UNIVERSIDAD DE LA HABANA

FACULTAD DE MATEMÁTICA Y COMPUTACIÓN

TRANSCRIPCIÓN AUTOMÁTICA DE MÚSICA

AUTORES

Leonardo Javier Ramirez Calatayud
Naomi Lahera Champagne
Eric Lopez Tornas
Loitzel Ernesto Morales Santiesteban

La Habana, Cuba.
Julio, 2024

Contenido

Índice de figuras III

Índice de tablas IV

1. Abstract **2**

2. Resumen **3**

3. Introducción **4**
3.1. Motivación 4
3.2. Problemática 4

4. Objetivos **6**

5. Estado del arte **7**
5.1. Análisis 7
5.2. Resultados en el Estado del Arte: 10
5.2.1. Desempeño en Datasets 10
5.2.2. Técnicas de Mejora 10
5.2.3. Principales Hallazgos 11
5.2.4. Selección de elementos para la investigación 11

6. Análisis de Datos **12**
6.1. Notas 12

7. Creación del Dataset **16**
7.1. Dataset Base 16
7.2. Transformación del dataset 17
7.2.1. Creación del .midi 17

7.3. División de los datos 17
7.3.1. Extracción de características de los archivos de audio 17
7.3.2. Extracción de componentes principales 18
7.3.3. Clustering 19
7.3.4. Selección de hiperparámetros 19
7.3.5. Resultados 21
7.4. Estructración del dataset 22
7.5. Extracción de características de los MIDI 23
7.5.1. Curva de tempo 24
7.5.2. Extracción de notas 24
7.5.3. Procesamiento de mensajes MIDI 24
7.5.4. Conversión de notas 25

8. Herramienta de Transcripción **27**
8.1. Adaptaciones del Código Base 27
8.1.1. Transcripción de Audio a MIDI 27
8.1.2. Transcripción de MIDI a partitura 28

9. Entrenamiento **29**
9.1. Modelo 29
9.2. Flujo de trabajo 30
9.2.1. Funciones de pérdida 31

10. Evaluación **33**
10.1. Evaluación de la Estimación Multi-Tono (MPE) 33
10.2. Evaluación de la Transcripción de Notas 34

11. Resultados **36**
11.1. Repercusiones Éticas 38

12. Recomendaciones **39**
12.1. Evaluación con Datasets Multiinstrumento 39
12.2. Evaluar y Mejorar la Robustez a Ruido 39
12.3. Limpieza de Partituras Resultantes 40
12.4. Colaboración con Expertos en Música 40

13. Conclusiones **41**
13.1. Importancia del Dataset: 41
13.2. Adaptación de Herramientas: 41
13.3. Análisis de Datos: 42

14. **43**

Índice de figuras

6.1. Histogramas de las notas de canciones del dataset 13
6.2. Histogramas de las notas de canciones del dataset 13
6.3. Histogramas de las notas de canciones del dataset 14
6.4. Datos de todas las notas del dataset 14

7.1. Muestreo de resultadados arrojados con diversos valores del hiperparámetro k (número de clusters) y p (cantidad de componentes principales) usando las métricas: Indice de Davies-Bouldin e Indice de Calinski-Harabasz 21
7.2. Histograma de frecuencias de las notas en una curva de densidad (entrenamiento). 22
7.3. Diagrama de caja de bigotes del número de notas(entrenamiento). . . 22
7.4. Histograma de frecuencias de las notas en una curva de densidad (validación). 22
7.5. Diagrama de caja de bigotes del número de notas (validación). 22
7.6. Histograma de frecuencias de las notas en una curva de densidad (test). 23
7.7. Diagramma de caja de bigotes del número de notas (test). 23

8.1. Diagrama de flujo del proceso de transcripción de audio a partitura . 28

9.1. Paso de ventana empleado 30
9.2. N: número de muestras en cada fragmento de procesamiento, M: longitud del padding, F: número de bins de frecuencia, P: número de tonos 31

Índice de tablas

7.1. Estadísticas de prueba y valores p para comparaciones entre conjuntos de datos 23
7.2. Conversión de notas. Etiqueta vs Referencia 26

11.1. Resultados en distintos modelos 37

Capítulo 1

Abstract

Automatic Music Transcription (AMT) is a task that has captured the attention of researchers in the fields of artificial intelligence and musicology for decades. The main objective of AMT is to convert audio music signals into readable musical notations, such as sheet music. This process involves identifying notes, rhythms, instruments, and other musical characteristics from audio recordings. In recent years, advances in machine learning, especially in deep learning techniques, have significantly propelled progress in AMT.

Capítulo 2

Resumen

Este estudio presenta un enfoque integral para la transcripción automática de música (AMT). El objetivo principal es transformar señales de audio de guitarra en notaciones musicales legibles, utilizando técnicas de machine learning. A lo largo de este trabajo, se exploran diversos modelos y métodos, con un énfasis particular en el modelo Hierachical Frequency-Time Transformer (que demostró ser altamente eficaz en la transcripción de partituras de piano).

El documento detalla el proceso completo, desde la creación y transformación del dataset hasta la optimización de hiperparámetros y el análisis de datos. Se implementaron variados métodos de análisis, incluyendo histogramas, curvas de densidad y diagramas de caja de bigotes, para asegurar la precisión y validez de los datos utilizados. Además, se emplearon técnicas de evaluación rigurosas, como la estimación multi-tono y la transcripción de notas, para medir el rendimiento del modelo desarrollado.

Los resultados obtenidos muestran una mejora significativa en la precisión y eficiencia de la transcripción automática de música, contribuyendo así al avance del estado del arte en este campo. Este trabajo no solo ofrece una herramienta valiosa para músicos y profesionales del audio, sino que también establece una base sólida para futuras investigaciones y desarrollos en la transcripción automática de música.

Capítulo 3

Introducción

3.1. Motivación

El desarrollo de un modelo de transcripción automática de música de guitarra no solo facilita el aprendizaje y la enseñanza de este instrumento mediante la generación automática de partituras, sino que también apoya a compositores y arreglistas en la creación y distribución de notaciones musicales. La creación de partituras es un proceso complejo incluso para profesionales; la automatización de este proceso presenta una oportunidad única para democratizar este conocimiento y hacerlo accesible a una audiencia más amplia. Además, esta tecnología puede contribuir significativamente a la musicología, permitiendo análisis más profundos y precisos de las piezas musicales, así como la preservación y estudio de obras musicales de forma más eficiente y exacta.

3.2. Problemática

La transcripción automática de música de guitarra presenta desafíos únicos y complejos. La guitarra es un instrumento polifónico capaz de producir múltiples notas simultáneamente, lo que da lugar a acordes y melodías superpuestas. Además, la guitarra posee una variedad de técnicas específicas de ejecución que deben ser correctamente interpretadas y representadas en la transcripción.

La variabilidad en la interpretación de diferentes guitarristas añade una capa adicional de complejidad. Cada músico puede influir significativamente en la dinámica, el timbre y el tempo de la música, lo que dificulta la tarea de desarrollar un modelo que pueda generalizar bien a través de estas diferencias. Estas variaciones pueden resultar en distintas versiones de una misma pieza musical, lo cual plantea un reto considerable para los sistemas automáticos que buscan generar una transcripción precisa y coherente.

Además, la calidad de las grabaciones de audio y los posibles ruidos de fondo pueden afectar la precisión de la transcripción, lo que requiere de técnicas avanzadas de procesamiento de señal para filtrar y extraer las notas y acordes correctos.

Capítulo 4

Objetivos

Los objetivos a analizar en el siguiente trabajo se pueden resumir a los siguientes.

Desarrollar un modelo eficiente para la transcripción automática de música: Implementar técnicas de Machine Learning para mejorar la precisión y eficiencia en la transcripción de señales de audio musicales a notaciones musicales legibles.

Evaluar el desempeño de diferentes modelos en datasets específicos: Analizar y comparar la efectividad de diversos modelos de transcripción automática en conjuntos de datos como Maestro, MAPS, GuitarSet, MusicNet, y URMP.

Optimizar la extracción de características y la selección de hiperparámetros: Refinar técnicas de procesamiento de audio y de extracción de componentes principales para mejorar la calidad de las transcripciones.

Implementar y probar un flujo de trabajo completo para la transcripción de audio a partitura: Desarrollar un sistema que abarque desde la transcripción de audio a MIDI hasta la generación de partituras, incluyendo la evaluación de la precisión en la estimación multi-tono y la transcripción de notas.

Contribuir al estado del arte en la transcripción automática de música: Publicar los hallazgos y avances obtenidos, para fomentar el desarrollo continuo en este campo de investigación y su aplicación práctica en diversas áreas musicales.

Capítulo 5

Estado del arte

5.1. Análisis

A continuación se muestra un resumen de los artículos que forman parte del estado del arte actual en la Transcripción Automática de Música.

1. **Automatic Piano Sheet Music Transcription with Machine Learning (2021)**

   Modelo: BiLSTM (Bidirectional Long Short-Term Memory)
   Resultados: Utilizado para la transcripción de partituras de piano. Se encontró que el **BiLSTM** era particularmente eficaz para esta tarea específica, logrando una precisión del 83 % en la identificación de notas y ritmos en grabaciones de piano en 90 epoch.

2. **AUTOMATIC PIANO TRANSCRIPTION WITH HIERARCHICAL FREQUENCY-TIME TRANSFORMER (2023)**

   Modelo: hTF-Transformer
   Datasets: Maestro, MAPS
   Resultados: El modelo hTF-Transformer, una variante del Transformer enfocada en la transformación jerárquica de frecuencia-tiempo, logró un desempeño notable, con aproximadamente un 90 % de precisión en el dataset Maestro y un 70 % en el dataset MAPS. Este modelo destaca por su capacidad para manejar la complejidad temporal y frecuencial de las señales de audio de música.

3. **Multi-Task Multitrack Music Transcription (2022)**

   Modelo: Transformer
   Datasets: Slakh1000, Maestro, GuitarSet, MusicNet, URMP
   R**esultados:** El modelo MT3, basado en la arquitectura Transformer, se utilizó para la transcripción multitarea de música multipista. Este modelo mostró un desempeño robusto en múltiples datasets, alcanzando aproximadamente un 90 % de precisión en Maestro, GuitarSet un 58 % URMP un 86 % y un 74 % en Slakh1000. La capacidad multitarea del modelo le permite manejar la transcripción de diversos instrumentos y estilos musicales.

4. **Multitrack Music Transcription with a Time-Frequency Perceiver (2023)**

   Modelo: Perceiver TF
   Datasets: Slakh1000, Maestro, GuitarSet
   Resultados: El Perceiver TF es un modelo innovador que utiliza una percepción conjunta en el dominio tiempo-frecuencia. Este modelo alcanzó un desempeño de aproximadamente un 90 % en los datasets Slakh1000 y Maestro, y en GuitarSet hasta un 83 %. Su enfoque novedoso permite una transcripción precisa y eficiente de música multipista.

5. **Jointist: Joint Learning for Multi-instrument Transcription and Its Applications (2022)**

   Modelo: Transformer, Deep Convolutional and Recurrent Neural Network
   Datasets: Slakh1000
   Resultados: El modelo Jointist combina técnicas de aprendizaje conjunto para la transcripción multi-instrumental y la separación de fuentes musicales. Aunque su precisión general en Slakh1000 es de aproximadamente un 85 %, destaca por su menor varianza entre diferentes instrumentos, lo que lo hace especialmente útil para aplicaciones prácticas donde la consistencia es crucial.

6. **Scaling Polyphonic Transcription with Mixtures of Monophonic Transcriptions (2022)**

   Modelo: Transformer
   Datasets: Slakh1000, Maestro, GuitarSet, MusicNet, URMP

Resultados: Este estudio explora la escalabilidad de la transcripción polifónica utilizando mezclas de transcripciones monofónicas. Los resultados muestran mejoras significativas en varios datasets, con un desempeño de aproximadamente un 80 % en Slakh1000, un 90 % en Maestro, un 69 % en GuitarSet, un 55 % en MusicNet y un 95 % en URMP. La técnica de mezcla monofónica demostró ser efectiva para mejorar la transcripción polifónica multi-instrumental.

7. **Transfer of Knowledge Among Instruments in Automatic Music Transcription (2023)**

   Modelo: Transformers
   Datasets: GuitarSet, MAPS
   Resultados: Este estudio explora cómo la transferencia de conocimiento sintetizado entre diferentes instrumentos puede mejorar el desempeño de los modelos de transcripción automática. Los resultados muestran mejoras de hasta un 20 % en algunas métricas al transferir *conocimiento* de un instrumento a otro. Aunque algunas métricas solo mejoran marginalmente, la consistencia entre los diferentes datasets es notable, lo que sugiere que la transferencia de conocimiento puede ser una estrategia efectiva para mejorar el rendimiento de los modelos de AMT.

8. **Timbre-Trap: A Low-Resource Framework for Instrument-Agnostic Music Transcription (2023)**

   Modelo: Transformers
   Datasets: GuitarSet, Bach10
   Resultados: El modelo Timbre-Trap se enfoca en la transcripción de música sin depender de grandes cantidades de datos etiquetados. Aunque este modelo no alcanza el estado del arte en términos de desempeño, su enfoque de bajo recurso es prometedor para aplicaciones donde los datos etiquetados son limitados. Este modelo demuestra la posibilidad de realizar transcripciones precisas con una cantidad reducida de datos de entrenamiento.

5.2. Resultados en el Estado del Arte:

5.2.1. Desempeño en Datasets

Los modelos presentados han sido evaluados en una variedad de datasets públicos, cada uno con características únicas que desafían a los sistemas de transcripción de diferentes maneras.

- **Maestro:**
  Este dataset es uno de los más utilizados debido a su alta calidad de grabación y la diversidad de composiciones pianísticas. Los modelos que han sido evaluados en Maestro, como el hTF-Transformer y MT3, han mostrado un desempeño de alrededor del 90 %.

- **Slakh1000:**
  Este dataset es conocido por su complejidad debido a las múltiples pistas e instrumentos. Los modelos como MT3 y Jointist han logrado precisiones entre el 74 % y el 85 %.

- **GuitarSet y MusicNet:** Estos datasets presentan desafíos específicos relacionados con la transcripción de instrumentos de cuerda y música clásica. Los resultados varían, con MusicNet siendo particularmente difícil, alcanzando solo un 50-55 % de precisión en algunos casos.

5.2.2. Técnicas de Mejora

Varios estudios han propuesto técnicas innovadoras para mejorar la precisión de la transcripción:

- **Transferencia de Conocimiento:** La transferencia de conocimiento entre instrumentos ha demostrado ser una estrategia eficaz. Este enfoque permite que los modelos aprendan características comunes entre diferentes instrumentos, mejorando la precisión general.

- **Mezcla Monofónica:** Utilizar transcripciones monofónicas para mejorar la transcripción polifónica ha sido otra técnica exitosa. Esta estrategia facilita la tarea al modelo, permitiendo que se enfoque en la combinación de pistas individuales en lugar de procesar la complejidad completa de una grabación polifónica.

- **Aprendizaje Conjunto:** El aprendizaje conjunto de transcripción y separación de fuentes ha demostrado ser beneficioso en modelos como Jointist, donde la combinación de múltiples tareas relacionadas mejora el rendimiento general. Conclusiones

5.2.3. Principales Hallazgos

- **Modelos Basados en Transformers:** Los modelos Transformers se han establecido como la arquitectura de elección para AMT, debido a su capacidad para capturar relaciones temporales y frecuenciales complejas.
- **Importancia del Dataset:** La calidad y la diversidad del dataset son cruciales para el desempeño del modelo. Los datasets como Maestro y Slakh1000 han demostrado ser especialmente útiles para evaluar y mejorar los modelos de AMT.
- **Transferencia de Conocimiento y Aprendizaje Conjunto:** Técnicas como la transferencia de conocimiento y el aprendizaje conjunto han mostrado ser efectivas para mejorar la precisión y la robustez de los modelos de transcripción.

5.2.4. Selección de elementos para la investigación

Para la investigación a realizar se decidió escoger el dataset de **GuitarSet** puesto que es uno de los que más dificultades afronta para obtener buenos resultados además de ser los datasets de piano los más atacados en la transcripción automática. Como modelo fue escogido, por los resultados presentados en el artículo, el **hTF-Transformer**, pero esta vez se pondrá a prueba con un dataset no utilizado en el mismo hasta el momento.

Capítulo 6

Análisis de Datos

6.1. Notas

Al analizar detenidamente los *.midi* de cada canción se llegó a la conclusión que lo más importante son las notas tocadas por cada una de ellas, por lo tanto se hace necesario realizar un análisis de la presencia de las notas en las canciones. Para ello serán utilizados diagramas de cajas y bigote aunado con histogramas de frecuencias con curvas de densidad calculadas mediante **KDE**.

En las *figuras 3.1, 3.2 y 3.3* representan ejemplos de canciones y las notas asociadas a las mismas y con ellas se puede observar el primer inconveniente que presenta el conjunto de datos pues las canciones son heterogéneas lo que indica que se debe tener un cuidado minucioso al separarlas en **train, validation y test**.

Luego en la figura 3.4 analiza la distribución de las notas de todo el conjunto de datos. En próximos análisis se debe asegurarnos que la distribución observada en los distintos splits(entrenamiento, validación y test) concuerde con la de dicha figura.

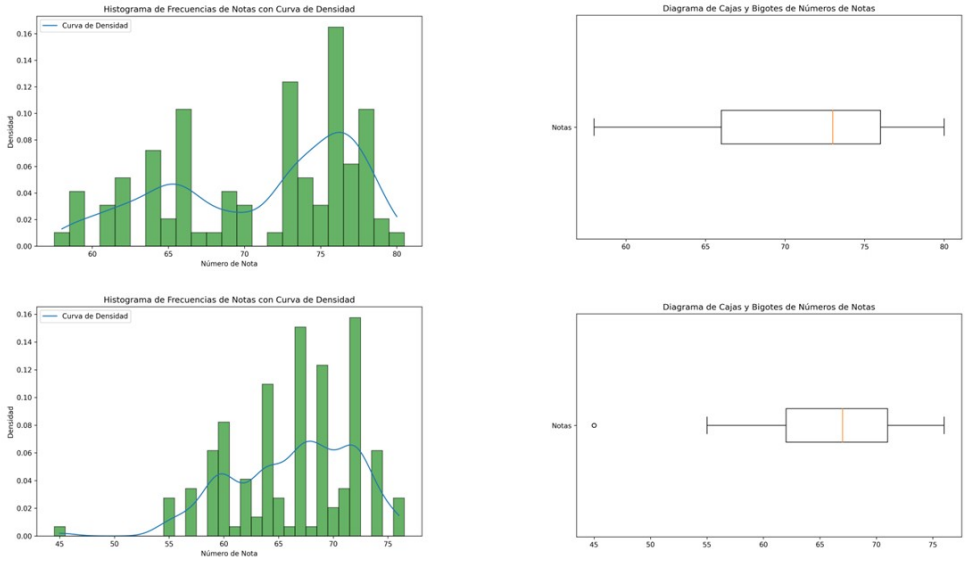

Figura 6.1: Histogramas de las notas de canciones del dataset

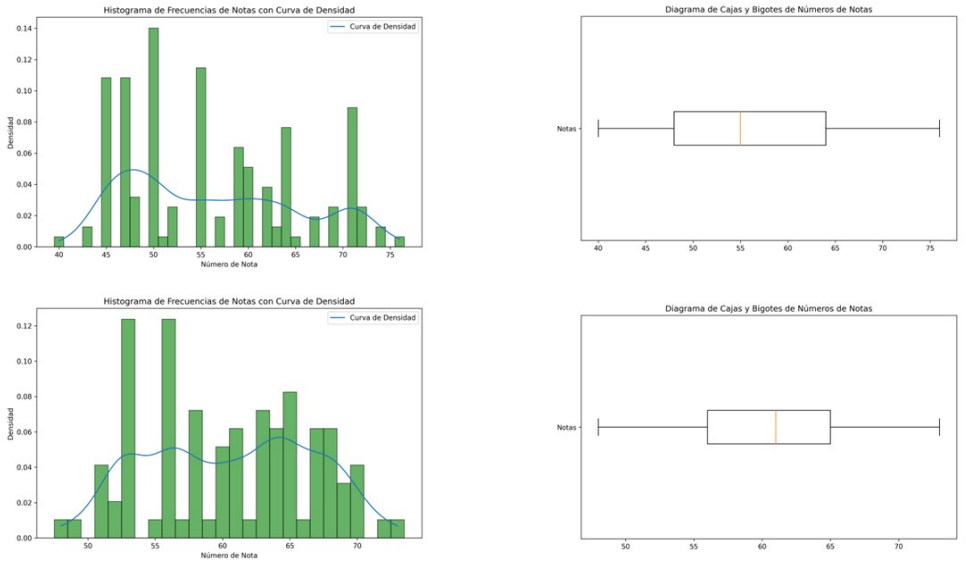

Figura 6.2: Histogramas de las notas de canciones del dataset

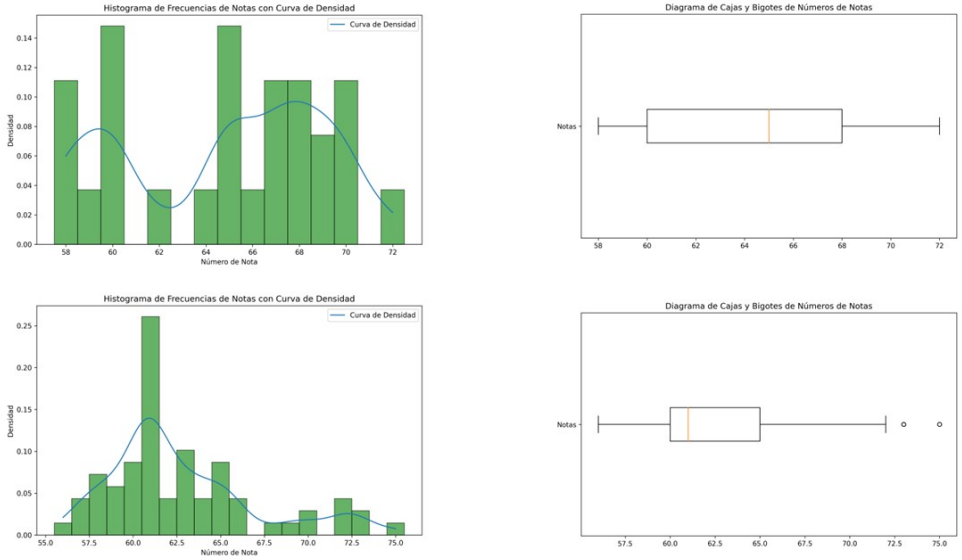

Figura 6.3: Histogramas de las notas de canciones del dataset

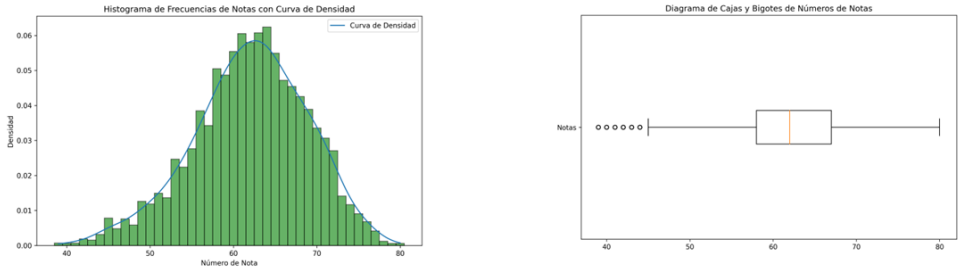

Figura 6.4: Datos de todas las notas del dataset

Luego se analizan las distintas medidas de tendencias centrales y momentos de tercer y cuarto orden, donde destaca la presencia en mayor medida de las notas centrales (no muy agudas ni muy graves), siendo la nota 64 la más presente y la media ubicada en la 62, con una desviación estandar de casi 7 notas, barriendo en total un espectro de 41 notas diferentes.

Media: **62.02289306684064**
Mediana: **62.0**
Moda: **64**
Varianza: **46.589078810011536**
Desviación estándar: **6.825619298643277**
Curtosis: **-0.029061325599949228**
Sesgo: **-0.327305242604834**
Rango: **41**
IQR (Rango intercuartil): **9.0**
Cuartil 1 (Q1): **58.0**
Cuartil 3 (Q3): **67.0**

Capítulo 7

Creación del Dataset

7.1. Dataset Base

Para el entrenamiento y la posterior evaluación del modelo, usamos el conjunto de datos: GuitarSet[1], que contiene grabaciones de una variedad de extractos musicales interpretados con una guitarra acústica, junto con anotaciones alineadas en el tiempo de posiciones de cuerdas y trastes, acordes, tiempos, tiempos fuertes y estilo de interpretación.

El conjunto de datos utilizado está compuesto por 360 archivos de audio (.wav) de aproximadamente 30 segundos de duración, cada uno acompañado de un archivo de anotaciones correspondiente (.jams). Estos archivos de anotaciones contienen información detallada sobre las características musicales de cada fragmento de audio, incluyendo:

- **Contorno de tono** : Esta anotación describe el cambio continuo en la frecuencia del sonido a lo largo del tiempo para cada cuerda individual.
- **Notas MIDI** : Esta anotación identifica la nota musical específica tocada en cada cuerda según el estándar MIDI (Musical Instrument Digital Interface).
- **Posición del pulso** : Esta anotación indica la ubicación temporal de cada pulso (golpe) dentro del compás.
- **Tempo**: Esta anotación indica la velocidad o ritmo general de la pieza musical.

7.2. Transformación del dataset

El modelo propuesto en este trabajo requiere archivos de audio en formato .mp3 y anotaciones musicales en formato .midi. Sin embargo, el conjunto de datos GuitarSet[1] no contiene archivos con estas características. Por lo tanto, se creó un nuevo conjunto de datos para entrenar y evaluar el modelo.

7.2.1. Creación del .midi

El proceso de conversión de archivos JAMS a MIDI se enfoca en la extracción las anotaciones de notas MIDI del archivo JAMS. Para cada nota, se crea una equivalente a una nota MIDI, asignando atributos como la velocidad, el tono, el tiempo de inicio y la duración. Estas notas se añaden al instrumento previamente creado (en este caso guitarra eléctrica limpia).

7.3. División de los datos

Para garantizar la correctitud en entrenamiento y evaluación del modelo es necesario dividir el conjunto de datos en subconjuntos disjuntos dedicados a cada tarea (entrenamiento, evaluación, test). Cada subconjunto debe tener proporciones similares en cuanto a heterogeneidad de los datos que contiene.

Con este objetivos llevamos a cabo el siguiente proceso:

- Extracción de características
- Extracción de componentes principales
- Clustering

7.3.1. Extracción de características de los archivos de audio

El flujo de trabajo para la extracción de características será el siguiente:

- Conversión de la señal de audio a mono.
- Remuestreo de una señal de audio.
- Construcción del Espectrograma de Mel.

Conversión de la señal de audio a mono

Los archivos de audio pueden tener varios canales que maximizan la percepción espacial e inmersión del oyente en el sonido.

Cuando un archivo de audio tiene múltiples canales la señal de audio se convierte a mono (único canal de audio) promediando los canales. Luego en lugar de tener dos o más canales de audio, se crea una sola señal de audio combinada. Dicho proceso se realiza calculando el promedio de las amplitudes de los canales para cada punto en el tiempo.

Remuestreo de una señal de audio

El proceso de remuestreo de una señal de audio consiste en cambiar el número de muestras de audio tomadas por segundo. La nueva tasa de muestreo será determinada por la configuración del modelo. Este proceso asegura que la señal de audio se ajuste a la tasa de muestreo requerida para posteriores análisis o transformaciones, como la creación del espectrograma de mel.

Construcción del Espectrograma de Mel

Hasta el momento el sonido está representado como un vector unidimensional. Sin embargo es muy conveniente preprocesar estas ondas para tener una representación temporal (eje x) de todas las frecuencias que aparecen (eje y).

Para esto usamos espectrogramas, que se basan en la aplicación de la transformada Fourier (FFT) en cada instante de tiempo. Utilizamos el espectrograma de Mel.

La escala Mel es una transformación logarítmica de la frecuencia de una señal, que permite obtener una representación del sonido similar a la capacidad auditiva humana. El ser humano no perciben frecuencias en una escala lineal . Es más difícil poder diferenciar entre frecuencias más altas y más fácil para las frecuencias más bajas.

7.3.2. Extracción de componentes principales

Debido a que los archivos de audio extraídos de GuitarSet presentan duraciones variables, las características extraídas también difieren en cantidad de un archivo a otro. Para estandarizar la dimensión de cada conjunto de características se aplicó

una técnica de extracción de componentes principales (PCA), logrando una representación de longitud invariante para todos los archivos.

7.3.3. Clustering

Una vez extraídas las características de los elementos del conjunto de datos y normalizadas sus dimensiones, se procedió a agruparlos por similitud utilizando técnicas de clustering. Posteriormente, se seleccionó el $70\,\%, 15\,\% y 15\,\%$ de cada cluster para conformar los subconjuntos de entrenamiento, validación y test, respectivamente.

Algoritmo: **Spectral Clustering**.
Un factor determinante en la elección de este algoritmo fue la naturaleza de los hiperparámetros requeridos. A diferencia de otros métodos de agrupamiento, el Agrupamiento Espectral solo necesita como entrada el número de clusters que deseamos extraer de los datos. Esto representa una ventaja significativa para nuestro problema, ya que no requerimos conocer de antemano la distancia o similitud entre los elementos del conjunto de datos. En espacios de alta dimensión, las distancias entre puntos tienden a ser muy grande, lo que afecta negativamente la capacidad de esta métrica para distinguir grupos bien definidos.

Una característica distintiva del clustering espectral radica en su capacidad para manejar datos que no presentan una estructura convexa. A diferencia de otros algoritmos de agrupamiento, como K-Means, que suponen que los grupos tienen formas convexas, el clustering espectral no impone esta restricción. Por esto seleccionamos este algoritmo, pues los datos con los que trabajamos no exhiben una estructura claramente definida.

7.3.4. Selección de hiperparámetros

Hiperparámetros

- **d**: Cantidad de dimensiones.
- **k**: Número de clusters.

En la figura 4.1 se muestran los resultados obtenidos al variar el valor de los hiperparámetros.

Interpretación de las métricas

- **Índice de Calinski-Harabasz:** Evalúa la calidad de un clustering basado en la relación entre la dispersión total entre clusters y la dispersión total dentro de los clusters. Cuanto mayor sea el valor, mejor será el clustering.

  - **Observaciones:**
    - Para ($k = 3$) y ($k = 4$), las configuraciones con menos componentes (por ejemplo, PCA=3) tienden a tener índices más altos, lo que sugiere que con un número menor de componentes principales se puede lograr una mejor separación entre los clusters.
    - Conforme aumenta k, los valores del índice tienden a disminuir para todas las configuraciones de PCA, indicando una menor calidad de clustering con más clusters.

- **Índice de Davies-Bouldin:** Mide la calidad de clustering basado en la relación entre la distancia dentro de los clusters y la distancia entre los clusters. Un valor más bajo indica un mejor clustering.

  - **Observaciones:**
    - Las configuraciones con PCA=8 y PCA=10 tienden a tener los valores más bajos para ($k = 4$), indicando que estos valores de PCA logran una mejor calidad de clustering con cuatro clusters.
    - Para la mayoría de las configuraciones de PCA, el índice tiende a disminuir a medida que (k) aumenta de 2 a 3 y luego tiende a estabilizarse o aumentar, indicando que los mejores valores de (k) están entre 2 y 4.
    - La configuración PCA=*3* generalmente tiene los valores más bajos para ($k = 2$) y ($k = 3$), sugiriendo que con menos componentes, se pueden obtener buenos resultados de clustering con pocos clusters.

Conclusión

- **Número óptimo de clusters (k)**

  - Según el índice de Calinski-Harabasz, el mejor número de clusters parece ser ($k = 3$) o ($k = 4$) para las configuraciones de PCA=3 y PCA=8.

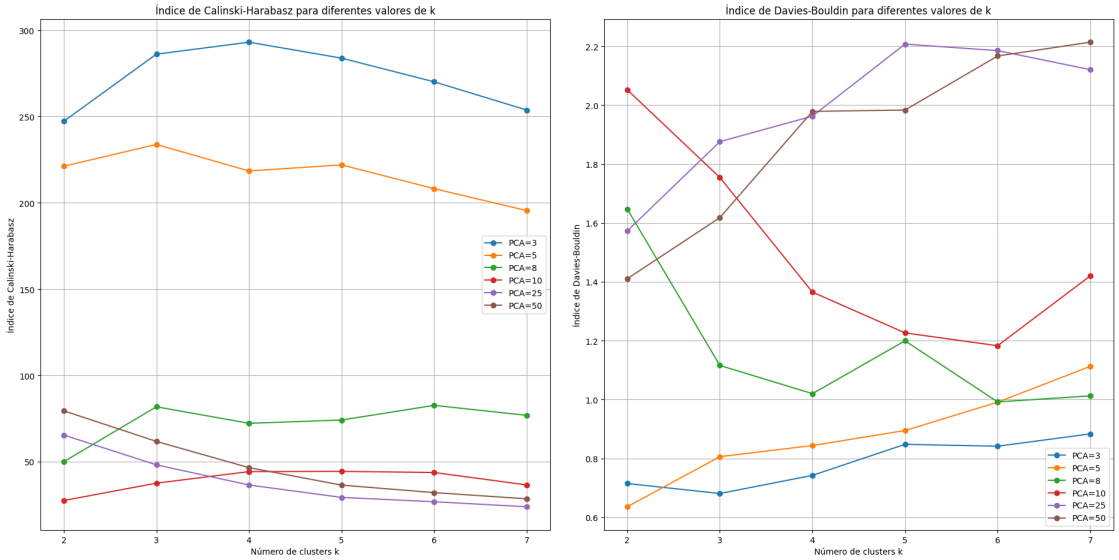

Figura 7.1: Muestreo de resultadados arrojados con diversos valores del hiperparámetro k (número de clusters) y p (cantidad de componentes principales) usando las métricas: Indice de Davies-Bouldin e Indice de Calinski-Harabasz

  - Según el índice de Davies-Bouldin, el mejor número de clusters también parece ser ($k = 3$) o ($k = 4$) para PCA=8 y PCA=10.

- **Configuración de *PCA* óptima:**
  - PCA=3 tiende a mostrar buenos resultados según ambas métricas, especialmente para ($k = 3$).
  - PCA=8 también muestra buenos resultados, especialmente según el índice de Davies-Bouldin para ($k = 4$).

Después de analizar las métricas de evaluación de clustering, decidimos utilizar una configuración de PCA con 3 componentes principales y un número de clusters $k = 5$ para nuestros datos. Esta combinación proporciona un equilibrio óptimo entre la calidad de separación de los clusters y la simplicidad del modelo.

7.3.5. Resultados

En el capítulo anterior se vieron las distribuciones de las notas presentes en los datos. Las figuras 4.2 - 4.7 muestran la distribución resultante de los datos en cada

partición final del dataset.

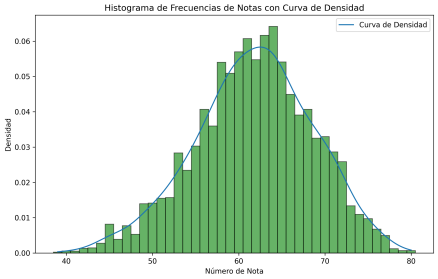

Figura 7.2: Histograma de frecuencias de las notas en una curva de densidad (entrenamiento).

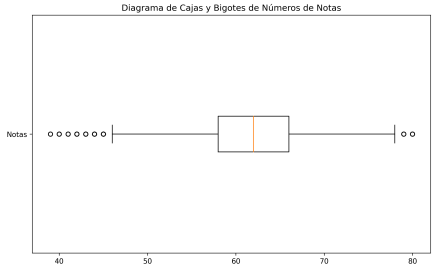

Figura 7.3: Diagrama de caja de bigotes del número de notas(entrenamiento).

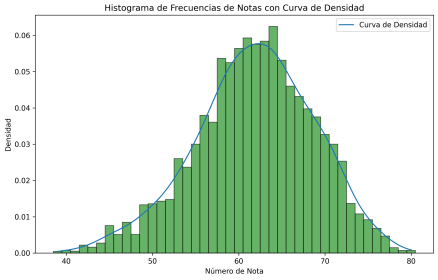

Figura 7.4: Histograma de frecuencias de las notas en una curva de densidad (validación).

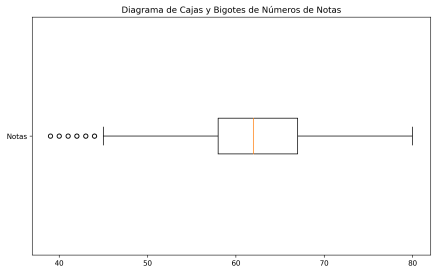

Figura 7.5: Diagrama de caja de bigotes del número de notas (validación).

Se realizó la prueba de hipótesis *Mannwhitneyu* para verificar la distribución de los datos en los distintos subsets. Los resultados de las pruebas de hipótesis realizadas no permiten rechazar la hipótesis nula. Esto indica que la distribución de los datos en cada uno de los subconjuntos no presenta diferencias significativas.

7.4. Estructración del dataset

Teniendo ya los subsets de entrenamiento, validación y test procedemos a estructurar y organizar los datos. Los archivos de audio del dataset tienen nombres

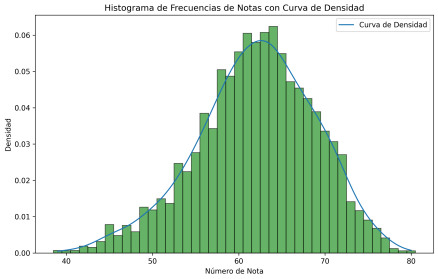

Figura 7.6: Histograma de frecuencias de las notas en una curva de densidad (test).

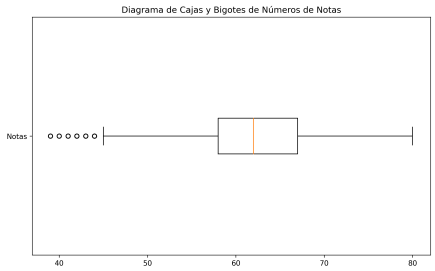

Figura 7.7: Diagramma de caja de bigotes del número de notas (test).

Comparación	Estadístico de prueba	Valor p
test vs train	77374216.0	1.0
train vs validation	50267026.0	0.999999981959656
validation vs test	13159618.0	0.9999993523704006

Tabla 7.1: Estadísticas de prueba y valores p para comparaciones entre conjuntos de datos

largos y con espacios en blanco lo que pueden posteriormente provocar errores imprevistos. Por esto creamos enlaces simbólicos entre archivos vacíos y los archivos originales. Estos nuevos archivos vacíos están ubicados en carpetas referentes a cada subset (entrenamiento, validación y test) y tiene nombre simples compuestos por el split (subset al que pertenecen) y un identificador (índice del archivo en el subset correspondiente) logrando así organizar y renombrar los archivos de audio y MIDI, creando enlaces simbólicos para ellos en nuevos directorios estructurados.

7.5. Extracción de características de los MIDI

Para garantizar la correcta evaluación del modelo necesitamos tener el valor real del objeto que estamos tratando de estimar con el mismo; dicho objeto es un archivo MIDI.

Pasos del proceso de extracción de características de un archivo MIDI:

- Creación de la curva de tempo.

- Extracción de notas.
- Procesamiento de mensajes MIDI.
- Conversión de notas.
    - Conversión de notas a etiquetas.
    - Conversión de notas a referencias.

7.5.1. Curva de tempo

Teniendo el archivo MIDI, se calcula el total de *ticks* en cada pista para encontrar el máximo número de *ticks*, y luego se procesan los mensajes de la primera pista para construir una curva de tiempo que convierte *ticks* a segundos, teniendo en cuenta los cambios de tempo.

Un *tick* es una pequeña unidad de tiempo en la música digital que se utiliza para medir la duración y el momento de los eventos en una pista musical. Ajustar un tick en función de los cambios de velocidad significa que se tiene en cuenta cómo varía la rapidez de la música (el tempo) en diferentes partes de la canción. Cuando la música se acelera o desacelera, el tiempo real que representa cada tick cambia por lo que se calculan estos ajustes para que cada tick corresponda con precisión a un tiempo en segundos, incluso si el ritmo de la música cambia a lo largo de la pista.

7.5.2. Extracción de notas

Teniendo ya el tempo por cada posible nota se extrae información detallada sobre cuándo se toca y se deja de tocar, así como el uso del pedal de sustain. Luego, se organiza y almacena esta información en un conjunto que detalla cada evento de nota, incluyendo su inicio, fin, tono y velocidad.

7.5.3. Procesamiento de mensajes MIDI

Un mensaje MIDI (Musical Instrument Digital Interface) es una unidad de información utilizada para comunicar datos entre dispositivos electrónicos musicales (sintetizadores, secuenciadores, computadoras). Estos mensajes permiten un control preciso de las notas que se tocan, su duración, la intensidad con que se ejecutan y

los cambios en los parámetros de los instrumentos.

En nuestro caso, los mensajes MIDI se analizan para identificar cuándo se activan y desactivan las notas, además de gestionar las variaciones del pedal de sustain. Este análisis es fundamental para reconstruir la interpretación musical contenida en un archivo MIDI.

7.5.4. Conversión de notas

Conversión de nota a etiqueta

Una vez extraídos los datos (inicio, fin, tono, velocidad) de forma individual de cada nota presente en el archivo de audio el próximo paso es confeccionar por cada etiqueta (inicio, fin, ocurrencia, velocidad) una representación en función de las ventanas de tiempo.

Finalmente tendremos las siguientes matrices de etiquetas:

- **MPE:** Indica la presencia o ausencia de una nota en cada ventana de tiempo (mpe: Estimación Multi-Tono, Tipo de datos: booleano).
- **OnSet:** Indica fuerza del inicio de una nota en cada ventana de tiempo (Tipo de datos: flotante).
- **OffSet:** Indica la fuerza del fin de una nota en cada ventana de tiempo (Tipo de datos: flotante).
- **Velocity:** Guarda la velocidad con la que se toca una nota en cada ventana de tiempo (Tipo de datos: entero).

Esta estructura de los datos permite almacenar y procesar información detallada sobre el comportamiento temporal de las notas musicales, incluyendo cuándo empiezan, cuándo terminan, su intensidad, y su presencia en cada ventana de tiempo.

Conversión de nota a referencia

Los archivos de notas musicales se procesan para generar archivos de referencia que contienen información sobre la frecuencia de las notas y su presencia en diferentes resoluciones temporales (16 ms y 10 ms). Nuevamente se llevan los datos de notas individuales a un formato estructurado.

Nota a etiqueta	**Nota a referencia**
Devuelve un diccionario con matrices que pueden ser fácilmente utilizadas para tareas de aprendizaje automático.	Genera archivos de texto que contienen información sobre las notas en un formato tabular, adecuado para análisis humanos o procesamiento adicional.
Se centra en la representación detallada de notas a través de matrices que incluyen la presencia de notas, onsets, offsets y velocidades.	Se enfoca en la representación de frecuencias de notas y su presencia en resoluciones temporales específicas, además de generar matrices de presencia de notas en fotogramas de 16 ms y 10 ms.

Tabla 7.2: Conversión de notas. Etiqueta vs Referencia

Capítulo 8

Herramienta de Transcripción

En este capítulo se presentan las adaptaciones realizadas al código base para permitir su uso como herramienta de transcripción de audio a partitura.

8.1. Adaptaciones del Código Base

El código original del proyecto estaba diseñado exclusivamente para la evaluación del modelo de transcripción y era altamente dependiente entre sus módulos, lo que limitaba su flexibilidad como herramienta de transcripción directa. Nos propusimos adaptar el código base con la adición de dos simples módulos para permitir su uso como una herramienta más versátil y fácil de integrar en diversos contextos de transcripción musical en un flujo de trabajo sencillo y directo (Figura 8.1)

8.1.1. Transcripción de Audio a MIDI

El primer módulo facilita la transcripción de un archivo de audio a formato MIDI utilizando un modelo de transcripción musical preentrenado. Este módulo ilustra de manera detallada el proceso completo de transcripción mediante el modelo.

Una vez que se carga el archivo de audio deseado para la transcripción, se utiliza una instancia de la clase AMT para extraer el espectrograma correspondiente. A continuación, el modelo se emplea para obtener todos los datos necesarios para la inferencia y la construcción subsiguiente del archivo MIDI. Este proceso se lleva a cabo dentro del ámbito original del código, sin necesidad de incorporar elementos externos

8.1.2. Transcripción de MIDI a partitura

Para esta segunda fase de transcripción se hace uso de la biblioteca externa **music21**. Esta biblioteca de Python está diseñada específicamente para la manipulación y análisis de partituras musicales en formato digital, permite cargar archivos MIDI y convertirlos directamente a representaciones de partitura que pueden ser procesadas y analizadas dentro del entorno de Python.

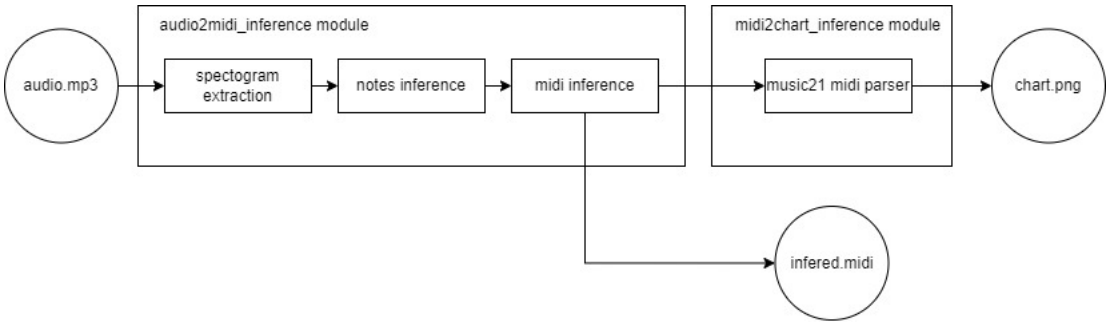

Figura 8.1: Diagrama de flujo del proceso de transcripción de audio a partitura

Capítulo 9

Entrenamiento

9.1. Modelo

El objetivo del entrenamiento es construir un modelo que, dado un matriz que espectrograma de un archivo de audio, identifique los fragmentos de interés y les asigne una de las siguientes etiquetas:

- **label_onset** : etiqueta de inicio de una nota.
- **label_offset** : etiqueta de fin de una nota.
- **label_mpe** : etiqueta de ocurrencia de una nota
- **label_velocity** : etiqueta de velocidad.

Para ello se hace uso del hFT-Transformer que es un método de transcripción automática de música que utiliza una arquitectura jerárquica de Transformer de frecuencia-tiempo de dos niveles. Este modelo muestra un buen desempeño para tener en cuenta las dependencias espectrales y temporales a largo plazo lo cual es esencial para la transcripción automática de piano. Esto es especialmente útil al determinar el inicio y el final precisos de cada nota, lo cual queremos explotar para el trabajo con la guitarra.

9.2. Flujo de trabajo

Una vez suministrados los archivos de audio utilizados para el entrenamiento y tras el preprocesamiento, se obtendrán cuatro archivos de tipo pkl. Estos archivos serán matrices, donde una será una será el espectrograma y las otras corresponderán a las etiquetas de velocidad, inicio y desplazamiento de notas.

Cada archivo pkl de características y sus correspondientes etiquetas son procesados por una clase denominada *MyDataset*, la cual hereda de la clase Dataset de PyTorch. Esta clase se encarga de manejar las muestras solicitadas en cada paso del entrenamiento, devolviendo en cada muestra un frame y las etiquetas correspondientes a ese fragmento de audio. La muestra a devolver es el resultado de la concatenación de la ventana de tiempo N solicitada con un padding de tamaño M en ambos extremos.

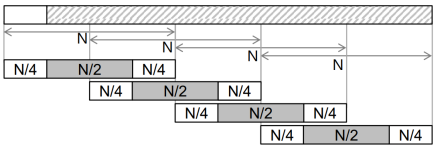

Figura 9.1: Paso de ventana empleado

El motivo de este enfoque es que el modelo pueda reconocer patrones en el inicio y el final de cada nota, además de servir para mitigar la pérdida de información en la primera etapa del entrenamiento.

Cada muestra de audio que ingresa al modelo atravesará primero un bloque convolucional unidimensional que opera en el eje temporal. El vector resultante de este procesamiento pasará por el primer codificador Transformer, que trabaja en el eje de frecuencia. Este codificador analiza la dependencia entre las características espectrales.
El resultado del codificador será procesado por un decodificador que convertirá el eje de frecuencia en un eje de tonos.

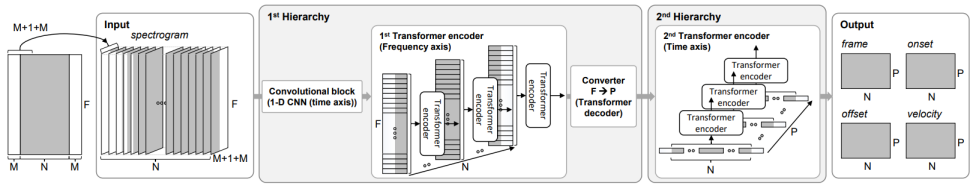

Figura 9.2: N: número de muestras en cada fragmento de procesamiento, M: longitud del padding, F: número de bins de frecuencia, P: número de tonos

La salida del decodificador se procesará con otro codificador Transformer en el eje temporal. Aquí se analiza la dependencia temporal de las características en cada cuadro de procesamiento temporal.

Similar a la primera jerarquía, las salidas de la segunda jerarquía se obtienen a través de un módulo lineal y una función sigmoide. Las pérdidas para las salidas de la segunda jerarquía (output_2nd) se evalúan de la misma manera que para las salidas de la primera jerarquía (output_1st), sumándolas con los coeficientes correspondientes para calcular la pérdida total del modelo.

9.2.1. Funciones de pérdida

Para el entrenamiento se utilizan las métricas:

- **Entropía Cruzada Binaria (Binary Cross-Entropy)** : Para las predicciones de frame, onset y offset.
- **Entropía Cruzada Categórica (Categorical Cross-Entropy)** : Para la predicción de velocidad, con 128 categorías.

La selección de estas se deben al registro como resultado experimental de un rendimiento mucho mejor que el MSE [8]

En el bucle de entrenamiento, se itera por la cantidad de épocas establecida. En cada época, se evalúa el modelo utilizando el conjunto de datos de validación y se calculan las pérdidas en los dos niveles de la jerarquía. Durante este proceso, se ajusta la tasa de aprendizaje para optimizar el rendimiento del modelo.

Una vez finalizado el proceso de optimización en cada época, se verifica la magnitud de la pérdida, registrándose la época que reporta la menor pérdida en el conjunto de validación. Esto se debe a que una menor pérdida en el conjunto de validación suele ser indicativa de un mejor rendimiento generalizado del modelo y ayuda a prevenir el sobreajuste (overfitting).

Aunque las salidas de los dos niveles de la jerarquía se utilizan para calcular las pérdidas durante el entrenamiento, solo la salida del último nivel se emplea en la inferencia. Para este experimento, se seleccionó un total de 20 épocas de entrenamiento utilizando el conjunto de datos GuitarSet.

Capítulo 10

Evaluación

En este capítulo se presentan los métodos mediante los cuales se extraen las métricas de interés y se listan de manera explícita dichas métricas

10.1. Evaluación de la Estimación Multi-Tono (MPE)

La primera etapa del proceso de evaluación busca determinar la precisión del modelo para identificar las frecuencias de las notas que suenan simultáneamente en un momento dado. Para ello, el código carga las anotaciones MPE de referencia del conjunto de datos GuitarSet, las cuales representan las frecuencias reales de las notas en cada frame. Luego, carga las estimaciones MPE generadas por el modelo durante la inferencia. Posteriormente, el código convierte el formato de las estimaciones MPE a un formato compatible con la librería `mir_eval`. Finalmente, se utiliza la función `mir_eval.multipitch.evaluate` para comparar las estimaciones del modelo con las anotaciones de referencia, calculando las siguientes métricas:

- Precisión: Porcentaje de notas correctamente identificadas por el modelo en relación al total de notas identificadas.
- Recuperación: Porcentaje de notas correctamente identificadas por el modelo en relación al total de notas presentes en la anotación de referencia.
- Exactitud: Porcentaje de frames en los que el modelo identifica correctamente todas las notas.
- Error de Sustitución: Porcentaje de frames en los que el modelo sustituye una nota correcta por otra incorrecta.

- Error de Omisión: Porcentaje de frames en los que el modelo no identifica una nota presente en la anotación de referencia.
- Error de Falsa Alarma: Porcentaje de frames en los que el modelo identifica una nota que no está presente en la anotación de referencia.
- Error Total: Suma de todos los errores anteriores.

10.2. Evaluación de la Transcripción de Notas

La segunda etapa de la evaluación se centra en determinar la precisión del modelo para transcribir las notas musicales, considerando su inicio, finalización, tono y, opcionalmente, su velocidad. El proceso comienza con la carga de las anotaciones de referencia de las notas de GuitarSet, las cuales incluyen la hora de inicio, la hora de finalización, el tono (frecuencia) y, si se especifica, la velocidad. Luego, se cargan las transcripciones de notas generadas por el modelo en formato JSON. El código convierte estas transcripciones a un formato compatible con la librería `mir_eval` y, finalmente, utiliza la función `mir_eval.transcription.evaluate` (o `mir_eval.transcription_velocity.evaluate` si se incluye la velocidad) para comparar las transcripciones del modelo con las anotaciones de referencia. Esto permite calcular las siguientes métricas:

- Precisión: Porcentaje de notas correctamente transcritas por el modelo en relación al total de notas transcritas.
- Recuperación: Porcentaje de notas correctamente transcritas por el modelo en relación al total de notas presentes en la anotación de referencia.
- F-measure: Media armónica de la precisión y la recuperación.
- Promedio de Relación de Superposición: Promedio de la proporción de superposición entre cada nota estimada y la nota de referencia más cercana.
- Precisión/Recuperación/F-measure sin desplazamiento: Métricas de precisión, recuperación y F-measure calculadas ignorando los errores de desplazamiento.
- Precisión/Recuperación/F-measure de inicio: Precisión, recuperación y F-measure para los inicios de las notas.

- Precisión/Recuperación/F-measure de final: Precisión, recuperación y F-measure para los finales de las notas.
- (Opcional) Precisión/Recuperación/F-measure con velocidad: Métricas que consideran la precisión de la velocidad de la nota.

Capítulo 11

Resultados

Los resultados experimentales indican que el modelo empleado muestra un desempeño notable en la transcripción de partituras de guitarra utilizando el conjunto de datos GuitarSet. Este dataset contiene 360 grabaciones de alta calidad. Aunque es relativamente pequeño en comparación con otros utilizados para la misma tarea, ha generado un desempeño destacado. Atribuimos estos buenos resultados a la metodología aplicada y a la calidad del modelo en el que se basa este trabajo [8]. Además:

- **Partición del Dataset**: Se empleó un enfoque de clustering para la confección de las particiones de entrenamiento, validación y prueba.
- **Variedad de Estilos**: GuitarSet contiene diversos estilos de guitarra, lo que permite al modelo adaptarse y aprender diferentes patrones y técnicas de interpretación.

Aunque estos puntos requieren más evidencia para ser plenamente validados. En contraste, el rendimiento del modelo fue deficiente al transcribir partituras de canciones multiinstrumentales o monoinstrumentales de cuerda con acompañamiento vocales. Se evaluó el modelo utilizando midis correspondientes a canciones que incluyen múltiples instrumentos y los resultados fueron insatisfactorios.

De manera similar ocurrió con canciones que tenían acompañamiento vocal. Consideramos que esto puede deberse a las siguientes razones:

- **Complejidad del Audio Multiinstrumental**: La presencia de múltiples instrumentos simultáneamente añade una complejidad adicional que el modelo no pudo manejar eficientemente.
- **Señales de Voz**: Las características únicas de las señales vocales, como la variabilidad en el timbre y la entonación, presentan un desafío significativo para el modelo.
- **Dataset Limitado**: El dataset empleado es pequeño y mono-instrumental, lo cual podría haber limitado la capacidad del modelo para generalizar a instrumentos que no son de cuerda.

Tabla 11.1: Resultados en distintos modelos

Modelo	Métricas		
	P	R	F1
htf-Transformer(nuestro)	**91.5**	**92.3**	**91.9**
Timbre-Trap	74.8	74.0	72.7
Transfer of Knowledge	79.5	73.3	75.4
MT3	-	-	78.8
MultiTrack	-	-	90.3

En la tabla se presentan las métricas de Precision (P), Recall (R) y F1-Score (F1) en distintos modelos utilizados en la tarea de identificación de notas , basados en las transcripciones obtenidas de cada archivo del dataset GuitarSet. Todos los modelos presentes en la tabla hacen uso de una arquitectura tipo transformer.

11.1. Repercusiones Éticas

La automatización de la transcripción musical puede impactar significativamente a los profesionales de la música, incluyendo transcriptores y músicos que dependen de la creación manual de partituras como fuente de ingresos. Lo que puede llevar a la desvalorización de habilidades y conocimientos especializados.

Además, la democratización de herramientas automatizadas plantea cuestiones sobre la calidad y la exactitud de las transcripciones generadas. Si bien los modelos de aprendizaje profundo han demostrado una alta precisión en ciertos contextos, siempre existe el riesgo de errores en las transcripciones, especialmente en estilos o técnicas de ejecución menos comunes. Esto puede resultar en la difusión de información musical incorrecta, afectando negativamente tanto a los intérpretes como a los estudiantes de música.

Otra consideración ética importante es la propiedad intelectual y los derechos de autor. La transcripción automática de música puede facilitar la copia y distribución no autorizada de obras protegidas por derechos de autor, planteando desafíos legales y éticos sobre el uso justo y la protección de la propiedad intelectual.

Capítulo 12

Recomendaciones

En este capítulo se presentan una serie de recomendaciones para mejorar la funcionalidad y el rendimiento de la herramienta de transcripción musical desarrollada, así como consideraciones que surgieron durante nuestro trabajo.

12.1. Evaluación con Datasets Multiinstrumento

Se recomienda probar el modelo con conjuntos de datos multiinstrumento para evaluar su capacidad de transcribir música con múltiples instrumentos simultáneos. Esto permitirá identificar las limitaciones del modelo en escenarios más complejos y permitirá desarrollar estrategias para mejorar su rendimiento en este tipo de situaciones.

12.2. Evaluar y Mejorar la Robustez a Ruido

Resulta sensato evaluar la robustez del modelo frente al ruido dada la naturaleza de la calidad de los audios en el internet, donde son sometidos a compresión y alteraciones de todo tipo. Esto podría implicar la implementación de técnicas de reducción de ruido o la inclusión de un componente de aprendizaje que sea capaz de identificar y filtrar el ruido en las señales de audio.

12.3. Limpieza de Partituras Resultantes

Se recomienda desarrollar algoritmos de post-procesamiento para limpiar las partituras resultantes de la transcripción. Estos algoritmos podrían identificar y corregir errores comunes, como la detección de notas fantasma (notas que no deberían estar presentes), la fusión de notas adyacentes que deberían estar separadas, y la corrección de errores de tono o duración.

12.4. Colaboración con Expertos en Música

Si bien tenemos una herramienta que nos permite extraer partituras, escapa de nuestra área de conocimiento la calidad de las mismas. Debemos consultar con expertos acerca de dicha calidad y valorar la posibilidad de realizar evaluaciones de la misma automáticamente, teniendo en cuenta los puntos señalados como importantes en una comparativa.

Capítulo 13

Conclusiones

La investigación confirma que los modelos basados en deep learning, específicamente los Transformers y sus variantes, han superado significativamente a las técnicas tradicionales en la transcripción automática de música. El modelo *hTF-Transformer* ha mostrado una capacidad notable para manejar la complejidad temporal y frecuencial de las señales de audio. Los resultados obtenidos con diferentes modelos indican que es posible alcanzar altas precisiones en datasets específicos. El modelo analizado en cuestión logró una precisión de aproximadamente un 90 % en datasets como Maestro y en esta ocasión se mantienen dichos resultados en *GuitarSet*, demostrando la eficacia del mismo para la transcripción de música.

13.1. Importancia del Dataset:

La creación y transformación del dataset fueron cruciales para el éxito del entrenamiento y evaluación de los modelos. Un dataset bien estructurado y diversificado permitió que los modelos generalizaran mejor a diferentes estilos musicales.

13.2. Adaptación de Herramientas:

Las adaptaciones realizadas al código base para la transcripción de audio a MIDI y de MIDI a partitura fueron efectivas, permitiendo una conversión precisa y eficiente. Estas herramientas son esenciales para el pipeline de transcripción y su calidad

impacta directamente en los resultados finales.

13.3. Análisis de Datos:

El análisis de los datos, incluyendo la extracción de características y la aplicación de técnicas de clustering, proporcionó insights valiosos para mejorar el rendimiento de los modelos. La selección de hiperparámetros y la extracción de componentes principales fueron particularmente importantes para optimizar el desempeño.

Capítulo 14
Bibliografía

[1] Q. Xi, R. Bittner, J. Pauwels, X. Ye, and J. P. Bello, *Guitarset: A Dataset for Guitar Transcription*, in 19th International Society for Music Information Retrieval Conference, Paris, France, Sept. 2018.

[2] N. Agudelo, J. Rada. Lower bounds of Nikiforov's energy over digraphs, Linear Algebra Appl. 494 (2016): 156-164.

[3] M. Aguieiras, M. Robbiano, A. Bonifacio. An improved upper bound of the energy of some graphs and matrices. MATCH Commun. Math. Comput. Chem. 74 (2015) 307-320.

[4] E. Andrade, M. Robbiano, B. San Mart'in. A lower bound for the energy of symmetric matrices and graphs. Linear Algebra Appl. 513 (2017) 264-275.

[5] M. Aouchiche, P. Hansen. A survey of Nordhaus-Gaddum type relations. Discret Appl. Math 161 (2013) 466-546.

[6] F. Ashraf, B. Tayfeh-Rezaie. Nordhaus-Gaddum type inequalities for Laplacian and signless Laplacian eigenvalues. The Electronic Journal of Combinatorics 21 (3) (2014) 3-6.

[7] R. Balakrishnan, K. Ranganathan. A Textbook Of Graph Theory. Springer Science+Business Media New York 2012.

[8] Keisuke Toyama, Taketo Akama, Yukara Ikemiya, Yuhta Takida, Wei-Hsiang Liao, and Yuki Mitsufuji. Automatic piano transcription with hierarchical frequency-time transformer. In *Proceedings of the 2023 Conference on Advanced Sound Processing*, 2023.