ha sido un factor clave en la evolución de la música generativa. El término *generative music* de Eno (1995) establece que es la música siempre diferente y cambiante, creada por un sistema. Aunque la música generada por inteligencia artificial vive actualmente un momento de auge y se presenta como una innovación relevante para el público general, la generación musical automatizada existe desde hace siglos en diversas formas. Desde métodos pre-computacional basados en procesos aleatorios, como el *Musikalisches Würfelspiel* o "juego de los dados" en el siglo XVIII (Wikipedia contributors, 2025), hasta la reciente explosión de la música generativa impulsada por el aprendizaje automático y los avances en la capacidad computacional.
En este contexto, Velardo (2023) identifica cinco etapas históricas en el desarrollo de la música generativa.

1. Pre-informática (1700-1956)

2. Académica (1957-2009)

3. Primera oleada de nuevas empresas (2010-2016)

4. Grandes experimentos tecnológicos (2017-2022)

5. Éxito de la música generada por IA (2023-presente)

Mi interés por la interacción entre la música y la inteligencia artificial ha sido el germen del presente proyecto, que busca profundizar en la relación entre música y emociones mediante técnicas de inteligencia artificial, con el objetivo de generar música que refleje estados emocionales concretos.
Al inicio de este proyecto, tuve la oportunidad de asistir al *Generative Music AI Workshop*, organizado por la Universitat Pompeu Fabra (UPF) y *The Sound of AI* del 16 al 20 de diciembre de 2024. Esta experiencia me sirvió para profundizar en la música generativa, aplicando los conocimientos adquiridos durante el máster en el campo de la música. Dicho curso ha sido una fuente de inspiración y un punto de partida para este proyecto.

Figura 1.2: *Generative Music AI Workshop organizado por la UPF y The Sound of AI (diciembre de 2024).*

Objetivos

2

2.1. Objetivo general

El objetivo de este proyecto es desarrollar un generador de música simbólica adaptada a emociones y sentimientos específicos.

2.2. Objetivos específicos

Con objeto de profundizar en el tema de este trabajo fin de máster y para poder desarrollar el objetivo general, se establecen los siguientes objetivos específicos:

1. Revisar el estado del arte de la generación musical automatizada.

2. Estudiar las implicaciones éticas y legales de la creación musical generada por inteligencia artificial.

3. Crear un conjunto de datos que vincule música con emociones específicas, permitiendo entrenar un modelo de inteligencia artificial.

4. Diseñar una arquitectura que sea capaz de generar composiciones musicales coherentes y novedosas que expresen sentimientos definidos.

5. Desarrollar entrenamientos que ajusten la arquitectura para obtener resultados sólidos.

6. Evaluar el modelo generado en base a su capacidad para transmitir emociones y sentimientos, así como analizar su coherencia y originalidad.

Estado del Arte 3

A continuación, se presentan los antecedentes de la música generativa, haciendo especial hincapié en aquellas que están condicionadas por las emociones. En primer lugar, se revisan los estudios previos sobre esta temática, destacando las metodologías, modelos y conjuntos de datos utilizados. A continuación, se presentan los conjuntos de datos más significativos, analizando su distribución y principales características. Finalmente, se analizan las implicaciones éticas y legales asociadas a la música generada por IA, incluyendo aspectos objeto de debate, como la autoría, la propiedad intelectual y los diferentes dilemas éticos relacionados con la automatización del proceso musical y la apropiación cultural.

3.1. Trabajos relacionados

En el campo de la música generativa se distinguen dos enfoques principales, la música simbólica, la cual se refiere a la creación de música mediante notación simbólica tradicional como partituras o a través de MIDI (Musical Instrument Digital Interface) y la música de audio que supone la creación de sonidos reales almacenados en archivos de audio como WAV, MP3 u otros formatos de audio.
En el campo de la **generación de música simbólica** destacan los siguientes trabajos.

- **DeepBach** (Hadjeres et al., 2017) es un modelo gráfico destinado a crear música polifónica y, más concretamente, piezas de tipo himno. Este modelo incorpora una herramienta de guía de la composición, permitiendo al usuario limitar la generación mediante restricciones posicionales como notas, ritmos o cadencias en la partitura generada.

- **MuseGAN** (Dong et al., 2017) es un modelo basado en la Red Generativa Antagónica o *Generative Adversarial Network* (GAN), diseñado para generar música simbólica en formato multipista. Este modelo permite la composición de pistas independientes o vinculadas a un generador común, así como la combinación de ambos enfoques, aumentando las posibilidades de creación de música simbólica en múltiples capas.

- **Magenta** (Huang et al., 2018) es un generador de piezas musicales lanzado por Google, capaz de crear composiciones coherentes a largo plazo. Este modelo es uno de los primeros en aplicar una estructura basada en *transformers* para la generación de música,

captando la periodicidad en varias escalas temporales y las relaciones entre características escalares.

- **SentiMozart** (Madhok et al., 2018) es un generador de música simbólica basado en el reconocimiento de emociones detectadas en el rostro. Este modelo está compuesto por una red neuronal convolucional creada para clasificar la expresión en siete categorías y un segundo modelo, basado en *Long Short-Term Memory* (LSTM), que genera música condicionada por la emoción detectada.

- **AWS DeepComposer** (Amazon Web Services, 2019) es una de las primeras soluciones comerciales basadas en IA generativa de música y fue lanzada por Amazon en 2019. Esta aplicación permite crear música nueva a partir de las entradas de música existentes en formato MIDI.

- **Music-sentneuron** (Ferreira y Whitehead, 2021) profundiza en el uso de LSTM para la generación de música basada en emociones. Este sistema utiliza un conjunto de entrenamiento de 95 piezas de piano, etiquetadas por humanos. Los resultados del modelo obtienen un mejor desempeño en generaciones asociadas a emociones positivas.

- **SymphonyNet** (Liu et al., 2022) es un generador de música simbólica multi-instrumental diseñado para crear música sinfónica. Este modelo propone un enfoque de lenguaje invariante a permutaciones, basado en *transformers*.

- **Midi-emotion** (Sulun et al., 2022) es un generador de música simbólica condicionada por emociones, utilizando una arquitectura basada en *transformers*. Frente a otros trabajos previos, incorpora un condicionamiento de la emoción continuo de acuerdo al modelo de Russell. La arquitectura es entrenada primero de forma no supervisada y finalmente se realiza un ajuste supervisado, incorporando el condicionamiento por emoción.

- **MuseCoco** (Lu et al., 2023) es un modelo diseñado para la generación de música simbólica a partir de indicaciones textuales. Este modelo utiliza una arquitectura basada en *transformers* con dos fases, lo que simplifica el proceso de aprendizaje y mejora la capacidad de control. MuseCoco reduce la dependencia de grandes cantidades de datos emparejados de texto y música.

- **MeloTrans** (Wang et al., 2024) es un modelo de composición que convierte texto en música simbólica, empleando un desarrollo de motivos basado en reglas para asegurar una estructura específica y un patrón cambiante. Utiliza una arquitectura basada en *transformers*.

- **MuPT** (Qu et al., 2024) es un sistema diseñado para la generación de música simbólica utilizando una arquitectura de *transformers*. Este modelo se basa en descriptores

simbólicos en formato MIDI o texto, para la generación de música.

- **EmoMusicTV** (Ji y Yang, 2024) es un modelo de generación de música simbólica basado en emociones. Su arquitectura se basa en un *transformer* con un VAE jerárquico e incorpora un condicionamiento tanto a nivel de pieza como de compás, permitiendo una generación más guiada y coherente.

En el campo de la **generación audio**, destacan los siguientes trabajos.

- **Melodrive** (Melodrive, 2018) es uno de los primeros modelos de IA generativos comercializados, desarrollado mediante un sistema de música de videojuegos en tiempo real. Este sistema genera nueva música ambiente que se adapta continuamente a la interacción del usuario y la escena del juego.

- **MelGAN** (Kumar et al., 2019) es un modelo basado en Red Generativa Antagónica o *Generative Adversarial Network* (GAN) diseñado para la generación de formas de onda coherentes de alta calidad. Este modelo trabaja con espectrogramas para la síntesis de la voz, conversión entre dominios musicales y síntesis musical.

- **Jukebox** (Dhariwal et al., 2020) es un modelo publicado por OpenAI, capaz de imitar diferentes estilos y artistas mediante *transformers* autoregresivos. Es uno de los primeros modelos capaz de introducir letra en las composiciones y destaca por su capacidad para generar piezas de varios minutos de duración con voces reconocibles y naturales.

- **RAVE 1** (Caillon y Esling, 2021) es el primer sintetizador publicado por la empresa Ircam, basado en un *Autoencoder* variacionales o *Variational AutoEncoders* (VAE) que permite una síntesis rápida y de alta calidad de formas de onda en tiempo real, optimizando la calidad y eficiencia en ordenadores convencionales.

- **From words to sound** (The Sound of AI Community, 2022) es un desarrollo de código abierto liderado por Sound Of AI de un sintetizador de sonidos de guitarra a partir de descriptores tímbricos derivados de la voz del usuario.

- **Rifussion** (Forsgren y Martiros, 2022) es un modelo de generación de audio condicionado por imagen y basado en redes neuronales convolucionales. Rifussion permite crear texturas sonoras complejas al aprovechar los descriptores extraídos de los espectrogramas de las imágenes.

- **MusicLM** (Agostinelli et al., 2023) es un modelo generativo de audio desarrollado por Google en 2023 que utiliza una arquitectura de *transformers* y se condiciona por texto para generar audio de alta calidad. Tiene la capacidad de generar audio a 24kHz con una duración de varios minutos mediante descriptores textuales, adaptando la composición al

estilo, duración y otros parámetros indicados. Su entrenamiento se basa en el conjunto de datos de MusicCaps, que está compuesto por 5.500 pares de música y texto de alta calidad.

- **Bark** (AI, 2023) es un modelo desarrollado por Suno para la generación de audio a partir de descriptores textuales. Se caracteriza por la generación de audio altamente realista y multilingüe, lo que incluye no solo música, sino también ruido de fondo y efectos de sonido. Además, Bark puede generar comunicación no verbal, como risas, suspiros y llantos.

- **MusicGen** (Copet et al., 2024) es un modelo generativo de audio desarrollado por Meta basado en una arquitectura de *transformers*. Este sistema está diseñado para crear audio a partir de descripciones textuales simples. Está orientado a la creación de música ambiental y de fondo, con duraciones cortas.

- **Mustango** (Melechovsky et al., 2024) es un modelo generativo de audio de código abierto publicado para la creación musical condicionada por texto. Este modelo se basa en una arquitectura de *Latent Diffusion Model* (LDM) y MuNET, y utiliza el dataset MusicBench para su entrenamiento.

La mayoría de los proyectos previos a 2018 emplean modelos de aprendizaje profundo y conjuntos de datos masivos, utilizando arquitecturas capaces de generar secuencias tanto de audio como simbólico. Sin embargo, estos enfoques presentan carencias en términos de duración y coherencia temporal. La introducción de la arquitectura tipo *transformer* en 2017 supone un avance significativo en la generación de piezas más largas y coherentes. A partir de ese momento, los avances en la generación condicionada toman mayor protagonismo, destacándose la generación de texto a música, permitiendo composiciones de mejor calidad basadas en indicaciones textuales.
Dado el interés de este trabajo en profundizar en los avances de la generación de música simbólica condicionada, a continuación se lleva a cabo un análisis más exhaustivo de dos de los trabajos previamente mencionados, Midi-emotion y EmoMusicTV.

3.1.1. Midi-emotion

Este proyecto presentado en el documento *Symbolic Music Generation Conditioned On Continuous-Valued Emotions* (Sulun et al., 2022) desarrolla un generador de música simbólica multi-intrumental bajo un condicionamiento emocional, de acuerdo al modelo de *valence-Arousal* de Russell (Russell, 1980), utilizando una arquitectura *transformer* como base. Este condicionamiento permite generar la música en base a un espacio continuo de *valence* (placer-desagrado) y *arousal* (relajación-excitación).
En primer lugar, se presenta el dataset MIDI con etiquetas continuas y discretas *valence-arousal*, mejorando significativamente el tamaño de los datasets existentes con características

similares. Este dataset, además de incorporar la información emocional de acuerdo al modelo de Russell, incluye características de audio extraídas de Spotify, añadiendo un valor extra a las representaciones de las emociones en la música.
El condicionamiento por emoción se presenta en los siguientes tres tipos (Figura 3.1).

1. *Discrete-token*, valores de *valence-arousal* discretizados en cinco contenedores, representando las categorías muy bajo, bajo, moderado, alto y muy alto. Estos contenedores se convierten en *tokens* de control, introducidos al principio de la secuencia de *tokens* musicales.

2. *Continuous-token*, valores de *valence-arousal* utilizados de forma continua, sin discretización. Cada valor continuo se transforma en un vector de condición mediante una capa lineal, teniendo la misma dimensión que los *embeddings* de los *tokens* musicales. Este vector se introduce al principio de la secuencia de *tokens* musicales para alimentar al modelo.

3. *Continuous-concatenated*, combina los valores de *valence-arousal* en un único vector de condición, que se repite y se concatena con cada *token* musical en la secuencia de entrada. Esto permite una mejor integración en cada paso de la generación, permitiendo que el modelo tenga en consideración las emociones de manera global y continua.

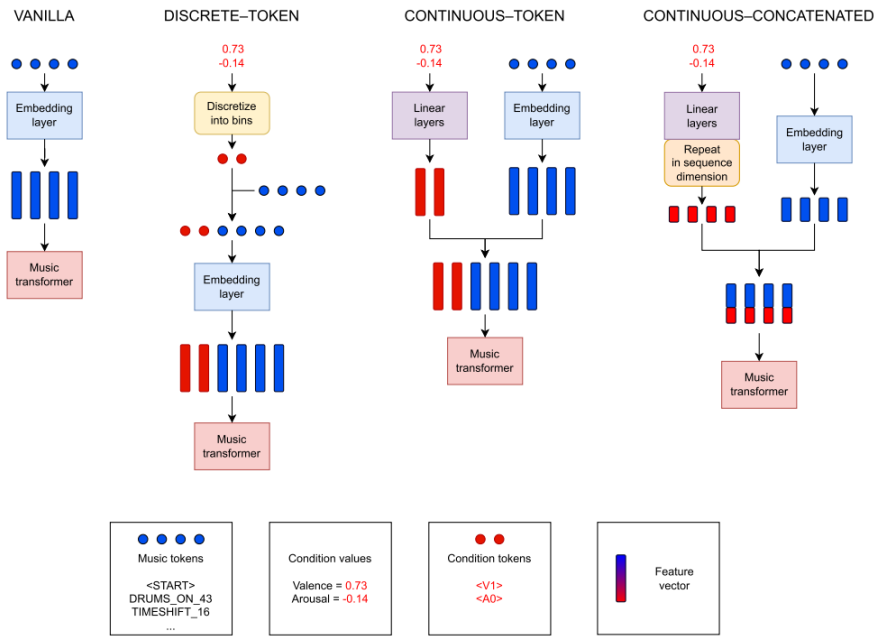

Figura 3.1: *Tipos de condicionamiento emocional en la generación de música simbólica.*

El modelo se preentrena en primer lugar de manera no supervisada mediante un conjunto de datos no etiquetado emocionalmente. Una vez preentrenado, se realiza un entrenamiento posterior mediante los valores de *valence-arousal* para condicionar la generación de música, mediante los métodos previamente expuestos. La arquitectura base es un *transformer* con 20

encoders, 16 capas de atención y una dimensión de 768, capturando dependencias de larga duración.

La evaluación del modelo se realiza según dos consideraciones principales. En primer lugar, se utilizan métricas de evaluación estándar, como la log-verosimilitud negativa y las precisiones top-1 y top-5 para evaluar la precisión de las notas predichas. En segundo lugar, se evalúa el condicionamiento de la emoción mediante un modelo de regresión entrenado para predecir los valores de *valence-arousal* de las muestras generadas. Estos valores se comparan con los valores de condición utilizados durante la inferencia, utilizando como métrica de error la distancia L_1 normalizada.

En líneas generales, el enfoque *continuous-concatenated* presenta los mejores resultados tanto en términos de precisión como a nivel de métricas de emoción, incorporando características adicionales como el condicionamiento continuo frente al condicionamiento *discrete-token*. Además, este enfoque supera a proyectos previos en la capacidad de incorporar información emocional de forma coherente.

3.1.2. EmoMusicTV

Este proyecto de generación de música simbólica tiene como objetivo explorar la generación condicionada por emociones mediante técnicas de aprendizaje profundo. Para ello, se desarrolla un modelo basado en *transformer* con un VAE jerárquico, incorporando un condicionamiento tanto a nivel de pieza como de compás para una mayor coherencia musical (Figura 3.2).

Los conjuntos de datos con etiquetas emocionales son escasos, destacando los datasets VGMIDI y EMOPIA, con etiquetas a nivel de pieza, pero sin contar con información temporal dentro de cada composición. Para resolver esta problemática, el proyecto amplía estos conjuntos de datos con otros públicos y aplica un proceso automatizado de identificación de emociones a nivel de compás, basado en las progresiones de acordes. De esta manera, se enriquecen los datos con etiquetas tanto a nivel de pieza como de compás, permitiendo al modelo un mayor entendimiento de la evolución emocional de la pieza.

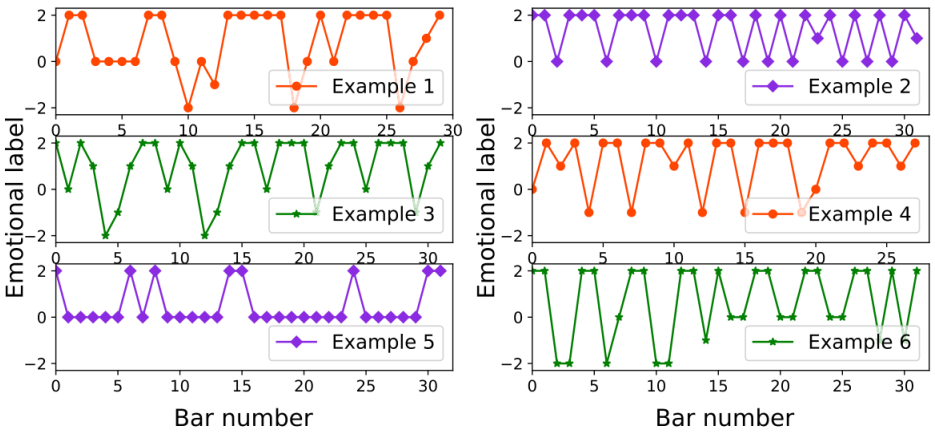

Figura 3.2: *Ejemplos de transiciones emocionales a lo largo de diferentes compases bajo la misma emoción global de positividad moderada.*

EmoMusic combina la arquitectura de *Transformer* con un VAE, ofreciendo importantes ventajas en la generación de música emocional frente a trabajos previos. Por un lado, el *transformer* permite capturar dependencia a largo plazo, asegurando coherencia a lo largo de toda la pieza, mientras que el VAE introduce cierta incertidumbre controlada, permitiendo generar diversas variaciones de una misma pieza sin perder coherencia. Además, al regularizar el espacio latente, se mejora la capacidad del modelo para generalizar nuevas entradas con *tokens* emocionales. Por último, EmoMusicTV supera los enfoques anteriores, sobre todo en tareas como la armonización de melodías, la generación de melodías a partir de armonías y la generación guiada. Esta evaluación se realiza a través de nueve métricas objetivas y mediante pruebas de escucha con evaluación humana, comparando las composiciones del modelo con piezas creadas por músicos para medir la precisión en la transmisión de emociones.

3.2. *Datasets*

Los conjuntos de datos cumplen un papel fundamental en la generación de música, como se ha descrito en la sección 3.1. Sin embargo, la mayoría están sujetos a derechos de autor, limitando por tanto su accesibilidad. A continuación se presentan algunos de los conjuntos más relevantes.

3.2.1. *The Lakh MIDI Dataset* (LMD-full)

The lakh MIDI Dataset (Raffel, 2016) es un conjunto de datos MIDI utilizado para entrenar modelos como Zeng et al. (2021). Está compuesto por una colección de 175.581 archivos MIDI, de los cuales 45.129 están emparejados y alineados con entradas del Million Song Dataset (Raffel y Ellis, 2011). Este conjunto de datos contiene una gran variedad de géneros musicales, estilos y composiciones.
Además, los archivos MIDI tienen asociados información adicional tales como archivos de audio, datos sobre artistas, géneros y letras.

3.2.2. MusicCaps

MusicCaps (Gemmeke et al., 2017) es un conjunto de datos derivado de Audioset, compuesto por 5.521 muestras de música, cada una de las cuales está asociada a un fragmento de 10 segundos extraído de un vídeo de YouTube. Cada muestra está etiquetada con una lista de características musicales en inglés y un texto descriptivo elaborado por músicos.
Un ejemplo de característica es, *brass, double bass, strings, instrumental, no voice, percussion*. El texto descriptivo asociado podría ser, *The instrumental music features an ensemble that resembles the orchestra. The melody is being played by a brass section while strings provide harmonic accompaniment. At the end of the music excerpt one can hear a double bass playing a long note and then a percussive noise.* Como se puede apreciar en estos ejemplos, el texto está centrado únicamente en describir cómo es la música y no en los datos asociados como el nombre del artista.

Toda esta información está contenida en un archivo CSV junto con diversa información adicional, como puede ser la identificación del autor que elaboró la descripción.

3.2.3. MAESTRO

MAESTRO (Hawthorne et al., 2019) es un conjunto de datos creado por Magenta, compuesto por archivos de audio y MIDI, y que incluye aproximadamente 200 horas de interpretaciones al piano. El conjunto de datos MIDI incluye información detallada como velocidades de pulsación de teclas y posiciones de pedal de *sustain*. Los archivos de audio y MIDI están alineados con una precisión de aproximadamente 3 ms y las piezas están organizadas de manera individual, con anotaciones sobre el compositor, título y año de la interpretación. El audio sin comprimir, en formato PCM estéreo, es de calidad CD o superior, con una resolución de 16 bits y una frecuencia de muestreo entre 44,1 y 48 kHz.

3.2.4. VGMIDI

VGMIDI (Ferreira y Whitehead, 2021) es un conjunto de datos compuesto por arreglos de piano, que incluye 200 archivos MIDI clasificados de acuerdo a emociones y 3.850 archivos sin clasificar. La clasificación de las piezas fue realizada por un grupo de 30 expertos, de acuerdo al modelo circumplejo (*valence-arousal*) de emociones de Russell.

3.2.5. EMOPIA

EMOPIA (Hung et al., 2021) es un conjunto de datos multimodal que combina representaciones de audio y MIDI, centrado en la emoción percibida en la música pop de piano. Está compuesto por 1.087 clips musicales extraídos de 387 canciones, con etiquetas asociadas descritas por cuatro anotadores.
En el dominio simbólico, se incluye información como la duración de las notas, velocidad, densidad de las notas rítmicas y la tonalidad. Para el dominio del audio, se utiliza un promedio de 20 dimensiones del *Mel-Frequency Cepstral Co-efficient* (MFCC).

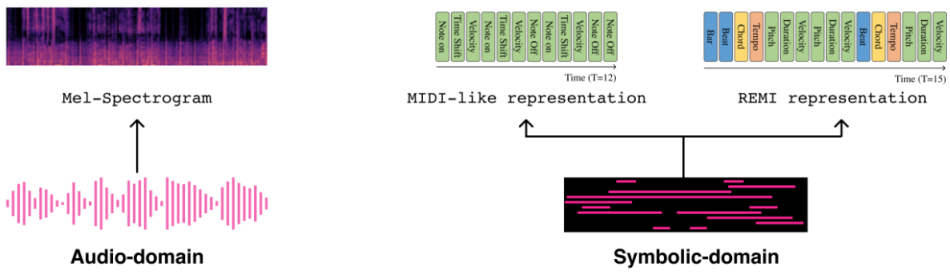

Figura 3.3: *Representación de entrada en formato espectrograma MEL para audio, y MIDI y REMI para simbólico.*

3.2.6. MetaMIDI (MMD)

MetaMIDI (MMD) (Ens y Pasquier, 2021) es un conjunto destinado a la investigación, compuesto por 436.631 archivos MIDI junto con su metadata asociada, como el artista, título y género. Los archivos MIDI se comparan con una colección de 32 millones de clips de audio de 30 segundos de Spotify y MusicBrainz (MusicBrainz, 2024).

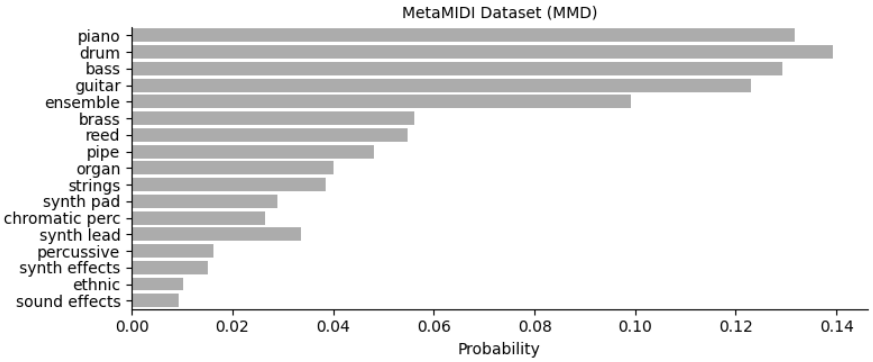

Figura 3.4: *Distribución por instrumento del conjunto de datos MetaMIDI.*

Además de los conjuntos de datos previamente citados, el *Music Technology Group* de la Universidad Pompeu Fabra (UPF) ofrece otros conjuntos de datos relacionados con el audio y MIDI, que son de gran relevancia para este proyecto (Music Technology Group, 2024).

3.3. Implicaciones éticas de la IA generativa de música

A pesar de tratarse de un trabajo de desarrollo de inteligencia artificial generativa, resulta importante considerar los aspectos éticos implícitos en este tipo de herramienta. Actualmente, son multitud las aplicaciones de IA presentes en la vida cotidiana, facilitando tareas como el diagnóstico médico, la identificación de uso fraudulento de tarjetas de crédito, la transcripción de discursos o el funcionamiento de vehículos autónomos. Sin embargo, también existen aplicaciones que generan controversia, como el reconocimiento biométrico o la creación y distribución de contenido falso.

El impacto de la IA se extiende también en el campo del audio y la música, interviniendo desde su distribución con tecnologías de *music recommendation* o *music retrieval*, que son utilizadas en plataformas como Spotify (2008) o Shazam Entertainment Ltd. (2002), hasta su creación, mediante herramientas como las citadas en la sección 3.1. Algunos ejemplos del uso de herramientas generativas en la música son Southern (2017), con el primer álbum compuesto íntegramente por inteligencia artificial o Machines (2018), primer álbum que mezcla composición de IA con artistas humanos.

En estas creaciones surgen aspectos controvertidos diversos, como la autoría y la creatividad de las obras, el uso de datos protegidos por derechos de autor para el entrenamiento de estos modelos, u otros aspectos como el *phishing*, la seguridad y privacidad, o las *deepfakes* en el

ámbito del audio. Además, se plantean diversos aspectos más generales en el ámbito de la IA, como son el alto consumo energético y el desempleo que pueden generar estas herramientas. El rápido crecimiento de esta tecnología y la falta de un acuerdo global sobre su regulación (European Parliament, 2023) (Joint Research Centre (JRC), 2023), dificultan la obtención de por ahora conclusiones definitivas. Como expresión de esta problemática e inquietud, existen numerosas publicaciones científicas (Barnett, 2023) (Sturm et al., 2019) y artículos de opinión que abarcan esta temática (Planet, 2023), (Kluwer Copyright Blog, 2023).
A continuación, se analizan algunos de los aspectos más relevantes que surgen de la implementación de la inteligencia artificial generativa en la música, centrándose en algunas de sus implicaciones éticas y legales.

- **Propiedad intelectual**

  En lo que respecta a la autoría de una obra generada por IA surgen dos cuestiones clave. Desde el punto de vista ético, se debate si la autoría de una creación debe atribuirse al usuario que mediante un *prompt* introduce las indicaciones, al programador del modelo o incluso a la propia IA. Este debate se intensifica cuando no hay intervención humana directa en la creación, planteando cuestiones como la intencionalidad de la obra.
  Desde el ámbito jurídico, la mayoría de los países reconocen únicamente a la persona física como posible autor legal de una obra. Concretamente en España, de acuerdo con Boletín Oficial del Estado (1996), se distingue entre obra, definida como cualquier creación con originalidad, tangible o intangible, y autor, como la persona física que la crea. Estas definiciones establecen que una IA puede crear una obra, pero nunca tendrá la autoría de ella.
  Aunque la mayoría de los países, como pueden ser China, Japón, Canadá o la Unión Europea, siguen discutiendo la necesidad de ajustar sus políticas de derechos de autor y mantienen que la autoría debe ser atribuida únicamente a una persona física. Sin embargo, Estados Unidos, mediante la modificación del U.S. Copyright Office (2024) ha comenzado recientemente a permitir que se registren los derechos de autor para obras generadas con ayuda de la IA, siempre que exista una intervención humana en el proceso de creación.
  En España, tras la aprobación del anteproyecto de ley para un uso ético, inclusivo y beneficioso de la inteligencia artificial (Ministerio para la Transformación Digital y de la Función Pública, Secretaría de Estado de Digitalización e Inteligencia Artificial, 2024), implementando lo establecido en Parlamento Europeo y Consejo de la Unión Europea (2024), establece algunas medidas como la obligatoriedad del etiquetado de contenido generado por IA. Sin embargo, no aborda cuestiones fundamentales como la autoría de las obras generadas mediante inteligencia artificial.
  La falta de un marco común a nivel global dificulta una regulación uniforme, generan diferencias y establece incertidumbres tanto para los creadores como para la industria.

- **Creatividad y originalidad**

  La música generativa, por su naturaleza, tiende a ser repetitiva, ya que el contenido

generado tiende a imitar los patrones detectados en los conjuntos de entrenamiento sin exponer verdadera creatividad (Kong et al., 2020). En una primera instancia, se puede considerar que el proceso de aprendizaje de una IA y el del ser humano son similares, dado que ambos aprenden de la imitación de los estímulos que le rodean. No obstante, a pesar de que las obras generadas por IA son convincentes, suelen carecer de profundidad conceptual, metáforas o significados complejos propios del pensamiento humano.
La creatividad humana está impulsada por experiencias personales y por la intencionalidad de transmitir un mensaje, características de las que carece una máquina. Introducir el concepto de creatividad en el aprendizaje automático es un desafío, ya que cuantificar explícitamente pérdidas en términos de creatividad es una tarea compleja (Chemla-Romeu-Santos y Esling, 2022).
Según indica Yin et al. (2022), la música generada por un modelo *transformer* presenta un nivel de originalidad muy inferior al de las composiciones humanas, con una confianza estadística del 95 %. Además, un análisis musical de los resultados muestra que el modelo *Transformer* con criterio de parada produce patrones de repetición de notas aisladas. Sin embargo, conforme se incrementan las épocas de entrenamiento, el modelo tiende al sobreajuste, replicando fragmentos de piezas del conjunto de entrenamiento.

- **Infracción de los derechos de autor**

  Una de las consideraciones más importantes de los modelos generativos de música tanto a nivel ético como legal, es la infracción de los derechos de autor o de *Copyright*. El *copyright* se refiere a los derechos que protegen la propiedad intelectual del autor sobre su obra artística, concediendo y protegiendo los derechos exclusivos sobre una obra para su explotación, divulgación, reproducción o edición.
  Como se mencionó anteriormente, es habitual que los modelos generativos produzcan resultados sustancialmente similares a los elementos del conjunto de entrenamiento (Kaur y Tuttosi, 2023). Esto suele producir contenidos similares a las piezas del conjunto de entrenamiento, pudiendo incluir obras protegidas por *copyright*, lo que resultaría en infracciones legales (New York Post, 2024). Esto plantea el debate sobre si el entrenamiento de IA puede considerarse legítimo o si constituye una violación de los derechos de autor (Sturm et al., 2019).
  Esta situación está originando que los proveedores de conjuntos de datos para entrenamientos de IA, como DataMXR (2024) o Rightsify (2023), ofrezcan conjuntos de datos musicales de alta calidad y legalmente conformes. De esta manera se apoyan tanto a los músicos como a las empresas de IA dentro del marco legal de la protección del marco científico.

- **Apropiación cultural**

  Los modelos generativos suelen entrenarse con grandes cantidades de datos sin considerar la procedencia cultural de los mismos. Este hecho ha generado múltiples discusiones al respecto, ya que estos sistemas pueden utilizar elementos de culturas marginales sin

ninguna investigación previa ni un compromiso con la comunidades creadoras, imposibilitando que estas reciban compensación por su aportación cultural.
Según Agostinelli et al. (2023), los modelos generativos de audio también presentan sesgos en los conjuntos de entrenamiento. La representación desigual de distintas culturas y estilos puede favorecer determinados estilos o tradiciones musicales mientras que otros se distorsionan, reforzando así la aparición de determinados sesgos. Además, los modelos de IA pueden extraer fragmentos musicales sin comprender su contexto social o psicológico, aislando la música de su significado original.

- ***Deepfakes***

  Las *deepfakes* en la música implican normalmente la creación de contenido de audio manipulado o falsificado. Este hecho tiene implicaciones tanto éticas como legales, ya que puede involucrar el uso no autorizado de la voz de un artista o la alteración de sus canciones sin el consentimiento de los propietarios de la obra. Además, genera confusión sobre la autenticidad de la música y tiene un efecto negativo en la reputación del artista, al difundir contenido falso (Music Business WorldWide, 2025).

- **Consumo energético**

  Los modelos generativos tienen un considerable impacto en la huella de carbono, tanto durante el entrenamiento como en la inferencia, al tratarse de procesos que requieren una alta demanda de recursos computacionales, especialmente de procesadores gráficos, durante largos periodos de tiempo.
  Según Anthony et al. (2020), las investigaciones actuales apuntan que los modelos de aprendizaje automático corren el riesgo de contribuir de forma significativa al cambio climático. Por ello, se propone incluir métricas que no sólo evalúen el rendimiento del modelo, sino también que midan el consumo energético y las emisiones de carbono. En este estudio se demuestra que el impacto ambiental de los modelos varía según la región y el país donde se genere la electricidad. Por ejemplo, un modelo de segmentación de imágenes médicas en Estonia emite 61 veces más de dióxido de carbono que el mismo modelo entrenado en Suecia (lo que equivale a una diferencia similar entre recorrer en coche 9,04 Km frente a 0,14 Km).

- **Desempleo**

  Según Research (2023), se estima que entre 2023 y 2033 en España se crearán un total de 1,6 millones de puestos de trabajo asociados con la inteligencia artificial, aunque se perderán 2 millones de trabajos previos. Esto también afecta al campo del audio, impactando especialmente en los puestos de trabajo de los ingenieros de mezcla y presentadores de radio (Huang et al., 2023). Como ocurre en toda revolución industrial/tecnológica, la automatización del trabajo provocará un aumento de la demanda de empleo en la IA y generará lo que se denomina como *desplazamiento de empleos*.

Marco Teórico

4

A continuación se presenta una revisión de los fundamentos teóricos que resultan necesarios para el correcto desarrollo de este proyecto.

4.1. Musical Instrument Digital Interface (MIDI)

En la generación de música simbólica, existen diversos formatos de representación de los eventos musicales, como la notación ABC, que utiliza texto para representar melodías y acordes y MIDI, que codifica los eventos musicales como notas, duración y dinámica. MIDI es el formato más extendido, debido a la gran cantidad de base de datos y al nivel de detalle que proporciona en la representación musical.
Este protocolo fue desarrollado a principios de 1980, con el objetivo de unificar la comunicación entre los distintos dispositivos de hardware musical. Este se basa en mensajes compuestos por varios bytes, cuyos valores oscilan entre 0x80 (128) y 0xFF (255).
Los mensajes MIDI están formados por dos tipos de bytes principalmente:

- ***Status byte***, es el primer byte del mensaje y contiene el bit de mayor peso (el bit 7) activado. Su función principal es indicar el tipo de información que se debe activar.
- ***Data byte***, contiene información de los eventos musicales, como la velocidad y el tono de la nota, entre otras.

Adicionalmente, los mensajes MIDI asociados a estos bytes se pueden dividir en diversas categorías, destacando como los siguientes:

1. ***Channel Messages***, contienen información relacionada con el canal procesado.
    a) *Note On*, indica el comienzo de una nota e incluye información sobre su tono (0-127) y velocidad (0-127). Por ejemplo [0x90, 60, 127] representa el inicio de la nota C4 con máxima velocidad.
    b) *Note Off*, muestra el final de una nota, con la misma estructura de datos que *Note On*.
2. ***System Messages***, mensajes destinados para controlar el sistema MIDI.
    a) *System Common Message*, proporciona información de sincronización entre archivos. Son de especial interés para la sincronización de equipos MIDI.

b) *System Real-Time Message*, se emplean para la sincronización y el control del tiempo. Entre los más relevantes están *Timing Clock* (0xF8), inicio de reproducción (0xFA) y parada de reproducción(0xFB).

3. ***SysEx***, son mensajes exclusivos del sistema, utilizados por los fabricantes para enviar configuraciones específicas entre sus dispositivos.

Estos mensajes MIDI se pueden procesar en Python mediante diversas bibliotecas que permiten leer, escribir y manipular los datos. Entre las más destacadas se encuentran *mido* (Bjørndalen y Doursenaud, 2024) y *pretty_midi* (Raffel y Ellis, 2014), que facilitan la interpretación y manipulación de los diferentes tipos de mensajes previamente citados.
Mido, es una de las más extendidas y representa las información mediante un objeto Python denominado *Message*, con métodos y atributos asociados. Los mensajes en Mido fundamentalmente se dividen en dos categorías:

1. **Mensajes de Canal**, contiene información asociada a eventos musicales concretos, tales como notas, controles y cambios de programa.
   - *Note On*, establece el inicio de una nota, con información del tono y la velocidad.
   - *Note Off*, indica el final de una nota.
   - *Control Change*, modifica parámetros de control como el volumen o el balance.
   - *Program Change*, indica cambio de instrumento o timbre de una pista.
   - *Pitch Bend*, marca ajuste de la afinación de una nota.
   - *delta time*, representa la cantidad de *ticks* transcurridos desde el último mensaje.
2. **Mensajes de Metadatos**, contiene información asociada a la estructura del archivo MIDI sin implicación directa en la reproducción del sonido, pero sí en la organización y desarrollo de la música.
   - *Ticks per beat*, establece el número de *ticks* por pulso (Figura 4.1).
   - *Time Signature*, define el compás del archivo o fragmento.
   - *Key Signature*, especifica la tonalidad de la pieza.
   - *Track Name*, define el nombre de la pista.
   - *End of Track*, marca el final de una pista dentro del archivo MIDI.

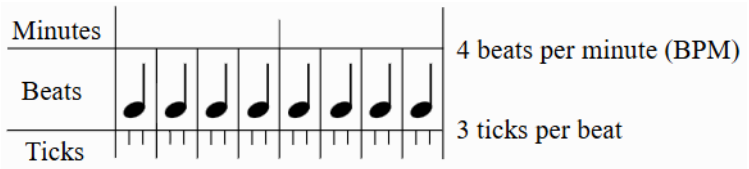

Figura 4.1: *Resolución temporal en MIDO.*

4.2. Técnicas de IA generativa

Las principales arquitecturas de aprendizaje profundo utilizadas en la generación de música, tanto en el campo simbólico como de audio, son las que se exponen a continuación.

4.2.1. *Autoencoder* variacionales o *Variational AutoEncoders* (VAE)

Modelo basado en el *Autoencoder* compuesto por dos módulos principales, el codificador y el decodificador (Figura 4.2) (Kingma y Welling, 2022). La principal característica entre el Auto-Encoder y el VAE reside en que este último codifica las muestras en el espacio latente como una distribución, en lugar de como un único punto. Esto evita el sobreajuste durante el entrenamiento, resultando un espacio latente capaz de generar muestras nuevas.
De esta manera, el codificador se encarga de obtener una representación comprimida de los datos y el decodificador utiliza la representación comprimida de los datos para reconstruir los datos originales.

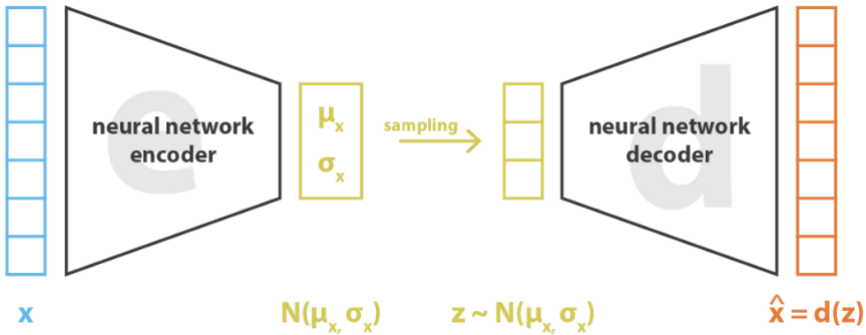

Figura 4.2: *Arquitectura de VAE. Fuente: Toward Data Science.*

Al introducir una distribución de probabilidad, es necesario una función de pérdidas distinta. De esta manera, la función de pérdida está compuesta por la divergencia KL (Kullback-Leiber) que evalúa la similitud entre dos distribuciones de probabilidad mediante los términos mu y sigma, y el término de error que compara la similitud entre la entrada y la imagen reconstruida.

$$f_{VAE} = ||x - \hat{x}||^2 \ KL N\mu_x, \sigma_x, N0,1 = ||x - dz||^2 \ KL N\mu_x, \sigma_x, N0,1$$

4.2.2. Red Generativa Antagónica o *Generative Adversarial Network* (GAN)

Es una arquitectura de aprendizaje desarrollada por un grupo de estudiantes (Goodfellow et al., 2014). Su enfoque se basa en enfrentar dos redes neuronales para que compitan entre sí para generar nuevos datos más auténticos a partir de un conjunto de datos de entrenamiento. Estas dos redes neuronales se denominan generador y discriminador (Figura 4.3).
El generador es el módulo encargado de crear datos inventados a partir de un vector de ruido, conocido como espacio latente. Este espacio latente es una representación comprimida de la distribución de los datos originales. A partir del muestreo de este espacio latente, el generador

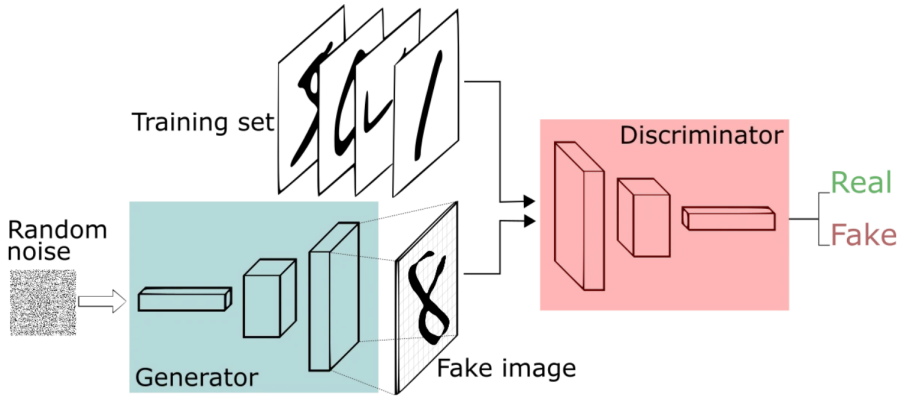

Figura 4.3: *Arquitectura de GAN. Fuente: Skyman*

puede crear datos diferentes a los de entrada.
El discriminador tiene como objetivo determinar si cada una de las instancias proporcionadas se corresponden o no con el conjunto de datos de entrenamiento. Es decir, un modelo de clasificación binario, etiquetando como cero las instancias identificadas como falsas y uno las reales.
El entrenamiento de las GAN comienza con la creación de contenido falso por parte del generador y el discriminador aprende a reconocer este contenido como falso. Conforme aumenta el número de iteraciones, el generador mejora en la creación de imágenes falsas, con el objetivo de que estas creaciones sean lo suficientemente realistas como para engañar al discriminador. En este punto, la precisión del discriminador disminuye, dado que el discriminador identifica imágenes falsas como reales,. La mejora de la calidad de este algoritmo se establece en función a la siguiente función de pérdidas adversarial.

$$f_{adv} = \min_G \max_D VD, G = {}_{\mathbf{x}\sim p_{\text{data}}\mathbf{x}} \left[\log D\mathbf{x}\right] \; {}_{\mathbf{z}\sim p_{\mathbf{z}}\mathbf{z}} \left[1 - \log DG\mathbf{z}\right]$$

Este error se transmite a través de ambas redes neuronales mediante la técnica *backpropagation*, con objeto de actualizar los pesos y ajustar sus parámetros.

4.2.3. *Transformers*

Esta es una arquitectura de aprendizaje profundo propuesto por investigadores de Google (Vaswani et al., 2023) y diseñado inicialmente para el procesamiento del lenguaje natural (NLP). Aunque está inspirado en las redes neuronales recurrentes (RNN) y convolucionales (CNN), introduce un nuevo mecanismo de atención (auto-regresiva o auto-atención) que es capaz de captar de manera más eficientemente las relaciones entre elementos distantes en una secuencia. Gracias a su capacidad de procesamiento paralelo, aprovecha mejor los recursos de GPU y permite analizar de manera rápida y eficiente secuencias largas.
La arquitectura de los *transformers* está basada en un modelo codificador-decodificador, originalmente estructurado en una serie de 6 codificadores y 6 decodificadores. Tanto el codificador

como el decodificador están compuestos por una pila de varias capas (Figura 4.4).

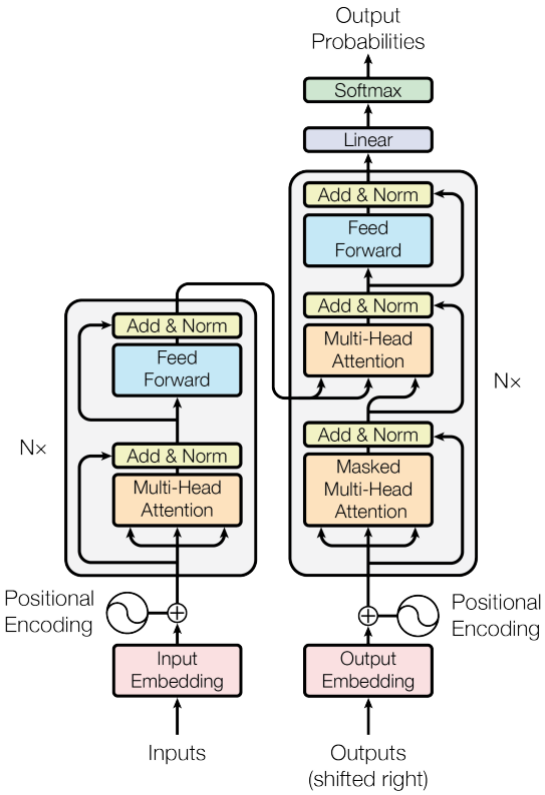

Figura 4.4: *Arquitectura del modelo transformer.*

Las primeras capas de esta arquitectura son:

- **Embedding** es el primer bloque a la entrada del primer codificador y decodificador, encargado de convertir cada token de entrada en vector mediante capas de incrustación.

- **Positional Encoding** añade información sobre la posición de cada *token* en la secuencia mediante la combinación de varias funciones seno y coseno para crear vectores posicionales. Este paso es fundamental, ya que el modelo comprende la posición de cada elemento en la secuencia.

$$Ppos, 2i = sin\frac{pos}{10000^{2idimension_{model}}}$$

$$Ppos, 2i\ \ 1 = cos\frac{pos}{10000^{2idimension_{model}}}$$

Después de estos bloques, situados fuera del codificador y decodificador, los datos pasan a las capas de codificación donde los *tokens* recibidos se transforman en representaciones continuas y contextualizadas de toda la secuencia. Entre las componentes del **codificador** se encuentra:

- **Multi-Head Attention** es el bloque fundamental, el cual implementa el mecanismo *self-attention*, permitiendo relacionar cada elemento de la secuencia con otros. Así, el codificador puede centrarse en diferentes partes de la secuencia de entrada mientras

procesa cada token.

Self-attention se basa en los vectores *Query* (Q), que representa un *token* concreto de la secuencia de entrada, *Key* (K). K se corresponde en cada *token* de la secuencia de entrada y *Value* (V) utilizado para construir la salida de la capa de atención. La primera operación es la multiplicación de los vectores Q y la transpuesta de K, generando una matriz de puntuación que refleja la similitud entre elementos. Esta matriz se divide por la raíz cuadrada de la dimensión de K, con ende de estabilizar los gradientes, seguidamente se aplica una función *softmax* para convertir las puntuaciones en probabilidades acotadas entre 0 y 1. Finalmente, se multiplica el resultado del *softmax* por los valores (V) para obtener la matriz de atención (Z), destacando los elementos con puntuaciones elevadas (Figura 4.5).

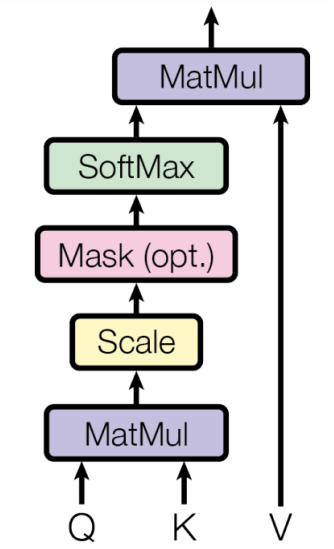

Figura 4.5: *Arquitectura self-attention.*

Para mejorar la capacidad representativa y reducir el riesgo de sobreajuste, *self-attention* se realiza en múltiples instancias paralelas, calculando tantas matrices Q, K, V y Z como bloques de autoatención, centrándose cada uno de los bloques en distintas partes de la secuencia de entrada.

$$Z = \text{concatenate} Z_1, Z_2, Z_3, \ldots, Z_n W_0$$

- **Add & normalization**, después de cada subcapa de los codificadores, se realiza una normalización de los datos a través de las características de cada posición, ajustando los datos a una media de 0 y una desviación estándar de 1. Esto estabiliza los valores y facilita una rápida convergencia al evitar variaciones bruscas. Además, se añade una conexión residual en la que la salida de cada subcapa se suma a su entrada original, suavizando el problema del gradiente evanescente.

- **Feedfodward** está compuesto por dos capas totalmente conectadas con una activación

ReLU entre ellas. Su objetivo es procesar cada dato de posición por separado.

La salida del codificador final se compone por un conjunto de vectores que permiten al codificador prestar atención a los elementos adecuados. De esta manera, mediante un conjunto de subcapas, el decodificador construye una nueva secuencia. Al igual que sucede en el codificador, en la entrada del primer decodificador se aplica una capa de incrustación y una codificación posicional. El resultante se convierte en la entrada de la primera capa de *multihead attention*. Entre las componentes del **decodificador** se encuentran:

- **Masked Multi-Head Attention**, similar al bloque de *multi-head attention* del codificador, pero con la diferencia que éste impide que se atiendan a posiciones futuras. De esta manera, se asegura que los elementos actuales no se vean influenciados por elementos futuros. El proceso elemental, análogo a self-attention, se define como:

$$Z_i\left(Q_i, K_i, V_i\right) = \text{softmax}\left(\frac{Q_i K_i^T}{\sqrt{d_k}}\right) V_i$$

  Al igual que en el codificador, este cálculo se realiza en múltiples instancias, con el fin de mejorar la representatividad y reducir el riesgo de sobreajuste.

- **Multi-Head Attention** es la segunda capa de *multi-head attention* en el decodificador, con la particularidad de que permite la interacción entre componentes del codificador y decodificador. La salida del codificador actúa como *query* (Q) y *key* (K), mientras que la salida del *masked multi-head attention* se utiliza como *value* (V), permitiendo que cada posición del decodificador atienda a todas las posiciones de la secuencia de entrada e incorpore la información del codificador.

$$Z\left(Q, K, V\right) = softmax\left(\frac{Q K^T}{\sqrt{d_k}}\right) V$$

- **Feedforward**, de manera análoga al codificador, esta capa incluye una red de avance totalmente conectada y en cada posición por separado.

- **Add & normalization**, al igual que sucede en el codificador, después de cada subcapa se normaliza y se añade una conexión residual.

Después de todas las capas del decodificador, los datos pasan finalmente por una capa lineal que actúa como clasificador y cuyo tamaño es equivalente al número de clases del modelo. Esta salida se dirige a una capa *softmax*, generando una distribución de probabilidad para cada clase, obteniendo la siguiente posición en la secuencia a partir de la clase con mayor probabilidad.

4.3. *Data Augmentation*

La aumentación de datos es una de las técnicas más extendidas en el *Machine Learning* para generar artificialmente nuevas muestras a partir de datos previos. Entre sus principales beneficios incluye la mejora el rendimiento del modelo, ya que enriquece los conjuntos de datos mediante variaciones, reduce la dependencia de grandes volúmenes de datos y minimiza el riesgo de sobreajuste durante el entrenamiento.
Esta técnica está presente en varios dominios, siendo el de las imágenes uno de los más intuitivos. En este caso, es común aplicar transformaciones diversas, como rotar en ciertos grados la imagen, cambiar su escala o distorsionarla, agregar ruido o disminuir el brillo.
Al igual que en otros campos, la aumentación de datos también se utiliza en el campo del audio y el MIDI, con el objetivo de crear nuevas instancias de datos que conserven las características esenciales del archivo original, pero con variaciones o perturbaciones. Alguna de las técnicas más empleadas en la música son:

- **Cambio de tono o transposición**. Modificación de la tonalidad de una pieza, desplazando todas las notas un número determinado de semitonos hacia abajo o hacia arriba.
- **Modificación del *tempo***. Acelerar o ralentizar la ejecución de la pieza cambiando su tempo, ya sea en secciones o de manera global.
- **Ajuste de la duración de las notas**. Modificar la duración de determinadas notas, acortándolas o alargándolas.
- **Desplazamiento rítmico**. Alterar la posición de las notas a lo largo del tiempo.
- **Fragmentación**. Eliminar ciertas notas aleatoriamente de una pieza.

Estas son algunas de las técnicas que comparten archivos musicales, tanto en formato MIDI como en audio. No obstante, existen otras técnicas técnicas específicas en el campo del audio tales como la adición de ruido, la reverberación o el enmascaramiento de tiempo y frecuencia.

4.4. Tokenización

Como se comentó en la sección 3.1, la mayoría de los modelos presentados utilizan *tokens* embebidos como entrada del modelo, en un proceso conocido como "tokenización". Este proceso consiste en convertir un elemento real en una representación digital mediante un conjunto de *tokens*. La "tokenización.[es] fundamental en los *Natural Language Processing* (NLP), donde cada *token* puede representar desde un carácter hasta una palabra completa. Esta división del lenguaje humano en fragmentos más pequeños y manejables facilita que los ordenadores lo analicen y comprendan mejor.
La música simbólica comparte múltiples características con los modelos del lenguaje natural, ya que está compuesta por una secuencia de notas, acordes y ritmos organizados en un orden

específico, con una estructura y un significado musical. Además, la música sigue reglas armónicas y rítmicas, como las progresiones de acordes, escalas y patrones melódicos, que pueden considerarse análogas a la gramática lingüística. Estas son algunas de las similitudes que destacan la relevancia de implementar un proceso de tokenización para la música simbólica.
En el ámbito de la tokenización en formato MIDI existen diferentes representaciones como *Revamped MIDI* (REMI) (Huang y Yang, 2020), MIDI-Like (Oore et al., 2018), Octuple (Zeng et al., 2021), MuMIDI (Ren et al., 2020) o *Compound Word Tokenize* (CPWord) (Hsiao et al., 2021).
REMI es la tokenización más utilizada para la representación de música simbólica, diseñada originalmente para modelar música pop de piano. En REMI, las notas se representan como una sucesión de *Pitch*, *Velocity* y *Duration*, mientras que el tiempo se codifica con los tokens *Bar* y *Position*. El token *Bar* indica el comienzo de un nuevo compás y *Position* señala la posición relativa dentro del compás actual (Figura 4.6). Este modelo destaca por su enfoque en la codificación rítmica frente a otras técnicas anteriores y está optimizado específicamente para arquitecturas basadas en *Transformers*.

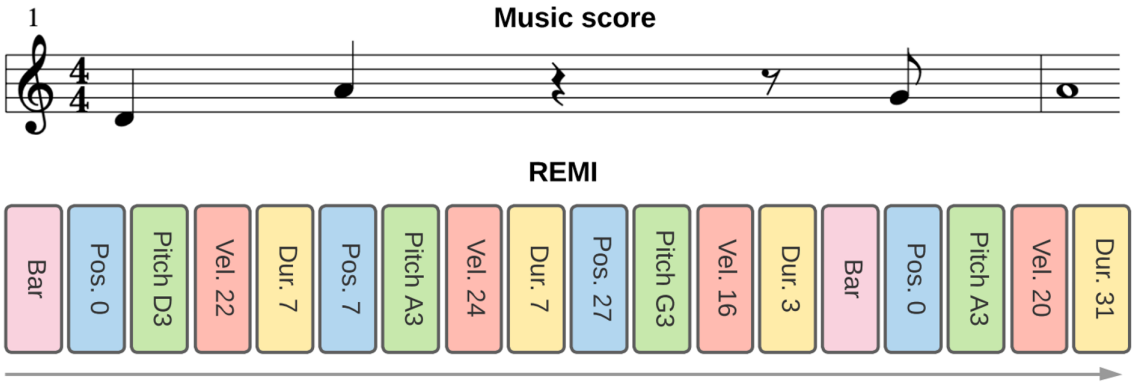

Figura 4.6: *Fuente: Despres, N. (s.f.). MidiTok: A Python package for MIDI file tokenization. Figura 2: A shell music and REMI token representations.*

Existen diversas bibliotecas que facilitan el proceso de codificación, siendo MIDITok (Fradet et al., 2021) la más completa y destacada. Esta biblioteca de Python de código abierto ofrece un proceso flexible y eficiente para la tokenización de archivos de música en formato simbólico, como MIDI o ABC, en diversos formatos.

Metodología

5

En este estudio se pretende elaborar un modelo generativo de música simbólica basada en emociones discretas. Para optimizar el desarrollo del proyecto, reducir los tiempos de implementación y garantizar la fiabilidad y escalabilidad del sistema, se ha empleado una metodología basada en un *Machine Learning pipeline*, siguiendo las siguientes etapas (Figura 5.1).

1. **Recopilación de datos**: Identificación y selección de los conjuntos de datos que más se adecúan al proyecto, priorizando los conjuntos con asociaciones entre archivos MIDI y emociones.

2. **Preprocesamiento de los datos y extracción de sus características principales**: Análisis detallado de los datos obtenidos, incluyendo la detección y corrección de valores erróneos o con inconsistencias musicales, la transformación y normalización de los datos brutos. Además, en esta fase se realiza la división del conjunto en entrenamiento y validación, y se aplican técnicas de *data augmentation* para mejorar la representatividad del conjunto de datos del modelo.

3. **Selección del modelo**: Exploración y selección de la arquitectura más adecuada para la generación de música simbólica basada en emociones, detallando las características del modelo seleccionado.

4. **Entrenamiento del modelo**: Ajuste del modelo al conjunto de datos preprocesado, optimizando los hiperparámetros para minimizar los errores de predicción y mejorar su capacidad de aprendizaje de los patrones y relaciones de los datos introducidos.

5. *Evaluación*. Análisis del rendimiento y calidad de la música generada, tanto mediante métricas objetivas como subjetivas.

A continuación, se describen en mayor profundidad estas fases del modelo[1].

[1] Github del repositorio: https://github.com/luissotomedina/KeyEmotions

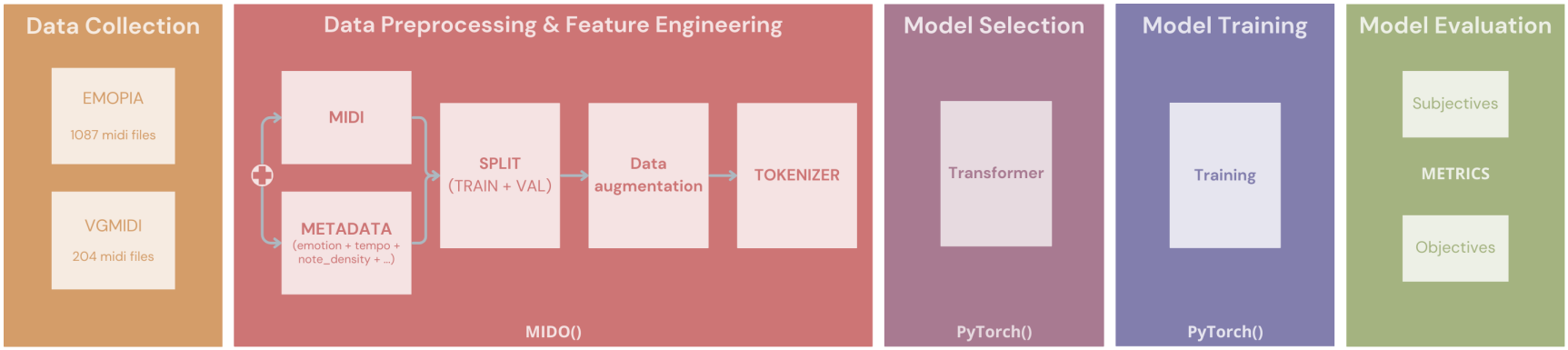

Figura 5.1: *Diagrama de flujo del desarrollo del proyecto.*

5.1. Recopilación de datos

Es necesario disponer de un conjunto de datos que relacionen archivos MIDI con información emocional. Sin embargo, como ya se ha presentado en la sección 3, uno de los principales problemas en este tipo de proyectos es la falta de datos etiquetados, lo que dificulta enormemente la generación de este tipo de música.
Tras analizar los conjuntos de datos que disponen de estas características, se han seleccionado los siguientes *datasets*:

- **Emopia**. Conjunto de datos compuesto por 1.097 clips de piano etiquetados por anotadores. Cada archivo está clasificado de acuerdo a los cuadrantes del modelo de Rusell y dispone de información adicional como tonalidad y tempo. La mayoría de estos archivos tienen una duración inferior al minuto.

- **VGMIDI**. *Datasets* formado por 3.850 archivos de audio y MIDI de piano, de los cuales 200 constan de etiquetas emocionales. Además, incluye un archivo de metadatos con la edad del compositor, género musical, título y anotaciones del etiquetador. En comparación con Emopia, esta base de datos dispone de *clips* de mayor duración. VGMIDI dispone de etiquetas tanto a nivel de pieza como por compás, siendo de especial utilidad para generación con condicionamiento continuo, como en EmoMusicTV.

La combinación de ambos conjuntos de datos da como resultado un conjunto bruto de 1.297 archivos de piano etiquetados a nivel de pieza, según el modelo de Russell. Este *dataset* conforma la base de experimentación para la generación de música simbólica basada en emociones, que se ha seguido en este estudio.

5.2. Preprocesamiento de los datos y extracción de características

Para que el modelo generativo pueda reconocer patrones y entrenarse adecuadamente, es necesario llevar a cabo una serie de procesos previos. La calidad del algoritmo dependerá en gran medida de la precisión y el correcto tratamiento previo de los datos. Esta fase se compone de los siguientes pasos.

5.2.1. Análisis de metadatos y archivos MIDI

A partir del conjunto de datos final, formado por 1.297 archivos, se aprecia una amplia variedad de clips con características diversas, tanto en sus metadatos como en sus características MIDI.

- **Distribución según los cuadrantes de Russell**

  Cada uno de los datasets seleccionados incluyen un archivo en formato texto con los metadatos asociados, especificando el nombre del archivo y el cuadrante de Russell

asociado, además de información extra. Sin embargo, la información asociada a los cuadrantes es diferente para cada uno de los conjuntos de datos. En el primer caso, cada archivo está directamente asociado a un cuadrante específico, mientras que en el otro, la clasificación se aporta mediante coordenadas biaxiales dentro del modelo de Russel. Unificando esta información a la proyección de cada archivo en su cuadrante correspondiente, se obtiene una distribución relativamente uniforme (Figura 5.2). El cuadrante 4 es el más representado, ya que posee un total del 29 % de los datos, mientras que el cuadrante 3 posee una menor representación, con un 22 % de la distribución población.

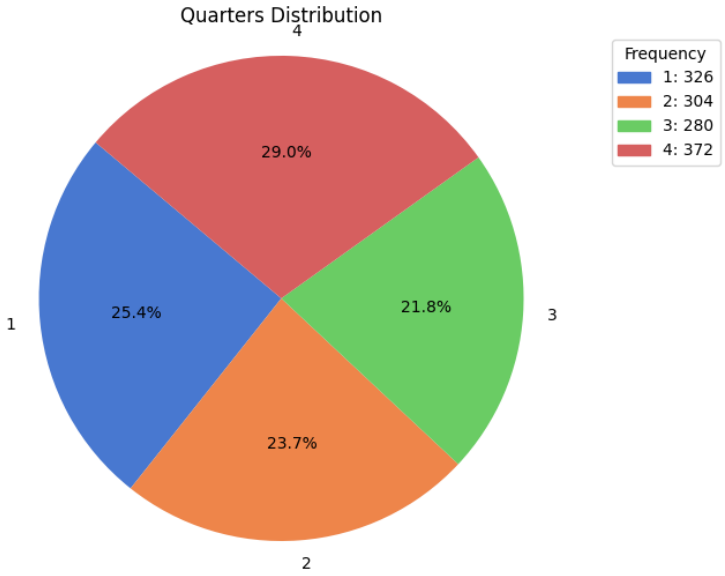

Figura 5.2: *Distribución de los datos según los cuadrantes del modelo de Russell.*

- **Densidad de notas por cuadrante de Russell**

El análisis por cuadrantes según el modelo de Russel permite extraer otra información musical relevante. Los cuadrantes 1 y 2, asociados a emociones como excitado o feliz (cuadrante 1) y tenso o frustrado (cuadrante 2), presentan una mayor densidad de notas (Figura 5.3). Esto refleja una musicalidad más dinámica y con gran presencia de notas de corta duración.

Por el contrario, los cuadrantes 3 y 4, relacionados a emociones de tristeza o aburrimiento (cuadrante 3) y dormido o calmado (cuadrante 4), muestran una menor concentración de notas, reflejando *clips* más pausados, con mayor duración de las notas, lo que aporta una sensación musical más relajada.

Este análisis refleja la diversidad de densidades rítmicas presentes en el conjunto de datos utilizado y su correlación con los cuadrantes de Russell; lo que resulta de gran utilidad para entender los resultados del modelo.

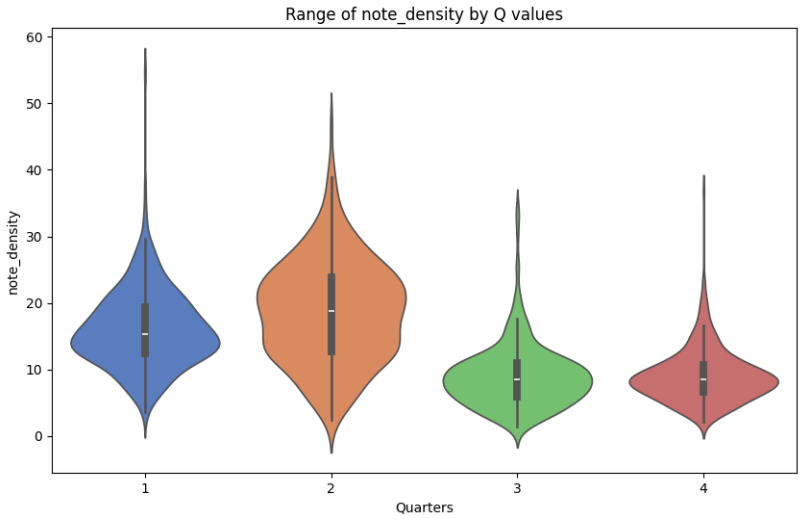

Figura 5.3: *Concentración de notas según los cuadrantes del modelo de Russell.*

- **Duración en compases**

  En lo que refiere a la duración de los archivos MIDI, se observa una gran variabilidad dentro del conjunto de datos. Como se ha descrito anteriormente, la ausencia de una duración estándar en los conjuntos de datos puede constituir un problema para este tipo de modelos.

  La duración promedio de los *clips* es de aproximadamente 25 compases (Figura 5.4). No obstante, existen clips con longitudes muy diversas, que van desde los 8 hasta los 300 compases, lo que podría afectar al entrenamiento del modelo.

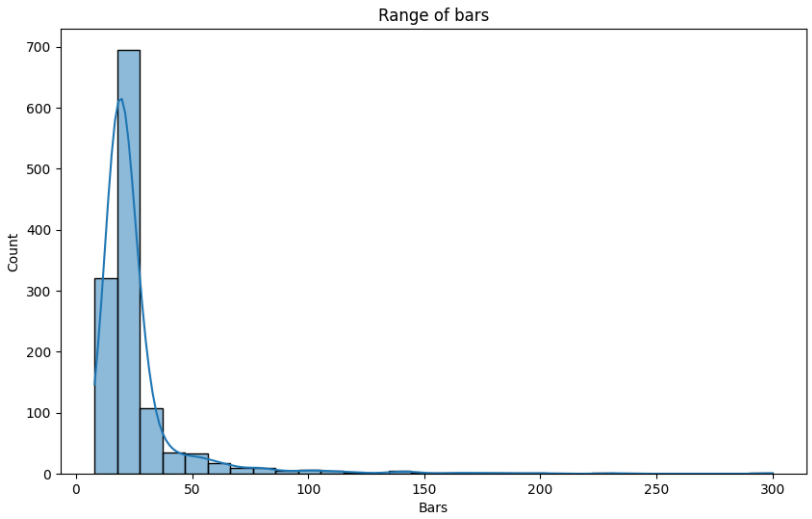

Figura 5.4: *Histograma de la duración de los clips en compases.*

- **Organización por pistas**

  Los archivos MIDI que conforman este conjunto de datos contienen la información distribuida en diversas pistas. Al tratarse de información de piano, la mayoría de los archivos comprimen la información en una sola pista (Figura 5.5). No obstante, otra parte se organiza en dos pistas, separando la información de la mano izquierda y la de la mano derecha. Habitualmente la primera está asociada con los acordes y la segunda con la línea melódica.

  Algunos archivos disponen ocasionalmente de más de tres pistas, alcanzando hasta ocho pistas en un único caso. Aunque sean una minoría, estos archivos contienen información importante, incluyendo múltiples líneas melódicas en diversos instantes y pistas destinadas únicamente a la configuración del archivo MIDI, como la información de *ticks per beat*.

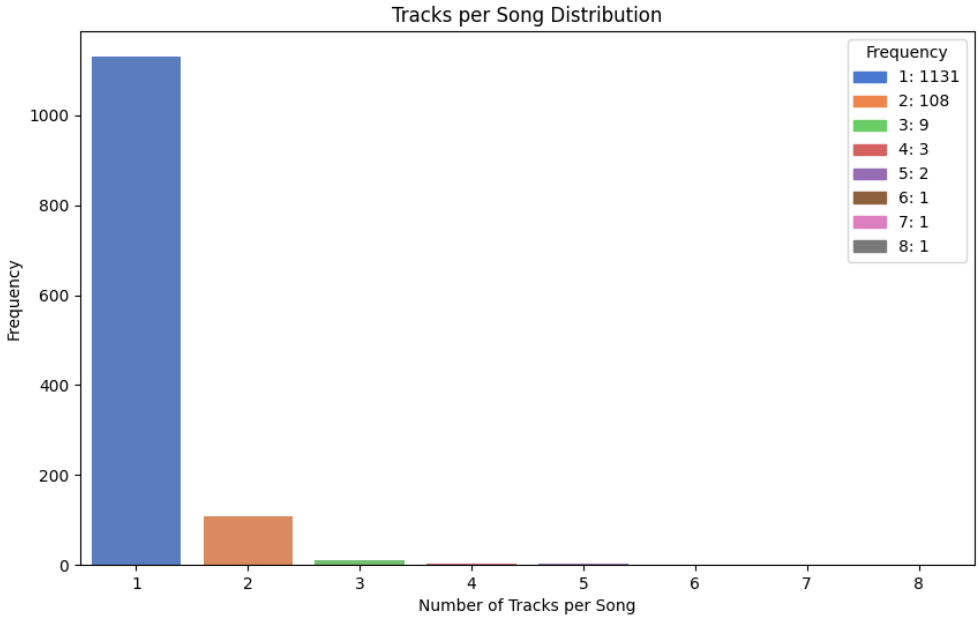

Figura 5.5: *Distribución del número de pista por clip.*

- ***Tempo***

  Los archivos MIDI utilizan los *ticks* como la unidad mínima de tiempo dentro del archivo. Cada mensaje MIDI incluye un *tiempo delta*, que indica la cantidad de *ticks* transcurridos desde el último mensaje. En la librería Mido utilizada para el preprocesamiento de los datos MIDI, existe la variable `ticks_per_beat`, que establece un valor fijo de *ticks* por pulso (*beat*) para todo el archivo.

  Este parámetro tiene una relación directa con el *tempo* del archivo y puede influir en la percepción de la emoción de cada *clip*. Al igual que sucede con los parámetros previamente analizados, el dataset dispone de una gran variedad de valores (Tabla 5.1).

Ticks per beat	Frecuencia
48	12
96	3
120	18
192	70
256	19
384	1072
480	14
1024	59

Tabla 5.1: *Distribución de ticks per beat.*

- **Compás**

  En Mido, la información del compás se obtiene mediante un mensaje de metadata denominado `time_signature`, incluyendo los parámetros *numerator* y *denominator*. En el conjunto de datos utilizado existe una gran variedad de compases que influyen en la musicalidad de la pieza, abarcando desde compases de subdivisión binaria y ternaria, hasta compases compuestos (Tabla 5.2).

Compás	Frecuencia
4/4	1373
5/4	79
3/4	76
2/4	26
6/4	21
2/2	14
6/8	12
7/8	11
9/8	7
7/4	5
5/8	7
11/8	4
1/4, 1/8, 11/16, 12/8, 17/16, 19/16, 1/32,1/8	1

Tabla 5.2: *Frecuencia por compás.*

- **Tono**

  El formato MIDI es capaz de codificar un total de 128 notas como se ha presentado en la sección 4.1, con una numeración específica para cada una de ellas. Esta numeración comienza en la nota MIDI 0 (C-2, 8.18Hz) y se extiende hasta la nota MIDI 127 (G9, 12543.85 Hz). El espectro audible del oído humano está comprendido entre los 20 Hz y los 20 kHz. Por este motivo, las notas situadas en los extremos del rango MIDI pueden no ser audibles directamente, aunque las frecuencias inferiores a 20 Hz pueden influir en la percepción sonora.

La distribución de las notas en el conjunto de datos sigue un patrón normal, con una moda próxima a la nota MIDI 62 (D4, 293.66 Hz), situada en la zona central del piano (Figura 5.7). La distribución se inicia alrededor de la nota 22 y se extiende hasta la nota 105. Sin embargo, y aunque no se aprecia fácilmente en el gráfico, fuera de este rango aparecen algunas notas aisladas, principalmente por debajo de la nota 20.

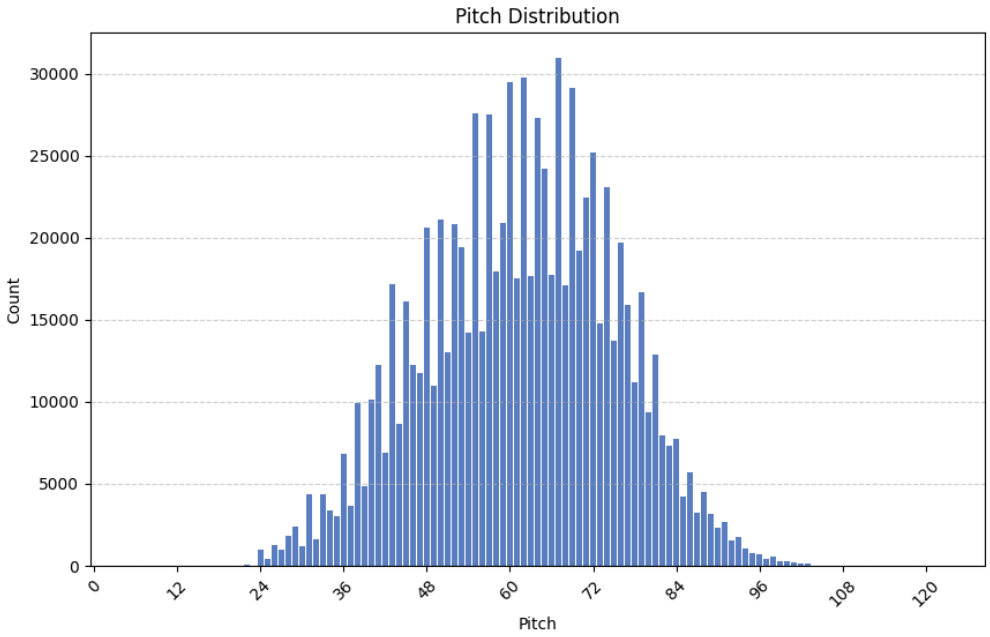

Figura 5.6: *Distribución de los tonos.*

- **Duración y posición de las notas**

  En el ámbito musical, las notas tienen duraciones establecidas, como negras, blancas y redondas, cada una con una duración numérica concreta dentro del compás y una posición específica. Sin embargo, cuando se interpretan en la realidad estas piezas, estas duraciones y las posiciones no se ejecutan con precisión absoluta, ya que los músicos acometen leves variaciones respecto a las indicaciones cualitativas presentadas en las partituras.

  Esto se observa en algunos archivos de nuestro conjunto de datos, ya que seguramente fueron obtenidos a partir de interpretaciones en directo de la obra. Esto aporta una musicalidad y una personalidad propia a cada pieza. Sin embargo, la casuística de duraciones y posiciones posibles se extiende significativamente.

5.2.2. Procesamiento de archivos MIDI

Como se ha comentado en la sección 5.2.1, el conjunto de datos seleccionado dispone de una gran diversidad de características MIDI. Esta variedad aporta musicalidad y personalidad a los archivos MIDI. Sin embargo, para este proyecto, se busca reducir la complejidad del vocabulario introducido, y por ende, alcanzar un mejor entrenamiento y rendimiento del modelo

desarrollado.
Por ello, se han realizado las siguientes operaciones para homogeneizar los archivos MIDI.

- **Segmentación por compases**

  Las duraciones en los archivos MIDI son muy diversas, con algunos casos extremos que pueden alcanzar los 300 compases (Figura 5.4). Este escenario de duraciones tan diversas puede afectar tanto al entrenamiento como al rendimiento del modelo.
  Por ello, se ha decidido dividir todos los *clips* en segmentos de 8 compases, correspondiente a la duración mínima de los archivos del conjunto. Esta decisión favorece al modelo en dos aspectos: primero, se homogeneiza la duración de los clips, y segundo, aumenta considerablemente el número de muestras disponibles para el modelo.

- **Selección del compás**

  La mayoría de la música no contemporánea utiliza compases de subdivisión binaria (como 2/4, 3/4 , o 4/4), así como compases de subdivisión ternaria (como 6/8, 9/8 o 12/8). También existen otros compases menos usuales, en el que se incluyen pulsos adicionales o variantes como el 2/2, siendo el compás predominante el 4/4 (Tabla 5.2).
  El análisis previo muestra la presencia de nueve *clips* con más de un compás por archivo, habitualmente combinaciones de compases menos usuales, situados en la parte inferior de dicha tabla.
  Con el objetivo de acortar el número de compases posibles en el vocabulario de entrada del modelo, se ha optado por restringir a únicamente diez compases (Tabla 5.2).

Compás	Compás
2/2	5/8
2/4	6/8
3/4	7/8
4/4	9/8
5/4	6/4

Tabla 5.3: *Compases utilizados*

- **Normalización del rango tonal**

  La distribución de tonos está comprendida principalmente en el rango de *pitch* entre 21 y 108, con alguna presencia esporádica fuera de estos límites (Figura 5.7).
  Con el fin de acortar el rango predeterminado por Mido (0-128) y, por tanto, para reducir la longitud del vocabulario de entrada del modelo, se ha decidido acortar el rango a 20-120. Aunque este rango es un tanto superior al del conjunto analizado (Figura 5.7), se amplía para contemplar posibles variaciones derivadas de la aplicación de técnicas de *Data augmentation*. Estas técnicas, como transposiciones y modificaciones tonales, pueden generar notas fuera del rango original, siendo conveniente de esta manera prever

un margen adicional.
Como resultado, todos los archivos que contengan notas fuera del rango establecido serán eliminados, garantizando una mayor uniformidad y facilitando el proceso de entrenamiento.

- **Cuantización de la duración y posición**

  Cuantizar es el proceso de alinear las notas de un archivo a la posición más cercana dentro de una cuadrícula predeterminada. A pesar de estar diseñado para corregir imprecisiones, puede reducir la personalidad o musicalidad de la pieza. Este proceso puede afectar tanto al inicio de los eventos de audio/MIDI como a su duración.
  Como se comentaba en el último punto de la sección 5.2.1, algunos de los archivos del conjunto de datos presentan imprecisiones tanto en el inicio del evento, como en su duración. A pesar de que estas variaciones aportan un *groove* único a cada pieza, también aumentan el número de posibles duraciones y posiciones. Por ello, se cuantizan los compases en 32 segmentos, acortando significativamente el vocabulario asociado con las duraciones e inicios de los eventos. De esta manera, se establece una duración mínima de semicorchea y se limita el inicio de todas las notas a 32 cuadrículas por compás.

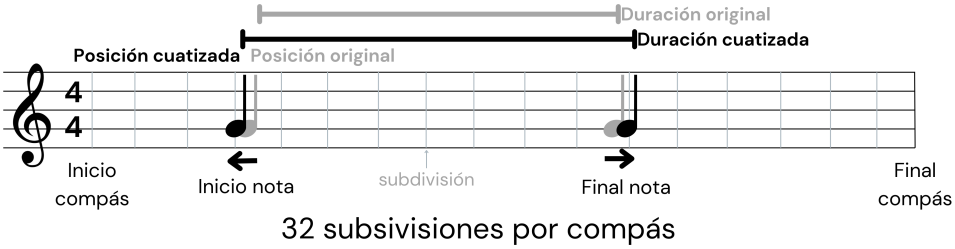

Figura 5.7: *Cuantización realizada.*

5.2.3. División del conjunto de datos

La calidad del entrenamiento del modelo viene condicionada en gran medida por la división del conjunto de datos en dos subconjuntos, entrenamiento (*train*) y test. Esta partición se realiza para garantizar que el modelo aprenda los patrones generales del conjunto de entrenamiento. El conjunto de test se utiliza también para evaluar el rendimiento del modelo frente a datos no vistos previamente. De esta manera, se pretende evitar un sobreajuste del modelo, asegurando que el modelo no solo memorice los datos del conjunto de entrenamiento, sino que también sea capaz de generalizar.
Concretamente, se ha elegido una división 90-10, destinando el 90 % de los datos para entrenamiento y el 10 % para test. Esta estrategia es óptima para casos cuando la cantidad de datos disponibles es limitada. Se garantiza así que el modelo cuente con la mayor cantidad de datos posibles para aprender patrones, sin olvidar la importancia de un conjunto de test para evaluar su rendimiento.

5.2.4. Ampliación de datos para archivos MIDI relacionados con emociones

Como se presenta en la sección 4.3, existen múltiples técnicas de ampliación de datos en el campo de la música, con el objetivo de ampliar y diversificar el conjunto de datos de entrenamiento.
Sin embargo, en el caso de este proyecto, es delicado aplicar algunas de estas técnicas, ya que al trabajar con datos relacionados con emociones, ciertas modificaciones pueden alterar dicha asociación. Un ejemplo claro se puede apreciar en la Figura 5.3, donde se muestra una relación directa entre la concentración de notas y la emoción. Esto limita el uso de técnicas de *data augmentation* que impliquen la adición o eliminación de notas, ya que podría verse afectada la percepción emocional.
De la misma manera, la modificación del *tempo* puede influir en la percepción de la emoción. Concretamente, las composiciones con tempos más lentos suelen estar asociadas a emociones correspondientes a los cuadrantes tres y cuatro del modelo de Russell.
Por lo tanto, se concluye que la única técnica de ampliación de datos que puede tener un mayor interés para este proyecto es el **cambio del tono**, aplicándose únicamente en intervalos de un número par de semitonos. Esta restricción tiene una justificación musical, ya que la conversión de modo mayor a menor está relacionada con transposiciones de un número impar de semitonos, lo que podría modificar la percepción de la emoción.
Por ello, se aplican transformaciones de *clips* dentro del mismo modo, es decir, las piezas de tonalidad mayor pasarán a otras tonalidades mayores, y las menores, a otras menores. En el código del proyecto, se habilita la transposición en intervalos de tonos según la cantidad de nuevos *clips* que se quieran obtener a partir del original, alternando tonos hacia arriba y hacia abajo de manera progresiva.

5.2.5. Tokenización

Como se resumió en la sección 4.4, *Revamped MIDI* (REMI) es la representación más extendida en formato MIDI, debido fundamentalmente a su buen comportamiento con diferentes líneas melódicas. A pesar de tratarse de la representación más extendida, no se contempla para la representación de varias líneas melódicas. Esta carencia se puede solucionar añadiendo un *token* adicional de duración. Con esta modificación, se considera que la representación resulta más adecuada para el desarrollo de este proyecto.
La tokenización desarrollada en este trabajo está compuesta por las siguientes etiquetas (Figura 5.8):

1. **Emoción (1-4)**. Primera etiqueta de la secuencia de *tokens*, correspondiente al condicionamiento de la generación. Cada valor se corresponde a un cuadrante del modelo de Russell.

    - **1**. Asociado a emociones de alta activación y valencia positiva, como excitado o feliz

- **2**. Asociado a emociones de alta activación y valencia negativa, como frustrado o enojado.
- **3**. Asociado a emociones de baja activación y valencia negativa, como triste o deprimido.
- **4**. Asociado a emociones de baja activación y valencia positiva, como relajado o satisfecho.

2. **Tempo (5-13)**. Representa los 9 valores de *ticks per beat* (Tabla 5.1).
3. **Compás (14-23)**. Correspondiente a los diez posibles valores de compás (Tabla 5.3).
4. **Inicio de compás (24)**. Establece el comienzo de cada compás.
5. **Posición (25-56)**. Indica el inicio de una nota cuantizada con respecto al comienzo del compás, con 32 posibles valores.
6. **Duración (57-88)**. Establece la duración de las notas cuantizadas, de acuerdo a las 32 subdivisiones de cada compás.
7. **Tono (89-178)**. Representa los 90 valores de tonos en el rango MIDI de 20 a 109.

Además, se añaden tres etiquetas adicionales correspondientes al *Start of Sentence* (SOS), *Padding* (PAD) y *End of Sentence* (EOS), con los valores 0, 179 y 180, respectivamente.
A la representación REMI, además de introducir las etiquetas emocionales, se le añaden etiquetas de duración para solucionar los problemas de polifonía de esta representación, eliminando aquellas asociadas a la velocidad MIDI. Aunque esta decisión pueda afectar a la musicalidad de las piezas, se considera útil debido al tamaño limitado del conjunto de datos.
Además, a diferencia de la tokenización REMI, las etiquetas de *tempo* y compás sólo aparecen al comienzo de cada secuencia, ya que en el preprocesamiento se han descartado los archivos con múltiples cambios de compás y tempo.
De esta manera, la secuencia de *tokens* comenzará con SOS, seguido de emoción, tempo y compás, continuando con la etiqueta de inicio de compás y las correspondientes a cada nota (posición, duración y tono).

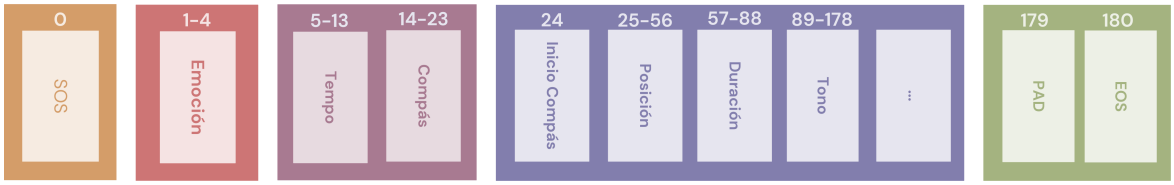

Figura 5.8: *Estructura de la tokenización MIDI basada en REMI.*

5.3. Modelo implementado

Anteriormente se ha proporcionado una idea general del funcionamiento de las herramientas generativas, particularizando en el campo de la música generativa simbólica condicionada

(Ver secciones 3 y 4). Una vez realizados los dos primeros pasos de una metodología basada en un *Machine Learning pipeline*. Como son la recopilación y el preprocesamiento de los datos, así como la extracción de sus características, es necesario establecer ahora la arquitectura de red.

Como se presentaba en las secciones 4.2 y 3.1, existen diversas arquitecturas de aplicación en proyectos de estas características. Entre ellas se encuentran las LSTM, habitualmente utilizada para procesar secuencias de texto o series temporales, las *Autoencoder* variacionales o *Variational AutoEncoders* (VAE), ideales para la generación de muestras de audio e imágenes mediante un conjunto formado por *encoder* y *decoder*, y las Red Generativa Antagónica o *Generative Adversarial Network* (GAN), que generan datos realistas mediante el enfrentamiento de dos redes neuronales que compiten entre sí. De la misma manera, los *Transformer* destacan por mejorar el entendimiento entre relaciones temporales en una secuencia, frente a arquitecturas previas como LSTM o *Convolutional Neural Network* (CNN). Incluso existen arquitecturas más recientes, como los *Transformers* basados en VAE, que combinan la atención a largo plazo con la representación del espacio latente.

Todas estas arquitecturas son las más adecuadas para este proyecto. Sin embargo, se considera que una arquitectura basada en **Transformers** es la ideal para alcanzar los objetivos de este proyecto. Esto se debe a su mayor capacidad por capturar dependencia a largo plazo de reconocimiento de patrones complejos, realizar generación condicionada y ofrecer un mejor rendimiento tanto en precisión como en tiempos de inferencia, debido a la paralelización de procesos.

A pesar de todas estas ventajas, los *Transformers* son una arquitectura que necesita de grandes volúmenes de datos para un correcto entrenamiento. Por ello, es común emplear técnicas como *Retrieval-Augmented Generation* (RAG) o *fine-tuning* en *Large Language Model* (LLM), aprovechando el conocimiento general de modelos como Llama, GPT, Gemini, Mistral, BERT o Falcon y adaptándolos al conjunto de datos en cuestión. Esta opción es la más extendida e idónea en cuanto a resultados y rendimiento. No obstante, en este proyecto se implementa un arquitectura **Transformer desde cero**, a pesar de saber que los resultados no igualarán los obtenidos con modelos pre-entrenados. Esta decisión viene motivada por profundizar en la comprensión de la arquitectura y su funcionamiento durante el desarrollo y entrenamiento.

Concretamente se implementa un ***Transformer* autoregresivo basado en un *decoder***, ya que se pretende generar secuencialmente a partir de una emoción los siguientes *tokens*. En la Figura 5.9 resumen el diagrama de clases del modelo y a continuación se detallan las características de cada uno de sus bloques.

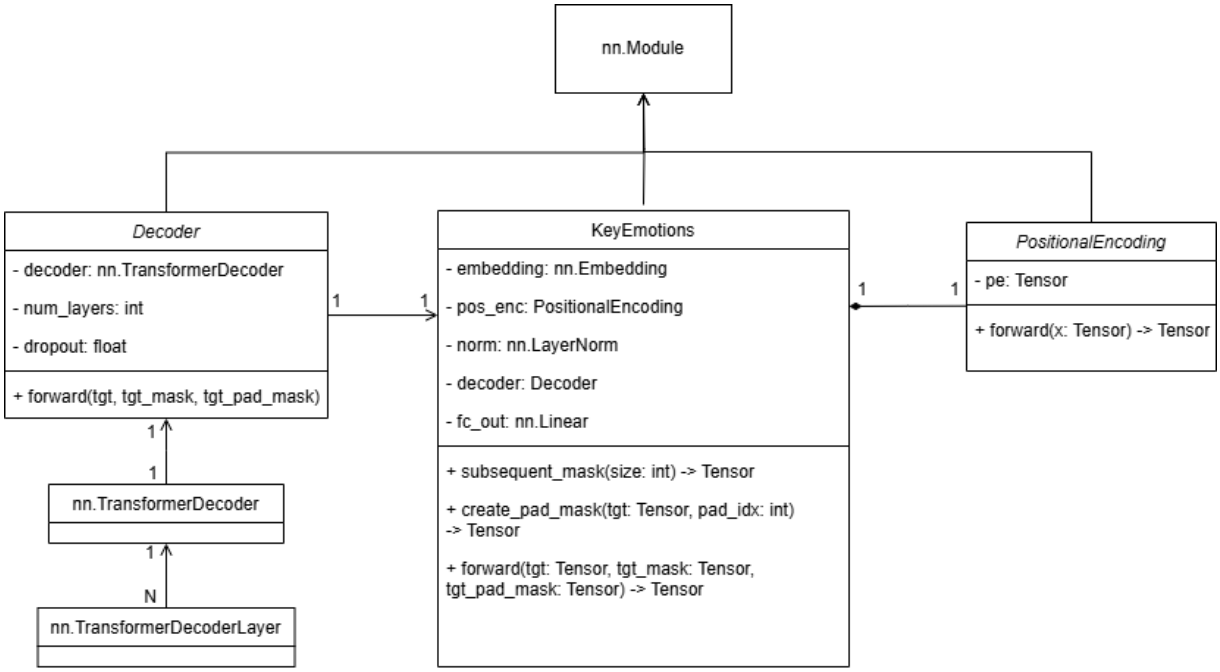

Figura 5.9: *Diagrama de clases del modelo.*

5.3.1. *Transformer* Autoregresivo

KeyEmotions se corresponde a la clase principal del modelo, siguiendo la estructura típica de un *transformer* autoregresivo basado en un *decoder*, incluyendo capas de normalización, enmascaramiento y una proyección final para la generación. Esta clase se hereda de `nn.Module`, la base en PyTorch para definir redes neuronales, permitiendo organizar las capas y operaciones de manera modular y reutilizable.
Concretamente esta arquitectura se inicializa con los siguientes componentes:

- **Embeddings**, heredado de `nn.Embedding`, encargándose de la conversión de índices de *tokens* en representaciones densas con un tamaño establecido.
- **Codificación Posicional**, incorporando información sobre la posición de cada *token*, al carecer la arquitectura *Transformer* de una estructura secuencial.
- ***Decoder***, compuesto por sus respectivas capas de autoatención, *feedforward* y normalización.
- **Capa de salida**, proyectando la salida del decodificador sobre el vocabulario establecido para predecir la secuencia de *tokens*.
- **Normalización (`LayerNorm`)**, reduciendo la varianza de los datos, estableciendo un escala común para todos ellos.
- **Dropout**, evitando el sobreajuste durante el entrenamiento al anular aleatoriamente cierto porcentaje de las conexiones de la red.

Una vez inicializadas las capas, se establece el flujo de trabajo del modelo en el método `forward`, que se compone de los siguientes bloques.

1. **Entrada**. Las secuencia objetivo (`tgt`) se procesa mediante las capas de *embedding* y codificación posicional.
2. **Decodificación**. Se pasa a través de múltiples capas de autoatención y *feedforward* para obtener una representación.
3. **Máscaras**.
   - **Máscara causal** (triangular superior) para evitar que un *token* acceda a información futura.
   - **Máscara de *padding*** para evitar el procesamiento de posiciones de relleno.
4. **Salida**. Formada por una capa lineal para convertir la representación generada por el *decoder* en probabilidades sobre el vocabulario.

Por último, todos los parámetros de las capas lineales y de *embedding* se inicializan mediante una distribución de *Xavier Uniforme*, asegurando así una distribución adecuada de los pesos y favoreciendo una mejor convergencia durante el entrenamiento.

5.3.2. *Decoder*

El objetivo del decodificador, llamado por la clase principal `KeyEmotions`, es obtener la secuencia de salida de manera autoregresiva, prediciendo cada *token* de manera individualizada a partir de los anteriores. Para ello, se emplean mecanismos de atención enmascarada sobre la secuencia objetivo y normalización de las capas.
En primera instancia, se inicializa el constructor formado por los siguientes bloques.

1. `nn.TransformerDecoderLayer`, encargado de crear las capa individuales del decodificador, con el parámetro `batch_first` activado para optimizar el procesamiento por lotes. Cada capa incluye:
   - *Masked Multi-Head Attention*,atendiendo a la secuencia objetivo, evitando el acceso a futuros *tokens* mediante una máscara triangular.
   - *Add & Norm*, aplicando normalización de capa y conexión residual tras la autoatención.
   - *Multi-Head Attention*, estableciendo las relaciones entre la secuencia actual y los *tokens* procesados hasta el momento.
   - *Add & Norm*, aplicando normalización de capa y conexión residual tras la atención cruzada.
   - *Feedforward*, encargado de transformar la representación obtenida tras el *Multi-Head Attention* en un espacio de mayor dimensión y aplicar una activación no lineal.

- *Add & Norm*, asegurando la estabilidad en la capa de salida.

2. `nn.TransformerDecoder`, agrupando la pila de decodificadores creados previamente.

El decodificador se instancia con los siguientes hiperparámetros.

- `d_model`, dimensión de la representación de los *tokens* en el espacio latente.
- `nhead`, número de cabezales de atención por capa.
- `num_layers`, cantidad de capas apiladas en el decodificador.
- `dropout`, porcentaje de desactivación de neuronas durante el entrenamiento para evitar sobreajuste.

Después de inicializar el decodificador, se describe el procesamiento secuencial mediante el método `forward`.

1. **Entrada**
   - `tgt`, tensor de entrada con dimensiones `(batch_size, seq_len, d_model)`.
   - `tgt_mask`, máscara triangular superior para asegurar únicamente el procesamiento de posiciones previas.
   - `tgt_pad_mask`, máscara empleada para ignorar las posiciones de *padding* de la secuencia objetivo.
2. **Procesamiento.** La secuencia objetivo pasa por cada capa del decodificador en donde se aplican los siguientes procesos:
   - Se aplica atención enmascarada sobre la secuencia (*Masked Multi-Head Self Attention*).
   - Se normaliza la salida mediante *Add & Norm*.
   - Se emplea atención cruzada sobre la memoria interna (*Multi-Head Cross Attention*).
   - Se normaliza la salida de la atención cruzada.
   - Se transforma la representación mediante la red *Feedforward*.
3. **Salida.** El resultado es un tensor de las mismas dimensiones que la entrada del decodificador, expresando la predicción para cada *token* de la secuencia objetivo.

5.3.3. *Positional Encoding*

Los *transformers* no cuentan con una estructura secuencial inherente como las RNN, por lo que es necesario codificar posicionalmente cada uno de los elementos (Ver sección 5.3.1). Esto se logra mediante una codificación sinusoidal basada en la posición y la dimensión de los *embeddings*, sumándose a las representaciones originales de los *tokens*.
En este modelo se realiza una codificación posicional de acuerdo a la posición pos y la dimensión

d_{model} del *embedding*. La codificación posicional, introducida por Vaswani et al. (2023), se define mediante las siguientes funciones:

$$PE_{pos,2i} = \sin\left(\frac{pos}{10000^{2id_{\text{model}}}}\right) \tag{5.1}$$

$$PE_{pos,2i1} = \cos\left(\frac{pos}{10000^{2id_{\text{model}}}}\right) \tag{5.2}$$

donde *pos* representa la posición del *token* en la secuencia e *i* es el índice de la dimensión del *embedding*. Estas funciones permiten captar las relaciones posicionales sin necesidad de aprenderlas.
En la implementación, se calcula inicialmente un tensor de ceros para la codificación posicional. Posteriormente, se calculan los valores de seno y coseno en posiciones alternas de acuerdo a las funciones previamente presentadas, generando una matriz de dimensiones (max_length, d_model). Este tensor se almacena en un búfer mediante la función `register_buffer`, asegurando que no se vea afectado durante el entrenamiento. Finalmente, la codificación posicional se suma a los *embeddings* originales de los *tokens* previamente procesados por el modelo.

5.4. Entrenamiento

Una vez procesados los datos del conjunto y definida la arquitectura de red, es necesario entrenar el modelo con diferentes configuraciones de hiperparámetros. Este proceso permite obtener la combinación que más se adecúa tanto a la arquitectura como al conjunto de datos. El entrenamiento del modelo se gestiona en la clase `Trainer`, desde la cual se gestiona tanto la inicialización del modelo `KeyEmotions`, la carga y preparación de los datos de entrenamiento, la validación mediante la clase `Loader`, así como la configuración, el registro del entrenamiento y finalmente la validación del modelo.
La clase `Trainer` requiere de los siguientes **atributos**.

- `config_dict`, configuración del entrenamiento y del modelo a través del archivo YAML, incluyendo hiperparámetros, rutas y otros parámetros.
- `device`, dispositivo donde se ejecuta el entrenamiento.
- `data_loader_train` y `data_loader_val`, cargadores de los datos de entrenamiento y validación respectivamente.
- `model`, instanciación del modelo *transformer* autoregresivo, `KeyEmotions`.

Además, la clase `Trainer` está compuesta por los siguientes **métodos**.

- `__init__`, inicialización del entrenador con la configuración del YAML y definición del dispositivo de ejecución.
- `set_seed`, configuración de las semillas de reproducibilidad.

- `load_data`, carga de los datos de entrenamiento y validación.
- `initialize_model`, inicialización del modelo `KeyEmotions`.
- `calculate_top_k_accuracy`, cálculo de la métrica de precisión top-1 y top-5.
- `train_iter`, ejecución de una iteración del entrenamiento.
- `validate`, evaluación del modelo con el conjunto de validación.
- `train`, gestión del proceso completo de entrenamiento.

La Figura 5.10 representa el diagrama de clases del entrenamiento del modelo, lo que proporciona una visión global del entrenamiento.

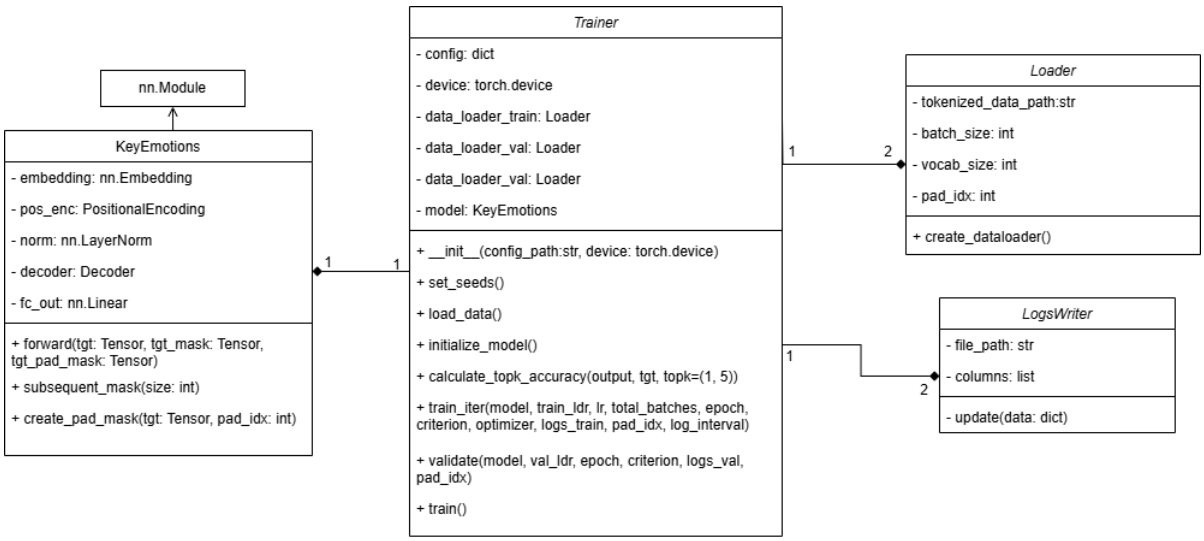

Figura 5.10: *Diagrama de clases del entrenamiento.*

Además de definir la estructura general del entrenamiento, a continuación se describen algunos de los principales mecanismos empleados para optimizar el rendimiento y mejorar la capacidad de generalización del modelo.

5.4.1. Función de pérdida

La función de pérdida en un modelo de inteligencia artificial cumple un papel fundamental, ya que se encarga de cuantificar la diferencia entre el valor real y el predicho por el modelo. Para este modelo, se emplea la función de pérdida de **entropía cruzada categórica** (`CrossEntropyLoss`).

Esta función calcula el grado de similitud de la distribución esperada y la distribución de probabilidad predicha, penalizando en mayor grado las predicciones incorrectas. Dado que el modelo trabaja con secuencias análogas al texto y emplea valores de relleno, se aplica un enmascaramiento para evitar que estos *tokens* afecten al cálculo de la pérdida.

5.4.2. Optimizador

El entrenamiento requiere de un algoritmo de optimización que ajuste los pesos de manera eficiente. En este caso, se utiliza **AdamW** (`AdamWOptimizer`), que es una variante de Adam que incorpora la regularización mediante decaimiento del peso. A diferencia de Adam, que aplica la regularización directamente a los gradientes, AdamW separa la actualización de la penalización, mejorando la generalización. Además, conserva las ventajas de Momentum y RMSprop, permitiendo una actualización estable y rápida de los parámetros, a la vez que reduce el riesgo de sobreajuste y mejora la convergencia.

5.4.3. *Scheduler* de la tasa de aprendizaje

El uso de un ajustador de la tasa de aprendizaje es fundamental en cualquier entrenamiento ya que favorece una convergencia más estable del modelo. Concretamente, para el modelo de este estudio se implementa un ***LambdaLR scheduler***, compuesto por dos fases.
Una primera fase, compuesta por el 10 % inicial de las iteraciones, donde se aplica un ***warmup***, incrementando lentamente la tasa de aprendizaje desde un valor mínimo hasta el objetivo. Esto permite estabilizar el entrenamiento durante las primeras etapas. A continuación, se aplica la tasa de aprendizaje predefinida. Esta estrategia permite un control más flexible y adaptativo del aprendizaje, al mejorar la capacidad de exploración y generalización del modelo.

5.4.4. *Early Stopping*

El mecanismo de detención anticipada sirve para detener el entrenamiento cuando el modelo deja de mostrar mejoraría en el conjunto de validación, evitando el sobreajuste. Ésta técnica monitoriza la métrica de desempeño durante varias épocas consecutivas, deteniendo el entrenamiento en caso de observar mejoría. Este número de iteraciones viene definido por el parámetro `patience`. Esta estrategia ayuda a optimizar el uso de recursos computacionales y evita un entrenamiento innecesario.

5.5. Evaluación

El último paso de la metodología de proyectos empleada es la evaluación del modelo, en donde se mide el desempeño de la tarea para la que ha sido diseñado. Para ello, se utilizan diversos indicadores que muestran la calidad de la música generada. En este caso, es necesario evaluar los siguientes aspectos:

- **Coherencia musical**, verificar que sigue una estructura armónica, melódica y rítmica coherentes.
- **Expresividad emocional**, determinando si la música refleja adecuadamente la emoción objetivo.
- **Diversidad de las composiciones**, analizando si el modelo repite estructuras o genera variedad en sus generaciones.

- **Eficiencia del entrenamiento**, para medir la velocidad de convergencia y la estabilidad del modelo durante el aprendizaje.

Esta evaluación se realiza gracias al menú de usuario, donde se puede seleccionar tanto la ubicación de los pesos, la configuración del modelo, el directorio de salida y la emoción a generar (Figura 5.11).

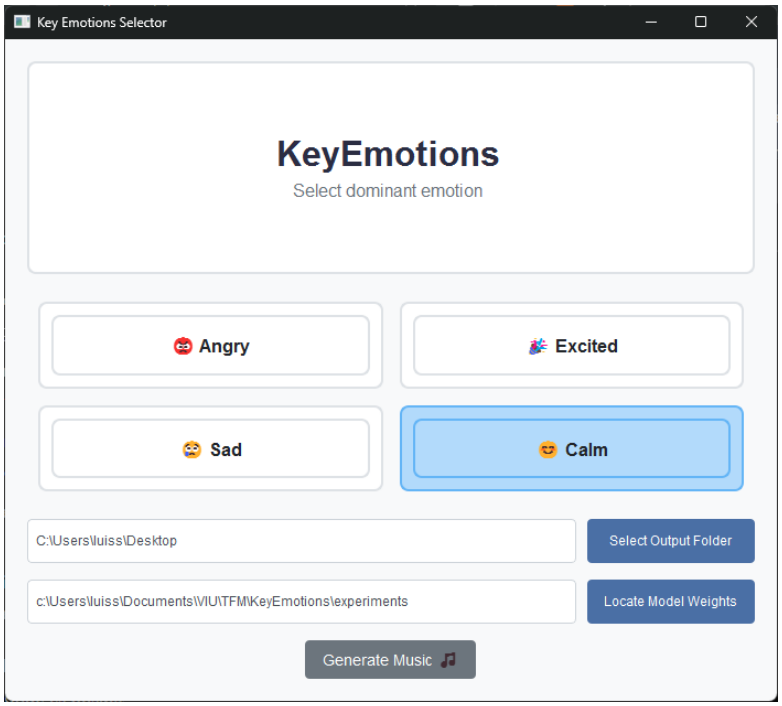

Figura 5.11: *Interfaz de usuario para la generación.*

Por todo ello, es necesario emplear las dos evaluaciones siguientes.

5.5.1. Evaluación objetiva

Esta evaluación se basa en métricas cuantificables y automáticas, permitiendo analizar tanto la calidad de la generación como la eficiencia del entrenamiento. Para valorar correctamente el rendimiento del modelo es necesario analizar individual y colectivamente los siguientes parámetros.

- **Duración por lote (`ms/batch`)**, tiempo promedio, en milisegundos, que tarda el modelo en procesar un lote de datos. Esta métrica sirve para evaluar la eficiencia computacional del modelo.
- **Pérdida (`loss`)**, muestra el error del modelo durante el entrenamiento y su validación posterior.
- **Pérdida de perplejidad (`ppl`)**, representa cuan bien predice el modelo la nota siguiente. Valores bajos indican predicciones más precisas y con menor incertidumbre en la generación.

- **Precisión del Top-1 (`top1_acc`)**, porcentaje del número de veces que el valor con mayor probabilidad coincide con la etiqueta correcta. Este indicador muestra la capacidad del modelo para acertar con la mejor opción en cada iteración.
- **Precisión del Top-5 (`top5_acc`)**, porcentaje de veces en que la etiqueta correcta está dentro de las cinco con mayor probabilidad. Es de gran utilidad para evaluar si el modelo genera opciones razonables aunque no siempre acierte con la primera posición.

5.5.2. Evaluación subjetiva

En esta evaluación, a diferencia de la objetiva, se hace hincapié en la percepción humana de las composiciones generadas. Dado que el objetivo del modelo es generar música en base a emociones, es crucial evaluar aspectos cualitativos que no pueden ser considerados por métricas automáticas.

Para ello, se evalúan personalmente los resultados según criterios como la coherencia, expresividad emocional, diversidad de las composiciones y calidad percibida. De esta forma se evalúan aspectos musicales como la armonía, la estructura musical, la concentración de notas, el *tempo* y las líneas melódicas, entre otros.

Además, se realiza un cuestionario dirigido tanto a personas con y sin formación musical para ampliar las evaluaciones personales del modelo. Este cuestionario tiene como objetivo evaluar tanto la percepción de la emoción transmitida como la calidad de la música generada.

Este tipo de evaluación es fundamental para determinar la efectividad del modelo en la generación de música y más concretamente, en su capacidad de expresión emocional.

Resultados y Discusión

6

En esta sección se pretende evaluar detalladamente el rendimiento del modelo diseñado, tanto mediante métricas objetivas como subjetivas. El objetivo es analizar el modelo bajo distintas configuraciones de hiperparámetros para identificar así la mejor combinación.
Para ello, se ajustan algunos de los siguientes hiperparámetros tanto del modelo como del entrenamiento.

- `n_head`, número de cabezas de atención del mecanismo de *multi-head attention*.
- `d_model`, dimensión del espacio latente de la representación.
- `d_ff`, dimensión de la capa *feedforward*.
- `num_layers`, número de capas de *Decoder*.
- `dropout`, tasa de *dropout*.
- `lr`, tasa de aprendizaje.
- `batch_size`, tamaño del *batch*.
- `max_epochs`, número máximo de épocas de entrenamiento.

Además de los resultados expuestos en esta memoria, se ha creado la siguiente página web[1], donde se pueden escuchar algunas de las generaciones obtenidas durante este trabajo, para que el lector pueda así tener una percepción propia del alcance de este estudio.

6.1. Análisis del entrenamiento

Se han realizado múltiples experimentos con diversas arquitecturas basadas en *Transformers*, incluyendo el modelo completo, únicamente con el *encoder* y el que se ha presentado en la memoria (sólo *decoder*). Además, se realizaron nuevos experimentos para analizar la implicación de los hiperparámetros en los resultados. Estos experimentos se realizaron bajo la misma duración de *max_epoch* (10 *epochs*), permitiendo así evaluar cómo influyen los ajustes en las mismas condiciones.
En la Tabla 6.1 se presentan las diferentes configuraciones, junto con el cálculo del número

[1] Página web con los resultados: https://luissotomedina.github.io/portfolio/KeyEmotions/

de parámetros entrenables de acuerdo al modelo diseñado en PyTorch. Es importante analizar el número, ya que es un factor clave en el diseño de redes neuronales, al tener una implicación directa en la capacidad de aprender patrones complejos, el riesgo de sobreajuste o el uso de recursos computacionales. En esta tabla, se muestra como parámetros como `d_model`, `d_ff` o `num_layers` condicionan el número de parámetros entrenable. Esto se aprecia claramente en los experimentos 1, 2 y 6. De igual manera resulta llamativo el experimento 7 frente al 8, ya que se mantienen fijos el número de parámetros. Esto se debe a que según la implementación realizada, `n_head` divide la dimensión del modelo en subespacios para calcular la atención multi-cabeza, pero no añade parámetros adicionales.

Nº exp	n_head	d_model	d_ff	num_layers	dropout	batch_size	l_r	Nº parámetros
1	8	128	2.048	8	0,1	16	0,0005	5.321.397
2	8	64	2.048	6	0,1	16	0,0005	1.810.997
3	8	128	2.048	6	0,1	32	0,0005	4.002.741
4	8	128	2.048	6	0,1	16	0,0001	4.002.741
5	8	128	2.048	6	0,3	16	0,0005	4.002.741
6	8	128	1.024	6	0,1	16	0,0005	2.423.733
7	16	128	2.048	6	0,1	16	0,0005	4.002.741
8	8	128	2.048	6	0,1	16	0,0005	4.002.741
9	8	256	2.048	6	0,2	16	0,0005	9.565.877

Tabla 6.1: *Experimentos iniciales.*

Los resultados obtenidos en los experimentos iniciales presentados en la Tabla 6.1 muestran que todavía existe un margen de mejora en el rendimiento del modelo. Concretamente, en los experimentos 5, 8 y 9 se aprecia un estancamiento claro del aprendizaje en torno a la época 4 como se ilustra en la Figura 6.1. En el caso del experimento 5, el estancamiento se puede atribuir claramente debido al incremento del *dropout*. La regularización con *dropout* suele ayudar a prevenir el sobreajuste. Sin embargo, cuando se aplica un valor demasiado alto (como el 0,3 utilizado), esto puede dificultar que el modelo aprenda patrones útiles. En el caso del experimento 8, no hay una justificación clara para este comportamiento, aunque podría atribuirse a la combinación de un gradiente más ruidoso como causa de emplear un `batch_size` pequeño, de valor 16.

En el resto de los experimentos, se muestra una mejora generalizada en todas las métricas a medida que avanzan las iteraciones. No obstante, se produce un estancamiento generalizado en las métricas de precisión, tanto en Top 1 como en Top 5. En las primeras iteraciones, habitualmente hay una mayor diferencia entre ambas métricas, destacando la precisión Top 5 sobre la Top 1, lo cual es esperable dado que Top5 es una métrica más relajada. Sin embargo, a medida que avanza el entrenamiento, la diferencia se reduce, lo que sugiere que el modelo está aprendiendo a predecir con mayor confianza (Figura 6.2).

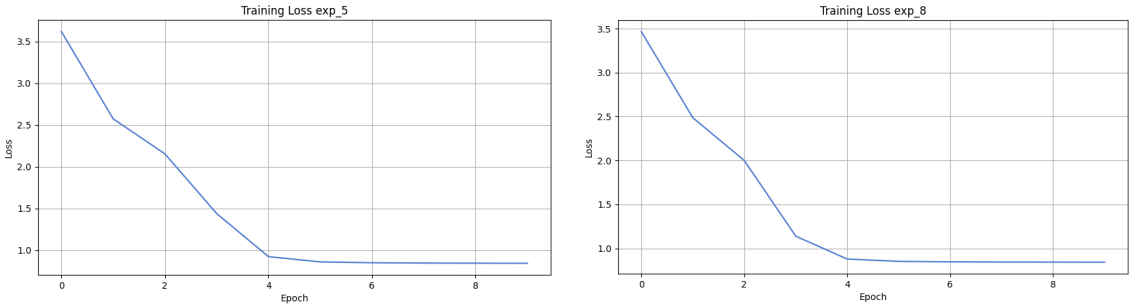

Figura 6.1: *Comparación de la métrica loss en entrenamiento entre los experimentos 5 y 8.*

A pesar de esta mejora, los valores de precisión siguen siendo relativamente bajos, dado que en la mayoría de los experimentos no superan el 0,4 en validación. Esto muestra que el modelo tiene dificultades para capturar todos los patrones de datos.

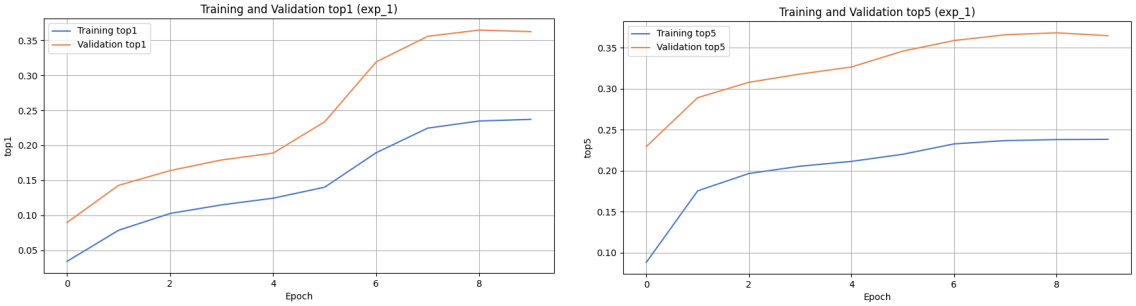

Figura 6.2: *Precisión Top 1 y Top 5 en el experimento 1.*

La mayoría de los experimentos demuestran la necesidad de utilizar un mayor número de iteraciones para completar el entrenamiento. Esto se aprecia en las últimas iteraciones, dado que en varias métricas hay una tendencia creciente en las precisiones y una descendente en el `loss` y `ppl` (Figura 6.3). El caso más evidente es el experimento 4 como se recoge en la Figura 6.3, dado que es el que comienza con la tasa de aprendizaje más baja.

En términos de consumo computacional, la mayoría de las pruebas poseen una duración promedio similar por lote, en torno a 40 ms/batch. Sin embargo, hay tres experimentos con una duración superior por lote. El primero de ellas registra una duración en torno a los 60 ms/batch, como resultado del incremento de `num_layers`. En el experimento 3, la duración promedio supera los 750 ms/batch como consecuencia del incremento del tamaño de batch. El experimento 7, por último, alcanza una duración promedio de 625 ms/batch, atribuida al incremento del parámetro `n_head`. A pesar de mantenerse el número de parámetros del modelo frente al resto de experimentos, este incremento se atribuye a la implementación de *multi-head attention* en la clase *Decoder* de PyTorch, al dividir la dimensión del modelo en subespacios.

Los nueve experimentos mostrados en la Tabla 6.1 utilizan las mismas condiciones de entrenamiento, ya que se pretende analizar el efecto de distintos hiperparámetros en el entrenamiento. Tras esta fase primera, se realizaron nuevos experimentos, incrementando el número de épocas y el tamaño de alguno de los modelos (Tabla 6.2).

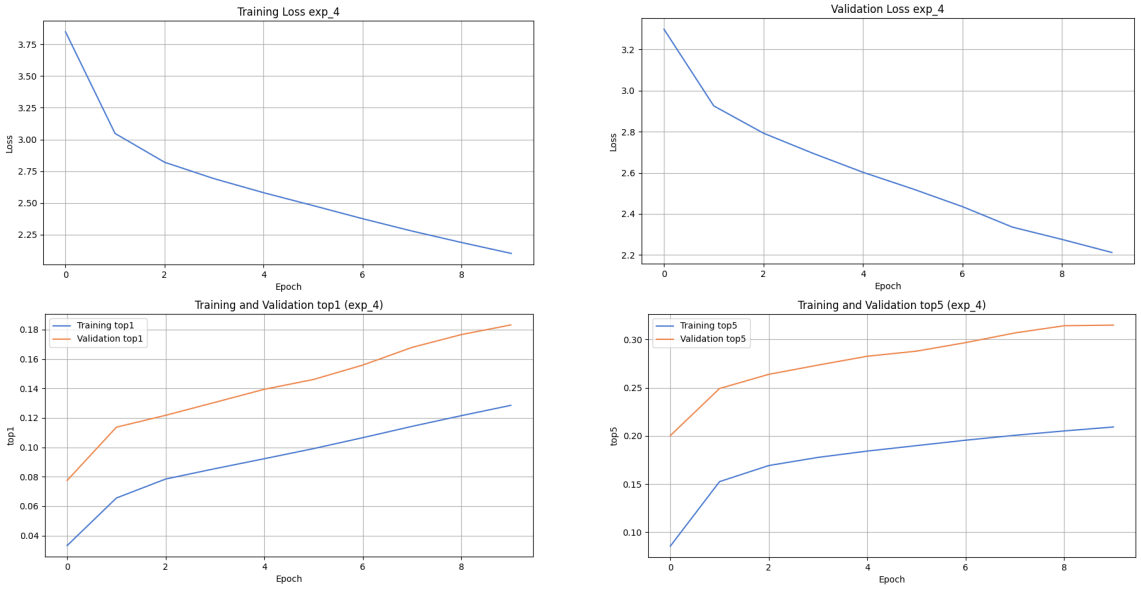

Figura 6.3: *Evaluación del experimento 4.*

Nº exp	n_head	d_model	d_ff	num_layers	dropout	batch_size	l_r	max_epochs	Nº parámetros
10	16	128	2048	8	0.15	16	0.0005	15	5.321.397
11	8	512	2048	12	0.15	16	0.0005	15	50.634.933
12	16	256	2048	12	0.15	16	0.0005	15	50.634.933
13	8	64	2048	6	0.1	16	0.0005	20	1.810.997
14	8	128	2048	6	0.1	16	0.00025	20	4.002.741
15	8	128	2048	6	0.1	16	0.0001	30	4.002.741
16	8	128	2048	6	0.3	16	0.0001	30	4.002.741
17	8	128	512	6	0.1	16	0.0001	20	1.634.229
18	8	128	512	6	0.1	16	0.0001	30	1.634.229
19	8	128	512	6	0.1	16	0.0001	40	1.634.229

Tabla 6.2: *Experimentos mejorados.*

Las características de estos nuevos experimentos se resumen en la Tabla 6.2. Existen claramente dos grupos de resultados.

En primer lugar, se encuentran los modelos más complejos (experimentos 11 y 12). A pesar de contar con un número considerablemente mayor de parámetros, no logran un mejor resultado (Figura 6.4). En esta figura se aprecia como los resultados al finalizar el entrenamiento son inferiores a los obtenidos en otros modelos, incluso con un menor número de iteraciones. Además, entre las primeras iteraciones se producen una mejora drástica en las métricas, después de esto los resultados se estabilizan.

Por otro lado, aumentar el número de parámetros implica un mayor consumo de recursos computacionales, incrementándose el tiempo promedio por batch hasta 500 y 800 ms en los experimentos 11 y 12 respectivamente. Estos resultados no tan favorables pueden atribuirse a que el modelo es demasiado grande para el conjunto de datos utilizado, imposibilitando un correcto aprendizaje de los patrones.

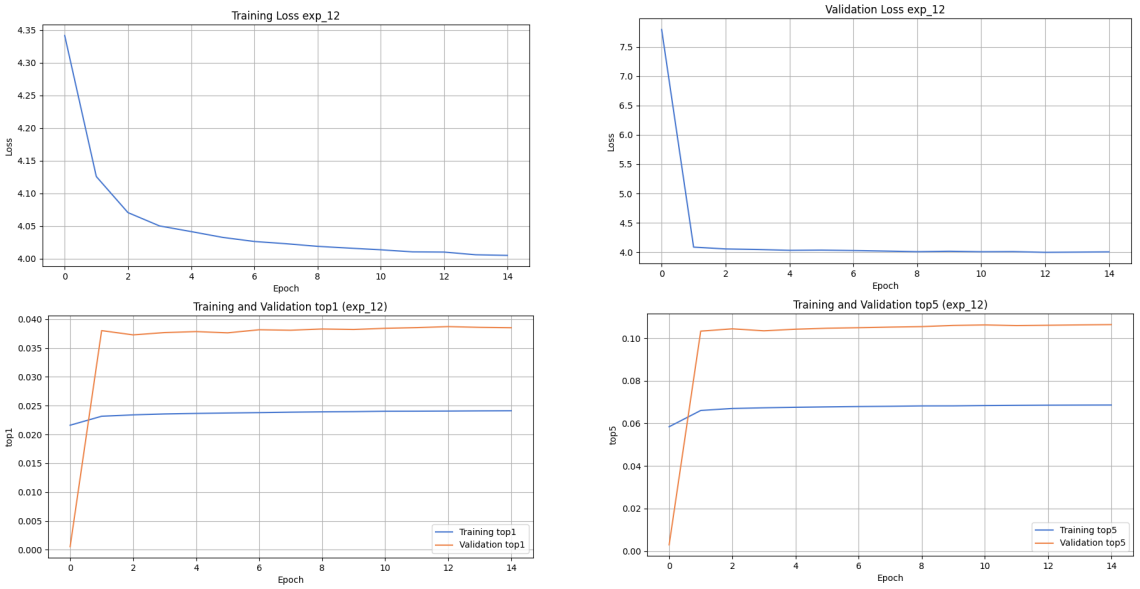

Figura 6.4: *Evaluación del experimento 12.*

En segundo lugar, se han entrenado modelos con un menor tamaño, similares a los presentados en la Tabla 6.1, pero incrementando el número de iteraciones. El objetivo ha sido definir si el aumento de tiempo de entrenamiento permite que las métricas continúen mejorando, manteniendo la tendencia positiva analizada anteriormente.

En la mayoría de los experimentos de la Tabla 6.2 se observa una mejora generalizada en todas las métricas, a pesar de la estabilización del aprendizaje en las últimas iteraciones. Por ejemplo, en el caso del experimento 13, se alcanza un *loss* cercano a 0,5 y una precisión Top 1 un tanto superior a 0,35 (Figura 6.5).

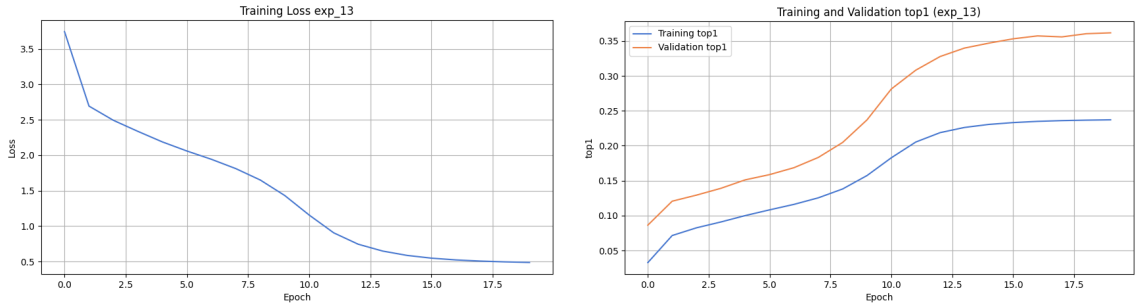

Figura 6.5: *Training Loss y Precisión Top 1 en el experimento 13.*

De la misma manera, en el experimento 19 se obtienen métricas ligeramente superiores a las del experimento 13, pero con un mayor número de iteraciones (Figura 6.6). A pesar de que estos últimos modelos alcanzan resultados muy similares en términos cuantitativos, se ha visto necesario realizar un análisis cualitativo para evaluar cómo afectan los diferentes modelos a la calidad y características de la salida.

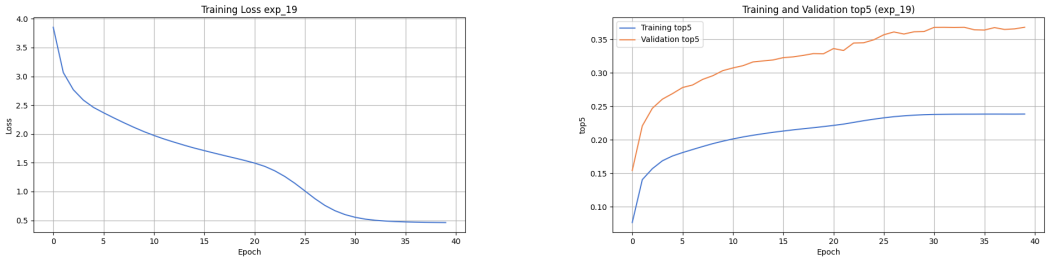

Figura 6.6: *Training Loss y Precisión Top 1 en el experimento 19.*

6.2. Perspectiva subjetiva

En este apartado se evalúan las generaciones obtenidas correspondientes a los modelos presentados en la sección 6.1. Esta evaluación se realiza ahora desde una perspectiva más subjetiva, a partir de la escucha y visualización de los resultados obtenidos. Las generaciones se realizan con una duración máxima de 8 compases, de acuerdo a las características del conjunto de datos elaborado, o con un límite de 1300 *tokens* por secuencia.

A pesar de que en la Tabla 6.2 los resultados obtenidos tanto en la métrica de pérdidas como en las precisiones son similares, las generaciones resultantes varían entre sí.

En primer lugar, es importante analizar los experimentos con un mayor número de parámetros (11 y 12). Como se muestra en la Figura 6.4, el entrenamiento no es el adecuado, ya que el mayor aprendizaje se realiza en las primeras épocas y a continuación se estanca. Las generaciones reflejan estas mismas conclusiones, ya que carecen de una línea melódica y presentan deficiencias musicales.

Estos experimentos tampoco muestran relación alguna con la emoción, ni en términos de concentración de notas ni en la presencia acordes tonalmente adecuados (Figura 6.7). Esto puede atribuirse a que, aunque el modelo tiene una mayor capacidad de aprendizaje, no dispone de suficientes datos para captar las relaciones implícitas en ellos. Estos modelos podrían ser más efectivos si se dispusiera de un conjunto mayor de datos.

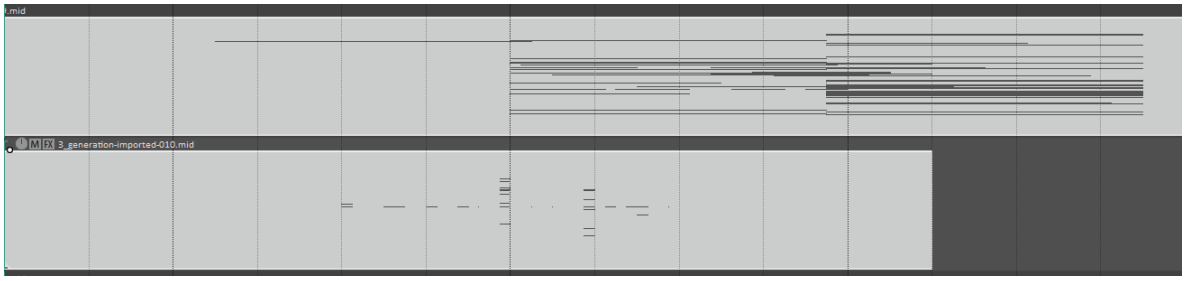

Figura 6.7: *Representación MIDI de las generaciones asociadas a la emoción del cuadrante tres en los modelos 11 y 12.*

En el resto de los experimentos (ver experimento 17 en Tabla 6.2) que utilizan modelos de menor tamaño, los resultados son ligeramente mejores. La mayoría de las generaciones presentan secciones mejor diferenciadas entre sí. No obstante, las generaciones tienden a momentos

de alta concentración de notas seguidos por una segunda sección con una distribución más dispersa (Figura 6.8).
En términos de expresividad emocional, estos modelos suelen reflejar mejor la relación entre la emoción y la concentración de notas. La Figura 6.8 muestra como ejemplo que la mayor parte del *clip* generado posee una alta densidad de notas, a diferencia de lo obtenido en la emoción del cuadrante 4 (Figura 6.9).
Sin embargo, la relación entre coherencia tonal y emoción resulta difícilmente identificables. Las generaciones correspondientes a los cuadrantes 1 y 4 no presentan tonalidades mayores, ni las de los cuadrantes 3 y 4 muestran tonalidades menores.
Por otro lado, en la mayoría de estas generaciones, la predicción de los *tokens* asociados al compás y especialmente al *tempo*, es mas preciso. Se muestra una tendencia en generar *tempos* más acelerados para las emociones de los cuadrantes 1 y 2, frente a los cuadrantes 3 y 4.

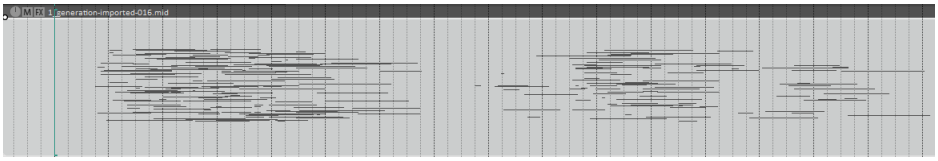

Figura 6.8: *Representación MIDI de las generaciones asociadas a la emoción del cuadrante uno en el modelo 19.*

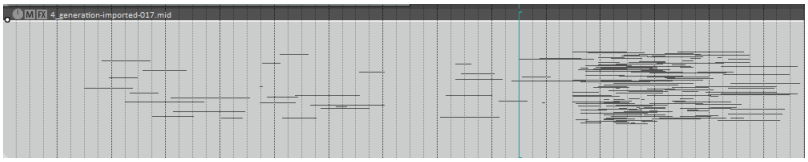

Figura 6.9: *Representación MIDI de las generaciones asociadas a la emoción del cuadrante cuatro en el modelo 19.*

Cabe resaltar que el modelo con una mejor generación musical y de percepción emocional es el experimento 19 (Tabla 6.2). Este modelo, a pesar de tener un menor tamaño en comparación al resto de los experimentos, logra distinguirse por la presencia de diversas secciones musicales y tener algo más de coherencia musical. Además, su calidad emocional es algo superior a la del resto de modelos.
Por último, y con vistas al futuro, en el apéndice A se presenta una encuesta diseñada para valorar cuál es la percepción humana sobre la musicalidad y la expresividad emocional del modelo. Esta encuesta podría ser utilizada para el análisis previo realizado únicamente por este autor.

6.3. Discusión

Además de profundizar en los aspectos técnicos de la IA generativa de música simbólica, en la sección 3.3 se han valorado algunas de las implicaciones éticas de estas tecnologías. Uno de los aspectos destacados en esa sección era el impacto ambiental que tienen las herramientas

de IA generativa. Esto es debido al consumo energético de los procesos de entrenamiento e inferencia. Anthony et al. (2020) resaltaba la importancia de complementar las métricas convencionales de los modelos con otras que evalúen su huella ambiental.

Por ese motivo, aún sabiendo que el impacto de los entrenamientos realizados en este trabajo es significativamente menor que el de los grandes modelos de IA generativa, se ha considerado relevante analizar su consumo energético.

El proceso de entrenamiento ha supuesto un total de 288 horas de computación. La tarjeta gráfica utilizada (NVIDIA RTX 4070) tiene un consumo energético en carga de entre 190 y 240 W. Sumando el consumo energético del resto de componentes del equipo, se calcula que la fase de entrenamiento ha empleado una potencia de 370 a 400 W. Esto se traduce en un consumo energético estimado de 115 kWh durante todo el proceso de entrenamiento. Considerando que la generación de electricidad a partir de combustibles fósiles emite entre 0,4 a 0,9 kg de CO_2 por kWh (Asociación Española de Normalización (UNE), 2019), el impacto ambiental global de este estudio ha supuesto una emisión de entre 46,1 a 103,7 kg de CO_2. De haber utilizado configuraciones de *hardware* más potentes, el impacto medio ambiental hubiera sido mayor, aunque el tiempo de computación se habría reducido sustancialmente.

Conclusiones

7

1. **Condicionamiento de los modelos *transformer* a los grandes conjuntos de datos**. La falta de datos etiquetados con información de emoción dificulta el proceso de aprendizaje de estas arquitecturas. A pesar de que se ha ampliado el conjunto de datos inicial mediante trasposición y segmentación de clips MIDI, sigue siendo insuficiente para que el modelo aprenda desde cero como es la musicalidad y establezca una relación con las emociones. Este hecho recalca la necesidad de ampliar los datos etiquetados emocionalmente, ya sea mediante procesos manuales o automatizados.

2. **Necesidad de un condicionamiento continuo**. Aunque los archivos MIDI pueden estar etiquetados por una emoción predominante, en detalle se aprecian variaciones en distintas de sus secciones. Esto puede afectar a la capacidad del modelo para entender la relación entre el condicionamiento emocional y los *tokens* MIDI, por lo que aplicar un enfoque de condicionamiento continuo podría resultar más positivo.

3. **Relación entre el tamaño del modelo y el conjunto de datos**. Los resultados muestran que incrementar el tamaño del modelo no garantiza mejores resultados. Modelos más pequeños logran una mejor comprensión del conjunto de datos limitado, mientras que modelos con "mayor capacidadrequieren volúmenes de datos muy superiores para comprender sus relaciones.

4. **Dificultades de entrenamiento de modelos generativos**. Los modelos generativos, y más concretamente los *transformer* creados desde creo, requieren una gran capacidad computacional y un volumen considerable de datos. A pesar de tener conocimiento previo de esta problemática, se ha confirmado cuan importante es dimensionar correctamente el proyecto conforme a los recursos disponibles. Esto puede hacerse mediante técnicas como *fine-tunning*, RAG o arquitecturas menos exigentes como cadenas de *Markov, generative grammar* o algoritmos genéticos.

5. **Necesidad de una herramienta completa**. Es fundamental desarrollar una solución de inteligencia artificial que responda a unas necesidades humanas concretas, haciéndola accesible y atractiva. Además, debe recalcarse que estas tecnologías son herramientas de apoyo y no un reemplazo de la creación humana.

6. **Consideraciones éticas y legales**. Es importante que más allá de los aspectos técnicos, se tengan siempre presentes las implicaciones éticas y legales de los modelos de inteligencia artificial, especialmente en los de tipo generativos.

Limitaciones y Perspectivas de Futuro

8

8.1. Limitaciones

El estudio realizado posee ciertas limitaciones que conviene destacar, con objeto de que puedan servir para trabajos posteriores.

- Escasez de datos etiquetados: Como se ha comentado a lo largo de la memoria, se dispone de un número limitado de datos MIDI con etiquetado emocional. Esta es la principal limitación de este estudio, ya que dificulta el entrenamiento supervisado del modelo y afecta significativamente a los resultados musicales obtenidos y su capacidad de expresividad emocional.
- Acceso a recursos computacionales: Los modelo generativos, y más concretamente los modelos de lenguaje natural, requieren de *hardware* especializado con varias unidades de procesamiento gráfico (GPU). La falta de acceso a estos recursos computacionales durante este trabajo de máster ha limitado considerablemente la experimentación, la construcción y el despliegue de modelos generativos.
- Falta de coherencia musical y expresividad emocional: A pesar de haberse identificado diferentes secciones musicales en la mayoría de los resultados, tamtp la emoción como la musicalidad que genera el modelo sigue siendo limita y debería mejorarse.

8.2. Perspectivas de futuro

Una vez realizado este trabajo fin de máster, se encuentran algunos aspectos que podrían implementarse y desarrollarse en el futuro.

- Desarrollar una herramienta mediante inteligencia artificial generativa que sirva de apoyo para músicos y compositores en el proceso creativo de composición musical. Esto ampliaría las posibilidades creativas, reduciendo el tiempo de experimentación y facilitando el flujo de ideas en las distintas etapas de la creación.
- Implementar un primer paso de entrenamiento no supervisado utilizando archivos MIDI sin condicionamiento emocional lo que permitiría que el modelo aprenda patrones musicales generales previos a la introducción del condicionamiento asociado a la emoción.

- Ampliar el conjunto de datos mediante un proceso automático de identificación de emociones en archivos MIDI no etiquetados, a partir de características MIDI como las descritas en la sección 5.2.1.
- Difundir la encuesta presentada en el Apéndice A para así ampliar la evaluación de las generaciones y recabar opiniones sobre el impacto de estas herramientas en la música.
- Explorar el rendimiento de otras arquitecturas como LSTM, GAN o *Transformer* basados en VAE, para explorar si existen arquitecturas más adecuada en cuanto a su coherencia musical y el condicionamiento emocional.
- Aplicar técnicas como *fine-tunning* o *Retrieval-Augmented Generation* (RAG) a modelos de lenguaje natural de gran escala como Llama, GPT u otros específicos del área de estudio, como *Music Transformer*. Esto permitiría explorar cómo afecta el conocimiento general de estos modelos a la generación de música simbólica.
- Integrar en el modelo un sistema de detección de emociones basado en indicaciones textuales (Nikam, 2024). Probablemente se mejoraría así la generación de música según los sentimientos identificados en un fragmento o estrofa de una canción.
- Introducir nuevas estructuras de condicionamiento y ampliar las opciones del modelo mediante un condicionamiento continuo.
- Profundizar en las métricas especializadas en la evaluación del condicionamiento.

Bibliografía

Agostinelli, A., Denk, T. I., Borsos, Z., Engel, J., Verzetti, M., Caillon, A., Huang, Q., Jansen, A., Roberts, A., Tagliasacchi, M., Sharifi, M., Zeghidour, N., y Frank, C. (2023). Musiclm: Generating music from text.

AI, S. (2023). Bark: Text-to-audio model. https://github.com/suno-ai/bark. GitHub repository.

Amazon Web Services (2019). AWS DeepComposer. Consultado: 22/11/2024.

Anthony, L. F. W., Kanding, B., y Selvan, R. (2020). Carbontracker: Tracking and predicting the carbon footprint of training deep learning models.

Asociación Española de Normalización (UNE) (2019). *UNE-ISO 14064: Gases de efecto invernadero. Especificaciones y directrices para la cuantificación y el informe de emisiones y remociones de gases de efecto invernadero*. UNE, Madrid, España. Norma basada en la ISO 14064.

Barnett, J. (2023). The ethical implications of generative audio models: A systematic literature review. In *Proceedings of the 2023 AAAI/ACM Conference on AI, Ethics, and Society*, AIES '23, page 146–161. ACM.

Bjørndalen, O. M. y Doursenaud, R. (2024). Mido: Midi i/o for python. Consultado: 11-02-2025.

Boletín Oficial del Estado (1996). Real decreto legislativo 1/1996, de 12 de abril, por el que se aprueba el texto refundido de la ley de propiedad intelectual.

Byrne, D. (2012). *How Music Works*. Canongate.

Caillon, A. y Esling, P. (2021). Rave: A variational autoencoder for fast and high-quality neural audio synthesis.

Chemla-Romeu-Santos, A. y Esling, P. (2022). Challenges in creative generative models for music: a divergence maximization perspective.

Copet, J., Kreuk, F., Gat, I., Remez, T., Kant, D., Synnaeve, G., Adi, Y., y Défossez, A. (2024). Simple and controllable music generation.

DataMXR (2024). Datamxr - exploring data and ai technologies.

Dhariwal, P., Jun, H., Payne, C., Kim, J. W., Radford, A., y Sutskever, I. (2020). Jukebox: A generative model for music.

Dong, H.-W., Hsiao, W.-Y., Yang, L.-C., y Yang, Y.-H. (2017). Musegan: Multi-track sequential generative adversarial networks for symbolic music generation and accompaniment.

Eno, B. (1995). Generative music: Evolving metamusic. In *In CD-ROM accompanying SSEYO Koan Software*, UK. Opal Ltd. / SSEYO Ltd. First use of "Generative Music"term by Brian Eno.

Ens, J. y Pasquier, P. (2021). Metamidi dataset.

European Parliament (2023). Eu ai act: first regulation on artificial intelligence. Consultado: 09-03-2025.

Ferreira, L. N. y Whitehead, J. (2021). Learning to generate music with sentiment.

Forsgren, S. y Martiros, H. (2022). Riffusion - Stable diffusion for real-time music generation.

Fradet, N., Briot, J.-P., Chhel, F., El Fallah Seghrouchni, A., y Gutowski, N. (2021). MidiTok: A python package for MIDI file tokenization. In *Extended Abstracts for the Late-Breaking Demo Session of the 22nd International Society for Music Information Retrieval Conference*.

Gemmeke, J. F., Ellis, D. P. W., Freedman, D., Jansen, A., Lawrence, W., Moore, R. C., Plakal, M., y Ritter, M. (2017). Audio set: An ontology and human-labeled dataset for audio events. In *Proc. IEEE ICASSP 2017*, New Orleans, LA.

Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., y Bengio, Y. (2014). Generative adversarial nets. In *Advances in neural information processing systems*, pages 2672–2680.

Hadjeres, G., Pachet, F., y Nielsen, F. (2017). Deepbach: a steerable model for bach chorales generation.

Hawthorne, C., Stasyuk, A., Roberts, A., Simon, I., Huang, C.-Z. A., Dieleman, S., Elsen, E., Engel, J., y Eck, D. (2019). Enabling factorized piano music modeling and generation with the MAESTRO dataset. In *International Conference on Learning Representations*.

Hsiao, W.-Y., Liu, J.-Y., Yeh, Y.-C., y Yang, Y.-H. (2021). Compound word transformer: Learning to compose full-song music over dynamic directed hypergraphs.

Huang, C.-Z. A., Vaswani, A., Uszkoreit, J., Shazeer, N., Simon, I., Hawthorne, C., Dai, A. M., Hoffman, M. D., Dinculescu, M., y Eck, D. (2018). Music transformer.

Huang, R., Huang, J., Yang, D., Ren, Y., Liu, L., Li, M., Ye, Z., Liu, J., Yin, X., y Zhao, Z. (2023). Make-an-audio: Text-to-audio generation with prompt-enhanced diffusion models.

Huang, Y.-S. y Yang, Y.-H. (2020). Pop music transformer: Beat-based modeling and generation of expressive pop piano compositions.

Hung, H.-T., Ching, J., Doh, S., Kim, N., Nam, J., y Yang, Y.-H. (2021). EMOPIA: A multi-modal pop piano dataset for emotion recognition and emotion-based music generation. In *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*.

Ji, S. y Yang, X. (2024). Emomusictv: Emotion-conditioned symbolic music generation with hierarchical transformer vae. *IEEE Transactions on Multimedia*, 26:1076–1088.

Joint Research Centre (JRC) (2023). Generative artificial intelligence: Assessment of ethical, societal, and policy implications. Consultado: 09-03-2025.

Kaur, N. y Tuttosi, P. (2023). Time out of mind: Generating rate of speech conditioned on emotion and speaker.

Kingma, D. P. y Welling, M. (2022). Auto-encoding variational bayes.

Kluwer Copyright Blog (2023). Should ai be attributed as an author of ai-generated works? *Kluwer Copyright Blog*. Consultado: 09-03-2025.

Kong, J., Kim, J., y Bae, J. (2020). Hifi-gan: Generative adversarial networks for efficient and high-fidelity speech synthesis. In *Advances in Neural Information Processing Systems 33*, pages 17022–17033.

Kumar, K., Kumar, R., de Boissiere, T., Gestin, L., Teoh, W. Z., Sotelo, J., de Brebisson, A., Bengio, Y., y Courville, A. (2019). Melgan: Generative adversarial networks for conditional waveform synthesis.

Liu, J., Dong, Y., Cheng, Z., Zhang, X., Li, X., Yu, F., y Sun, M. (2022). Symphony generation with permutation invariant language model.

Lu, P., Xu, X., Kang, C., Yu, B., Xing, C., Tan, X., y Bian, J. (2023). Musecoco: Generating symbolic music from text.

Machines, F. (2018). Hello world - the first ai-generated pop album. Consultado: 09-03-2025.

Madhok, R., Goel, S., y Garg, S. (2018). Sentimozart: Music generation based on emotions. In *International Conference on Agents and Artificial Intelligence*.

Melechovsky, J., Guo, Z., Ghosal, D., Majumder, N., Herremans, D., y Poria, S. (2024). Mustango: Toward controllable text-to-music generation.

Melodrive (2018). Melodrive on itch.io. Consultado: 18-11-2024.

Ministerio para la Transformación Digital y de la Función Pública, Secretaría de Estado de Digitalización e Inteligencia Artificial (2024). Anteproyecto de Ley para el Buen Uso y la Gobernanza de la Inteligencia Artificial. Consultado: 24-03-2025.

Music Business WorldWide (2025). Sony music reveals 75.000 ai deepfake takedowns, slams uk's 'rushed' copyright plans. *Music Business WorldWide*. Consultado: 10-03-2025.

Music Technology Group (2024). Software and datasets. Consultado: 19-11-2024.

MusicBrainz (2024). Musicbrainz. Consultado: 19-11-2024.

New York Post (2024). Sony, universal, warner sue ai startups suno, udio for infringement. *New York Post*. Consultado: 10-03-2025.

Nikam, S. (2024). Emotion detection in text. https://github.com/SannketNikam/Emotion-Detection-in-Text. Consultado: 16-03-2025.

Oore, S., Simon, I., Dieleman, S., Eck, D., y Simonyan, K. (2018). This time with feeling: Learning expressive musical performance.

Parlamento Europeo y Consejo de la Unión Europea (2024). Reglamento (UE) 2024/1689 del Parlamento Europeo y del Consejo, de 13 de junio de 2024, por el que se establecen normas armonizadas en materia de inteligencia artificial y por el que se modifican varios reglamentos y directivas. Consultado: 24-03-2025.

Planet, U. (2023). Ethics in generative ai for the radio jockeys. *UX Planet*. Consultado: 09-03-2025.

Qu, X., Bai, Y., Ma, Y., Zhou, Z., Lo, K. M., Liu, J., Yuan, R., Min, L., Liu, X., Zhang, T., Du, X., Guo, S., Liang, Y., Li, Y., Wu, S., Zhou, J., Zheng, T., Ma, Z., Han, F., Xue, W., Xia, G., Benetos, E., Yue, X., Lin, C., Tan, X., Huang, S. W., Fu, J., y Zhang, G. (2024). Mupt: A generative symbolic music pretrained transformer.

Raffel, C. (2016). The lakh midi dataset. https://colinraffel.com/projects/lmd/. Consultado: 22-11-2024.

Raffel, C. y Ellis, D. P. W. (2011). Optimizing Music Transcription for MIDI Sequencing. In *Proceedings of the 12th International Society for Music Information Retrieval Conference (ISMIR)*, pages 9–14. Consultado: 22-11-2024.

Raffel, C. y Ellis, D. P. W. (2014). Intuitive analysis, creation and manipulation of midi data with pretty_midi. *Late-Breaking/Demo Session at the 15th International Society for Music Information Retrieval Conference (ISMIR)*.

Ren, Y., He, J., Tan, X., Qin, T., Zhao, Z., y Liu, T.-Y. (2020). Popmag: Pop music accompaniment generation.

Research, R. (2023). Ia y el mercado de trabajo en españa. Consultado: 10-03-2025.

Rightsify (2023). Gcx - the world's 1st ai music dataset licensing service. Consultado: 24-03-2025.

Russell, J. (1980). A circumplex model of affect. *Journal of Personality and Social Psychology*, 39:1161–1178.

Shazam Entertainment Ltd. (2002). Shazam - music discovery, charts song lyrics.

Southern, T. (2017). Taryn southern is making an album co-written with artificial intelligence. *The Verge*. Consultado: 09-03-2025.

Spotify (2008). Spotify - music for everyone.

Sturm, B. L. T., Iglesias, M., Ben-Tal, O., Miron, M., y Gómez, E. (2019). Artificial intelligence and music: Open questions of copyright law and engineering praxis. *Arts*, 8(3).

Sulun, S., Davies, M. E. P., y Viana, P. (2022). Symbolic music generation conditioned on continuous-valued emotions. *IEEE Access*, 10:44617–44626.

The Sound of AI Community (2022). From words to sound: Neural audio synthesis of guitar sounds with timbral descriptors. In *Proceedings of the 3rd Conference on AI Music Creativity (AIMC)*. AIMC.

U.S. Copyright Office (2024). Copyright and artificial intelligence: Part 2 – copyrightability report. Consultado: 09-03-2025.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., y Polosukhin, I. (2023). Attention is all you need.

Velardo, V. (2023). The history of generative music. YouTube.

Wang, Y., Yang, W., Dai, Z., Zhang, Y., Zhao, K., y Wang, H. (2024). Melotrans: A text to symbolic music generation model following human composition habit.

Wikipedia contributors (2025). Musikalisches Würfelspiel. Consultado: 25-03-2025.

Yin, Z., Reuben, F., Stepney, S., y Collins, T. (2022). Measuring when a music generation algorithm copies too much: The originality report, cardinality score, and symbolic fingerprinting by geometric hashing. *SN Computer Science*, 3.

Zeng, M., Tan, X., Wang, R., Ju, Z., Qin, T., y Liu, T.-Y. (2021). Musicbert: Symbolic music understanding with large-scale pre-training.

Apéndice A

A

Cuestionario de evaluación de las generaciones musicales

A continuación, se presentan las preguntas del cuestionario de evaluación de las generaciones musicales.

Datos generales

1. Nombre
2. ¿En qué rango de edad se encuentra?
   - Menos de 18 años
   - 18 - 24 años
   - 25 - 30 años
   - 31 - 40 años
   - Más de 40 años
3. ¿Tiene formación musical?
   - Sí, de manera profesional
   - Sí, pero de forma autodidacta o amateur
   - No, pero tengo conocimientos básicos
   - No, no tengo formación musical

Evaluación de la calidad de la música generada

4. ¿Cómo calificaría la calidad general de la música generada?
   - Muy baja
   - Baja
   - Aceptable
   - Buena
   - Muy buena

5. ¿Considera que la música generada es armónica y coherente?
    - Sí, completamente
    - En su mayoría sí
    - En algunos momentos sí, en otros no
    - No, suena desordenada
    - No, no tiene coherencia musical
6. ¿Cómo describiría la complejidad de la música generada?
    - Demasiado simple
    - Equilibrada
    - Compleja en exceso

Percepción de la emoción transmitida

7. ¿La música generada le transmite alguna emoción clara?
    - Sí, claramente
    - Algo, pero no del todo
    - No, suena neutral
    - No, suena confusa
8. ¿Qué emoción cree que transmite?
    - Alegría, entusiasmo o euforia
    - Miedo, ansiedad o tensión
    - Relajación, calma o serenidad
    - Tristeza, melancolía o nostalgia

Percepción sobre el uso de herramientas generativas en la música

9. ¿Cree que las herramientas de inteligencia artificial pueden ser útiles en la composición musical?
    - Sí, como apoyo para los compositores
    - Sí, pero aún tienen muchas limitaciones
    - No, la creatividad humana es irremplazable
    - No estoy seguro/a
10. ¿En qué aspectos cree que la IA puede contribuir al proceso creativo de la música? (Seleccione todas las opciones que considere)
    - Generación de ideas iniciales

- Creación de melodías o armonías base
- Composición completa de piezas musicales
- Mejora o refinamiento de composiciones humanas
- No creo que la IA pueda aportar al proceso creativo

11. ¿Le parece interesante utilizar herramientas generativas como la del presente trabajo en el proceso creativo?

    - Sí, definitivamente
    - Sí, pero solo como referencia o inspiración
    - No lo sé, necesitaría probarlas
    - No, prefiero métodos tradicionales

12. En su opinión, ¿cuáles son los principales dilemas éticos asociados al uso de herramientas de IA en la creación musical? (Seleccione todas las opciones que considere relevantes)

    - Posibles problemas con los derechos de autor y la originalidad de la obra.
    - Impacto en el empleo de músicos y compositores.
    - Falta de creatividad y expresión artística en la música generada.
    - Alto consumo energético y su impacto ambiental.
    - Uso indebido o no transparente de la tecnología en la industria musical.
    - No veo aspectos negativos significativos.
    - Otra