Escola Tècnica Superior d'Enginyeria ETSE-UV

MÁSTER UNIVERSITARIO EN CIENCIA DE DATOS

TRABAJO DE FIN DE MÁSTER

ANÁLISIS COMPARATIVO DE MODELOS PROFUNDOS PARA LA GENERACIÓN AUTOMÁTICA DE MELODÍAS ARMONIZADAS

AUTOR:
SERGIO CONDE SÁNCHEZ

TUTORES:
FRANCESC J. FERRI RABASA
MÁXIMO COBOS SERRANO

NOVIEMBRE, 2023

Abstract

The ever-evolving music industry is characterized by a frenetic pace of creation, driven by advances in recording, production and distribution thanks to new technologies. However, the process of musical composition remains fundamentally handcrafted, ultimately dependent on the skill and experience of the composer. Unlike other artistic disciplines, such as visual art (with the example of Dall-e), where artificial intelligence has completely revolutionized the way we relate to this type of art, in the case of music, we do not yet have such widespread cases dedicated to the generation of musical pieces.

This study focuses on the automatic generation of music by Deep Learning and the associated evaluation methods. To illustrate this, two different models are proposed as examples. The goal is to generate music as naturally as possible and to evaluate the resulting compositions.

Through this work, we seek to provide a detailed view of how Artificial Intelligence is impacting and transforming music composition in the current artistic and technological landscape.

Resumen

La industria musical, en constante evolución, se caracteriza por un ritmo frenético de creación, impulsado por avances en grabación, producción y distribución gracias a nuevas tecnologías. Sin embargo, el proceso de composición musical sigue siendo fundamentalmente artesanal, dependiendo en última instancia de la habilidad y experiencia del compositor. A diferencia de otras disciplinas artísticas, como las visuales (con el ejemplo de Dall-e), donde la inteligencia artificial ha revolucionado por completo nuestra manera de relacionarnos con este tipo de arte, en el caso de la música aún no contamos con casos tan ampliamente difundidos dedicados a la generación de piezas musicales.

Este estudio se centra en la generación automática de música mediante Deep Learning y en los métodos de evaluación asociados. Para ilustrarlo, se proponen dos modelos diferentes como ejemplos. El objetivo es generar música de forma lo más natural posible y evaluar las composiciones resultantes.

A través de este trabajo, buscamos ofrecer una visión detallada de cómo la Inteligencia Artificial está impactando y transformando la composición musical en el actual panorama artístico y tecnológico.

Resum

La indústria musical, en constant evolució, es caracteritza per un ritme frenètic de creació, impulsat per avanços en gravació, producció i distribució gràcies a noves tecnologies. No obstant això, el procés de composició musical continua sent fonamentalment artesanal, depenent en última instància de l'habilitat i experiència del compositor. A diferència d'altres disciplines artístiques, com les visuals (amb l'exemple de Dall-e), on la intel·ligència artificial ha revolucionat per complet la nostra manera de relacionar-nos amb

esta mena d'art, en el cas de la música encara no comptem amb casos tan àmpliament difosos dedicats a la generació de peces musicals.

Este estudi se centra en la generació automàtica de música mitjançant Deep Learning i en els mètodes d'avaluació associats. Per a il·lustrar-ho, es proposen dos models diferents com a exemples. L'objectiu és generar música de forma el més natural possible i avaluar les composicions resultants.

A través d'este treball, busquem oferir una visió detallada de com la Intel·ligència Artificial està impactant i transformant la composició musical en l'actual panorama artístic i tecnològic.

Índice general

Capítulo 1

Introducción

La música, desde sus albores, ha sido una manifestación única de la creatividad humana, capaz de evocar emociones, contar historias y unir a las personas a través de los siglos. La creación musical, sin embargo, ha sido históricamente una empresa exclusiva de compositores y músicos con talento y formación especializada. Sin embargo, en la actualidad, estamos viviendo una revolución en la forma en que la música se crea y se comparte. Esta revolución se ha visto impulsada por el surgimiento de técnicas de inteligencia artificial, y en particular, por el aprendizaje profundo, que ha comenzado a desempeñar un papel crucial en la composición musical.

El campo del aprendizaje profundo ha experimentado un rápido desarrollo en los últimos años, impulsado por avances en hardware, grandes conjuntos de datos y algoritmos cada vez más sofisticados. Esto ha permitido a los investigadores y músicos explorar nuevas formas de crear música de manera automatizada y semiautomatizada. La idea de que las máquinas puedan generar música original y cautivadora ha capturado la imaginación de músicos, compositores y entusiastas de la inteligencia artificial en todo el mundo. El uso de la inteligencia artificial en la composición musical no es nuevo. En las décadas pasadas, se han desarrollado sistemas basados en reglas y técnicas de procesamiento de señales para generar música de forma automática. Sin embargo, el aprendizaje profundo ha llevado esta evolución a un nivel completamente nuevo. A través de redes neuronales artificiales profundas, se ha logrado capturar la complejidad y la riqueza de la música de una manera nunca antes vista. Además, ha abierto nuevas posibilidades para músicos y compositores al ofrecer herramientas creativas que pueden impulsar su proceso creativo y ampliar sus horizontes musicales.

El interés en la composición automática de música impulsada por el aprendizaje profundo es multifacético. En primer lugar, brinda la oportunidad de explorar nuevas fronteras musicales y desafiar las convenciones establecidas. Los algoritmos de aprendizaje profundo pueden producir música que va más allá de las limitaciones de las estructuras musicales tradicionales y desafía la creatividad de los compositores humanos. Además, este campo tiene un gran potencial para democratizar la creación musical, permitiendo a personas con poca o ninguna formación musical participar en la composición y experimentación musical. También puede servir como una herramienta de apoyo para músicos profesiona-

les, ayudándoles a generar ideas y expandir su creatividad. La composición automática de música también encuentra aplicaciones en la industria del entretenimiento y la publicidad, donde la música desempeña un papel fundamental en la creación de experiencias emocionales y atractivas para el público.

1.1. Motivación y objetivos

Este Trabajo Fin de Máster (TFM) tiene como objetivo principal explorar la generación musical a través de técnicas de Deep Learning. Para ello, se llevará a cabo un exhaustivo análisis de las herramientas y técnicas empleadas en la manipulación y representación de la música en el ámbito digital, prestando especial atención al estándar MIDI y su implementación en Python. Asimismo, se proporcionará un panorama detallado del estado actual de la Inteligencia Artificial en este campo, abordando los modelos más vanguardistas, los conjuntos de datos más utilizados y las diversas métricas de evaluación propuestas.

Para ilustrar los conceptos expuestos, se emplearán dos modelos como ejemplos de generación musical mediante Inteligencia Artificial: uno basado en *Long-Short Term Memory* (LSTM) y otro en una arquitectura Transformer-XL. Además, se llevará a cabo una evaluación de estos modelos utilizando algunas de las métricas propuestas, con el fin de determinar tanto la calidad musical de las piezas generadas como la eficacia de las propias métricas en la tarea de calificación.

Así, los objetivos de este trabajo se pueden resumir en los siguientes cuatro puntos:

- Estudiar metodologías para el análisis y generación de secuencias MIDI en Python.
- Hacer una revisión del estado del arte en la generación musical mediante Inteligencia Artificial.
- Entrenar y evaluar dos modelos de Deep Learning para la generación musical.
- Analizar las métricas de evaluación objetivas para modelos de generación musical automática.

Capítulo 2

Musica y Deep Learning

2.1. Composición algorítmica: contexto histórico

La composición algorítmica se refiere al proceso de creación musical con una intervención humana mínima [1]. Aunque el término puede parecer contemporáneo y relacionado con el uso de computadoras, esta idea tiene profundas raíces históricas. Desde tiempos antiguos, los griegos reconocieron que la música descansaba sobre una base objetiva y, aunque seguía siendo una herramienta de expresión espiritual, comenzaron a estudiarla como una disciplina científica. Para figuras como Pitágoras, la música y las matemáticas estaban inextricablemente unidas, y juntas gobernaban el Cosmos. Pitágoras incluso postuló que los planetas, al orbitar alrededor de la Tierra, emitían notas musicales cuyos tonos estaban determinados por los radios de sus órbitas, lo que se denominó la "música de las esferas". De esta manera, sentaron los cimientos de lo que eventualmente se convertiría en la teoría musical, explorando aspectos como la armonía y estableciendo formalismos y reglas que la describían.

A pesar de estos avances, la noción contemporánea de composición algorítmica aún estaba lejos de materializarse en su forma más rigurosa. Los antiguos griegos habían creado un sistema musical con intervalos y modos, que los músicos empleaban al interpretar piezas musicales, pero no habían logrado eliminar por completo la participación del músico en el proceso creativo. No fue hasta el siglo XV, que el Ser Humano comenzó a utilizar la teoría musical para generar piezas musicales basadas en reglas, del latín, "*canon*":

> "*El método predominante consistía en escribir una única parte vocal y dar instrucciones a los cantantes para que derivaran las voces adicionales a partir de ella. La instrucción o regla mediante la cual se derivaban estas partes adicionales se llamaba "*canon*", que significa 'regla' o 'ley'. Por ejemplo, se podía instruir a la segunda voz para que cantara la misma melodía comenzando un cierto número de pulsos o compases después de la original; la segunda voz podía ser una inversión de la primera o la misma en sentido contrario.*" [2].

Estas reglas de composición permiten eliminar al compositor del proceso de creación,

lo que las convierte en los primeros ejemplos de composición algorítmica. Casos posteriores son los de Mozart y su "*Musikalisches Wurfelspiel*"("Juego de dados musicales"), o los trabajos de John Cage, que utilizaban la aleatoriedad para combinar distintas piezas musicales y crear otras nuevas.

El siguiente paso, llegaría con la aparición de los ordenadores. Ada Lovelace, pionera en el desarrollo de las computadoras, ya reconocía en el siglo XIX:

> "*Suponiendo, por ejemplo, que las relaciones fundamentales del tono en cuanto a la armonía y la composición musical fueran susceptibles de tal expresión y adaptaciones, la máquina podría componer piezas musicales elaboradas y científicas de cualquier grado de complejidad o extensión.*"[1].

El primer ejemplo de composición generada por computadora data de Lejaren Hiller y Leonard Isaacson en la Universidad de Illinois en 1955-56. Utilizando la computadora digital de alta velocidad Illiac, lograron programar material básico y parámetros estilísticos que resultaron en la Suite Illiac [3]. La partitura de la pieza fue compuesta por la computadora y luego transcrita a notación musical tradicional para ser interpretada por un cuarteto de cuerdas. Lo que Hiller e Isaacson lograron en la Suite Illiac fue (a) generar ciertos "materiales básicos" con la computadora, (b) modificar estos materiales musicales de acuerdo a diversas funciones y luego (c) seleccionar los mejores resultados de estas modificaciones según diferentes reglas [1]. Otro uso pionero de la computadora en la composición algorítmica es el de Iannis Xenakis, quien creó un programa que generaría datos para sus composiciones "estocásticas" [4]. Xenakis utilizó las velocidades de cálculo de la computadora para calcular diversas teorías de probabilidad que ayudaron en composiciones como Atrées (1962) y Morsima-Amorsima (1962). El programa "deduciría" una partitura a partir de una lista de densidades de notas y pesos probabilísticos suministrados por el programador, dejando decisiones específicas a un generador de números aleatorios[1].

Se distinguen por tanto, dos metodologías para la composición por computadora: la estocástica y la basada en reglas; a la que se le añadiría más tarde, una tercera categoría basada en sistemas de Inteligencia Artificial (IA):

- Estocástica: implican aleatoriedad y pueden ser tan simples como generar una serie aleatoria de notas, como se ha visto en el caso de *Musikalisches Wurfelspiel* de Mozart. Sin embargo, también se puede introducir una gran cantidad de complejidad conceptual en los cálculos a través de la computadora utilizando teoría estadística y cadenas de Markov.

- Basada en reglas: el proceso se fundamenta en una serie de pruebas o reglas que guían el avance del programa. Estas etapas se diseñan de manera que el resultado de cada paso conduzca al siguiente de manera secuencial. A diferencia de la delegación de decisiones al azar que caracteriza a los métodos estocásticos previamente mencionados, los sistemas basados en reglas establecen previamente una gramática según la cual el proceso de composición debe desenvolverse una vez que se pone en marcha.

- Inteligencia artificial: en este caso es el propio sistema el que genera sus propias reglas y gramática para automatizar el proceso de creación musical. A continuación, desarrollaremos en profundidad esta rama.

2.1.1. Desarrollo de la IA en la generación musical

En la década de 1980, surgieron los primeros trabajos que empleaban los avances en el campo de la inteligencia artificial para la música. Entre estos, es de especial importancia la serie de experimentos llevados a cabo por David Cope, denominados Experimentos en Inteligencia Musical (EMI) [5]. Estos esfuerzos se centraron en la exploración de metodologías innovadoras con el propósito de enriquecer la comprensión de los sistemas musicales desde la perspectiva de la máquina. La concepción de Cope giraba en torno a la reconfiguración y alteración de composiciones musicales preexistentes con el fin de generar nuevas creaciones. En esta línea, Cope planteó la utilización inicial de bases de datos musicales previamente codificadas a formatos comprensibles por el computador. Posteriormente, llevó a cabo la análisis y extracción de fragmentos específicos de las composiciones de acuerdo con un conjunto predeterminado de reglas, las cuales le permitieron emular los estilos compositivos y generar composiciones originales a partir de estas premisas. Esta estructura basada en reglas de la música, junto con su enfoque analítico, sentó las bases para los posteriores modelos generativos musicales, influyendo de manera notable en las estructuras actuales de redes neuronales. No obstante, la limitada generalización en la capacidad compositiva y la necesidad de imponer reglas predefinidas de manera manual mermaron el potencial de estos modelos.

Desde la década de 1980 hasta los 2000, comenzaron a aparecer los primeros modelos que empleaban redes neuronales (NN) [6]. En los últimos años, gracias al crecimiento del Aprendizaje Profundo (DL), se ha visto un aumento considerable en la cantidad de investigaciones que exploran cómo las Redes Neuronales Profundas pueden aplicarse en la composición musical. Estos modelos de DL para la generación musical frecuentemente adoptan arquitecturas de NN que ya han demostrado un desempeño sobresaliente en dominios afines, como la Visión por Computadora o el Procesamiento del Lenguaje Natural (NLP). Otros ejemplos emplean modelos pre-entrenados en estos dominios con el propósito de fomentar la generación musical, fenómeno que se conoce como Transfer Learning. En la actualidad se están adoptando representaciones de entrada y arquitecturas de NN de aplicaciones de NLP a gran escala, como modelos basados en Transformers, que están demostrando un rendimiento muy bueno en esta tarea. Esto se debe a que la música puede entenderse como un lenguaje en el que cada estilo o género musical tiene sus propias reglas. Más adelante en este artículo se presentarán algunas de las técnicas y arquitecturas de NN más utilizadas [7].

Existen dos enfoques principales para la generación musical mediante IA, en función del tipo de representación que se haga de la misma. Por un lado, está el procesamiento del audio, que ofrece una representación continua de la composición, y se centra más en aspectos acústicos del sonido, como el timbre. De esta manera, se analiza la forma de onda del audio y su espectrograma. Por otro lado, está la representación simbólica del audio, una representación discreta y que trata de recoger las distintas cualidades musicales del audio (tono, ritmo, armonía, etc) a través de la notación musical. Este trabajo se centra

en los modelos que utilizan el segundo enfoque. En concreto, aquellos que utilizan la representación MIDI para extraer la información de cada pieza musical, la cual, trataremos de explicar a continuación.

2.2. MIDI

El MIDI (Musical instrument digital interface) es un estándar de la industria que describe un protocolo para posibilitar la comunicación entre diversos instrumentos electrónicos y programas de software. Este protocolo permite que dichos dispositivos se conecten y compartan información de manera efectiva [8]. Músicos, DJ's y productores de todo el mundo lo utilizan en su día a día para crear, actuar y compartir música.

El sistema MIDI transporta toda la información en forma de eventos, que especifican el tono, el tempo o la intensidad, que avanzan a través del tiempo. Esta representación, cuenta con 128 tonos (notas musicales), y los cuales va situando en el tiempo con eventos Note-On (que señalan el inicio de una nota musical) y Note-Off (que indican el final de la misma). Además de esto, se le añaden otro tipo de eventos como la intensidad de la nota, el tempo, o el instrumento que la hace sonar (tabla 1).

Octave	**Note numbers**											
	Do	**Do#**	**Re**	**Re#**	**Mi**	**Fa**	**Fa#**	**Sol**	**Sol#**	**La**	**La#**	**Si**
	C	**C#**	**D**	**D#**	**E**	**F**	**F#**	**G**	**G#**	**A**	**A#**	**B**
0	0	1	2	3	4	5	6	7	8	9	10	11
1	12	13	14	15	16	17	18	19	20	21	22	23
2	24	25	26	27	28	29	30	31	32	33	34	35
3	36	37	38	39	40	41	42	43	44	45	46	47
4	48	49	50	51	52	53	54	55	56	57	58	59
5	60	61	62	63	64	65	66	67	68	69	70	71
6	72	73	74	75	76	77	78	79	80	81	82	83
7	84	85	86	87	88	89	90	91	92	93	94	95
8	96	97	98	99	100	101	102	103	104	105	106	107
9	108	109	110	111	112	113	114	115	116	117	118	119
10	120	121	122	123	124	125	126	127				

Figura 2.1: Representación de las notas en el estándar MIDI [9].

En los sistemas MIDI, la unidad de tiempo es el "tick", que representa una fracción de un pulso musical. De este modo, aunque el intervalo de tiempo en segundos entre eventos puede variar según el tempo de la composición, el número de ticks por pulso permanece constante, y por defecto en un MIDI la resolución es de 48 ticks por pulso. Asimismo, la ubicación de cada evento en dicha pieza se determina con respecto al evento anterior, y no con respecto al incio de la pieza. A esta duración entre eventos sucesivos se le conoce como "Delta Time". Por tanto, por defecto en un MIDI, si el Delta Time entre dos eventos es de 0 ticks, estos eventos ocurren simultáneamente; mientras que si el Delta Time es de 24, habrá una diferencia de medio pulso, es decir, equivalente a una corchea.

Tipo de Evento	Descripción
Note-On	Indica que una nota ha comenzado a sonar en uno de los 128 tonos.
Note-Off	Indica que una nota ha dejado de sonar en uno de los 128 tonos.
Note Velocity	Define la intensidad de la nota.
Note Duration	Tiempo que pasa entre un evento Note-On y su correspondiente Note-Off. Se calcula acumulando los eventos de Time Shift que hay en medio.
Time Shift	Avanza en el tiempo el número de pasos correspondientes.
Position	Apunta a distintas localizaciones dentro de un compás.
Bar	Marca los comienzos de cada compás
Piece Start	Marca el comienzo de la pieza
Program Select/Instrument	Establece un instrumento para cada canal
Track	Establece qué canal suena
Tempo	Define el tempo (ritmo) de la música.

Tabla 2.1: Tipos de Eventos en Archivos MIDI, [8]

2.2.1. Piano-roll

Existen varios métodos para visualizar de manera gráfica el contenido de un archivo MIDI. Uno de los más comunes es conocido como piano-roll, el cual se inspira en los rollos de papel perforado utilizados en pianos mecánicos. Este enfoque implica la creación de una matriz en la que el eje horizontal representa el tiempo y el eje vertical representa las notas musicales. En esta matriz, cada fila corresponde a una nota musical (hay un total de 128 notas posibles) y las columnas varían en función de la duración de la pieza y su resolución temporal. De esta manera, es posible representar visualmente cualquier contenido MIDI, incluyendo múltiples canales y pistas. Existen diversas librerías de Python que permiten representar el piano-roll de un MIDI muy fácilmente, como Music21 [1], pretty_midi [2] o Pypianoroll [3].

Existen diversas técnicas para representar de manera gráfica el contenido de un MIDI. La más extendida es el piano-roll, inspirado en los rollos de papel perforado que se utilizaban para hacer sonar los pianos mecánicos. Consiste en una matriz, donde el eje X representa el tiempo y el eje Y el tono. De este modo, se obtiene una matriz de 128 filas (una por nota) y un número de columnas que dependerá de la duración de la pieza y su resolución. Se consigue así, representar cualquier tipo de MIDI con todos sus canales de manera visual.

[1] https://web.mit.edu/music21/doc/usersGuide/usersGuide_22_graphing.html
[2] https://craffel.github.io/pretty-midi/
[3] https://salu133445.github.io/pypianoroll

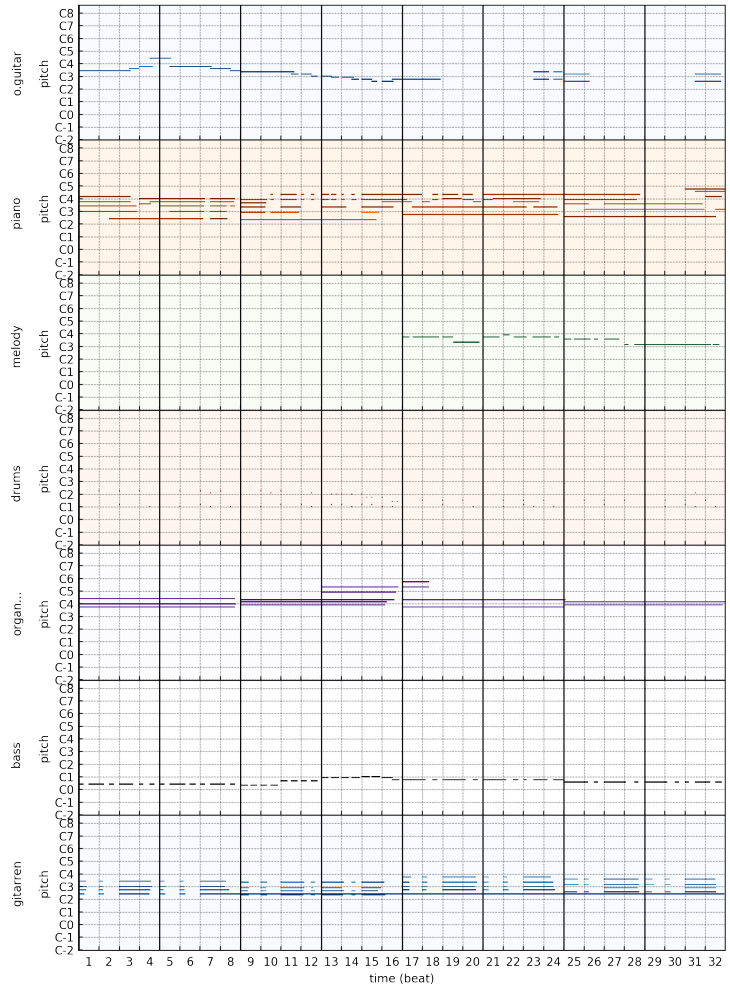


Figura 2.2: Ejemplo de piano-roll multipista creado con la librería Pypianoroll.

2.2.2. MIDI en Python

Existen varias bibliotecas en Python diseñadas para manipular archivos MIDI, algunas de las cuales hemos mencionado anteriormente, cada una con sus propias características distintivas. En nuestro caso, hemos utilizado dos de ellas. Por un lado, Music21 [10] es una biblioteca sumamente útil para extraer información de archivos MIDI, así como para editarlos y crear nuevos. Esta herramienta ofrece una interfaz muy intuitiva para representar y generar nuevos archivos MIDI de manera sencilla. Con esta librería podemos leer fácilmente los MIDI, para después representarlos fácilmente en forma de texto o en un pentagrama (Figura 2.3). Esta librería utiliza una representación de secuencias de eventos, que pueden anidarse unas dentro de otras. Cada evento puede ser una nota, un silencio, un cambio de tempo, etc. Además, cada evento tiene sus propios atributos, como tono, duración, posición, entre otros. Otras bibliotecas similares incluyen Mido [11] y pretty_midi [12]. Por otro lado, se utilizó MusPy [13] para calcular las métricas de los modelos. Esta librería no solo es capaz de leer archivos MIDI, si no que puede trabajar con interfaces como las ya mencionadas Music21 o pretty_midi. Muspy, también utiliza una represen-

tación basada en eventos como las anteriores. Sin embargo, su principal ventaja radica en las funciones incorporadas para calcular métricas objetivas para las composiciones, las cuales se definirán en el siguiente apartado. Otra librería muy similar a esta es Pypianoroll [14], que ofrece funciones similares pero utiliza una representación basada en piano-roll mediante vectores de numpy.

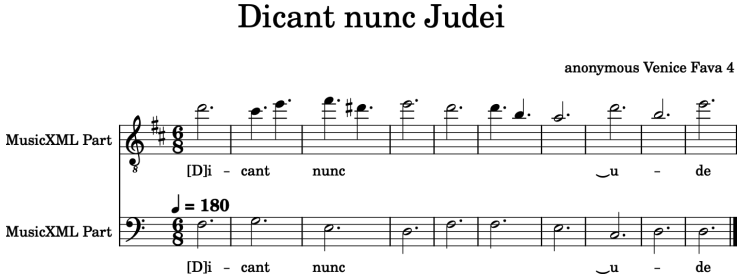


Figura 2.3: Pentagrama creado con Music21.

2.3. Modelos neuronales profundos

Comenzaremos describiendo brevemente las arquitecturas de NN más utilizadas en el campo de la música generativa. Es importante notar que las aplicaciones del DL a la música son incontables: composición, síntesis de audio, transcripción de partituras, producción, etc. En este caso, nos centraremos en la parte compositiva, en la cual, los modelos con más éxito son los generativos, como los *Variational Autoencoder* (VAE) o *Generative Adversarial Network* (GAN), y los basados en procesado del lenguaje como las *Long Short-Term Memory* (LSTM) o los *Transformer* [7].

Variational Auto-Encoders (VAEs)

Se construye a partir de un autoencoder formado por dos redes: un encoder y un decoder, a los que añade una función de pérdida para evaluar cuánto se parece la salida del decoder a la entrada del encoder. De este modo, la primera red (encoder) se encarga de codificar el input hacia un espacio normalmente de más baja dimensionalidad, denominado espacio latente; y la segunda red es la encarga de recomponer el input a partir de su representación en el estado latente. Este es el funcionamiento general de cualquier autoencoder, sin embargo, el Variational Autoencoder tiene la característica de que el espacio latente que construye es continuo, permitiéndole generar nuevos objetos a partir de la interpolación en las representaciones latentes de los objetos de entrada. Esto lo hace considerando el espacio latente no como un espacio de codificación, si no como un distribución de probabilidad [15].

Generative Adversarial Networks (GANs)

Las GAN [16] son Modelos Generativos compuestos por dos Redes Neuronales: el Generador, que se encarga de generar outputs a partir de un vector de ruido; y el Discriminador, que aprende a distinguir los objetos reales de los producidos por el generador.

El entrenamiento se realiza de manera que el discriminador maximice la probabilidad de asignar la etiqueta correcta a las muestras de entrenamiento y a las muestras generadas por el generador, mientras que el generador trata de maximizar el error de clasificación del discriminador a través de generar objetos cada vez más realistas. El generador y el discriminador pueden estar compuestos por diferentes capas de redes neuronales, como Perceptrones Multicapa (MLP), LSTM o Redes Neuronales Convolucionales (CNN).

Long Short-Term Memory (LSTM)

Es un tipo de red recurrente (RNN), capaz de solucionar el problema del desvanecimiento del gradiente, que se producía en las RNN clásicas y limitaba su memoria a largo plazo [17]. Este problema lo solucionan a través de un mecanismo de compuertas, que le permite controlar el flujo de información y decidir qué información olvidar y cual recordar. En los siguientes puntos, se explicarán este tipo de redes con más profundidad y se construirá un modelo basado en ellas.

Transformers

Los Transformers [18] son un tipo de red ampliamente utilizados en el área del NLP, y su principal innovación está es su mecanismo de atención. Primero la entrada se divide en segmentos que posteriormente se convierten a vectores mediante un word embedding, para más tarde contextualizarlos y relacionarlos entre sí mediante su mecanismo de atención. Esto los convierte en arquitecturas más eficientes que otros tipo de redes recurrentes, como las LSTM.

2.3.1. Desafíos del Deep Learning en la composición musical

El proceso de composición musical, al igual que en cualquier otra disciplina artística, puede abordarse desde diversas perspectivas. Este procedimiento exhibe una naturaleza profundamente subjetiva, dando lugar a una variedad de interrogantes concernientes a la elección de la representación idónea para la entrada del modelo, la determinación de la arquitectura más pertinente, y la selección del método apropiado para evaluar el resultado obtenido. En este contexto, el presente trabajo trata de abordar tales interrogantes y proponer un modelo ejemplar con la capacidad de concebir música y melodías, las cuales pueden desempeñar un papel de soporte al compositor en la generación de nuevas ideas. No obstante, primero se examinará en profundidad el proceso de composición y se intentarán establecer unos principios fundamentales que deben sustentar una obra musical.

2.4. Proceso de composición

Como se comentaba anteriormente, el proceso de composición musical es un proceso complejo que depende de una gran cantidad de decisiones. Además de ser profundamente personal, está también sujeto a factores como el estilo con el que estemos trabajando. La música clásica se caracteriza por comenzar con un pequeño motivo que se va desarrollando y repitiendo a lo largo de la obra para generar melodías más complejas, mientras que en

estilos como el Jazz o el Pop, es más común partir de una progresión de acordes sobre los que se compone o improvisa una melodía. Aún así, independientemente del estilo, existen cualidades que comparten todas las obras musicales. Cualquier pieza debe partir de una progresión armónica sobre la que se desarrolla un motivo o una melodía, y que además, lo hace siguiendo una estructura determinada. Por último, es necesaria la instrumentación de cada parte de manera que la melodía resalte sobre la base armónica.

La música, por tanto, tiene dos dimensiones: la dimensión temporal y la armónica. La dimensión temporal está representada por el ritmo, cuyo nivel más bajo de representación es la duración de la nota. Estas notas se agrupan para formar frases y motivos, y además, se estructuran en torno a compases, que sirven para establecer una métrica en la pieza y marcar el ritmo. Por otro lado, la dimensión armónica está formada por el tono e la nota, que a su vez, se pueden agrupar para formar acordes cuando suenan a la vez. De este modo, en una representación gráfica la dimensión temporal correspondería al eje horizontal, mientras que en la vertical se situaría la dimensión armónica. Este es el caso de los pentagramas o el piano-roll, una representación muy utilizada en softwares de producción musical e instrumentos digitales.

Podemos establecer por tanto, los siguientes cuatro principios básicos y objetivos [19], sobre los que se debe apoyar un sistema de generación y análisis musical basado en DL:

- Armonía: Es la superposición de notas que forman acordes que componen una progresión de acordes. El nivel de la nota podría considerarse como el nivel más bajo en armonía, seguido por el nivel de acorde. El nivel más alto se puede considerar como el nivel de progresión, que generalmente pertenece a una determinada tonalidad.

- Estructura: En música, la estructura se refiere a la organización y disposición de los elementos musicales en una composición. Es la forma en la que los componentes individuales de una pieza musical se interconectan y se presentan en una secuencia coherente. La estructura musical es esencial para dar sentido y cohesión a una composición, permitiendo que los oyentes sigan y comprendan la narrativa musical.

- Melodía y textura: La textura en términos musicales se refiere a los contenidos melódicos, rítmicos y armónicos que deben combinarse en una composición para formar la pieza musical. La música puede ser monofónica o polifónica según las notas que se suenen en el mismo instante de tiempo.

- Instrumentación: Estas son técnicas musicales que abordan la elección de instrumentos o pistas en una pieza musical.En contextos de grabación o representación digital, los instrumentos se estructuran como pistas. Por lo tanto, una obra que emplea múltiples instrumentos puede ser etiquetada como multi-pista. Donde cada pista puede contener una nota o varias notas que suenan simultáneamente, lo que da lugar a pistas monofónicas y polifónicas, respectivamente.

Estas cuatro categorías están relacionadas entre sí, de modo que un modelo encaminado a la generación musical debe ser capaz de asimilar estas cuatro cualidades del conjunto de datos de entrada, para después replicarlas en sus resultados.

2.5. Deep Learning para la generación de melodías

Los primeros ejemplos de modelos de DL capaces de generar melodías cortas comenzaron a aparecer a partir de 2016, y estaban basados en RNN [20]. Sin embargo, estos modelos tenían problemas a la hora de generar secuencias más largas, por lo que, con el tiempo, comenzaron a aparecer modelos que combinaban estas RNN con métodos probabilísticos, como Melody RNN de Google Magenta [21] o DeepBach [22], que era capaz de generar corales de 4 voces al estilo de Bach.

Más tarde se comenzaría a probar con modelos generativos, para mejorar los problemas que tenían los modelos previos a la hora de generar melodías creativas desde cero. Así, apareció MusicVAE [23], que estaba entrenado con 1.5 millones de canciones del Lakh MIDI Dataset [24] y puede generar melodías polifónicas para 3 instrumentos: melodía, bajo y batería. Más tarde, con el desarrollo de los Transformers, comenzaron a aparecer nuevos modelos como Music Transformer [25] en 2018, también de Google Magenta, o MuseNet [26] en 2019, de OpenAI. Estos modelos basados en Transformers pueden generar melodías más largas y continuar una secuencia dada, pero después de unos compases o segundos, la melodía termina siendo un tanto aleatoria, es decir, hay notas y armonías que no siguen el sentido musical de la pieza. Más recientemente, se han comenzado a desarrollar modelos que combinan distintos tipos de arquitecturas, como TransformerVAE [27], con muy buen rendimiento incluso en música polifónica. Por otro lado, también surgieron modelos condicionados, como BebopNet [28], introducido en 2020 y focalizado en el Jazz. Este modelo muestra un rendimiento destacado al crear melodías basadas en progresiones de acordes específicas. Si nos centramos en la música multi-instrumental, los trabajos que mejor resultado han dado son aquellos que utilizan modelos generativos, como GANs o VAEs. Un ejemplo de esto es MuseGAN [29], que utiliza una arquitectura de DCGAN [30], entrenada con el Lakh Midi Dataset, para generar música con 5 pistas diferentes. El planteamiento inicial de este modelo, es el de utilizar una representación gráfica de la música, como los piano-roll, para tratar las piezas como imágenes.

No obstante, a pesar de la abundancia de ejemplos capaces de crear melodías y composiciones agradables al oído, todos presentan deficiencias cuando se trata de dar estructura a estas piezas. El principal problema de todos estos modelos radica en su incapacidad para comprender y asimilar las estructuras dentro de una composición, son incapaces de diferenciar entre una estrofa, un estribillo o un puente; lo cual limita su capacidad para crear canciones completas desde cero. Por lo que actualmente juegan un papel de apoyo para los artistas al momento de inspirar ideas para nuevas melodías.

2.6. Datasets

Igual de importante que la arquitectura de un modelo, lo es el conjunto de datos con el que se entrena. A continuación, se exponen tres de los datasets de música simbólica más utilizados:

- Maestro Dataset [31]: creado por Google Magenta, y con casi 200 horas de piano

capturadas a lo largo de diez ediciones del International Piano-e-Competition [4], donde los pianistas interpretaban en pianos equipados con sistemas de grabación MIDI. Se trata de un dataset de música clásica, polifónica y para piano. Además cada archivo MIDI está emparejado con el audio correspondiente.

- Weimar Jazz Database [32]: una gran base de datos con 456 transcripciones de solos de Jazz en distintos formatos y con toda su metadata. Además de contener los MIDI de cada melodía, proporciona distintas tablas con información adicional como acordes, género, intérprete, BPM, etc. Este conjunto de datos se ha utilizado en modelos como BebopNet o Jazz Transformer y será la que emplearemos en este trabajo.

- Lakh MIDI Dataset: una colección de 176.581 archivos MIDI de diversos géneros e instrumentos. Es un dataset a gran escala para un uso general. Como se comentó previamente, este conjunto ha sido utilizado en modelos como MuseGAN o Music-VAE.

2.7. Métodos de evaluación

En el contexto de la música generada por ordenador, es posible clasificar las métricas de evaluación de los modelos en dos categorías generales: subjetivas y objetivas [8]. Sin embargo, uno de los desafíos fundamentales en el avance de este campo radica en la dificultad de evaluar elementos sumamente subjetivos de los resultados, como el nivel de creatividad o la calidad auditiva de una melodía. Para tales aspectos, aún carecemos de métodos que puedan medir estas cualidades de manera imparcial y precisa. En el mejor de los casos, los métodos objetivos pueden brindarnos una comprensión de la calidad de los sonidos en un sentido más amplio, aunque resulta complicado vincularlos con atributos como la creatividad o la estética. A continuación, se presentan las métricas establecidas en diversos modelos del estado del arte para evaluar sus resultados desde las dos perspectivas: objetiva y subjetiva.

2.7.1. Métodos Objetivos

La evaluación objetiva mide el rendimiento del modelo y la calidad de sus resultados utilizando funciones definidas a partir de los datos, que cumplen propiedades relacionadas con las (pseudo)métricas. En el ámbito de la generación de música, surge el desafío de comparar modelos entrenados con distintos propósitos y conjuntos de datos variados. En este contexto, podemos dividir las métricas de evaluación objetiva en dos categorías: métricas de modelo y métricas musicales [8].

Al evaluar el rendimiento de un modelo en el aprendizaje profundo para la generación musical, se emplean comúnmente métricas como la pérdida, la perplejidad, el puntaje BLEU [33], la precisión (P), la recuperación (R) y la puntuación F (F1). Estas métricas se utilizan para comparar diferentes modelos de DL creados con el mismo propósito. La

[4] https://www.piano-e-competition.com/

pérdida (o *loss* en inglés) se utiliza para cuantificar la diferencia entre las entradas y salidas del modelo desde un punto de vista matemático, mientras que la perplejidad nos indica la capacidad de generalización del modelo. Por ejemplo, Music Transformer [25] utiliza la pérdida y la perplejidad para comparar las salidas entre diferentes arquitecturas de Transformers, mientras que MusicVAE [23] solo utiliza una medida que indica la calidad de la reconstrucción que posee el modelo.

En lo que respecta a las métricas específicamente relacionadas con la música, se puede observar que estas métricas ayudan a medir la calidad de una composición. Cada una de estas métricas pueden centrarse en distintos aspectos de la teoría musical. Existen métricas relacionadas con el tono, como la consistencia de la escala o la entropía de la tonalidad; con el ritmo, que tienen en cuenta la duración o el patrón de las notas, como las variaciones rítmicas, el número de tres o cuatro notas concurrentes o la duración de notas repetidas; o con la armonía, que miden la entropía, la distancia o la cobertura de los acordes. Estas tres categorías de métricas han sido utilizadas en trabajos de investigación relacionados como MuseGAN [29], BebopNet [28] o Jazz Transformer[34].

Métricas relacionadas con el tono:

- Entropía del Histograma de Clases de Tonalidad ($\mathcal{H}$) [34]: se calcula la entropía de los tonos que aparecen en un periodo concreto (por ejemplo, un compás):

$$\mathcal{H}(\vec{h}) = \sum_{i=0}^{11} h_i \circledast log_2(h_i) \tag{2.1}$$

  Donde $\vec{h}$ es el histograma de clases de tonalidad de 12 dimensiones (Do, Do#, ..., La#, Si), normalizado por el número total de notas en dicho periodo, de manera que $\sum_i h_i = 1$. La entropía, en teoría de la información, es una medida de la incertidumbre de una distribución de probabilidad, por lo tanto, la adoptamos aquí como una métrica para ayudar a evaluar la calidad musical en cuanto a la tonalidad. Si la tonalidad de una pieza es clara, varias clases de tonalidad deberían dominar el histograma de tonalidad (por ejemplo, la tónica y la dominante), lo que resultaría en una $\vec{h}$ de baja entropía; por el contrario, si la tonalidad es inestable, es probable que el uso de clases de tonalidad esté disperso, lo que daría lugar a una $\vec{h}$ con alta entropía.

  En este trabajo emplearemos esta métrica en ventanas de 1 y 4 compases, de modo que $\mathcal{H}_1$ será el histograma de clases en un compás y $\mathcal{H}_4$ la entropía calculada en periodos de 4 compases consecutivos.

- Consistencia de la escala ($\mathcal{SC}$) [20]: Esta métrica proporciona un valor que indica la afinidad de una pieza o fragmento musical con respecto a una de las escalas musicales mayores o menores. Para lograrlo, se calcula el ratio de coincidencia de todas las notas dentro de dicho periodo con cada una de las escalas mayores y menores para luego devolver el valor más alto. De esta forma, se busca cuantificar la presencia de disonancias en la pieza.

En nuestro caso, utilizaremos esta métrica, de nuevo, para ventanas de 1 ($\mathcal{SC}_1$) y cuatro compases ($\mathcal{SC}_1$)

Métricas relacionadas con el ritmo:

- Groove Pattern Similarity ($\mathcal{GS}$)[34]: El patrón de ritmo (Groove) representa las posiciones en un compás en las que hay al menos un inicio de nota, denotado por $\vec{g}$. Este es un vector binario cuya dimensión depende del compás y de la unidad mínima de tiempo definida (en un compás de 4/4 y con unidad mínima la semicorchea, el vector tendrá longitud $l = 16$). Se define la similitud entre un par de patrones de ritmo $\vec{g_a}$, $\vec{g_b}$ de la siguiente manera:

$$GS(\vec{g_a}, \vec{g_b}) = 1 - \frac{1}{Q}\sum_{i=0}^{Q-1} XOR(\vec{g_{a,i}}, \vec{g_{b,i}}) \tag{2.2}$$

  donde Q es la dimensionalidad de $\vec{g_a}$, $\vec{g_b}$, y XOR$(\cdot, \cdot)$ es la operación "OR" exclusivo. De este modo, para un segmento de múltiples compases, GS será el promedio entre cada uno de esos pares de compases consecutivos.

  La similitud de patrones de ritmo ayuda a medir la rítmica de la música. Si una pieza posee un claro sentido de ritmo, los patrones de ritmo entre pares de compases deberían ser similares, produciendo GS altos; por otro lado, si el ritmo se siente inestable, los patrones de ritmo entre compases deberían ser erráticos, lo que resultaría en GS bajos.

- Ratio de pulsos/compasés vacíos [29]: ratio de pulsos o compases en los que no suena ninguna nota, con respecto al total de la pieza. De esta manera, se trata de cuantificar la dinámica de una pieza y de las frases que la forman.

Métricas para las progresiones de acordes:

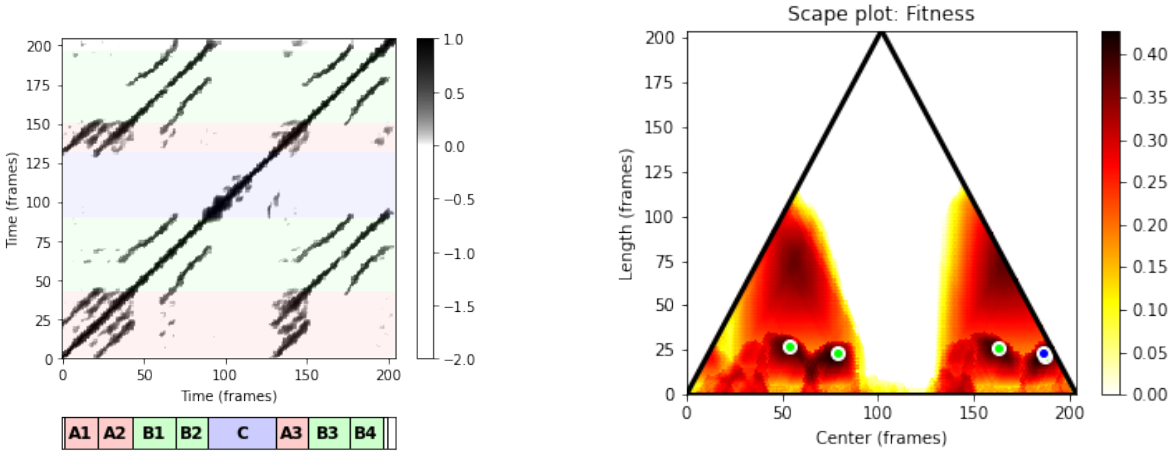
- Para medir la irregularidad de una progresión de acordes, se puede introducir el término "trigrama de acordes" [34], un trío compuesto por 3 acordes consecutivos en una progresión de acordes; por ejemplo, (Dm, G, C). Luego, la irregularidad de la progresión de acordes (CPI, por sus siglas en inglés) se define como el porcentaje de trigramas de acordes únicos en la progresión de acordes de una pieza musical completa. Es importante tener en cuenta que se consideran diferentes dos trigramas de acordes si alguno de sus elementos no coincide.

  Es común que las composiciones de Jazz hagan uso de plantillas de progresiones de acordes de 8 o 12 compases de longitud (conocidas como blues de 8 o 12 compases), que a su vez se pueden descomponer en subestructuras similares, como base de una sección, y luego se pueden copiar y pegar más o menos para formar la canción completa, por ejemplo, con partes ABBA. Por lo tanto, una pieza de Jazz bien compuesta debería tener una irregularidad en la progresión de acordes que no sea demasiado alta.

Métricas relacionadas con la estructura:

- Indicadores de estructura ($\mathcal{SI}$)[34]: La estructura en una composición se basa en la repetición, la cual puede implicar múltiples granularidades, que van desde una idea musical breve hasta una sección completa. Desde una perspectiva psicológica, la aparición de estructuras repetidas es la esencia de lo pegadizo y la capacidad de provocar emociones en la música.

  Los indicadores de estructura, $\mathcal{SI}$, [35] ofrecen una manera de calcular y representar este tipo de repeticiones en una composición. Para ello, se calcula una matriz de similitud, $S_{N\times N}$, donde $s_{ij} \in [0, 1]$ representa el grado de similitud entre cada ventana i y j de la composición (figura 2.4a), que a su vez, son vectores de características extraídos del propio audio [36]. Una vez calculada la matriz de auto-similitud (SSM), se procede a definir todos los segmentos de una pieza en función de su duración y de la posición de su centro en la pieza. Así, se consigue también una representación de las relaciones entre partes de una composición de manera jerárquica, denominada Fitness Scape Plot (FSP). Ejemplo de esto es la figura 2.4, donde se muestran la SSM y el FSP de la Danza Húngara Nº 5 de Johannes Brahms, con estructura $A_1A_2B_1B_2CA_3B_3B_4$. En estas gráficas se puede observar perfectamente, mediante la escala de colores, la repetición de las partes A y, sobre todo, B; así como la ausencia de la misma en la parte central, C.



(a) Matriz de auto-similitud, SSM.

(b) Fitness Scape Plot.

Figura 2.4: Johannes Brahms - Danza Húngara Nº 5.

Más allá de una representación visual, los indicadores de estructura, $\mathcal{SI}$, proporcionan un valor númerico sobre la repetición más destacada dentro de un intervalo de tiempo determinado. Así, asumiendo una frecuencia de sampleo de 1 Hz en nuestra pieza para simplificar (así, N será la duración de la pieza en segundos), se define el indicador de estructura como:

$$\mathcal{SI}_l^u(S) = \max_{\substack{l\leq i\leq u \\ 1\leq j\leq N}} S \tag{2.3}$$

Donde l, u son los límites inferior y superior de la duración del intervalo. En nuestro caso, trataremos de analizar las repeticiones a corto, medio y largo plazo, por lo que emplearemos segmentos de entre 3 y 8 segundos, 8 y 15 y mayores de 15, respectivamente.

Otras métricas:

- Predicciones de continuaciones:[37]: este tipo de métricas consisten en alimentar al modelo con un segmento de una pieza real para después, analizar su predicción. Así, se le proporciona como input un vector, $\vec{s}$, con los primeros N compases y un conjunto de posibles continuaciones, $X = \{\vec{x_0}, \vec{x_1}, \vec{x_2}, ...\}$, en el cual una de las opciones es la correcta y el resto son respuestas incorrectas seleccionadas al azar de otras piezas. De este modo, el modelo calculará la probabilidad promedio de generar los eventos de cada continuación:

$$P(\vec{x_i}) = \frac{1}{L}\sum_{j=0}^{L-1} p(x_{ij}|\tilde{x}_{j-1}, \ldots, \tilde{x}_0; \vec{s}) \tag{2.4}$$

donde L es la longitud de la continuación proporcionada más corta en X, $x_{i,j}$ es el j-ésimo token de evento en la continuación $\vec{x_i}$, y $\tilde{x}_{j-1}, \ldots, \tilde{x}_0$ son los eventos de salida del modelo. Por lo tanto, la probabilidad condicional $p(x_{i,j})$ en cada paso de tiempo se puede obtener de manera directa. Finalmente, el modelo devuelve $arg\,max_i P(\vec{x_i})$ como su respuesta. Si el modelo puede lograr una alta precisión en esta tarea de predicción de continuación, podemos decir que posee una buena comprensión general de la música de Jazz, suficiente para distinguir entre lo correcto y lo incorrecto cuando se le presentan opciones múltiples.

2.7.2. Métodos subjetivos

El objetivo final de un modelo de generación musical es el de reproducir sonidos que puedan ser considerados creativos, novedosos y estéticos. Características imposibles de medir con métricas objetivas, al menos hasta el día de hoy. Por eso, la aplicación de métricas subjetivas es fundamental en la evaluación de una herramienta de este tipo.

El método más utilizado en evaluaciones subjetivas es la prueba de audición, que a menudo consiste en que humanos intenten diferenciar entre música generada por máquinas o creada por humanos. Este método se conoce como el test de Turing, que se realizó para evaluar DeepBach [22]. En este modelo, 1.272 personas de diferentes grupos de experiencia musical tomaron la prueba. La cual demostró que cuanto más complejo era el modelo, mejores resultados obtenía. La misma prueba se hizo con MusicVAE [23] y con MuseGAN [29]. Estos últimos reunieron a 144 usuarios divididos en grupos con diferentes niveles de experiencia musical y los sometieron a un cuestionario con preguntas predefinidas en las

que los usuarios debían votar en una escala del 1 al 5: complacencia armónica, ritmo, estructura, coherencia y calificación general.

Otros métodos implican calificar la música generada, en torno a diversos aspectos como la creatividad del modelo o la naturalidad de la pieza generada, entre otras preguntas, según el objetivo de generación del modelo. Un aspecto importante en las pruebas de audición es la variabilidad de la población elegida para la prueba en cuanto a su nivel de conocimientos musicales. Los oyentes deben recibir los mismos estímulos y escuchar las mismas piezas, y tener como referencia (si es aplicable) las mismas piezas creadas por humanos. También se debe tener en cuenta la fatiga auditiva, ya que puede inducir sesgos en los oyentes si escuchan muestras similares durante un período prolongado.

En resumen, las pruebas de audición son indispensables cuando se trata de generación de música, ya que proporcionan un mejor feedback sobre la calidad del modelo y también pueden ayudar a encontrar la mejor arquitectura de red neuronal o modelo de DL que se esté estudiando.

Capítulo 3

Redes LSTM y Transformer

En este trabajo se implementan dos ejemplos de modelos para la generación de música: uno basado en LSTM y otro basado en Transformer-XL (Jazz Transformer). A continuación, desarrollamos los aspectos teóricos de este tipo de redes.

3.1. Long Short-Term Memory

Las redes LSTM [17] (Long Short-Term Memory) son una evolución de las Redes Recurrentes (RNN), diseñadas para lidiar con el problema del desvanecimiento de gradiente ("*vanishing gradient problem*") [38]. Las LSTM son especialmente útiles en tareas que involucran secuencias de datos, como el procesamiento del lenguaje natural, las series temporales, o en este caso, la generación de música.

Las RNN tradicionales cuentan con un único estado oculto para transmitir la información a través del tiempo. Imaginemos una red capaz de predecir un único componente. Entonces, para describir el pasado de dicho componente, x_t, usaremos el estado oculto H-dimensional en el instante t, $h_{t-1} \in \mathbb{R}^H$. De esta manera, podremos determinar recursivamente el estado oculto del instante posterior, $t+1$:

$$h_t = f(x_t, h_{t-1}) \tag{3.1}$$

donde f es una función no lineal que depende de la arquitectura de la red. En teoría, esta definición permite llevar el seguimiento de cualquier secuencia larga. Sin embargo, en la práctica, al actualizar los pesos en la red mediante " *back-propagation*", nos encontramos que para secuencias muy largas, el gradiente puede tender a cero (desvanecerse) o infinito (explotar) [38], debido a la precisión finita de los números en computación.

Las LSTM solucionan este problema incorporando una célula de memoria, que le permite retener la información por un periodo más largo de tiempo. A su vez, esta unidad está controlada por un sistema de tres puertas: input gate, forget gate y output gate; que deciden que información añadir, eliminar, y cual será el output.

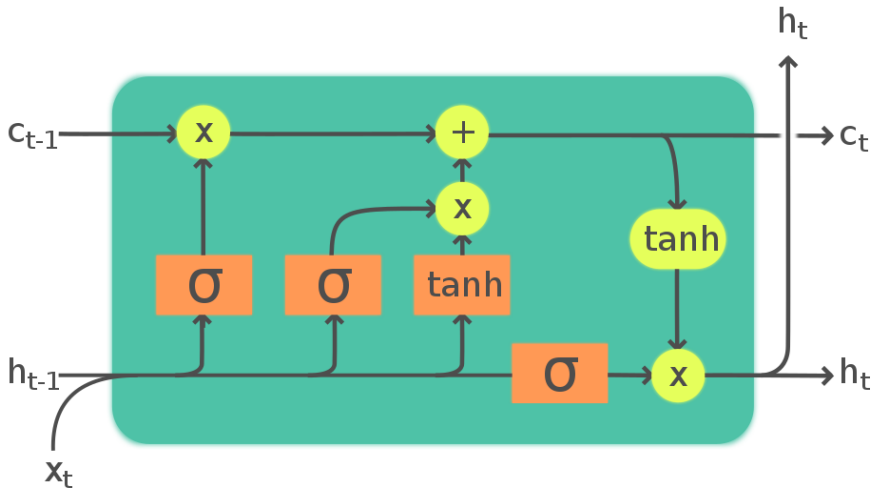


Figura 3.1: Esquema de una célula Long-Short Term Memory [39].

Forget Gate

Elimina la información que ya no es útil en su memoria. Depende del input de la red en el instante t, x_t; del output previo de la célula, h_{t-1} y de las matrices de pesos, W_f, y los bias asociados, b_k, que se van actualizando durante el entrenamiento. El resultado de este cálculo se le pasa a una función de activación (sigmoide), que tendrá un output binario: 0 si olvida la información y 1 si la mantiene. La ecuación de la Forget Gate es la siguiente:

$$f_t = \sigma(W_f \cdot [h_{t-1}, x_t] + b_f) \tag{3.2}$$

Input Gate

Decide qué nueva información debe agregarse a la memoria. Primero, regula la información con una sigmoide, i_t de manera similar a la Forget Gate, utilizando x_t, h_{t-1},la matriz de pesos, W_i y los bias b_i:

$$i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i) \tag{3.3}$$

A continuación, se crea un vector, $\hat{C}_t$, utilizando una tangente hiperbólica que contiene todos los valores de x_t, h_{t-1} y sus correspondientes pesos y bias:

$$\hat{C}_t = tanh(W_C * [h_{t-1}, x_t] + b_c) \tag{3.4}$$

Por último, se multiplica este vector, $\hat{C}_t$, por los outputs de la input gate y la forget gate para obtener la información útil:

$$C_t = f_t \circledast C_{t-1} + i_t \circledast \hat{C}_t \tag{3.5}$$

Output Gate

Extrae la información necesaria para generar el output de la célula, o_t. Además, genera el nuevo estado oculto que servirá de input en la siguiente iteración, h_t. Sus ecuaciones son las siguientes:

$$o_t = \sigma(W_o \cdot [h_{t-1}, x_t] + b_o) \tag{3.6}$$

$$h_t = o_t \circledast tanh(C_t) \tag{3.7}$$

donde W_o y b_o son la matriz de pesos y los bias que la definen.

3.1.1. LSTM profundas para la generación de música

La capacidad de las LSTM para capturar patrones temporales en datos secuenciales las hacen perfectas para la generación musical. Un modelo básico de este tipo, debe ser capaz de asimilar una secuencia de longitud variable y predecir el siguiente elemento en la lista. Esta secuencia debe contener toda la información necesaria: nota, duración, etc. De este modo, todas las piezas musicales que conformen el dataset de entrenamiento se segmentarán en secuencias sucesivas de una longitud, l, donde el elemento $l + 1$ será el output a predecir. Así, el entrenamiento de esta red consistirá en utilizar las secuencias de longitud l como input a la red, y el elemento siguiente como la etiqueta a predecir, que a su vez, formará parte de la próxima secuencia de input (figura 3.2).

Una vez el modelo esté entrenado, la generación de música puede hacerse a partir de una melodía aleatoria como punto de partida, o a partir de una melodía existente, de modo que la continúe (no necesariamente de la misma manera). En ambos casos, la salida en cada iteración se introducirá en el input de la siguiente iteración sucesivamente.

Por último, el modelo debe contar con una función para convertir la nueva pieza generada en un formato musical legible, como partituras MIDI o audio. Además, se le pueden añadir técnicas de postprocesamiento que mejoren la calidad de la posterior escucha.

3.2. Transformer

El Transformer original consiste en una estructura Encoder-Decoder (Figura 3.3) basada en mecanismos de atención [18]. Una de las principales diferencias con respecto a las LSTM, es que permite la paralelización de las secuencias de entrada, lo que incrementa considerablemente la eficacia de las GPU y su velocidad de entrenamiento. Además, los mecanismos de atención que utiliza, le permite asimilar secuencias mucho más largas. Todo esto, los ha llevado a un nuevo estado del arte en el NLP.

Por un lado, el encoder está compuesto por 6 capas idénticas, que a su vez, están formadas por 2 subcapas. La primera la denominan *multi-head self-attention mechanism*, que da contexto a cada elemento del input. La segunda es una red *feed-forward* (FFN), que aplica dos transformaciones lineales, con una ReLU en medio:

$$FFN(x) = max(0, xW_1 + b_1)W_2 + b_2 \tag{3.8}$$

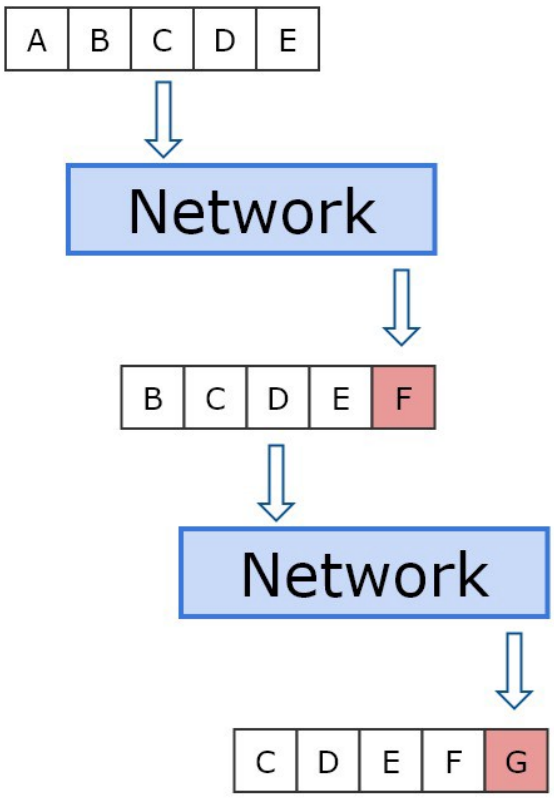


Figura 3.2: Entrenamiento de un modelo para generación musical basado en LSTM [40].

Donde W_1, b_1 ,W_2 y b_2 son los pesos y bias asignados a dicha capa y x su entrada.

Por otro lado, el decoder está conformado por otras 6 capas compuestas por dos subcapas de atención seguidas de una FFN. Además, a la primera subcapa de atención se le añade una máscara para impedir al modelo acceder a información futura. Por último, es importante mencionar que los inputs del encoder y decoder, están previamente codificados para añadir información sobre la posición relativa de los elementos de cada secuencia.

Los Transformer son modelos especializados en la traducción de texto. De esta manera al encoder se le introduce como input una frase en el idioma original y al decoder se le introduce la frase ya traducida. Así, el modelo procurará generar cada palabra de la traducción de manera paralelizada.

3.2.1. Mecanismo de atención o *Self-Attention*

La principal novedad que presentan los Transformer, es su mecanismo de atención, el cual sirve para dar contexto a cada elemento de una secuencia [18]. Este mecanismo consiste en asignar una medida de cuánto se relaciona cada elemento de una secuencia con el resto mediante una función, lo que permite analizar secuencias mucho más largas.

La atención viene descrita por una función que depende de tres vectores, que en el paper original se denominan *query* (q) y *key* (k), de longitud d_k, y *value* (v) de longitud d_v. Estos vectores son proyecciones lineales aprendidas de los elementos de entrada. Así, el output de

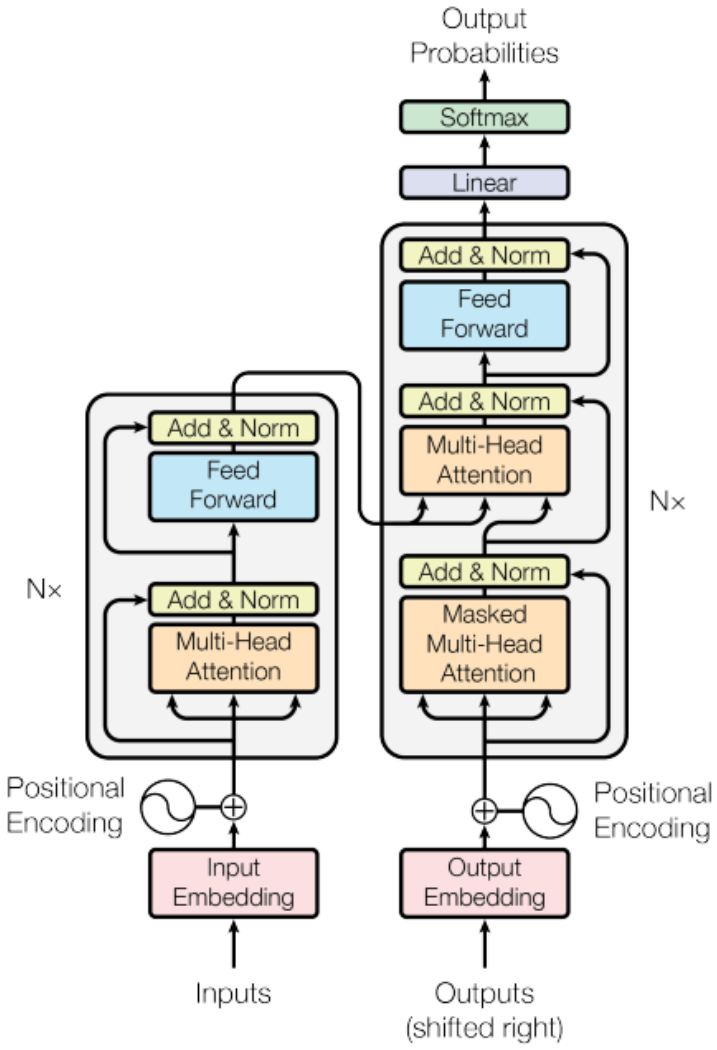


Figura 3.3: Arquitectura de un Transformer [18].

la función de atención será una suma pesada de todos los valores, v, donde los pesos vienen dados por una función de compatibilidad (producto escalar) entre q y k. En la práctica se calcula la función de atención sobre un set de múltiples *queries* simultáneamente, por lo que se unificarán mediante matrices. Así, se define la función de atención como:

$$Attention(Q, K, V) = softmax(QK^T/\sqrt{d_k})V \tag{3.9}$$

Además, para solucionar los problemas que pueden derivar de utilizar una única proyección lineal de las matrices, se realizan proyecciones distintas, que se aprenden durante el entrenamiento. Sobre cada una de estas proyecciones se aplica, por tanto, la función de atención. Por último, cada output de dicha función se concatena y se vuelve a proyectar, resultando el output final (Figura 3.4). Esto es lo que se conoce como *Multi-head attention mecanism* .

De esta manera, el modelo consigue calcular una medida de la relación entre los dis-

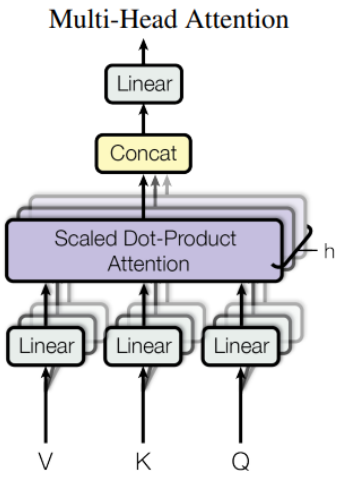


Figura 3.4: Esquema Multi-head attention mecanism con h cabezas[18].

tintos eventos

3.2.2. Transformer-XL

Aunque el Transformer suponía un gran avance en el desarrollo del NLP, mantenía algunas limitaciones con respecto a su memoria. El Transformer original, por razones prácticas, necesitaba dividir las secuencias muy largas en segmentos más cortos, lo que daba lugar a una fragmentación del contexto original de la secuencia. Para solucionarlo, en el Transformer-XL [41] se añadió una memoria caché para almacenar la información del segmento previo. Básicamente, se concatenan los estados ocultos del segmento anterior al input actual para calcular la atención. De esta manera, el modelo es capaz de mantener una mayor coherencia a lo largo de la evaluación.

En la figura 3.5 se puede observar un esquema del decoder de un Transformer original que utiliza segmentos de longitud 4. En este ejemplo se ve la limitación del contexto en la predicción del output, que se limita a los cuatro elementos del propio input.

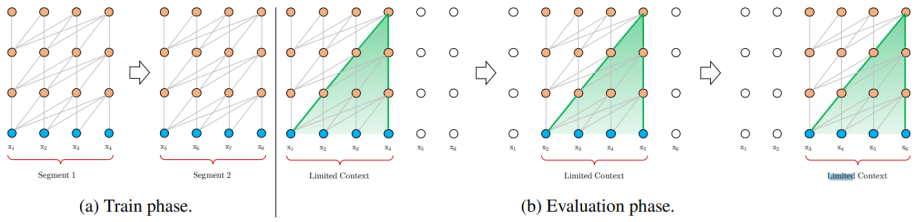


Figura 3.5: Esquema de un Transformer con segmentos de longitud 4 [41].

Sin embargo, en 3.6 se puede observar cómo el contexto es mucho más amplio. En este caso, a la hora de calcular la self-attention, mientras que las queries provienen de los elementos del segmento actual, los keys y values se obtienen de la unión de los dos segmentos. De esta forma, se calcula la atención para dos segmentos sucesivos.

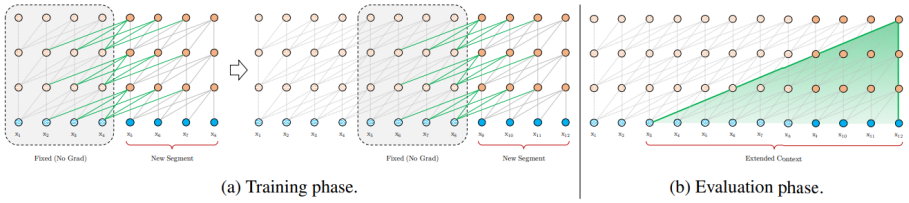


Figura 3.6: Esquema de un Transformer-XL con segmentos de longitud 4.

Como hemos mencionado, los modelos Transformer están especializados en la traducción de texto, aún así, se pueden adaptar a la generación. Ejemplos de esto son los CTRL [42] y GPT [43]. Estos modelos están formados únicamente por el decoder, que utiliza como input una secuencia (en este caso de texto) y trata de predecir como output el elemento siguiente. De esta manera, el entrenamiento de este tipo de redes consiste en introducir como input un secuencia determinada y utilizar como etiqueta la misma secuencia desplazada un paso por delante 3.7. Así, con ayuda de la máscara aplicada en la capa de atención, el modelo procurará predecir el elemento siguiente a cada uno de los subsegmentos del texto original. De esta manera, la salida será un vector de probabilidades de longitud igual al tamaño del vocabulario, que comparará con las etiquetas para calcular una pérdida que permita actualizar los pesos del modelo.

La adaptación de estos modelos a la generación de música es simple. Consiste en interpretar el lenguaje musical como un lenguaje sintáctico, en el que cada palabra se corresponderá a un evento musical. Así, la generación de música se convierte en un problema estándar de NLP. Más adelante, desarrollaremos el ejemplo de Jazz Transformer, que utiliza una arquitectura basada en este tipo de modelos para la generación de piezas musicales.

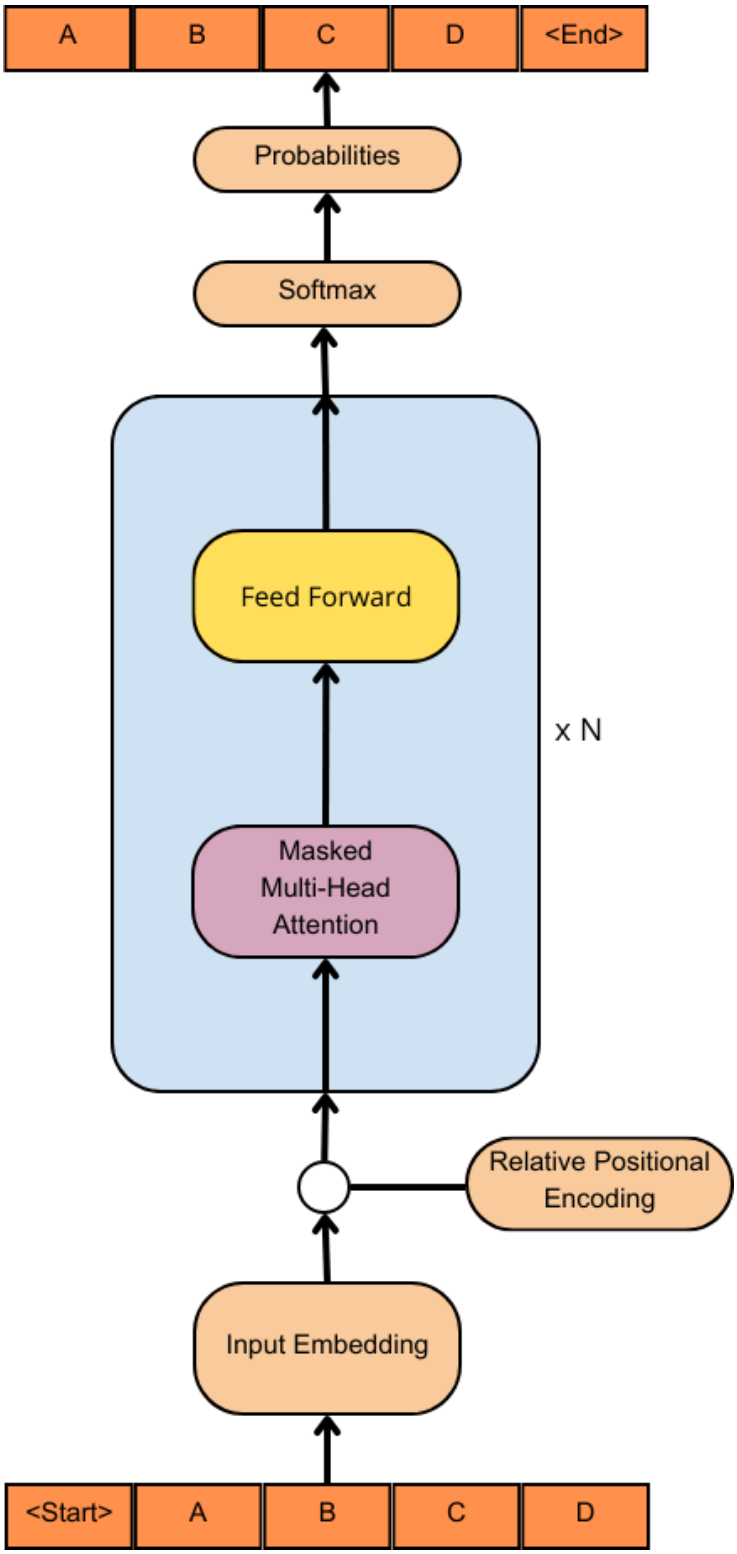


Figura 3.7: Esquema del entrenamiento de un Transformer-XL, con N bloques, para la generación de texto.

Capítulo 4

Implementación y entrenamiento de modelos

4.1. LSTM

Para el modelo basado en LSTM, emplearemos una arquitectura inspirada en el modelo elaborado por Jordan J. Bird [44], que expandiremos para integrar una nueva pista de acordes que acompañe a la melodía.

4.1.1. Preparación de los datos

Como ya hemos mencionado, en este trabajo emplearemos el Weimar Jazz Database, que contiene 456 transcripciones de solos de jazz en instrumentos de viento (saxofon, trompeta, clarinete...). A diferencia otros conjuntos de datos, el Weimar Jazz Database, no solo cuenta con los archivos MIDI, sino también una amplia variedad de información adicional organizada en diferentes tablas. Uno de los principales desafíos en este tipo de modelos radica en definir de manera óptima todas las cualidades de una pieza musical. Para este modelo, se utiliza una representación basada en eventos musicales inspirada en el formato MIDI [20] [45]. Para ello, representamos cada nota en una melodía por su tono, su duración y la distancia entre dicha nota y la anterior. Además, para añadir la base armónica, se etiquetará cada nota con el acorde que la acompaña. A continuación se definirá cada una de estas variables.

Note

La secuencia de notas MIDI de cada pieza musical. Es un vector de números enteros en el que cada número representa una de las notas MIDI (figura 2.1). Aunque existan 128 notas MIDI, en la práctica, los instrumentos nos suelen alcanzar las notas más graves o más agudas. Un piano moderno, por ejemplo, suele tener 88 teclas, es decir, es capaz de hacer sonar 88 notas diferentes. Estas van desde un La_0 hasta un Do_8. En el caso del Weimar Jazz Database, al tratarse de instrumentos de viento, este rango se reduce. En nuestro caso, contamos con 62 notas, que van desde el Do_3 hasta el $DO\#_8$.

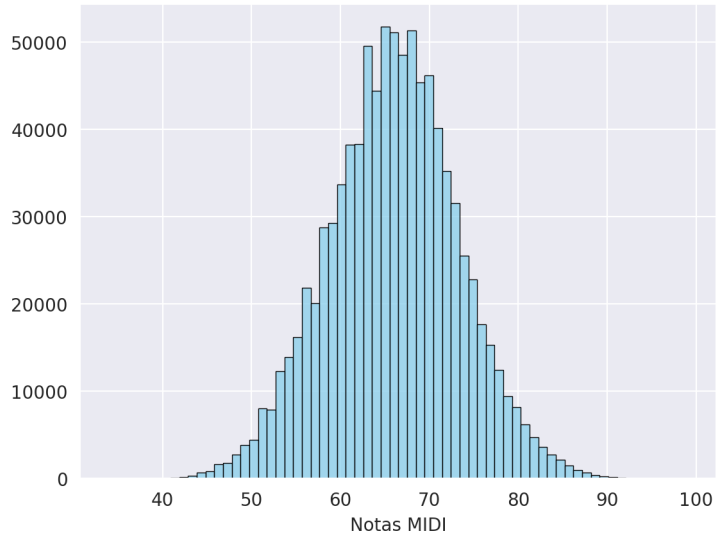


Figura 4.1: Histograma de notas contenidas en el dataset.

Duration

Acompañando a cada nota, va su duración. Se trata de una variable numérica de tipo real, que está medida en pulsos y toma como referencia la negra (es decir, una duración de 1.0 equivale a una negra, mientras que una duración de 0.5 equivale a una corchea). Por otro lado, como los MIDI no están cuantizados, y para reducir el número de clases, tomaremos como unidad mínima de tiempo la fusa (1/8 de negra) y como máxima la redonda (4 negras). También tendremos en cuenta los tercios de negra, para capturar figuras musicales muy utilizadas en el Jazz como los tresillos. A partir de ahí, redondearemos todos los valores de tiempo obtenidos. Esto nos permite reducir el número de duraciones diferentes, lo que facilitará el entrenamiento. De esta manera, quedarán 33 clases distintas.

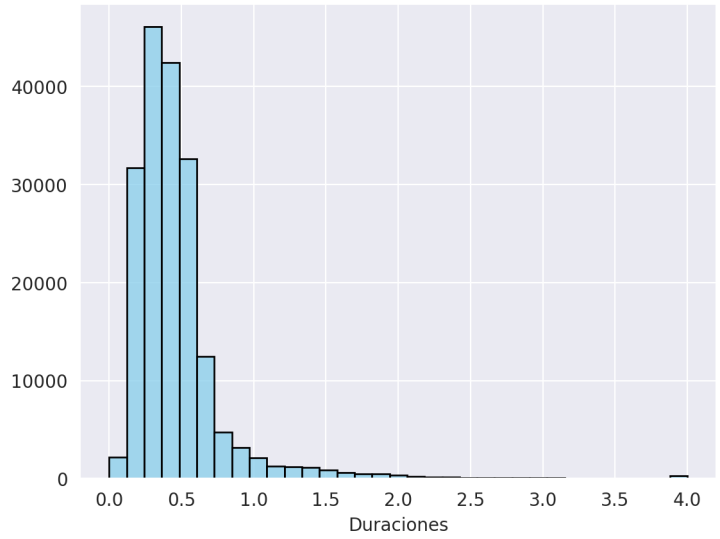


Figura 4.2: Histograma de duraciones contenidas en el dataset.

Offset

Para ubicar las notas en el tiempo, utilizaremos el offset. El offset se define como el tiempo transcurrido entre dos eventos, es decir, entre cada nota. En este caso, lo representaremos como el timepo que ha tardado en sonar una nota con respecto a la anterior (es decir, el offset de la primera nota será 0). Similar a la duración, se mide en pulsos y se preprocesa de la misma manera. La presencia del Offset es muy importante porque nos permite capturar los silencios de manera implícita, al relacionarlo con la duración de las notas. Igual que con las duraciones, tendremos 33 clases.

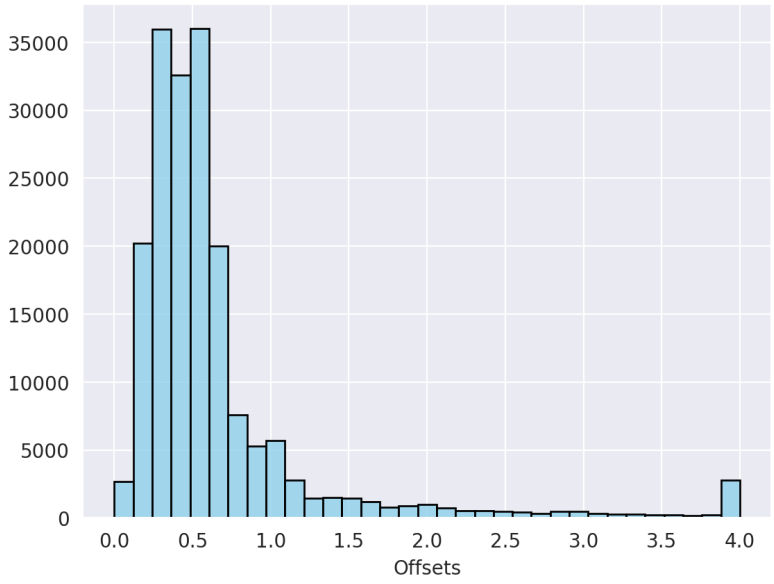


Figura 4.3: Histograma de offsets contenidos en el dataset.

Chord

Por último, se añadirá el acorde que acompaña a cada nota. Esta es una variable categórica, que utiliza cadenas de texto para indicar qué acorde se toca junto a cada nota. Dado que en el dataset original existen 418 acordes diferentes, se llevará a cabo un proceso de preprocesamiento para simplificar la representación. En este proceso, los acordes se reducirán a sus formas más fundamentales, distinguiendo solo entre acordes mayores y menores. De esta manera, se definirán un total de 24 acordes diferentes (dos tipos de acorde por cada nota), al que hay que añadir una nueva clase para indicar aquellos momentos en los que sobre una nota no suena ningún acorde. Entonces, tendremos una variable categórica de 25 clases para representar los acordes de una pieza musical.

En resumen, nuestro modelo utilizará cuatro variables categóricas como entrada: notas, duración, offset y acordes (tabla 4.1). Estas variables deben ser divididas para el entrenamiento en N segmentos, cada uno de longitud l, junto con sus correspondientes resultados a predecir. En este contexto, hemos optado por emplear secuencias de longitud $l = 100$ con el objetivo de predecir la nota posterior.

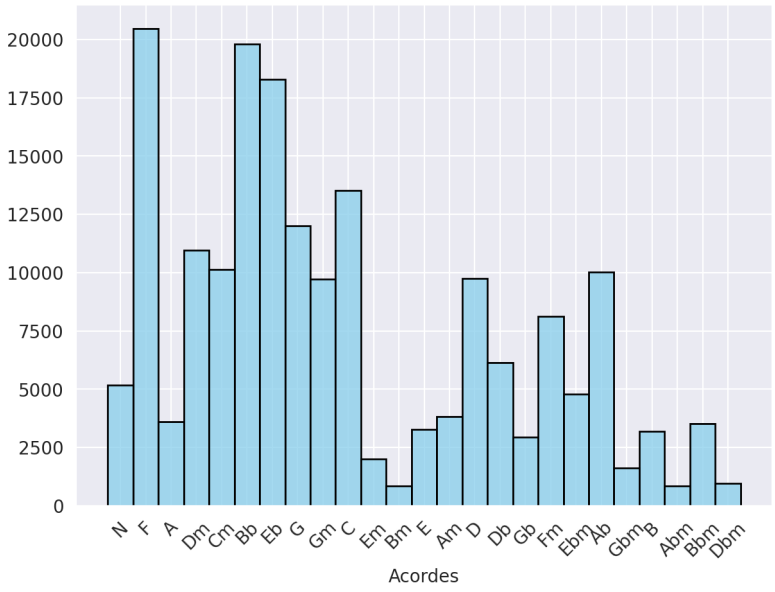


Figura 4.4: Acordes según su presencia en el dataset.

Después, trataremos de aumentar el dataset original utilizando la transposición de las piezas. Este método, consiste en trasladar de tono tanto las notas individuales como los acordes, dejando invariantes los intervalos entre ellos. En nuestro caso, transpondremos las melodías dos tonos arriba y dos tonos abajo, multiplicando por 4 el número de segmentos del dataset. Esto resulta en un conjunto de entrenamiento que consta de casi 200,000 segmentos.

4.1.2. Arquitectura y entrenamiento

El modelo está construido con Keras, la API de Tensorflow, y podemos dividir su arquitectura en tres partes diferentes. La primera estaría formada por cuatro bloques idénticos separados, para el input de cada variable. Posteriormente se juntarían en un nuevo bloque para asimilar las cuatro entradas. Justo después, el modelo volverá a ramificarse en cuatro bloques, para generar cada uno el output correspondiente 4.6.

Nota MIDI	Acorde	Duración (Negra)	Offset (Negra)
65	Bb	0.5	0.0
63	Bb	0.625	1.125
58	Bb	0.25	0.75
61	Bb	0.875	0.375
63	Bb	0.5	1.0
58	Bb	0.75	1.25
58	Bb	1.75	1.125
57	Gm	0.5	0.5
60	Gm	0.375	0.625
58	Cm	0.5	0.5
55	Cm	0.375	0.625
58	Cm	0.375	1.0
61	F	1.0	0.375
60	F	0.5	1.125

Tabla 4.1: Representación de entrada para el modelo LSTM de los primero 4 compases de Art Pepper- Anthropology (Figura 4.5). Tanto duración como offset están presentadas teniendo como referencia la duración de una negra. En los cuatro casos se trata de variables categóricas..

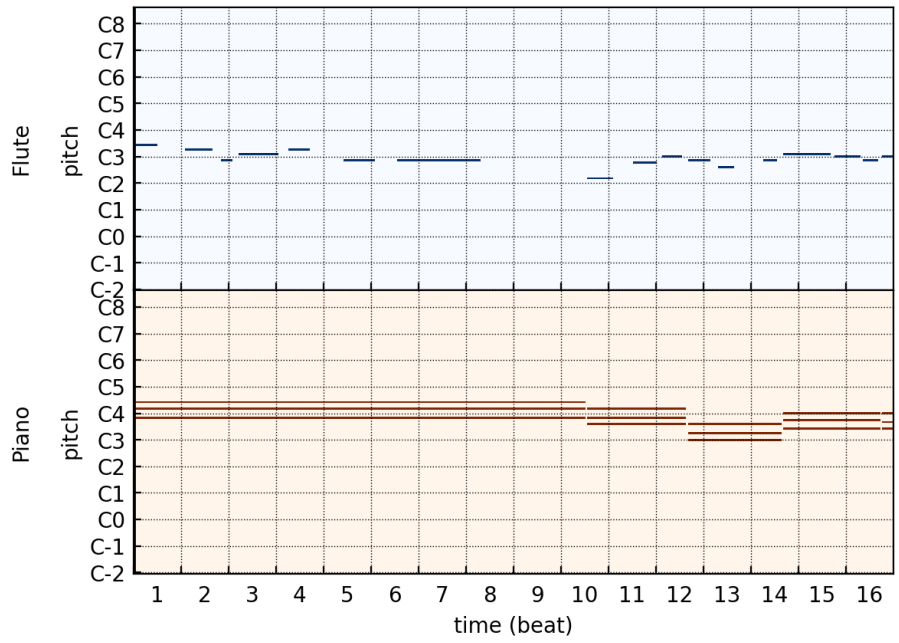


Figura 4.5: Piano roll de los cuatro primeros compases de Art Pepper - Anthropology (Tabla 4.1).

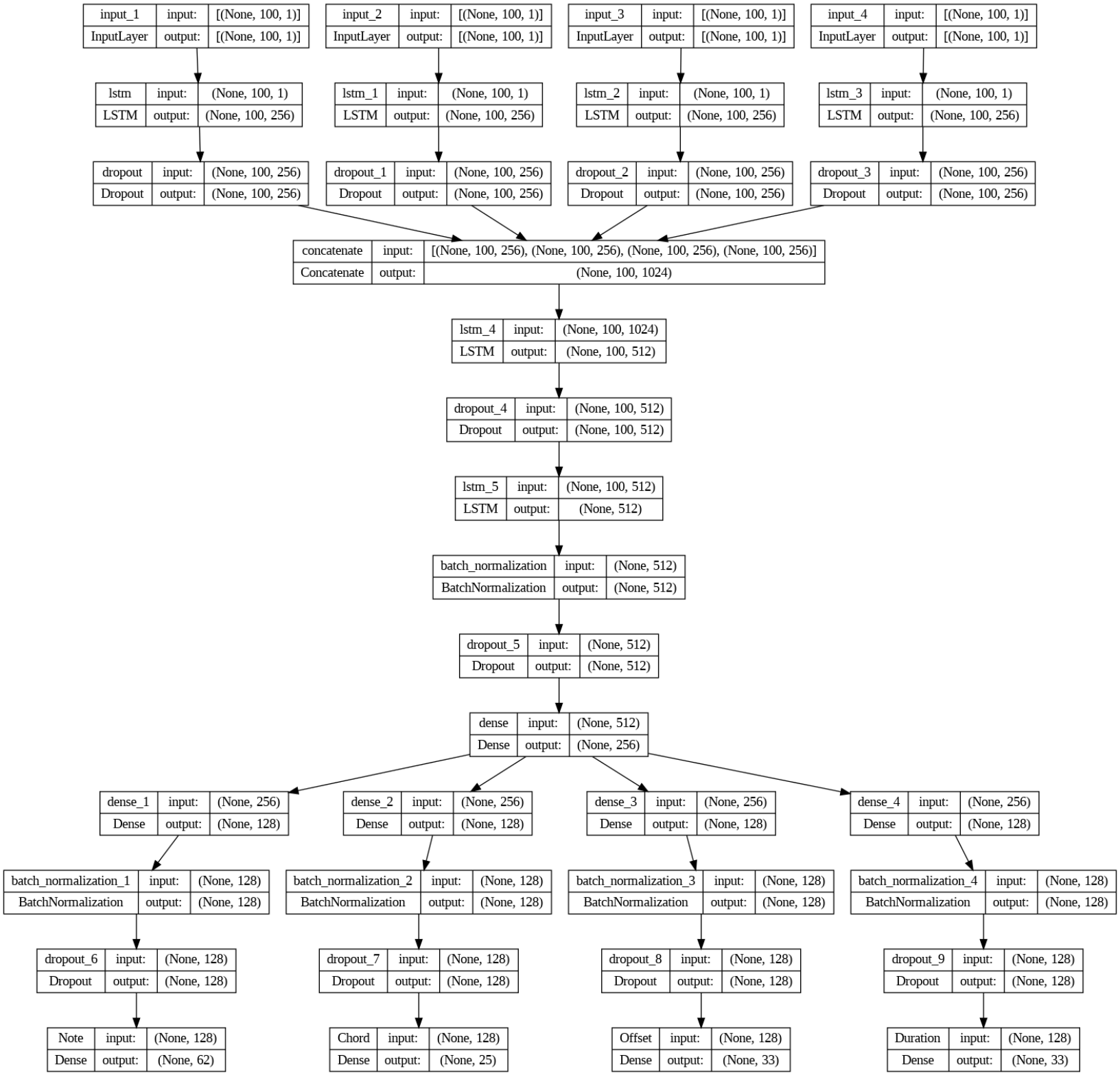


Figura 4.6: Estructura del modelo LSTM.

Los bloques de entrada tienen, cada uno como input, una secuencia de 100 valores, y están formados por una capa LSTM de 256 unidades, seguidas de un Dropout con una tasa del 20 %. Esta capa sirve como medida de regularización para prevenir el overfitting [46]. Consiste en "desactivar" aleatoriamente un porcentaje predeterminado de neuronas (en este caso, el 20 %) durante el entrenamiento, de modo que no participen, ni en la predicción, ni en el backpropagation. Así, la red neuronal no podrá depender en exceso de ninguna neurona en particular, lo que la obliga a aprender características más robustas y generalizables de los datos. Estos bloques se encargan de interpretar y transformar cada una de las cuatro características por separado.

A continuación, los cuatro bloques se unen en uno único, conformado por dos LSTM con 512 unidades, dos Dropout del 30 %, una capa de Batch Normalization y una capa densa de 256 neuronas y ReLu como función de activación. Donde el Batch Normalization (normalización por lotes) es una técnica que se utiliza para mejorar la velocidad de convergencia y la estabilidad de las redes neuronales. Se basa en normalizar los valores de activación de una capa oculta en cada batch de entrenamiento. Este bloque de capas se encarga de asimilar todo lo aprendido en los bloques previos y combinarlo.

Por último, tras la capa densa, la red se vuelve a separar en cuatro ramas para la clasificación. Esta parte está formada por una capa densa inicial con 128 neuronas y ReLu, otra de Batch Normalization y otro Dropout del 30 %, para acabar con una última capa densa de clasificación que utiliza softmax. Las dimensiones de esta última capa dependen del número de clases de cada característica (62 notas diferentes, 33 duraciones y offsets distintos y 25 acordes), que será codificada mediante One-Hot Encoding.

Para el entrenamiento, al tratarse todas de variables categóricas, utilizaremos la Categorical Cross Entropy como función de pérdidas con un optimizador Adam y batches de 64 secuencias. Como ya hemos comentado, cada secuencia consistirá en un vector de 4 características y 100 elementos, que el modelo empleará como input para predecir el elemento 101. De este modo, con el vector de probabilidades de salida, el modelo calculará la función de pérdidas para actualizar sus pesos. Así, obtenemos un modelo con 6.5 millones de parámetros, que entrenamos hasta las 116 épocas con la GPU V100 de colab, lo que supuso cerca de 6.5 horas de entrenamiento.

En la figura 4.7, se presentan los valores de la función de pérdida tanto para el conjunto de entrenamiento como para el de validación (90 % y 10 %, respectivamente). Aunque es evidente el overfitting de la época 116 y el modelo parece óptimo alrededor de la época 20, se ha optado por utilizar el modelo sobreentrenado. Esto se debe a que las secuencias generadas en la época 20 carecían de cualquier tipo de cualidad musical. Por eso, se decidió elegir el modelo con menos pérdida de entrenamiento (época 116).

4.1.3. Generación de melodías

Una vez entrenado el modelo, la generación de nueva música se hará partiendo de una melodía previa, con un formato idéntico al de los datos de entrenamiento (cuatro secuencias de longitud $l = 100$). Para generar música, el modelo se basa en estas cuatro secuencias de entrada y predice las notas siguientes una a una. Cada predicción se agrega a un vector que almacena la pieza musical generada. Luego, este vector de predicciones se convierte a formato MIDI para su reproducción utilizando la biblioteca Music21.

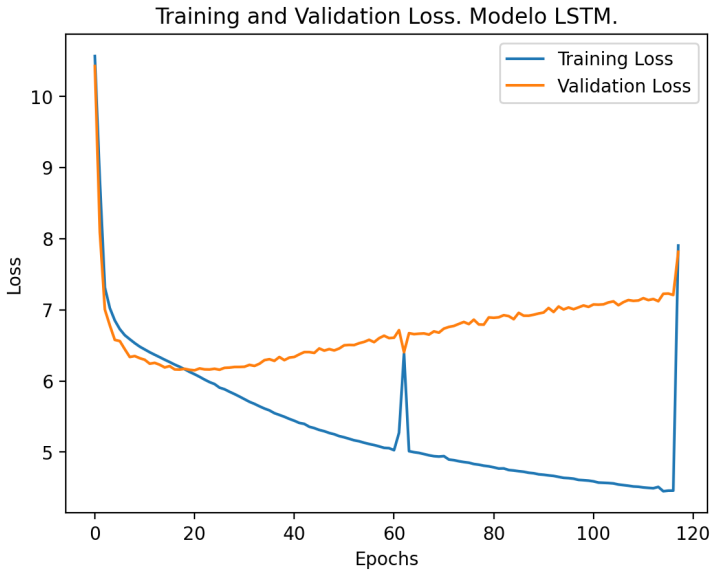


Figura 4.7: Función de pérdidas del modelo LSTM durante el entrenamiento.

El resultado generado consiste en un archivo MIDI de duración variable compuesto por dos canales. El primer canal contiene la melodía, que en las piezas originales era interpretada por un instrumento de viento. El segundo canal contiene la progresión de acordes sobre la cual suena la melodía. Estos acordes se generan a partir de sus etiquetas, superponiendo tres notas: la fundamental (la nota que da nombre al acorde), la tercera (que puede ser mayor o menor según el tipo de acorde) y la quinta.

En concreto, las nuevas melodías se construirán a partir de secuencias del conjunto de datos de entrenamiento seleccionadas de manera aleatoria. Esto implica elegir un vector de notas, duraciones, desplazamientos (offsets) y acordes, cada uno tomado de una secuencia diferente del conjunto de datos de entrenamiento. Luego, estos elementos se combinarán para crear una nueva entrada musical que servirá como punto de partida para la generación. De este modo, para el cálculo de métricas de dichas piezas, se ha optado por crear un conjunto de 400 melodías de 500 notas cada una.

Por último,

4.2. Jazz Transformer

4.2.1. Preparación de los datos

Como se comentó anteriormente, la representación de datos que utilizan en [34], es una versión modificada de la REMI [47], basada en eventos y adaptándola a la estructura del Weimar Jazz Database. Se trata de una representación sintáctica, en la que los eventos se van sucediendo en orden. Es por tanto un problema de NLP clásico, en el que el lenguaje empleado es el musical, intercambiando palabras por eventos de notación musical. A continuación, trataremos de describir los distintos tipos de eventos (palabras) que se emplean en el Jazz Transformer para codificar la información musical a un lenguaje interpretable

por el modelo. Estos eventos, se pueden dividir en cuatro categorías.

Eventos relacionados con las Notas

Cada nota en una melodía se puede representar mediante tres eventos: NOTE-VELOCITY, NOTE-ON y NOTE-DURATION.

- NOTE-VELOCITY: indica la intensidad con que suena cada nota, esta cuantizada en 32 bins.
- NOTE-ON: se refiere al tono de la nota, el cual es un número entero en el rango [0, 127].
- NOTE-DURATION: es la duración de la nota medida en semifusas (1/64 de redonda) y con valores entre 1 y 32. Es decir, la duración máxima comprendida para la melodía es la blanca.

Eventos relacionados con la métrica

Para ubicar las notas en el tiempo utilizan una combinación entre el compás, BAR, y la posición, POSITION, de la nota, en lugar de utilizar el offset. Además, para definir el ritmo al que se debe tocar cada nota, se añaden las variables de TEMPO y TEMPO-CLASS.

Los eventos BAR se añaden al inicio de cada compás, y a su vez, se dividen en 64 subunidades representadas por POSITION. De este modo POSITION = 16 maracaría el comienzo del segundo pulso en un compás. Por otro lado, TEMPO-CLASS y TEMPO indican el ritmo en BPM de cada pulso en una composición.

Eventos relacionados con los acordes

Igual que con el modelo anterior, en Jazz Transformer también llevan a cabo un proceso de reducción de los tipos de acorde, aunque de una manera más compleja. En este caso, representan cada acorde mediante tres variables: CHORD-TONE, CHORD-TYPE, y CHORD-SLASH.

- CHORD-TONE: determina la tonalidad del acorde mediante 12 clases distintas, que son las 12 notas de la escala cromática (C, C#, D...).
- CHORD-TYPE: representa la cualidad del acorde que suena, mediante la combinación de diferentes notas en relación a la nota fundamental, e.g., un acorde de séptima (CHORD-TYPE = 7), equivaldría a [0, 4, 7, 10].
- CHORD-SLASH: permite alterar el bajo del acorde. Si coincide con CHORD-TONE, el bajo será la fundamental, mientras que si el evento es distinto, el bajo del acorde cambiará [1].

[1] En música, un acorde *"slash"*, es aquel que no utiliza su nota fundamental como bajo, si no que emplea otra de las notas del acorde. De este modo, un acorde de do séptima con bajo en Sol se escribiría C7/G. Por tanto, este acorde en la representación de Jazz Transformer sería [CHORD-TONE(C), CHORD-TYPE(7), CHORD-SLASH(G)]

Eventos relacionados con la estructura

En el trabajo de [34], se incorporan eventos diseñados para capturar la estructura de una melodía. Destaca el evento *PHRASE*, que se coloca al inicio de cada frase musical, proporcionando al modelo indicaciones sobre cuándo tomar respiros.[en]tre secuencias de notas. Además, se introducen los *MLU* (Unidades de Nivel Medio, según [48]), que actúan como eventos intermedios encargados de categorizar y subdividir estas frases en nueve categorías principales (línea, fraseo, tema, cita, melodía, ritmo, expresivo, fragmento y vacío). A través de estos eventos, se busca representar el rol que desempeña cada grupo de notas dentro de una frase o en la frase en su totalidad. Por último, se incluyen los eventos *PART* (*START* y *END*) y *REPETITION* (*START* y *END*), que demarcan el inicio y final de cada sección y repetición (también se indica el número de veces que se ha repetida una parte, hasta un máximo de 6). Esto contribuye a que el modelo genere estructuras más complejas, con la repetición de motivos y frases de manera más coherente.

4.2.2. Arquitectura y entrenamiento del Jazz Transformer

Debido a la gran cantidad de notas que contiene cada solo, para el Jazz Transformer se escogió como modelo un Transformer-XL [41]: una versión mejorada del primer Transformer, que introduce la recurrencia entre segmentos de entrenamiento.

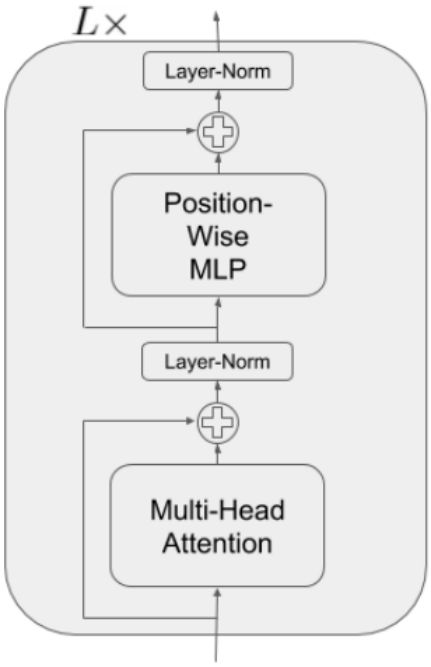


Figura 4.8: Bloque principal del Transformer-XL [49]. Para Jazz Transformer, $L = 12$

Como ya se mencionó, el Jazz Transformer carece de encoder, por lo que está formado por 12 bloques principales similares a los del Transformer original (Figura 4.8). Estos bloques están formados por una capa de atención de 8 cabezales y una capa de FFN. Además, a estas capas de atención se les añade una capa de máscara para evitar que el

index	EVENT	VALUE	index	EVENT	VALUE
0	Part-Start	I	60	Chord-Tone	Bb
1	Rep-Start	1	61	Chord-Type	6
2	Bar	-	62	Chord-Slash	Bb
3	Position	0	63	Phrase	-
4	Tempo-Class	4	64	MLU-Type	melody
5	Tempo	191.67	65	Note-Velocity	21
6	Position	16	66	Note-On	65
7	Tempo-Class	4	67	Note-Duration	8
8	Tempo	203.33	68	Position	16
9	Position	32	69	Tempo-Class	4
10	Tempo-Class	4	70	Tempo	215.00
11	Tempo	215.00	71	Note-Velocity	23
12	Position	48	72	Note-On	63
13	Tempo-Class	4	73	Note-Duration	10
14	Tempo	203.33	74	Position	28
15	Bar	-	75	Note-Velocity	21
16	Position	0	76	Note-On	58
17	Tempo-Class	4	77	Note-Duration	5
18	Tempo	203.33	78	Position	32
19	Chord-Tone	Bb	79	Tempo-Class	4
20	Chord-Type	6	80	Tempo	203.33
21	Chord-Slash	Bb	81	Bar	-
22	Phrase	-	82	Position	0
23	MLU-Type	melody	83	Tempo-Class	4
24	Note-Velocity	21	84	Tempo	215.00
25	Note-On	65	85	Chord-Tone	F
26	Note-Duration	8	86	Chord-Type	7
27	Position	16	87	Chord-Slash	F
28	Tempo-Class	4	88	Position	33
29	Tempo	215.00	89	Note-Velocity	22
30	Note-Velocity	23	90	Note-On	61
31	Note-On	63	91	Note-Duration	15
32	Note-Duration	10	92	Position	48
33	Position	28	93	Tempo-Class	4
34	Note-Velocity	21	94	Tempo	203.33
35	Note-On	58	95	Position	49
36	Note-Duration	5	96	Note-Velocity	24
37	Position	32	97	Note-On	60
38	Tempo-Class	4	98	Note-Duration	8
39	Tempo	203.33	99	Position	58

Tabla 4.2: Representación sintáctica de Art Pepper- Anthropology.

modelo acceda a información futura. Como input del modelo se emplean secuencias de longitud 512, que serán codificadas también en un espacio de dimensión 512 (cada uno de los eventos está definido por un vector normalizado de dimensión 512). Estos embedding serán analizados por los 12 bloques de atención, que tendrán como salida un vector de probabilidades de longitud 450 (la longitud del vocabulario). Esto resulta en un modelo con alrededor de 41 millones de parámetros entrenables.

En este trabajo, además de evaluar el modelo preentrenado de Jazz Transformer [2], se tratará de reentrenar el mismo para comparar los resultados. Se utilizará también la GPU V100 de colab, con optimizador Adam, learning rate 1e−4 y batch size de 8. Cada uno de estos batches está conformado por 3 secuencias consecutivas, que se introducirán de manera ordenada como input. Así, se consigue aumentar el contexto del modelo mediante un vector de memoria, de longitud 512, que mantiene el flujo de información entre secuencias consecutivas (tal y como se comentó en la sección 3.2.2)). Además, para mejorar el entrenamiento, se transpone cada secuencia aleatoriamente en un rango de [-3,+3] semitonos en cada época.

En la figura 4.9 se muestran las funciones de entrenamiento de Jazz Transformer, que presentan un comportamiento similar al del modelo LSTM 4.7. Sin embargo, la validación que se hace en [34] no consiste en interpretar las funciones de pérdidas. Por el contrario, utilizan la métrica de "predicción de continuaciones´´ definida en la sección 2.7.1. De este modo, concluyen que el modelo óptimo es aquel que alcanza un loss de 0.25. En nuestro caso, esto se corresponde con la época 250. Así, hemos tardado un total de 11 horas en alcanzar el modelo que proponen.

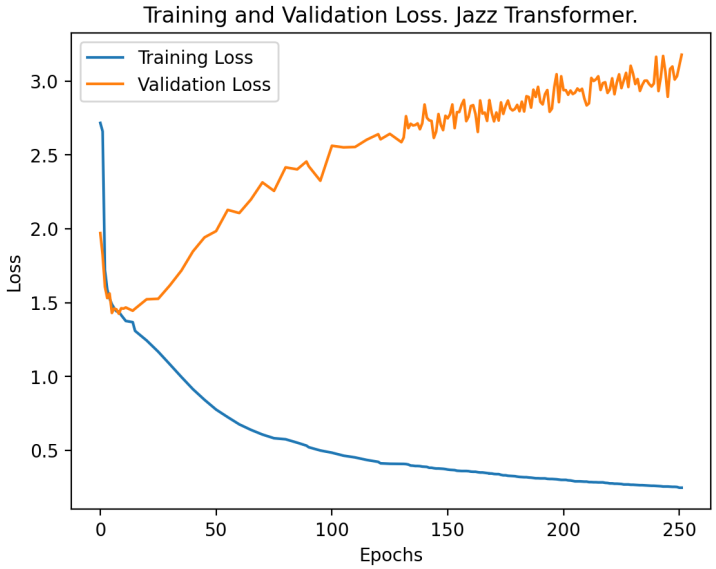


Figura 4.9: Funciones de pérdidas de Jazz Transformer.

A continuación, mostramos una breve tabla como resumen de las características de ambos modelos (tabla 4.3):

[2] https://github.com/slSeanWU/jazz_transformer

	Modelo LSTM	**Jazz Transformer**
Nº de parámetros	$6.6 \cdot 10^6$	$41.2 \cdot 10^6$
Dimensióm de entradas/salidas	100	512
Épocas ejecutadas	116	250
Tiempo de entrenamiento (horas)	6.5	11.0

Tabla 4.3: Breve resumen descriptivo de algunas de las características de ambos modelos.

4.2.3. Generación de melodías

La generación de nueva música en Jazz Transformer se hace de cero, partiendo de unas bases muy generales a partir de las cuales se va construyendo la frase. En primer lugar, se inicializan los eventos de comienzo de frase, comienzo de compás, etc. Además, se establece un tempo aleatorio para la pieza y se elige, también de manera aleatoria, un acorde con el que comenzar. A partir de ahí, el modelo irá infiriendo los nuevos eventos de manera sucesiva con una serie de reglas definidas para asegurar la coherencia en los mismos. Así, si el último evento fue un NOTE-VELOCITY, el siguiente deberá ser un NOTE-ON. En caso contrario, el modelo volverá a predecir la nueva palabra hasta dar con el evento adecuado. Este proceso se repite el número de veces necesario hasta completar 32 compases. En este trabajo, para evaluar las piezas generadas por Jazz Transformer hemos tratado de generar 500 composiciones diferentes de 32 compases.

Capítulo 5

Discusión y resultados

A continuación, procederemos a evaluar el rendimiento de cada uno de los modelos y su capacidad para crear música de manera "realista". Para ello, utilizaremos algunas de las métricas objetivas descritas en la sección 2.7.1. De esta manera, intentaremos analizar de forma objetiva las características tonales, rítmicas y estructurales de las melodías generadas. Además, trataremos de estudiar mediante la escucha de distintos ejemplos de cada conjunto si existe una relación entre estas métricas objetivas y el resultado final.

Para evaluar estas métricas utilizaremos los resultados de ambos modelos, los cuales compararemos con el propio dataset de entrenamiento. En teoría, estas métricas deberían ser capaces de recoger las características que definen a las distintas composiciones, y que definen su género musical. Por ello, el objetivo de los modelos será alcanzar unos resultados lo más parecidos posible a los del Weimar Jazz Database.

En la tabla 5.1 se muestran los resultados de las métricas de evaluación . En este caso, $\mathcal{H}_1$ y $\mathcal{H}_4$ son las entropías del histograma de clases para segmentos de 1 y 4 compases, $\mathcal{SC}_1$ y $\mathcal{SC}_4$ las consistencias de escala para uno y cuatro compases y $\mathcal{SI}_3^8$, $\mathcal{SI}_8^{15}$ y $\mathcal{SI}_{15}$ los indicadores de estructura a corto, media y largo plazo. Para el análisis, hemos generado tres conjuntos con un longitud similar, teniendo en cuenta que el Weimar Jazz Database cuenta con 459 solos y un total de casi 200 mil notas. De este modo, se han generado 400 MIDI de 500 notas con el modelo LSTM (200 mil notas) y 500 MIDI de 32 compases (unas 150 mil notas) con el Jazz Transformer preentrenado y reentrenado. En la figura 5.1 se muestra un ejemplo de cada uno de estos conjuntos junto con una muestra real de una de las melodías de Weimar Jazz Database. Además, se ha creado un repositorio de github, donde se pueden encontrar varios ejemplos de melodías generadas y demás material relacionado con el trabajo: https://github.com/sergioconde44/Music_Generation_TFM

5.1. Métricas armónicas

El valor de la entropía proporciona información sobre la incertidumbre en la tonalidad. Una entropía baja indica una gran estabilidad tonal, mientras que una entropía alta sugiere lo contrario. Esta medida contribuye a definir el género musical que se está abordando. Es importante tener en cuenta que una entropía más alta o más baja no implica un mejor o peor rendimiento del modelo, sino que define las características tonales del género con

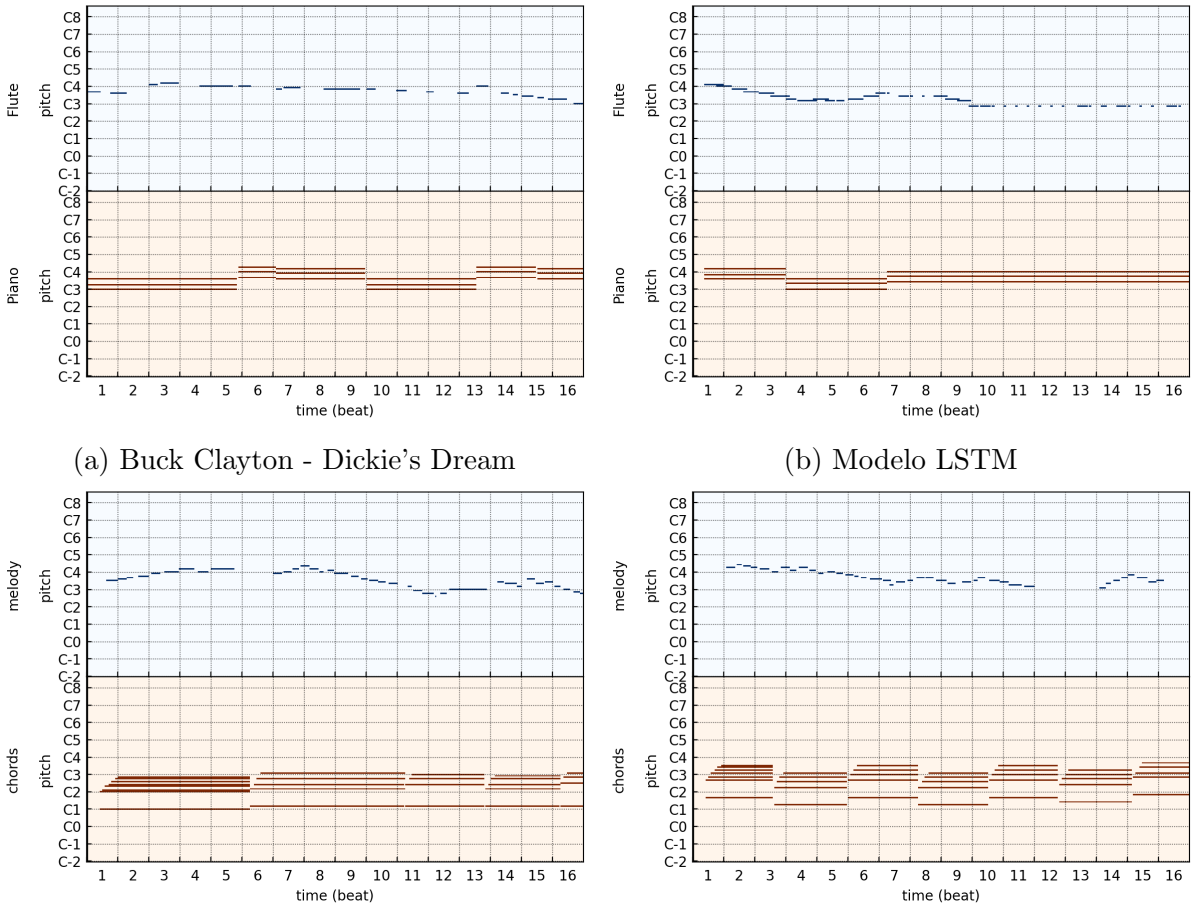
	Modelo LSTM	**Jazz Transformer (Preentrenado)**	**Jazz Transformer (Reentrenado)**	**Real**
$\mathcal{H}_1$	1.98	2.72	2.72	2.55
$\mathcal{H}_4$	2.46	3.09	3.07	3.12
$\mathcal{SC}_1$	0.91	0.89	0.89	0.89
$\mathcal{SC}_4$	0.87	0.85	0.85	0.82
$\mathcal{GS}$	0.70	0.84	0.84	0.87
$\mathcal{SI}_3^8$	0.45	0.28	0.27	0.30
$\mathcal{SI}_8^{15}$	0.42	0.17	0.19	0.30
$\mathcal{SI}_{15}$	0.35	0.08	0.11	0.28

Tabla 5.1: Métricas de evaluación para las piezas generadas por el modelo LSTM, por Jazz-Transformer (preentrenado y reentrenado) y las contenidas en el Weimar Jazz Database. $\mathcal{H}_1$ y $\mathcal{H}_4$ son las entropías del histograma de clases para segmentos de 1 y 4 compases, $\mathcal{SC}_1$ y $\mathcal{SC}_4$ las consistencias de escala para uno y cuatro compases y $\mathcal{SI}_3^8$, $\mathcal{SI}_8^{15}$ y $\mathcal{SI}_{15}$ los indicadores de estructura a corto, media y largo plazo.

el que se está trabajando. Por ejemplo, un género musical más simple, como el Pop, se caracterizará por melodías más simples y con menos variedad de notas (entropía más baja). Todo lo contrario ocurre con el Jazz, que destaca por sus melodías y escalas complejas, es decir, una mayor variedad de notas (mayor entropía). El objetivo de los modelos será aproximarse lo máximo posible al valor del conjunto de datos original. Esto servirá como medida de cuánto se acerca nuestra composición al género original. Lo contrario ocurre con la consistencia de la escala, la cual nos indica el grado de pertenencia de nuestras notas a las de una escala musical. En teoría, cuanto mayor sea este valor, más agradable será al oído dicha pieza. Aún así, la música está marcada por las imperfecciones, por lo que lo ideal es la búsqueda de un equilibrio entre entropía y consistencia. Que haya una gran riqueza tonal, pero a su vez, mantener las diferentes notas dentro de una tonalidad.

La Tabla 5.1 exhibe los resultados de las métricas objetivas, las cuales han sido calculadas para uno y cuatro compases en el caso de las métricas de armonía, siguiendo los pasos de [34]. Para ello, se han dividido las piezas generadas en segmentos de 1 y 4 compases y se ha realizado el promedio de cada métrica. Como era de esperar, el Jazz Transformer supera significativamente al modelo LSTM en comparación con el conjunto de datos original. Se observan entropías más bajas y consistencias de escalas más altas en el modelo LSTM, lo que indica melodías más simples y con menos variedad tonal en comparación con el Jazz Transformer.

Al analizar las características armónicas de cada composición de forma individual, se observan notables diferencias entre los dos modelos. El LSTM tiende a generar melodías de una complejidad armónica más reducida, mostrando una tendencia a la repetición y simplicidad, y alternando secciones ascendentes y descendentes en una escala, con repeticiones continuas del mismo tono (figura 5.1b). Esto se aplica también a los acordes, que tienden a mantenerse estáticos. En contraste, el Jazz Transformer ofrece melodías mucho más ricas y variadas. No se limita a seguir una pauta ascendente o descendente en una



(a) Buck Clayton - Dickie's Dream

(b) Modelo LSTM

(c) Jazz Transformer (Reentrenado)

(d) Jazz Transformer (Preentrenado

Figura 5.1: Melodías generadas por los distintos modelos junto al Piano Roll de una de las melodías de Weimar Jazz Database.

escala, sino que propone frases más creativas y complejas. Además, los acordes generados por el Jazz Transformer son por definición más elaborados, lo que resulta en una progresión de acordes notablemente más interesante (figura 5.1).

En términos generales, las composiciones del modelo LSTM, si bien no llegan a sonar mal (excepto por algunas disonancias puntuales en algunas secciones), muestran una cierta carencia de intencionalidad en su música. Las melodías parecen tener menos originalidad en comparación con el Jazz Transformer. Esto concuerda con las métricas objetivas de armonía, las cuales arrojaron entropías más bajas para el LSTM que para el Jazz Transformer, tal y como se detalló en la sección anterior.

5.2. Métricas rítmicas

Como métrica de ritmo hemos empleado el Groove Pattern Similarity que se define en la sección 2.7.1, y que indica la similitud en los patrones de ritmo entre dos compases

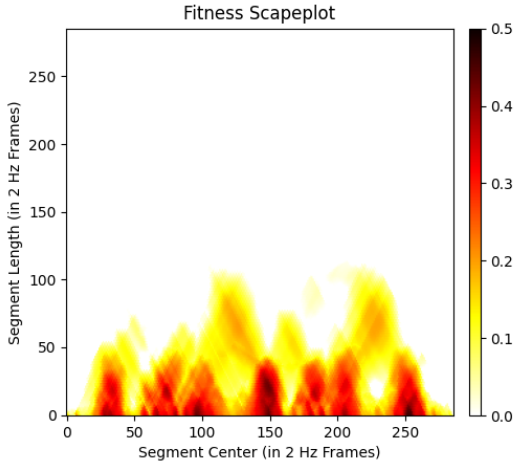
sucesivos. En la tabla 5.1, se puede observar que Jazz Transformer obtiene resultados muy similares a los del Weimar Jazz Database, a diferencia del modelo LSTM, lo que sugiere una falta de consistencia en las estructuras rítmicas del LSTM. Estas diferencias entre ambos modelos pueden atribuirse a las diferencias en la complejidad del lenguaje que cada uno emplea. Jazz Transformer define la posición de cada evento a partir del compás en el que se encuentra y de su ubicación dentro de dicho compás, mientras que el modelo LSTM solo considera la posición de la nota previa. A esto se suma el nivel de cuantización de cada modelo, ya que en Jazz Transformer se divide cada compás en 64 partes, mientras que en el LSTM se divide en 32. Todo esto puede ser la causa de la baja puntuación del modelo LSTM para el $\mathcal{GS}$.

Si atendemos directamente a las melodías generadas por cada modelo, podemos apreciar claramente estos resultados. Por un lado, Jazz Transformer es capaz de utilizar distintos tipos de duraciones e intercalar silencios. Además, parece comprender el concepto de compás y puede repetir estructuras compás a compás. Esto hace que los cambios de acorde sean mucho más dinámicos y las melodías tengan un mayor sentido del ritmo. Por otro lado, el modelo LSTM se limita a generar melodías monótonas con muy pocas pausas, favoreciendo las semicorcheas como duración más utilizada. Como era de esperar, no emplea los compases como referencia, lo que resulta en la producción de notas de manera errática y sin ninguna intención rítmica.

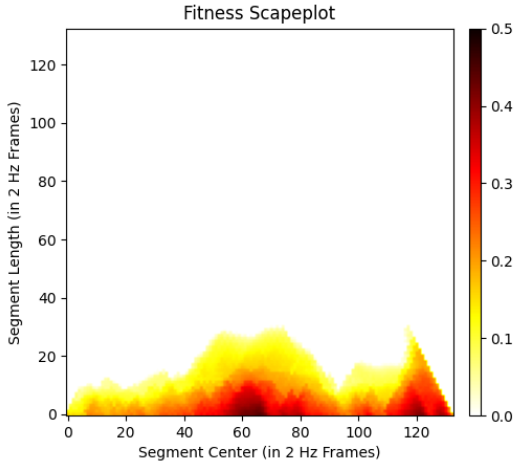
5.3. Métricas de estructura

En cuanto a los indicadores de estructura, se aprecia un gran grado de repetición en el modelo LSTM, mientras que esto no es evidente en el caso del Jazz Transformer. En este último, se percibe una marcada ausencia de estructura a medio y largo plazo. Sin embargo, los valores elevados del LSTM podrían sugerir un exceso de repetición en las melodías, lo cual concuerda con los resultados del resto de métricas. Por otro lado, el Jazz Transformer es capaz de replicar de manera destacada la estructura a corto plazo, pero enfrenta dificultades con fragmentos más extensos.

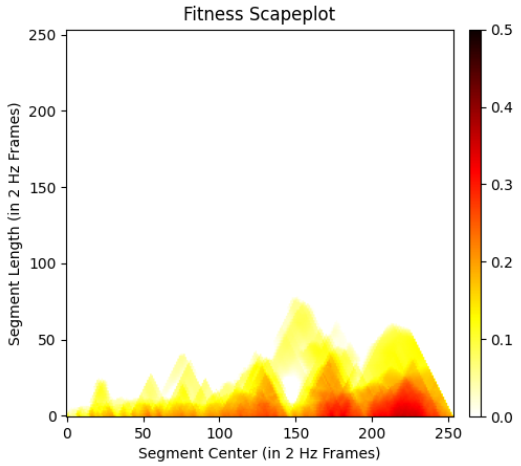
Para representar de manera visual estas características de las composiciones, elegiremos las piezas generadas con los mayores $\mathcal{SI}$ y las compararemos con una pieza análoga del Weimar Jazz Database (figura 5.2) mediante sus FSP. Como se puede observar, en la melodía "real" se pueden identificar distintas secciones que se repiten de manera clara, separadas entre sí por fragmentos nuevos o variaciones. En este caso, los fragmentos repetidos alcanzan una duración que oscila entre 30 y 60 segundos. Todo lo contrario ocurre con el Jazz Transformer, cuyo FSP es mucho más difuso y no permite distinguir entre las diferentes partes. A pesar de que estas melodías presentan una alta similitud en los segmentos más cortos, esta es es prácticamente nula en los segmentos de duración media y larga. Por último, para el modelo LSTM observamos múltiples segmentos con un alto índice de similitud que se superponen entre sí. Esto concuerda con los resultados de los índices en la tabla 5.1 y puede ser producto de un exceso de repetitividad en las melodías.



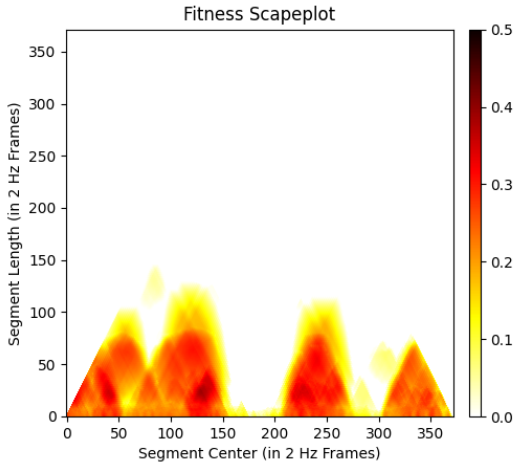
(a) modelo LSTM



(b) Jazz Transformer (Preentrenado)



(c) Jazz Transformer (Reentrenado)



(d) David Liebman - There Will Never Be Another You

Figura 5.2: Fitness scape plot de las melodías con mayores índices de similitud. Se encuentran representados todos los segmentos de una composición en función de la ubicación de su centro en la pieza (eje horizontal) y su duración en frames de 2 Hz (Eje vertical, 2 frames = 1 segundo), asignando el grado de similituda cada uno con una escala de color

Capítulo 6

Conclusiones

En este trabajo, hemos abordado las diferentes herramientas y procesos para la composición automática de música mediante algoritmos de DL, así como los distintos métodos de evaluación de dichas composiciones. Para ilustrarlo, hemos expuesto dos modelos distintos para la generación de música, uno basado en LSTM y otro en Transformer-XL. Cada uno, además, con una representación de los datos diferente. En ambos casos, los modelos han sido capaces de generar nueva música, pero con ciertas limitaciones.

Por un lado, el modelo basado en LSTM ha demostrado su capacidad para generar melodías sobre una base armónica, aunque de manera bastante simple. Estas melodías se limitaban a progresiones monótonas de notas y algo irregulares en cuanto al ritmo. Además, se observa una tendencia del modelo a colapsar en una cierta nota o acorde durante distintos tramos de las piezas musicales. Sin embargo, el principal problema en estas melodías es la falta de intención musical. Las limitaciones impuestas por la forma en que se representan los datos de entrada para el modelo hacen que éste se vea restringido a generar secuencias de notas sin prestar atención a los motivos estructurales. Hay una clara ausencia de frases en las distintas piezas, que a su vez estén compuestas por motivos más pequeños e impriman dinamismo a la pieza. Por el contrario, el modelo se limita a crear una única frase monótona y plana, sin ninguna finalidad clara.

Por otra parte, el Jazz Transformer ha demostrado ser capaz de generar música notablemente más elaborada. En este caso, las melodías producidas eran mucho más originales y ricas en armonía. En términos generales, las melodías del Jazz Transformer se asemejan mucho más a las del conjunto de datos original que las generadas por el LSTM. Sin embargo, la mayor diferencia radica en la estructura de estas melodías. En el Jazz Transformer, la forma en que se define la posición de las notas y la presencia de eventos de estructura facilita una composición mucho más natural. Se logra distinguir distintas frases dentro de cada pieza, separadas entre sí por pequeñas pausas o silencios. De esta manera, las melodías del Jazz Transformer parecen poseer la intencionalidad que le faltaba al modelo LSTM. A pesar de ello, aún son evidentes las deficiencias en comparación con composiciones reales. Aunque se diferencien distintas frases dentro de la pieza, esta carece de la coherencia de una composición real. Seguimos teniendo frases inconexas entre ellas que se suceden durante toda la pieza y que empeoran la escucha.

En cuanto a las métricas objetivas que hemos aplicado, en general han demostrado ser efectivas en la evaluación de las cualidades que nos propusimos analizar. En primer lugar,

la entropía nos ha proporcionado una medida clara del nivel de complejidad armónica presente en las composiciones. Esto nos permitió identificar la simplicidad del modelo LSTM en comparación con las demás composiciones. Sin embargo, la consistencia de la escala no presentó grandes variaciones en ninguno de los conjuntos, por lo que no sirvió especialmente para cuantificar las cualidades armónicas. Esto puede deberse a la ausencia de disonancias significativas entre las notas. En cuanto al ritmo, el Groove Pattern Similarity fue útil para detectar la falta de consistencia en los patrones rítmicos del modelo LSTM. Por último, los indicadores de estructura nos permitieron identificar problemas en la organización a corto, medio y largo plazo en ambos modelos. Estos indicadores fueron especialmente valiosos para visualizar la estructura de las composiciones y analizar dichas características.

En resumen, las métricas empleadas proporcionan una guía para analizar las diversas características de la música, pero requieren de una interpretación humana. Y aunque son útiles, el propósito final de estos modelos es generar música que suene lo más auténtica y original posible para el oído humano. Por esta razón, sigue siendo necesario llevar a cabo un análisis subjetivo para evaluar adecuadamente este tipo de modelos y sus piezas generadas.

En general, ambos modelos han logrado generar música nueva de manera satisfactoria, aunque con las limitaciones mencionadas previamente. La música resultante es agradable al oído en términos generales, pero carece de originalidad y realismo. Es importante tener en cuenta que lo que imprime naturalidad y expresión a una pieza son las pequeñas imperfecciones en la interpretación del ser humano. Las pequeñas variaciones en la intensidad de las notas, el tempo, etc; casi imperceptibles, pero que humanizan las composiciones. Son características muy complicadas de definir con un lenguaje matemático y presentan el verdadero reto de la generación automática de música. Aún así, estos modelos son ejemplo de un avance en la generación por computadora, y aunque aún no son capaces de crear melodías completamente realistas por sí solos, sí pueden integrarse en el proceso creativo como apoyo al artista.

Bibliografía

[1] Adam Alpern. Techniques for algorithmic composition of music. 1995.

[2] Donald Jay Grout and Claude V. Palisca. *A History of Western Music*. Norton & Company, Incorporated, W. W., 1996.

[3] Lejaren Arthur Hiller and Leonard M. Isaacson. *Experimental Music; Composition with an Electronic Computer*. Greenwood Publishing Group Inc., USA, 1979. ISBN 0313221588.

[4] Iannis Xenakis. *Formalized music : thought and mathematics in composition*. Indiana University Press, 1971.

[5] David Cope. Experiments in musical intelligence (emi): Non-linear linguistic-based composition. *Interface*, 18(1-2):117–139, 1989. doi: 10.1080/09298218908570541. URL https://doi.org/10.1080/09298218908570541.

[6] Michael Mozer. Neural network music composition by prediction: Exploring the benefits of psychoacoustic constraints and multi-scale processing. *Connection Science - CONNECTION*, 6:247–280, 01 1994. doi: 10.1080/09540099408915726.

[7] Carlos Hernandez-Olivan and Jose R. Beltran. Music composition with deep learning: A review, 2021.

[8] Shulei Ji, Jing Luo, and Xinyu Yang. A comprehensive survey on deep music generation: Multi-level representations, algorithms, evaluations, and future directions. *ArXiv*, abs/2011.06801, 2020. URL https://api.semanticscholar.org/CorpusID:226956064.

[9] Davide Baccherini, Donatella Merlini, and Renzo Sprugnoli. Tablatures for stringed instruments and generating functions. pages 40–52, 06 2007. ISBN 978-3-540-72913-6. doi: 10.1007/978-3-540-72914-3_6.

[10] Michael Scott Cuthbert and Christopher Ariza. Music21: A toolkit for computer-aided musicology and symbolic music data. In J. Stephen Downie and Remco C. Veltkamp, editors, *ISMIR*, pages 637–642. International Society for Music Information Retrieval, 2010. ISBN 978-90-393-53813. URL http://dblp.uni-trier.de/db/conf/ismir/ismir2010.html#CuthbertA10.

[11] Ole Martin Bjørndalen and Raphaël Doursenaud. Mido. github.com/mido/mido, 2013.

[12] Colin Raffel and Daniel P. W. Ellis. Intuitive analysis, creation and manipulation of midi data with pretty_midi. In *ISMIR*. International Society for Music Information Retrieval, 2014. URL https://colinraffel.com/publications/ismir2014intuitive.pdf.

[13] Hao-Wen Dong, Ke Chen, Julian McAuley, and Taylor Berg-Kirkpatrick. Muspy: A toolkit for symbolic music generation, 2020. URL https://salu133445.github.io/muspy/.

[14] Hao-Wen Dong, Wen-Yi Hsiao, and Yi-Hsuan Yang. Pypianoroll: Open source python package for handling multitrack pianoroll. In *ISMIR*. International Society for Music Information Retrieval, 2018. URL https://github.com/salu133445/pypianoroll.

[15] Diederik Kingma and Max Welling. Auto-encoding variational bayes. *ICLR*, 12 2013.

[16] Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial networks, 2014.

[17] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9:1735–80, 12 1997. doi: 10.1162/neco.1997.9.8.1735.

[18] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. 06 2017.

[19] C.W. Walton. *Basic Forms in Music*. Alfred Publishing Company, 2005. ISBN 9781457408229. URL https://books.google.es/books?id=QYttVHpkoFAC.

[20] Olof Mogren. C-rnn-gan: Continuous recurrent neural networks with adversarial training. 11 2016.

[21] Elliot Waite et al. Generating long-term structure in songs and stories. 2016. URL https://magenta.tensorflow.org/2016/07/15/lookback-rnn-attention-rnn.

[22] Gaëtan Hadjeres and Francois Pachet. Deepbach: a steerable model for bach chorales generation. 12 2016.

[23] Adam Roberts, Jesse Engel, Colin Raffel, Curtis Hawthorne, and Douglas Eck. A hierarchical latent vector model for learning long-term structure in music. In *International Conference on Machine Learning (ICML)*, 2018. URL http://proceedings.mlr.press/v80/roberts18a.html.

[24] Colin Raffel. Lakh MIDI Dataset (LMD). https://colinraffel.com/projects/lmd/, 2016.

[25] Cheng-Zhi Anna Huang, Ashish Vaswani, Jakob Uszkoreit, Noam Shazeer, Curtis Hawthorne, Andrew M Dai, Matthew D Hoffman, , and Douglas Eck. Music transformer: Generating music with long-term structure. In *International Conference on Machine Learning (ICML)*, 2018. URL http://proceedings.mlr.press/v80/roberts18a.html.

[26] Christine Payne. Musenet. 2019. URL https://openai.com/research/musenet.

[27] Junyan Jiang, Gus G. Xia, Dave B. Carlton, Chris N. Anderson, and Ryan H. Miyakawa. Transformer vae: A hierarchical model for structure-aware and interpretable music representation learning. In *ICASSP 2020 - 2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 516–520, 2020. doi: 10.1109/ICASSP40776.2020.9054554.

[28] Shunit Haviv Hakimi, Nadav Bhonker, and Ran El-Yaniv. Bebopnet: Deep neural models for personalized jazz improvisations. In *Proceedings of the 21st International Society for Music Information Retrieval Conference (ISMIR)*, 2020.

[29] Hao-Wen Dong, Wen-Yi Hsiao, Li-Chia Yang, and Yi-Hsuan Yang. Musegan : Demonstration of a convolutional gan based model for generating multi-track piano-rolls. 2017. URL https://api.semanticscholar.org/CorpusID:3731499.

[30] Alec Radford, Luke Metz, and Soumith Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. In Yoshua Bengio and Yann LeCun, editors, *4th International Conference on Learning Representations, ICLR 2016, San Juan, Puerto Rico, May 2-4, 2016, Conference Track Proceedings*, 2016. URL http://arxiv.org/abs/1511.06434.

[31] Curtis Hawthorne, Andriy Stasyuk, Adam Roberts, Ian Simon, Cheng-Zhi Anna Huang, Sander Dieleman, Erich Elsen, Jesse Engel, and Douglas Eck. Enabling factorized piano music modeling and generation with the MAESTRO dataset. In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=r1lYRjC9F7.

[32] Martin Pfleiderer, Klaus Frieler, Jakob Abeßer, Wolf-Georg Zaddach, and Benjamin Burkhart, editors. *Inside the Jazzomat - New Perspectives for Jazz Research*. Schott Campus, 2017.

[33] Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. Bleu: a method for automatic evaluation of machine translation. In Pierre Isabelle, Eugene Charniak, and Dekang Lin, editors, *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pages 311–318, Philadelphia, Pennsylvania, USA, July 2002. Association for Computational Linguistics. doi: 10.3115/1073083.1073135. URL https://aclanthology.org/P02-1040.

[34] Shih-Lun Wu and yi-hsuan Yang. The jazz transformer on the front line: Exploring the shortcomings of ai-composed music through quantitative measures. 08 2020.

[35] Meinard Müller, Peter Grosche, and Nanzhu Jiang. A segment-based fitness measure for capturing repetitive structures of music recordings. pages 615–620, 01 2011.

[36] Jonathan Foote. Visualizing music and audio using self-similarity. In *Proceedings of the Seventh ACM International Conference on Multimedia (Part 1)*, MULTIMEDIA '99, page 77–80, New York, NY, USA, 1999. Association for Computing Machinery. ISBN 1581131518. doi: 10.1145/319463.319472. URL https://doi.org/10.1145/319463.319472.

[37] Berit Janssen, Tom Collins, and Iris Ren. Algorithmic ability to predict the musical future: Datasets and evaluation. *Proc. International Society for Music Information Retrieval Conference (ISMIR)*, page 208–215, 2019.

[38] Sepp Hochreiter. The vanishing gradient problem during learning recurrent neural nets and problem solutions. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, 6:107–116, 04 1998. doi: 10.1142/S0218488598000094.

[39] Guillaume Chevalier. The lstm cell. Wikipedia, Long short-term memory article, 2018. URL: https://en.wikipedia.org/wiki/Long_short-term_memory#/media/File:LSTM_Cell.svg.

[40] Sigurour Skúli. How to generate music using a lstm neural network in keras, 2017. URL: https://towardsdatascience.com/how-to-generate-music-using-a-lstm-neural-network-in-keras-68786834d4c5.

[41] Zihang Dai, Zhilin Yang, Yiming Yang, Jaime Carbonell, Quoc Le, and Ruslan Salakhutdinov. Transformer-XL: Attentive language models beyond a fixed-length context. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 2978–2988, Florence, Italy, July 2019. Association for Computational Linguistics. doi: 10.18653/v1/P19-1285. URL https://aclanthology.org/P19-1285.

[42] Nitish Shirish Keskar, Bryan McCann, Lav R. Varshney, Caiming Xiong, and Richard Socher. Ctrl: A conditional transformer language model for controllable generation, 2019.

[43] Alec Radford and Karthik Narasimhan. Improving language understanding by generative pre-training. 2018. URL https://api.semanticscholar.org/CorpusID:49313245.

[44] Jordan J. Bird. Keras lstm music generator. https://github.com/jordan-bird/Keras-LSTM-Music-Generator, 2021.

[45] Moseli Mots'oehli, Anna Sergeevna Bosman, and Johan Pieter De Villiers. Comparison of adversarial and non-adversarial lstm music generative models. In Kohei Arai, editor, *Intelligent Computing*, pages 428–458, Cham, 2023. Springer Nature Switzerland. ISBN 978-3-031-37717-4.

[46] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15:1929–1958, 06 2014.

[47] Yu-Siang Huang and Yi-Hsuan Yang. Pop music transformer: Beat-based modeling and generation of expressive pop piano compositions. MM '20, page 1180–1188, New York, NY, USA, 2020. Association for Computing Machinery. ISBN 9781450379885. doi: 10.1145/3394171.3413671. URL https://doi.org/10.1145/3394171.3413671.

[48] K. Frieler, M. Pfleiderer, W.-G. Zaddach, and J. Abeßer. Midlevel analysis of monophonic jazz solos: A new approach to the study of improvisation, 2016.

[49] Emilio Parisotto, H. Francis Song, Jack W. Rae, Razvan Pascanu, Caglar Gulcehre, Siddhant M. Jayakumar, Max Jaderberg, Raphael Lopez Kaufman, Aidan Clark, Seb Noury, Matthew M. Botvinick, Nicolas Heess, and Raia Hadsell. Stabilizing transformers for reinforcement learning, 2019.