UNIVERSIDAD PONTIFICIA COMILLAS
ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA (ICAI)
MÁSTER EN INGENIERÍA DE TELECOMUNICACIÓN

ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA (ICAI)

MÁSTER EN INGENIERÍA DE TELECOMUNICACIÓN

ANÁLISIS MUSICAL BASADO EN IA A PARTIR DE SEÑALES DE AUDIO

Autor: CARLES OLUCHA ROYO
Directores: MIGUEL ÁNGEL SANZ BOBI
JAVIER MATANZA DOMINGO

Madrid
Junio 2025

Declaro, bajo mi responsabilidad, que el Proyecto presentado con el título

Análisis musical basado en IA a partir de señales de audio

en la ETS de Ingeniería - ICAI de la Universidad Pontificia Comillas en el curso académico 2024/25 es de mi autoría, original e inédito y no ha sido presentado con anterioridad a otros efectos.

El Proyecto no es plagio de otro, ni total ni parcialmente y la información que ha sido tomada de otros documentos está debidamente referenciada.

Fdo: CARLES OLUCHA ROYO

Firmado digitalmente por CARLES|OLUCHA|ROYO
Nombre de reconocimiento (DN): cn=CARLES|OLUCHA|ROYO, serialNumber=53794604L, givenName=CARLES, sn=OLUCHA ROYO, ou=CIUDADANOS, o=ACCV, c=ES

Autorizada la entrega del proyecto

DIRECTORES DEL PROYECTO

Miguel Ángel Sanz Bobi

Firmado por SANZ BOBI MIGUEL ANGEL - ***6599** el día 30/06/2025 con un certificado emitido por AC FNMT Usuarios

Javier Matanza Domingo

Firmado por MATANZA DOMINGO JAVIER - ***4397** el día 02/07/2025 con un certificado emitido por AC FNMT Usuarios

UNIVERSIDAD PONTIFICIA COMILLAS
ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA (ICAI)
MÁSTER EN INGENIERÍA DE TELECOMUNICACIÓN

ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA (ICAI)

MÁSTER EN INGENIERÍA DE TELECOMUNICACIÓN

ANÁLISIS MUSICAL BASADO EN IA A PARTIR DE SEÑALES DE AUDIO

Autor: CARLES OLUCHA ROYO
Directores: MIGUEL ÁNGEL SANZ BOBI
JAVIER MATANZA DOMINGO

Madrid
Junio 2025

Agradecimientos

A mi familia, por su apoyo constante durante toda la carrera.
A mis tutores de TFM, Miguel Ángel Sanz Bobi y Javier Matanza Domingo, por su orientación y ayuda en cada fase del proyecto.
A Èlia Villar Llopico (clarinete), Ana Cano Leal y Merche Sánchez Marco (flauta), Andreu Villar Llopico (saxo) y Joan López de Soria (violín), por prestarme sus audios y permitirme probar la aplicación con distintos instrumentos.
Y a Lluís Castañ Fernández por enseñarme a apreciar la guitarra clásica.

Resumen y palabras clave

Resumen

El presente Trabajo de Fin de Máster tiene como objetivo desarrollar un sistema capaz de transcribir señales de audio en partituras mediante el uso de técnicas de procesamiento digital de señales y algoritmos de inteligencia artificial. Para ello, se han implementado modelos específicos que permiten identificar notas musicales tanto en instrumentos melódicos, como la flauta, como en instrumentos polifónicos, como la guitarra. El sistema se basa en la conversión de las señales temporales en representaciones en frecuencia, la extracción de características relevantes y la clasificación automática de las notas mediante modelos de aprendizaje automático (KNN, MLP y RNN). Además, se ha incorporado un módulo de filtrado personalizado en frecuencia para mejorar la precisión del análisis en entornos ruidosos. El sistema desarrollado permite visualizar los resultados en forma de partitura y presenta un enfoque modular y extensible que sienta las bases para futuras mejoras.

Palabras clave

Procesamiento de audio, inteligencia artificial, transcripción musical automática, aprendizaje automático, flauta, guitarra, señales digitales, reconocimiento de notas, filtrado en frecuencia, Fourier.

Abstract

This Master's Thesis aims to develop a system capable of transcribing audio signals into musical scores through the use of digital signal processing techniques and artificial intelligence algorithms. To achieve this, specific models have been implemented to identify musical notes from both monophonic instruments, such as the flute, and polyphonic instruments, such as the guitar. The system is based on the transformation of time-domain signals into frequency-domain representations, the extraction of relevant features, and the automatic classification of notes using machine learning models (KNN, MLP, and RNN). Additionally, a custom frequency filtering module has been integrated to enhance accuracy in noisy environments. The developed system enables the visualization of results as a musical score and offers a modular, extensible framework that lays the foundation for future improvements.

Keywords

Audio processing, artificial intelligence, automatic music transcription, machine learning, flute, guitar, digital signals, note recognition, frequency filtering, Fourier.

Contenido

Índice de ilustraciones

Índice de tablas

Índice de ecuaciones

Capítulo 1. Introducción

La música constituye una de las formas de expresión más antiguas y universales de la humanidad. Su análisis, interpretación y transcripción han evolucionado a lo largo del tiempo gracias a los avances tecnológicos, en particular en el ámbito del procesamiento digital de señales (DSP) y la inteligencia artificial (IA). En la actualidad, estas tecnologías permiten abordar con mayor precisión y automatización tareas complejas como la transcripción musical, el reconocimiento de notas y la clasificación de instrumentos.

Este Trabajo de Fin de Máster se enmarca en dicho contexto tecnológico y propone el desarrollo de un sistema de análisis musical automatizado a partir de señales de audio. El objetivo principal es transformar grabaciones musicales en partituras legibles, aplicando un enfoque híbrido que combina técnicas clásicas de análisis en el dominio de la frecuencia con modelos modernos de aprendizaje automático. La solución propuesta aborda tanto instrumentos melódicos, que producen una sola nota a la vez, como instrumentos polifónicos, capaces de emitir múltiples notas simultáneamente.

A pesar de los avances actuales en tecnologías de análisis musical, la transcripción automática de audio sigue presentando numerosos retos. (van den Oord, 2016) Los sistemas existentes enfrentan dificultades para manejar grabaciones con ruido de fondo, variaciones tímbricas, superposición de armónicos o interpretación libre por parte del músico. Además, muchos modelos requieren grandes volúmenes de datos etiquetados para alcanzar niveles aceptables de precisión, lo cual no siempre es viable en entornos reales o con instrumentos menos comunes. (Benetos E.) La identificación exacta de las notas musicales en una grabación depende de múltiples factores: calidad de la señal, claridad del ataque de la nota, presencia de armónicos y otros elementos acústicos que pueden interferir en la detección. Esta complejidad se agrava en instrumentos polifónicos, donde las frecuencias de diferentes notas se combinan en el espectro, dificultando la separación e identificación individual de cada una. (sigsep, 2024)

Por tanto, surge la necesidad de diseñar un sistema robusto, modular y extensible que sea capaz de operar en contextos realistas, tanto con instrumentos melódicos como polifónicos, y que permita una interpretación fiable de la señal sonora. Este sistema debe ser capaz de transformar el contenido auditivo en una representación estructurada como una partitura, facilitando así su análisis, estudio o reutilización musical. El presente proyecto busca dar respuesta a esta necesidad mediante el uso de técnicas de segmentación en el dominio temporal, filtrado adaptativo en el dominio de la frecuencia, y clasificación automatizada de notas con modelos de aprendizaje supervisado. Esta propuesta pretende contribuir al desarrollo de soluciones accesibles y precisas en el ámbito de la transcripción musical automática, con aplicaciones tanto en el entorno educativo como en la industria creativa.

1.1 Objetivos

El proyecto “Análisis musical basado en inteligencia artificial a partir de señales de audio” tiene como finalidad principal el desarrollo de un sistema capaz de interpretar grabaciones musicales y transformarlas en partituras legibles. Esta transcripción se realizará mediante el análisis de señales digitales y la aplicación de modelos de aprendizaje automático diseñados para identificar con precisión las notas tocadas por diferentes instrumentos.

Para alcanzar este objetivo general, se han definido los siguientes objetivos específicos:

1. **Diseñar e implementar un modelo de detección de notas para instrumentos melódicos:** Este objetivo se centra en el tratamiento de señales correspondientes a instrumentos que emiten una única nota a la vez, como la flauta travesera. La tarea incluye la segmentación del audio, la representación en frecuencia mediante técnicas como la transformada de Fourier, y la posterior identificación de la nota dominante en cada segmento.
2. **Desarrollar un modelo de detección de notas para instrumentos polifónicos:** En este caso, se abordan instrumentos como la guitarra, capaces de emitir varias notas de forma simultánea. Se requiere el diseño de vectores de características adecuados para representar el contenido armónico y frecuencial del audio, así como la implementación de modelos supervisados (como KNN, MLP o RNN) capaces de clasificar de forma robusta las notas presentes en entornos más complejos.
3. **Implementar un sistema de transcripción musical automatizada:** Una vez identificadas las notas en los diferentes segmentos temporales, el sistema debe ser capaz de estructurar dicha información y representarla en forma de partitura. Esta representación facilita la comprensión musical y la posterior interpretación humana, y se plantea como una salida legible y coherente que respete el orden y la duración relativa de las notas.
4. **Diseñar una interfaz gráfica sencilla e intuitiva para el usuario final:** Para facilitar el acceso al sistema por parte de músicos, estudiantes o usuarios no expertos en programación, se desarrollará una interfaz interactiva que permita cargar audios, seleccionar parámetros como el instrumento o el modelo de análisis, y visualizar tanto el espectrograma como la partitura generada. Esta interfaz busca democratizar el uso de la herramienta y fomentar su aplicación en contextos educativos y creativos.
5. **Evaluar el rendimiento del sistema en distintos contextos musicales:** Para validar la eficacia del sistema, se realizarán pruebas sobre grabaciones con distintos niveles de complejidad, estilos musicales y condiciones de ruido. Esta evaluación permitirá analizar métricas como precisión, sensibilidad y robustez, identificando fortalezas y posibles mejoras del sistema propuesto.

Estos objetivos permiten estructurar el desarrollo del trabajo en fases claras y consecutivas, asegurando un enfoque sistemático en el diseño, implementación y validación del sistema de análisis musical automático propuesto, así como su accesibilidad para un amplio rango de usuarios.

1.2 Justificación

La transcripción automática de música a partir de grabaciones de audio representa uno de los desafíos más complejos en el ámbito del procesamiento digital de señales y la inteligencia artificial (Benetos E. D., 2013). A pesar de los avances recientes, las soluciones actuales siguen enfrentándose a múltiples limitaciones, especialmente en lo que respecta al reconocimiento de notas en entornos con ruido, en la presencia de instrumentos polifónicos y en la generación de partituras legibles.

Este proyecto nace como respuesta a esa necesidad técnica, educativa y creativa. Por un lado, proporciona una herramienta que facilita el análisis musical tanto a músicos como a docentes, permitiendo estudiar piezas sin necesidad de realizar una transcripción manual. Por otro, se orienta hacia la accesibilidad tecnológica, ya que permite que cualquier usuario sin conocimientos avanzados en programación o en procesamiento de señales pueda obtener una partitura a partir de un archivo de audio.

Además, este trabajo busca superar las limitaciones de herramientas comerciales actuales, que muchas veces presentan restricciones en cuanto al número de instrumentos que pueden analizar, la calidad de la predicción o la capacidad para operar con grabaciones reales no estandarizadas. En este contexto, el sistema desarrollado no solo pretende ofrecer una solución funcional, sino también servir de base para futuras investigaciones, incorporando técnicas de inteligencia artificial en un marco aplicable, comprensible y replicable (Celemony Software GmbH / Lunaverus / Ableton / Magenta).

1.3 Metodología del trabajo

Este proyecto sigue una metodología estructurada que permite abordar el problema de la transcripción musical desde una perspectiva técnica, progresiva y aplicada. El trabajo se ha dividido en varias fases, cada una centrada en resolver de forma específica los desafíos relacionados con la interpretación de señales musicales y la generación de partituras a partir de grabaciones reales.
En primer lugar, se ha llevado a cabo una fase de investigación y revisión del estado del arte en tecnologías de transcripción musical, procesamiento de señales y modelos de aprendizaje automático aplicados al análisis sonoro. Esta fase ha permitido identificar tanto las limitaciones de las soluciones existentes como los enfoques más prometedores en entornos reales.

Para estructurar adecuadamente este enfoque, el proyecto se ha dividido en once fases sucesivas que reflejan tanto la progresión lógica como técnica del desarrollo, tal y como se muestra en la Ilustración 1.

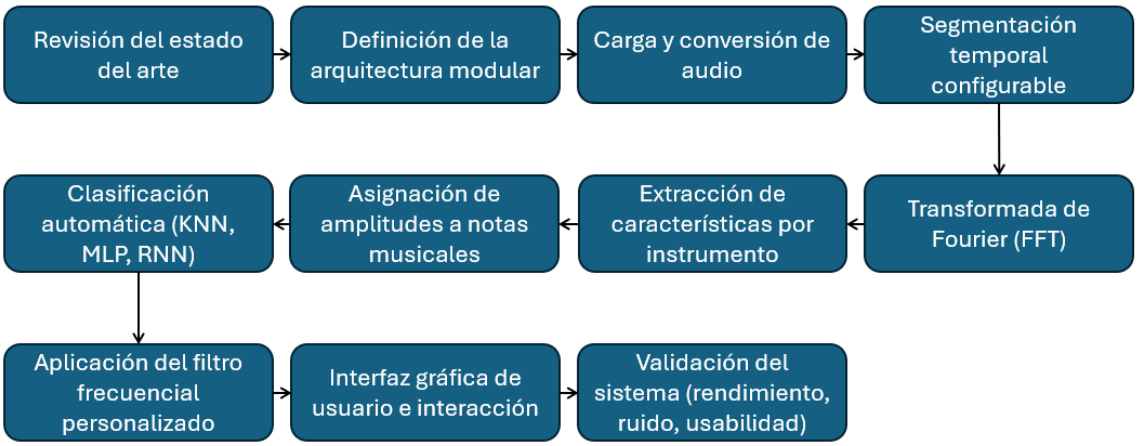


Ilustración 1

A continuación, se ha definido la arquitectura del sistema, basada en un conjunto de módulos funcionales independientes. El proceso comienza con la carga y conversión del archivo de audio, seguido de una segmentación temporal en intervalos configurables. Posteriormente, se realiza la transformación de la señal al dominio de la frecuencia mediante la Transformada Rápida de Fourier (FFT), lo que permite obtener un espectro detallado de frecuencias por cada segmento.

En la etapa de extracción de características, se calcula la distribución de amplitudes dentro de un rango de frecuencias adaptado al instrumento seleccionado. Estas amplitudes se asignan a notas musicales mediante un diccionario de referencia generado automáticamente. En el caso de los instrumentos melódicos (como la flauta), se analiza la nota dominante de cada segmento; en el caso de los polifónicos (como la guitarra), se genera un vector de 108 características representando la energía asociada a cada nota posible.

La siguiente fase corresponde a la clasificación automática. Se han desarrollado y probado diferentes modelos supervisados, entre ellos:

- **K-Nearest Neighbors (KNN)**, por su bajo coste computacional y eficacia con vectores bien definidos.
- **Redes Neuronales Multicapa (MLP)**, entrenadas sobre representaciones de amplitud por nota.
- **Redes Neuronales Recurrentes (RNN)**, capaces de modelar secuencias temporales para instrumentos melódicos.

Cada modelo se ha entrenado con grabaciones propias, asegurando control sobre la señal y etiquetas, y evaluado posteriormente con grabaciones distintas para verificar su capacidad de generalización.

Además, se ha desarrollado un **módulo de filtrado en frecuencia**, diseñado para reforzar las frecuencias características de cada instrumento y reducir la influencia del

ruido. Este módulo permite aplicar un perfil de ganancia personalizado sobre el espectro, lo que mejora la fiabilidad de la detección en condiciones no ideales.

Por último, se ha implementado una **interfaz de usuario gráfica**, que permite interactuar con el sistema sin necesidad de conocimientos técnicos. A través de esta interfaz, el usuario puede cargar un archivo de audio, seleccionar el instrumento y el modelo, visualizar el espectrograma, y obtener una partitura estimada en tiempo real.

Esta metodología modular facilita la validación del sistema, la integración de nuevas funcionalidades y su aplicación práctica en entornos reales, tanto educativos como musicales o investigativos. El sistema ha sido sometido a diversas pruebas de rendimiento que han evaluado su precisión, sensibilidad al ruido y facilidad de uso por parte de usuarios no técnicos.

A continuación, se presenta el cronograma de trabajo seguido durante el desarrollo del proyecto:

	septiembre				**octubre**				**noviembre**				**diciembre**			
Tarea	**1**	**2**	**3**	**4**	**1**	**2**	**3**	**4**	**1**	**2**	**3**	**4**	**1**	**2**	**3**	**4**
Revisión de la documentación existente	■	■	■	■												
Búsqueda y organización de datos			■	■	■	■										
Generación del dataset para el entrenamiento de modelos							■	■	■							
Realización de los primeros algoritmos para instrumentos melódicos										■	■	■	■	■	■	■
Realización de los primeros modelos para instrumentos polifónicos																
Añadir otros instrumentos melódicos																
Desarrollo de una interfaz																
Redacción del TFM y sus anexos	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■

	enero				**febrero**				**marzo**				**abril**				**mayo**			
Tarea	**1**	**2**	**3**	**4**	**1**	**2**	**3**	**4**	**1**	**2**	**3**	**4**	**1**	**2**	**3**	**4**	**1**	**2**	**3**	**4**
Revisión de la documentación existente																				
Búsqueda y organización de datos																				
Generación del dataset para el entrenamiento de modelos																				
Realización de los primeros algoritmos para instrumentos melódicos																				
Realización de los primeros modelos para instrumentos polifónicos	■	■	■	■	■	■	■	■												
Añadir otros instrumentos melódicos									■	■	■									
Desarrollo de una interfaz												■	■	■	■	■				
Redacción del TFM y sus anexos	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■			

Capítulo 2. Marco Teórico

2.1 Fundamentos del sonido musical: frecuencia, armónicos y codificación de notas

El sonido es una onda mecánica que se propaga a través de un medio, normalmente el aire, y se caracteriza por tres atributos físicos fundamentales: frecuencia, amplitud y duración (Rossing, 2019). En el ámbito musical, estos parámetros se traducen respectivamente en tono, volumen e intensidad temporal de una nota. La frecuencia, expresada en hercios (Hz), determina la altura de una nota musical y se asocia directamente con la percepción del tono por parte del oído humano. (Howard, 2017) La amplitud, por su parte, define la intensidad con la que se percibe el sonido y puede medirse como el valor absoluto de la señal en el dominio temporal o mediante su transformada en el dominio de la frecuencia. La duración, medida en segundos, está vinculada al número de muestras que ocupa una señal en una grabación digital, en función de la frecuencia de muestreo del sistema. (Zölzer, 2008)

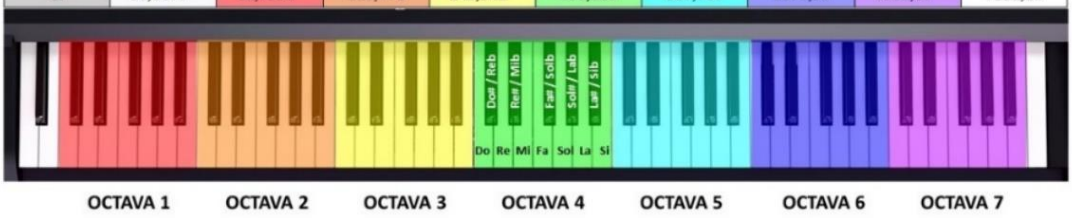
El sistema musical temperado occidental divide cada octava en 12 semitonos equidistantes en una escala logarítmica (Hall, 2002). Esta división implica que las frecuencias de las notas se relacionan entre sí mediante potencias de razón $2^{1/12}$. Dado un tono de referencia, como el La4 (440 Hz), la frecuencia de cualquier otra nota puede calcularse mediante la Ecuación 1 (Pierce, 1983)

$$f_n = f_0 * 2^{\frac{n}{12}}$$

Ecuación 1: Cálculo de la frecuencia de una nota en el sistema temperado occidental

donde n representa el número de semitonos de diferencia respecto a la nota base. Esta relación permite construir un diccionario completo de notas y sus frecuencias correspondientes, que resulta esencial para traducir información espectral en etiquetas musicales comprensibles tal y como se muestra en la siguiente Ilustración 2:

FRECUENCIA DE LAS NOTAS MUSICALES EN HERCIOS (Hz)									
	OCTAVA 0	OCTAVA 1	OCTAVA 2	OCTAVA 3	OCTAVA 4	OCTAVA 5	OCTAVA 6	OCTAVA 7	OCTAVA 8
Do	16,3516	32,7032	65,4064	130,813	261,626	523,251	1046,50	2093,00	4186,01
Do# / Reb	17,3239	34,6479	69,2957	138,591	277,183	554,365	1108,73	2217,46	4434.92
Re	18,3540	36,7081	73,4162	146,832	293,665	587,330	1174,66	2349,32	4698,64
Re# / Mib	19,4454	38,8909	77,7817	155,563	311,127	622,254	1244,51	2489,02	4978,04
Mi	20,6017	41,2035	82,4069	164,814	329,628	659,255	1318,51	2637,02	5274,04
Fa	21,8268	43,6536	87,3071	174,614	349,228	698,456	1396,91	2793,83	5587,66
Fa# / Solb	23,1246	46,2493	92,4986	184,997	369,994	739,989	1479,98	2959,96	5919,92
Sol	24,4997	48,9995	97,9989	195,998	391,995	783,991	1567,98	3135,96	6271,92
Sol# / Lab	25,9565	51,9130	103,826	207,652	415,305	830,609	1661,22	3322,44	6644,88
La	27,5000	55,0000	110,000	220,000	440,000	880,000	1760,00	3520,00	7040,00
La# / Sib	29,1353	58,2705	116,541	233,082	466,164	932,328	1864,66	3729,31	7458,62
Si	30,8677	61,7354	123,471	246,942	493,883	987,767	1975,53	3951,07	7902,14



OCTAVA 1 OCTAVA 2 OCTAVA 3 OCTAVA 4 OCTAVA 5 OCTAVA 6 OCTAVA 7

Ilustración 2: Tabla de frecuencias de notas musicales y su representación en el teclado según la octava (Ciudad Pentagrama, 2020)

Sin embargo, los instrumentos musicales no producen señales puras compuestas por una única frecuencia. En la mayoría de los casos, especialmente en instrumentos acústicos, cada nota genera un conjunto de componentes frecuenciales que incluyen una frecuencia fundamental y varios armónicos (Rossing, 2019). Estos armónicos son múltiplos enteros de la frecuencia fundamental y aportan riqueza tímbrica al sonido, aunque también introducen complejidad en su análisis (Hall, 2002). Si una nota posee una frecuencia fundamental f_0, los armónicos aparecen en las frecuencias $f_k = k * f_0$, donde k ∈ N y k>1. Esto significa que, en el análisis espectral, además de la frecuencia principal, aparecen picos en otras bandas que pueden ser malinterpretados como notas distintas si no se filtran adecuadamente.

Esta presencia de armónicos puede inducir errores significativos en la detección automática de notas (Benetos E.). Por ejemplo, al analizar una grabación en la que se ha tocado un Do3 (aproximadamente 130.81 Hz), podrían observarse picos en 261.62 Hz, 392.43 Hz y 523.24 Hz, que corresponden a sus armónicos. Si el sistema de detección no aplica ningún tipo de filtrado o validación, es posible que asocie esas frecuencias con otras notas distintas, como Do4, Sol4 o Do5. Este problema se acentúa en instrumentos polifónicos, donde múltiples notas y sus armónicos se solapan en el espectro, dificultando aún más la separación de fuentes y la identificación de componentes fundamentales.

Para abordar esta dificultad, resulta imprescindible implementar mecanismos de filtrado de armónicos. Uno de los enfoques más utilizados es establecer un umbral de prominencia, de manera que sólo se consideren relevantes las frecuencias cuya amplitud supere un cierto porcentaje del máximo del espectro (Smith, Spectral Audio Signal Processing, 2011). Otra estrategia consiste en analizar la relación entre las

frecuencias detectadas: si una frecuencia es aproximadamente un múltiplo entero de otra previamente identificada como fundamental, puede clasificarse como armónico y descartarse (Klapuri, 2006). Formalmente, se considera armónica una frecuencia f si existe otra frecuencia $\left|\frac{f}{f_0} - k\right| < \varepsilon$ para algún k ∈ N, siendo ε una tolerancia predefinida. Esta comparación ayuda a mantener únicamente aquellas frecuencias que son verdaderamente representativas de las notas interpretadas.

En el contexto de este proyecto, el filtrado de armónicos se ha abordado mediante el diseño de una máscara espectral adaptativa, basada en funciones de ganancia exponencial parametrizadas por instrumento. Esta máscara refuerza las zonas del espectro donde es más probable encontrar la frecuencia fundamental del instrumento en cuestión, y atenúa aquellas donde predominan armónicos o ruido (Klapuri, 2006). Este preprocesamiento mejora notablemente la precisión del sistema, especialmente en grabaciones reales con variabilidad tímbrica, efectos de reverberación o ruido de fondo.

La correcta identificación de la frecuencia fundamental es una etapa crítica en todo el pipeline de transcripción automática, ya que de ella depende que el sistema pueda asignar etiquetas musicales coherentes a cada segmento de audio (Zölzer, 2008). Una detección errónea en esta fase se propaga a las etapas posteriores, afectando a la calidad de la partitura generada. Por tanto, la comprensión de la estructura armónica del sonido y la aplicación de estrategias eficaces para aislar la componente principal de una señal constituyen uno de los pilares fundamentales de este trabajo (Rossing, 2019).

2.2 Transformada de Fourier y análisis espectral

El análisis de una señal de audio en el dominio de la frecuencia constituye una de las herramientas fundamentales en el reconocimiento automático de contenido musical (Zölzer, 2008). A diferencia del dominio temporal, que describe la evolución de la señal en función del tiempo, el dominio frecuencial permite descomponer la señal en sus componentes armónicas, es decir, en las frecuencias que la conforman y su correspondiente energía o amplitud. Esta descomposición se logra a través de la Transformada de Fourier, que permite representar cualquier señal periódica o segmentada en una suma de senos y cosenos de distintas frecuencias (Oppenheim, 2010).

En el contexto de señales digitales, como las capturadas por un micrófono o cargadas desde un archivo de audio, se trabaja con una versión discreta de la Transformada de Fourier: la Transformada Discreta de Fourier (DFT), cuya implementación computacional eficiente es la Transformada Rápida de Fourier (FFT, por sus siglas en inglés) (Smith, Mathematics of the Discrete Fourier Transform (DFT), 2007). Esta operación transforma una secuencia de N muestras en una representación espectral

de N frecuencias, permitiendo analizar la distribución de energía de la señal a lo largo del espectro. Formalmente, la DFT de una señal discreta x[n], con n=0, 1, ..., N−1, se define como se muestra en la Ecuación 2 (Lyons, 2011):

$$\mathrm{X[k]}=\sum_{n=0}^{N-1} x[n] * e^{-j*2*\pi*k*n/N}, \qquad k = 0,1,\dots N-1$$

Ecuación 2: Definición de la Transformada Discreta de Fourier (DFT)

Cada coeficiente X[k] representa la contribución de la frecuencia $f_k = \frac{k*f_s}{N}$, donde f_s es la frecuencia de muestreo del sistema (Lyons, 2011). El módulo |X[k]| da la amplitud asociada a dicha frecuencia, y el argumento $\arg(|X[k]|)$ representa su fase. En análisis musical, se suele trabajar solo con la magnitud espectral, ya que la fase no tiene relevancia directa para la detección de notas.

La frecuencia de muestreo f_s tiene un papel determinante en el análisis espectral, ya que establece el rango de frecuencias que pueden observarse sin aliasing. Según el teorema de Nyquist, la frecuencia máxima detectable es $f_{max} = \frac{f_s}{2}$ (Oppenheim, 2010). En este proyecto, se han utilizado valores de muestreo como 100 Hz, 150 Hz y 200 Hz no sobre la señal original, sino sobre la tasa de segmentación para aplicar la FFT en tramos temporales. Esto implica dividir el audio en fragmentos de duración fija, normalmente del orden de decenas de milisegundos, sobre los que se calcula su contenido espectral. La elección de esta frecuencia de segmentación tiene un impacto directo en la resolución temporal del sistema, ya que determina cada cuánto tiempo se evalúa el contenido armónico del audio.

Por otro lado, el número de puntos N utilizados en la FFT determina la resolución espectral del análisis (Smith, Spectral Audio Signal Processing, 2011). La separación entre dos frecuencias consecutivas del espectro viene dada por $\Delta f = \frac{f_s}{N}$. Así, un mayor número de puntos proporciona una mayor resolución en el eje de frecuencia, pero implica analizar tramos más largos en el tiempo, lo que reduce la capacidad de detectar variaciones rápidas en la señal. Esta relación inversa entre resolución temporal y resolución frecuencial constituye una de las principales limitaciones estructurales del análisis mediante ventanas fijas (Allen, 1977). En términos prácticos, una ventana corta permite seguir con mayor precisión la evolución temporal de la señal, pero presenta menor precisión en la estimación de las frecuencias presentes. Por el contrario, una ventana larga mejora la resolución en frecuencia, pero difumina los cambios rápidos en el tiempo, tal y como se muestra en la Ilustración 3.

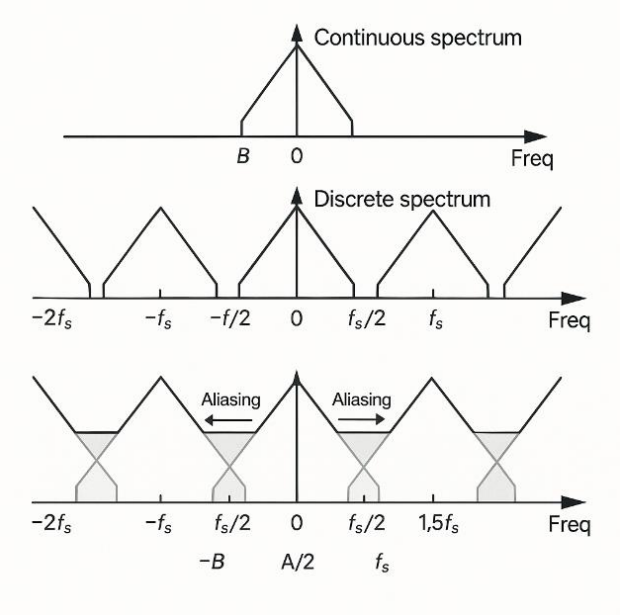


Ilustración 3: Efectos del muestreo en el dominio de la frecuencia

Este fenómeno afecta directamente a la calidad del análisis cuando se aplican técnicas de transcripción musical. Si la frecuencia fundamental de una nota no coincide exactamente con uno de los bins de la FFT, puede producirse un error de cuantización (Lyons, 2011). Este error se manifiesta en que la energía de la señal se reparte entre varios bins cercanos, lo que dificulta la identificación clara de la nota correspondiente. Además, en situaciones reales, la afinación no siempre es perfecta, y las notas pueden estar ligeramente desplazadas respecto a su frecuencia ideal. Para mitigar estos efectos, en este proyecto se ha implementado un sistema de interpolación mediante tramos frecuenciales.

El procedimiento consiste en definir, para cada nota teórica, un intervalo alrededor de su frecuencia central que recoge las desviaciones típicas esperadas en entornos reales. Estos tramos se calculan a partir del diccionario de frecuencias, utilizando interpolación entre los valores adyacentes para establecer márgenes de tolerancia personalizados. De este modo, la suma de la energía dentro del tramo se asocia a la nota correspondiente, compensando posibles imprecisiones en la medición, variaciones en la afinación o efectos derivados del tamaño de la ventana FFT. Esta estrategia es particularmente útil para instrumentos acústicos, donde el contenido espectral no está perfectamente centrado en frecuencias exactas, y la energía puede estar repartida en una región continua del espectro (Rossing, 2019).

Además, se ha contemplado el uso de solapamiento entre ventanas, mediante el parámetro de “overlap” o solape porcentual (Allen, 1977). Esta técnica permite cubrir más densamente el eje temporal sin perder resolución en frecuencia tal y como se muestra en la Ilustración 2: Tabla de frecuencias de notas musicales y su representación en el teclado según la octava. Por ejemplo, un solape del 90 % implica que cada ventana comienza justo después del 10 % final de la anterior. Esta práctica

es habitual en análisis de señales musicales, ya que mejora la suavidad de la evolución temporal y reduce la probabilidad de perder información entre ventanas.

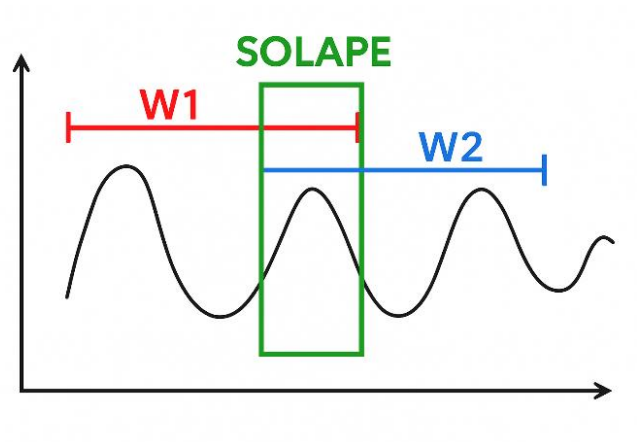


Ilustración 4: Segmentación temporal con solape entre ventanas de análisis

Cabe destacar que todas estas decisiones técnicas afectan directamente a los vectores de características que alimentan a los modelos de aprendizaje automático. Una segmentación deficiente, una baja resolución frecuencial o la ausencia de interpolación pueden generar vectores incompletos o erróneos, afectando negativamente la clasificación de notas (Benetos E.). Por el contrario, una correcta parametrización del análisis espectral permite capturar de forma precisa la información relevante de la señal y facilita que el sistema de reconocimiento funcione con mayor precisión y robustez (Smith, Spectral Audio Signal Processing, 2011).

En definitiva, el análisis en frecuencia mediante la Transformada de Fourier, junto con una configuración adecuada de parámetros como la frecuencia de muestreo, la longitud de las ventanas, el solape entre segmentos y la interpolación espectral, constituye un pilar esencial del sistema desarrollado. Estas herramientas permiten traducir una señal compleja y dinámica en un conjunto estructurado de datos que puede ser interpretado y procesado por los algoritmos de detección de notas, acercando la representación digital del sonido a su interpretación musical real (Oppenheim, 2010).

2.3 Procesamiento digital de señales aplicado a la transcripción musical

El procesamiento digital de señales (DSP, por sus siglas en inglés) constituye la base técnica sobre la que se construyen los sistemas modernos de análisis de audio (Oppenheim, 2010). Su objetivo principal es transformar señales analógicas captadas por un micrófono o almacenadas en archivos digitales en representaciones manipulables, comprensibles y estructuradas que puedan ser interpretadas por algoritmos computacionales. En el caso específico del análisis musical, el DSP permite detectar patrones frecuenciales asociados a notas, modelar la evolución

temporal de la señal y facilitar la clasificación de eventos musicales mediante técnicas automáticas.

El primer paso fundamental en cualquier sistema de DSP aplicado al audio es la **digitalización de la señal**, que implica dos operaciones clave: el **muestreo** y la **cuantificación**. El muestreo consiste en tomar lecturas periódicas de la amplitud de la señal analógica a una frecuencia determinada, conocida como frecuencia de muestreo f_s (Lyons, 2011). Esta frecuencia debe ser al menos el doble de la frecuencia más alta presente en la señal, de acuerdo con el teorema de Nyquist, para evitar aliasing (Proakis, 2007). Por ejemplo, si se desea capturar información hasta los 4.000 Hz, la frecuencia de muestreo debe ser como mínimo de 8000 Hz, aunque en la práctica se utilizan valores mucho mayores (como 44100 Hz o 48000 Hz) para garantizar fidelidad. En este proyecto, la señal original puede partir de frecuencias estándar, pero posteriormente se somete a una segmentación y análisis con tasas de muestreo adaptadas, como 100 Hz o 200 Hz, con el objetivo de aplicar la FFT de forma controlada sobre fragmentos temporales.

Tras el muestreo, la señal se cuantifica, es decir, se asigna a cada muestra un valor numérico discreto, generalmente en una escala de 16 o 32 bits, lo que limita la precisión pero permite almacenar y procesar digitalmente el audio (Zölzer, 2008). Una vez digitalizada, la señal puede tratarse mediante diversas técnicas para mejorar su calidad, su representación o su relevancia para la tarea concreta que se desea resolver.

Uno de los procesos más importantes es la **segmentación temporal**, que consiste en dividir la señal continua en ventanas o tramos de duración fija. En el sistema desarrollado, esta operación se realiza para aplicar posteriormente la transformada de Fourier y analizar el contenido frecuencial de cada fragmento. La duración del segmento y el grado de solapamiento entre ventanas (por ejemplo, un 90 %) son parámetros críticos que determinan la resolución temporal del análisis y la suavidad de la transcripción. Una segmentación muy gruesa puede pasar por alto cambios rápidos en la nota, mientras que una segmentación demasiado fina puede generar ruido o variabilidad excesiva (Allen, 1977).

Además de segmentar, es necesario **normalizar la señal** antes de su análisis (Müller, 2015). La normalización consiste en ajustar la amplitud de la señal para que se mantenga dentro de un rango controlado, lo que permite comparar grabaciones realizadas con distintas intensidades o condiciones. Esta operación es especialmente útil cuando se utilizan algoritmos de aprendizaje automático, ya que evita que las diferencias de escala afecten a la clasificación. La normalización puede realizarse dividiendo por el valor absoluto máximo, por la energía total de la señal o mediante métodos estadísticos como la estandarización (z-score), en función del tipo de modelo y de los datos (Bishop, Pattern Recognition and Machine Learning, 2006).

Otro aspecto clave del preprocesamiento es la **limpieza del ruido**. Las grabaciones de audio, especialmente en entornos no controlados, pueden contener componentes

no deseados que distorsionan el análisis: ruido ambiental, vibraciones, reverberaciones, zumbidos eléctricos, entre otros. Para minimizar su impacto, se aplican técnicas de filtrado, tanto en el dominio temporal como en el frecuencial (Smith, Spectral Audio Signal Processing, 2011). Entre las más comunes se encuentran los filtros pasa banda, que eliminan las frecuencias fuera del rango útil del instrumento analizado. Por ejemplo, para un análisis de flauta se puede limitar el análisis entre 250 Hz y 2100 Hz, ignorando tanto los graves como los agudos fuera del rango del instrumento. Esto permite reducir la cantidad de ruido procesado y aumentar la fiabilidad del sistema.

En este proyecto, además de los filtros clásicos, se ha desarrollado un **módulo de filtrado adaptativo**, cuya función es aplicar una máscara espectral personalizada por instrumento. Este filtro se genera dinámicamente mediante funciones exponenciales que actúan como perfil de ganancia sobre el eje de frecuencia, reforzando las zonas donde se espera encontrar la información musical más relevante y atenuando aquellas menos útiles o más propensas al ruido (Slaney, 1998). Esta técnica se inspira en conceptos de enmascaramiento auditivo y tiene como objetivo destacar las componentes principales de la señal, mejorando la relación señal-ruido desde un punto de vista computacional.

Otro procedimiento importante en DSP aplicado a transcripción musical es la detección de picos espectrales, que permite localizar las frecuencias dominantes dentro de un segmento (Smith, Spectral Audio Signal Processing, 2011). Para ello se emplean algoritmos que identifican máximos locales en la magnitud de la transformada de Fourier, considerando umbrales de altura, prominencia y separación entre picos. Este paso resulta esencial para distinguir las frecuencias fundamentales de los armónicos, y para alimentar correctamente los clasificadores que asignan notas musicales a partir de vectores de características (Lyons, 2011).

Por último, se deben considerar las **limitaciones introducidas por el entorno de grabación**, como la latencia, la calidad del micrófono, la distancia del instrumento a la fuente de captura o la presencia de eco (Bello, 2005). Si bien estas condiciones no pueden eliminarse completamente desde el procesamiento digital, el sistema puede ser diseñado de forma robusta a través de un conjunto de decisiones acumulativas: ventanas suficientemente densas en el tiempo, solapamiento alto, uso de tramos de frecuencia interpolados, y modelos de clasificación entrenados con ejemplos representativos de grabaciones reales.

En conjunto, el procesamiento digital de señales proporciona el marco técnico necesario para transformar una señal de audio en bruto en una representación estructurada y útil para el análisis musical automático. Cada paso —desde la digitalización inicial hasta la limpieza de ruido y extracción de picos— contribuye a construir una representación más precisa del contenido musical. Estas técnicas no son solo complementarias, sino que forman una cadena de operaciones interdependientes que permiten que el sistema desarrollado pueda interpretar,

analizar y transcribir una interpretación musical en condiciones diversas, manteniendo una precisión razonable sin requerir equipamiento profesional o entornos de grabación controlados (Müller, 2015).

2.4 Modelos de aprendizaje automático para clasificación de notas musicales

El reconocimiento automático de notas musicales a partir de señales de audio requiere un componente de decisión capaz de transformar las características extraídas de la señal en etiquetas musicales concretas. Para esta tarea, se emplean modelos de aprendizaje automático supervisado, entrenados previamente con ejemplos en los que la entrada (vector de características del audio) está asociada a una salida conocida (la nota tocada) (Bishop, Pattern Recognition and Machine Learning, 2006). Una vez entrenados, estos modelos pueden generalizar su conocimiento y clasificar correctamente nuevas señales. En este proyecto se han utilizado tres enfoques principales: el algoritmo K-Nearest Neighbors (KNN), el Perceptrón Multicapa (MLP) y las Redes Neuronales Recurrentes (RNN), cada uno con propiedades y ventajas particulares según el tipo de instrumento y el contexto del análisis (Humphrey, 2012).

En primer lugar, el algoritmo **K-Nearest Neighbors (KNN)** es una técnica sencilla pero eficaz basada en la similitud entre muestras (Duda, 2001). Dado un vector de entrada, el modelo busca los K ejemplos más cercanos en el conjunto de entrenamiento según una métrica de distancia, normalmente la distancia euclídea. La predicción se obtiene por mayoría de votos entre las etiquetas de estos vecinos. Este método no requiere una fase explícita de entrenamiento, más allá del almacenamiento del conjunto de datos. Su fortaleza reside en su capacidad de adaptación directa a los datos observados, y su comportamiento es robusto cuando los vectores están bien definidos y normalizados. En este proyecto, los vectores de entrada son arrays de 108 componentes, que representan la suma de amplitudes espectrales asociadas a cada una de las 108 notas musicales definidas en el diccionario de frecuencias. Estos vectores resumen la energía contenida en los tramos asignados a cada nota, y permiten comparar perfiles espectrales entre muestras diferentes.

El algoritmo KNN resulta especialmente útil en instrumentos como la guitarra, donde los patrones de amplitud que definen cada nota o acorde tienen estructuras bien diferenciadas en el espacio de características. Al no requerir un modelo paramétrico, KNN es rápido de aplicar, interpretable, y ofrece buenos resultados si se dispone de un conjunto representativo de ejemplos (Géron, 2019). Sin embargo, su rendimiento decrece en presencia de ruido, si el espacio de características es muy grande o si las clases no están bien separadas. Además, su complejidad computacional durante la predicción es proporcional al tamaño del conjunto de entrenamiento, lo que limita su escalabilidad en entornos con muchos datos.

Por otro lado, el **Perceptrón Multicapa (MLP)** es un tipo de red neuronal artificial que aprende funciones de decisión no lineales mediante capas de neuronas interconectadas (Goodfellow, 2016). Cada neurona recibe una combinación lineal de sus entradas, le aplica una función de activación (como ReLU o sigmoide), y propaga el resultado a la siguiente capa. El modelo se entrena ajustando los pesos de las conexiones mediante el algoritmo de retropropagación del error, con optimización mediante descenso del gradiente (Rumelhart, 1986). En su configuración básica, un MLP consta de una capa de entrada, una o más capas ocultas, y una capa de salida con tantas neuronas como clases posibles.

En el caso del sistema desarrollado, el MLP toma como entrada los mismos vectores de 108 características, y emite como salida la predicción de la nota correspondiente. A diferencia de KNN, el MLP puede aprender representaciones internas y relaciones complejas entre los valores de entrada, permitiendo manejar variaciones de timbre, ruidos leves o afinaciones imperfectas (Géron, 2019). Su entrenamiento es más costoso, pero una vez entrenado, realiza predicciones de forma rápida y eficiente. Este modelo es especialmente adecuado cuando se dispone de un conjunto moderadamente grande de datos anotados y se desea un clasificador capaz de capturar relaciones no triviales. En instrumentos polifónicos como la guitarra, el MLP puede detectar combinaciones de amplitudes características de cada cuerda y posición, lo que mejora la precisión respecto a métodos más simples.

El tercer modelo implementado es la **Red Neuronal Recurrente (RNN)**, diseñada para el tratamiento de datos secuenciales (Goodfellow, 2016). A diferencia del MLP, que trata cada vector de forma independiente, la RNN incorpora bucles internos que le permiten conservar memoria del estado anterior, es decir, tener en cuenta el contexto temporal de las entradas. Esto es especialmente útil en instrumentos melódicos como la flauta, donde la sucesión de notas forma una secuencia lógica, y la transición entre notas consecutivas puede aportar información adicional sobre la nota actual (Sigtia, 2016). En el presente proyecto, la RNN se ha utilizado en una configuración simple de una capa recurrente con unidades de tipo LSTM o GRU, seguidas de una capa de salida completamente conectada (Chung, 2014). La entrada al modelo consiste en secuencias de vectores de características, donde cada vector representa un segmento temporal del audio. La red analiza estas secuencias para emitir la predicción de la nota presente en cada instante.

Las RNN son especialmente potentes para modelar dinámicas temporales, y pueden aprender a distinguir entre ataques de nota, sostenidos y transiciones suaves, lo que es clave para generar partituras legibles y coherentes. Sin embargo, requieren un mayor esfuerzo de entrenamiento, una cuidadosa preparación de los datos secuenciales y técnicas adicionales como regularización o normalización para evitar el sobreajuste (Géron, 2019). En este proyecto, se ha entrenado el modelo con ejemplos segmentados y etiquetados cuidadosamente, con secuencias cortas pero representativas de fragmentos reales de interpretación musical.

Todos estos modelos comparten un flujo de trabajo común: a partir del audio original se extraen características espectrales por tramos de tiempo, estas se agrupan en vectores que resumen la información musical contenida en la señal, y estos vectores se utilizan como entrada al clasificador. La salida es una etiqueta correspondiente a la nota más probable en ese tramo. Esta predicción puede usarse directamente para construir una partitura en tiempo real o para analizar patrones musicales posteriores. La elección del modelo depende del tipo de instrumento, la calidad del audio, la cantidad de datos de entrenamiento y las restricciones de tiempo o recursos computacionales.

La combinación de técnicas de aprendizaje automático como KNN, MLP y RNN permite abordar de forma flexible el problema de la clasificación de notas musicales en distintos contextos. Cada modelo aporta ventajas específicas: KNN ofrece simplicidad y resultados razonables con pocos datos, MLP permite aprender relaciones no lineales con una buena velocidad de inferencia, y RNN captura dependencias temporales útiles para la interpretación secuencial del sonido. Esta variedad metodológica ha sido clave en el desarrollo del sistema de transcripción propuesto, adaptándose a las características particulares de los instrumentos y escenarios de análisis tratados en este trabajo (Humphrey, 2012).

Capítulo 3. Estado del Arte

3.1 Reconocimiento de notas musicales

El reconocimiento automático de notas musicales a partir de señales de audio ha sido una de las áreas más activas y desafiantes en el campo del procesamiento digital de señales y la inteligencia artificial aplicada a la música (Müller, 2015). Esta tarea, conocida también como *note onset detection* o *note transcription*, consiste en identificar las notas individuales tocadas en una grabación, determinando su frecuencia, duración y momento de aparición. Aunque la percepción humana lo realiza con relativa facilidad, su implementación computacional presenta múltiples dificultades derivadas de la complejidad de las señales reales, los timbres variados y la presencia de ruido o superposición armónica.

Los enfoques tradicionales se han basado en el análisis espectral mediante la Transformada de Fourier o el uso de bancos de filtros, combinados con reglas heurísticas para detectar picos de energía en determinadas bandas de frecuencia (Boulanger-Lewandowski, 2012). En estos sistemas, la detección de notas se realiza frecuentemente a partir del seguimiento de las frecuencias fundamentales estimadas en el espectrograma, aplicando algoritmos de *peak picking*, estimación de tono (*pitch estimation*) y detección de eventos transitorios (*onset detection*). Si bien estos métodos pueden funcionar razonablemente bien en grabaciones limpias y monofónicas, su rendimiento se degrada significativamente ante ruido, afinaciones no estándar o polifonía.

Con el auge del aprendizaje automático, especialmente a partir de la década de 2010, comenzaron a desarrollarse sistemas de reconocimiento de notas basados en clasificadores entrenados sobre grandes conjuntos de datos musicales. En estos sistemas, en lugar de buscar reglas fijas, se extraen vectores de características de la señal (por ejemplo, espectros, MFCCs, cromas, etc.) y se entrenan modelos como máquinas de vectores soporte (SVM), k-vecinos más cercanos (KNN) o redes neuronales para predecir directamente la nota más probable (Bishop, Pattern Recognition and Machine Learning, 2006). Esta estrategia ha demostrado ser más robusta ante variabilidad tonal, ruidos o diferencias tímbricas, aunque su efectividad depende en gran medida de la calidad y diversidad del conjunto de datos de entrenamiento.

En los últimos años, los sistemas más avanzados han incorporado modelos de redes neuronales profundas, incluyendo arquitecturas convolucionales (CNN), recurrentes (RNN, LSTM) y, más recientemente, transformadores (Hawthorne, 2017). Estas redes permiten capturar tanto patrones espaciales (en el espectrograma) como temporales (en la sucesión de notas), lo que ha permitido alcanzar niveles de precisión cercanos a la percepción humana en ciertos contextos (Goodfellow, 2016). Herramientas como Onsets and Frames (Hawthorne et al., 2017) han combinado CNNs para detección de

onsets con LSTMs para el modelado de eventos musicales en el tiempo, logrando transcripciones precisas en piano solo. Sin embargo, estos modelos suelen requerir una gran cantidad de datos anotados, lo que limita su aplicabilidad en instrumentos menos comunes o con estilos interpretativos no estandarizados (Benetos E. D., 2013).

En cuanto a las herramientas comerciales, softwares como **Melodyne**, **AnthemScore** o **Ableton Live** incorporan módulos de reconocimiento de notas que permiten editar o visualizar la transcripción de una grabación (Celemony Software GmbH) (Lunaverus) (Ableton). No obstante, muchos de estos sistemas están optimizados para voces solistas, instrumentos melódicos o contextos musicales muy concretos, y suelen presentar limitaciones al enfrentarse a polifonía densa, audio con ruido de fondo o técnicas instrumentales no convencionales.

A nivel de investigación, han surgido también proyectos de código abierto como **Sonic Visualiser**, que permiten visualizar manualmente el contenido espectral del audio y explorar posibles transcripciones, aunque no están diseñados para ofrecer una solución automática al reconocimiento de notas (Centre for Digital Music, Queen Mary University of London).

3.2 Clasificadores en análisis musical

El uso de clasificadores automáticos en el análisis musical ha cobrado un papel central en las últimas dos décadas, especialmente en tareas como el reconocimiento de notas, identificación de instrumentos, clasificación de géneros o análisis de acordes (Benetos E. D., 2013). Estas técnicas permiten transformar señales complejas y multivariadas en categorías discretas que pueden interpretarse musicalmente, como una nota individual, una clase de instrumento o una etiqueta de estilo (Müller, 2015).

En las primeras etapas del desarrollo de estos sistemas, los clasificadores se aplicaban sobre vectores de características extraídos manualmente de la señal de audio. Estas características podían ser espectrales (como los coeficientes cepstrales MFCC, la energía por banda, el centroide espectral), temporales (como la envolvente de amplitud o el ritmo), o derivadas de representaciones perceptuales como el cromagrama. Una vez obtenido el vector, se entrenaban clasificadores clásicos como *k*-vecinos más cercanos (KNN), árboles de decisión o máquinas de vectores soporte (SVM), con buenos resultados en tareas acotadas y con conjuntos de datos bien definidos.

El algoritmo KNN, por ejemplo, ha sido ampliamente utilizado por su simplicidad y efectividad cuando las clases están bien separadas en el espacio de características. En el contexto del reconocimiento de notas, se emplea para asignar una nota a partir de la similitud entre el espectro de entrada y los espectros previamente etiquetados

(Duda, 2001). Aunque su precisión puede ser razonable, presenta limitaciones en cuanto a escalabilidad y sensibilidad al ruido.

Posteriormente, la introducción de modelos más flexibles como los **perceptrones multicapa (MLP)** supuso una mejora sustancial, al permitir modelar relaciones no lineales entre características y salidas. Los MLP han sido aplicados con éxito en tareas como la clasificación de acordes, identificación de notas simultáneas y detección de patrones rítmicos (Goodfellow, 2016). Su principal ventaja reside en su capacidad de generalización y aprendizaje de representaciones internas útiles, incluso a partir de entradas ruidosas o parcialmente incompletas.

El avance más significativo en la última década ha sido la adopción de **redes neuronales profundas** en el análisis musical. Entre ellas, destacan las **redes convolucionales (CNN)**, empleadas principalmente para procesar espectrogramas como si fuesen imágenes, capturando estructuras espaciales y patrones locales característicos de diferentes notas, timbres o texturas musicales. Las CNN han demostrado ser muy eficaces en la clasificación de géneros musicales, reconocimiento de instrumentos y detección de onsets (Choi, 2017).

En paralelo, las **redes neuronales recurrentes (RNN)**, y especialmente sus variantes LSTM (Long Short-Term Memory) y GRU (Gated Recurrent Unit), se han consolidado como herramientas fundamentales para modelar secuencias temporales (Sigtia, 2016). Estas redes permiten capturar dependencias a lo largo del tiempo, lo que es especialmente relevante en música, donde la sucesión de eventos tiene una estructura inherente. Se han aplicado con éxito en sistemas de transcripción de piano, generación de melodías, y detección de notas en instrumentos melódicos, permitiendo una mejor detección de transiciones, articulaciones y ataques suaves.

Modelos más recientes como las **redes Transformer** también están comenzando a explorarse en el ámbito del análisis musical, especialmente en tareas de modelado generativo y transcripción compleja, aunque su uso todavía está limitado por la necesidad de grandes volúmenes de datos y recursos computacionales elevados (Huang, 2019).

En cuanto a las aplicaciones prácticas, diversos estudios han empleado combinaciones de clasificadores para abordar tareas musicales complejas (Hawthorne, 2017). Por ejemplo, algunos sistemas emplean una CNN para detectar el espectro de notas activas, seguida de una RNN para refinar la secuencia temporal. Otros combinan modelos tradicionales con filtros heurísticos, como técnicas de *post-processing* para eliminar errores comunes en la predicción (como armónicos mal clasificados o notas duplicadas).

Una limitación común en muchos estudios previos es la dependencia de grandes datasets anotados, como MAPS (para piano) o NSynth (para notas aisladas), lo cual restringe la aplicabilidad de los modelos en escenarios con datos limitados o con

instrumentos no representados en dichos conjuntos (Google Brain / Magenta, 2017). Esto motiva enfoques como el de este proyecto, que se apoyan en modelos más ligeros como KNN y MLP, entrenados con ejemplos propios y adaptados a las características específicas del instrumento analizado.

3.3 Extracción de características espectrales

La extracción de características espectrales es una etapa clave en los sistemas de análisis musical automático. Su objetivo es traducir una señal de audio, en su forma bruta y difícilmente interpretable, en un conjunto de variables representativas que capturan la información más relevante del contenido musical. Estas variables —conocidas como *features*— permiten alimentar clasificadores automáticos y modelar patrones musicales mediante algoritmos de aprendizaje.

El punto de partida para la mayoría de estas técnicas es el análisis espectral de la señal mediante la **Transformada Rápida de Fourier (FFT)** (Smith, Spectral Audio Signal Processing, 2011). Esta operación transforma una señal en el dominio temporal en una representación frecuencial, revelando la distribución de energía a lo largo del espectro. En el contexto musical, esta información permite identificar las frecuencias dominantes presentes en cada segmento de audio, muchas de las cuales están asociadas a notas musicales. Sin embargo, la FFT por sí sola genera un volumen de datos excesivo, difícil de utilizar directamente en modelos de clasificación. Por tanto, es necesario reducir y estructurar esa información en vectores de características compactos, robustos y relevantes.

Uno de los enfoques clásicos es la utilización de **ventanas temporales móviles** para segmentar el audio en bloques cortos, sobre los cuales se calcula la FFT. Cada ventana genera un espectro, y el análisis de estas ventanas en secuencia produce una matriz tiempo-frecuencia conocida como **espectrograma**. A partir de este espectrograma, pueden extraerse distintas medidas espectrales. Algunas de las más utilizadas en tareas de clasificación musical son:

- **Centroides espectrales**, que indican la “media ponderada” de las frecuencias activas, asociadas a la percepción del brillo del sonido.
- **Bandas de energía**, que cuantifican la distribución de energía en diferentes rangos frecuenciales, útiles para distinguir timbres e instrumentos.
- **Chroma features**, que agrupan la energía del espectro según las doce clases de notas (Do, Do#, Re…), ignorando la octava, lo que las hace útiles para tareas como reconocimiento de acordes (Fujishima, 1999).
- **Coeficientes cepstrales en las frecuencias de Mel (MFCCs)**, ampliamente usados en reconocimiento de voz, pero también adaptados al análisis musical (Rabiner, 2011).

No obstante, en el caso específico de este proyecto, se ha optado por una estrategia de extracción directa basada en un **diccionario de notas y frecuencias** generado programáticamente. Este diccionario permite definir con precisión los intervalos de frecuencia asociados a cada una de las 108 notas utilizadas en la clasificación. Para

cada nota, se calcula un tramo de frecuencia centrado en su frecuencia fundamental y ampliado mediante interpolación controlada, con el objetivo de capturar energía realista incluso si la nota no está perfectamente afinada.
La característica principal extraída en este sistema es la **suma de amplitudes espectrales en cada tramo de nota**. Este vector representa cuánta energía hay en cada nota posible, y sirve como entrada para los modelos clasificadores (KNN, MLP o RNN). Esta técnica de extracción está inspirada en el concepto de *pitch salience*, pero adaptada a un sistema entrenable y con bajo coste computacional (Gómez, 2006).
Una de las ventajas de esta representación es su interpretabilidad directa: si una componente del vector tiene un valor muy alto, se puede asumir que esa nota está activa. Además, esta estrategia permite operar tanto en contextos melódicos (donde solo una nota está activa a la vez) como en polifónicos (donde varias notas pueden estarlo simultáneamente), simplemente analizando los valores relativos entre componentes.
Otro aspecto importante en la extracción de características es la **robustez frente al ruido y a las variaciones tímbricas**. Por eso, además del cálculo de amplitudes, se ha aplicado previamente un **filtrado espectral adaptativo**, que atenúa las frecuencias menos relevantes para el instrumento en cuestión y refuerza aquellas en las que es más probable que aparezca la señal útil. Este filtrado se implementa mediante funciones exponenciales ajustadas a cada instrumento, y mejora la relación señal-ruido en los vectores resultantes.
En investigaciones recientes, se han propuesto enfoques más sofisticados para la extracción de características, como redes convolucionales aplicadas directamente al espectrograma o autoencoders que aprenden representaciones compactas de forma no supervisada (Lee, 2009). Si bien estas técnicas ofrecen ventajas en escenarios con grandes volúmenes de datos, en proyectos como el presente, donde el enfoque es modular, interpretable y específico por instrumento, la extracción de características espectrales manuales y guiadas sigue siendo una estrategia eficaz y flexible.

3.4 Proyectos similares y soluciones existentes

En la última década han emergido múltiples herramientas, tanto comerciales como de código abierto, orientadas a la transcripción musical automática, es decir, a la conversión de grabaciones de audio en partituras legibles. Aunque estos sistemas han mejorado notablemente, siguen presentando limitaciones que justifican el desarrollo de nuevas soluciones adaptadas a contextos específicos, como el abordado en este proyecto.

Entre las soluciones comerciales más conocidas se encuentra **Melodyne**, un software ampliamente utilizado en entornos profesionales de edición musical (Celemony Software GmbH). Melodyne ofrece análisis en profundidad de tono y tiempo, con

capacidad para modificar notas individualmente dentro de una grabación. Aunque es muy preciso en entornos controlados y con señales monofónicas o levemente polifónicas, requiere una señal muy limpia para funcionar bien y no está orientado a generar una partitura tradicional, sino más bien a editar audio de forma musical.

Otra herramienta relevante es **AnthemScore**, que tiene como objetivo principal generar automáticamente partituras a partir de archivos de audio (Lunaverus). Su enfoque está basado en aprendizaje automático y está optimizado para instrumentos melódicos. Si bien proporciona resultados razonablemente precisos en estos casos, su rendimiento disminuye en entornos polifónicos, como el de una guitarra tocando acordes, y no permite adaptar el sistema a instrumentos concretos ni ajustar la estrategia de filtrado o segmentación.

En el ámbito del software libre destaca **Sonic Visualiser**, una plataforma de visualización de audio muy útil para análisis espectrales detallados, pero que no ofrece una solución completa de transcripción musical automática. Requiere una intervención manual significativa para analizar o interpretar los datos espectrales, por lo que se orienta más a la investigación o la docencia que a la producción de partituras automáticas (Centre for Digital Music, Queen Mary University of London).

En cuanto a proyectos de investigación, uno de los más avanzados es **Onsets and Frames**, desarrollado por Google Magenta. Utiliza redes neuronales profundas (CNN + LSTM) para detectar onsets y mantener notas sostenidas, y ha mostrado un rendimiento muy alto en el dominio del piano. Sin embargo, está entrenado específicamente para ese instrumento, en condiciones de grabación muy controladas y con grandes volúmenes de datos, lo que limita su aplicabilidad a otros instrumentos como flauta o guitarra sin retraining (Hawthorne, 2017).

Estas herramientas, si bien representan avances notables en la transcripción musical automática, no cubren adecuadamente todas las necesidades presentes en contextos reales. Algunas no permiten trabajar con grabaciones hechas en condiciones no profesionales, otras no son adaptables a instrumentos específicos, y muchas requieren una curva de aprendizaje alta o recursos computacionales significativos. Además, pocas de ellas ofrecen una interfaz sencilla para el usuario final ni control sobre aspectos técnicos como el filtrado en frecuencia, la frecuencia de muestreo o la segmentación temporal.

Frente a estas limitaciones, el sistema desarrollado en este proyecto ofrece una solución modular, adaptable y extensible, que permite analizar grabaciones de instrumentos concretos (flauta y guitarra), aplicar estrategias de preprocesamiento personalizadas, y seleccionar entre distintos modelos de clasificación según el contexto. Además, proporciona una salida en forma de partitura coherente y legible, y puede ejecutarse sin necesidad de grandes volúmenes de datos ni hardware especializado.

En la Tabla 1 se observa una comparativa entre algunas de las soluciones más representativas en el ámbito de la transcripción musical automática, destacando qué funcionalidades ofrecen y cuáles siguen siendo necesidades no cubiertas, especialmente en lo que respecta a adaptabilidad por instrumento, facilidad de uso y robustez frente a grabaciones no profesionales.

Herramienta	Transcripción automática	Inst. melódicos	Inst. polifónicos	Filtrado adaptativo	Selección modelo	Interfaz intuitiva	Ajuste por instrumento	Requiere datos grandes	Salida en partitura
Melodyne	✔ / ✘	✔	✘	✘	✘	✔	✘	✘	✘
AnthemScore	✔	✔	✘	✘	✘	✔	✘	✘	✔
Sonic Visualiser	✘	✘	✘	✔ (manual)	✘	✘	✘	✘	✘
Onsets and Frames	✔	✔	✘	✘	✘	✘	✘	✔	✔
Este proyecto	✔	✔	✔	✔	✔	✔	✔	✘	✔

Tabla 1: Resumen de las características de las soluciones actuales

Capítulo 4. Descripción de las Tecnologías

Para el desarrollo del sistema de transcripción musical automática, se ha optado por el uso del lenguaje de programación **Python** como núcleo tecnológico (Foundation, Python). Esta elección ha estado motivada por múltiples razones:

- Versatilidad y legibilidad, ya que Python se caracteriza por una sintaxis clara y orientada a objetos, lo que ha facilitado el diseño modular del sistema y ha mejorado su mantenibilidad.
- Ecosistema científico consolidado, que dispone de un conjunto muy amplio de librerías especializadas en ciencia de datos, tratamiento de señales digitales, aprendizaje automático y visualización (Developers S.)
- Comunidad activa y soporte extenso, al tratarse de una herramienta estándar en proyectos de inteligencia artificial y procesamiento de audio, lo que garantiza documentación exhaustiva, foros de ayuda y múltiples ejemplos prácticos.

Gracias a estas cualidades, Python ha permitido integrar de forma eficiente todas las fases del sistema: desde la lectura y preprocesamiento de los audios, hasta el análisis en frecuencia, la predicción de las notas mediante modelos entrenados y su visualización en forma de partitura (Géron, 2019).

La interfaz y ejecución del sistema se han centralizado mediante una aplicación web desarrollada con **Flask**, un microframework que permite construir aplicaciones ligeras en Python (Projects). Su implementación ha permitido encapsular el sistema completo en un entorno accesible al usuario, quien puede cargar un archivo de audio, seleccionar el instrumento correspondiente, aplicar filtros si lo desea, y obtener como resultado la partitura generada a partir del análisis. Flask ha facilitado la separación clara entre la lógica de servidor y la visualización desde el navegador, permitiendo una gestión ordenada de funciones como la segmentación del audio, la carga de modelos y la predicción de notas. Además, su diseño modular y su compatibilidad con servidores como Gunicorn o uWSGI han simplificado el despliegue, haciendo posible su ejecución tanto en entornos locales como en servidores remotos (Team) (Unbit).

El sistema ha sido diseñado con una arquitectura modular basada en el uso de librerías especializadas. A continuación, se detallan las principales y su papel dentro del sistema:

a) Librerías para procesamiento de audio

- **Librosa**: Es la base del sistema para cargar y manipular archivos de audio. Ha sido utilizada para calcular espectrogramas, aplicar transformadas de Fourier y extraer características como las frecuencias dominantes en cada segmentolecturaAudios (McFee).

- **MoviePy**: Ha permitido convertir archivos de distintos formatos (MP3, MP4, WAV) en un formato estándar que puede ser procesado internamente por el sistemalecturaAudios (Zulko).

- **Soundfile y Pydub** (mencionadas en el Anexo B): Complementan el tratamiento del audio, permitiendo tareas de corte, unión y modificación de archivos para experimentos o depuración (Developers S.) (Riehl).

b) Librerías científicas y matemáticas

- **NumPy y SciPy**: Han sido esenciales para realizar transformadas rápidas de Fourier (FFT), filtrar frecuencias, realizar interpolaciones y manipular arrays de datos sonoros con alto rendimientolecturaAudiosmakeFiltro (Developers N. , numpy.org) (Developers S.).

- **Pandas**: Ha sido clave en el manejo de datos estructurados, como la exportación de filtros o resultados a hojas Excel y la organización de los datos de entrada y salida del sistemacargarFiltro (pandas-dev).

c) Librerías de visualización

- **Matplotlib**: Utilizada para graficar espectros y curvas exponenciales en el diseño de filtros personalizados. Proporciona una base sólida para representar funciones de atenuación y curvas de amplitudmakeFiltro (Hunter).

- **Plotly**: Ha permitido representar gráficamente y de forma interactiva la frecuencia y la amplitud de cada segmento del audio. Esto ha sido útil no solo para visualizar los resultados, sino también para validar manualmente la calidad de los datos procesadoslecturaAudios (Plotly Technologies Inc.).

d) Librerías de aprendizaje automático

- **TensorFlow y Keras**: Utilizadas para cargar y ejecutar modelos de redes neuronales recurrentes (RNN), aplicados a la detección secuencial de notas en instrumentos melódicos como la flautafuncionesModelos (Google Brain Team).

- **Scikit-learn**: Fundamental para implementar y entrenar modelos de tipo KNN y MLP, especialmente utilizados en el reconocimiento de notas en instrumentos polifónicos como la guitarrafuncionesModelos (Pedregosa).

- **Joblib y Pickle**: Utilizadas para guardar, cargar y reutilizar los modelos entrenados, así como para gestionar los objetos de codificación de etiquetas en el caso de las redes neuronalesfuncionesModelos (Joblib Developers) (Foundation, Python pickle module).

En conjunto, esta infraestructura tecnológica ha permitido construir un sistema robusto, eficiente y extensible, capaz de adaptarse tanto a instrumentos melódicos como polifónicos y de integrar funcionalidades avanzadas como la aplicación de filtros personalizados en el dominio de la frecuencia.

La combinación de estas herramientas ha permitido desarrollar un sistema robusto, flexible y extensible, capaz de adaptarse tanto a instrumentos melódicos como polifónicos, incorporando funcionalidades avanzadas como la aplicación de filtros personalizados en el dominio de la frecuencia, y garantizando una experiencia de uso sencilla y accesible a través de la interfaz web.

Capítulo 5. Arquitectura de la Aplicación

La arquitectura del sistema desarrollado está concebida bajo una estructura **modular, escalable y flexible**, lo que permite abordar de forma ordenada y eficiente todas las etapas del procesamiento musical desde un archivo de audio hasta la generación final de una partitura interactiva. Este diseño modular favorece la separación de responsabilidades, facilita la integración de nuevos instrumentos o algoritmos, y permite ajustar fácilmente distintos parámetros del flujo según el contexto o el tipo de señal sonora que se desea analizar.

El sistema está pensado para funcionar con **instrumentos melódicos** (como la flauta travesera, que produce una única nota a la vez) y **polifónicos** (como la guitarra, que puede producir acordes con múltiples notas simultáneas). Para ello, se han implementado diferentes rutas de procesamiento y clasificación de notas que se activan según el instrumento seleccionado por el usuario. El motor central del sistema es compatible con distintos tipos de modelos de machine learning (KNN, MLP, RNN), lo que proporciona una gran versatilidad a la hora de adaptar el análisis a distintos escenarios de complejidad sonora.

El flujo de datos general del sistema puede dividirse en seis grandes etapas principales:

1. **Conversión y carga del audio:** El primer paso consiste en convertir el archivo de entrada (que puede estar en formatos como .mp4, .mp3 u otros compatibles) al formato .wav, que es más adecuado para su tratamiento técnico. Esta conversión garantiza la homogeneidad en la resolución de muestreo y asegura la compatibilidad con las funciones de análisis acústico implementadas, principalmente basadas en la biblioteca librosa. Una vez convertido, el archivo se carga en memoria y se prepara para su procesamiento.
2. **Preprocesamiento del audio:** En esta fase, el sistema aplica un primer filtrado espectral sobre el audio. Se recorta el segmento temporal especificado por el usuario (sec_inicio, sec_final) y se filtran aquellas zonas del espectro cuya energía no supera un determinado umbral (db_min). Esta limpieza permite eliminar ruido de fondo, silencios o fragmentos irrelevantes, y garantiza que los análisis posteriores se realicen únicamente sobre las partes más significativas de la señal. Este paso es fundamental para mejorar la precisión del sistema y reducir los falsos positivos en la detección de notas.
3. **Transformada de Fourier:** El núcleo técnico del análisis espectral se basa en la aplicación de la Transformada de Fourier de tiempo corto (STFT). Esta técnica permite descomponer el audio en ventanas temporales (por ejemplo, de 200 ms) con cierto grado de solapamiento entre ellas, obteniendo para cada una una representación en frecuencia (espectro). De esta forma, se transforma la señal del dominio temporal al dominio tiempo-frecuencia, lo que resulta esencial para poder detectar y clasificar las notas musicales. La resolución temporal y frecuencial de este análisis puede ajustarse mediante los parámetros rate y solape.

4. **Filtrado adicional (opcional):** Una funcionalidad avanzada del sistema permite aplicar filtros personalizados en el dominio de la frecuencia. Estos filtros se generan previamente y actúan como máscaras que enfatizan las bandas de frecuencia más relevantes para las notas musicales, atenuando el resto. El diseño de estos filtros se basa en funciones exponenciales centradas en las frecuencias objetivo de cada nota, con una caída suave que permite aislar los armónicos principales. Esta etapa es opcional, pero puede resultar especialmente útil para analizar instrumentos con mayor ruido espectral o grabaciones con condiciones acústicas desfavorables.
5. **Clasificación de notas:** Una vez calculados los vectores de características espectrales, el sistema los analiza para estimar qué nota musical corresponde a cada segmento. El método utilizado varía según el tipo de instrumento:
    - En el caso de instrumentos melódicos, se identifica directamente la frecuencia dominante y se compara con una base de datos para asignar la nota más cercana.
    - Para instrumentos polifónicos, se utiliza un vector de 108 características (una por cada nota posible), que se introduce en un modelo de clasificación previamente entrenado. El sistema es compatible con varios tipos de clasificadores:
        - K-Nearest Neighbors (KNN): útil por su simplicidad y eficacia en contextos controlados.
        - Multi-Layer Perceptron (MLP): ofrece mayor capacidad de generalización con poco coste computacional.
        - Redes Neuronales Recurrentes (RNN): especialmente eficaces cuando se quiere capturar la evolución secuencial de las notas, como en interpretaciones musicales con dependencia temporal.
6. **Generación de la partitura:** El resultado de la etapa anterior es una secuencia ordenada de notas musicales (o silencios, marcados como "ZZZ") asociadas a intervalos temporales. Esta información se procesa para generar una partitura visual e interactiva, representada mediante gráficos tipo piano roll. Cada nota se muestra como una barra horizontal cuya posición en el eje X representa el instante de aparición y cuya longitud indica la duración relativa. El usuario puede visualizar esta partitura, reproducir el audio en paralelo o ajustar parámetros para refinar el resultado.

Este diseño modular, tal y como se muestra en la Ilustración 5 facilita la adaptación a nuevos instrumentos, el entrenamiento de nuevos modelos y la personalización del preprocesamiento.

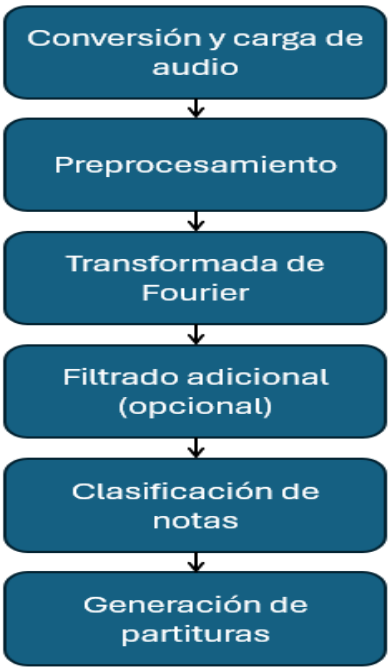


Ilustración 5

El repositorio del proyecto ha sido diseñado con un enfoque modular, permitiendo una organización clara y escalable del código. Esta estructura favorece la separación de responsabilidades entre los distintos componentes del sistema: por un lado, la lógica de visualización e interacción (frontend/backend web) y, por otro, la lógica del procesamiento musical (análisis de audio, filtrado, predicción y generación de partituras). Actualmente, el repositorio es privado, dado que el desarrollo del sistema aún está en curso y se está trabajando en una versión de la aplicación para dispositivos móviles. Se espera hacerlo público una vez se garantice una mayor fiabilidad y estabilidad del sistema.

A continuación, se describe en detalle la organización general del repositorio, explicando la función de cada directorio y archivo principal:

backend/: Infraestructura web y servidor

Este directorio contiene todos los elementos necesarios para el despliegue del servidor web, que actúa como interfaz entre el usuario y el sistema de análisis musical. Está organizado en subdirectorios con roles bien definidos:

- **static/**: Contiene archivos estáticos como hojas de estilo (.css), imágenes, fuentes y scripts JavaScript. Estos recursos son esenciales para el diseño visual, la animación de la partitura y la interacción en el navegador.

- **templates/**: Incluye las vistas HTML que se renderizan mediante el motor de plantillas (por ejemplo, Jinja2 si se usa Flask). Estas plantillas permiten mostrar

formularios, resultados y visualizaciones de forma dinámica en función de los datos procesados por el backend.

- **utils/**: Carpeta dedicada a utilidades generales empleadas por la lógica web. Puede incluir funciones para el manejo de archivos, validación de parámetros o generación de rutas temporales, entre otras tareas auxiliares que no pertenecen directamente al procesamiento de audio.

modelos/: Lógica de procesamiento musical y predicción

Este directorio reúne los scripts encargados del núcleo funcional del sistema: la lectura y análisis de archivos de audio, el procesamiento del contenido espectral y la posterior clasificación de notas. Aquí se definen los pipelines de ejecución diferenciados por tipo de instrumento (por ejemplo, flauta o guitarra), así como las llamadas a los modelos de machine learning. Esta capa es totalmente independiente del entorno web, lo que facilita su reutilización, testeo y extensión.

funcionesComunes/: Funciones reutilizables y lógica compartida

El sistema centraliza en este directorio las funciones que son utilizadas de forma transversal en diferentes partes del proyecto. Esto permite evitar duplicidades y mejora la mantenibilidad del código. Los scripts más relevantes incluidos en esta carpeta son:

- **constantesComunes.py**: Define el conjunto completo de notas musicales (108 notas) y sus frecuencias correspondientes. Estas constantes se usan en todo el sistema como referencia para asignar notas según sus frecuencias dominantes.

- **getConfig.py**: Ofrece una función para determinar el rango de frecuencias relevantes según el instrumento seleccionado. Esto permite acotar la búsqueda espectral y mejorar el rendimiento de los clasificadores.

- **lecturaAudios.py**: Contiene las funciones principales para la manipulación de archivos de audio, incluyendo la conversión a WAV, recorte temporal, aplicación de umbral de decibelios, transformada de Fourier por ventanas, generación del espectrograma, y extracción de frecuencias dominantes.

- **organizarData.py**: Se encarga de construir los vectores de características espectrales. Para instrumentos polifónicos, calcula la suma de amplitudes por cada tramo de frecuencia asociado a una nota, permitiendo la creación de vectores de entrada de 108 dimensiones (una por nota).

- **funcionesModelos.py**: Implementa las funciones para cargar y aplicar modelos de machine learning previamente entrenados. Soporta varios tipos de clasificadores: KNN (modelo simple por vecindad), MLP (perceptrón multicapa) y RNN (redes neuronales recurrentes). Además, maneja la carga de LabelEncoders y otros artefactos necesarios para la predicción.

FILTRO/: Submódulo para generación y aplicación de filtros espectrales

Este subdirectorio encapsula la lógica relacionada con los filtros en el dominio de la frecuencia. Estos filtros son opcionales, pero permiten aplicar máscaras que resaltan o atenúan determinadas bandas del espectro, lo cual puede mejorar el rendimiento del sistema en entornos ruidosos o grabaciones complejas. Se compone de dos scripts principales:

- **makeFiltro.py**: Permite construir filtros personalizados basados en funciones exponenciales centradas en las frecuencias musicales. Los filtros pueden ser exportados a Excel para facilitar su análisis, modificación o reutilización posterior.

- **cargarFiltro.py**: Encargado de importar un filtro desde un archivo externo (como .xlsx) y aplicarlo al contenido espectral calculado sobre las ventanas del audio. La aplicación del filtro se realiza mediante multiplicación punto a punto sobre los vectores de amplitud.

INSTRUMENTOS/: Esta carpeta contiene la lógica personalizada para cada instrumento musical tratado en el sistema. En este caso, se incluyen dos subdirectorios: FLAUTA/ y GUITARRA/. Cada uno aloja un script principal que define el flujo completo para procesar audios y extraer la partitura correspondiente.

- mainFlauta.py: Este script contiene la función mainFlauta, diseñada específicamente para instrumentos **melódicos** como la flauta travesera. El flujo implementado es:

    - Conversión del audio a .wav.

    - Generación y visualización del espectrograma.

    - Aplicación de un filtrado basado en un umbral de decibelios y rango de frecuencias.

    - División del audio en segmentos y extracción del contenido espectral.

    - Predicción de una única nota por segmento basándose en la frecuencia dominante.
    - Construcción de la partitura final a partir de las notas detectadas.
    - Además, existe una versión extendida mainFlautaFiltro, que incorpora la **aplicación de un filtro espectral** (si se ha generado previamente y se pasa su ruta). Este filtro actúa sobre los datos de frecuencia para eliminar ruido o armónicos no deseados antes de la detección de notasmainFlauta.
- mainGuitarra.py: Este script está orientado a instrumentos polifónicos, como la guitarra. La lógica incluye:
    - Conversión del audio y preprocesamiento como en el caso de la flauta.
    - Segmentación del audio y cálculo de vectores espectrales.
    - Vectorización en 108 características correspondientes a notas musicales.
    - Aplicación de un modelo de aprendizaje automático para clasificar cada segmento soportando modelos KNN (joblib), MLP (pickle) y RNN (.h5 + LabelEncoder).
    - Predicción de una nota por segmento o marcación como "ZZZ" si el vector es demasiado débil y por tanto se trata de un silencio.

Scripts principales en la raíz del proyecto:

La raíz del repositorio contiene los scripts fundamentales que permiten ejecutar, probar y desplegar la aplicación en sus distintas funcionalidades. Cada archivo tiene un propósito específico y está diseñado para facilitar tanto el flujo principal de análisis como las tareas auxiliares de entrenamiento, pruebas o ejecución del servidor web. A continuación, se detalla la función de cada uno de estos archivos clave:

- **mainAplicacion.py**: Este script constituye el **punto de entrada principal del sistema** para el procesamiento de archivos de audio. Permite ejecutar el análisis completo desde consola, desde la carga del audio hasta la obtención del vector de notas o la partitura. El archivo es totalmente configurable por el usuario, quien puede definir el tipo de instrumento a analizar (melódico o polifónico), el rango de frecuencias, los parámetros de segmentación, la inclusión o no de un filtro espectral y el modelo de clasificación a emplear (KNN, MLP, RNN). Es una herramienta útil tanto para pruebas en local como para procesamientos independientes del entorno web.

- **mainModelos.py**: Script dedicado al **entrenamiento y evaluación de modelos de clasificación**. Permite entrenar nuevos modelos con datos etiquetados, validar su rendimiento y guardarlos en disco para su posterior utilización durante la fase de predicción. Este script soporta distintos algoritmos de clasificación, permitiendo experimentar con múltiples configuraciones y exportar los modelos en formatos compatibles con el sistema.

- **app.py**: Archivo que actúa como **script principal del servidor web**. Contiene la configuración y ejecución del backend de la aplicación, que suele estar implementado en un framework como Flask. Gestiona las rutas (endpoints) asociadas a la carga de archivos, procesamiento de los mismos, visualización del espectrograma y generación de la partitura. También controla la interacción entre el usuario y el sistema, gestionando los formularios, peticiones y respuestas.

- **requirements.txt**: Archivo de texto que define todas las **dependencias necesarias del entorno Python** para ejecutar correctamente la aplicación. Incluye librerías como librosa, numpy, pandas, scikit-learn, tensorflow, plotly, entre otras. Este fichero es imprescindible para garantizar que el entorno de ejecución pueda replicarse fácilmente en otras máquinas o entornos virtuales, utilizando comandos como pip install -r requirements.txt.

uploads/ — Carpeta de almacenamiento temporal de audios

Este directorio actúa como el **repositorio temporal de archivos de audio** que son subidos por el usuario a través de la interfaz web. Cada vez que se realiza una carga de archivo desde el navegador, el audio se guarda en esta carpeta con un nombre único, desde donde luego se accede para realizar su procesamiento. Esta separación física permite aislar los archivos del usuario de la lógica del sistema y facilita el control de versiones, limpieza y persistencia de los datos.

En la Ilustración 6 se puede observar la estructura de repositorio seguida en este proyecto:

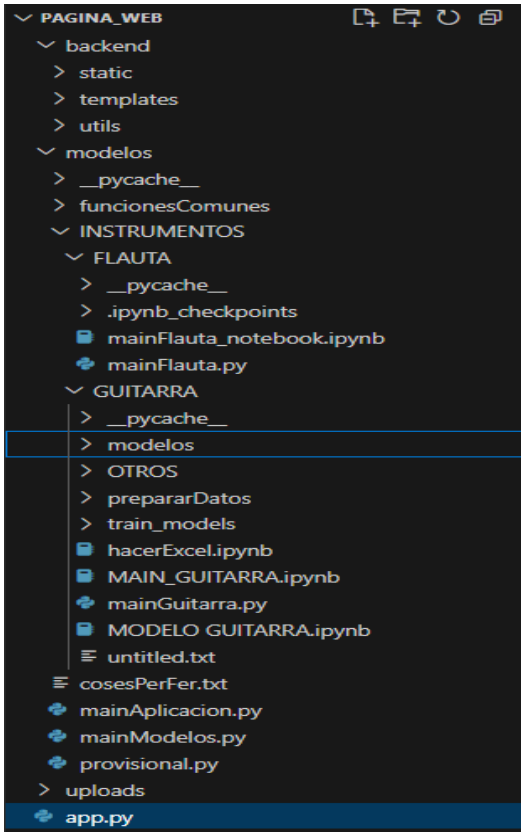


Ilustración 6

El módulo de datos constituye el núcleo del procesamiento en esta aplicación. Su función principal es transformar un archivo de audio en una representación visual (espectrograma) y luego traducir esa información espectral en una secuencia estructurada de notas musicales. Este proceso se divide en dos fases principales: la generación del espectrograma y la construcción de la partitura.

Generación del espectrograma interactivo

Este paso constituye una de las fases iniciales y más importantes del sistema, ya que permite transformar la señal de audio en bruto, registrada en el dominio temporal, en una representación más informativa y visual: el espectrograma. Esta representación tiempo-frecuencia posibilita observar de forma detallada, tal y como se ve en la Ilustración 7, cómo varía la energía asociada a cada componente frecuencial del sonido a lo largo del tiempo. Así, el espectrograma se convierte en una herramienta fundamental tanto para la depuración del sistema como para la comprensión del contenido musical por parte del usuario.

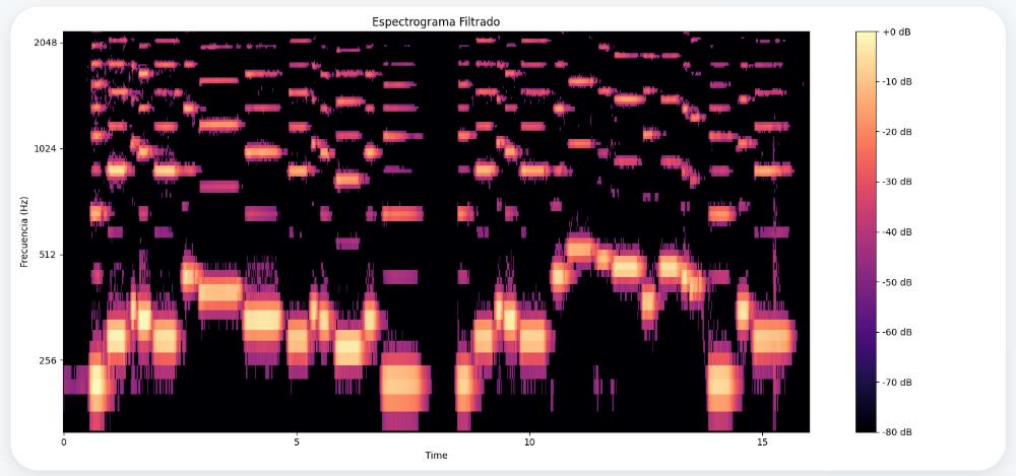


Ilustración 7

El flujo seguido para la generación del espectrograma interactivo es el siguiente:

- **Conversión del archivo a WAV**: El primer paso consiste en estandarizar el formato del archivo de audio. Dado que los usuarios pueden subir archivos en múltiples formatos (por ejemplo, .mp3, .mp4, .aac), es necesario convertirlos a un formato compatible con las bibliotecas utilizadas para el análisis, como librosa o moviepy. El formato .wav es el elegido por su compatibilidad, preservación de calidad y facilidad de lectura. Esta conversión garantiza que el sistema pueda operar de forma uniforme con cualquier archivo, independientemente de su origen.

- **Selección temporal**: Una vez convertido el audio, se recorta según el intervalo de tiempo definido por el usuario. Esta selección se realiza mediante los parámetros sec_inicio y sec_final, que permiten acotar el análisis a una sección específica del archivo. Este recorte es fundamental para evitar procesar partes innecesarias del audio (como silencios prolongados, ruidos iniciales o finales) y centrarse únicamente en la región de interés, lo que mejora la eficiencia computacional y la precisión de los análisis posteriores.

- **Aplicación de un umbral dB**: Tras delimitar el intervalo temporal, se aplica un filtro espectral basado en un umbral mínimo de energía (db_min). Este paso consiste en eliminar aquellas regiones del audio cuya intensidad esté por debajo de dicho umbral, lo que suele corresponder a silencios, ruido de fondo o señales débiles no relevantes para el análisis musical. Esta limpieza inicial del contenido espectral ayuda a reducir el ruido en el espectrograma, destacando únicamente las zonas con contenido sonoro significativo.

- **Cálculo del espectrograma**: A continuación, se realiza la Transformada de Fourier por ventanas temporales, también conocida como **Short-Time Fourier Transform (STFT)**. Esta técnica consiste en dividir el audio en pequeños segmentos (ventanas), y calcular la distribución de frecuencias presentes en

cada uno. El resultado es una matriz tridimensional donde cada punto representa la amplitud (o intensidad) de una determinada frecuencia en un instante concreto. Esta representación permite identificar tanto las frecuencias fundamentales como los armónicos, así como observar cambios dinámicos en la estructura sonora del audio.

- **Visualización interactiva**: Finalmente, la matriz generada se representa de forma visual mediante un gráfico interactivo. Para ello, se emplea la biblioteca Plotly, que permite construir gráficos dinámicos en el navegador. En esta visualización, el eje horizontal representa el tiempo, el eje vertical representa la frecuencia, y la intensidad del color indica la amplitud o energía en cada punto. Esta herramienta visual resulta especialmente útil para verificar la correcta captura del contenido musical, identificar errores o artefactos, y realizar ajustes en tiempo real. El usuario puede inspeccionar visualmente el espectrograma y, si considera que el resultado no es suficientemente claro o limpio, puede volver atrás y modificar parámetros como el umbral dB, el rango temporal o la resolución temporal para generar un nuevo espectrograma más refinado.

A continuación, se presenta en la Ilustración 8 la interacción entre el frontend y el backend durante el proceso de generación del espectrograma interactivo, detallando el envío de parámetros, el procesamiento del audio y la visualización del resultado:

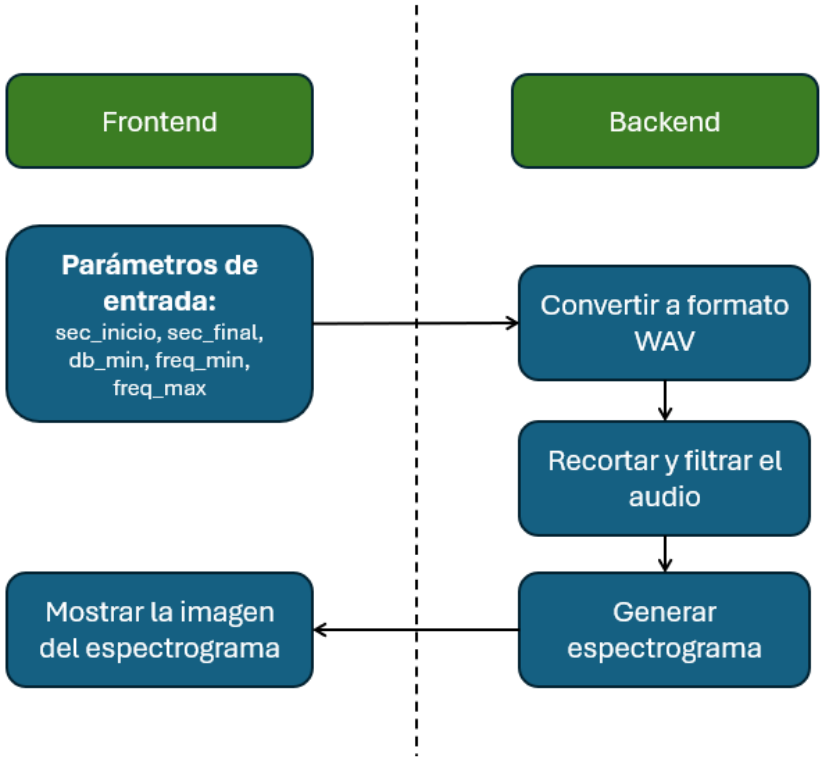


Ilustración 8

Generación del vector de notas

La generación del vector de notas constituye una de las etapas fundamentales dentro del sistema de análisis musical propuesto. Esta fase es responsable de transformar la información obtenida en el dominio de la frecuencia, producto del procesamiento espectral del audio, en una secuencia estructurada de notas musicales, comprensible y útil tanto para la posterior visualización como para análisis musicales avanzados.

Esta fase comienza una vez que el audio ha sido segmentado y transformado mediante técnicas como la Transformada de Fourier por ventanas temporales, lo que proporciona una representación detallada del contenido frecuencial del audio en función del tiempo. A partir de esta representación, el objetivo principal es identificar qué nota se está interpretando en cada tramo de tiempo analizado. El resultado de este proceso es un vector que contiene, en orden temporal, una etiqueta por cada segmento de audio, correspondiente a la nota estimada en dicho tramo.

El flujo de trabajo comienza con la recepción de parámetros de configuración definidos por el usuario a través de la interfaz web (frontend). Estos parámetros no solo permiten personalizar el análisis, sino que son esenciales para adaptar el sistema al tipo de instrumento musical, al tipo de grabación y al contexto de uso. A continuación, se detallan estos parámetros y su relevancia dentro del proceso:

- **Instrumento (INSTRUMENTO):** El usuario debe indicar el instrumento al que pertenece el audio (por ejemplo, flauta, guitarra, piano, etc.). Este dato es fundamental, ya que cada instrumento posee un rango de frecuencias característico y una estructura armónica distinta. En función del instrumento seleccionado, el sistema adaptará el rango de frecuencias a analizar, la granularidad del análisis, e incluso seleccionará el modelo de machine learning más apropiado. Por ejemplo, en el caso de instrumentos melódicos como la flauta, se prioriza la frecuencia dominante, mientras que en instrumentos polifónicos como la guitarra se utilizan vectores de amplitud por tramos de frecuencia.

- **Intervalo temporal (SEC_INICIO, SEC_FINAL):** Estos parámetros permiten definir el segmento del audio que se desea analizar. Esta acotación temporal es útil tanto para centrar el análisis en las partes más relevantes como para evitar procesar regiones no deseadas del audio, como silencios o ruidos iniciales. Además, al trabajar solo con una porción del archivo, se mejora la eficiencia y se reduce la carga computacional.

- **Frecuencias de interés (freq_min, freq_max):** Especificar el rango de frecuencias que se desea considerar durante el análisis espectral permite enfocar el sistema únicamente en las bandas relevantes, evitando así interferencias o contenido fuera del espectro musical del instrumento. Este

ajuste es especialmente útil para filtrar ruidos de fondo o armónicos indeseados, y para mejorar la precisión del proceso de clasificación.

- **Resolución temporal (rate):** Define la duración de cada ventana de análisis en milisegundos. Este parámetro determina el equilibrio entre resolución temporal y resolución frecuencial: ventanas más cortas permiten captar cambios rápidos en el tiempo, mientras que ventanas más largas proporcionan mayor precisión en la estimación de las frecuencias presentes. Elegir un valor adecuado es clave para representar fielmente la evolución tonal del audio.

- **Solapamiento (solape):** Establece el porcentaje de superposición entre ventanas consecutivas. Un alto grado de solapamiento permite una mayor suavidad en la transición entre tramos de análisis y reduce la posibilidad de perder información relevante entre ventanas. Esto es particularmente importante en interpretaciones musicales rápidas o con transiciones tonales sutiles.

- **Ruido mínimo (db_min):** Define el umbral mínimo de energía (en decibelios) por debajo del cual se considera que el audio carece de contenido relevante. Los segmentos cuya amplitud no supere este umbral se interpretan como silencios o ruido de fondo, y se etiquetan como "ZZZ" en el vector de notas, lo que permite distinguir claramente las zonas activas del audio.

- **Tipo de modelo (model_type):** Permite al usuario elegir entre distintos algoritmos de clasificación, como K-Nearest Neighbors (KNN), Multi-Layer Perceptron (MLP) o Recurrent Neural Network (RNN). Cada modelo tiene sus propias características, capacidades y rendimiento en función del tipo de datos y del instrumento. Esta selección afecta directamente al comportamiento del sistema en la fase de predicción de notas.

- **Modelo y codificador (paths_models):** El sistema requiere conocer la ubicación de los modelos previamente entrenados y, en su caso, del codificador de etiquetas (LabelEncoder) utilizado para traducir las predicciones numéricas a etiquetas de notas musicales. Estos archivos son cargados dinámicamente en el backend según las rutas proporcionadas, lo que permite trabajar con diferentes configuraciones o actualizar los modelos sin modificar la lógica del sistema.

Estos parámetros se estructuran como argumentos en la llamada a la función principal (mainFlauta o mainGuitarra) y son consumidos por las funciones del backend para ejecutar el análisis siguiendo el formado de entrada que se muestra en la Ilustración 9

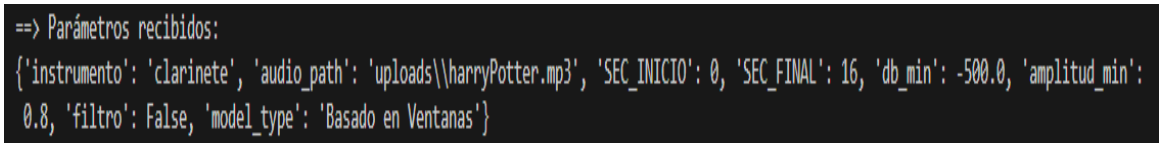


Ilustración 9

Una vez que el audio ha sido preprocesado, se calcula su representación espectral segmentada. Cada segmento temporal se convierte en un **vector de características**, a partir del cual se determina la nota correspondiente. El proceso se adapta ligeramente según el tipo de instrumento:

Generador del vector de notas para instrumentos melódicos (ej. flauta):

Cuando el sistema detecta que el instrumento seleccionado por el usuario es de tipo melódico, aquellos que producen una sola nota a la vez, como la flauta, el clarinete o la trompeta, se emplea una estrategia de análisis simplificada pero muy precisa, basada en la extracción de la frecuencia dominante de cada segmento del audio. Este tipo de instrumentos se caracteriza por emitir un único tono fundamental acompañado de armónicos, por lo que resulta más efectivo centrarse en la frecuencia principal.

El flujo de procesamiento para este tipo de instrumentos se compone de las siguientes etapas:

1. **Segmentación del audio y cálculo de la Transformada de Fourier (FFT):** En primer lugar, el archivo de audio se divide en múltiples ventanas temporales de duración fija, cuya longitud ha sido previamente especificada por el usuario mediante el parámetro rate (en milisegundos). Además, se aplica un cierto porcentaje de solapamiento entre ventanas consecutivas (solape), con el fin de asegurar una transición suave entre tramos y capturar con mayor precisión los cambios tonales.

   A cada uno de estos fragmentos temporales se le aplica la Transformada Rápida de Fourier (FFT), lo que permite obtener la distribución de energía en las diferentes frecuencias contenidas en ese tramo. El resultado es un espectro de amplitudes, en el que se puede observar claramente cuál es la frecuencia que predomina en ese instante del audio.

2. **Selección de la frecuencia dominante en cada ventana:** Una vez obtenido el espectro de amplitudes de cada ventana temporal, el siguiente paso consiste en identificar la frecuencia dominante, es decir, aquella que presenta la mayor amplitud en el espectro. Esta frecuencia es, en la mayoría de los casos, la que representa la nota que el instrumento está ejecutando en ese momento. Esta etapa es crítica, ya que errores en la identificación de la frecuencia dominante pueden llevar a una asignación incorrecta de la nota. Por ello, el sistema suele aplicar filtros adicionales para descartar picos espurios o

armónicos más intensos que el fundamental, y enfoca el análisis en el rango de frecuencias característico del instrumento.

3. **Asignación de nota musical a partir de la frecuencia:** Una vez identificada la frecuencia dominante en cada segmento, se procede a convertirla en una nota musical reconocible. Para ello, se compara dicha frecuencia con un diccionario predefinido (frecuenciasNotas) que contiene las frecuencias estándar de todas las notas musicales dentro de un determinado rango (por ejemplo, de C3 a B6).

   Se calcula la distancia en frecuencia entre la dominante obtenida y cada una de las notas del diccionario, y se selecciona la más cercana. De esta manera, si la frecuencia dominante es, por ejemplo, 440.2 Hz, se identificará como un "A4" (la estándar a 440 Hz). Este proceso se repite para cada ventana, generando así una serie de etiquetas musicales.

4. **Construcción del vector de notas:** Finalmente, se construye un vector ordenado de notas que representa la secuencia temporal de sonidos detectados en el audio. Cada elemento del vector corresponde a una ventana temporal y contiene la nota musical estimada en ese intervalo. El resultado es una lista cronológica que refleja cómo varían las notas a lo largo del tiempo, lo que permite obtener una representación fiel de la interpretación musical grabada. Este vector es la base para generar la partitura interactiva tipo piano roll y puede ser utilizado también como entrada para modelos de entrenamiento o validación de otras herramientas de análisis musical.

La Ilustración 10 muestra el flujo completo de generación del vector de notas para instrumentos melódicos a partir del audio, diferenciando claramente el papel del frontend en la definición de parámetros y el procesamiento backend según el tipo de instrumento.

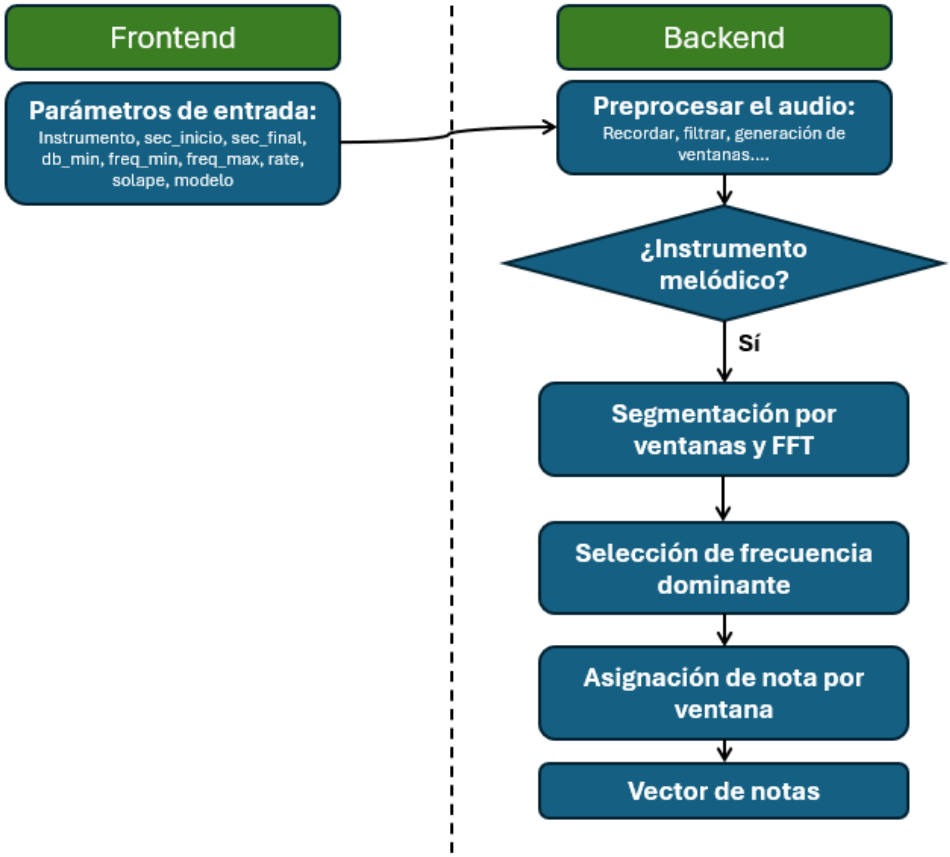


Ilustración 10

Generador del vector de notas para instrumentos polifónicos (ej. guitarra):

En el caso de instrumentos polifónicos, como la guitarra, el piano o el arpa, el proceso de análisis y extracción de notas se vuelve considerablemente más complejo debido a la posibilidad de que varias notas suenen simultáneamente en cada instante temporal. A diferencia de los instrumentos melódicos, donde se puede identificar una única frecuencia dominante, en los polifónicos es necesario representar el contenido espectral de manera más rica y multivariada.

Para abordar este desafío, el sistema implementa una estrategia basada en vectores de características de alta dimensión, que luego son procesados mediante modelos de machine learning previamente entrenados para identificar la nota predominante o más representativa de cada segmento.

El proceso se estructura en las siguientes fases:

1. **Segmentación del audio y aplicación de la FFT:** En primer lugar, se divide el archivo de audio en múltiples ventanas temporales, de duración definida por el parámetro rate y con un grado de solapamiento especificado por el usuario (solape). Este procedimiento garantiza una resolución temporal suficiente para detectar cambios rápidos en la armonía o la dinámica de la interpretación. A cada una de estas ventanas se le aplica la Transformada Rápida de Fourier

(FFT), que convierte el contenido de la señal desde el dominio temporal al dominio frecuencial. El resultado es un espectro de amplitudes que refleja cómo se distribuye la energía sonora entre las distintas frecuencias dentro de ese fragmento de audio.

2. **Cálculo del vector de amplitudes (108 dimensiones):** A diferencia del caso melódico, aquí no se busca una única frecuencia dominante. En su lugar, el sistema utiliza una función denominada makeTramos() que divide el rango de frecuencias de interés en 108 tramos o bandas, cada una asociada a una nota musical teórica.

   Para cada una de estas bandas, se suma la energía o amplitud presente en el espectro correspondiente al segmento de audio. El resultado es un vector de 108 componentes, donde cada dimensión representa la energía acumulada en el tramo de frecuencia correspondiente a una nota musical concreta. Este vector encapsula una huella espectral del contenido armónico del audio en ese instante.

3. **Clasificación mediante un modelo de machine learning:** Una vez calculado el vector de características, este se introduce como entrada en un modelo de clasificación previamente entrenado. El sistema permite al usuario seleccionar entre diversos tipos de algoritmos, incluyendo:

   - K-Nearest Neighbors (KNN): Basado en la proximidad a ejemplos etiquetados del conjunto de entrenamiento.
   - Multi-Layer Perceptron (MLP): Una red neuronal con una o más capas ocultas que aprende relaciones no lineales complejas.
   - Recurrent Neural Network (RNN): Ideal para capturar la secuencia temporal, útil si se entrenó con contexto histórico de notas.
     El modelo devuelve como salida una etiqueta de nota musical estimada para ese segmento. Aunque el instrumento puede tener múltiples notas en una ventana, el sistema predice la más destacada o representativa, dependiendo del entrenamiento y la arquitectura del modelo utilizado.

4. **Filtrado por energía mínima:** No todos los segmentos del audio contienen contenido musical relevante. En ocasiones, especialmente en pasajes con silencios, transiciones o ruido de fondo, la energía del vector puede ser muy baja.

   Para evitar clasificaciones erróneas en estos casos, se aplica un filtro basado en el valor máximo del vector de amplitudes. Si dicho valor no supera un umbral mínimo (amplitud_min), el sistema considera que el segmento no contiene una nota reconocible y lo etiqueta con el marcador especial "ZZZ", indicando que ha sido descartado por falta de información sonora significativa.

5. **Construcción del vector final de notas:** Finalmente, se construye un vector ordenado que representa la secuencia temporal de notas estimadas a lo largo del audio. Cada posición del vector corresponde a una ventana temporal del análisis, y su valor puede ser una nota musical (por ejemplo, "G3", "F#4") o el marcador "ZZZ" en caso de ausencia de nota.

   Este vector conserva la estructura secuencial del audio original, lo que permite reconstruir posteriormente la interpretación musical en una visualización tipo piano roll o para convertirlo en notación tradicional. También resulta útil para análisis posteriores, comparación con partituras reales o evaluación automática del rendimiento del sistema.

La Ilustración 11 muestra el flujo completo de generación del vector de notas para instrumentos polifónicos a partir de un archivo de audio, diferenciando claramente el papel del **frontend**, encargado de definir los parámetros de entrada, y el **backend**, responsable del procesamiento del audio. Además, se detallan las distintas ramas de ejecución según el modelo de clasificación seleccionado (KNN, MLP o RNN), cada una con su lógica específica para generar la secuencia final de notas musicales.

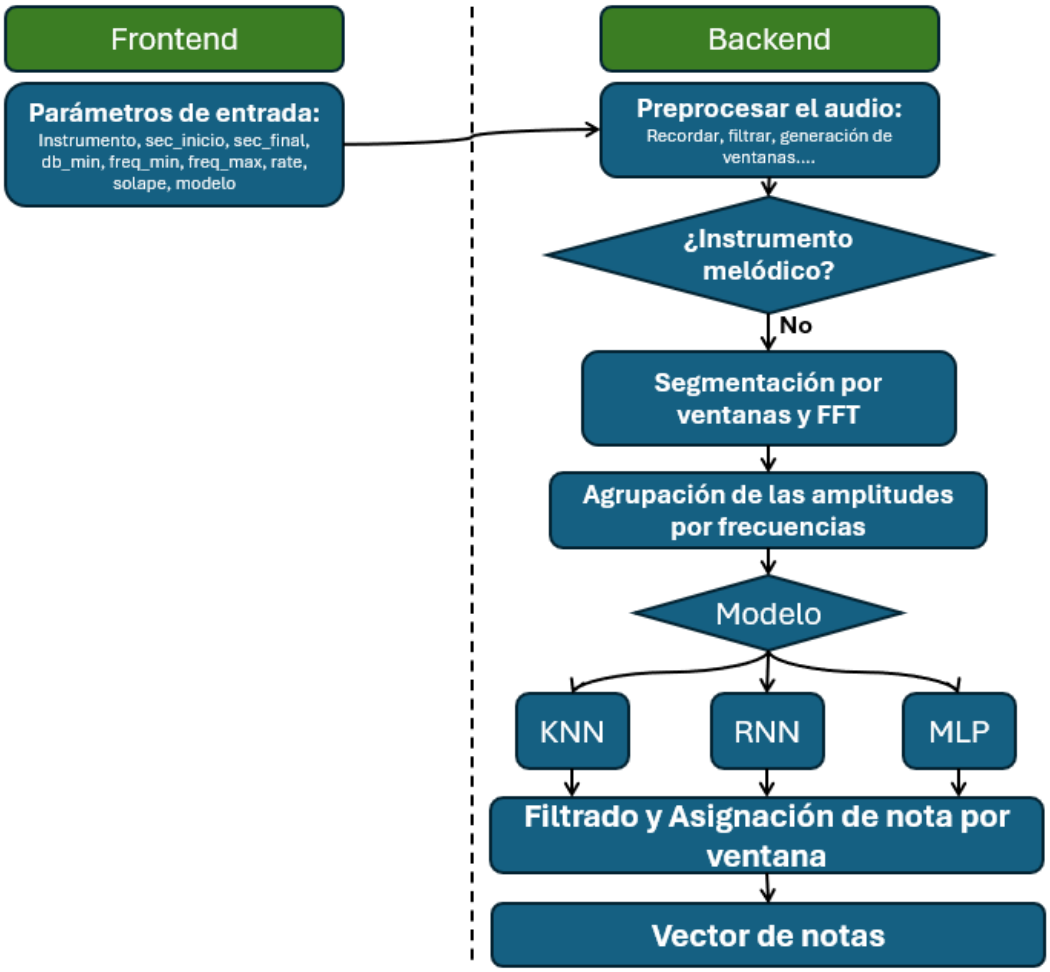


Ilustración 11

El resultado es una **lista ordenada de notas musicales** (o símbolos "ZZZ") que representan la estimación del sistema sobre lo que se toca en el audio, tramo a tramo tal y como se muestra en la Ilustración 12. Este vector es la base para la generación posterior de la **partitura interactiva**, ya que conserva tanto la **secuencia temporal** como la **identidad tonal** de cada segmento.

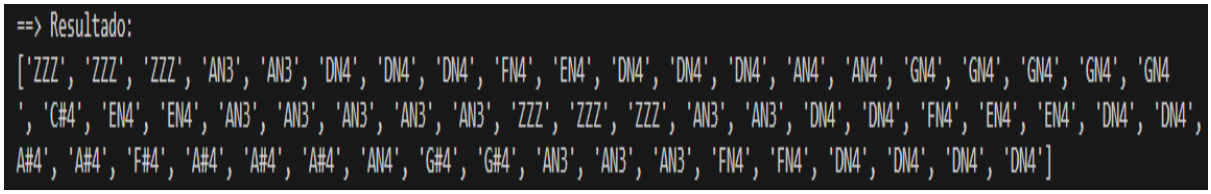


Ilustración 12

Generación de la partitura interactiva

Una vez finalizado el proceso de análisis espectral y clasificación tonal del audio, y tras haber generado el correspondiente vector de notas, que representa la secuencia temporal de las notas musicales estimadas por el sistema, se inicia la última etapa clave del sistema: la generación de una partitura interactiva. Este módulo tiene como finalidad ofrecer al usuario una representación visual clara, dinámica y comprensible del contenido musical contenido en el archivo de audio.
La partitura no solo permite visualizar la sucesión de notas estimadas, sino también facilita su análisis, revisión y comparación con una posible partitura real. Además, al ser interactiva, permite una experiencia más cercana a la ejecución musical digital, integrando elementos visuales dinámicos como líneas de reproducción o animaciones.
El flujo es el siguiente:

- **Recepción del vector de notas:** El sistema parte de un vector previamente generado, en el que cada elemento representa la nota estimada en un segmento temporal concreto del audio. Esta lista está **ordenada cronológicamente**, reflejando la evolución del contenido musical conforme se avanza en el tiempo.
  Cada entrada del vector puede contener una nota musical reconocible (por ejemplo, "CN4", "D#5", "AN3", etc.) o, en caso de que el sistema haya detectado que no había suficiente energía o contenido tonal en un segmento concreto, un marcador especial "ZZZ" que indica una ausencia de nota o un posible silencio. Este enfoque permite representar de forma explícita los tramos no musicales o ambiguos, preservando así la fidelidad temporal del análisis.
- **Asignación de tiempos relativos**: Para representar las notas en una partitura visual es necesario traducir la secuencia de notas a **un eje temporal continuo**. Para ello, se calcula el tiempo relativo de cada nota en función de dos variables:
  La duración de cada ventana de análisis (definida por el parámetro rate, normalmente en milisegundos).
  El número de veces consecutivas que una misma nota se repite en el vector.

Por ejemplo, si una nota como "G4" aparece cinco veces seguidas y cada ventana representa 50 ms, el sistema interpreta que "G4" se ha mantenido durante 250 ms. De esta manera, se construye una secuencia de notas donde cada una tiene una posición de inicio (en milisegundos o segundos desde el comienzo del audio) y una duración estimada, expresada también en tiempo relativo. Esta etapa es esencial para una visualización coherente y proporcional del contenido musical.

- **Conversión a formato gráfico**: Una vez definidas la duración y posición temporal de cada nota, se genera una estructura intermedia que relaciona cada nota con su instante de inicio y su duración total, descartando aquellas etiquetas "ZZZ" si se desea una visualización más limpia.
  Esta estructura se organiza como una lista de diccionarios u objetos, donde cada entrada contiene al menos los siguientes campos: nota, tiempo de inicio, duración, tal y como se muestra en la Ilustración 13. Esta estructura es la base que utilizarán posteriormente las bibliotecas de visualización para construir el gráfico musical.

```
==> JSON generado:
{'nota': 'DN4', 'tiempo_inicio': 2.211, 'tiempo_fin': 3.418}
{'nota': 'FN4', 'tiempo_inicio': 3.619, 'tiempo_fin': 3.82}
{'nota': 'A#4', 'tiempo_inicio': 4.021, 'tiempo_fin': 4.423}
{'nota': 'DN4', 'tiempo_inicio': 5.629, 'tiempo_fin': 6.634}
{'nota': 'FN4', 'tiempo_inicio': 6.835, 'tiempo_fin': 7.237}
{'nota': 'A#4', 'tiempo_inicio': 7.237, 'tiempo_fin': 7.639}
{'nota': 'DN4', 'tiempo_inicio': 8.845, 'tiempo_fin': 9.448}
{'nota': 'FN4', 'tiempo_inicio': 9.448, 'tiempo_fin': 9.649}
{'nota': 'A#4', 'tiempo_inicio': 9.649, 'tiempo_fin': 10.052}
{'nota': 'CN5', 'tiempo_inicio': 10.052, 'tiempo_fin': 10.454}
```

Ilustración 13

- **Visualización interactiva**: La partitura se representa mediante gráficos dinámicos tipo piano roll utilizando bibliotecas como Plotly o p5.js. El eje X representa el tiempo y el eje Y representa las notas musicales tal y como se muestra en Ilustración 14. Cada nota se dibuja como una barra horizontal cuya longitud depende de su duración, y puede acompañarse de una línea vertical que simula la lectura en tiempo real.

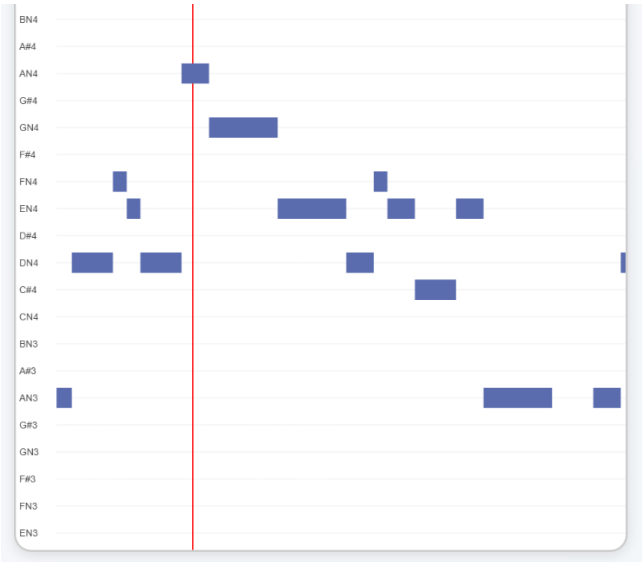


Ilustración 14

A continuación, se presenta en la Ilustración 15 el flujo de datos correspondiente a la generación de la partitura interactiva a partir del vector de notas previamente calculado, incluyendo el tratamiento de la secuencia y su visualización gráfica.

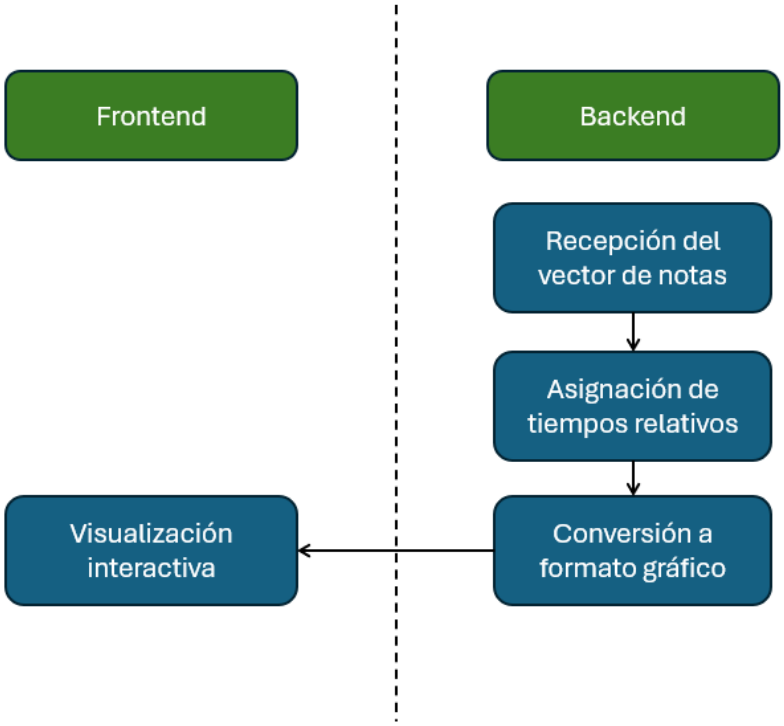


Ilustración 15

Capítulo 6. Preprocesado del audio

Este capítulo describe el conjunto de operaciones de preprocesado aplicadas a los archivos de audio antes de ser analizados por el sistema de detección de notas. Dicho preprocesado es esencial para maximizar la calidad de los datos de entrada, minimizar el ruido y optimizar el rendimiento de los modelos.
En la Ilustración 16 se muestra de forma esquemática el flujo completo de procesamiento implementado, desde la conversión inicial del audio hasta la aplicación de filtros personalizados.

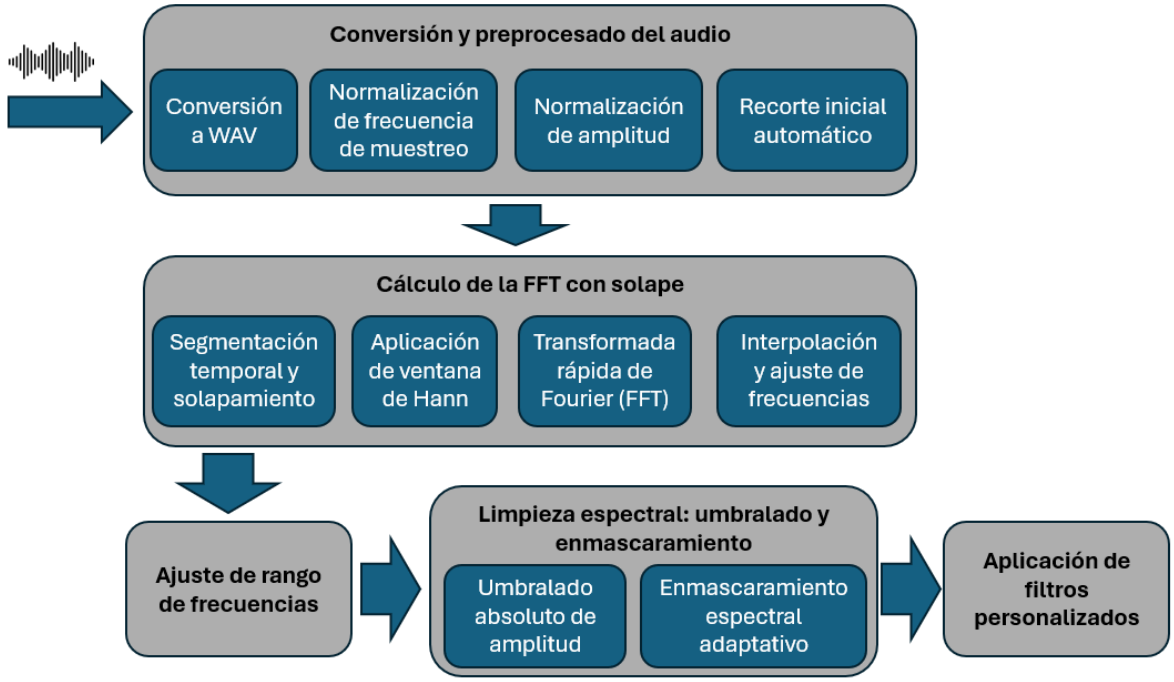


Ilustración 16

6.1 Conversión y preprocesado del audio

En este primer paso, los archivos de audio grabados son sometidos a un proceso de estandarización y limpieza que garantiza la homogeneidad de los datos a analizar. Las decisiones técnicas adoptadas aquí tienen un impacto directo en la precisión de los modelos posteriores, por lo que se detalla a continuación cada subproceso:

6.1.1 Formatos de entrada y conversión a WAV

Los archivos originales pueden estar en diversos formatos, principalmente MP3, debido a que las grabaciones iniciales se realizaron con grabadoras portátiles y teléfonos móviles. Sin embargo, el formato MP3 es un formato de compresión con pérdida (lossy) (Pohlmann, 2010), lo que implica pérdida de información en el espectro de frecuencias, especialmente en las bandas armónicas más altas, lo cual es crítico en el análisis musical donde los armónicos determinan el timbre y la identificación de la nota fundamental (Bosi, 2003).

Por este motivo, se ha optado por convertir todos los archivos al formato WAV, un formato sin compresión (PCM 16 bits, 44.1 kHz), que preserva la información completa de la señal (Sethares, 2005). Esta conversión se realiza automáticamente mediante la librería pydub en Python, que permite manipular diversos formatos de audio y realizar la exportación controlada a WAV (Brandenburg, 1999).

6.1.2 Normalización de frecuencia de muestreo

La frecuencia de muestreo se normaliza a 44.100 Hz (frecuencia estándar de audio digital profesional y la misma utilizada en los CDs de audio). Esta frecuencia asegura que se cumple el teorema de Nyquist para las frecuencias de interés, permitiendo capturar armónicos hasta aproximadamente 22.050 Hz, lo cual es más que suficiente para los instrumentos analizados (guitarra y flauta, cuyas frecuencias fundamentales no superan los 2 kHz y sus armónicos más relevantes quedan por debajo de 10 kHz). Para el remuestreo se emplea la librería librosa, que ofrece algoritmos de resampleo de alta calidad con técnicas de sinc-interpolation y antialiasing para evitar la distorsión en la conversión de frecuencias.

6.1.3 Normalización de amplitud

Las grabaciones originales pueden presentar diferentes niveles de ganancia y amplitud máxima, dependiendo de la distancia al micrófono, el volumen de ejecución y la sensibilidad del dispositivo de grabación. Por tanto, se aplica una normalización de amplitud que ajusta la señal para que su amplitud máxima sea uniforme en todos los audios, escalando la onda de forma que el valor absoluto máximo de la señal sea igual a 1. Este paso es crítico para evitar que diferencias de volumen afecten al cálculo posterior de amplitudes espectrales en el análisis de Fourier.
Este proceso se implementa recorriendo la señal tras la lectura del archivo WAV mediante librosa.load() y dividiendo toda la señal por su máximo absoluto.

6.1.4 Recorte inicial automático

En las grabaciones es habitual que existan fragmentos iniciales y finales que no contienen información musical útil. Estos pueden incluir ruido ambiente de la sala, el sonido del botón al iniciar o detener la grabación, o incluso pequeños golpeteos y ruidos al posicionar el instrumento o ajustar la grabadora. Estos tramos no aportan valor al análisis espectral, pero sí pueden introducir ruido, falsos picos de amplitud y dificultar la correcta detección de las frecuencias fundamentales.
Para evitar este problema, se ha implementado un algoritmo automático de detección de la zona útil de la grabación basado en la evolución de la amplitud de la señal:

- **Detección del pico máximo:** se localiza el instante temporal donde la señal alcanza su valor máximo absoluto de amplitud. Este punto, generalmente, coincide con el ataque de la nota, es decir, el momento en que se inicia la emisión estable del sonido musical.
- **Margen anterior de seguridad:** para asegurar que se captura completamente la fase de ataque y evitar cortes abruptos al inicio, se establece un margen de seguridad hacia atrás respecto al máximo detectado (por ejemplo, 200 ms

antes). Este margen permite incluir la transición desde el silencio hasta el inicio real de la nota.

- **Determinación del final efectivo:** tras el pico máximo, se analiza la evolución descendente de la amplitud hasta que esta cae por debajo de un umbral relativo preestablecido (por ejemplo, el 10% de la amplitud máxima). Este criterio garantiza que la zona capturada incluye únicamente la fase sostenida de la nota y que se excluyen las colas de sonido residual o el ruido de fondo que podría persistir después de la ejecución.
- **Recorte de la señal:** finalmente, el audio es truncado entre estos dos puntos, conservando únicamente el segmento donde la nota está claramente definida.

Este procedimiento ha sido implementado en Python empleando las librerías numpy (para el análisis vectorial eficiente de la señal) y scipy (para algunas operaciones de procesado digital de señales). Gracias a esta limpieza automática, se consigue que cada archivo de audio mantenga únicamente la porción musicalmente relevante de la grabación, eliminando artefactos no deseados.

Además de mejorar la calidad de la señal de entrada, este recorte tiene un impacto directo en la eficiencia computacional del sistema: al reducir la duración efectiva de cada archivo, disminuye el volumen de datos a procesar en las siguientes etapas (FFT por tramos, extracción de características, clasificación, etc.), lo que agiliza el tiempo de procesamiento sin pérdida de precisión.

6.2 Cálculo de la FFT con solape

En esta sección se describe en detalle el flujo completo de análisis espectral aplicado a cada señal de audio, desde la división temporal hasta la limpieza espectral final. El objetivo es extraer una representación frecuencia-tiempo precisa que permita alimentar el modelo de detección de notas musicales.

6.2.1 Segmentación temporal y solapamiento

La primera etapa del análisis espectral consiste en segmentar la señal de audio en fragmentos de duración fija, de modo que puedan analizarse de forma independiente las variaciones de frecuencia a lo largo del tiempo. Esta segmentación es necesaria porque las señales musicales, a diferencia de las señales estacionarias, presentan cambios continuos en frecuencia, amplitud y timbre que deben ser capturados con precisión para permitir la identificación de las notas ejecutadas.

Para este trabajo, se ha seleccionado un tamaño de ventana de **100 milisegundos** por defecto (aunque es adaptable por parte del usuario) debido a las siguientes razones técnicas:

- A una frecuencia de muestreo de 44100 Hz, cada ventana contiene 4410 muestras.
- Este tamaño de ventana proporciona un compromiso óptimo entre **resolución temporal y resolución frecuencial**:
    - La resolución de frecuencia (bin size) de la FFT se calcula en la Ecuación 3 como:

$$\Delta f = \frac{f_s}{N} = \frac{44100}{4410} = 10$$

Ecuación 3

- Una resolución de 10 Hz permite discriminar correctamente notas separadas por semitonos en el sistema temperado occidental, cuya separación es de aproximadamente 5.95% de frecuencia entre dos notas consecutivas (por ejemplo, entre 440 Hz y 466.16 Hz).
- Al mismo tiempo, una duración de 100 ms asegura que se capturen correctamente eventos musicales breves como **corcheas, semicorcheas o fusas**, permitiendo analizar la evolución de notas rápidas sin degradar la resolución temporal.

Además, para minimizar las pérdidas de información y mejorar la continuidad del análisis, se implementa un **solapamiento proporcional al tamaño de ventana**, generalmente expresado como un porcentaje respecto el tamaño total de la ventana. Esto significa que:

- Cada nueva ventana comienza tras un desplazamiento equivalente al (100−X)% del tamaño de ventana. Por ejemplo, si se utiliza un solapamiento del 50%, cada nueva ventana comienza cuando ha transcurrido el 50% de la duración de la ventana anterior.

El uso de solapamiento aporta dos ventajas fundamentales en el análisis espectral:

- **Mejora en la detección de transiciones rápidas**: dado que cada instante temporal es cubierto por múltiples ventanas superpuestas, se reduce el riesgo de perder información crítica durante los cambios bruscos de frecuencia o los ataques iniciales de las notas.
- **Suavizado de la evolución espectral**: el solapamiento garantiza que cada porción de la señal contribuya a varios análisis consecutivos, lo que reduce los artefactos provocados por las discontinuidades en los bordes de las ventanas (efectos de borde), mejorando así la estabilidad del espectrograma resultante.

En conjunto, el procedimiento de segmentación con solapamiento permite generar una representación tiempo-frecuencia de alta densidad, capaz de capturar tanto la estabilidad de notas largas como los rápidos cambios de notas en la interpretación musical.

El valor óptimo de solape se ajusta en función de las características específicas del instrumento, la velocidad de ejecución y la precisión temporal deseada.

6.2.2 Aplicación de ventana de Hann

Antes de aplicar la Transformada de Fourier a cada uno de los segmentos temporales de la señal, es necesario aplicar una función de ventana que suavice los extremos de cada segmento. En este trabajo, se emplea la **ventana de Hann** (también conocida como ventana de Hanning), ampliamente utilizada en análisis espectral debido a su buen compromiso entre resolución en frecuencia y reducción del leakage espectral.

Matemáticamente, la ventana de Hann se define en la Ecuación 4 como:

$$w[n] = 0.5 * \left(1 - \cos\left(\frac{2\pi n}{N-1}\right)\right), \;\; n = 0,1,2,\ldots,N-1$$

Ecuación 4

donde:

- w[n] es el coeficiente aplicado a la muestra n,
- N es el número total de muestras de la ventana.

Esta expresión genera una función de forma suavemente acampanada que vale cero en los extremos y alcanza un máximo en el centro de la ventana.

Cuando se aplica la FFT sobre una ventana de datos, se asume implícitamente que la señal es periódica dentro del marco de análisis. Si existen discontinuidades entre el inicio y el final de la ventana (lo cual es habitual cuando la ventana corta arbitrariamente una señal musical), dichas discontinuidades introducen componentes artificiales de alta frecuencia conocidas como **leakage espectral**. Este leakage dispersa energía hacia frecuencias no presentes en la señal original, dificultando la identificación precisa de los armónicos reales.

La aplicación de la ventana de Hann atenúa progresivamente las muestras cercanas a los bordes, minimizando estas discontinuidades. De este modo:

- Se reduce drásticamente el leakage espectral.
- Se mejora la separación entre picos de frecuencia adyacentes.
- Se incrementa la precisión en la estimación de la amplitud de cada armónico.

Cabe señalar que el uso de cualquier ventana introduce siempre un compromiso entre:

- **Resolución en frecuencia:** cuanto más suave es la ventana (mayor atenuación de los bordes), mayor es el ensanchamiento de los picos espectrales.
- **Reducción del leakage:** ventanas más suaves reducen mejor el leakage, pero sacrifican la definición exacta de frecuencias cercanas.

La ventana de Hann se considera en muchos casos el mejor compromiso general, especialmente para análisis de señales musicales donde los armónicos están claramente separados en frecuencia.

6.2.3 Transformada rápida de Fourier (FFT)

Una vez realizadas las ventanas y suavizados los segmentos temporales mediante la ventana de Hann, cada uno de estos fragmentos es procesado para obtener su contenido espectral. Para ello, se aplica la **Transformada Rápida de Fourier (FFT)** utilizando la función numpy.fft.fft().

Cada segmento de audio —que contiene un número fijo de muestras (por ejemplo, 4410 muestras en el caso de ventanas de 100 ms a 44.100 Hz)— es transformado desde el dominio temporal (amplitud de la señal a lo largo del tiempo) al dominio frecuencial (contenido de frecuencias presentes en ese tramo).

El objetivo principal de este paso es identificar las frecuencias y sus respectivas intensidades que componen la señal en ese instante.

Concretamente, el código ejecuta *espectro = np.fft.fft(ventana_hann).*

donde ventana_hann es el vector de muestras del segmento tras aplicar la ventana de Hann se obtiene lo siguiente:

- La salida de numpy.fft.fft() es un vector de números complejos.
- A partir de estos valores, se calcula el **módulo** de cada componente: *amplitudes = np.abs(espectro)*

Estas amplitudes representan la energía contenida en cada frecuencia analizada. Cabe destacar que solo se emplea la mitad de los coeficientes de la FFT, dado que la señal original es real y la FFT es simétrica (redundante a partir de N/2)

El último paso consiste en convertir los índices en frecuencias reales mediante la siguiente línea de código: *frecuencias = np.fft.fftfreq(N, d=1/f_s)[:N//2]* donde:

- N es el número de muestras de la ventana.
- f_s es la frecuencia de muestreo (44.100 Hz en este sistema).
- d es el periodo de muestreo, igual a 1/fs.

Este mapeo permite obtener directamente las frecuencias en hercios correspondientes a cada bin de la FFT.

6.2.4 Interpolación y ajuste de frecuencias

La Transformada Rápida de Fourier devuelve una representación discreta del contenido frecuencial de la señal, donde cada componente calculado corresponde a una frecuencia central fija, determinada por el tamaño de la ventana NNN y la frecuencia de muestreo Fs. Esto implica que la separación entre bins de frecuencia es constante.

Este escalonamiento discreto introduce una limitación: la frecuencia real de la nota tocada no tiene por qué coincidir exactamente con uno de los bins de la FFT, sino que puede situarse entre dos bins consecutivos. Este fenómeno, denominado **desajuste de bin**, puede provocar imprecisiones de varios Hz en la estimación de la frecuencia fundamental, afectando la posterior identificación de la nota musical, especialmente en el caso de notas próximas en afinación.

Para mitigar este efecto, en el sistema desarrollado se ha implementado una **interpolación parabólica de máximos** sobre los picos espectrales detectados. El procedimiento aplicado es el siguiente:

1. **Detección inicial del bin máximo:** Se identifica el índice k correspondiente al bin con mayor magnitud dentro del espectro de amplitudes calculado para cada ventana.
2. **Selección de vecinos adyacentes:** Se obtienen las amplitudes de los dos bins vecinos inmediatos, k−1 y k+1, que junto al bin central k conforman el conjunto de tres puntos necesarios para ajustar una parábola local.
3. **Ajuste parabólico:** Se estima el desplazamiento δ respecto al bin central mediante la Ecuación 5:

$$\delta = \frac{1}{2} * \frac{A_{k-1} - A_{k+1}}{A_{k-1} - 2A_k + A_{k+1}}$$

Ecuación 5

donde A_{k-1}, A_k, A_{k+1} son las amplitudes correspondientes a los tres bins.

4. **Cálculo de la frecuencia interpolada:** La frecuencia refinada se en la Ecuación 6 como:

$$f_{interpolada} = (k + \delta) * \Delta f$$

Ecuación 6

Esta interpolación permite alcanzar resoluciones de frecuencia sub-binaria sin necesidad de aumentar el tamaño de la ventana, lo cual preserva la resolución temporal del sistema.

6.3 Ajuste de rango de frecuencias

Una vez obtenida la representación espectral de cada segmento mediante la Transformada Rápida de Fourier, se aplica una restricción adicional sobre el rango de frecuencias analizado con el fin de optimizar tanto la precisión del sistema como su eficiencia computacional.

El sistema ha sido diseñado para analizar instrumentos melódicos como flauta dulce, saxofón y clarinete como instrumentos polifónicos como la guitarra clásica y el violín. Estos instrumentos presentan rangos frecuenciales bien definidos:

- **Guitarra clásica:**
  - Fundamental mínima: ~82 Hz (E2, sexta cuerda al aire).
  - Fundamental máxima: ~880 Hz (A5, posiciones altas).
- **Flauta dulce:**
  - Fundamental mínima: ~262 Hz (C4).
  - Fundamental máxima: ~2093 Hz (C7).
- **Saxofón (alto, tenor y soprano):**
  - Fundamental mínima: ~110 Hz (A2 en saxofón tenor).
  - Fundamental máxima: ~1700 Hz (aprox. E6 en saxofón soprano).
- **Violín:**
  - Fundamental mínima: ~196 Hz (G3, cuerda más grave).
  - Fundamental máxima: ~3136 Hz (D7).
- **Clarinete:**
  - Fundamental mínima: ~147 Hz (D3).
  - Fundamental máxima: ~2093 Hz (C7), aunque con armónicos característicos más allá.

Además de las frecuencias fundamentales, el sistema debe analizar los armónicos de cada instrumento, ya que estos son esenciales para la correcta identificación del timbre y la detección precisa de la nota fundamental.

Las ventajas principales de limitar el rango de análisis son:

- **Reducción de ruido no informativo:** Las frecuencias por debajo de 80 Hz contienen principalmente ruido de fondo, vibraciones mecánicas o artefactos de grabación, que no aportan contenido musical útil.
- **Supresión de contenido ultrasónico:** Las frecuencias por encima de 2000 Hz contienen armónicos de orden muy alto cuya energía es habitualmente muy

reducida, y que pueden ser enmascarados por ruido o interferencias, complicando la detección fiable.

- **Optimización computacional:** Limitar el análisis a las frecuencias relevantes permite reducir el tamaño de los vectores de características y acelerar los procesos de extracción de características, entrenamiento y predicción.

El filtrado de frecuencias se realiza de forma directa sobre los resultados de la FFT, seleccionando únicamente los bins cuya frecuencia se encuentra dentro del rango de interés:

frecuencias_validas = (frecuencias >= 80) & (frecuencias <= 2000)
amplitudes_filtradas = amplitudes[frecuencias_validas]
frecuencias_filtradas = frecuencias[frecuencias_validas]

De esta forma, el sistema trabaja exclusivamente con el contenido espectral realmente relevante desde el inicio del análisis, optimizando tanto la estabilidad como el rendimiento global del procesamiento.

6.4 Limpieza espectral: umbralado y enmascaramiento

Tras el cálculo de la Transformada de Fourier y la selección del rango de frecuencias de interés, la señal espectral obtenida sigue presentando habitualmente componentes de baja energía. Estas componentes no forman parte de la señal musical relevante, sino que provienen de:

- Ruido ambiente residual durante la grabación.
- Imperfecciones instrumentales (ligeros ruidos mecánicos, soplidos, roces).
- Artefactos numéricos propios del procesamiento digital.

Si no se eliminan, estas pequeñas energías pueden interferir en la detección de la frecuencia fundamental, afectar la estimación de armónicos y, en consecuencia, perjudicar la identificación correcta de las notas.

6.4.1 Umbralado absoluto de amplitud

Como primera medida de limpieza espectral, se aplica un **umbral fijo de amplitud** sobre las magnitudes obtenidas tras la FFT. Este umbral se calcula relativo al máximo de amplitud de cada ventana:

- Todas aquellas componentes cuya amplitud esté por debajo de un determinado umbral — en este caso, típicamente **-60 dB respecto al máximo** — son eliminadas directamente del análisis.
- Este valor de -60 dB ha sido ajustado experimentalmente para maximizar el equilibrio entre eliminación de ruido y conservación de armónicos relevantes en los instrumentos considerados.

Este filtrado básico permite eliminar el grueso de las componentes residuales de ruido sin afectar a las frecuencias musicales dominantes.

6.4.2 Enmascaramiento espectral adaptativo

Como segunda capa de limpieza, puede aplicarse un **enmascaramiento adaptativo**. Esta técnica consiste en:

- Comparar cada frecuencia con las frecuencias dominantes dentro de la misma ventana.
- Eliminar componentes que, aun superando el umbral absoluto, presentan una energía muy inferior respecto a los picos principales de la señal.
- Este enfoque es útil para evitar falsos armónicos generados por ruido de fondo estructurado o artefactos de grabación.

6.5 Aplicación de filtros personalizados

Una vez realizada la limpieza espectral general (umbralado absoluto y enmascaramiento básico), el sistema incorpora un módulo de filtrado específico para cada instrumento, diseñado para eliminar armónicos no relevantes y estabilizar la representación frecuencial antes de pasar a la fase de detección de notas.

Este filtrado especializado ha sido implementado en el script propio *makeFiltro.py*, el cual permite adaptar dinámicamente el perfil de filtrado en función del instrumento analizado. Los objetivos generales de este tipo de filtrado son:

- **Eliminar armónicos irrelevantes:** Algunos armónicos de orden muy alto aportan poca información discriminativa para la identificación de la nota fundamental, pero pueden introducir ruido en los modelos de clasificación si no se controlan.
- **Controlar picos anómalos de amplitud:** Las grabaciones pueden contener transitorios breves (golpes de cuerda, soplidos, pequeñas imperfecciones) que generan picos espurios de amplitud. Estos picos pueden confundir al sistema si no son estabilizados.
- **Refinar la energía espectral:** Se eliminan aquellas frecuencias que, aun permaneciendo tras el umbralado anterior, tienen niveles de energía insuficientes para ser consideradas componentes válidas de la señal armónica principal.

Para cada instrumento considerado, el sistema define de forma diferenciada:

- El número máximo de armónicos a conservar
- El margen de tolerancia frecuencial permitido alrededor de cada armónico, expresado como un porcentaje ε% respecto al armónico teórico.
- Los umbrales de energía y tolerancia de fluctuación utilizados en los suavizados posteriores.

Estos parámetros son configurables a través de los ficheros de parámetros del sistema, lo que permite adaptar el comportamiento de filtrado a las características armónicas específicas de cada instrumento.

Tras la limpieza general del espectro, el sistema implementa un filtrado armónico adaptativo desarrollado específicamente en el módulo *makeFiltro.py*. Este filtro permite ponderar cada bin de frecuencia en función de su proximidad a los armónicos teóricos de la frecuencia fundamental candidata f0f_0f0, aplicando distintas reglas según la desviación.

Para cada armónico teórico $f_h = h * f_0$, se calcula para cada bin f_k su desviación relativa siguiendo la Ecuación 7:

$$\delta_{h,k} = \frac{|f_k - f_h|}{f_h}$$

Ecuación 7

El filtro ajusta las amplitudes espectrales de la siguiente forma:

- **Si el bin corresponde exactamente a un armónico (dentro de un margen mínimo de tolerancia absoluta):** Se aplica un factor de refuerzo multiplicativo G, configurado como parámetro específico de cada instrumento, siguiendo la Ecuación 8:

$$A'_k = A_k * G$$

Ecuación 8

Este refuerzo potencia la contribución de los armónicos correctamente alineados con la serie armónica teórica.

- **Si la desviación relativa $\delta_{h,k}$ se encuentra dentro de un margen de tolerancia ε:** La amplitud se conserva sin modificaciones siguiendo la Ecuación 9:

$$A'_k = A_k$$

Ecuación 9

- **Si la desviación relativa supera el margen $\delta_{h,k}$ > ε:** Se aplica una penalización progresiva mediante decaimiento exponencial siguiendo la Ecuación 10:

$$A'_k = A_k * e^{(-\alpha * \frac{\delta_{h,k} - \varepsilon}{\varepsilon})}$$

Ecuación 10

donde α es el parámetro que controla la agresividad de la penalización exponencial fuera del margen de tolerancia.

Finalmente, si la nueva amplitud penalizada A'_k cae por debajo de un umbral mínimo A_{min}, se trunca a cero tal y como se observa en la Ecuación 11:

$$A'_k = 0 \; si \; A'_k < A_{min}$$

Ecuación 11

Después de ejecutar el script queda un filtro en frecuencia siguiendo la estructura mostrada en la Ilustración 17:

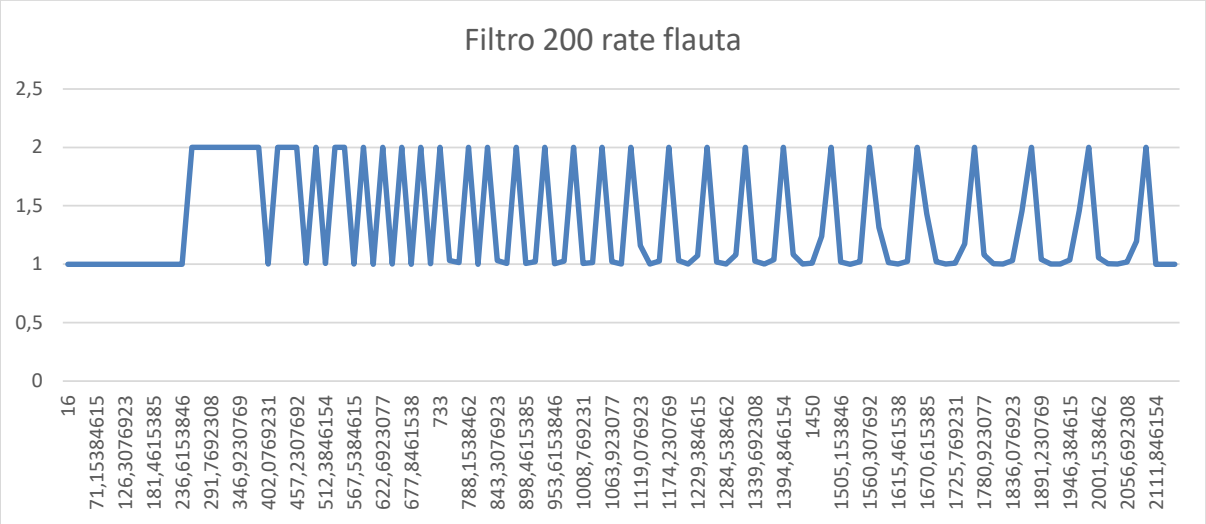


Ilustración 17

Esta estrategia permite:

- Reforzar los armónicos bien alineados.
- Conservar los armónicos dentro de un margen razonable.
- Penalizar gradualmente los armónicos desviados.
- Eliminar definitivamente los armónicos que resultan claramente irrelevantes.

La aplicación de este filtrado adaptativo mejora notablemente la estabilidad de la extracción armónica y reduce la probabilidad de detección de falsos positivos en presencia de ruido, pequeñas desafinaciones o armónicos parásitos.

Capítulo 7. Procesamiento de Instrumentos

En el desarrollo de sistemas de transcripción musical automática es fundamental distinguir claramente entre el tratamiento de instrumentos melódicos y polifónicos, ya que presentan características acústicas y desafíos de análisis espectral sustancialmente diferentes. Esta diferenciación no es solo conceptual, sino que condiciona de forma directa los algoritmos de detección, los modelos de extracción de características y la arquitectura global del sistema de análisis.

Los instrumentos melódicos (como la flauta, el clarinete, el saxofón, el trombón, la trompeta o la voz cuando se canta una única línea melódica) están caracterizados por emitir exclusivamente una única nota a la vez en cada instante de tiempo. Desde el punto de vista espectral, esto implica que el contenido frecuencial de la señal se encuentra dominado por una única frecuencia fundamental f_0,acompañada de su correspondiente serie armónica, compuesta por múltiplos enteros de dicha fundamental: $2f_0$, $3f_0$, $4f_0$ ... La energía espectral está por tanto claramente concentrada y estructurada, y el análisis puede basarse en localizar el máximo espectral, analizar las relaciones de sus armónicos o incluso modelar matemáticamente las series armónicas esperadas para cada instrumento.

Además, en los instrumentos melódicos los armónicos suelen estar razonablemente estables en fase, amplitud y posición frecuencial, salvo pequeñas fluctuaciones debidas a la interpretación humana (como vibratos, glissandos o ligeras variaciones de afinación), lo que permite el empleo de filtros armónicos adaptativos, interpolación de máximos, y algoritmos de refinamiento de pitch relativamente robustos.

Por el contrario, los instrumentos polifónicos (como la guitarra, el piano, el violín o el arpa cuando ejecutan acordes o múltiples voces simultáneas) presentan una complejidad considerablemente mayor. Estos instrumentos pueden generar varias frecuencias fundamentales activas al mismo tiempo, cada una con su respectiva serie armónica, lo que provoca inevitablemente una superposición de múltiples armónicos en el dominio frecuencial. Esta superposición genera interferencias y solapamientos entre armónicos de distinto orden, haciendo que, por ejemplo, el segundo armónico de una nota pueda coincidir muy cerca de la frecuencia fundamental de otra, o que los armónicos superiores compartan bins en el espectro de la FFT. Además, algunos armónicos de distinto origen pueden enmascararse o reforzarse mutuamente, dificultando el aislamiento de cada componente sonora de forma directa.

Esta problemática convierte la detección automática de notas en los instrumentos polifónicos en un problema de separación de fuentes (source separation), donde ya no basta con analizar individualmente cada pico espectral, sino que es necesario aplicar técnicas avanzadas como la descomposición multipitch simultánea, el modelado estadístico de las envolventes armónicas, la factorización espectral no negativa (NMF), o métodos más recientes basados en redes neuronales de separación armónica.

Por este motivo, los algoritmos implementados en este trabajo establecen un pipeline diferenciado en función del tipo de instrumento. Para los instrumentos melódicos, el sistema aplica técnicas basadas en análisis espectral armónico refinado, que aprovechan la alineación de los armónicos y permiten filtrar armónicos desintonizados, penalizar desplazamientos de frecuencia mediante ponderaciones exponenciales, y reforzar selectivamente los armónicos dominantes. Para los instrumentos polifónicos, sin embargo, sería necesario incorporar módulos adicionales que permitan descomponer correctamente los espectros complejos resultantes de la interacción simultánea de varias notas, tarea que queda identificada como una línea de desarrollo futura.

Esta distinción conceptual y técnica resulta esencial en cualquier sistema práctico de transcripción musical asistida por IA, ya que condiciona tanto la dificultad del problema como la arquitectura algorítmica más adecuada para su resolución.

7.1 Instrumentos Melódicos

Una vez se ha realizado el preprocesado del audio, explicado en detalle en el capítulo anterior (incluyendo la conversión a WAV, el recorte temporal, el filtrado por espectrograma y la segmentación en ventanas temporales solapadas), el sistema procede a la identificación de notas musicales dominantes en cada uno de los segmentos de audio generados. Este proceso constituye el núcleo de la predicción de notas en los instrumentos melódicos.

Análisis Espectral en Cada Ventana

Para cada ventana de audio generada, se aplica la Transformada Rápida de Fourier (FFT) con el objetivo de convertir la señal desde el dominio temporal al dominio frecuencial, permitiendo así analizar las componentes armónicas presentes en ese intervalo de tiempo.

La FFT transforma la serie de muestras de amplitud de la ventana (por ejemplo, un segmento de 200 ms de duración) en un vector complejo donde cada componente representa una frecuencia específica y su correspondiente amplitud y fase. A partir de este vector complejo se obtiene el módulo de la FFT, que refleja la magnitud de energía presente en cada frecuencia.

Formalmente, si la ventana contiene N muestras, el resultado de la FFT es un vector de N componentes, donde cada índice k está asociado a una frecuencia siguiendo la Ecuación 12:

$$f_k = \frac{k * f_s}{N}$$

Ecuación 12

Donde f_s es la frecuencia de muestreo (por ejemplo, 44.1 kHz).

Sin embargo, dado que la señal original es real, la FFT presenta simetría conjugada, por lo que sólo se consideran las frecuencias positivas (hasta $f_s/2$) al analizar las amplitudes útiles del espectro. Por tanto, se eliminan todas las frecuencias negativas, quedándonos únicamente con el rango físicamente interpretable.

Como resultado de esta etapa, para cada ventana de audio obtenemos un vector reducido de frecuencias relevantes y sus correspondientes amplitudes, que posteriormente serán utilizados para el proceso de detección de picos, identificación de fundamentales e interpolación de notas.

Identificación de Picos de Frecuencia

Una vez calculado el espectro de frecuencias de cada ventana temporal mediante la Transformada Rápida de Fourier (FFT), el siguiente paso en el pipeline de análisis consiste en identificar las frecuencias dominantes que potencialmente representan las notas musicales interpretadas en dicha ventana.

Este proceso es crucial, ya que permite aislar los máximos locales de energía dentro del espectro, discriminando entre las frecuencias fundamentales y los armónicos, así como eliminando posibles fluctuaciones de ruido.

Aplicación del algoritmo find_peaks

En este proyecto, la detección de picos se realiza utilizando la función find_peaks de la librería **SciPy**.

El algoritmo actúa directamente sobre el vector de amplitudes calculado previamente para cada ventana. Su objetivo es identificar aquellos puntos en el espectro en los que se produce un máximo relativo de amplitud. Sin embargo, para evitar falsas detecciones debidas a ruido o pequeñas oscilaciones del espectro, se introducen dos parámetros fundamentales de control:

- **Prominencia mínima (implícita por defecto en el código):** Aunque en esta implementación no se especifica directamente el parámetro prominence, se podría incluir si fuera necesario para entornos más ruidosos. Este parámetro filtra los picos cuya diferencia de altura respecto a los valles adyacentes no alcanza un umbral mínimo, eliminando así pequeños picos secundarios.
- **Distancia mínima entre picos (distance=50):** Este parámetro está fijado en el código y fuerza a que los picos detectados estén suficientemente separados unos de otros en el eje de frecuencia. De este modo, se evita que el algoritmo detecte múltiples picos cercanos dentro de lo que en realidad corresponde a un único armónico. El valor 50 está expresado en unidades de índice de la FFT (no en Hz directamente), por lo que su valor efectivo depende del tamaño de la ventana y de la frecuencia de muestreo.

Conversión de picos a frecuencias

Una vez localizados los índices de los picos, estos se transforman a unidades de frecuencia (Hz), teniendo en cuenta la tasa de muestreo (rate) y la longitud de la FFT siguiendo la Ecuación 13:

$$f = \frac{(k+1) * 100}{rate}$$

Ecuación 13

Este ajuste está implementado en la función find_peak_frequencies, que devuelve un listado de frecuencias correspondientes a los máximos detectados.

Representación gráfica del espectro de una ventana

Para ilustrar visualmente este proceso, se incluye la Ilustración 18, donde se muestra el espectro de frecuencias calculado para una de las ventanas del audio procesado. En la gráfica se pueden observar claramente:

- Las frecuencias analizadas (eje horizontal en Hz).
- Las amplitudes obtenidas en cada frecuencia (eje vertical).
- Los picos de frecuencia detectados los cuales representan las posibles frecuencias fundamentales candidatas a ser identificadas como notas musicales.

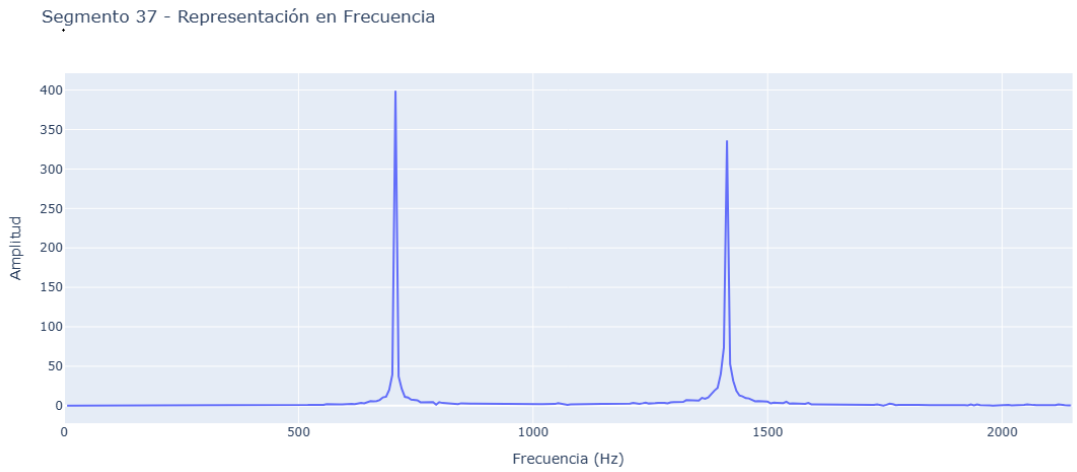


Ilustración 18

Eliminación de Armónicos

Una vez identificados los picos principales de frecuencia en el espectro de cada ventana, es necesario aplicar un proceso de filtrado de armónicos con el objetivo de conservar exclusivamente las **frecuencias fundamentales** de cada nota. Esta etapa resulta esencial, ya que los instrumentos melódicos analizados en este proyecto generan, además de la frecuencia fundamental de cada nota, múltiples armónicos que corresponden a múltiplos enteros de dicha frecuencia.

Si estos armónicos no fueran eliminados adecuadamente, podrían provocar que el sistema asignase incorrectamente notas superiores, confundiendo armónicos de órdenes elevados con nuevas notas musicales.

El procedimiento detallado que sigue el algoritmo es el siguiente:

1. **Ordenación previa de frecuencias:** Las frecuencias obtenidas tras la detección de picos se ordenan de menor a mayor. Este paso es fundamental para que las comparaciones siempre comiencen evaluando la frecuencia más baja como candidata a frecuencia fundamental. A partir de ahí, el resto de las frecuencias se evalúan en función de si pueden considerarse armónicos de las anteriores.
2. **Comparación iterativa entre frecuencias:** Para cada frecuencia f_i de la lista (a partir de la segunda), se compara con todas las frecuencias anteriores f_j. En cada comparación se calcula el cociente R mediante la Ecuación 14 :

$$R = \frac{f_i}{f_j}$$

Ecuación 14

Este cociente permite evaluar si f_i es aproximadamente un múltiplo entero de f_j.

3. **Aproximación al múltiplo entero más cercano:** El cociente R se redondea al entero más cercano, obteniendo así el posible orden armónico (por ejemplo, 2, 3, 4...).
4. **Verificación del margen de tolerancia:** Para evitar rechazos indebidos por pequeñas imprecisiones numéricas (producto de la discretización de la FFT o de ligeras variaciones en la ejecución musical), se admite un margen de tolerancia relativa configurable. Por defecto, este margen está fijado en un 10% de error relativo (error_tolerance = 0.1), lo que permite cierta flexibilidad en la detección.

Si esta condición se cumple, la frecuencia f_i se considera armónico de f_j y, por tanto, se descarta.

5. **Selección final de fundamentales:** Si una frecuencia f_i no se clasifica como armónico respecto a ninguna frecuencia anterior, se conserva como frecuencia fundamental.

De esta forma, se garantiza que en el conjunto de frecuencias resultantes únicamente se mantengan aquellas que muy probablemente correspondan a frecuencias fundamentales.

Un ejemplo de todo esto podría ser si tras aplicar la detección de picos en una ventana, se obtienen los valores de frecuencia de la Tabla 2:

Frecuencia detectada (Hz)	Observación
440	Fundamental (nota A4)
880	Armónico de segundo orden (2 × 440 Hz)
1320	Armónico de tercer orden (3 × 440 Hz)

Tabla 2

El filtrado de armónicos identificará que tanto 880 Hz como 1320 Hz son múltiplos enteros de 440 Hz, por lo que ambos serán descartados, conservándose únicamente la frecuencia fundamental de 440 Hz.

La lista de frecuencias fundamentales obtenida a través de este proceso es la que posteriormente se emplea para la asignación definitiva de notas musicales, mediante la comparación con los tramos de frecuencias preestablecidos para cada nota.

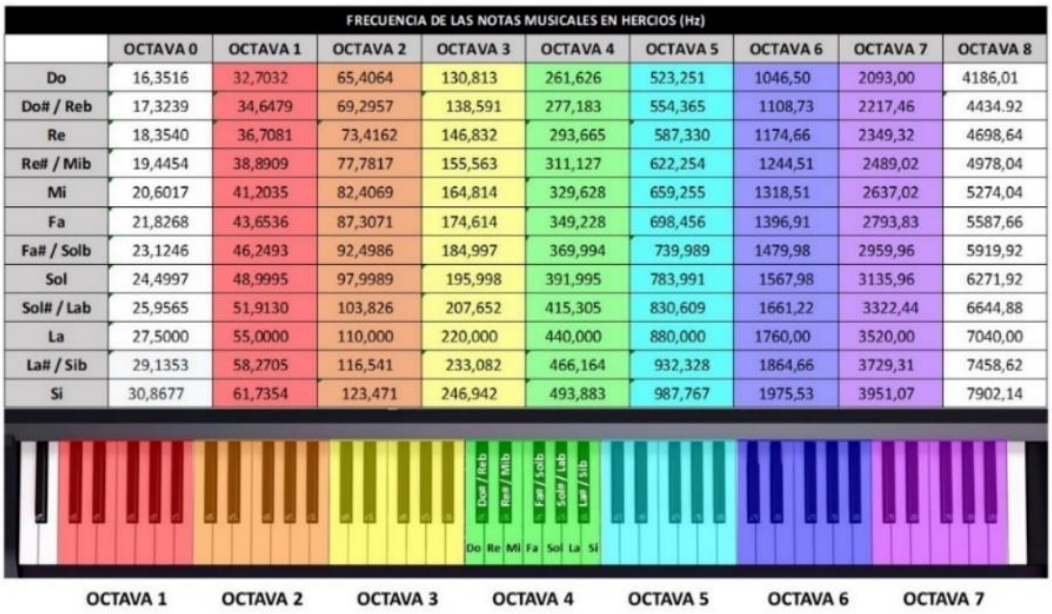
Este mecanismo de eliminación de armónicos permite aumentar de forma significativa la robustez del sistema, especialmente en la detección de instrumentos reales, donde el contenido armónico es elevado y puede inducir a errores si no se aplica este filtrado de manera previa.

Asignación de Nota

Una vez obtenidas las frecuencias fundamentales de cada ventana de audio (tras la detección de picos y el filtrado de armónicos), el siguiente paso consiste en determinar a qué nota musical concreta corresponde cada una de estas frecuencias. Este proceso es lo que permite transformar el análisis espectral en una secuencia de notas que posteriormente constituirá la partitura digital.

Antes de poder realizar la asignación, es necesario disponer de un conjunto de tramos de frecuencia que definan los márgenes de tolerancia de cada nota musical. Estos tramos son generados en las fases previas del programa. El proceso de construcción de estos tramos sigue los siguientes pasos:

- Se parte de un diccionario completo de frecuencias estándar de todas las notas musicales siguiendo la Ilustración 19:



FRECUENCIA DE LAS NOTAS MUSICALES EN HERCIOS (Hz)									
	OCTAVA 0	OCTAVA 1	OCTAVA 2	OCTAVA 3	OCTAVA 4	OCTAVA 5	OCTAVA 6	OCTAVA 7	OCTAVA 8
Do	16,3516	32,7032	65,4064	130,813	261,626	523,251	1046,50	2093,00	4186,01
Do# / Reb	17,3239	34,6479	69,2957	138,591	277,183	554,365	1108,73	2217,46	4434.92
Re	18,3540	36,7081	73,4162	146,832	293,665	587,330	1174,66	2349,32	4698,64
Re# / Mib	19,4454	38,8909	77,7817	155,563	311,127	622,254	1244,51	2489,02	4978,04
Mi	20,6017	41,2035	82,4069	164,814	329,628	659,255	1318,51	2637,02	5274,04
Fa	21,8268	43,6536	87,3071	174,614	349,228	698,456	1396,91	2793,83	5587,66
Fa# / Solb	23,1246	46,2493	92,4986	184,997	369,994	739,989	1479,98	2959,96	5919,92
Sol	24,4997	48,9995	97,9989	195,998	391,995	783,991	1567,98	3135,96	6271,92
Sol# / Lab	25,9565	51,9130	103,826	207,652	415,305	830,609	1661,22	3322,44	6644,88
La	27,5000	55,0000	110,000	220,000	440,000	880,000	1760,00	3520,00	7040,00
La# / Sib	29,1353	58,2705	116,541	233,082	466,164	932,328	1864,66	3729,31	7458,62
Si	30,8677	61,7354	123,471	246,942	493,883	987,767	1975,53	3951,07	7902,14

Ilustración 19

- Para cada nota, se calcula un tramo de tolerancia en torno a su frecuencia central, aplicando un porcentaje (normalmente un 25%) de la distancia a las notas adyacentes. Esto permite manejar variaciones de afinación, ligeras desviaciones interpretativas o errores inherentes al procesamiento digital.

- El resultado es un conjunto de intervalos de frecuencia asociados a cada nota, que cubren de forma continua todo el espectro auditivo considerado para el instrumento en cuestión.

Una vez definidos estos tramos, se procede a la asignación efectiva de las notas mediante la comparación de las frecuencias fundamentales detectadas con los intervalos definidos.

Para cada frecuencia $f_{detectada}$ obtenida en una ventana:

1. Se recorre la lista de tramos disponibles.
2. Para cada tramo asociado a una nota concreta, se verifica si la frecuencia $f_{detectada}$ se encuentra dentro de los límites inferior y superior del tramo:

$$Si\ f_{inferior} < f_{detectada} < f_{superior} \rightarrow Se\ asgina\ la\ frecuencia\ correspondiente$$

3. Si se encuentra un tramo que contiene la frecuencia, la asignación de la nota es directa.
4. Si ninguna de las frecuencias detectadas cae dentro de los tramos predefinidos, no se asigna ninguna nota.

Comprobación adicional mediante umbral de amplitud

De forma complementaria, antes de confirmar la asignación de nota, se introduce un criterio adicional basado en la energía de la ventana:

- Se evalúa la amplitud máxima de la señal procesada en la ventana tras la eliminación de componentes débiles.
- Si esta amplitud máxima no supera un umbral mínimo predefinido (ampliMin), se considera que la ventana no contiene ninguna nota claramente detectable.
- En ese caso, se asigna el código "ZZZ" para indicar la ausencia de nota musical en ese fragmento de tiempo.

En la Ilustración 20 se observa un primer pico principal se encuentra en 660 Hz, que corresponde a la nota EN5. Además, aparecen dos picos adicionales: uno en 1320 Hz, que es exactamente el doble de la frecuencia fundamental (660 Hz × 2), indicando un armónico de segundo orden asociado a la nota EN6; y otro pico en 1973 Hz, cuya relación respecto a la frecuencia fundamental es:

$$\frac{1973}{660} = 2.989 \approx 3$$

Este cociente puede aproximarse a 3, identificando así un armónico de tercer orden correspondiente a la nota EN7.

Por tanto, el análisis de los picos detectados permite concluir que la frecuencia fundamental dominante en este espectro es 660 Hz (EN5), siendo los picos restantes armónicos derivados de esta frecuencia principal.

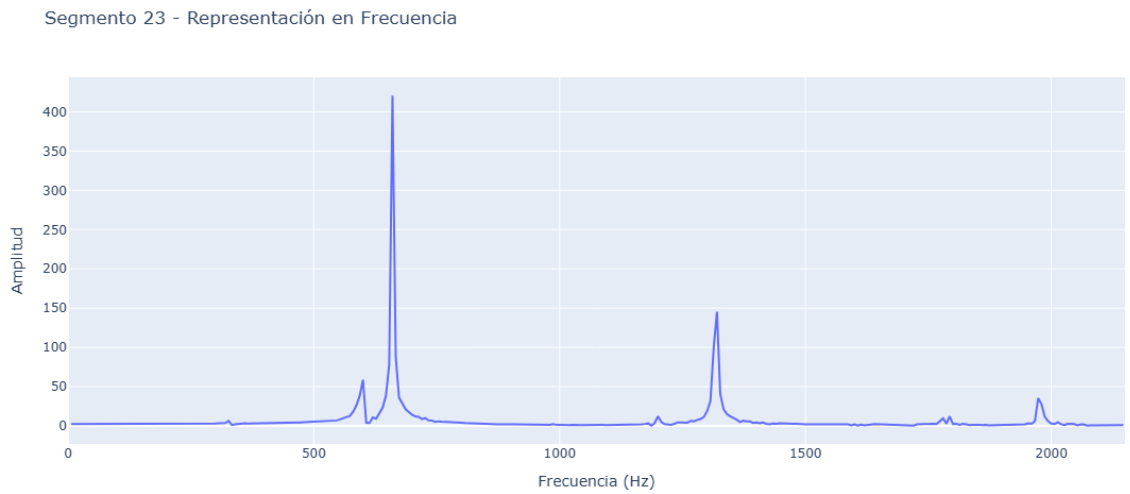


Ilustración 20

Consideración de continuidad melódica

Además, cuando existen varias frecuencias fundamentales dentro de una ventana, el sistema gestiona la asignación de forma conservadora:

- Si solo se detecta una frecuencia fundamental, se asigna directamente la nota correspondiente.
- Si existen múltiples frecuencias candidatas en la ventana (situación infrecuente pero posible), el sistema aplica una regla de continuidad:
    - Si es la primera ventana, asigna "ZZZ" en caso de ambigüedad.
    - Si existen ventanas anteriores ya clasificadas, reutiliza la última nota asignada como medida de estabilidad melódica.

Construcción de la Partitura Secuencial

Una vez ejecutado el proceso de detección y asignación de notas en cada ventana de análisis, el sistema construye de forma automática la partitura secuencial a partir de los resultados obtenidos.

El sistema ha sido diseñado para trabajar con una estrategia ventana a ventana, es decir, procesa el audio dividiéndolo en múltiples segmentos temporales de igual duración y con solape parcial configurable. Cada una de estas ventanas es analizada de manera independiente, obteniéndose una predicción de nota (o ausencia de nota) para dicho intervalo temporal.

Acumulación secuencial de notas

El núcleo de esta etapa reside en la construcción iterativa de la lista partitura, que almacena la secuencia de notas resultantes. Para cada ventana procesada se asigna la nota correspondiente y se añade a la variable lista *partitura*.

Este mecanismo garantiza que la secuencia final de notas mantenga el orden temporal original del audio, generando así una representación cronológica del contenido musical interpretado.

Resolución temporal de la partitura

La granularidad temporal de la partitura generada está directamente relacionada con los parámetros de segmentación definidos previamente:

- **Tamaño de ventana (rate)**: determina la duración de cada segmento.
- **Porcentaje de solape (solape)**: define cuánto se solapan dos ventanas consecutivas.

Por ejemplo, con un tamaño de ventana de 200 ms y un solape del 90 %, el sistema genera una predicción de nota cada 20 ms, lo que permite capturar cambios de notas con alta resolución temporal.

Tratamiento de silencios o incertidumbres

En los casos en que no se detecta ninguna frecuencia válida (por debajo del umbral de amplitud definido), se asigna el código "ZZZ". Este marcador permite:

- Identificar los momentos de silencio o ruido en la ejecución.
- Evitar falsas asignaciones de nota en ventanas sin contenido armónico suficiente.
- Facilitar posteriormente el posprocesamiento, donde podrían unificarse tramos consecutivos de "ZZZ" como pausas musicales.

A modo ilustrativo, una salida típica del sistema podría ser la que se muestra en la Ilustración 21:

```
==> Resultado:
['ZZZ', 'ZZZ', 'ZZZ', 'ZZZ', 'ZZZ', 'ZZZ', 'ZZZ', 'ZZZ', 'ZZZ', 'ZZZ', 'CN4', 'CN4', 'CN4', 'CN4', 'CN4', 'CN4', 'C#4', 'C#4', 'C#4', 'C#4'
, 'C#4', 'C#4', 'DN4', 'DN4', 'DN4', 'DN4', 'DN4', 'DN4', 'D#4', 'D#4', 'D#4', 'D#4', 'D#4', 'D#4', 'D#4', 'EN4', 'EN4', 'EN4', 'EN4', 'EN4
', 'FN4', 'FN4', 'FN4', 'FN4', 'FN4', 'FN4', 'F#4', 'F#4', 'F#4', 'F#4', 'F#4', 'F#4', 'F#4', 'ZZZ', 'ZZZ', 'GN4', 'GN4', 'GN4', 'GN4', 'GN4', 'GN
4', 'G#4', 'G#4', 'G#4', 'G#4', 'G#4', 'G#4', 'AN4', 'AN4', 'AN4', 'AN4', 'AN4', 'AN4', 'A#4', 'A#4', 'A#4', 'A#4', 'A#4', 'BN4', 'BN4', 'B
N4', 'BN4', 'BN4', 'CN5', 'CN5', 'CN5', 'CN5', 'CN5', 'CN5', 'C#5', 'C#5', 'C#5', 'C#5', 'ZZZ', 'ZZZ', 'DN5', 'DN5', 'DN5', 'DN5', 'DN5', '
DN5', 'D#5', 'D#5', 'D#5', 'D#5', 'D#5', 'D#5', 'EN5', 'EN5', 'EN5', 'EN5', 'EN5', 'EN5', 'EN5', 'EN5', 'FN5', 'FN5', 'FN5', 'FN5', 'F#5', 'F#5', 'F#5',
'F#5', 'F#5', 'F#5', 'F#5', 'GN5', 'GN5', 'GN5', 'ZZZ', 'ZZZ', 'G#5', 'G#5', 'G#5', 'G#5', 'G#5', 'G#5', 'AN5', 'AN5', 'AN5', 'AN5', 'AN5',
 'A#5', 'A#5', 'A#5', 'A#5', 'A#5', 'A#5', 'BN5', 'BN5', 'BN5', 'BN5', 'BN5', 'CN6', 'CN6', 'CN6', 'CN6', 'CN6', 'C#6', 'C#6', 'C#6', 'C#6'
, 'C#6', 'ZZZ', 'DN6', 'DN6', 'DN6', 'DN6', 'DN6', 'DN6', 'D#6', 'D#6', 'D#6', 'D#6', 'D#6', 'D#6', 'EN6', 'EN6', 'EN6', 'EN6', 'EN6', 'FN6', 'FN6
', 'FN6', 'FN6', 'FN6', 'FN6', 'F#6', 'F#6', 'F#6', 'F#6', 'ZZZ', 'GN6', 'GN6', 'GN6', 'GN6', 'G#6', 'G#6', 'G#6', 'G#6', 'G#6', 'G#6', 'G#
6', 'AN6', 'AN6', 'AN6', 'AN6', 'AN6', 'AN6', 'A#6', 'A#6', 'A#6', 'A#6', 'ZZZ', 'BN6', 'BN6', 'BN6', 'CN7', 'CN7', 'ZZZ', 'ZZZ', 'C#7', 'C
#7', 'C#7', 'C#7', 'C#7', 'C#7', 'ZZZ']
```

Ilustración 21

Esta secuencia representa la evolución temporal de las notas ejecutadas a lo largo del fragmento de audio analizado, incluyendo momentos de ausencia de nota.

Finalmente, la lista partitura generada se emplea directamente como entrada para el módulo de representación gráfica de la partitura, que transforma esta secuencia secuencial de notas en una visualización tipo piano-roll, facilitando así la interpretación visual del resultado por parte del usuario.

Resumen del proceso

La Ilustración 22 muestra el esquema general del proceso seguido para la identificación de notas musicales a partir del análisis de señales de audio de instrumentos melódicos. El flujo se organiza de manera secuencial, aplicando distintos módulos de procesamiento desde la segmentación inicial del audio hasta la construcción final de la partitura. A continuación, se describen detalladamente cada una de las etapas representadas:

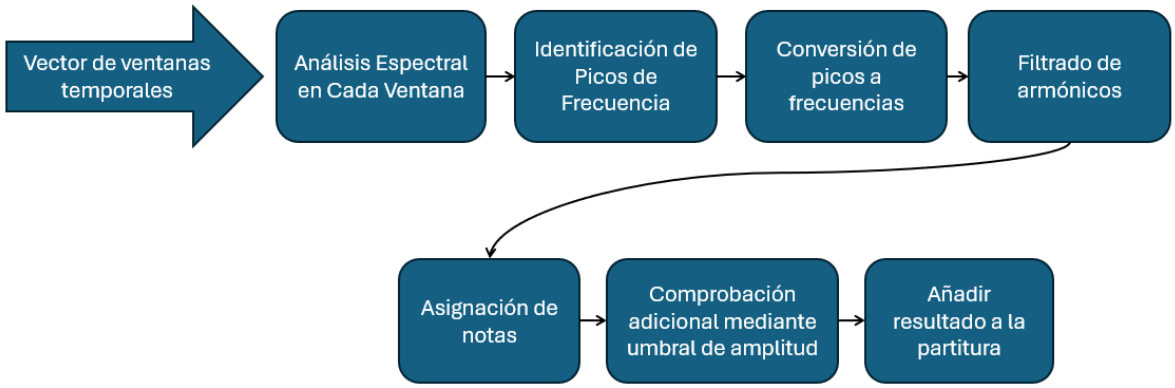


Ilustración 22

1. **Vector de ventanas temporales (input):** Representado como una lista de listas, donde cada sublista contiene las muestras de una ventana concreta del audio. El audio completo es segmentado previamente en ventanas de duración fija (por ejemplo, 200 ms), aplicando un porcentaje de solape configurable (por ejemplo, 90%) para asegurar continuidad entre ventanas consecutivas y capturar correctamente los cambios de nota. Como resultado, la señal de entrada se estructura como una lista de elementos, donde cada elemento es a su vez una lista de muestras temporales que corresponden a un fragmento concreto de la señal original. Esta estructura permite analizar individualmente cada ventana en las etapas posteriores de análisis espectral y detección de frecuencias.
2. **Análisis Espectral en Cada Ventana:** A cada ventana segmentada se le aplica la Transformada Rápida de Fourier (FFT). Este procedimiento convierte el contenido de la señal desde el dominio temporal al dominio frecuencial, obteniendo el espectro de frecuencias y sus correspondientes amplitudes. Se eliminan las frecuencias negativas

(derivadas de la simetría de la FFT) y se trabaja únicamente con las frecuencias positivas, que representan el contenido físico de la señal.

3. **Identificación de Picos de Frecuencia:** Sobre el espectro obtenido, se aplica el algoritmo find_peaks de SciPy. Este algoritmo identifica los máximos locales de amplitud en el espectro de cada ventana, que representan las frecuencias dominantes potencialmente asociadas a notas musicales o a sus armónicos. Para garantizar la calidad de las detecciones, se introducen restricciones como la distancia mínima entre picos, evitando la identificación de picos espurios demasiado cercanos.
4. **Conversión de Picos a Frecuencias**: Los índices de los picos detectados en la FFT son convertidos a frecuencias expresadas en Hz. Esta conversión tiene en cuenta tanto la frecuencia de muestreo como el tamaño de la ventana analizada, proporcionando las frecuencias reales que conforman el espectro de la ventana.
5. **Filtrado de Armónicos:** No todas las frecuencias detectadas corresponden a fundamentales; muchos de los picos representan armónicos (múltiplos enteros de las fundamentales). Se implementa un filtrado de armónicos donde, mediante el análisis de relaciones de múltiplos entre frecuencias, se descartan aquellos picos que sean múltiplos enteros (dentro de un margen de tolerancia) de frecuencias inferiores. De este modo, se preservan únicamente las frecuencias fundamentales.
6. **Asignación de Notas:** Cada frecuencia fundamental restante es comparada con un conjunto de tramos de tolerancia predefinidos para cada nota musical. Si la frecuencia se encuentra dentro de los límites de un tramo concreto, se le asigna directamente la nota correspondiente.
7. **Comprobación adicional mediante umbral de amplitud**: Antes de confirmar la asignación de nota, se comprueba que la amplitud máxima de la ventana supere un umbral mínimo de energía (ampliMin). Si la amplitud no alcanza este umbral, se considera que no existe suficiente información para determinar una nota, asignándose el código "ZZZ".
8. **Añadir resultado a la partitura:** Finalmente, la nota asignada (o "ZZZ" si corresponde) se añade a la secuencia global de la partitura. Este proceso se repite para todas las ventanas de análisis, construyendo así la partitura digital que representa la evolución temporal de las notas musicales interpretadas en el fragmento de audio.

La Ilustración 23 muestra el espectro de frecuencias correspondiente a la ventana número 135 del audio analizado. Este será el caso práctico que desarrollaremos en detalle a lo largo de cada una de las etapas del sistema descritas en el bloque anterior.

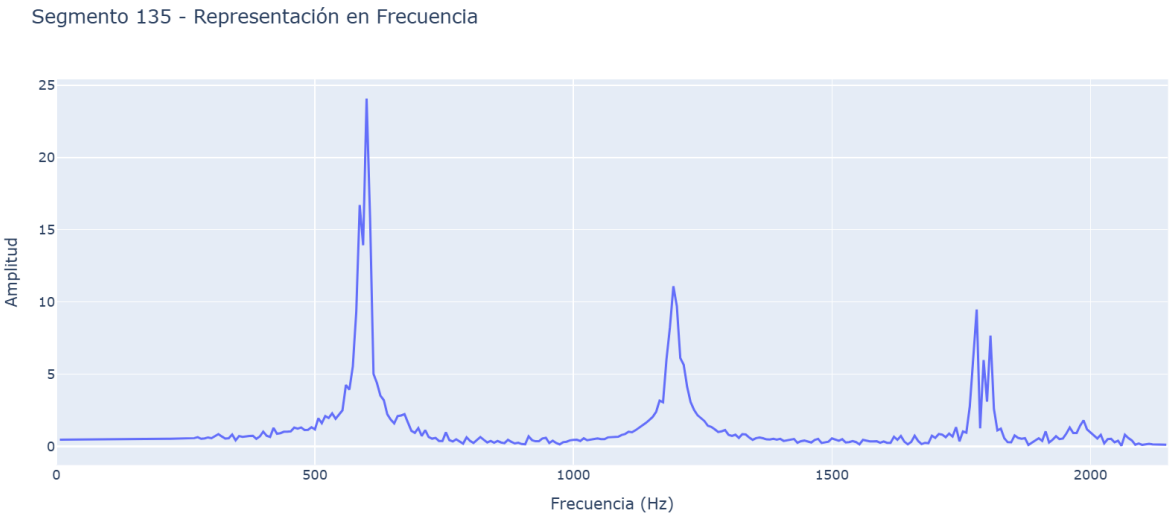


Ilustración 23

1. **Entrada: vector de ventanas temporales (input):** Previamente, la señal de audio fue segmentada en múltiples ventanas de 200 ms de duración, con un solape del 90%.
2. **Análisis espectral en la ventana:** Se aplica la Transformada Rápida de Fourier (FFT) sobre la ventana. Como resultado, se obtiene el espectro de frecuencias representado en la gráfica.
3. **Identificación de picos de frecuencia:** Sobre el espectro obtenido, se aplica el algoritmo find_peaks de SciPy para detectar los máximos locales. El algoritmo identifica como picos principales:
   - **586 Hz**
   - **593 Hz**
   - **600 Hz (máximo principal)**
   - **1193 Hz**
   - **1780 Hz**
   - **1793 Hz**
   - **1806 Hz**

   Se aplica una distancia mínima entre picos (distance=50) para evitar detecciones excesivamente cercanas.

4. **Conversión de picos a frecuencias:** En esta fase, los índices obtenidos de la FFT se convierten a frecuencias absolutas en Hz mediante la Ecuación 13 y se filtran aquellos picos que se encuentren a menos de 50Hz unos de otros por lo que de las frecuencias mostradas anteriormente nos quedaremos solo con:
   - **600 Hz (máximo principal)**
   - **1193 Hz**
   - **1780 Hz**
5. **Filtrado de armónicos:** Se analiza ahora si los picos detectados son armónicos o fundamentales. El algoritmo compara cada frecuencia con las anteriores según la Ecuación 14. Obteniendo los siguientes resultados en la Tabla 3:

fj\fi	600	1193	1780
600	1,00	-	-
1193	1,99	1,00	-
1780	2,97	1,49	1,00

Tabla 3

Por ello, podemos decir que 1193 Hz y 1780 Hz son armónicos fundamentales de la nota de 600Hz por lo que podemos decir que la nota predominante en esta ventana es la de 600Hz

6. **Asignación de notas:** La frecuencia fundamental restante (600 Hz) es comparada con los tramos de tolerancia de cada nota. Según la tabla de frecuencias estándar generada previamente, la frecuencia de 587.33 Hz corresponde a la nota DN5 (Re5). Como 600 Hz cae dentro del tramo de tolerancia definido alrededor de DN5 (25% del intervalo entre notas adyacentes), se asigna la nota **DN5**.
7. **Comprobación adicional mediante umbral de amplitud:** Se verifica que la amplitud máxima de la ventana supera el umbral de detección ampliMin (por ejemplo, 5). Como la amplitud máxima es superior a 20 se confirma la detección y no se asigna "ZZZ".
8. **Inclusión en la partitura:** La nota **DN5** es añadida a la secuencia global partitura como resultado de la ventana 135.

7.2 Instrumentos Polifónicos

Vectorización de amplitudes por rango de notas

El análisis de instrumentos polifónicos presenta una complejidad adicional respecto a los melódicos, ya que estos son capaces de producir múltiples notas de manera simultánea. En consecuencia, el sistema requiere una estrategia distinta para representar la información contenida en cada segmento de audio antes de realizar la predicción.

Para ello, se implementó un sistema de vectorización de amplitudes basado en los rangos de notas musicales. Este procedimiento permite transformar la información espectral contenida en cada segmento en un vector numérico de longitud fija que posteriormente puede ser procesado por modelos de machine learning.

En primer lugar, se generan los rangos de frecuencias asociados a cada nota musical. Para ello, se parte de un diccionario completo de notas y frecuencias generado programáticamente a partir de la Ecuación 15:

$$f = 16.3516 * 2^{(octava + \frac{i}{12})}$$

Ecuación 15

donde octava varía de 0 a 8 y i corresponde a las 12 notas de la escala cromática.

Posteriormente, se calcula un intervalo de tolerancia alrededor de cada frecuencia nominal de nota. Este intervalo tiene en cuenta un porcentaje configurable (por ejemplo, un 25%) del espacio entre la frecuencia de la nota y sus adyacentes, proporcionando así un rango dentro del cual se acepta la energía espectral como perteneciente a dicha nota. Esta interpolación permite mitigar pequeñas variaciones de afinación, vibratos o desviaciones armónicas propias de los instrumentos.

Una vez calculados los tramos, el sistema analiza cada segmento de audio previamente filtrado y transformado al dominio frecuencial (mediante FFT). Para cada segmento, se determina la energía total contenida en cada uno de los tramos definidos, obteniendo así un vector de características donde cada posición representa la suma de amplitudes de una nota concreta en el rango completo.

Formalmente, si A(f) es la amplitud de la frecuencia f, y $[f_{min}, f_{max}]$ es el tramo de la nota n, el componente Vn del vector de características se calcula en la Ecuación 16:

$$V_n = \sum_{f \in [f_{min}, f_{max}]} A(f)$$

Ecuación 16

El resultado es un vector de 108 elementos (uno por cada posible nota) que representa la actividad espectral del segmento. Este vector es posteriormente normalizado o directamente introducido en el modelo de clasificación.

Este vector de características es la entrada para los modelos supervisados empleados en el sistema, tanto para los modelos KNN, MLP o RNN. La ventaja de este enfoque es que permite resumir la información compleja de un segmento polifónico en un único vector, independiente de la duración temporal del segmento, y en el que se conserva la información musical relevante.

Este sistema de vectorización por tramos ha demostrado ser robusto incluso en presencia de armónicos, ruidos de fondo o pequeñas desafinaciones, permitiendo así abordar el reto de la transcripción de instrumentos polifónicos con un alto grado de precisión.

La secuencia realizada se muestra en la Ilustración 24:

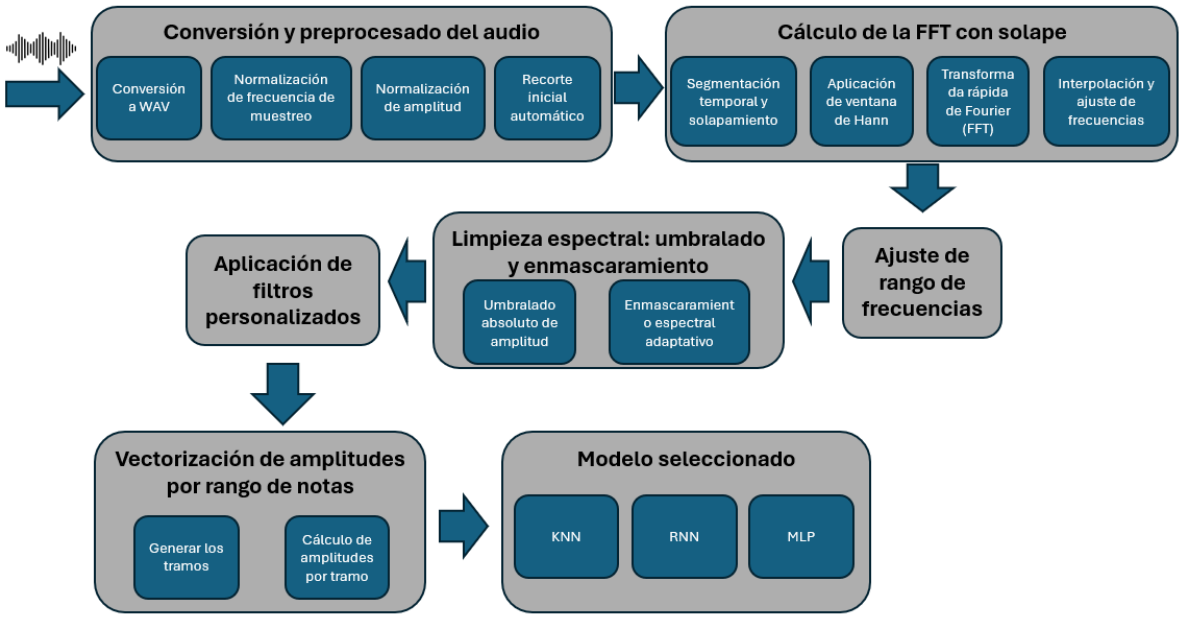


Ilustración 24

Creación del dataset de entrenamiento

El sistema de transcripción automática de instrumentos polifónicos se entrenó utilizando un conjunto de datos específicamente diseñado y adaptado al problema concreto. El dataset final tiene forma matricial, donde cada muestra está compuesta por:

- Un vector de 108 características, correspondiente a las 108 notas posibles consideradas en el sistema. Cada característica representa la suma de amplitudes dentro del tramo de frecuencias definido para cada nota, calculado a partir del espectro de cada audio grabado.
- Una columna adicional (columna 109), que contiene la etiqueta de la clase, es decir, la nota exacta que se grabó en ese fichero de audio.

De este modo, cada fila del dataset constituye un ejemplo de entrenamiento donde se asocia un patrón de amplitudes espectrales con la nota que lo ha generado.

Debido a la ausencia de datasets públicos suficientemente precisos y controlados para el problema tratado, se optó por realizar una grabación manual de las muestras. Este proceso consistió en:

1. Grabación individualizada de cada nota del instrumento (en este caso, por ejemplo, de guitarra), de forma que cada fichero de audio contenía únicamente una nota aislada.
2. Las grabaciones se realizaron en condiciones controladas, asegurando que el sonido capturado correspondiera de forma clara a una única nota sin presencia de acordes, ruidos de fondo significativos o armónicos de otros instrumentos.
3. Cada audio fue etiquetado manualmente con la nota correspondiente, asegurando así la fiabilidad de las clases objetivo.
4. Los audios fueron posteriormente procesados para:
    - Segmentar el fichero en fragmentos de duración fija.
    - Aplicar la Transformada de Fourier (FFT) a cada segmento.
    - Calcular la suma de amplitudes en los tramos de frecuencia asignados a cada nota.

Una vez generado el conjunto completo de muestras vectorizadas, se procedió a su división en subconjuntos de entrenamiento y prueba. Para ello:

Se realizó una partición estratificada del dataset, garantizando que todas las clases (notas) estuvieran representadas tanto en el conjunto de entrenamiento como en el de validación.

El reparto utilizado fue, generalmente, del 80% para entrenamiento y el 20% para prueba, de forma que se dispone de suficiente información para el aprendizaje y, al mismo tiempo, un conjunto independiente para evaluar el rendimiento final del modelo.

Esta división es especialmente importante en este tipo de problemas multiclase para evitar el desbalance de algunas notas con menos representaciones.

Gracias a esta estrategia de generación manual, segmentación, vectorización y partición, se obtiene un dataset limpio, robusto y perfectamente etiquetado para el entrenamiento de modelos de clasificación de notas polifónicas tal y como se muestra en la Ilustración 25.

	CN0	C#0	DN0	D#0	EN0	FN0	F#0	GN0	G#0	AN0	...	GN8	G#8	AN8	A#8	BN8	NOTA
0	0.000110	0.000000	0.000111	0.000000	0.000116	0.000119	0.000113	0.000121	0.000117	0.000113	...	0.012763	0.006519	0.002176	0.001314	0.002239	EN4
1	0.000019	0.000023	0.000000	0.000005	0.000000	0.000007	0.000016	0.000014	0.000038	0.000010	...	0.001451	0.001361	0.000626	0.000562	0.000690	EN4
2	0.000003	0.000000	0.000006	0.000000	0.000020	0.000010	0.000015	0.000016	0.000008	0.000012	...	0.003874	0.005424	0.002113	0.000554	0.001188	EN4
3	0.000022	0.000025	0.000014	0.000029	0.000029	0.000022	0.000019	0.000013	0.000029	0.000012	...	0.003393	0.003429	0.003210	0.002997	0.002771	EN4
4	0.000000	0.000000	0.000021	0.000017	0.000022	0.000016	0.000022	0.000017	0.000018	0.000017	...	0.001731	0.001045	0.006462	0.000551	0.000686	DN5

Ilustración 25

Modelos disponibles para instrumentos polifónicos

KNN

El análisis de instrumentos polifónicos presenta un reto computacional importante debido a la posibilidad de aparición simultánea de múltiples notas y la presencia de armónicos. En este contexto, el modelo de k-Nearest Neighbors (KNN) ha demostrado ser especialmente adecuado para la tarea de clasificación de notas a partir de los vectores de amplitudes generados en la fase de preprocesado.

El modelo KNN es un clasificador basado en distancia que no realiza un entrenamiento explícito de parámetros, sino que compara directamente las nuevas muestras con el conjunto de datos previamente etiquetado. Esta propiedad resulta muy ventajosa en el caso concreto del proyecto, por varias razones:

Naturaleza directa de los vectores de amplitudes: Los vectores generados en este proyecto tienen un significado físico claro (suma de amplitudes en cada tramo de nota). Las muestras correspondientes a una misma nota tienden a tener concentradas sus amplitudes en posiciones específicas del vector, y muy bajas en el resto. Esto genera una fuerte separación espacial en el espacio de características, lo que favorece enormemente a los algoritmos basados en distancia.

Poca necesidad de hiperparametrización compleja: A diferencia de otros modelos como redes neuronales, donde el entrenamiento y la arquitectura tienen un impacto crítico en el rendimiento, el modelo KNN únicamente depende del número de vecinos considerados (k) y de la métrica de distancia empleada (en este caso, Euclídea).

Robustez ante el tamaño limitado del dataset: En tareas de clasificación musical, disponer de grandes volúmenes de datos etiquetados es complejo. El KNN es capaz de funcionar de manera razonablemente estable incluso con conjuntos de datos de tamaño moderado, como es el caso de este trabajo.

Para seleccionar el mejor valor de k, se realizó un proceso de validación empírica analizando el comportamiento del modelo para valores de k entre 1 y 30. En la Ilustración 26, se observa la evolución de la precisión tanto en el conjunto de entrenamiento como en el de prueba:

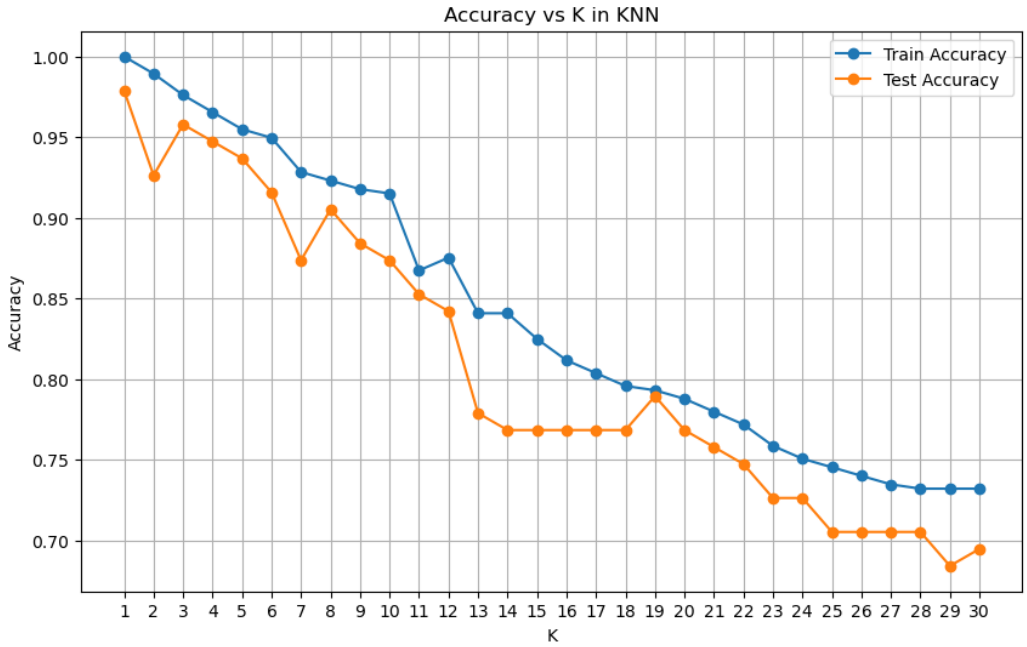


Ilustración 26

Como puede apreciarse, los valores bajos de k provocan un sobreajuste evidente, con precisiones cercanas al 100% en entrenamiento, pero caídas notables en validación. A medida que aumenta k, se mejora la generalización, pero a costa de perder capacidad discriminativa si el valor es excesivamente alto.

El mejor compromiso entre estabilidad y precisión se obtuvo para k = 7, donde se alcanzaron los siguientes resultados:

- Precisión en entrenamiento: 92.5%
- Precisión en validación (test): 87.5%

Estos valores muestran un rendimiento sólido del modelo, con una pequeña diferencia entre entrenamiento y prueba que indica buena capacidad de generalización sin haber incurrido en sobreajuste excesivo.

Para entender mejor los aciertos y errores del modelo, se analizaron las matrices de confusión correspondientes a las predicciones sobre los conjuntos de entrenamiento y prueba, mostradas en la Ilustración 27:

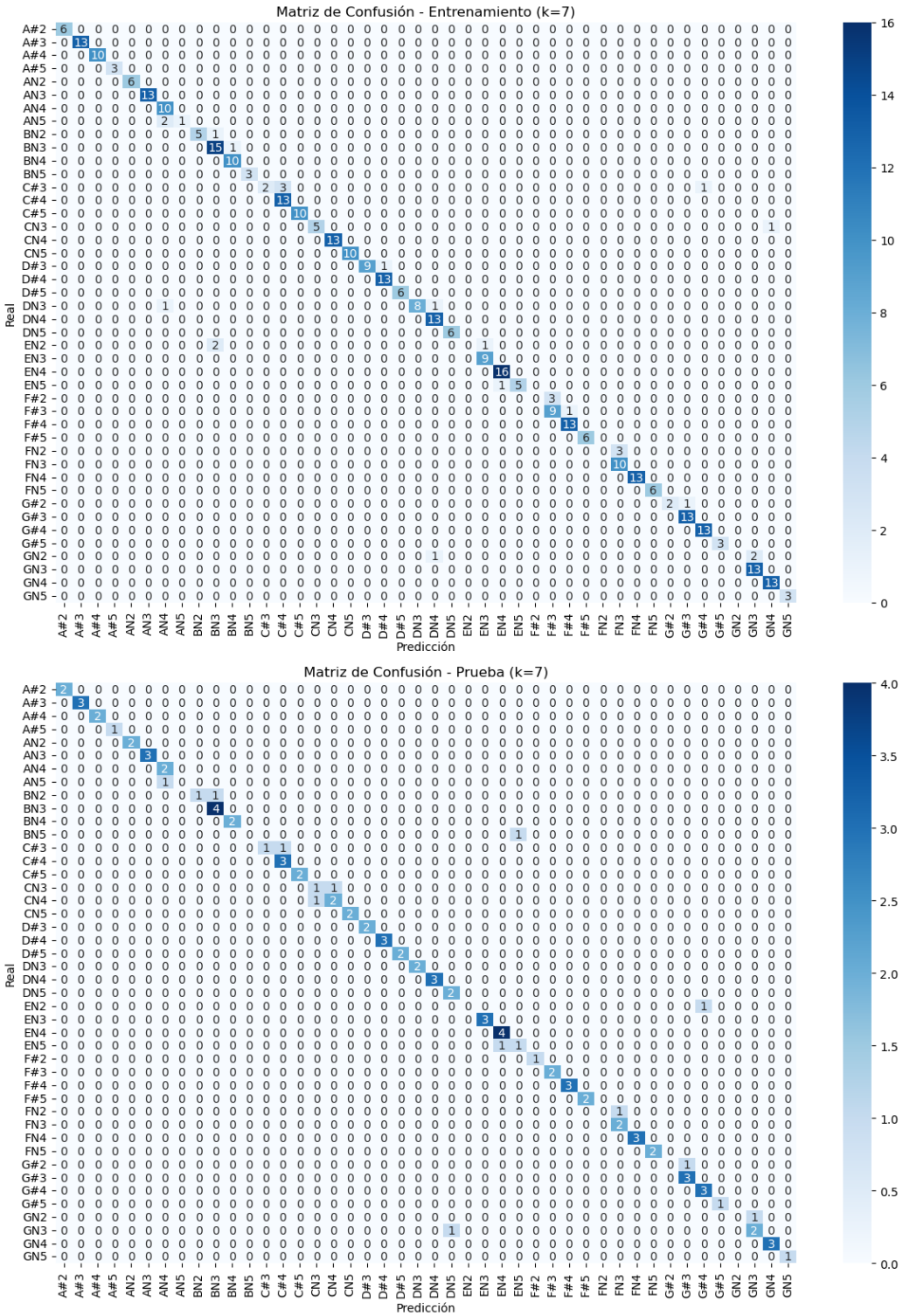


Ilustración 27

En ambas matrices puede observarse cómo los errores tienden a concentrarse en notas vecinas dentro de la escala musical, es decir, los fallos se producen habitualmente en notas adyacentes en frecuencia, lo que es coherente con la propia estructura del espectro de frecuencias musicales y la presencia de armónicos.

En el conjunto de entrenamiento, la mayoría de las notas muestran una clasificación perfecta o casi perfecta, con algunos casos aislados de confusiones entre semitonos cercanos. Esto es consistente con el 92.5% de precisión alcanzada.

En el conjunto de prueba, se mantiene una estructura similar, aunque como es esperable, aparecen algunas confusiones adicionales, especialmente en notas centrales de ciertas octavas. A pesar de ello, el modelo conserva un alto grado de acierto global (87.5%), confirmando que el KNN es una alternativa robusta para este tipo de problema.

Durante el proceso de ajuste de hiperparámetros, además del valor finalmente seleccionado de k = 7, se analizó también el comportamiento del modelo para otros valores bajos de k, particularmente k = 3, donde el modelo mostró un comportamiento especialmente interesante.

Tal y como se observa en la curva de validación en la Ilustración 26, en el caso de k = 3 se consiguió alcanzar un rendimiento superior en el conjunto de prueba, logrando los siguientes resultados:

- Precisión en entrenamiento: 92.5%
- Precisión en validación (test): superior al 95%

Tal y como se muestra en las matrices de confusión de la Ilustración 28:

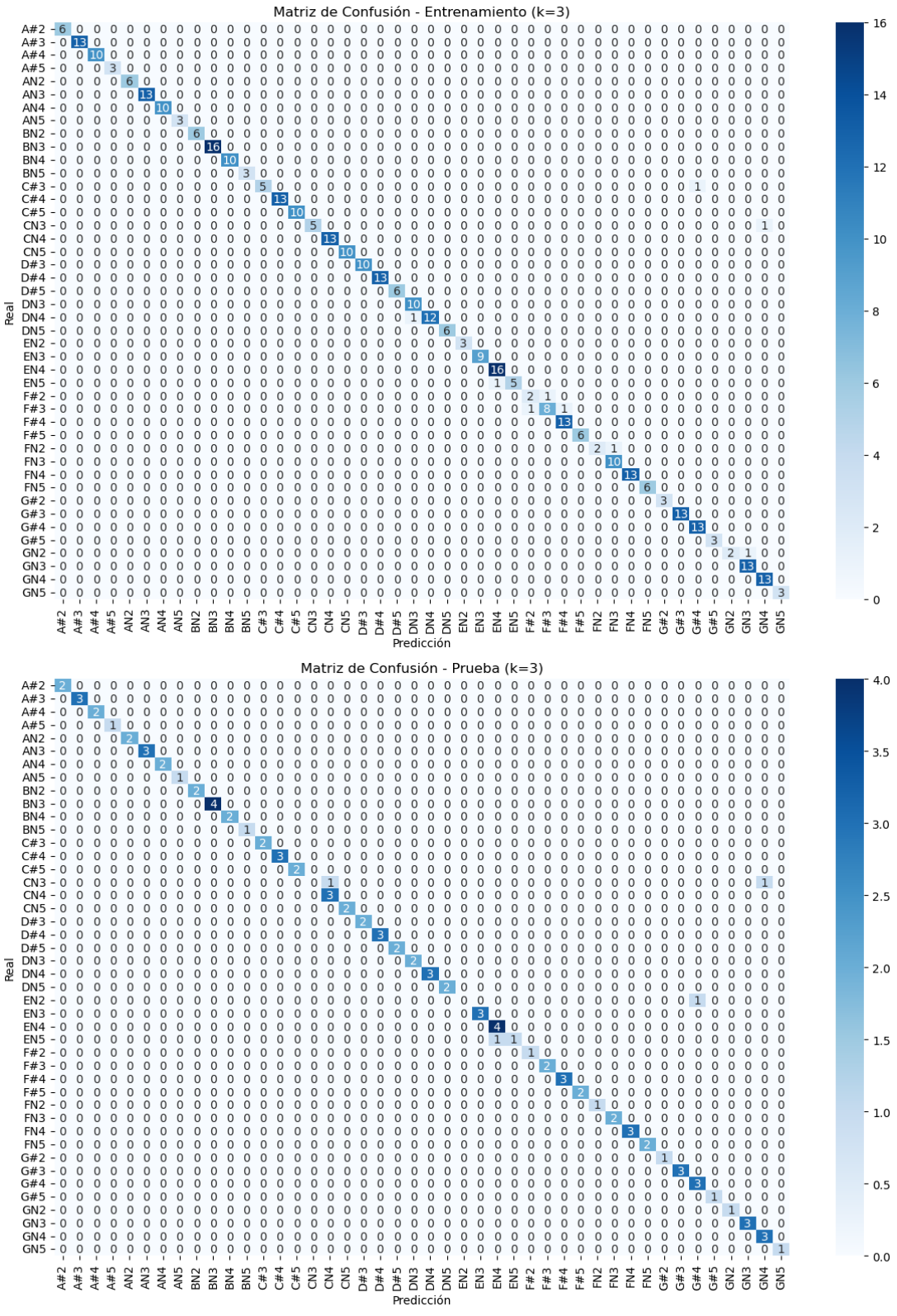


Ilustración 28

Este incremento en la precisión de test respecto al caso de k = 7 indica que, para este conjunto concreto de datos, el modelo es capaz de beneficiarse de un k relativamente bajo, ya que:

- Las muestras correspondientes a cada clase están bien separadas en el espacio de características.
- El diseño del vector de entrada (suma de amplitudes en tramos bien definidos) favorece una clara diferenciación entre notas.
- Las grabaciones controladas y el buen etiquetado reducen la probabilidad de que muestras cercanas sean de clases incorrectas, lo que permite que un valor pequeño de k no incurra en sobreajuste severo.

No obstante, cabe señalar que, aunque el rendimiento en test es superior para k = 3, este tipo de resultados debe interpretarse con cierta cautela:

- El mayor riesgo con k pequeños es la sensibilidad a ruidos o variabilidad en los datos.
- Al trabajar con datasets más grandes o con mayor heterogeneidad, es posible que un k algo mayor (como 7) ofrezca mejor estabilidad global.

Por este motivo, se decidió utilizar finalmente k = 7 como configuración base, priorizando un equilibrio entre precisión y robustez a posibles futuras ampliaciones del conjunto de datos. Sin embargo, el excelente comportamiento observado en k = 3 constituye una validación adicional de la idoneidad del modelo KNN para el problema abordado.

MLP

Además de los modelos KNN y RNN previamente descritos, se evaluó también el rendimiento de un modelo de Multi-Layer Perceptron (MLP) para la clasificación de notas polifónicas. El MLP representa uno de los enfoques clásicos dentro del aprendizaje profundo supervisado, caracterizándose por su capacidad de modelar relaciones no lineales complejas entre las características de entrada y las clases objetivo.

En el contexto de este trabajo, el MLP constituye un punto intermedio entre los modelos puramente estadísticos (como el KNN) y los modelos temporales (como la RNN), permitiendo explotar la estructura interna de los vectores de amplitudes mediante el ajuste de pesos sin necesidad de explotar explícitamente la dimensión secuencial.

El proceso de preparación de los datos fue idéntico al empleado en el modelo RNN:

- Codificación de etiquetas: Las clases fueron transformadas mediante LabelEncoder().
- Normalización: Los 108 vectores de entrada fueron escalados usando StandardScaler().
- Partición de entrenamiento y prueba: División en 80% entrenamiento y 20% prueba, con estratificación por clases.

- Balanceo mínimo: Asegurando al menos 3 ejemplos por clase mediante sobresampling en caso necesario.

La única diferencia con respecto a la RNN es que los datos no precisaron ser remodelados a estructura tridimensional, dado que el MLP trabaja directamente sobre vectores 2D estándar.

La red neuronal MLP empleada para este experimento fue configurada con:

- Una capa oculta principal de varias neuronas (ajustadas empíricamente).
- Función de activación ReLU en capas intermedias.
- Función de activación softmax en la capa de salida multiclase.
- Optimizador Adam y función de pérdida categorical_crossentropy.

El entrenamiento se ejecutó durante un número fijo de iteraciones, tal y como se puede visualizar en la Ilustración 29, que muestra la evolución de la función de coste a lo largo del proceso de entrenamiento:

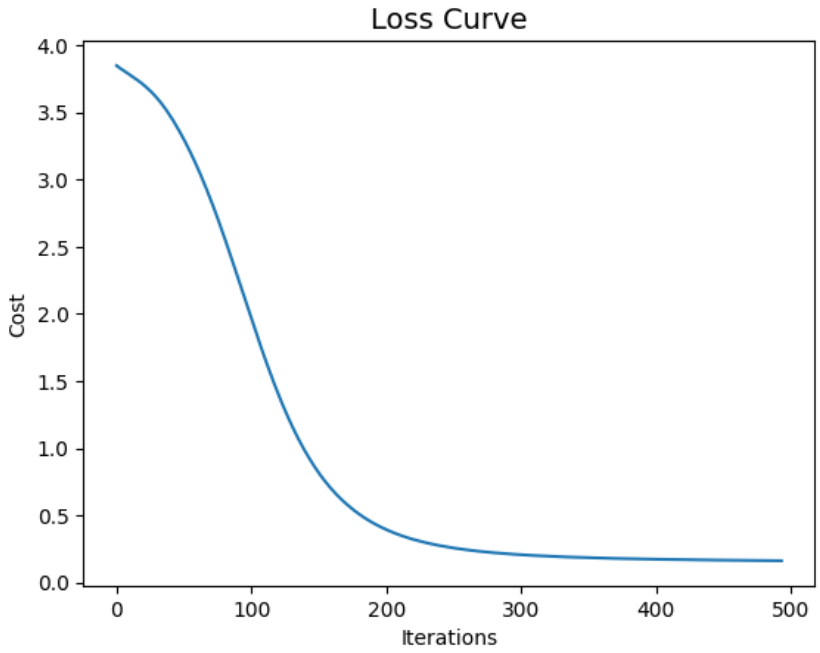


Ilustración 29

Puede observarse cómo el coste decrece progresivamente, alcanzando rápidamente valores cercanos a 0, lo que refleja la buena capacidad del MLP para ajustar el conjunto de entrenamiento.

Una vez completado el entrenamiento, el MLP alcanzó los siguientes resultados globales:

- Precisión global en test: 98%
- Macro-average accuracy: 98%
- Weighted-average accuracy: 98%

Este excelente rendimiento refleja la capacidad del MLP para capturar patrones no lineales complejos en el espacio de amplitudes vectorizadas.

Al igual que en los modelos anteriores, se construyeron las matrices de confusión tanto para el conjunto de entrenamiento como para el de prueba, recogidas en la Ilustración 30 (entrenamiento) y la Ilustración 31 (test):

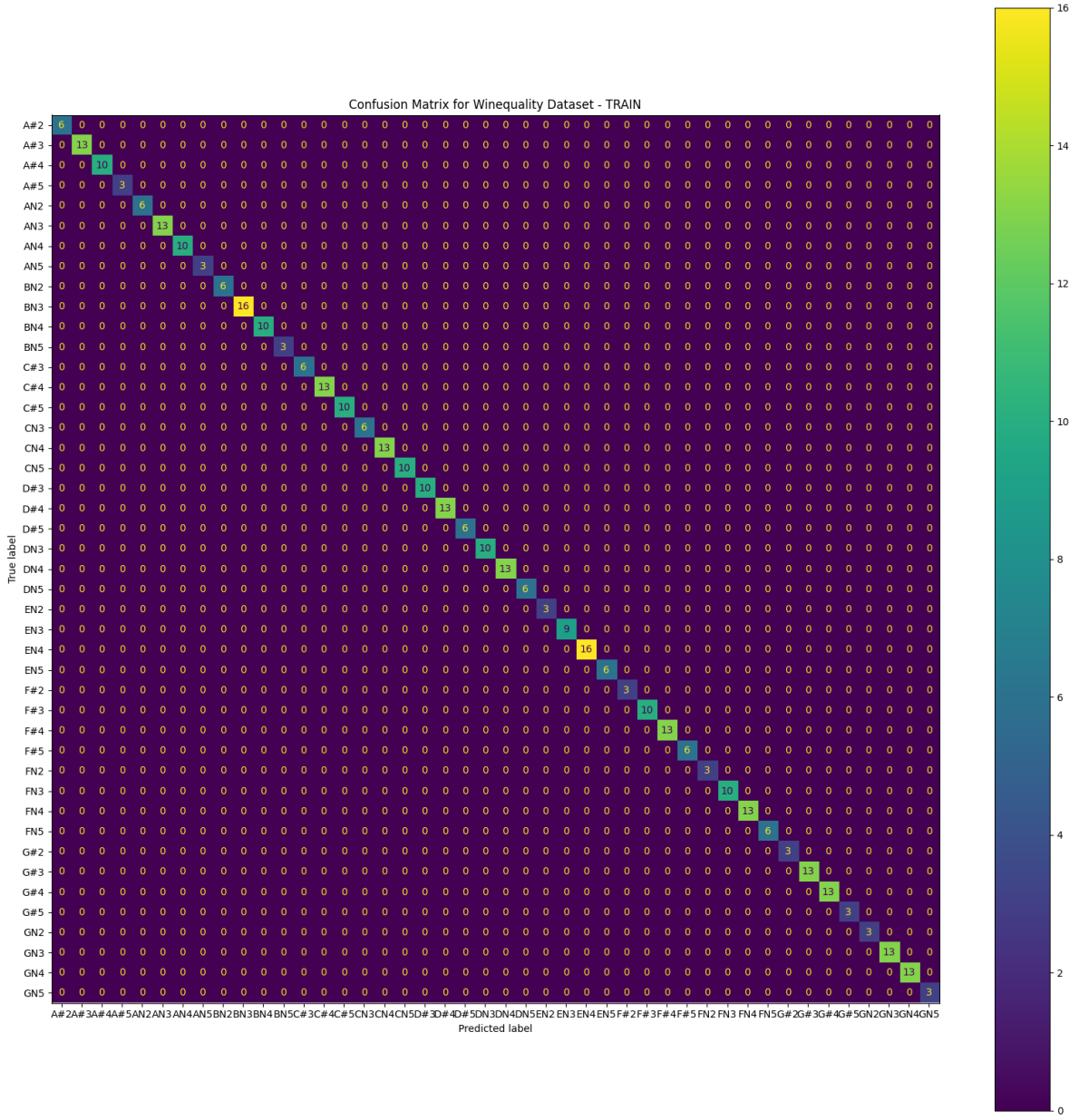


Ilustración 30

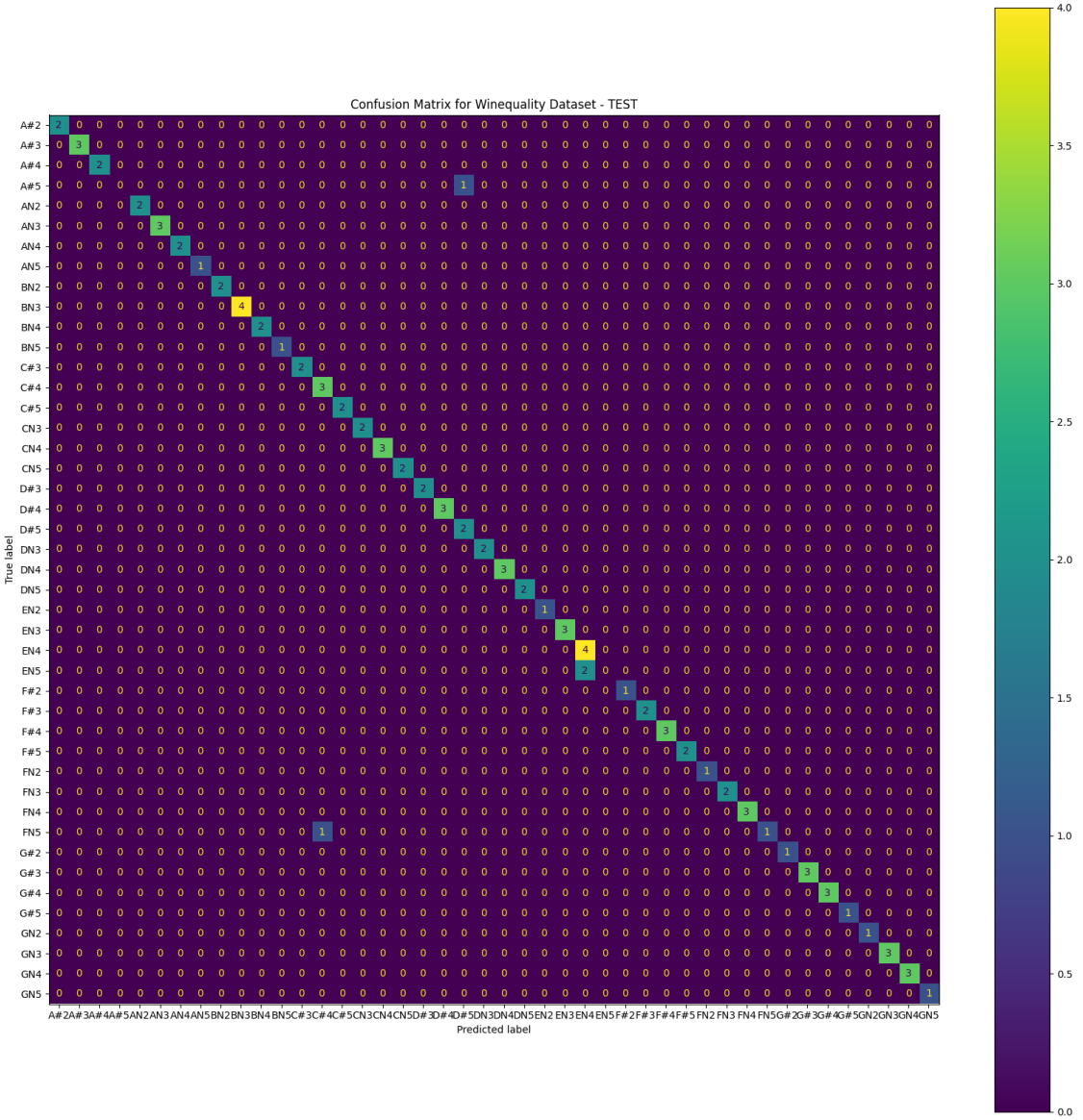


Ilustración 31

En la matriz de entrenamiento (Ilustración 30), prácticamente todas las predicciones se concentran en la diagonal, lo que confirma que el MLP ha aprendido perfectamente los patrones del conjunto de entrenamiento.

En la matriz de prueba (Ilustración 31), se observa también un comportamiento muy sólido, con escasos errores dispersos, mayoritariamente en notas adyacentes o dentro del rango de la misma octava, manteniendo así la robustez general del modelo.

El rendimiento alcanzado por el MLP es especialmente destacable en este problema:

- Es capaz de alcanzar resultados incluso superiores a los observados en los modelos KNN y RNN.
- Su capacidad de modelar relaciones no lineales complejas es muy adecuada para el tipo de características extraídas (suma de amplitudes por tramo).
- Presenta además una ventaja práctica: su entrenamiento es rápido y estable una vez normalizados los datos.

- A diferencia de los KNN, no depende de memorizar los datos de entrenamiento ni de calcular distancias en tiempo de inferencia.

En resumen, el MLP constituye una de las alternativas más sólidas dentro del sistema global desarrollado, alcanzando muy altas tasas de clasificación con un comportamiento consistente tanto en entrenamiento como en validación.

RNN

Como alternativa al modelo basado en KNN y MLP, se implementó también un modelo de aprendizaje profundo utilizando una Red Neuronal Recurrente (RNN). Este tipo de arquitectura resulta especialmente interesante en tareas musicales por su capacidad natural de manejar información secuencial y capturar patrones temporales o relaciones internas entre las características de entrada.

El proceso de preparación de los datos para entrenar la RNN siguió varias etapas:

- **Codificación de etiquetas:** Las etiquetas correspondientes a cada nota fueron transformadas en valores numéricos mediante LabelEncoder(), lo que permite trabajar directamente en formato multiclase.
- **Estandarización de características:** Para estabilizar el entrenamiento de la red, las 108 características fueron escaladas usando StandardScaler(), evitando así problemas de escala dispar entre amplitudes de distintas notas.
- **Partición de entrenamiento y prueba:** El dataset fue dividido en un 80% para entrenamiento y un 20% para validación, garantizando la representatividad de todas las clases mediante estratificación.
- **Sobremuestreo controlado:** Se verificó que cada clase tuviera al menos 3 ejemplos en el conjunto de entrenamiento, aplicando sobremuestreo (resampling) cuando alguna clase minoritaria presentaba un número inferior de muestras. Esto asegura una representación mínima uniforme de cada nota, aspecto crítico en aprendizaje supervisado multiclase.
- **Reshape para RNN:** Finalmente, los datos fueron remodelados al formato requerido por las RNN de Keras: una estructura tridimensional [muestras, pasos de tiempo, características], en este caso con un único paso de tiempo por muestra.

Arquitectura del modelo:

- La red neuronal diseñada consta de:
- Una capa SimpleRNN de 64 unidades con activación ReLU.
- Una capa densa intermedia de 32 unidades ReLU.
- Una capa de salida Dense con activación softmax, adaptada al número total de clases musicales presentes.
- La compilación se realizó con el optimizador Adam

En la Ilustración 32 se presenta la evolución de la precisión a lo largo de las 50 épocas para los conjuntos de entrenamiento y validación. Se observa cómo el modelo logra un aprendizaje progresivo y estable, alcanzando precisiones muy elevadas ya en torno a la época 15.

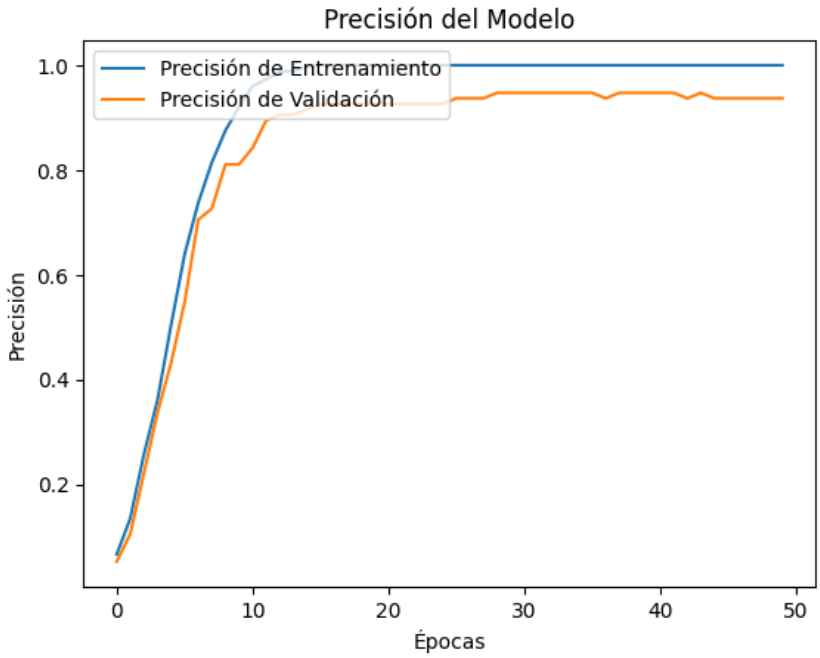


Ilustración 32

En paralelo, la evolución de la función de pérdida durante el entrenamiento se muestra en la Ilustración 33. Puede comprobarse cómo tanto en entrenamiento como en validación, la pérdida converge de manera progresiva, sin presencia significativa de overfitting.

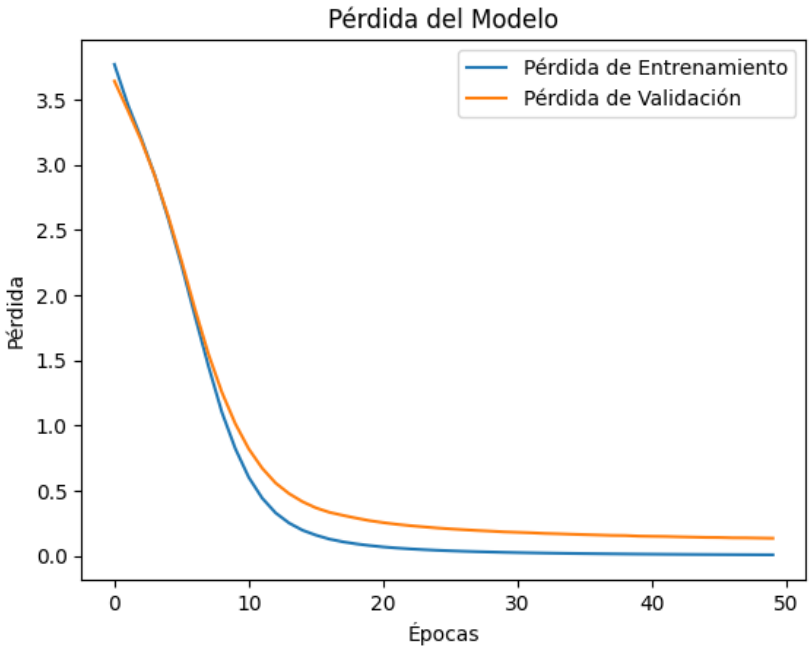


Ilustración 33

Al finalizar el entrenamiento, los resultados obtenidos fueron los siguientes:

- Precisión en el conjunto de entrenamiento: 100%
- Precisión en el conjunto de validación (prueba): 95%

Estos valores son indicativos de un aprendizaje muy sólido del modelo. El hecho de alcanzar el 100% en entrenamiento confirma la capacidad de la RNN para adaptarse al patrón de datos tras el preprocesamiento, mientras que mantener un 95% en test demuestra la buena generalización alcanzada.

Para complementar el análisis de rendimiento, en la Ilustración 34 se presentan las matrices de confusión obtenidas tanto en entrenamiento como en prueba:

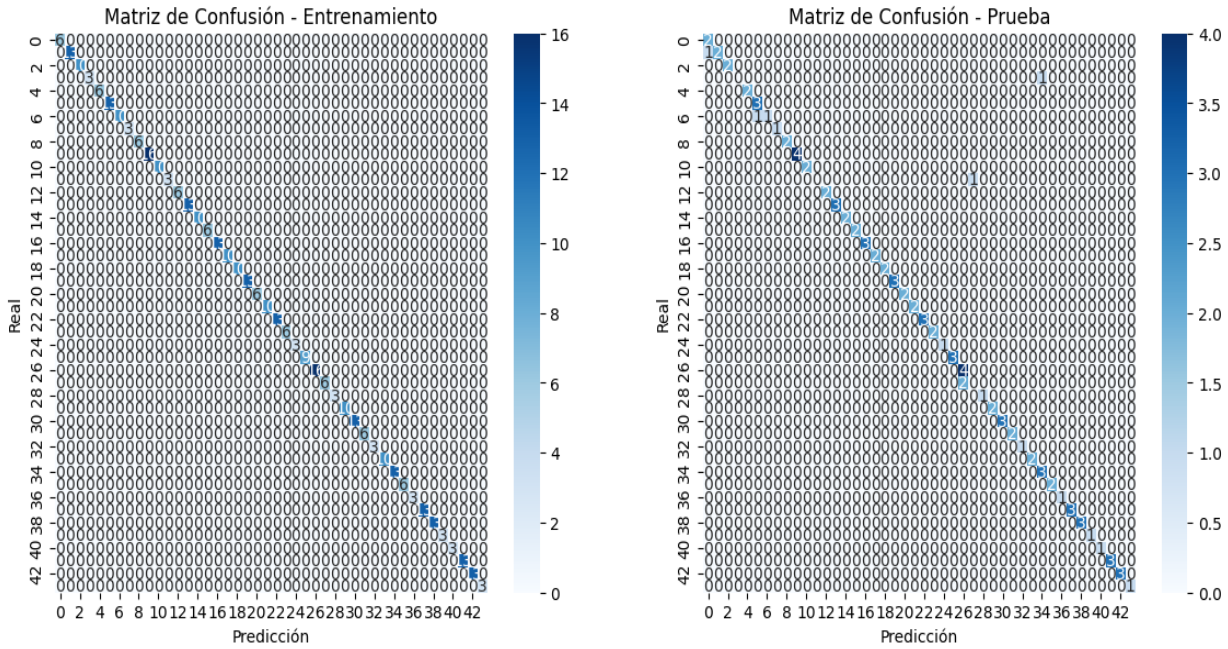


Ilustración 34

Como puede apreciarse, el modelo consigue una clasificación prácticamente perfecta en el conjunto de entrenamiento (concentración absoluta en la diagonal), mientras que en el conjunto de prueba se observan muy pocas confusiones residuales, siempre entre clases adyacentes, lo que resulta coherente con los patrones armónicos del instrumento. El bajo número de errores confirma la estabilidad del modelo en escenarios no vistos durante el entrenamiento.

El modelo RNN ha demostrado ser muy eficaz en este contexto:

- La estructura interna del vector de amplitudes (distribución entre notas) se adapta muy bien al aprendizaje secuencial implícito de las RNN.
- La normalización previa ha permitido estabilizar el entrenamiento, evitando saturaciones en las activaciones.
- El preprocesamiento robusto, con tramos de notas bien definidos, permite a la RNN discernir claramente los patrones espectrales asociados a cada nota.

Gracias a su capacidad de modelar relaciones internas entre características, la red es capaz de resolver incluso casos donde existe solape parcial entre notas próximas en frecuencia.

En conjunto, la RNN se presenta como un modelo altamente competitivo dentro del sistema global, alcanzando resultados muy próximos al límite teórico de precisión para este problema supervisado.

Visualización tipo Piano-Roll

Como fase final del sistema desarrollado, se implementó un módulo de visualización animada tipo **Piano-Roll**, con el objetivo de ofrecer una representación gráfica clara y dinámica de la partitura generada a partir de los audios procesados.

El piano-roll es un tipo de visualización muy utilizado en contextos de análisis musical y edición MIDI. Su principal ventaja es que permite mostrar simultáneamente:

- **Eje horizontal (X):** representa el tiempo, dividido en ventanas o segmentos sucesivos.
- **Eje vertical (Y):** representa las notas musicales, ordenadas de grave a agudo.
- **Barras horizontales:** indican cuándo una nota está activa, es decir, detectada en cada ventana de análisis. La duración de cada barra refleja cuánto tiempo permanece activa la nota en la partitura generada.

De este modo, la representación permite visualizar intuitivamente cuándo suenan las diferentes notas y su duración aproximada, simulando así un equivalente visual de la partitura obtenida tal y como se muestra en la Ilustración 35:

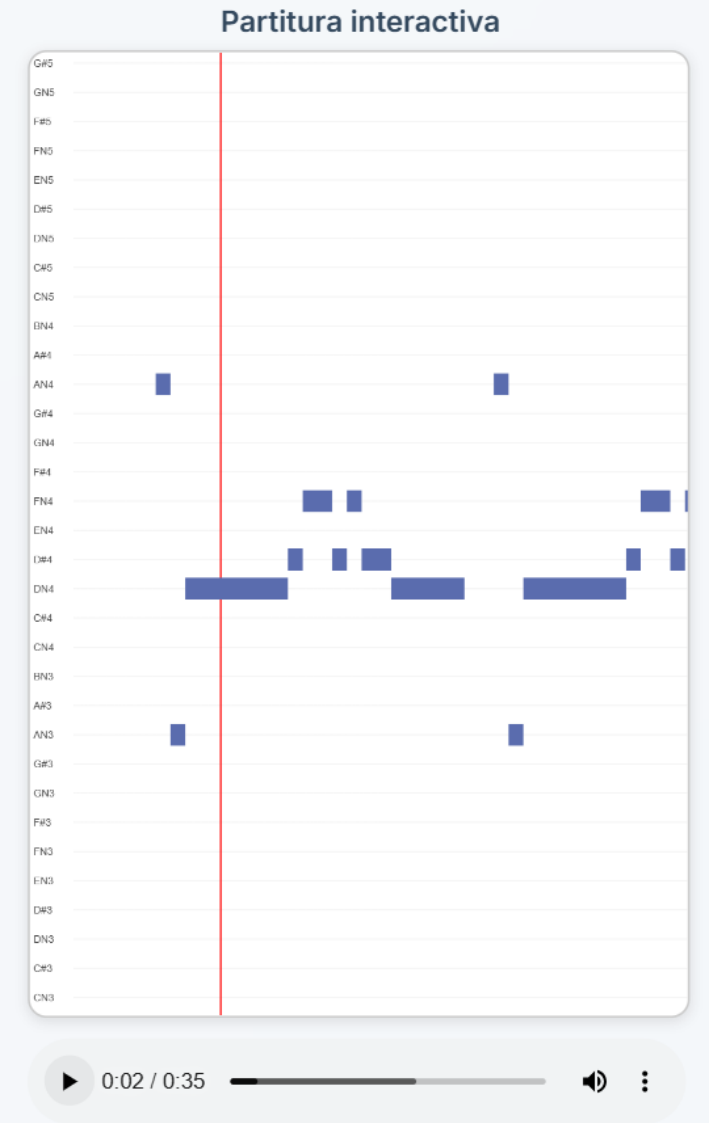


Ilustración 35

La visualización se genera directamente a partir de la **partitura calculada por el sistema**, que internamente es una lista de notas donde cada elemento corresponde a la predicción realizada para cada ventana de análisis.

El proceso de construcción es el siguiente:

1. **Entrada de datos:**
    - La partitura es una lista ordenada de notas musicales (["C4", "C4", "G4", "G4", "E4", ...]).
    - Se conoce el **rate de muestreo de ventanas** utilizado durante la segmentación (por ejemplo, rate=200 milisegundos).
    - Se conoce el **solape (overlap)** entre ventanas, que puede variar entre 0 y 1.
    - Se conoce el instrumento analizado (flauta, guitarra, etc.).

2. **Cálculo del tiempo real de cada ventana:** Cada posición de la partitura corresponde a un instante temporal calculado con la Ecuación 17:

$$tiempo_i = i * duración\ de\ ventana\ efectiva$$

Ecuación 17

donde la duración efectiva de cada ventana viene dada por la Ecuación 18:

$$duración\ ventana\ efectiva = intervalo * (1 - solape)$$

Ecuación 18

De este modo, incluso con solapes elevados (por ejemplo 90%), se ajusta correctamente la posición de cada nota en el eje de tiempos.

3. **Conversión de notas a frecuencias:** A partir de la lista de notas detectadas, se transforma cada nota a su frecuencia correspondiente mediante los diccionarios notasFrecuencias y frecuenciasNotas generados previamente en el sistemaconstantesComunes.

Esto permite convertir las notas categóricas a valores numéricos continuos, facilitando su representación gráfica en el eje vertical.

4. **Determinación del rango vertical de frecuencias:** El sistema detecta el instrumento procesado (por ejemplo flauta o guitarra) y obtiene automáticamente su rango de frecuencias.

Este rango determina los límites superior e inferior del eje Y del piano-roll, asegurando que la visualización sea siempre específica al instrumento tratado y optimice el espacio gráfico.

5. **Gestión de transiciones y ligados:** Para simular una representación musical más natural, el sistema aplica una lógica de transición suave:
    - Si en una ventana determinada no se detecta ninguna nota (ZZZ), pero en la anterior sí, se prolonga la última nota mantenida.
    - Este mecanismo simula el comportamiento de ligados, vibratos o prolongaciones de notas comunes en interpretaciones reales.
6. **Representación final:** Finalmente, el piano-roll se dibuja animadamente utilizando herramientas de visualización como matplotlib o plotly, generando:
    - Barras horizontales continuas mientras la nota permanece activa.
    - Nuevas barras cuando aparece un cambio de nota.
    - Un desplazamiento continuo en el eje de tiempos conforme avanza la animación.

De este modo, se consigue una representación fiel y coherente con el análisis de audio realizado, permitiendo al usuario visualizar el resultado completo de la transcripción automática de forma clara, didáctica y musicalmente interpretable.

Capítulo 8. Análisis del comportamiento de la aplicación MelodyMapper

Consideraciones por tipo de instrumento (melódico vs polifónico)

El análisis del comportamiento de la aplicación *MelodyMapper* ha evidenciado diferencias clave en el rendimiento del sistema según el tipo de instrumento analizado. Para instrumentos melódicos como la flauta travesera, el sistema ha mostrado una alta fiabilidad en la detección de notas individuales, gracias a la claridad espectral de estas señales. La aplicación consigue transformar con precisión grabaciones monofónicas en secuencias de notas, lo que ha facilitado la generación de partituras legibles y coherentes. Esta solidez se debe en gran parte al uso de algoritmos basados en transformadas de Fourier y al filtrado personalizado en el dominio de la frecuencia, que permite aislar eficazmente la nota activa en cada instante.

En contraste, el tratamiento de instrumentos polifónicos, como la guitarra, presenta desafíos adicionales que afectan directamente al comportamiento de *MelodyMapper*. La superposición de armónicos y la simultaneidad de varias notas dificultan la segmentación y clasificación de las señales. Pese a ello, la incorporación de modelos de aprendizaje automático, como KNN, MLP y RNN, ha permitido alcanzar resultados razonables al convertir la representación frecuencial en vectores de características. Aunque la precisión aún es menor que en el caso de los instrumentos melódicos, el sistema ha demostrado ser capaz de reconocer patrones relevantes y generar una partitura útil como base para su posterior edición o interpretación. Estos resultados remarcan la importancia de adaptar las técnicas de análisis al tipo de instrumento y sientan las bases para mejoras futuras orientadas a optimizar el rendimiento en contextos polifónicos complejos.

Resultados para instrumentos melódicos

El sistema de análisis por ventanas ha sido evaluado específicamente sobre instrumentos melódicos (clarinete, flauta travesera y saxofón), considerando fragmentos reales en los que cada nota aparece de forma consecutiva y aislada. Este enfoque permite estimar de forma robusta la precisión del sistema de transcripción automática en condiciones próximas al uso real.

El modelo segmenta la señal de audio en ventanas temporales y extrae la información espectral de cada tramo para inferir la nota más probable. Los resultados obtenidos con esta metodología muestran una precisión muy elevada, **superando el 98 %** en todos los casos tal y como se muestra en la Ilustración 36:

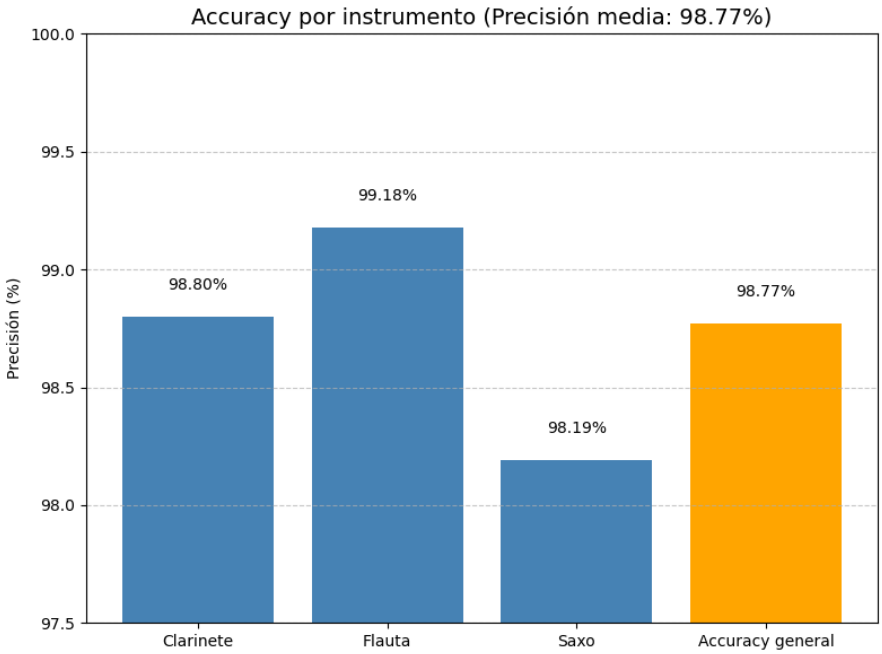


Ilustración 36

Estos valores reflejan una excelente capacidad de generalización del modelo por ventanas, incluso sin necesidad de recurrir a modelos de aprendizaje profundo en esta fase. La robustez observada se debe en parte al uso de filtros de frecuencia que atenúan armónicos no deseados y al diseño preciso de los tramos frecuenciales asignados a cada nota musical.

No obstante, el sistema presenta una limitación significativa en la detección de notas de muy corta duración. Dado que el intervalo de análisis recomendado es de 200 ms por ventana (equivalente a 5 notas por segundo), las notas que duran menos de 300 ms pueden no ser correctamente capturadas si su energía no queda contenida en al menos una ventana completa o si coinciden con los bordes entre ventanas. Esto afecta especialmente a pasajes rápidos con semicorcheas, apoyaturas o trinos, donde la precisión disminuye.

En conclusión, el modelo por ventanas demuestra ser una herramienta muy efectiva y precisa para la transcripción de instrumentos melódicos en contextos controlados o de tempo moderado. Su simplicidad, junto con la alta precisión alcanzada (98.77 %), lo convierten en una solución óptima para sistemas de análisis musical en tiempo diferido. Para contextos más rápidos o con micro-notas, se sugiere el desarrollo futuro de mecanismos de detección de onsets o adaptaciones dinámicas de la ventana de análisis.

Resultados para instrumentos polifónicos (guitarra)

El modelo desarrollado para instrumentos polifónicos (como la guitarra) ha sido entrenado utilizando un dataset compuesto por audios de nota única, en los que únicamente suena una nota por vez, sin interferencias de otras cuerdas. Este enfoque

ha permitido construir un clasificador robusto y preciso en un entorno controlado, alcanzando un accuracy elevado sobre el conjunto de validación, tal como se presenta en el apartado correspondiente.
No obstante, para evaluar la viabilidad del modelo en contextos reales, es fundamental analizar su comportamiento cuando se emplea en combinación con el sistema de análisis por ventanas temporales, que divide el audio completo en segmentos cortos y solapados (habitualmente de 200 ms) para realizar la predicción nota a nota.
En este entorno, las condiciones son más exigentes: los fragmentos pueden incluir transiciones entre notas, arrastres de sonido (slide, ligado), resonancias de cuerdas no pulsadas o superposición de armónicos, lo que complica el patrón espectral que el modelo ha aprendido a reconocer. Entre los principales retos se identifican:

- Contaminación armónica entre notas: especialmente relevante en guitarras, donde una cuerda pulsada puede excitar parcialmente otras cuerdas, generando componentes espectrales adicionales que el modelo puede malinterpretar.
- Transiciones rápidas: en pasajes con semicorcheas o ligados, las ventanas pueden capturar parcialmente dos notas distintas, dando lugar a predicciones inestables o ambiguas.
- Variabilidad tímbrica: la misma nota tocada en diferentes posiciones del mástil puede generar perfiles de armónicos distintos, dificultando la generalización del modelo si no ha sido entrenado con esa variabilidad.

A pesar de estas dificultades, los resultados obtenidos muestran un rendimiento aceptable al aplicar el modelo sobre melodías reales mediante ventanas. La estrategia de solapamiento y la estabilidad temporal introducida en el sistema permiten reducir errores puntuales y reconstruir una partitura razonablemente coherente. Sin embargo, se observa una ligera pérdida de precisión respecto al entorno de entrenamiento ideal, debido a los factores antes mencionados.

Además, como se ha mencionado anteriormente, el sistema ofrece al usuario la posibilidad de elegir entre tres modelos de clasificación distintos que tienen diferente desempeño tal y como se muestra en la Ilustración 37:

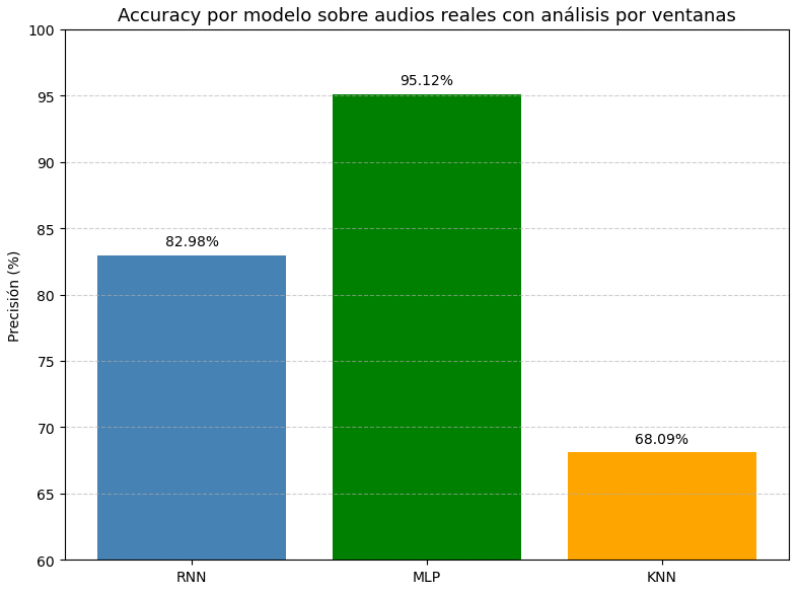


Ilustración 37

Estas cifras reflejan la capacidad de generalización de cada arquitectura ante condiciones acústicas más complejas que las del entrenamiento. En particular, el modelo MLP muestra una excelente adaptación al contexto real, seguido del modelo RNN, que ofrece un equilibrio entre precisión y estabilidad temporal.

Cabe destacar que, aunque el modelo KNN presenta el menor porcentaje de aciertos globales, es el que mejor identifica la nota fundamental en términos de altura relativa. El principal motivo de su bajo desempeño general se debe a que comete errores frecuentes en la identificación de la octava correcta. Es decir, acierta en la nota (Do, Re, Mi…) pero falla en determinar si pertenece, por ejemplo, a la tercera o cuarta octava, lo que penaliza el resultado final en términos de exactitud estricta. Esto sugiere que el modelo KNN, a pesar de su simplicidad, captura patrones relevantes en la estructura tonal, pero podría beneficiarse de una postcorrección en la predicción de octavas para mejorar su precisión global.

Métricas utilizadas y tiempos de ejecución

Uno de los aspectos fundamentales en la evaluación de la aplicación MelodyMapper ha sido el análisis de los tiempos de ejecución asociados al proceso completo de transcripción automática. Este proceso incluye tanto el preprocesado del audio como la clasificación de cada fragmento para determinar la nota correspondiente. El estudio detallado de estos tiempos permite valorar la viabilidad del sistema en distintos contextos de uso, desde entornos educativos hasta aplicaciones más exigentes o incluso interactivas.

En la Ilustración 38 se muestra el tiempo medio de ejecución del sistema completo al utilizar el modelo KNN. El comportamiento es claramente creciente con respecto al

número de ventanas procesadas. A medida que se incrementa la granularidad del análisis, el tiempo requerido se dispara, superando los 16 segundos al alcanzar las 2000 ventanas. Esto se debe a que el modelo KNN realiza comparaciones con todo el conjunto de entrenamiento para cada ventana, lo que conlleva un coste computacional significativo. Aunque este modelo presenta algunas virtudes en términos de identificación tonal básica, su escalabilidad resulta limitada.

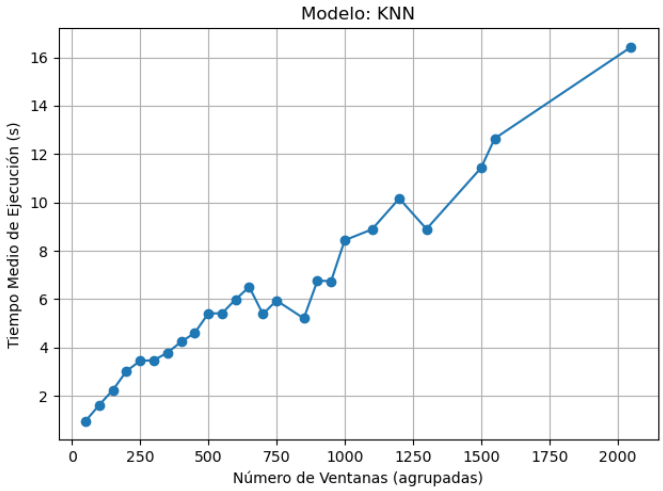


Ilustración 38

En la Ilustración 39 se representa el tiempo total de ejecución utilizando el modelo MLP. En este caso, se observa un comportamiento mucho más eficiente. Incluso con un número elevado de ventanas, el tiempo no supera los 14.5 segundos. La estructura del MLP permite un procesamiento más paralelo y directo, haciendo de esta arquitectura una opción muy competitiva cuando se requiere una buena combinación de precisión y velocidad.

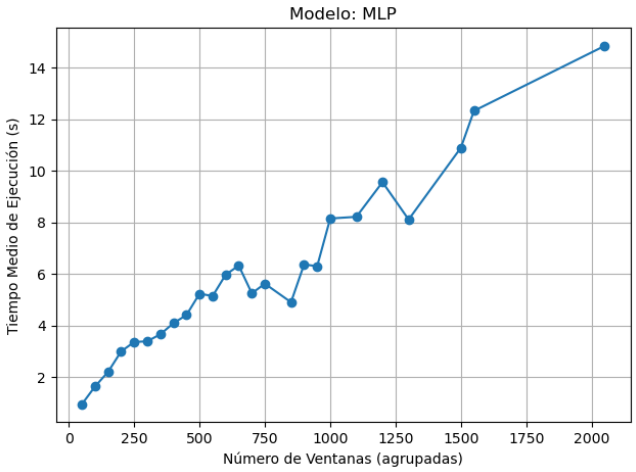


Ilustración 39

La Ilustración 40 recoge los resultados del modelo RNN, que muestra los tiempos más elevados de todo el sistema. Este modelo, diseñado para capturar dependencias temporales, introduce una carga computacional considerable. El tiempo de ejecución

escala rápidamente, alcanzando más de 100 segundos al procesar 2000 ventanas. Si bien puede ofrecer cierta estabilidad en la predicción de secuencias, su alto coste lo convierte en una opción poco práctica para la mayoría de los escenarios fuera de pruebas o entornos muy controlados.

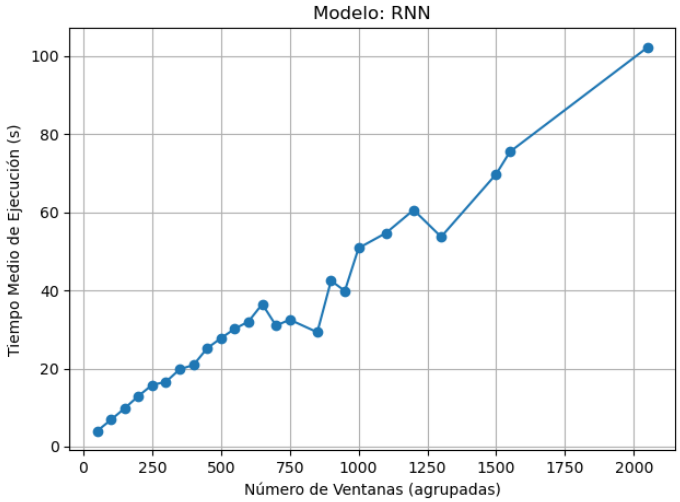


Ilustración 40

Por último, la Ilustración 41 corresponde al modelo Ventana, una aproximación basada en reglas determinísticas que analiza directamente las frecuencias dominantes de cada segmento mediante FFT y heurísticas simples. Este modelo resulta ser el más eficiente en términos temporales: incluso en configuraciones con 2000 ventanas, el tiempo total no supera los 3.6 segundos. Su ligereza lo hace extremadamente rápido, aunque su precisión es más limitada en comparación con los modelos de aprendizaje automático. Aun así, se comporta bien con instrumentos melódicos simples y en entornos con poco ruido.

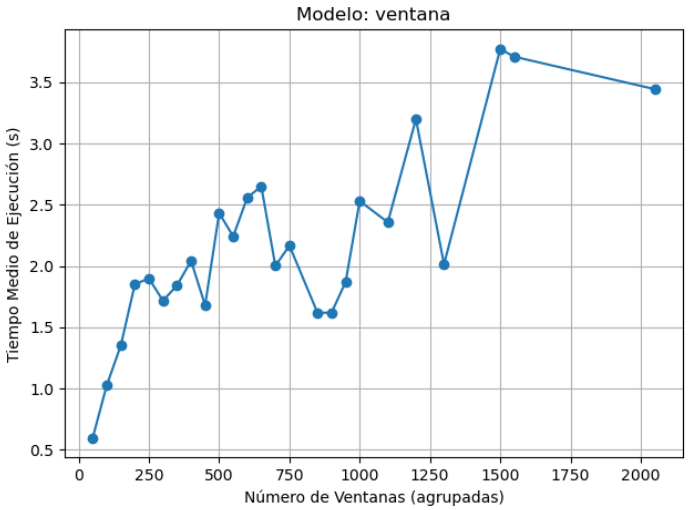


Ilustración 41

Una vez analizados los tiempos de ejecución, resulta imprescindible valorar la relación entre este coste computacional y la precisión ofrecida por cada modelo. El modelo

MLP representa el mejor equilibrio: presenta tiempos bajos y un accuracy alto en contextos reales, superando el 90 % en instrumentos polifónicos como la guitarra. Es ideal para sistemas en tiempo diferido, educativos o para usuarios que buscan fiabilidad sin comprometer la velocidad. El modelo RNN, aunque diseñado para capturar patrones temporales más complejos, no mejora de forma significativa el accuracy respecto al MLP y, sin embargo, multiplica por siete el tiempo de ejecución, lo que cuestiona su utilidad práctica fuera de contextos muy específicos. El modelo KNN ofrece un rendimiento modesto: acierta frecuentemente en la nota base, pero suele fallar en la octava, y su escalabilidad es limitada. Finalmente, el modelo Ventana, a pesar de ser el más rápido con diferencia, obtiene buenos resultados solo en audios simples, especialmente con instrumentos melódicos, donde su enfoque basado en picos resulta efectivo. Sin embargo, pierde precisión en pasajes rápidos, polifonía o grabaciones con ruido.

En conclusión, la elección del modelo debe realizarse considerando tanto el tiempo de ejecución como la calidad esperada en la transcripción. MelodyMapper permite adaptar su comportamiento según el caso de uso, y este análisis evidencia que MLP es, en general, la opción más equilibrada, mientras que Ventana puede ser útil en situaciones donde se prioriza la rapidez y el entorno es controlado. KNN y RNN, aunque válidos, requieren un mayor coste de procesamiento que debe justificarse por necesidades específicas.

Análisis de errores y casos límite

Durante el desarrollo y validación de MelodyMapper se han identificado distintos tipos de errores sistemáticos, así como situaciones límite que afectan significativamente la fiabilidad del sistema. Estos casos no solo evidencian las limitaciones técnicas actuales, sino que también orientan posibles mejoras futuras para aumentar la robustez y adaptabilidad del sistema ante condiciones reales.

Uno de los errores más frecuentes está relacionado con la proximidad entre notas adyacentes, especialmente cuando estas comparten una estructura armónica similar o tienen frecuencias muy próximas (en frecuencias muy graves). Este fenómeno afecta tanto al modelo basado en ventanas como a los modelos de aprendizaje automático. En particular, cuando dos notas tienen picos de energía cercanos en frecuencia (por ejemplo, Fa# y Sol), el sistema puede confundirse y asignar la nota incorrecta, especialmente si la señal está contaminada con armónicos o resonancias de fondo. Este problema se agrava en pasajes rápidos o cuando se producen transiciones muy cortas entre notas, en las que una ventana temporal puede capturar parcialmente ambas.

Otro caso límite observado se da en notas de muy corta duración, como semicorcheas o apoyaturas. En estos casos, si la nota no queda completamente contenida dentro de una sola ventana de análisis, o si coincide con el borde entre dos ventanas, es

posible que su energía quede diluida y no se reconozca como nota válida. Esta limitación afecta principalmente al modelo Ventana, pero también puede provocar predicciones erráticas en MLP y KNN, donde los vectores de entrada quedan distorsionados. En contextos melódicos simples esto no supone un gran problema, pero en interpretaciones rápidas o ligadas, se convierte en una fuente importante de errores.

En el caso de la guitarra y otros instrumentos polifónicos, se presentan errores derivados de la contaminación armónica. Cuando una cuerda es pulsada, puede excitar otras cuerdas de forma pasiva, generando componentes espectrales adicionales. Esto puede inducir al sistema a reconocer múltiples notas simultáneamente o a equivocarse en la selección de la nota predominante. Además, el uso de técnicas como el slide, el ligado o el vibrato produce transiciones espectrales complejas que no siempre son correctamente interpretadas, especialmente por los modelos más simples.

También se ha observado que el modelo KNN para guitarra, aunque capaz de capturar la nota base correcta, comete errores frecuentes en la clasificación de la octava. Esto se debe a la similitud entre los vectores de entrada de notas con igual estructura tonal pero diferente altura. Así, por ejemplo, puede predecirse un Sol4 cuando en realidad se está tocando un Sol3. Aunque este tipo de error no altera la esencia musical de la nota, penaliza la métrica de exactitud global del modelo, lo que debe tenerse en cuenta al interpretar los resultados.

Finalmente, en grabaciones con presencia de ruido ambiental o fondo musical, se ha observado que los modelos basados en reglas (Ventana) son más susceptibles a fallos, mientras que los modelos de aprendizaje, en particular el MLP, muestran mayor tolerancia. Sin embargo, en todos los casos, una mala calidad de la señal o un exceso de reverberación puede conducir a predicciones incorrectas o secuencias de notas “ZZZ” (nota no identificada).

En conjunto, estos errores y casos límite evidencian la necesidad de seguir perfeccionando los mecanismos de análisis espectral, segmentación adaptativa y entrenamiento del modelo, incluyendo datos más variados y realistas. Asimismo, se vislumbra la utilidad de incorporar un sistema de detección de confiabilidad que alerte sobre predicciones dudosas, o un módulo de postprocesado que revise las secuencias predichas para corregir inconsistencias armónicas o rítmicas. Este análisis no solo ayuda a entender los límites actuales del sistema, sino que también orienta su evolución hacia una herramienta más robusta y versátil ante las múltiples complejidades de la música real.

Capítulo 9. Manual de Usuario y Despliegue

Requisitos de instalación

Para ejecutar correctamente MelodyMapper se requiere contar con un entorno Python debidamente configurado, así como una serie de librerías especializadas para el tratamiento de audio, visualización y aprendizaje automático. A continuación, se detallan los requisitos mínimos necesarios:

1. **Sistema Operativo:** Compatible con Windows 10/11, macOS o distribuciones Linux modernas (Ubuntu 20.04+, Debian, etc.).
2. **Versión de Python:** Python 3.10 o superior. Se recomienda crear un entorno virtual para evitar conflictos con otras instalaciones.
3. **Librerías necesarias:** Las siguientes librerías de Python deben estar instaladas:
    - numpy, pandas: procesamiento numérico y estructuración de datos.
    - matplotlib, seaborn, plotly: visualización estática e interactiva.
    - librosa: análisis y procesamiento de señales de audio.
    - moviepy: conversión de formatos de audio y vídeo (mp3, wav, mp4).
    - scikit-learn, joblib: modelos KNN y preprocesamiento.
    - tensorflow, keras: ejecución de modelos RNN y MLP.
    - openpyxl: lectura y escritura de ficheros Excel (.xlsx) para los filtros.

Ejemplo de ejecución del sistema

Una vez instaladas todas las dependencias y colocados los archivos del proyecto en su estructura adecuada, el sistema puede ejecutarse mediante una interfaz web interactiva desarrollada con **Flask**.

Para lanzar la aplicación, basta con abrir una terminal dentro de la carpeta principal del proyecto (por ejemplo, pagina_web) y ejecutar el siguiente comando:

python app.py

Al hacerlo, el sistema iniciará un servidor local en modo desarrollo. En la terminal se mostrará un mensaje indicando que la aplicación está corriendo, junto con la dirección web local a la que puede accederse para utilizarla. Esto se muestra en la Ilustración 42:

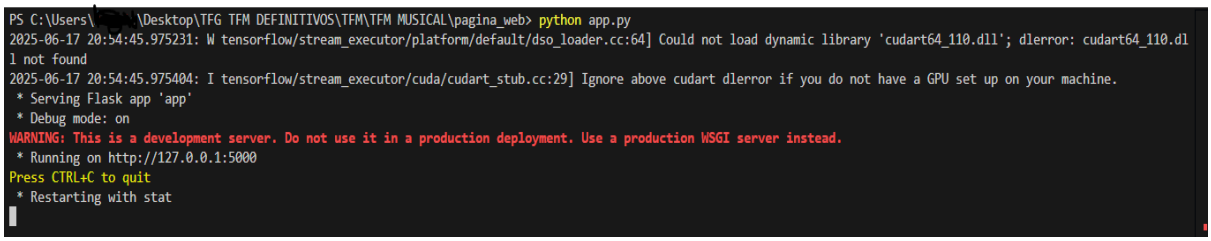


Ilustración 42

En este mensaje aparece también la dirección http://127.0.0.1:5000, que corresponde al servidor web local desplegado. Al abrir este enlace en cualquier navegador, el usuario accede a la interfaz gráfica de la herramienta, como se muestra en la Ilustración 43:

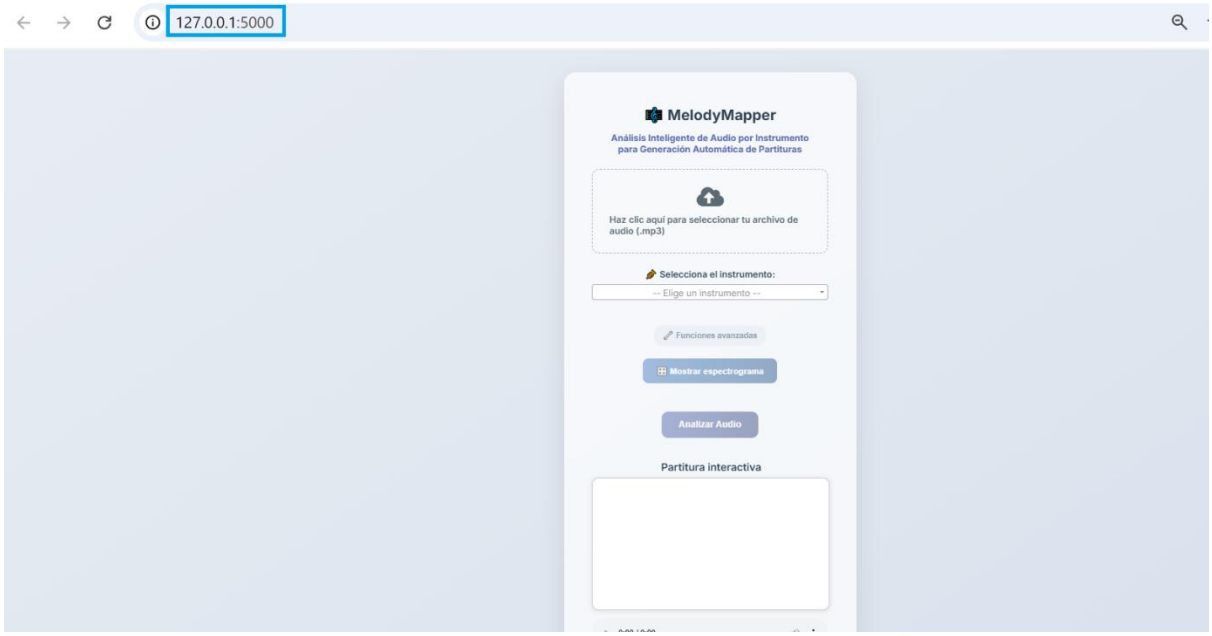


Ilustración 43

Desde esta interfaz, el usuario puede:

- Subir un archivo de audio en formato .mp3.
- Seleccionar el instrumento que desea analizar.
- Visualizar el espectrograma del audio cargado.
- Analizar automáticamente el contenido musical y obtener la partitura interactiva basada en el modelo seleccionado.
- Explorar funciones avanzadas y modificar parámetros de análisis en versiones extendidas de la app.

Esta solución facilita el uso del sistema incluso por usuarios sin conocimientos técnicos, encapsulando toda la lógica de preprocesamiento, predicción y visualización dentro de una interfaz amigable tal y como se muestra en la Ilustración 44:

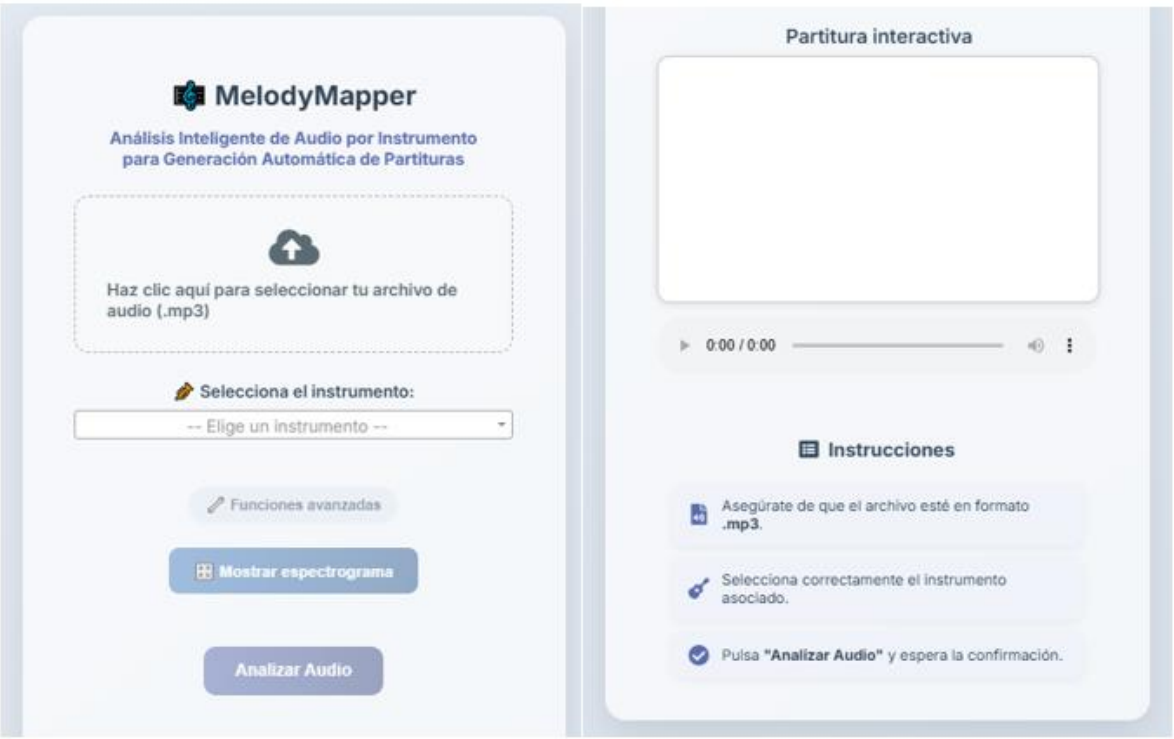


Ilustración 44

La interfaz web del sistema ha sido diseñada para ser intuitiva, funcional y apta para usuarios sin conocimientos técnicos. A continuación, se describe el procedimiento completo para utilizar la aplicación desde el navegador:

1.Subir un archivo de audio: Al abrir la aplicación en el navegador tal y como se muestra en la Ilustración 45, el usuario se encuentra con una interfaz clara bajo el nombre **MelodyMapper**. En la parte central de la pantalla aparece un recuadro con el texto:

"Haz clic aquí para seleccionar tu archivo de audio (.mp3)"

Aquí debe cargarse el archivo a analizar. Es fundamental que el archivo esté en formato .mp3, ya que otros formatos no están soportados por la versión actual.

Ilustración 45

2.Seleccionar el instrumento: Una vez subido el audio, el siguiente paso es seleccionar el instrumento que aparece en el archivo. El menú desplegable contiene una lista predefinida con los instrumentos soportados por el sistema, entre los que se encuentran:

- Guitarra
- Piano
- Violín
- Saxofón
- Trompeta
- Flauta
- Clarinete

Este selector se muestra en la Ilustración 46, donde el usuario ha desplegado la lista para elegir el instrumento correspondiente.

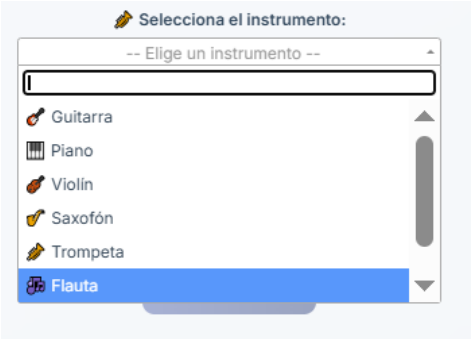


Ilustración 46

La elección correcta del instrumento es esencial, ya que el sistema aplicará modelos y configuraciones adaptadas a su rango de frecuencias y características tímbricas.

3.Activación de los botones de análisis: Una vez que se cumplen estas dos condiciones:

- Se ha cargado correctamente un archivo válido (.mp3)
- Se ha seleccionado un instrumento compatible,

el sistema valida los datos introducidos y, si todo es correcto, **se activan automáticamente** los siguientes botones:

- Funciones avanzadas: permite configurar parámetros adicionales.
- Mostrar espectrograma: representa visualmente el contenido espectral del audio.
- Analizar audio: ejecuta todo el pipeline de detección de notas y genera la partitura.

Esto se muestra en la Ilustración 47, donde tras subir el archivo mp3 y seleccionar "Saxofón", se habilitan los botones de análisis.

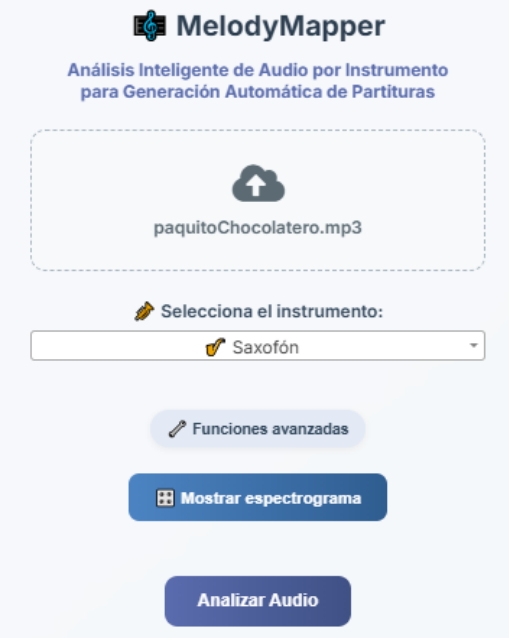


Ilustración 47

La aplicación ofrece la posibilidad de personalizar el proceso de análisis musical mediante la opción Funciones avanzadas. Al hacer clic en este botón, se despliega un menú configurable que permite al usuario ajustar una serie de parámetros técnicos clave para adaptar el análisis a las características específicas del archivo de audio. Esta funcionalidad se muestra en la Ilustración 48.

Ilustración 48

Segundo inicial / Segundo final: Permite definir el intervalo temporal del audio que se desea analizar. Esto es especialmente útil cuando se quiere ignorar silencio inicial, ruido de fondo, o simplemente acotar la sección útil del audio. Por ejemplo, si se desea analizar únicamente del segundo 5 al 30, basta con introducir esos valores.

Umbral mínimo (dB): Determina el nivel mínimo de intensidad (en decibelios) para considerar información válida del espectrograma. Valores más negativos permiten incluir más contenido, aunque también pueden introducir ruido. Un valor típico seguro es -500 dB.

Amplitud mínima: Se utiliza para descartar ventanas con niveles muy bajos de energía, que probablemente no contengan ninguna nota útil. Este filtro evita predicciones espurias en momentos de silencio o baja señal. Un valor razonable suele ser 20.

Frecuencia mínima / máxima (Hz): Define el rango de frecuencias que se analizará. Estos valores **se ajustan automáticamente según el instrumento** seleccionado siguiendo la Tabla 4. Por ejemplo:

Instrumento	Tipo	Frecuencia mínima (Hz)	Frecuencia máxima (Hz)	Notas aproximadas
Flauta	Melódico	250 Hz	2100 Hz	Do4 (C4) – Do7 (C7)
Saxofón alto	Melódico	130 Hz	850 Hz	Do3 (C3) – Fa6 (F6)
Clarinete	Melódico	150 Hz	1600 Hz	Mi2 (E2) – Do6 (C6)
Violín	Melódico	196 Hz	3136 Hz	Sol3 (G3) – Do7 (C7)
Guitarra	Polifónico	80 Hz	1000 Hz	Mi2 (E2) – Si5 (B5)

Tabla 4

Este ajuste automático garantiza que el análisis se centre únicamente en la zona relevante del espectro para cada instrumento, evitando artefactos de otras fuentes.

Frecuencia de muestreo (ms): Corresponde al tamaño de las ventanas de análisis, expresado en milisegundos. Una tasa más pequeña (200ms) implica ventanas más cortas, lo que ofrece mayor resolución temporal, pero puede introducir más ruido. Una tasa más baja (por ejemplo 100 ms) suaviza los resultados, pero puede perder notas breves.
El valor recomendado por defecto es 200 ms (5 notas por segundo)

Porcentaje de solape (0–1): Controla cuánto se solapan las ventanas entre sí. Un valor de 0.8 indica que cada ventana comparte un 80% de su contenido con la anterior.

- **Mayor solape (0.8 – 0.9)**: útil para instrumentos melódicos, sonidos sostenidos o grabaciones con mucho ruido, ya que suaviza los cambios bruscos.
- **Menor solape (0.2 – 0.5)**: útil cuando se desea priorizar la rapidez de ejecución o cuando las notas están claramente separadas.

Aplicar filtro: Permite activar el uso de un filtro espectral personalizado explicado en el apartado *6.5 Aplicación de filtros personalizados* usado para enfatizar o suprimir determinadas zonas del espectro.
El uso de este filtro es opcional y puede resultar útil para eliminar armónicos o mejorar la limpieza del audio.

Modelo para usar: Este campo depende del instrumento seleccionado. El sistema adapta automáticamente las opciones disponibles según el tipo de análisis requerido:

- Para **instrumentos melódicos** como saxofón o flauta, aparece la opción "Basado en Ventanas", ya que solo se analiza una nota a la vez por ventana.
- Para **instrumentos polifónicos** como guitarra o piano, el menú permite seleccionar entre diferentes modelos supervisados entrenados:
    - KNN: Modelo por similitud de vecinos.
    - RNN: Red neuronal recurrente para relaciones internas.
    - MLP: Perceptrón multicapa para clasificación.

Esta adaptación automática garantiza que el sistema utilice la estrategia de análisis más apropiada según la naturaleza del instrumento.

4.Evaluación previa del audio mediante espectrograma: Antes de ejecutar el análisis completo y generar la partitura automática, el sistema ofrece al usuario la posibilidad de visualizar el espectrograma del archivo de audio cargado. Esta funcionalidad resulta especialmente útil para evaluar la calidad del audio y comprobar si presenta problemas como:

- Ruidos de fondo excesivos.
- Distorsiones.
- Notas poco diferenciadas.
- Grabaciones mal recortadas o con silencios largos.

Al pulsar el botón “Mostrar espectrograma”, se representa una vista logarítmica del contenido en frecuencia y tiempo del audio seleccionado, permitiendo así al usuario inspeccionar visualmente la señal antes de procesarla con los modelos tal y como se muestra en la Ilustración 49 se muestra una comparativa de como aparece un mismo audio filtrando más por amplitud y umbral mínimo que en otro y como en el espectrograma de la derecha aparece una señal más nítida y clara donde se puede seguir la melodía de la canción:

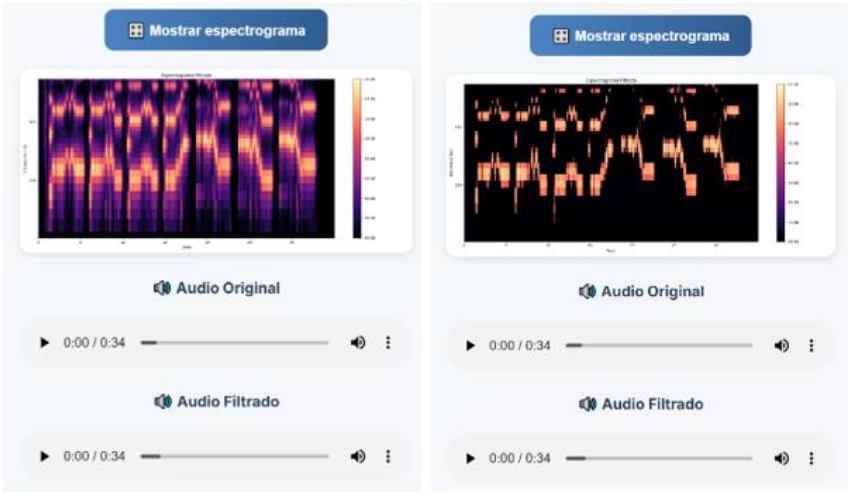


Ilustración 49

Reconfiguración flexible: Una de las ventajas del sistema es que **el espectrograma puede regenerarse tantas veces como el usuario desee**. Esto permite:

1. **Ajustar parámetros de análisis** (por ejemplo, rango de frecuencias, umbral mínimo, duración del análisis o solape).
2. Volver a pulsar **“Mostrar espectrograma”** y observar el efecto de los cambios realizados.
3. Repetir este proceso de prueba y error hasta encontrar la configuración óptima.

Este enfoque es especialmente útil cuando se trabaja con grabaciones no limpias o no estandarizadas, como audios grabados con móvil, en directo o con instrumentos poco definidos.

5.Confirmación final antes del análisis: Una vez el usuario esté satisfecho con la visualización del espectrograma y la configuración seleccionada, podrá finalmente pulsar el botón **“Analizar audio”** para lanzar el análisis automático mediante los modelos internos del sistema.

Este paso es clave, ya que evita ejecutar el análisis sin un control previo y proporciona al usuario una forma de verificar la calidad del audio y de los parámetros elegidos. Este flujo garantiza mayor **fiabilidad** y **precisión interpretativa** en los resultados.

Al completarse el análisis, el sistema genera automáticamente dos elementos principales:

- Una **partitura en formato piano-roll**, donde cada barra horizontal representa una nota detectada, su duración y su posición en el tiempo.
- Una **visualización interactiva sincronizada con el audio**, permitiendo al usuario escuchar la grabación mientras observa cómo se desplaza una línea roja que recorre la partitura en tiempo real.

Esta partitura interactiva se muestra en la Ilustración 50, donde puede verse cómo las notas se distribuyen a lo largo del tiempo, facilitando una lectura visual clara de la ejecución musical.

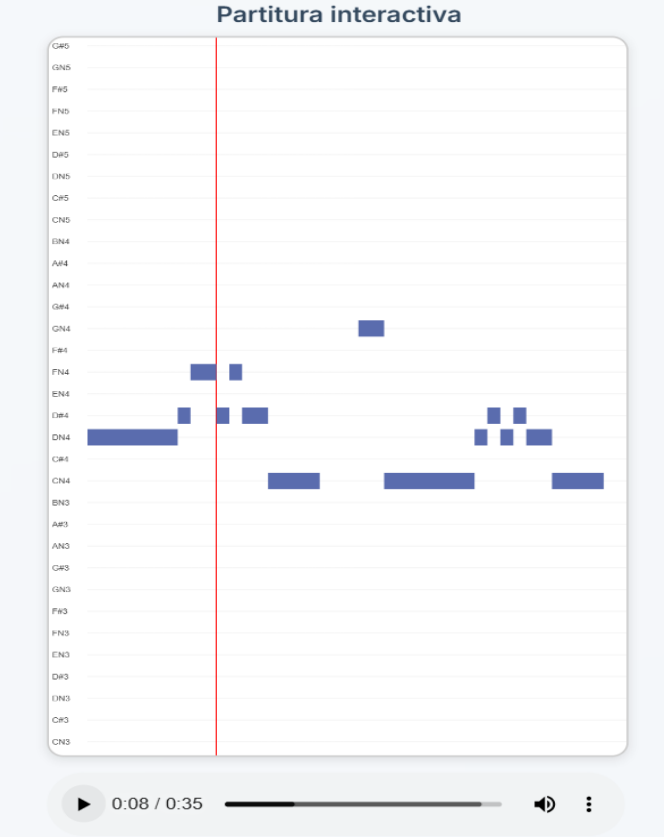


Ilustración 50

Este componente visual no solo mejora la experiencia del usuario, sino que permite también una validación intuitiva de la transcripción automática realizada. Gracias al diseño basado en piano-roll, incluso usuarios sin conocimientos musicales avanzados pueden identificar visualmente patrones, repeticiones o errores en la detección de notas.

Capítulo 10. Conclusiones y Trabajos Futuros

El presente proyecto ha culminado con el desarrollo de un sistema funcional capaz de analizar grabaciones de audio y transformarlas en partituras musicales legibles. Esta herramienta aborda tanto instrumentos melódicos, como la flauta travesera, que producen una nota a la vez, como instrumentos polifónicos, como la guitarra, que permiten la ejecución simultánea de varias notas. A través de la combinación de técnicas de procesamiento digital de señales (PDS) y algoritmos de inteligencia artificial (IA), se ha demostrado que es posible automatizar procesos tradicionalmente complejos como la transcripción musical, reduciendo significativamente el esfuerzo manual requerido.

Uno de los principales logros ha sido la creación de una arquitectura modular y escalable. Esto ha permitido incorporar múltiples modelos de aprendizaje automático (KNN, MLP, RNN), adaptándose así a las características específicas de cada instrumento y contexto musical. Asimismo, el sistema incluye funciones avanzadas para la generación de espectrogramas, la segmentación temporal del audio, y la identificación de frecuencias fundamentales mediante análisis en el dominio de Fourier, todo ello de forma automatizada y flexible.

Adicionalmente, se han desarrollado herramientas de visualización interactivas basadas en Plotly, que facilitan la interpretación de los resultados por parte del usuario. También se ha implementado un sistema de filtrado personalizado en el dominio de la frecuencia, lo que mejora significativamente la calidad del análisis, especialmente en ambientes con ruido o con presencia de armónicos. En conjunto, el sistema constituye una base sólida para futuras ampliaciones y mejoras en el ámbito de la transcripción musical automática asistida por IA.

Aportaciones personales

Entre las aportaciones más destacadas se encuentra la implementación desde cero de todo el pipeline de procesamiento de audio, que incluye la conversión y segmentación de archivos, la aplicación de filtros espectrales personalizados, y la extracción de características relevantes en el dominio de la frecuencia. También he desarrollado el sistema de predicción con tres modelos distintos (KNN, MLP y RNN), validando sus rendimientos respectivos y adaptando sus entradas mediante técnicas como normalización, padding y selección de amplitudes relevantes.

Por último, he trabajado en la representación visual de los datos de audio y sus transformaciones, empleando herramientas como Plotly para la creación de gráficos interactivos, y desarrollando funciones que permiten generar espectrogramas filtrados y piano rolls simplificados. Todas estas aportaciones no solo han contribuido a la creación de un producto técnico funcional, sino que también han fortalecido mis

competencias en ingeniería de datos, aprendizaje automático y procesamiento digital de señales, consolidando una experiencia integral en un campo interdisciplinar.

Limitaciones encontradas

Una de las principales limitaciones del sistema desarrollado es su dependencia de datos etiquetados y bien segmentados para entrenar y validar los modelos de aprendizaje automático. Especialmente en el caso de instrumentos polifónicos como la guitarra, la obtención de grabaciones limpias y bien anotadas resulta costosa y laboriosa, lo que puede afectar a la capacidad del modelo para generalizar a nuevas grabaciones con diferente timbre, interpretación o calidad.

Además, si bien el sistema es capaz de identificar correctamente notas individuales en condiciones controladas, su precisión disminuye en grabaciones con ruido de fondo, armónicos no deseados o superposición de frecuencias. Estos elementos introducen complejidad en la señal y pueden dificultar la detección de la frecuencia fundamental, especialmente cuando varias notas suenan al mismo tiempo o en registros similares.

Otro reto importante ha sido la ausencia de un sistema robusto para la detección de la duración y el ritmo de las notas. El enfoque actual se basa en la identificación de notas por bloques temporales discretos, lo que impide capturar con precisión la estructura rítmica completa de una interpretación musical. Esta limitación condiciona la fidelidad de la partitura generada, reduciendo su utilidad para fines interpretativos o educativos más avanzados.

Por último, aunque se ha implementado un sistema de filtrado en frecuencia que mejora el rendimiento en muchos casos, su ajuste sigue siendo manual y dependiente del instrumento y del tipo de audio. Esto requiere una cierta experiencia técnica por parte del usuario y limita la automatización del proceso. La generalización del sistema a otros instrumentos o contextos musicales aún requiere ajustes específicos y pruebas adicionales.

Propuestas de mejora y ampliación

A pesar de los resultados satisfactorios obtenidos, el sistema desarrollado presenta un amplio margen de mejora y expansión. Algunas funcionalidades pueden perfeccionarse para aumentar la precisión y aplicabilidad del sistema, mientras que otras aún no han sido abordadas pero representan oportunidades claras para enriquecer el proyecto. A continuación, se detallan diversas propuestas concretas orientadas a mejorar el rendimiento, ampliar la compatibilidad con otros instrumentos

y facilitar su uso por parte de usuarios no técnicos. Estas mejoras permitirían evolucionar el sistema hacia una herramienta más completa, precisa y accesible.

1. **Separación de fuentes sonoras**: Para abordar audios reales con múltiples instrumentos, es recomendable integrar módulos de separación de fuentes (source separation) como Open-Unmix o Spleeter. Esto permitiría aplicar el modelo a pistas completas o conciertos, separando primero cada instrumento y luego aplicando el algoritmo de transcripción individualmente sobre cada uno.
2. **Mejora del sistema de filtrado:** Actualmente, los filtros se definen y aplican manualmente. Una ampliación útil sería crear un sistema de ajuste automático de filtros basado en la estadística del espectro de entrada, o mediante aprendizaje automático que determine qué frecuencias son relevantes para cada instrumento y contexto. También sería interesante permitir filtros adaptativos por instrumento o estilo musical.
3. **Soporte para más instrumentos y modelos personalizados:** Aunque el sistema soporta flauta y guitarra, se podría extender fácilmente a otros instrumentos melódicos y polifónicos (como violín, piano, saxofón). Para ello, sería necesario recopilar nuevos datasets y entrenar modelos específicos, ajustando los rangos de frecuencia, umbrales de amplitud y arquitectura del modelo (por ejemplo, CNNs para piano).

Exportación en formato de partitura real (MusicXML o MIDI): Actualmente, la partitura generada es textual. Una mejora concreta sería implementar la exportación automática en formatos estándar como MIDI o MusicXML, lo cual permitiría su visualización en editores musicales como MuseScore o Finale, facilitando su uso en contextos educativos o interpretativos.

Bibliografía

Ableton. (n.d.). *Ableton Live.* Retrieved from ableton.com: https://www.ableton.com/en/live/

Allen, J. B. (1977). A unified approach to short-time Fourier analysis and synthesis. *Proceedings of the IEEE*, 1336–1337.

Bello, J. P. (2005). A robust mid-level representation for harmonic content. *IEEE Transactions on Audio, Speech, and Language Processing*, 650–661.

Benetos, E. (n.d.). Automatic Music Transcription. *JIIS-MIRrors*.

Benetos, E. D. (2013). Automatic music transcription: challenges and future directions. *Journal of Intelligent Information Systems*.

Bishop, C. M. (2006). *Pattern Recognition and Machine Learning.* New York: Springer.

Bishop, C. M. (2006). *Pattern Recognition and Machine Learning.* New York: Springer.

Bosi, M. &. (2003). *Introduction to Digital Audio Coding and Standards.* New York: Springer.

Boulanger-Lewandowski, N. (2012). *Modeling Temporal Dependencies in High-Dimensional Sequences: Application to Polyphonic Music Generation and Transcription.* Montréal: Université de Montréal.

Brandenburg, K. (1999). MP3 and AAC Explained. *AES 17th International Conference on High-Quality Audio Coding*.

Celemony Software GmbH / Lunaverus / Ableton / Magenta. (n.d.). *Melodyne / AnthemScore / Ableton Live / Onsets and Frames.* Retrieved from celemony.com / lunaverus.com / ableton.com / magenta.tensorflow.org.

Celemony Software GmbH. (n.d.). *Melodyne.* Retrieved from celemony.com: https://www.celemony.com/en/melodyne/what-is-melodyne

Centre for Digital Music, Queen Mary University of London. (n.d.). *Sonic Visualiser.* Retrieved from sonicvisualiser.org: https://www.sonicvisualiser.org/

Choi, K. F. (2017). Convolutional recurrent neural networks for music classification. *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*.

Chung, J. G. (2014). Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling. *arXiv preprint*.

Ciudad Pentagrama. (2020). *Tabla de frecuencias de notas musicales.* Retrieved from Ciudad Pentagrama: https://www.ciudadpentagrama.com/2020/01/tabla-frecuencias-notas-musicales.html

Developers, N. (n.d.). *NumPy.* Retrieved from numpy.org: https://numpy.org

Developers, N. (n.d.). *numpy.org.* Retrieved from https://numpy.org

Developers, S. (n.d.). *PySoundFile.* Retrieved from https://pysoundfile.readthedocs.io

Developers, S. (n.d.). *SciPy.* Retrieved from scipy.org: https://scipy.org

Duda, R. O. (2001). *Pattern Classification.* New York: Wiley.

Foundation, P. S. (n.d.). *Python.* Retrieved from python.org: https://www.python.org

Foundation, P. S. (n.d.). *Python pickle module.* Retrieved from https://docs.python.org/3/library/pickle.html

Fujishima, T. (1999). Realtime chord recognition of musical sound: A system using common Lisp music. *Proceedings of the International Computer Music Conference*.

Géron, A. (2019). *Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow.* Sebastopol: O'Reilly Media.

Gómez, E. (2006). Tonal description of music audio signals. *PhD Thesis, Universitat Pompeu Fabra*.

Goodfellow, I. B. (2016). *Deep Learning.* Cambridge, MA: MIT Press.
Google Brain / Magenta. (2017). *NSynth: Neural Audio Synthesis.* Retrieved from magenta.tensorflow.org: https://magenta.tensorflow.org/datasets/nsynth
Google Brain Team. (n.d.). *tensorflow.org.* Retrieved from https://www.tensorflow.org/
Hall, D. E. (2002). *Musical Acoustics.* Pacific Grove: Brooks Cole.
Hawthorne, C. e. (2017). Onsets and Frames: Dual-Objective Piano Transcription. *Proceedings of the International Society for Music Information Retrieval (ISMIR)*.
Howard, D. M. (2017). *Acoustics and Psychoacoustics.* London: Routledge.
Huang, C. Z. (2019). Music Transformer: Generating Music with Long-Term Structure. *International Conference on Learning Representations (ICLR)*.
Humphrey, E. J. (2012). Moving beyond feature design: Deep architectures and automatic feature learning in music informatics. *Proceedings of the International Society for Music Information Retrieval (ISMIR)*, 403–408.
Hunter, J. D. (n.d.). *matplotlib.org.* Retrieved from https://matplotlib.org
Joblib Developers. (n.d.). *joblib.readthedocs.io.* Retrieved from https://joblib.readthedocs.io
Klapuri, A. (2006). Multiple Fundamental Frequency Estimation. *Journal of the Audio Engineering Society*, 112–126.
Lee, H. P. (2009). Unsupervised feature learning for audio classification using convolutional deep belief networks. *Advances in Neural Information Processing Systems (NeurIPS)*.
Lunaverus. (n.d.). *AnthemScore.* Retrieved from lunaverus.com: https://www.lunaverus.com
Lyons, R. G. (2011). *Understanding Digital Signal Processing.* Boston: Prentice Hall.
McFee, B. e. (n.d.). *Librosa.* Retrieved from librosa.org: https://librosa.org
Müller, M. (2015). *Fundamentals of Music Processing.* Berlin: Springer.
Oppenheim, A. V. (2010). *Discrete-Time Signal Processing.* Upper Saddle River, New Jersey: Pearson.
pandas-dev. (n.d.). *pandas.pydata.org.* Retrieved from https://pandas.pydata.org
Pedregosa, F. e. (n.d.). *scikit-learn.org.* Retrieved from https://scikit-learn.org/
Pierce, J. R. (1983). *The Science of Musical Sound.* New York: W.H. Freeman.
Plotly Technologies Inc. (n.d.). *plotly.com.* Retrieved from https://plotly.com
Pohlmann, K. C. (2010). *Principles of Digital Audio.* New York: McGraw-Hill.
Proakis, J. G. (2007). *Digital Signal Processing: Principles, Algorithms, and Applications.* New Jersey: Pearson.
Projects, P. (n.d.). *Flask.* Retrieved from flask.palletsprojects.com: https://flask.palletsprojects.com/
Rabiner, L. R. (2011). *Theory and Applications of Digital Speech Processing.* Upper Saddle River: Prentice Hall.
Riehl, J. (n.d.). *Pydub.* Retrieved from https://github.com/jiaaro/pydub
Rossing, T. D. (2019). *The Science of Sound.* Boston: Pearson.
Rumelhart, D. E. (1986). Learning representations by back-propagating errors. *Nature*, 533–536.
Sethares, W. A. (2005). *Tuning, Timbre, Spectrum, Scale.* London: Springer.
sigsep. (2024). *GitHub - Open-Unmix.* Retrieved from GitHub: https://github.com/sigsep/open-unmix-pytorch
Sigtia, S. B. (2016). An end-to-end neural network for polyphonic piano music transcription. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 1744–1753.
Slaney, M. (1998). Auditory Toolbox. *Interval Research Corporation Technical Report*.

Smith, J. O. (2007). *Mathematics of the Discrete Fourier Transform (DFT).* Stanford, CA: W3K Publishing.

Smith, J. O. (2011). *Spectral Audio Signal Processing.* Stanford, CA: W3K Publishing.

Team, G. (n.d.). *Gunicorn WSGI HTTP Server.* Retrieved from gunicorn.org: https://gunicorn.org

Unbit. (n.d.). *uWSGI Project.* Retrieved from uwsgi-docs.readthedocs.io: https://uwsgi-docs.readthedocs.io

van den Oord, A. e. (2016). WaveNet: A Generative Model for Raw Audio. *arXiv*.

Zölzer, U. (2008). *Digital Audio Signal Processing.* Chichester: Wiley.

Zulko. (n.d.). *MoviePy.* Retrieved from zulko.github.io/moviepy: https://zulko.github.io/moviepy/

Anexos

Contribución del proyecto a los Objetivos de Desarrollo Sostenible (ODS)

El presente proyecto, titulado “Análisis musical basado en IA a partir de señales de audio”, se enmarca en una visión tecnológica con impacto social y educativo, alineándose con varios Objetivos de Desarrollo Sostenible (ODS) definidos por la Organización de las Naciones Unidas. A través de la aplicación de inteligencia artificial y técnicas de procesamiento digital de señales para el análisis y transcripción automática de música, el proyecto contribuye de forma directa a los siguientes ODS:

- **ODS 4: Educación de calidad:** Este proyecto promueve el acceso a herramientas educativas innovadoras que facilitan el aprendizaje musical. Al automatizar la transcripción de partituras a partir de grabaciones, se facilita el estudio y la comprensión de piezas musicales incluso a personas con conocimientos limitados en notación musical. Asimismo, puede ser utilizado como recurso didáctico por docentes y estudiantes, democratizando el acceso a la educación musical, especialmente en contextos con recursos limitados.
- **ODS 9: Industria, innovación e infraestructura:** El sistema desarrollado representa un avance en la integración de tecnologías emergentes como el aprendizaje automático y el procesamiento de señales en el ámbito artístico. Promueve la innovación en la industria musical y educativa, al ofrecer nuevas formas de interacción con el sonido y la música. Además, fomenta el desarrollo de infraestructura digital útil tanto para investigación como para aplicaciones comerciales o educativas.
- **ODS 10: Reducción de las desigualdades:** El proyecto contribuye a reducir desigualdades al ofrecer una herramienta accesible que puede ser utilizada por músicos, estudiantes y docentes independientemente de su nivel económico o ubicación geográfica. Al reducir la dependencia de software comercial costoso y automatizar procesos complejos, se eliminan barreras que suelen limitar el acceso a la formación musical formal.
- **ODS 17: Alianzas para lograr los objetivos:** El carácter modular y abierto del sistema permite que este proyecto sea fácilmente compartido, adaptado y ampliado por otras instituciones educativas, centros de investigación o comunidades desarrolladoras. Esto fomenta la cooperación entre distintos actores del ecosistema tecnológico y educativo, impulsando alianzas que pueden fortalecer el impacto colectivo de este tipo de iniciativas.