Reconocimiento automático de notas del piano

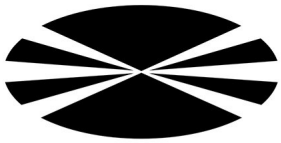
David Rodríguez Bacelar

Kevin Millán Canchapoma

Luca D'angelo Sabín

Jorge Hermo González

15 de mayo de 2022



UNIVERSIDADE DA CORUÑA

Índice

1. Glosario **3**

2. Introducción **3**

3. Descripción del problema **5**
3.1. Restricciones 6
3.2. Características 6

4. Análisis bibliográfico **7**

5. Desarrollo **9**
5.1. Primera aproximación 9
5.1.1. Descripción 9
5.1.2. Resultados 12
5.1.3. Discusión 14
5.2. Segunda aproximación 17
5.2.1. Descripción 17
5.2.2. Resultados 21
5.2.3. Discusión 23
5.3. Tercera aproximación 24
5.3.1. Descripción 24
5.3.2. Resultados 30
5.3.3. Discusión 32
5.4. Cuarta aproximación 34
5.4.1. Descripción 34
5.4.2. Resultados 40
5.4.3. Discusión 42
5.5. Quinta aproximación 43
5.5.1. Descripción 43
5.5.2. Resultados 47
5.5.3. Discusión 49
5.6. Sexta aproximación 50
5.6.1. Descripción 50
5.6.2. Resultados 51
5.6.3. Discusión 55

6. Conclusiones **59**

7. Trabajo futuro **60**

8. Bibliografía **62**

1. Glosario

- Sample: muestra de carácter musical
- Dataset: Conjunto de datos
- Cover: Reinterpretación de una canción por parte de alguien diferente al que la compuso.
- VP: Verdadero positivo
- VN: Verdadero negativo
- FP: Falso positivo
- FN: Falso negativo
- RNA: Red de neuronas artificiales.
- SVM: *Support Vector Machine* (Máquina de soporte vectorial).
- kNN: *K-nearest neighbors* (k vecinos más cercanos).
- poly: *Kernel* polinomial.
- rbf: *Kernel* de función de base radial (Gaussiano)
- sigmoid: *Kernel* sigmoidal

2. Introducción

A raíz de la pandemia, el aumento del interés por el aprendizaje en diferentes ámbitos llegó tambíen a la música, y con él, la aparición de herramientas para aprender a tocar diferentes instrumentos de forma autodidacta.

Así, para cualquiera que esté aprendiendo, el escuchar una canción que te gusta e intentar tocarla es algo que acaba siendo un proceso frustante y que requiere una gran cantidad de horas intentando sacar las notas que la componen.

Nuestro sistema se encargaría entonces de reconocer y diferenciar a partir de audios las notas de una pieza de piano pudiendo, en un futuro, ser capaz de detectar acordes y tonalidades, siendo útil en aplicaciones como Spotify, Tidal... Para ello, haremos uso de diferentes técnicas de aprendizaje automático como las redes de neuronas artificiales, árboles de decisión y máquinas de soporte vectorial, comparando su rendimiento y eligiendo la que mejor resultados nos ofrezca.

A lo largo de esta memoria analizaremos a fondo el problema a resolver en la Sección 3, desarrollaremos las diferentes soluciones en la Sección 5, hablaremos sobre las

conclusiones del trabajo en la Sección 6 y finalizaremos comentando las aplicaciones al mundo real en la Sección 7. También se podrá consultar las bibliografía utilizada en la Sección 4 y 8.

Para el análisis utilizaremos distintas técnicas de aprendizaje supervisado como son:

- **Redes de Neuronas Artificiales** o *RNAs*, un modelo de aprendizaje automático clásico que intenta simular el funcionamiento natural del cerebro, haciendo uso de estrucutras como las neuronas artificiales y en la que la información se almacena y modifica en los pesos de las conexiones de dichas neuronas. Ver Krogh, 2008.
- **Support Vector Machine** o *SVMs*, un modelo basado, como en las *RNAs*, en la división del espacio del problema para clasificar los diferentes patrones y del que se puede profundizar más en el artículo que las introduce de Cortes y Vapnik, 1995.
- **Árboles de decisión**, técnicas que permiten también subdividir el espacio donde se sitúan los patrones, esta vez siguiendo una estructura de árbol donde se va tomando una rama u otra en función del valor de los distintos atributos. Ver Myles y col., 2004.
- **k-Nearest Neighbors** o *k-NN*, un método con un enfoque diferente, basado en instancias, donde la clasificación se realiza en función de los patrones que más se parecen al de entrada. Ver Guo y col., 2003.

3. Descripción del problema

Nuestro sistema se centrará en reconocer, a partir de un audio, la nota del piano que se está tocando. Escogimos este instrumento por la cantidad de recursos que podemos encontrar y por su naturaleza invariable al ser tocada por una u otra persona.

Dada la naturaleza de nuestro sistema, descartamos utilizar la especificidad o la sensibilidad ya que nos es indiferente las clases en las que se clasifiquen las notas (el coste de un falso positivo o un falso negativo es el mismo).

Así, como solo nos interesa una correcta clasificación global, pensamos en utilizar la precisión, la cual sigue la fórmula:

$$Precision = \frac{VN + VP}{(VN + FN + VP + FP)} \tag{1}$$

El incoveniente de esta métrica está en que si tenemos un conjunto de patrones desbalanceado (gran diferencia en el número de patrones positivos y negativos), la precisión podría alcanzar valores muy altos con sistemas que clasifiquen todos los patrones en la clase con mayor número de ellos en el entrenamiento.

La métrica que utilizaremos entonces para valorar los resultados obtenidos y que palia los problemas mencionados anteriormente será la **F1-score**. Esta se corresponde con la media armónica de la sensibilidad y el valor predictivo positivo y está caracterizada por la fórmula:

$$F1 - Score = \left(\frac{Sensibilidad^{-1} + VPP^{-1}}{2} \right)^{-1} \tag{2}$$

donde,

$$Sensibilidad = \frac{VP}{(FN + VP)} \tag{3}$$

$$VPP = \frac{VP}{(VP + FP)}, \tag{4}$$

Quizá el único inconveniente de esta métrica es su difícil interpretación, más allá de comparar los valores que ofrecen diferentes sistemas. El valor más alto que puede alcanzar es de 1 (todos los patrones se clasificaron correctamente) y el más bajo es de 0 (todos los patrones se clasificaron incorrectamente). Entonces, para facilitar la interpretación de la métrica, también se mostrará la precisión de cada modelo, pero se utilizará el F1-Score para determinar el mejor.

3.1. Restricciones

Como única restricción, en dicho audio solo puede haber una nota sonando a la vez para que el sistema sea capaz de reconocerla correctamente.

3.2. Características

La base de datos con la que contamos tiene un total de 5.406 audios con una media de más de 50 *samples* por cada una de las 85 notas del piano (a partir de *C1*), tocadas desde posiciones e intensidades distintas y grabadas con micrófonos diferentes. Dichos *samples* están en formato *.wav* en estéreo, con un *bitrate* de *2304kbps*, *24 bits per sample* y un *sample rate* de *48kHz*. Todo ello ocupa un total de *34.5GB* en disco. La duración de los *samples* es de 23,735 $\pm$ 14,754 segundos, siendo la duración mínima de 1,669 segundos y la duración máxima de 73,646 segundos.

Se puede ver un ejemplo del espectrograma de una de las *samples* que vamos a utilizar en la Figura 1.

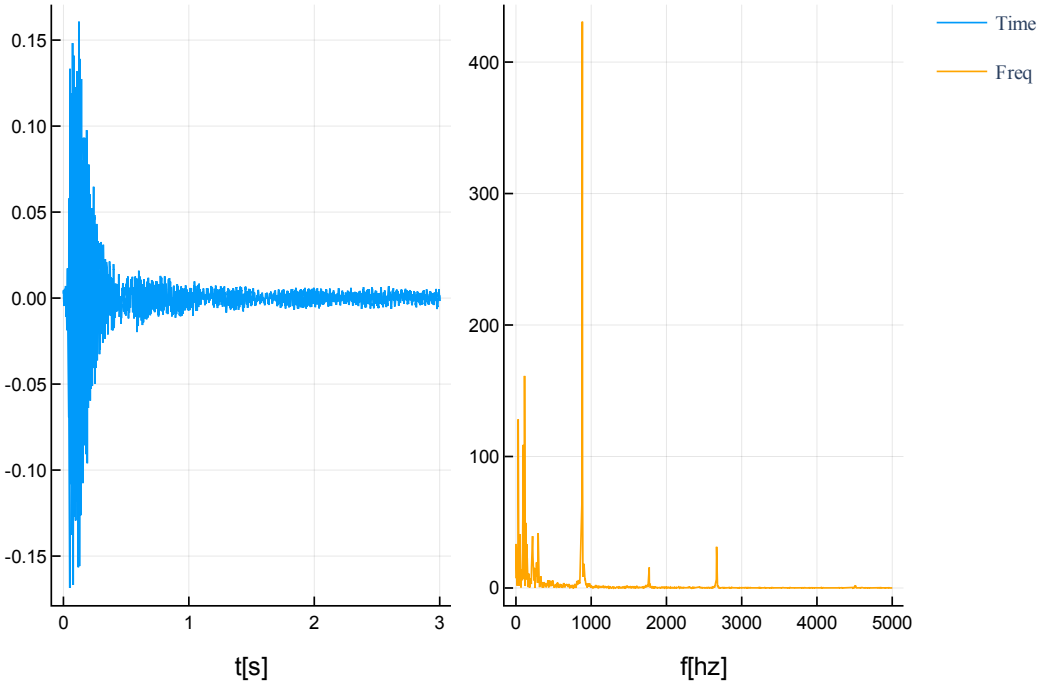


Figura 1: Notas A5 en tiempo y frecuencia

El origen de la base de datos es una librería de piano de la compañía *FluffyAudio*

(https://www.fluffyaudio.com/shop/scoringpiano) grabada en 2016 y pensada para jazz, música clásica y bandas sonoras.

4. Análisis bibliográfico

Para profundizar en el tema antes de abordarlo, en esta sección se analizan diferentes artículos científicos relacionados con la inteligencia artificial y el reconocimiento de audio, ya sea específicamente relacionado o no con el mundo del reconocimiento de piezas o notas musicales.

Trabajos como el de Osmalsky y col., 2012 nos aportan nuevos enfoques, en el que, en lugar de detectar las notas por separado, analizan todo el espectro de frecuencias para poder reconocer acordes completos de diferentes instrumentos; utilizan una técnica llamada Pitch Class Profile (PCP), que obtiene las relaciones energéticas de cada nota en la escala a partir de un audio.

Además, como resumen Benetos y col., 2018, a pesar del estado avanzado de la transcripción automática de canciones, aún están presentes retos tales como la independencia de los intrumentos, de los estilos musicales o la interpretación de la expresividad.

Otros trabajos mas antiguos como los de Foo y Wong, 1999, describen un algoritmo capaz de reconocer notas de un piano a partir de piezas sintetizadas o acústicas, que son digitalmente muestreadas y transformadas al dominio de frecuencia usando la transformada de Q constante a partir de la cual se la aplican diferentes técnicas para identificar las notas.

En un terreno más general, trabajos como el de Chang y col., 2017, exploran la identificación de *covers* utilizando estructuras más novedosas que no desarrollaremos en este trabajo como son las Redes de Neuronas Convolucionales. Las salidas de este sistema corresponderían con la probabilidad de ser una *cover* (comparándola con la canción original), y se ordenarían por dicha probabilidad elaborando un *ranking*.

También encontramos la tesis de Klapuri, 2004 que propone un sistema capaz de generar una representación simbólica a partir de un audio, centrándose en el desarrollo de los algoritmos que pueden ser usados para detectar y observar sonidos harmónicos en señales polifónicas. Los cuales fueron evaluados aplicandolos en un programa de transcripción de música para piano implementado y simulado en un entorno Matlab.

En el trabajo de Marolt y col., 2002 destacamos el uso de sistemas conexionistas para la detección de inicio en señales musicales basandose en la combinacion de un banco de filtros auditivos , un red neuronal pulsante y un perceptron multicapa. Este uso de sistemas conexionistas también es empleado para sistemas de transcripción de musica de piano polifónico, mostrando ciertas ventajas sobre otros métodos debido a su estructura y presentándose como una alternativa viable a los algoritmos ya existentes

El uso de redes de neuronas artificiales también están presentes en trabajos como Solanki y Pandey, 2019, que aborda la identificación de los instrumentos que forman part de piezas polifónicas. Utiliza una red de neuronas convolucional de 8 capas y se apoya en los espectrogramas MEL para mapear datos del audio.

Para finalizar, ya en un campo algo más alejado del musical, podemos destacar el trabajo elaborado por Baevski y col., 2021, el cual profundiza en el campo de reconocimiento del habla mediante inteligencia artificial. A diferencia de otros sistemas de reconocimiento, este trabajo no usa datos etiquetados que limiten el reconocimiento a un grupo reducido de idiomas. Esta técnica necesita menos requerimientos, aprovechando representaciones auto supervisadas del habla para segmentar el audio y aprender a mapear desde estas representaciones a fonemas via *Adversarial Training*.

A pesar de que algunos trabajos difieren ligeramente de la tematica de nuestro sistema, estos concluyen en técnicas favorables y aplicables a nuestro problema. Podemos concluir que sus aproximaciones y conclusiones son útiles como base para nuestro trabajo.

5. Desarrollo

Para el desarrollo de este sistema utilizaremos un método basado en aproximaciones, es decir, comenzaremos acotando el problema e iremos aumentando la complejidad a medida que obtenemos resultados satisfatorios.

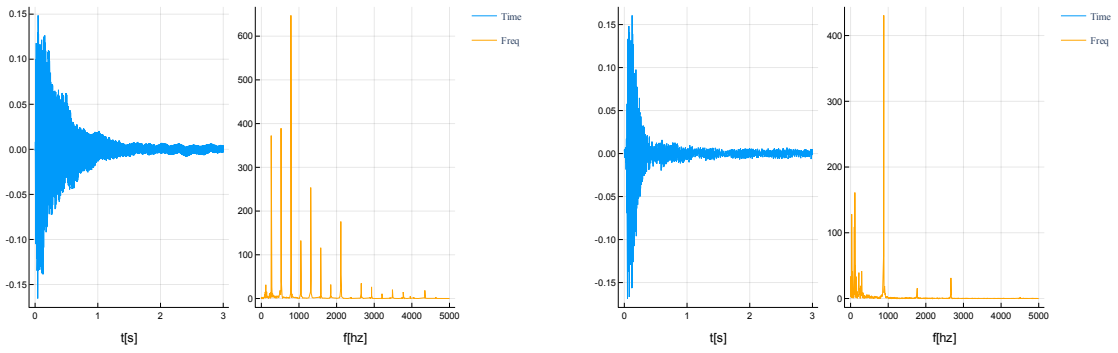
Además, para mejorar la calidad de los experimentos y no depender de la selección aleatoria del conjunto de test y de entrenamiento, utilizaremos la técnica de *cross validation* (validación cruzada). Esta técnica consiste en repetir y calcular la media aritmética obtenida de las medidas de la evaluación sobre k *folds* (conjuntos) distintos.

En concreto, se realizará un *cross validation* con 10 *folds*. El valor del F1-Score y de la precisión del modelo será el promedio de dichos valores obtenidos en los *folds*.

5.1. Primera aproximación

5.1.1. Descripción

En esta primera aproximación nos limitaremos a diferenciar únicamente entre dos notas. Escogimos las notas *C4* y *A5* de los cuales usaremos 92 y 56 audios de cada una, respectivamente, en total se utilizarán 148 patrones. En las Figuras 2a y 2b podemos ver a la izquierda la señal en tiempo y, a la derecha, la señal en el dominio de la frecuencia.



(a) C4 (b) A5

Figura 2: Notas en tiempo y frecuencia

Dado que cada muestra tiene una duración diferente, decidimos quedarnos solo con los 3 primeros segundos de cada audio (que es donde está la información más relevante de la nota) y, para la frecuencia, acotaremos cada señal entre 0 y 5000 Hz. Ya que la frecuencia máxima que se alcanza en el piano clásico es de 4186 Hz (*C8*) y la más baja, en nuestra base de datos, es de 32.7Hz (*C1*), escogimos un rango de frecuncias ampliado debido a que es posible que exista información que nos ayude a identificar la nota.

Además, como las notas de un mismo instrumento se diferencian principalmente por la frecuencia (una nota más aguda tiene una mayor frecuencia y una más grave, una menor), calcularemos de las señales en el dominio del tiempo su relación en el dominio de la frecuencia utilizando la Transformada de Fourier y, posteriormente, extraeremos las siguientes características:

- **Energia media de la señal**: ya que las señales con más frecuencia (más agudas) tienen más energía, esta característica nos podría ayudar a diferenciar entre notas tocadas con la misma intensidad.
- **Media, desviación tipica, y valor máximo de la intensidad en el dominio de la frecuencia, en intervalos no uniformes**: la frecuencia de cada nota aumenta de forma exponencial siguiendo la fórmula:

$$f_{i+1} = f_i \cdot (\sqrt[12]{2}), f_0 = 27{,}5Hz \tag{5}$$

  Por ello decidimos dividir el espectro en **10 intervalos** de longitud variable siguiendo dicha distribución. Podríamos entonces obtener un intervalo donde la media y la desviación típica de la frecuencia fueran más elevados que el resto, ayudándonos a identificar la nota.

  Los intervalos que usaremos son los siguientes:

  (0.0, 380.3), (380.3, 783.21), (783.21, 1210.08), (1210.08, 1662.33), (1662.33, 2141.47), (2141.47, 2649.11), (2649.11, 3186.93), (3186.93, 3756.73), (3756.73, 4360.42) y (4360.42, 5000.0)

- **Zero-crossing/s**: esta característica determina las veces que la señal, en el dominio del tiempo, toma el valor 0 por segundo. Como dicha cracterística nos proporcionaría valores similares a la frecuencia media de la señal, podría contribuír a su correcta clasificación.

En lo referente al preprocesado de los datos, hemos decidido optar por una **normalización de media cero**, debido a que los datos no están acotados en un intervalo y pueden tomar cualquier valor, ya sea en tiempo o en frecuencia. Aunque las frecuencias de las notas sí que están acotadas en un intervalo, en la señal puede haber ruido que sobrepase los valores típicos, por lo que una normalización entre mínimo y máximo no sería lo adecuado.

Así, obtuvimos para cada característica, la media y la desviación típica. Se muestran dichos datos en la Tabla 1.

Tabla 1: Parámetros de normalización para cada característica

Característica	Media	Desviación Típica
E	0,004567	0,003572
zero-crossing	1797,2139	1352,7441
m_1	16,8211	11,6166
m_2	9,4079	5,8349
m_3	14,6074	11,5159
m_4	3,8720	3,5469
m_5	2,3520	2,1539
m_6	0,4869	0,6487
m_7	0,7153	1,09730
m_8	0,4007	0,5004
m_9	0,2236	0,3412
m_{10}	0,2323	0,5112
std_1	54,2823	46,6795
std_2	31,5828	27,1859
std_3	57,8961	38,4394
std_4	11,1088	12,6960
std_5	7,004131	6,7326
std_6	0,5213	0,7275
std_7	2,7318	4,6750
std_8	0,9234	1,2807
std_9	0,4775	0,8984
std_{10}	0,5695	1,2890
max_1	832,8222	779,3061
max_2	567,7255	505,1785
max_3	901,9232	548,2265
max_4	192,8098	237,4715
max_5	98,9383	103,5578
max_6	4,3920	6,07776
max_7	40,5960	65,9938
max_8	11,6721	15,5244
max_9	5,7904	10,4402
max_{10}	5,6131	11,5651

Donde:

- E indica la **energía de la señal**.
- m_i indica la **media de la intensidad** en el intervalo i en frecuencia.
- std_i indica la **desviación típica de la intensidad** en el intervalo i en frecuencia.
- max_i indica el **máximo de la intensidad** en el intervalo i en frecuencia.

Como el entrenamiento de una RNA es un proceso no determinista, en cada *fold* tendremos que entrenar la RNA varias veces, y promediar los resultados. Para estos experimentos, se ha repetido 30 veces el entrenamiento de la RNA en cada *fold*, devolviendo el resultado promedio de esas 30 ejecuciones.

Para poder evaluar el modelo con mejor rendimiento, como ya se ha dicho antes, utilizaremos como medida el promedio del **F1-Score** en los *folds*. Si hay empate en dicha media, se elegirá el modelo con menor desviación típica de la métrica.

5.1.2. Resultados

Para permitir la reproducibilidad de los experimentos, se utilizó *100* como semilla de generación de números aleatorios.

5.1.2.1 RNA

Para el entrenamiento de las RNA, se han utilizado los siguientes parámetros en común:

- Ratio de aprendizaje: 0,01
- Ratio de patrones para el conjunto de validación: 0,2.
- Máximo número de ciclos (*epochs*) de entrenamiento: 1500.
- Máximo número de ciclos (*epochs*) de entrenamiento sin mejorar el error de validación: 5.
- Algoritmo de optimización: ADAM.
- Función de *loss*: *Binary Cross Entropy*.

El hiperparámetro que hemos variado ha sido la arquitectura de neuronas a utilizar. Utilizamos 8 arquitecturas distintas, donde la arquitectura $[i]$ denota una RNA con una única capa oculta con i neuronas y la arquitectura $[i, j]$ denota una neurona con dos capas ocultas; con i neuronas en la primera capa oculta y j neuronas en la segunda capa oculta.

Se muestran los resultados de la experimentación en la Tabla 2.

5.1.2.2 SVM

Para las máquinas de soporte vectorial hemos utilizado 8 configuraciones distintas de los hiperparámetros de *kernel*, grado del *kernel* (sólo se utiliza en *kernel* polinomial), *gamma* y *C*.

Se muestran los resultados de la experimentación en la Tabla 3.

Tabla 2: Resultados RNA

Arquitectura	F1-Score	Precisión
[2]	$1{,}0 \pm 0{,}0$	$1{,}0 \pm 0{,}0$
[4]	$1{,}0 \pm 0{,}0$	$1{,}0 \pm 0{,}0$
[8]	$1{,}0 \pm 0{,}0$	$1{,}0 \pm 0{,}0$
[2, 4]	$0{,}9119 \pm 0{,}12531$	$0{,}96487 \pm 0{,}05054$
[2, 8]	$0{,}90667 \pm 0{,}14724$	$0{,}96524 \pm 0{,}05412$
[4, 2]	$1{,}0 \pm 0{,}0$	$1{,}0 \pm 0{,}0$
[4, 4]	$0{,}99 \pm 0{,}0225$	$0{,}99643 \pm 0{,}00804$
[4, 8]	$0{,}9621 \pm 0{,}0974$	$0{,}97934 \pm 0{,}05699$

Tabla 3: Resultados SVM

(kernel, grado, gamma, C)	F1-Score	Precisión
(poly, 3, 1, 1)	$1{,}0 \pm 0{,}0$	$1{,}0 \pm 0{,}0$
(rbf, 3, 1, 1)	$0{,}94766 \pm 0{,}05627$	$0{,}94744 \pm 0{,}06608$
(sigmoid, 3, 1, 1)	$0{,}73509 \pm 0{,}10847$	$0{,}76196 \pm 0{,}10288$
(poly, 5, 10, 0,1)	$0{,}98565 \pm 0{,}03158$	$0{,}98619 \pm 0{,}02913$
(rbf, 3, 10, 0,1)	$0{,}76689 \pm 0{,}01542$	$0{,}62214 \pm 0{,}02026$
(sigmoid, 3, 10, 0,1)	$0{,}69648 \pm 0{,}16555$	$0{,}7772 \pm 0{,}06479$
(rbf, 3, 0,01, 100)	$1{,}0 \pm 0{,}0$	$1{,}0 \pm 0{,}0$
(rbf, 3, 100, 0,01)	$0{,}76689 \pm 0{,}01542$	$0{,}62214 \pm 0{,}02026$

5.1.2.3 Árboles de decisión

Para los árboles de decisión hemos utilizado 6 configuraciones distintas del hiperparámetro de profundidad máxima.

Se muestran los resultados de la experimentación en la Tabla 4.

Tabla 4: Resultados Árbol de decisión

Altura máxima	F1-Score	Precisión
2	$1{,}0 \pm 0{,}0$	$1{,}0 \pm 0{,}0$
4	$1{,}0 \pm 0{,}0$	$1{,}0 \pm 0{,}0$
8	$1{,}0 \pm 0{,}0$	$1{,}0 \pm 0{,}0$
16	$1{,}0 \pm 0{,}0$	$1{,}0 \pm 0{,}0$
32	$1{,}0 \pm 0{,}0$	$1{,}0 \pm 0{,}0$
64	$1{,}0 \pm 0{,}0$	$1{,}0 \pm 0{,}0$

5.1.2.4 kNN

Para el modelo kNN hemos utilizado 6 configuraciones distintas del hiperparámetro de k (número de vecinos más cercanos).

Se muestran los resultados de la experimentación en la Tabla 5.

Tabla 5: Resultados kNN

K	F1-Score	Precisión
2	$1{,}0 \pm 0{,}0$	$1{,}0 \pm 0{,}0$
3	$1{,}0 \pm 0{,}0$	$1{,}0 \pm 0{,}0$
5	$1{,}0 \pm 0{,}0$	$1{,}0 \pm 0{,}0$
8	$1{,}0 \pm 0{,}0$	$1{,}0 \pm 0{,}0$
13	$1{,}0 \pm 0{,}0$	$1{,}0 \pm 0{,}0$
21	$1{,}0 \pm 0{,}0$	$1{,}0 \pm 0{,}0$

5.1.3. Discusión

Los resultados han sido muy buenos, ya que en la mayoría de casos se alcanza una **precisión** y un **F1-Score** del 100 % o muy alto.

En general, los sistemas se equivocan en muy pocos patrones. Esto se puede deber a que el problema elegido en esta aproximación era excesivamente sencillo para el elevado número de características escogidas, permitiéndonos separar perfectamentes los patrones en las dos clases.

En lo referente a las características, por una parte estas han sido correctamente elegidas, ya que alcanzamos una precisión muy alta. Pero por otra parte, el hecho de que algunos sistemas concretos no funcionen tan bien (por ejemplo, algunos SVM), se podría deber al elevado número de características, ya que algunas de ellas pueden ser innecesarias o hasta contraproducentes en este sencillo problema. Una buena estrategia sería el intentar reducir el número de características en futuras aproximaciones, pero hay que tener cuidado, ya que si aumentamos el número de clases, podría darse el caso de haber eliminado una característica necesaria para una correcta clasificación.

En concreto, las características que creemos que son las que producen mejores resultados son las que se refieren a los intervalos en frecuencia de *(0.0, 380.3)* y *(783.21, 1210.08)*, debido a que es en ellos donde se sitúan las frecuencia de las notas *C4* y *A5*, respectivamente. Las características que se corresponden a dichos intervalos son m_1, std_1, max_1 para el primer intervalo y m_3, std_3, max_3 para el segundo intervalo.

Otra característica que podría funcionar sería la frecuencia en la que se alcanza el pico absoluto en intensidad de la señal, o la media de las K frecuencias con mayor intensidad, sin tener en cuenta la división en intervalos. También se podría aumentar el número de intervalos en los que obtener las características, para poder así separar mejor las notas con frecuencias más cercanas.

Todos los algoritmos tienen al menos una configuración con un **F1-Score** de 1,0±0,0, por lo que es complicado encontrar uno que sea el mejor.

Podemos destacar la RNA con una arquitectura de una única capa oculta con 2 neuronas, ya que es un modelo muy sencillo y podemos esperar que, cuanto más sencillo sea un modelo, mayor será también su capacidad de generalización.

En la Figura 3 podemos ver un gráfico del entrenamiento de dicha RNA, donde observamos el error cometido en los conjuntos de test, validación y entrenamiento para cada ciclo de entrenamiento. Este se ha realizado utilizando todos los patrones, separando un 20 % para el conjunto de test. Además, se puede ver que se ha producido una parada temprana, ya que el sistema no ha realizado el máximo número de ciclos de entrenamiento (1500 ciclos) y se puede distinguir también como la curva de la gráfica desciende muy rápido. Esto se puede deber a que tenemos un ratio de aprendizaje alto para este problema concreto.

En la Tabla 6 podemos ver una matriz de confusión para la RNA entrenada anteriormente. Las predicciones se hicieron sobre un conjunto de test que supone un 20 % de todos los patrones utilizados. El 80 % de los patrones restantes se utilizó para el entrenamiento de la RNA, ya que el conjunto de entrenamiento no se debería utilizar para realizar predicciones.

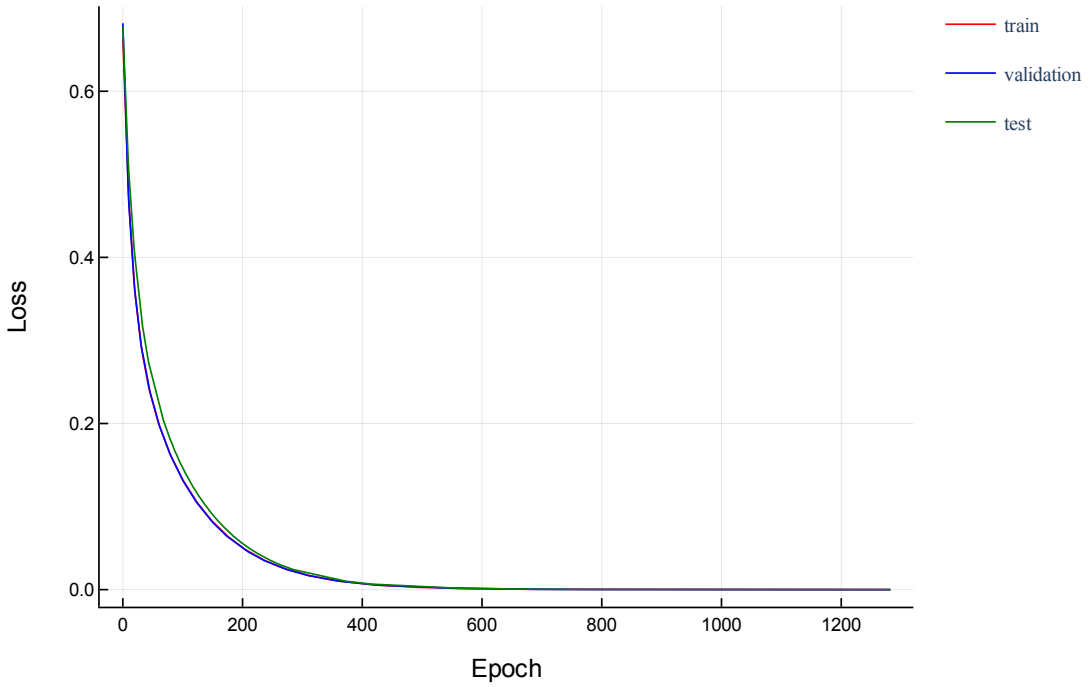


Figura 3: Entrenamiento RNA

		Predicc. A5	Predicc. C4
Real	A5	15	0
	C4	0	15

Tabla 6: Matriz de confusión RNA

Pero aunque se alcance ya un buen resultado con un modelo sencillo como puede ser la RNA anterior, caben destacar los árboles de decisión, que son modelos que pueden ofrecer explicabilidad sobre sus predicciones, además de presentar un entrenamiento determinista, por lo que si tenemos que elegir el mejor modelo, nos decantaríamos por este (en concreto, el de altura máxima de 2, ya que es el más sencillo), ya que presenta también un 100 % de **F1-Score**.

Aunque si deseamos un modelo con una base matemática más fuerte, en vez de un modelo con una base estadística como puede ser un árbol de regresión, utilizar una máquina de soporte vectorial con un *kernel* polinomial de grado 3, y ambos valores de *gamma* y C de 1, también sería una buena opción.

Los algoritmos que han funcionado peor han sido ciertas configuraciones de los SVM, en concreto las que utilizan un *kernel* sigmoidal. Entonces, utilizar un *kernel* sigmoidal no parece ofrecer buenos resultados para estos patrones.

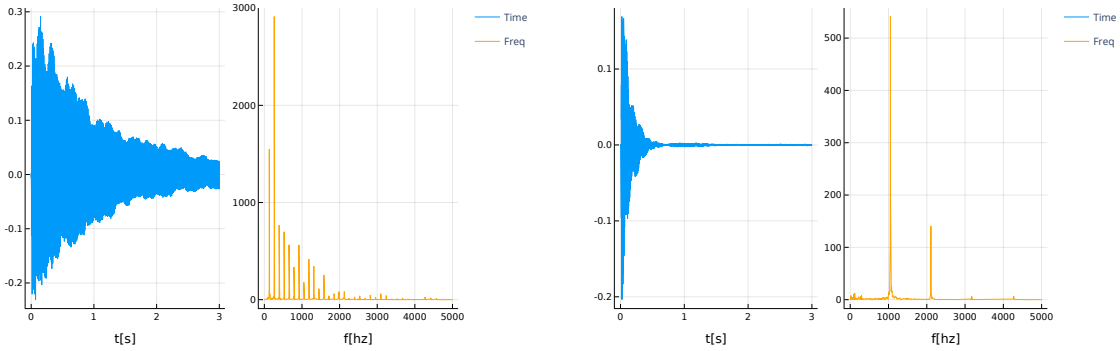
5.2. Segunda aproximación

5.2.1. Descripción

Como hemos obtenido resultados muy buenos en la aproximación anterior, aumentaremos la complejidad del problema, es decir, añadiremos más notas a ser clasificadas. Se clasificarán entonces 15 notas y se dispondrá de 1090 patrones para ser utilizados por el sistema.

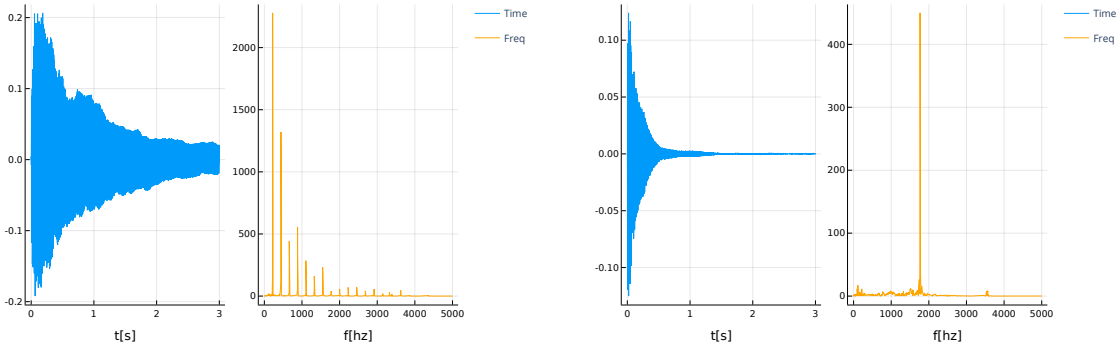
Las notas que vamos a clasificar son: *C1*, *C2*, *C3*, *C4*, *C5*, *C6*, *C7*, *C8*, *A1*, *A2*, *A3*, *A4*, *A5*, *A6* y *A7*.

Se puede ver algunos ejemplos de los *samples* que vamos a utilizar, con su señal en el dominio del tiempo y en el dominio de la frecuencia, en las Figuras 4a, 4b, 5a y 5b.



(a) C3 (b) A6

Figura 4



(a) A3 (b) A6

Figura 5

Como hemos aumentado mucho el número de clases a clasificar, decidimos aumentar también el número de características para lograr una correcta diferenciación entre las clases que sean más similares.

En concreto, añadiremos las siguientes características:

- **Frecuencia en la que se alcanza la máxima intensidad** en cada uno de los 10 intervalos que ya se especificaron en la aproximación anterior.

No se producen cambios sobre las características que fueron incorporadas en la aproximación anterior.

Para el preprocesado de los datos, utilizaremos, como se mencionó en la aproximación anterior, una normalización de media cero. Así, obtuvimos para cada característica, la media y la desviación típica. Se muestran dichos datos en las Tablas 7 y 8.

Tabla 7: Parámetros de normalización (I)

Característica	Media	Desviación Típica
E	0,005088	0,008891
zero-crossing	3743,606	5035,344
m_1	15,5785	18,3973
m_2	11,1477	9,1793
m_3	9,65644	9,2952
m_4	4,09436	4,2257
m_5	3,66450	4,6995
m_6	1,03469	1,38888
m_7	0,715403	1,00623
m_8	0,683842	1,3073
m_9	0,389782	0,83504
m_{10}	0,447576	1,40784
std_1	49,611	74,6580
std_2	39,125	41,2061
std_3	29,0237	34,236
std_4	8,5606	11,0015
std_5	9,4632	12,9204
std_6	1,56267	2,65298
std_7	1,63015	2,81702
std_8	1,25008	2,2953
std_9	0,53424	0,980414
std_{10}	0,73478	2,21081
max_1	944,372	1596,25
max_2	748,072	844,440
max_3	442,8393	524,4263
max_4	130,525	192,106
max_5	129,601	179,401
max_6	19,1590	36,384
max_7	22,5892	40,294
max_8	13,2673	21,8992
max_9	5,34739	10,3271
max_{10}	5,5207	13,5945

Tabla 8: Parámetros de normalización (II)

Característica	Media	Desviación Típica
max-freq$_1$	174,899	87,0582
max-freq$_2$	521,6259	125,1140
max-freq$_3$	961,683	100,9045
max-freq$_4$	1394,950	126,0789
max-freq$_5$	1897,956	174,8465
max-freq$_6$	2353,771	191,6192
max-freq$_7$	2791,137	166,8256
max-freq$_8$	3426,202	169,1325
max-freq$_9$	4093,241	226,9745
max-freq$_{10}$	4583,387	193,168

Donde:

- E indica la **energía de la señal**.
- m_i indica la **media de la intensidad** en el intervalo i en frecuencia.
- std_i indica la **desviación típica de la intensidad** en el intervalo i en frecuencia.
- max_i indica el **máximo de la intensidad** en el intervalo i en frecuencia.
- *max-freq*$_i$ indica la **frecuencia donde se alcanza el máximo de intensidad** en el intervalo i en frecuencia.

Para estos experimentos se ha repetido 20 veces el entrenamiento de la RNA en cada *fold*, devolviendo el resultado promedio de esas 20 ejecuciones.

Para poder evaluar el modelo con mejor rendimiento, como ya se ha dicho antes, utilizaremos como medida el promedio del **F1-Score** en los *folds*. Si hay empate en dicha media, se elegirá el modelo con menor desviación típica de la métrica.

5.2.2. Resultados

Para permitir la reproducibilidad de los experimentos, se utilizó *100* como semilla de generación de números aleatorios.

5.2.2.1 RNA

Para el entrenamiento de las RNA, se han utilizado los siguientes parámetros en común:

- Ratio de aprendizaje: 0,01
- Ratio de patrones para el conjunto de validación: 0,2.
- Máximo número de ciclos (*epochs*) de entrenamiento: 1500.
- Máximo número de ciclos (*epochs*) de entrenamiento sin mejorar el error de validación: 5.
- Algoritmo de optimización: ADAM.
- Función de *loss*: *Cross Entropy*.

El hiperparámetro que hemos variado ha sido la arquitectura de neuronas a utilizar. Utilizamos 8 arquitecturas distintas, donde la arquitectura $[i]$ denota una RNA con una única capa oculta con i neuronas y la arquitectura $[i, j]$ denota una neurona con dos capas ocultas; con i neuronas en la primera capa oculta y j neuronas en la segunda capa oculta.

Se muestran los resultados de la experimentación en la Tabla 9

Tabla 9: Resultados RNA

Arquitectura	F1-Score	Precisión
[8]	$0{,}92977 \pm 0{,}01923$	$0{,}95523 \pm 0{,}0131$
[16]	$0{,}96092 \pm 0{,}0163$	$0{,}97591 \pm 0{,}0112$
[32]	$0{,}96305 \pm 0{,}01805$	$0{,}97745 \pm 0{,}01137$
[8, 8]	$0{,}87808 \pm 0{,}01593$	$0{,}92368 \pm 0{,}00628$
[16, 4]	$0{,}76012 \pm 0{,}03119$	$0{,}8518 \pm 0{,}0233$
[16, 8]	$0{,}92264 \pm 0{,}02515$	$0{,}95168 \pm 0{,}01708$
[32, 16]	$0{,}9506 \pm 0{,}01989$	$0{,}97044 \pm 0{,}01033$
[32, 32]	$0{,}95122 \pm 0{,}02256$	$0{,}97004 \pm 0{,}013$

5.2.2.2 SVM

Para las máquinas de soporte vectorial hemos utilizado 8 configuraciones distintas de los hiperparámetros de *kernel*, grado del *kernel* (sólo se utiliza en *kernel* polinomial), *gamma* y *C*.

Se muestran los resultados de la experimentación en la Tabla 10

Tabla 10: Resultados SVM

(kernel, grado, gamma, C)	F1-Score	Precisión
(poly, 3, 1, 1)	0,97261 ± 0,02358	0,98158 ± 0,01438
(rbf, 3, 1, 1)	0,67559 ± 0,04297	0,69947 ± 0,05196
(sigmoid, 3, 1, 1)	0,28516 ± 0,03747	0,3575 ± 0,03929
(poly, 3, 10, 0,1)	0,97619 ± 0,02456	0,98415 ± 0,01345
(rbf, 3, 10, 0,1)	0,01607 ± 0,00059	0,13703 ± 0,00568
(sigmoid, 3, 10, 0,1)	0,23652 ± 0,03974	0,33481 ± 0,04109
(rbf, 3, 0,01, 100)	0,97415 ± 0,02285	0,98444 ± 0,0122
(poly, 3, 100, 0,001)	0,9678 ± 0,03578	0,97987 ± 0,01547

5.2.2.3 Árboles de decisión

Para los árboles de decisión hemos utilizado 6 configuraciones distintas del hiperparámetro de profundidad máxima.

Se muestran los resultados de la experimentación en la Tabla 11

Tabla 11: Resultados Árbol de decisión

Altura máxima	F1-Score	Precisión
4	0,44767 ± 0,00668	0,66703 ± 0,01287
8	0,86809 ± 0,02258	0,92653 ± 0,01909
16	0,96505 ± 0,02104	0,9725 ± 0,0143
32	0,96786 ± 0,01849	0,97302 ± 0,01609
64	0,97438 ± 0,01244	0,97779 ± 0,0102
128	0,97415 ± 0,0161	0,9779 ± 0,01186
256	0,9672 ± 0,02388	0,97452 ± 0,0172

5.2.2.4 kNN

Para el modelo kNN hemos utilizado 6 configuraciones distintas del hiperparámetro de k (número de vecinos más cercanos).

Se muestran los resultados de la experimentación en la tabla 12

Tabla 12: Resultados kNN

K	F1-Score	Precisión
2	$0{,}95086 \pm 0{,}01526$	$0{,}97073 \pm 0{,}0088$
3	$0{,}9323 \pm 0{,}03125$	$0{,}95942 \pm 0{,}0156$
5	$0{,}90323 \pm 0{,}03494$	$0{,}94503 \pm 0{,}01712$
8	$0{,}86641 \pm 0{,}03906$	$0{,}93135 \pm 0{,}0199$
13	$0{,}84065 \pm 0{,}02462$	$0{,}90985 \pm 0{,}01911$
21	$0{,}80344 \pm 0{,}03804$	$0{,}88737 \pm 0{,}02667$

5.2.3. Discusión

En general los resultados son muy buenos, ya que hay varios modelos con un alto F1-Score. El modelo que mejor funciona es un SVM con función de *kernel* polinomial de grado 3, con un *gamma* y C de 10 y 0,1, respectivamente.

Este modelo tiene un F1-Score de $0{,}97619 \pm 0{,}02456$, con una precisión de $0{,}98415 \pm 0{,}01345$ en los *folds*. De este resultado podemos concluír que nuestro sistema funciona muy bien, ya que clasificar 15 notas es un problema relativamente complejo.

Cabe destacar modelos como la RNA con una única capa oculta de 32 neuronas; el SVM con *kernel* de base radial, con los parámetros de *gamma* 0,01 y de C 100; un árbol de decisión con altura máxima de 64; y un modelo kNN con un k de 2. Estos modelos también presentan muy buenos resultados, aunque no los mejores.

El modelo con peor rendimiento es un SVM con kernel de base radial con *gamma* y C de 10 y 0,1, respectivamente. Esta configuración del modelo un *F1-Score* de tan solo $0{,}01607 \pm 0{,}00059$, desconociendo realmente a qué se debe realmente este tan bajo resultado.

En cuanto a las características, creemos que fue un acierto añadir la frecuencia a la que se encuentra el pico de intensidad en cada intervalo, ya que es una característica muy representativa de la señal y que sirve para diferenciar mejor las notas. Sin estas nuevas características (y habiendo hecho las pruebas), los resultados no serían tan buenos.

El problema que podemos observar en la matriz de confusión (Tabla 13) para el mejor sistema es que el sistema se equivoca en las notas más cercanas, como pueden ser *A1*, *A2*, *C1* y *C2*; notas que tienen una frecuencia muy similar y se encuentran en el primer intervalo, *(0.0, 380.3)*.

Para poder diferenciar mejor entre estas notas (y entre futuras notas que puedan ser similares entre ellas), sería una buena opción que se aumentase el número de intervalos a dividir las frecuencias, para así poder diferenciar de mejor manera dichas notas.

		Predicc.														
		C1	C2	C3	C4	C5	C6	C7	C8	A1	A2	A3	A4	A5	A6	A7
Real	C1	3	0	0	0	0	0	0	0	1	0	0	0	0	0	0
	C2	1	6	1	0	0	0	0	0	0	0	0	0	0	0	0
	C3	0	0	9	0	0	0	0	0	0	0	0	0	0	0	0
	C4	0	0	1	18	0	0	0	0	0	0	0	0	0	0	0
	C5	0	0	0	0	25	0	0	0	0	0	0	0	0	0	0
	C6	0	0	0	0	0	10	0	0	0	0	0	0	0	0	0
	C7	0	0	0	0	0	0	23	0	0	0	0	0	0	0	0
	C8	0	0	0	0	0	0	0	19	0	0	0	0	0	0	1
	A1	0	0	0	0	0	0	0	0	16	0	0	0	0	0	0
	A2	0	0	0	0	0	0	0	0	1	15	0	0	0	0	0
	A3	0	0	0	3	0	0	0	0	0	0	12	0	0	0	0
	A4	0	0	0	0	0	0	0	0	0	0	0	13	0	0	0
	A5	0	0	0	0	0	0	0	0	0	0	0	0	15	0	0
	A6	0	0	0	0	0	0	0	0	0	0	0	0	0	15	1
	A7	0	0	0	0	0	0	0	0	0	0	0	0	0	0	9

Tabla 13: Matriz de confusión del mejor modelo

5.3. Tercera aproximación

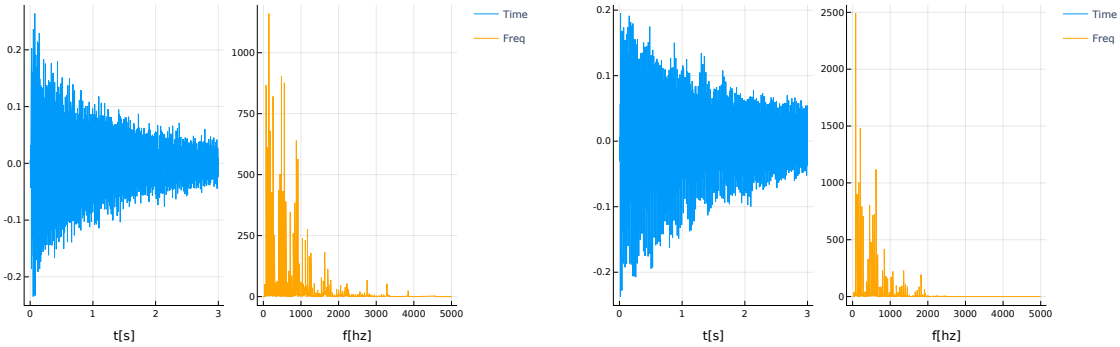
5.3.1. Descripción

Como hemos obtenido resultados muy buenos en la aproximación anterior, seguiremos aumentando la complejidad del problema, es decir, añadiremos más notas a ser clasificadas. Serán un total de 50 notas a clasificar, y se dispondrá de 3194 patrones para ser utilizados por los modelos.

Las notas que vamos a clasificar son: *C1*, *C2*, *C3*, *C4*, *C5*, *C6*, *C7*, *C8*, *A1*, *A2*, *A3*, *A4*, *A5*, *A6*, *A7*, *B1*, *B2*, *B3*, *B4*, *B5*, *B6*, *B7*, *D1*, *D2*, *D3*, *D4*, *D5*, *D6*, *D7*, *E1*, *E2*, *E3*, *E4*, *E5*, *E6*, *E7*, *F1*, *F2*, *F3*, *F4*, *F5*, *F6*, *F7*, *G1*, *G2*, *G3*, *G4*, *G5*, *G6* y *G7*.

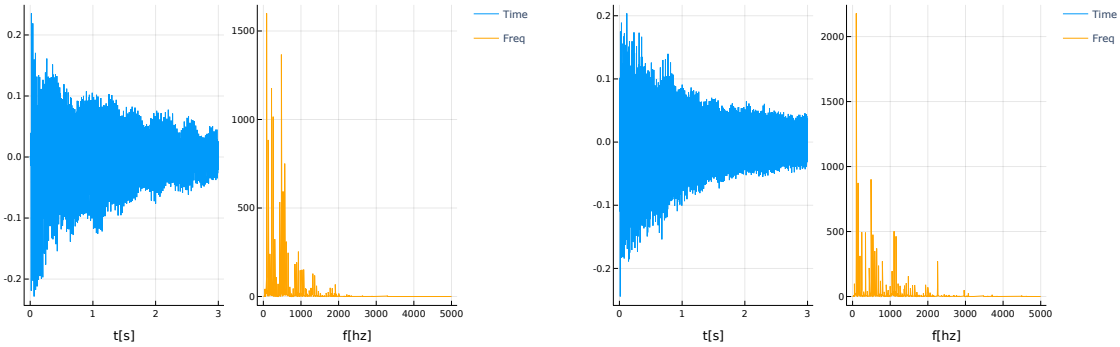
Se pueden ver algunos ejemplos de estas notas, en el dominio del tiempo y de la frecuencia, en las Figuras 6a, 6b, 7a y 7b.

Como en esta aproximación aumentamos mucho el número de clases a clasificar (50 en total), nos parece buena opción aumentar también las características utilizadas, pero desde un enfoque diferente a la anterior aproximación. Lo que haremos será aumentar el número de intervalos a dividir el dominio de la frecuencia; de 10 intervalos que teníamos anteriormente, a 30 intervalos. De esta manera, esperamos que se puede clasificar mejor notas que tengan frecuencias muy cercanas, además de repartir mejor el número de notas que coinciden en cada intervalo.



(a) D1 (b) E1

Figura 6



(a) F1 (b) G1

Figura 7

En concreto, los intervalos de frecuencias que utilizaremos serán los siguientes:
(0.0, 63.845),(63.845, 131.486), (131.486, 203.149),(203.149, 279.074), (279.074, 359.513),(359.513, 444.735), (444.735, 535.025),(535.025, 630.684), (630.684, 732.032), (732.032, 839.405), (839.405, 953.163), (953.163, 1073.686), (1073.686, 1201.376), (1201.376, 1336.658), (1336.658, 1479.984), (1479.984, 1631.833), (1631.833, 1792.712), (1792.712, 1963.157), (1963.157, 2143.737), (2143.737, 2335.055), (2335.055, 2537.749), (2537.749, 2752.496), (2752.496, 2980.013), (2980.013, 3221.059), (3221.059, 3476.437), (3476.437, 3747.002), (3747.002, 4033.655), (4033.655, 4337.353), (4337.353, 4659.11) y *(4659.11, 5000.0)*

Para el preprocesado de los datos, utilizaremos, como se mencionó en aproximaciones anteriores, una normalización de media cero. Así, obtuvimos para cada característica, la media y la desviación típica. Se muestran dichos datos en las Tablas 14, 15 16 y 17.

Tabla 14: Parámetros de normalización (I)

Característica	Media	Desviación Típica
E	0,005412	0,009019
zero-crossing	3397,5913	4540,8813
m_1	2,5406	5,2477
m_2	20,31	34,20
m_3	18,736	33,777
m_4	19,506	27,648
m_5	17,283	26,778
m_6	14,071	21,961
m_7	15,653	21,659
m_8	7,2894	11,942
m_9	10,673	22,342
m_{10}	8,886	17,860
m_{11}	8,768	14,431
m_{12}	9,399	13,1116
m_{13}	7,141	16,054
m_{14}	5,671	9,146
m_{15}	4,4691	7,688
m_{16}	4,622	7,50
m_{17}	2,8246	5,453
m_{18}	2,1192	3,5254
m_{19}	3,2075	5,558
m_{20}	1,3101	2,2760
m_{21}	1,5997	4,51
m_{22}	1,4330	3,7737
m_{23}	0,9662	2,9858
m_{24}	0,79353	1,8264
m_{25}	0,48343	0,8716
m_{26}	0,6009	1,4477
m_{27}	0,3706	0,93128
m_{28}	0,41856	1,3392
m_{29}	0,38214	1,6580
m_{30}	0,1347	0,2588

Tabla 15: Parámetros de normalización (II)

Característica	Media	Desviación Típica
std_1	3,777	9,5573
std_2	43,746	101,179
std_3	44,765	106,52
std_4	44,928	86,98
std_5	36,79	74,466
std_6	34,05	69,12
std_7	35,642	58,005
std_8	14,239	31,195
std_9	20,148	46,739
std_{10}	18,305	41,97
std_{11}	15,096	33,64
std_{12}	17,944	31,481
std_{13}	14,973	42,54
std_{14}	10,101	21,576
std_{15}	7,4685	17,61
std_{16}	7,9542	17,202
std_{17}	5,18	11,849
std_{18}	2,358	5,2823
std_{19}	6,350	12,173
std_{20}	1,757	4,281
std_{21}	2,7778	8,633
std_{22}	2,4016	6,636
std_{23}	1,6487	5,640
std_{24}	1,3393	3,0001
std_{25}	0,606	1,4506
std_{26}	0,9203	2,2596
std_{27}	0,4674	1,0911
std_{28}	0,64302	2,1712
std_{29}	0,45122	1,628
std_{30}	0,1932	0,4538

Tabla 16: Parámetros de normalización (III)

Característica	Media	Desviación Típica
$\max_1$	33,78	108,59
$\max_2$	448,6	1190,1
$\max_3$	473,44	1207,2
$\max_4$	464,8	994,3
$\max_5$	362,47	770,89
$\max_6$	343,7	698,5
$\max_7$	339,79	571,13
$\max_8$	146,53	331,86
$\max_9$	187,07	410,4
$\max_{10}$	172,76	362,9
$\max_{11}$	142,42	321,43
$\max_{12}$	170,99	295,21
$\max_{13}$	140,03	364,36
$\max_{14}$	98,230	208,93
$\max_{15}$	68,81	154,13
$\max_{16}$	74,12	160,24
$\max_{17}$	51,78	117,74
$\max_{18}$	20,73	55,032
$\max_{19}$	55,946	99,87
$\max_{20}$	16,036	44,92
$\max_{21}$	23,404	60,15
$\max_{22}$	19,944	46,585
$\max_{23}$	14,242	44,026
$\max_{24}$	11,504	23,90
$\max_{25}$	5,2844	13,773
$\max_{26}$	7,49	17,74
$\max_{27}$	3,6148	8,03
$\max_{28}$	4,8835	14,053
$\max_{29}$	3,2500	9,766
$\max_{30}$	1,6198	3,8292

Tabla 17: Parámetros de normalización (IV)

Característica	Media	Desviación Típica
max-freq$_1$	35,3772	14,7797
max-freq$_2$	109,410	14,3100
max-freq$_3$	157,088	22,2041
max-freq$_4$	237,345	25,3701
max-freq$_5$	310,146	24,3124
max-freq$_6$	403,75	27,4108
max-freq$_7$	489,34	23,0169
max-freq$_8$	584,60	31,9395
max-freq$_9$	672,226	27,8011
max-freq$_{10}$	781,45	30,985
max-freq$_{11}$	904,91	33,5659
max-freq$_{12}$	1004,27	35,984
max-freq$_{13}$	1136,57	49,0341
max-freq$_{14}$	1269,48	45,246
max-freq$_{15}$	1409,39	50,925
max-freq$_{16}$	1535,10	41,0053
max-freq$_{17}$	1704,38	49,309
max-freq$_{18}$	1873,18	58,2878
max-freq$_{19}$	2052,24	60,409
max-freq$_{20}$	2217,66	66,807
max-freq$_{21}$	2417,30	57,0100
max-freq$_{22}$	2638,40	58,7791
max-freq$_{23}$	2857,43	74,333
max-freq$_{24}$	3086,93	80,973
max-freq$_{25}$	3342,32	80,322
max-freq$_{26}$	3589,78	78,697
max-freq$_{27}$	3879,98	98,4089
max-freq$_{28}$	4168,6	103,714
max-freq$_{29}$	4459,12	91,550
max-freq$_{30}$	4837,68	100,403

Donde:

- E indica la **energía de la señal**.
- m_i indica la **media de la intensidad** en el intervalo i en frecuencia.
- std_i indica la **desviación típica de la intensidad** en el intervalo i en frecuencia.
- max_i indica el **máximo de la intensidad** en el intervalo i en frecuencia.
- *max-freq*$_i$ indica la **frecuencia donde se alcanza el máximo de intensidad** en el intervalo i en frecuencia.

Para estos experimentos se ha repetido 20 veces el entrenamiento de la RNA en cada *fold*, devolviendo el resultado promedio de esas 20 ejecuciones.

Para poder evaluar el modelo con mejor rendimiento, como ya se ha dicho anteriormente, utilizaremos como medida el promedio del **F1-Score** en los *folds*. Si hay empate en dicha media, se elegirá el modelo con menor desviación típica de la métrica.

5.3.2. Resultados

Para permitir la reproducibilidad de los experimentos, se utilizó *100* como semilla de generación de números aleatorios.

5.3.2.1 RNA

Para el entrenamiento de las RNA, se han utilizado los siguientes parámetros en común:

- Ratio de aprendizaje: 0,01
- Ratio de patrones para el conjunto de validación: 0,2.
- Máximo número de ciclos (*epochs*) de entrenamiento: 1500.
- Máximo número de ciclos (*epochs*) de entrenamiento sin mejorar el error de validación: 5.
- Algoritmo de optimización: ADAM.
- Función de *loss*: *Cross Entropy*.

El hiperparámetro que hemos variado ha sido la arquitectura de neuronas a utilizar. Utilizamos 8 arquitecturas distintas, donde la arquitectura $[i]$ denota una RNA con una única capa oculta con i neuronas y la arquitectura $[i, j]$ denota una neurona con dos capas ocultas; con i neuronas en la primera capa oculta y j neuronas en la segunda capa oculta.

Se muestran los resultados de la experimentación en la Tabla 18.

Tabla 18: Resultados RNA

Arquitectura	F1-Score	Precisión
[16]	$0{,}95148 \pm 0{,}00551$	$0{,}96105 \pm 0{,}00471$
[32]	$0{,}98759 \pm 0{,}00345$	$0{,}9897 \pm 0{,}00221$
[64]	$0{,}99345 \pm 0{,}00309$	$0{,}99428 \pm 0{,}00162$
[128]	$0{,}99293 \pm 0{,}00258$	$0{,}99424 \pm 0{,}00123$
[256]	$0{,}99269 \pm 0{,}00345$	$0{,}9936 \pm 0{,}00299$
[512]	$0{,}98348 \pm 0{,}0061$	$0{,}98603 \pm 0{,}00423$
[64, 64]	$0{,}98322 \pm 0{,}00373$	$0{,}98554 \pm 0{,}00292$
[64, 128]	$0{,}9818 \pm 0{,}00226$	$0{,}98428 \pm 0{,}00227$

5.3.2.2 SVM

Para las máquinas de soporte vectorial hemos utilizado 8 configuraciones distintas de los hiperparámetros de *kernel*, grado del *kernel* (sólo se utiliza en *kernel* polinomial), *gamma* y *C*.

Se muestran los resultados de la experimentación en la Tabla 19.

Tabla 19: Resultados SVM

(kernel, grado, gamma, C)	F1-Score	Precisión
(poly, 3, 1, 1)	$0{,}99264 \pm 0{,}0053$	$0{,}99382 \pm 0{,}00422$
(rbf, 3, 1, 1)	$0{,}10019 \pm 0{,}01854$	$0{,}11513 \pm 0{,}0161$
(sigmoid, 3, 1, 1)	$0{,}13051 \pm 0{,}02229$	$0{,}20283 \pm 0{,}02546$
(poly, 3, 10, 0,1)	$0{,}99113 \pm 0{,}00724$	$0{,}99314 \pm 0{,}00464$
(rbf, 3, 10, 0,1)	$0{,}00179 \pm 7{,}0e-5$	$0{,}04673 \pm 0{,}00189$
(sigmoid, 3, 10, 0,1)	$0{,}10752 \pm 0{,}01502$	$0{,}18363 \pm 0{,}01528$
(rbf, 3, 0,01, 100)	$0{,}99504 \pm 0{,}00411$	$0{,}99563 \pm 0{,}00257$
(poly, 3, 100, 0,001)	$0{,}99257 \pm 0{,}00408$	$0{,}99379 \pm 0{,}00278$

5.3.2.3 Árboles de decisión

Para los árboles de decisión hemos utilizado 6 configuraciones distintas del hiperparámetro de profundidad máxima.

Se muestran los resultados de la experimentación en la Tabla 20.

5.3.2.4 kNN

Para el modelo kNN hemos utilizado 6 configuraciones distintas del hiperparámetro de k (número de vecinos más cercanos).

Se muestran los resultados de la experimentación en la tabla 21.

Tabla 20: Resultados Árbol de decisión

Altura máxima	F1-Score	Precisión
4	$0{,}09189 \pm 0{,}01043$	$0{,}17819 \pm 0{,}0157$
8	$0{,}3111 \pm 0{,}01697$	$0{,}42453 \pm 0{,}01404$
16	$0{,}71777 \pm 0{,}04105$	$0{,}79892 \pm 0{,}03264$
32	$0{,}94074 \pm 0{,}01447$	$0{,}94584 \pm 0{,}01159$
64	$0{,}94887 \pm 0{,}01332$	$0{,}95193 \pm 0{,}01202$
128	$0{,}95088 \pm 0{,}00891$	$0{,}9555 \pm 0{,}00573$
256	$0{,}95003 \pm 0{,}00775$	$0{,}95269 \pm 0{,}0076$

Tabla 21: Resultados kNN

K	F1-Score	Precisión
2	$0{,}98683 \pm 0{,}00434$	$0{,}98845 \pm 0{,}00303$
3	$0{,}98153 \pm 0{,}00991$	$0{,}98636 \pm 0{,}00479$
5	$0{,}98086 \pm 0{,}00835$	$0{,}98617 \pm 0{,}00599$
8	$0{,}97161 \pm 0{,}01603$	$0{,}98024 \pm 0{,}01049$
13	$0{,}96203 \pm 0{,}01287$	$0{,}97342 \pm 0{,}00864$
21	$0{,}95836 \pm 0{,}00902$	$0{,}96928 \pm 0{,}00517$

5.3.3. Discusión

Los resultados de esta iteración son muy buenos, ya que hemos aumentado en gran medida el número de notas a clasificar y hemos conseguido varios modelos con más de un 99 % de F1-Score y precisión.

En concreto, el modelo que mejor funciona es un SVM con un *kernel* de base radial y con valores de *gamma* y C de 0,01 y 100, respectivamente. Este modelo alcanzó un F1-Score de $0{,}99594 \pm 0{,}00411$ y una precisión de $0{,}99563 \pm 0{,}00257$, unos valores muy altos y que indican que el sistema funciona muy bien.

Caben destacar también modelos que alcanzan un rendimiento parecido, como puede ser un SVM con un *kernel* polinomial de grado 3, con ambos valores de *gamma* y C de 1, además de una RNA con una única capa oculta con 32 neuronas.

En contra, el modelo que peor rendimiento presenta es un SVM con un *kernel* de base radial, con valores de *gamma* y C de 10 y 0,1, respectivamente. Cabe destacar que este modelo es similar al modelo que presenta el mejor rendimiento, siendo lo único que cambia los parámetros de *gamma* y C. Es notable ver que un ligero cambio en estos dos parámetros puede producir que un sistema pase de ser muy malo, a ser muchísimo mejor.

Como hemos podido ver, los resultados son muy buenos en general, pero algunos de estos sistemas tienen un problema (principalmente las RNAs), y es que el tiempo de

entrenamiento es mucho mayor que en aproximaciones anteriores. Creemos que esto se puede deber tanto por aumentar el número de instancias a utilizar por el sistema (anteriormente 1090, y en esta aproximación 3194), como por el número de características que se utilizan (122). Así, y ante resultados tan satisfactorios, creemos que en futuras aproximaciones deberíamos intentar reducir el número de características, siempre tratando de no perder el rendimiento de clasificación del sistema, permitiéndonos producir modelos que sean más rápidos de entrenar, a la vez que más rápidos para realizar predicciones ante nuevos patrones.

En la matriz de confusión de la Tabla 22 hemos omitido por razones de tamaño las filas y las columnas donde el sistema no se confunde, a excepción de las 5 primeras y las 8 ultimas para definir el rango de notas de esta aproximación y su patrón diagonal de aciertos. Al igual que anteriores aproximaciones podemos destacar que los fallos son entre notas con frecuencias muy parecidas como pueden ser *G1(48,99 Hz)* y *A1(55 Hz)* o *G1(48,99 Hz)* y *E2(82,40 Hz)*.

		Predicc.																														
		C1	C2	C3	C4	C5	C6	C7	C8	A1	A2	...	B4	B5	B6	B7	D1	...	E1	E2	E3	...	E6	E7	F1	F2	F3	...	F6	F7	G1	G2	G3	G4	G5	G6	G7
Real	C1	7	0	0	0	0	0	0	0	0	0		0	0	0	0	0		0	0	0		0	0	0	0	0		0	0	0	0	0	0	0	0	0
	C2	0	7	0	0	0	0	0	0	0	0		0	0	0	0	0		0	0	0		0	0	0	0	0		0	0	0	0	0	0	0	0	0
	C3	0	0	11	0	0	0	0	0	0	0	,,,	0	0	0	0	0	,,,	0	0	0	,,,	0	0	0	0	0	,,,	0	0	0	0	0	0	0	0	0
	C4	0	0	0	15	0	0	0	0	0	0		0	0	0	0	0		0	0	0		0	0	0	0	0		0	0	0	0	0	0	0	0	0
	C5	0	0	0	0	32	0	0	0	0	0		0	0	0	0	0		0	0	0		0	0	0	0	0		0	0	0	0	0	0	0	0	0
	...	...																	...																		
	B5	0	0	0	0	0	0	0	0	0	0		0	14	0	0	0		0	0	0		0	0	0	0	0		0	0	0	0	0	0	0	0	0
	B6	0	0	0	0	0	0	0	1	0	0	...	0	0	18	0	0	...	0	0	0	...	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0
	B7	0	0	0	0	0	0	0	0	0	0		0	0	0	14	0		0	0	0		0	0	0	0	0		0	0	0	0	0	0	0	0	0
	...	...																	...																		
	E7	0	0	0	0	0	0	0	0	0	0		0	0	0	0	0		0	0	0		0	17	0	0	0		0	0	0	0	0	0	0	0	0
	F1	0	0	0	0	0	0	0	0	2	0	...	0	0	0	0	0	...	0	0	0	..	0	0	4	0	0	...	0	0	0	0	0	0	0	0	0
	F2	0	0	0	0	0	0	0	0	0	0		0	0	0	0	0		0	0	0		0	0	0	12	0		0	0	0	0	0	0	0	0	0
	...	...																	...																		
	F7	0	0	0	0	0	0	0	0	0	0		0	0	0	0	0		0	0	0		0	0	0	0	0		0	15	0	0	0	0	0	0	0
	G1	0	0	0	0	0	0	0	0	3	0		0	0	0	0	0		0	1	0		0	0	0	0	0		0	0	5	0	0	0	0	0	0
	G2	0	0	0	0	0	0	0	0	0	0		0	0	0	0	0		0	0	0		0	0	0	0	0		0	0	0	12	0	0	0	0	0
	G3	0	0	0	0	0	0	0	0	2	0		0	0	0	0	0		0	0	0		0	0	0	0	0		0	0	0	0	5	0	0	0	0
	G4	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	...	0	0	0	...	0	0	0	0	0	...	0	0	0	0	0	11	0	0	0
	G5	0	0	0	0	0	0	0	0	0	0		0	0	0	0	0		0	0	0		0	0	0	0	0		0	0	0	0	0	0	10	0	0
	G6	0	0	0	0	0	0	0	0	0	0		0	0	0	0	0		0	0	0		0	0	0	0	0		0	0	0	0	0	0	0	7	0
	G7	0	0	0	0	0	0	0	0	0	0		0	0	0	0	0		0	0	0		0	0	0	0	0		0	0	0	0	0	0	0	0	17

Tabla 22: Matriz de confusión del mejor modelo

5.4. Cuarta aproximación

5.4.1. Descripción

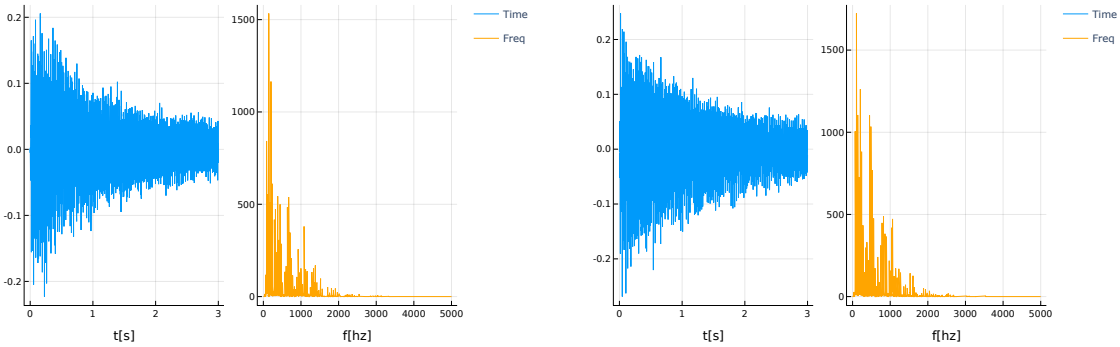
Como seguimos obteniendo resultados muy satisfactorios, en esta aproximación aumentaremos a 85 las notas a clasificar y dispondremos de 5406 patrones para ser utilizados por los modelos.

Las notas que vamos a clasificar constituyen el total de notas del piano, por lo que en esta aproximación será donde veamos cómo de bien se comporta nuestro sistema ante el problema completo. En concreto, las nuevas notas que añadimos son los sostenidos de cada escala. Estos presentan un especial problema, ya que solo están a distancia de un semitono con respecto a la nota original, resultando en frecuencias muy cercanas.

Para intentar paliar este inconveniente, decidimos añadir dos nuevas características que son:

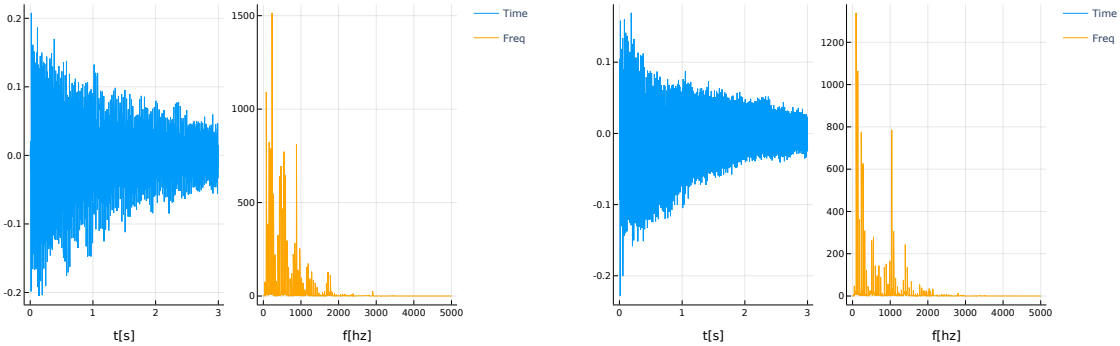
- **Intensidad absoluta máxima**: Que determina la intensidad máxima que alcanza la señal en frecuencia.
- **Frecuencia absoluta máxima**: Que determina la frecuencia absoluta máxima que toma la señal.

Así, las notas que vamos a clasificar son: *A1*, *A2*, *A3*, *A4*, *A5*, *A6*, *A7*, *A#1*, *A#2*, *A#3*, *A#4*, *A#5*, *A#6*, *A#7*, *B1*, *B2*, *B3*, *B4*, *B5*, *B6*, *B7*, *C1*, *C2*, *C3*, *C4*, *C5*, *C6*, *C7*, *C8*, *C#1*, *C#2*, *C#3*, *C#4*, *C#5*, *C#6*, *C#7*, *C8*, *D1*, *D2*, *D3*, *D4*, *D5*, *D6*, *D7*, *D#1*, *D#2*, *D#3*, *D#4*, *D#5*, *D#6*, *D#7*, *E1*, *E2*, *E3*, *E4*, *E5*, *E6*, *E7*, *F1*, *F2*, *F3*, *F4*, *F5*, *F6*, *F7*, *F#1*, *F#2*, *F#3*, *F#4*, *F#5*, *F#6*, *F#7*, *G1*, *G2*, *G3*, *G4*, *G5*, *G6*, *G7*. *G#1*, *G#2*, *G#3*, *G#4*, *G#5*, *G#6* y *G#7*.
Se pueden ver algunos ejemplos de estas notas, en el dominio del tiempo y de la frecuencia, en las Figuras 8a, 8b, 9a y 9b.



(a) A#1 (b) C#1

Figura 8



(a) D#1 (b) F#1

Figura 9

Se muestran a continuación los parámetros de normalización (volveremos a utilizar media cero) para cada característica en las Tablas 23, 24 25 y 26.

Tabla 23: Parámetros de normalización (I)

Característica	Media	Desviación Típica
E	0,005290	0,009127
zero-crossing	3317,7852	4477,6226
abs-max	1445,9364	1557,9128
abs-max-freq	980,5882	1027,9849
m_1	2,5213	4,892
m_2	19,776	34,19
m_3	18,500	33,448
m_4	20,293	28,46
m_5	15,832	24,200
m_6	13,741	21,621
m_7	13,71	20,745
m_8	10,299	17,33
m_9	9,490	18,469
m_{10}	9,750	19,446
m_{11}	9,340	13,882
m_{12}	7,7851	11,177
m_{13}	6,734	13,808
m_{14}	5,5433	8,790
m_{15}	4,2768	6,906
m_{16}	4,459	7,8797
m_{17}	2,9236	5,253
m_{18}	2,6162	5,561
m_{19}	2,5326	4,6802
m_{20}	1,656	4,287
m_{21}	1,6121	4,2043
m_{22}	1,1647	3,044
m_{23}	0,87163	2,4234
m_{24}	0,83163	2,1765
m_{25}	0,61601	1,4981
m_{26}	0,5027	1,2157
m_{27}	0,44925	1,2129
m_{28}	0,3179	1,0582
m_{29}	0,30106	1,2961
m_{30}	0,12668	0,24906

Tabla 24: Parámetros de normalización (II)

Característica	Media	Desviación Típica
std_1	3,7071	8,894764
std_2	41,934	100,382
std_3	41,4962	100,464
std_4	50,629	97,2513
std_5	32,271	66,256
std_6	31,7720	64,7034
std_7	30,2005	56,2919
std_8	24,334	49,0758
std_9	17,220	40,076
std_{10}	21,475	49,6641
std_{11}	16,3562	32,2951
std_{12}	13,8094	27,0130
std_{13}	13,4940	35,2555
std_{14}	9,5112	19,509
std_{15}	7,41763	16,48
std_{16}	7,5551	18,8730
std_{17}	5,6317	11,9381
std_{18}	3,8923	10,9001
std_{19}	4,6173	10,13361
std_{20}	2,56702	8,3345
std_{21}	2,7792	7,7385
std_{22}	1,88428	5,5109
std_{23}	1,4960	4,5867
std_{24}	1,4450	3,7847
std_{25}	0,9040	2,5514
std_{26}	0,7006	1,8386
std_{27}	0,6376	1,6106
std_{28}	0,4845	1,7358
std_{29}	0,3862	1,3208
std_{30}	0,1850	0,458

Tabla 25: Parámetros de normalización (III)

Característica	Media	Desviación Típica
$\max_1$	33,1101	99,812
$\max_2$	415,371	1154,78
$\max_3$	422,344	1112,56
$\max_4$	536,34	1134,93
$\max_5$	317,4	698,87
$\max_6$	316,106	644,35
$\max_7$	293,75	569,42
$\max_8$	239,867	471,90
$\max_9$	162,639	375,881
$\max_{10}$	207,691	447,703
$\max_{11}$	156,600	323,314
$\max_{12}$	132,938	264,185
$\max_{13}$	129,619	310,572
$\max_{14}$	90,649	186,69
$\max_{15}$	72,608	163,450
$\max_{16}$	68,424	165,545
$\max_{17}$	55,8599	115,989
$\max_{18}$	35,6530	97,610
$\max_{19}$	40,5178	84,0128
$\max_{20}$	22,4045	67,513
$\max_{21}$	23,7228	55,7828
$\max_{22}$	16,0673	41,5164
$\max_{23}$	13,3513	36,8658
$\max_{24}$	11,762	26,1254
$\max_{25}$	7,43738	19,2017
$\max_{26}$	5,70138	14,6421
$\max_{27}$	4,86746	10,7949
$\max_{28}$	3,84794	11,6458
$\max_{29}$	2,92127	8,3125
$\max_{30}$	1,58208	3,95033

Tabla 26: Parámetros de normalización (IV)

Característica	Media	Desviación Típica
max-freq$_1$	34,5451	14,4183
max-freq$_2$	108,991	14,5430
max-freq$_3$	157,662	22,506
max-freq$_4$	237,148	26,1689
max-freq$_5$	308,711	24,8007
max-freq$_6$	404,293	26,9718
max-freq$_7$	488,652	23,4741
max-freq$_8$	584,631	31,8488
max-freq$_9$	671,49	29,4884
max-freq$_{10}$	784,34	33,2132
max-freq$_{11}$	905,138	34,9168
max-freq$_{12}$	1005,738	39,893
max-freq$_{13}$	1136,83	46,6669
max-freq$_{14}$	1264,43	42,461
max-freq$_{15}$	1412,91	49,6299
max-freq$_{16}$	1537,37	44,498
max-freq$_{17}$	1700,55	47,240
max-freq$_{18}$	1871,94	52,690
max-freq$_{19}$	2058,19	61,45
max-freq$_{20}$	2220,44	62,3123
max-freq$_{21}$	2422,43	62,4462
max-freq$_{22}$	2629,67	65,414
max-freq$_{23}$	2861,0	72,737
max-freq$_{24}$	3082,92	77,842
max-freq$_{25}$	3346,0	74,7681
max-freq$_{26}$	3592,84	87,462
max-freq$_{27}$	3867,42	90,456
max-freq$_{28}$	4171,07	102,076
max-freq$_{29}$	4471,4	94,3442
max-freq$_{30}$	4826,7	102,899

Donde:

- E indica la **energía de la señal**.
- m_i indica la **media de la intensidad** en el intervalo i en frecuencia.
- std_i indica la **desviación típica de la intensidad** en el intervalo i en frecuencia.
- max_i indica el **máximo de la intensidad** en el intervalo i en frecuencia.
- $max\text{-}freq_i$ indica la **frecuencia donde se alcanza el máximo de intensidad** en el intervalo i en frecuencia.

Para estos experimentos se volvió a repetir 20 veces el entrenamiento de la RNA en cada *fold*, devolviendo el resultado promedio.

Utilizaremos de nuevo como medida el promedio del **F1-Score** en los *folds*. Si hay empate en dicha media, se elegirá el modelo con menor desviación típica de la métrica.

5.4.2. Resultados

Para permitir la reproducibilidad de los experimentos, se utilizó *100* como semilla de generación de números aleatorios.

5.4.2.1 RNA

Para el entrenamiento de las RNA, se han utilizado los siguientes parámetros en común:

- Ratio de aprendizaje: 0,01
- Ratio de patrones para el conjunto de validación: 0,2.
- Máximo número de ciclos (*epochs*) de entrenamiento: 200.
- Máximo número de ciclos (*epochs*) de entrenamiento sin mejorar el error de validación: 5.
- Algoritmo de optimización: ADAM.
- Función de *loss*: *Cross Entropy*.

El hiperparámetro que hemos variado ha sido, de nuevo, la arquitectura de neuronas a utilizar, obteniendo 8 configuraciones diferentes.

Se muestran los resultados de la experimentación en la Tabla 27.

Tabla 27: Resultados RNA

Arquitectura	F1-Score	Precisión
[16]	$0{,}95148 \pm 0{,}00551$	$0{,}96105 \pm 0{,}00471$
[32]	$0{,}98759 \pm 0{,}00345$	$0{,}9897 \pm 0{,}00221$
[64]	$0{,}99345 \pm 0{,}00309$	$0{,}99428 \pm 0{,}00162$
[128]	$0{,}99293 \pm 0{,}00258$	$0{,}99424 \pm 0{,}00123$
[256]	$0{,}99269 \pm 0{,}00345$	$0{,}9936 \pm 0{,}00299$
[512]	$0{,}98348 \pm 0{,}0061$	$0{,}98603 \pm 0{,}00423$
[64, 64]	$0{,}98322 \pm 0{,}00373$	$0{,}98554 \pm 0{,}00292$
[64, 128]	$0{,}9818 \pm 0{,}00226$	$0{,}98428 \pm 0{,}00227$

5.4.2.2 SVM

Para las máquinas de soporte vectorial hemos utilizado 8 configuraciones distintas de los hiperparámetros de *kernel*, grado del *kernel* (sólo se utiliza en *kernel* polinomial), *gamma* y *C*.

Se muestran los resultados de la experimentación en la Tabla 28.

Tabla 28: Resultados SVM

(kernel, grado, gamma, C)	F1-Score	Precisión
(poly, 3, 1, 1)	$0{,}98957 \pm 0{,}00723$	$0{,}9911 \pm 0{,}00594$
(rbf, 3, 1, 1)	$0{,}08773 \pm 0{,}01527$	$0{,}09143 \pm 0{,}01179$
(sigmoid, 3, 1, 1)	$0{,}07182 \pm 0{,}00909$	$0{,}1393 \pm 0{,}01431$
(poly, 3, 10, 0,1)	$0{,}99152 \pm 0{,}00418$	$0{,}99191 \pm 0{,}00394$
(rbf, 3, 10, 0,1)	$0{,}00068 \pm 3{,}0e-5$	$0{,}02965 \pm 0{,}00136$
(rbf, 3, 0,001, 100)	$0{,}99605 \pm 0{,}00408$	$0{,}99612 \pm 0{,}00324$
(rbf, 3, 0,01, 100)	$0{,}9947 \pm 0{,}00389$	$0{,}99459 \pm 0{,}00313$
(poly, 3, 100, 0,001)	$0{,}99142 \pm 0{,}00449$	$0{,}99237 \pm 0{,}00367$

5.4.2.3 Árboles de decisión

Para los árboles de decisión hemos utilizado 6 configuraciones distintas del hiperparámetro de profundidad máxima.

Se muestran los resultados de la experimentación en la Tabla 29.

5.4.2.4 kNN

Para el modelo kNN hemos utilizado 6 configuraciones distintas del hiperparámetro de k (número de vecinos más cercanos).

Se muestran los resultados de la experimentación en la tabla 30.

Tabla 29: Resultados Árbol de decisión

Altura máxima	F1-Score	Precisión
32	$0{,}93688 \pm 0{,}0118$	$0{,}94355 \pm 0{,}00995$
64	$0{,}92919 \pm 0{,}01756$	$0{,}93871 \pm 0{,}01425$
128	$0{,}92578 \pm 0{,}02024$	$0{,}93693 \pm 0{,}01502$
256	$0{,}93336 \pm 0{,}01516$	$0{,}94104 \pm 0{,}01578$
512	$0{,}93528 \pm 0{,}01411$	$0{,}9438 \pm 0{,}01019$
1024	$0{,}93482 \pm 0{,}01642$	$0{,}94221 \pm 0{,}01563$

Tabla 30: Resultados kNN

K	F1-Score	Precisión
2	$0{,}98126 \pm 0{,}00699$	$0{,}98422 \pm 0{,}00492$
3	$0{,}98306 \pm 0{,}00593$	$0{,}98473 \pm 0{,}00477$
7	$0{,}97125 \pm 0{,}00862$	$0{,}9767 \pm 0{,}00708$
15	$0{,}95195 \pm 0{,}01088$	$0{,}96343 \pm 0{,}00783$
31	$0{,}93762 \pm 0{,}01577$	$0{,}95012 \pm 0{,}01025$
63	$0{,}88327 \pm 0{,}01502$	$0{,}91107 \pm 0{,}00954$

5.4.3. Discusión

Como se puede ver en los resultados, el funcionamiento de nuestro sistema resolviendo el problema completo, es decir, clasificando todas las notas del piano, es sorprendente. Con el mejor modelo (de nuevo un SVM), obtuvimos una precisión de $0{,}99612 \pm 0{,}00324$ y un F1-Score de $0{,}99605 \pm 0{,}00408$, haciendo uso de un *kernel* de base radial, un valor de *gamma* 0,001 y un valor de C 100.

De nuevo destacar que, a pesar de obtener RNAs con valores de F1-Score muy altos (>99 %), el tiempo invertido en el entrenamiento las hace modelos menos interesantes, sobre todo comparados con los modelos basados en SVM o kNN. Incluso tras la reducción del número máximo de ciclos de entrenamiento de 1500 en las aproximaciones pasadas, a 200 ciclos en esta aproximación.

Con respecto a las características, creemos que tanto la **frecuencia máxima absoluta** y la **intensidad máxima absoluta en frecuencia**, son importantes a la hora de clasificar correctamente las notas. Sin embargo, ahora que tenemos el problema completo resuelto, consideramos que es el momento de tratar de minimizar el número de características necesarias, perdiendo la menor precisión en la clasificación posible. Este aspecto será en el que nos centremos en la siguiente aproximación.

Se muestra en la Tabla 31 la matriz de confusión del mejor modelo obtenido. En ella, podemos observar como se confunde mayoritariamente en notas que se encuentran a distancias cercanas como son *C7* y *C#7* o *A1* y *A#1*.

Real \ Predicc.	A#1	A#2	A#3	...	C#7	C1	...	C5	C6	C7	C8	D#1	D#2	D#3	D#4	D#5	D#6	...	G5	G6	G7
A#1	12	0	0		0	0		0	0	0	0	0	0	0	0	0	0		0	0	0
A#2	0	9	0	...	0	0	...	0	0	0	0	0	0	0	0	0	0	...	0	0	0
A#3	0	0	8		0	0		0	0	0	0	0	0	0	0	0	0		0	0	0
...	...																				
A#7	0	0	0		0	0		0	0	0	0	0	0	0	0	0	0		0	0	0
A1	1	0	0	...	0	0	...	0	0	0	0	0	0	0	0	0	0	...	0	0	0
A2	0	0	0		0	0		0	0	0	0	0	0	0	0	0	0		0	0	0
...	...																				
C6	0	0	0		0	0		0	12	0	0	0	0	0	0	0	0		0	0	0
C7	0	0	0	...	1	0	...	0	0	23	0	0	0	0	0	0	0	...	0	0	0
C8	0	0	0		0	0		0	0	0	20	0	0	0	0	0	0		0	0	0
...	...																				
D#3	0	0	0		0	0		0	0	0	0	0	0	4	0	0	0		0	0	0
D#4	0	0	0		0	0		0	0	0	0	0	0	0	12	0	0		0	0	0
D#5	0	0	0	...	0	0	...	0	0	0	0	0	0	0	1	20	0	...	0	0	0
D#6	0	0	0		0	0		0	0	0	0	0	0	0	0	0	10		0	0	0
...	...																				
G#7	0	0	0		0	0		0	0	0	1	0	0	0	0	0	0		0	0	0
...	...																				
G5	0	0	0		0	0		0	0	0	0	0	0	0	0	0	0		8	0	0
G6	0	0	0	...	0	0	...	0	0	0	0	0	0	0	0	0	0	...	0	10	0
G7	0	0	0		0	0		0	0	0	0	0	0	0	0	0	0		0	0	28

Tabla 31: Matriz de confusión mejor modelo

5.5. Quinta aproximación

5.5.1. Descripción

Una vez ya hemos comprobado que nuestro sistema obtiene resultados muy satisfactorios ante el problema completo, ahora nos centraremos en reducir el número de características necesarias, tratando de perder la menor precisión posible. Esto nos ayudará a disminuir, tanto el tiempo de entrenamiento, como el tiempo de predicción, resultando así en sistemas más rápidos y más eficientes en términos de energía consumida.

Para seleccionar las características más relevantes en la clasificación, hemos utilizado la herramienta *mRMR-enhanced* (véase Hermo González, 2022).

Esta herramienta utiliza el concepto de información mutua para realizar un *ranking* de las características del *dataset*, siguiendo el algoritmo de mínima redundancia-máxima relevancia (*mRMR*), propuesto por Ding y Peng, 2005. Este algoritmo busca obtener un *ranking* de características que maximicen la ganancia de información (relevancia) respecto a la clase, pero que también se minimice la redundancia entre las características previamente seleccionadas.

Para su funcionamiento, es necesario que los datos estén discretizados, por lo que se ha preprocesado el *dataset*, utilizando una discretización de 10 *bins*, siguiendo una estrategia de reparto de los datos por cuantiles.

Así, tras algunas pruebas, hemos decidido seleccionar las 70 primeras características. Se muestran los parámetros de normalización para cada una de ellas (ordenadas según el *ranking* producido por el algoritmo *mRMR*) en las Tablas 32, 33 y 34.

Tabla 32: Parámetros de normalización (I)

Característica	Media	Desviación Típica
abs-max-freq	980,58826	1027,9849
max-freq$_{19}$	2058,1985	61,4545
max-freq$_{17}$	1700,5549	47,24032
max-freq$_{14}$	1264,4369	42,46175
max-freq$_{13}$	1136,8329	46,666904
max-freq$_{21}$	2422,4363	62,446247
max-freq$_{12}$	1005,73865	39,89346
max-freq$_{27}$	3867,4265	90,45616
max-freq$_{11}$	905,13873	34,916843
max-freq$_{20}$	2220,4458	62,312325
max-freq$_{16}$	1537,3774	44,49866
max-freq$_{23}$	2861,094	72,73729
max-freq$_{7}$	488,65253	23,474133
max-freq$_{24}$	3082,9204	77,84288
max-freq$_{10}$	784,3489	33,213223
max-freq$_{15}$	1412,9193	49,629932
max-freq$_{22}$	2629,6775	65,41493
max-freq$_{9}$	671,4978	29,488453
max-freq$_{18}$	1871,9404	52,69005
max-freq$_{8}$	584,63116	31,848839
max-freq$_{25}$	3346,068	74,768135
max-freq$_{26}$	3592,8464	87,46203
max-freq$_{28}$	4171,0786	102,07654
zero-crossing	3317,7852	4477,6226
abs-max	1445,9364	1557,9128

Tabla 33: Parámetros de normalización (II)

Característica	Media	Desviación Típica
m_1	2,5213237	4,892542
m_2	19,776129	34,19082
m_3	18,500994	33,448643
m_4	20,293629	28,46877
m_5	15,832674	24,200994
m_6	13,741416	21,621166
m_7	13,71829	20,745993
m_8	10,299257	17,33558
m_9	9,490465	18,469109
m_{10}	9,750799	19,446434
m_{11}	9,340946	13,882007
max-$freq_6$	404,29333	26,971804
max-$freq_{29}$	4471,467	94,344215
max-$freq_{30}$	4826,714	102,89994
max_8	239,86774	471,9059
max-$freq_4$	237,14865	26,168978
max_6	316,10635	644,3593
std_8	24,33468	49,075897
std_6	31,772034	64,70347
abs-max	1445,9364	1557,9128
std_7	30,2005	56,2919
max_{11}	156,60005	323,31445
max_9	162,63959	375,88123

Tabla 34: Parámetros de normalización (III)

Característica	Media	Desviación Típica
max_{12}	132,93877	264,18588
max_{10}	207,69164	447,7031
max_{14}	90,64936	186,6918
max_{7}	293,7506	569,4251
max_{13}	129,61961	310,57278
max-$freq_{2}$	108,991	14,5430
max_{5}	317,401	698,8781
std_{10}	21,47563	49,6641
max_{17}	55,8599	115,98919
max_{15}	72,60896	163,45084
std_{9}	17,22069	40,07678
max-$freq_{3}$	157,66272	22,5065
max_{16}	68,42438	165,54524
max_{4}	536,3476	1134,9387
m_{8}	10,299257	17,33558
std_{11}	16,3562	32,2951
E	0,00529	0,00912
m_{6}	13,741416	21,621166
std_{14}	9,511233	19,50926
std_{13}	13,4940	35,2555
std_{5}	32,27192	66,25659
std_{12}	13,809475	27,01305
std_{4}	50,62945	97,2513
m_{7}	13,718	20,745993
std_{15}	7,4176	16,4802
max_{18}	35,6530	97,61015
max-$freq_{1}$	34,5448	14,4189
max_{19}	40,517	84,0128
std_{16}	7,5551	18,8730
std_{17}	5,63178	11,9381
m4	20,2936	28,468
max_{21}	23,7228	55,7828
m_{10}	9,7507	19,4464

Donde:

- E indica la **energía de la señal**.
- m_i indica la **media de la intensidad** en el intervalo i en frecuencia.
- std_i indica la **desviación típica de la intensidad** en el intervalo i en frecuencia.
- max_i indica el **máximo de la intensidad** en el intervalo i en frecuencia.
- $max\text{-}freq_i$ indica la **frecuencia donde se alcanza el máximo de intensidad** en el intervalo i en frecuencia.

Utilizaremos de nuevo como medida el promedio del **F1-Score** en los *folds*. Si hay empate en dicha media, se elegirá el modelo con menor desviación típica de la métrica.

5.5.2. Resultados

Para permitir la reproducibilidad de los experimentos, se utilizó *100* como semilla de generación de números aleatorios.

5.5.2.1 RNA

Para el entrenamiento de las RNA, se han utilizado los siguientes parámetros en común:

- Ratio de aprendizaje: 0,01
- Ratio de patrones para el conjunto de validación: 0,2.
- Máximo número de ciclos (*epochs*) de entrenamiento: 200.
- Máximo número de ciclos (*epochs*) de entrenamiento sin mejorar el error de validación: 5.
- Algoritmo de optimización: ADAM.
- Función de *loss*: *Cross Entropy*.

Se muestran los resultados de la experimentación en la Tabla 35.

5.5.2.2 SVM

Para las máquinas de soporte vectorial hemos utilizado 8 configuraciones distintas de los hiperparámetros de *kernel*, grado del *kernel* (sólo se utiliza en *kernel* polinomial), *gamma* y *C*.

Se muestran los resultados de la experimentación en la Tabla 36.

Tabla 35: Resultados RNA

Arquitectura	F1-Score	Precisión
[16]	$0{,}91452 \pm 0{,}00544$	$0{,}92437 \pm 0{,}00475$
[32]	$0{,}97012 \pm 0{,}00554$	$0{,}97169 \pm 0{,}00485$
[64]	$0{,}98034 \pm 0{,}00697$	$0{,}98149 \pm 0{,}00567$
[128]	$0{,}98007 \pm 0{,}00613$	$0{,}98054 \pm 0{,}00449$
[256]	$0{,}9757 \pm 0{,}0058$	$0{,}97508 \pm 0{,}00582$
[512]	$0{,}94062 \pm 0{,}00657$	$0{,}94438 \pm 0{,}00534$
[64, 64]	$0{,}95589 \pm 0{,}00402$	$0{,}95973 \pm 0{,}00353$
[64, 128]	$0{,}95232 \pm 0{,}00708$	$0{,}95488 \pm 0{,}00592$

Tabla 36: Resultados SVM

(kernel, grado, gamma, C)	F1-Score	Precisión
(poly, 3, 1, 1)	$0{,}98565 \pm 0{,}00523$	$0{,}9872 \pm 0{,}00429$
(rbf, 3, 1, 1)	$0{,}13958 \pm 0{,}01127$	$0{,}12843 \pm 0{,}0095$
(sigmoid, 3, 1, 1)	$0{,}0372 \pm 0{,}00548$	$0{,}10149 \pm 0{,}00667$
(poly, 3, 10, 0,1)	$0{,}98545 \pm 0{,}00508$	$0{,}98636 \pm 0{,}00365$
(rbf, 3, 10, 0,1)	$0{,}00068 \pm 3{,}0e-5$	$0{,}02965 \pm 0{,}00136$
(rbf, 3, 0,001, 100)	$0{,}99186 \pm 0{,}00543$	$0{,}99156 \pm 0{,}00571$
(rbf, 3, 0,01, 100)	$0{,}99375 \pm 0{,}00263$	$0{,}99315 \pm 0{,}00212$
(poly, 3, 100, 0,001)	$0{,}98557 \pm 0{,}00386$	$0{,}98683 \pm 0{,}00308$

5.5.2.3 Árboles de decisión

Para los árboles de decisión hemos utilizado 6 configuraciones distintas del hiperparámetro de profundidad máxima.

Se muestran los resultados de la experimentación en la Tabla 37.

Tabla 37: Resultados Árbol de decisión

Altura máxima	F1-Score	Precisión
32	$0{,}93524 \pm 0{,}01075$	$0{,}9389 \pm 0{,}00772$
64	$0{,}92723 \pm 0{,}01297$	$0{,}93348 \pm 0{,}01031$
128	$0{,}93011 \pm 0{,}01126$	$0{,}93468 \pm 0{,}01126$
256	$0{,}93144 \pm 0{,}01067$	$0{,}93798 \pm 0{,}00985$
512	$0{,}92927 \pm 0{,}00971$	$0{,}93616 \pm 0{,}00711$
1024	$0{,}92633 \pm 0{,}01424$	$0{,}93278 \pm 0{,}01285$

5.5.2.4 kNN

Para el modelo kNN hemos utilizado 6 configuraciones distintas del hiperparámetro de k (número de vecinos más cercanos).

Se muestran los resultados de la experimentación en la tabla 38.

Tabla 38: Resultados kNN

K	F1-Score	Precisión
2	$0{,}9699 \pm 0{,}00444$	$0{,}9699 \pm 0{,}00401$
3	$0{,}9739 \pm 0{,}0055$	$0{,}97356 \pm 0{,}00475$
7	$0{,}96825 \pm 0{,}00482$	$0{,}96936 \pm 0{,}00492$
15	$0{,}95356 \pm 0{,}01199$	$0{,}95761 \pm 0{,}00756$
31	$0{,}92823 \pm 0{,}01575$	$0{,}93464 \pm 0{,}01214$
63	$0{,}88728 \pm 0{,}01083$	$0{,}90243 \pm 0{,}00881$

5.5.3. Discusión

Como se puede ver en los resultados, el funcionamiento de nuestro sistema sigue siendo muy bueno, a pesar de haber reducido el número de características de 122 a 70. El mejor modelo es, de nuevo, un SVM caracterizado por un *kernel* de base radial, un valor de *gamma* de 0,01 y un valor de C de 100. Dicho modelo alcanza una precisión de $0{,}99315 \pm 0{,}00212$ y un *F1-Score* de $0{,}99375 \pm 0{,}00263$, unos valores más que suficientes para considerar que resuleve el problema satisfactoriamente.

Aunque el mejor modelo de esta aproximación sea muy similar al mejor modelo de la anterior, cabe destacar que en las RNAs sí hay arquitecturas en las que se alcanzan peores resultados. Sin embargo, ya que las mejores RNAs siguen teniendo un *F1-Score* superior al 99 %, esto no es muy grave.

También cabe destacar que, con la reducción del número de las características y, por consecuente, la reducción de la dimensionalidad del problema, los modelos basados en *kNN* presentan resultados aún mejores que en la aproximación anterior. Esto es un resultado que tiene sentido que sea así, ya que las instancias se pueden distribuír de una mejor manera en el espacio latente de nuestro problema.

Podemos concluír, por lo tanto, que llevar a cabo una reducción de características ha sido satisfactorio y presenta una mejor solución que no hacerlo, puesto que, aunque el mejor modelo sea similar, la reducción de características hace que los modelos sean más rápidos de entrenar y más rápidos en hacer predicciones. Además, sabiendo que cuanto más sencillo sea un modelo, mejor capacidad de generalización tiene, esta reducción favorece también al aumento de la potencia de generalización.

Se muestra en la tabla 39 la matriz de confusión del mejor modelo obtenido en una ejecución con una separación de 20 % para test. Presenta resultados muy similares a los de la aproximación anterior.

		Predicc.																						
		A#1	...	A#6	A#7	A1	...	A6	A7	B1	...	C#5	C#6	C#7	...	C7	C8	D#1	...	G#6	G#7	G1	...	G7
Real	A#1	11	...	0	0	0	...	0	0	0	...	0	0	0	...	0	0	0	...	0	0	0	...	0
	...	...											...											
	A#6	0		12	0	0		0	0	0		0	0	0		0	0	0		0	0	0		0
	A#7	0	...	0	20	0	...	0	0	0	...	0	0	0	...	0	1	0	...	0	0	0	...	0
	A1	0		0	0	26		0	0	0		0	0	0		0	0	0		0	0	0		0
	...	...											...											
	A6	0		0	0	0		10	0	0		0	0	0		0	0	0		0	0	0		0
	A7	0	...	0	0	0	...	0	18	0	...	0	0	0	...	0	0	0	...	0	1	0	...	0
	B1	0		0	0	0		0	0	15		0	0	0		0	0	0		0	0	0		0
	...	...											...											
	C#5	0		0	0	0		0	0	0		33	0	0		0	0	0		0	0	0		0
	C#6	0	...	0	0	0	...	0	0	0	...	0	13	0	...	0	0	0	...	0	0	0	...	0
	C#7	0		0	0	0		0	0	0		0	0	18		0	0	0		0	0	0		0
	...	...											...											
	D5	0		0	0	0		0	0	0		0	0	0		0	0	0		0	0	0		0
	D6	0	...	0	0	0	...	0	0	0	...	0	1	0	...	0	0	0	...	0	0	0	...	0
	D7	0		0	0	0		0	0	0		0	0	0		0	0	0		0	0	0		0
	...	...											...											
	E7	0	...	0	1	0	...	0	0	0	...	0	0	0	...	0	0	0	...	0	0	0	...	0
	...	...											...											
	G#6	0		0	0	0		0	0	0		0	0	0		0	0	0		10	0	0		0
	G#7	0	...	0	0	0	...	0	1	0	...	0	0	0	...	0	0	0	...	0	21	0	...	0
	G1	0		0	0	0		0	0	0		0	0	0		0	0	0		0	0	7		0
	...	...											...											
	G7	0	...	0	0	0	...	0	0	0	...	0	0	0	...	0	0	0	...	0	0	0	...	18

Tabla 39: Matriz de confusión mejor modelo

5.6. Sexta aproximación

5.6.1. Descripción

En esta aproximación utilizaremos técnicas de *Deep Learning* para producir los modelos. En concreto, utilizaremos redes convolucionales o *CNN* en inglés.

Similares a las *RNAs*, las redes convolucionales hacen uso de capas de neuronas artificiales en conjunto con filtros de convolución, intercaladas con funciones de *pooling*, que permiten una mayor capacidad de generalización de dichas neuronas, además de no necesitar de una extracción previa de características de los datos. Ver Indolia y col., 2018.

Las notas que vamos a clasificar serán las mismas que en la aproximación anterior, es decir, las 85 notas del piano, que juntas forman un total de 5046 patrones.

Sin embargo, ahora utilizaremos nuestra base de datos de otra manera. En lugar de extrar las características en los primeros 3 segundos de cada *sample*, utilizaremos solo los primeros 0.1 segundos, ya que con 3 segundos el tiempo de entrenamiento es excesivo con 5406 instancias. Entonces, la matriz que representa la base de datos será de 500 x 5046. Las 500 filas se obtienen de la siguiente manera:

$filas = fs \cdot d_s \cdot \frac{f_{max}}{fs}$

Donde, en nuestro caso, $fs = 48000hz$, $d_s = 0{,}1s$ y $f_{max} = 5000hz$

Al reducir los segundos que utizamos de las muestras, además de hacer que tarden menos en entrenarse las redes, también estamos intentando mejorar el funcionamiento

de nuestro modelo, ya que, si puede funcionar bien utilizando muestras de 0,1 segundos, sería un avance para poder reconocer notas en tiempo real, sin necesidad de tener que utilizar *samples* de 3 segundos.

En lo referente de los datos, se ha realizado una normalización de media cero y desviación típica 1.

Utilizaremos de nuevo como medida el promedio del **F1-Score** en los *folds*. Si hay empate en dicha media, se elegirá el modelo con menor desviación típica de la métrica.

5.6.2. Resultados

Para permitir la reproducibilidad de los experimentos, se utilizó *100* como semilla de generación de números aleatorios.

5.6.2.1 Redes Convolucionales

Para el entrenamiento de las Redes Convolucionales, se han utilizado los siguientes parámetros en común:

- Métrica a maximizar: *F1-Score*
- Ratio de aprendizaje: 0,01
- Máximo número de ciclos de entrenamiento sin mejorar la métrica antes de disminuír la tasa de aprendizaje: 5.
- Máximo valor de *F1-Score* para parar el entrenamiento: 99,9 %.
- Máximo número de ciclos de entrenamiento sin mejorar el valor de la métrica antes de parar el entrenamiento: 10.
- Algoritmo de optimización: *ADAM*.
- Función de *loss*: *Cross Entropy*.

Se han utilizado 10 arquitecturas diferentes en los resultados, variando el filtro de convolución, el tipo de capa de *pooling* y el tamaño de la ventana de dicha capa. En todas las arquitecturas se ha utilizado una función de transferencia para las capas convolucionales de *RELU* (*Rectified Linear Unit*, o rectificador), y como función de salida de la red se ha utilizado una funcion *softmax*.

Debido al elevado tiempo de entrenamiento que suponía entrenar estas redes, hemos decidido no realizar múltiples ejecuciones en cada *fold*. Esto produce resultados menos fiables, ya que la inicialización de los pesos de las neuronas se realiza de forma aleatoria. Si dispusiésemos de más recursos y más tiempo, lo más correcto sería realizar múltiples

ejecuciones dentro de un mismo *fold* y devolver la media de resultados de estas ejecuciones. Pero, con una única ejecución, el entrenamiento de las 10 arquitecturas supuso aproximadamente 4h en nuestros ordenadores.

Además, hay que tener en cuenta también el papel que juega en la disminución del tiempo de entrenamiento la reducción de la duración de las muestras de 3 a 0,1 segundos. Esto implica una reducción de las filas de la matriz de patrones de 7500 a 500.

Se muestran los resultados de la experimentación en la Tabla 40.

Tabla 40: Resultados Redes Convolucionales

Arquitectura	F1-Score	Precisión
conv_1	$0{,}95735 \pm 0{,}0161$	$0{,}96734 \pm 0{,}01078$
conv_2	$0{,}93055 \pm 0{,}01604$	$0{,}94942 \pm 0{,}01134$
conv_3	$0{,}63134 \pm 0{,}43667$	$0{,}65577 \pm 0{,}43699$
conv_4	$0{,}00066 \pm 2{,}0e-5$	$0{,}02906 \pm 0{,}00111$
conv_5	$0{,}72455 \pm 0{,}37877$	$0{,}75191 \pm 0{,}37616$
conv_6	$0{,}00066 \pm 2{,}0e-5$	$0{,}02906 \pm 0{,}00111$
conv_7	$0{,}00154 \pm 0{,}00225$	$0{,}03017 \pm 0{,}00326$
conv_8	$0{,}65012 \pm 0{,}24465$	$0{,}70488 \pm 0{,}24947$
conv_9	$0{,}00124 \pm 0{,}00181$	$0{,}02993 \pm 0{,}00253$
conv_{10}	$0{,}00169 \pm 0{,}00324$	$0{,}02924 \pm 0{,}0012$

Denotación de las arquitecturas utilizadas:

Cada arquitectura $conv_i$ tendrá varias capas concatenadas. Dichas capas podrán ser capas de *pooling*, denotando *MaxPool(i)* como una capa de pooling en la que se calcula el máximo de una ventana de tamaño i y *MeanPool(i)* como una capa en la que se calcula la media de una ventana de tamaño i. También habrá capas de filtros convolucionales denotadas como *Conv(i, j => z, t)*, donde i especifica el tamaño del kernel convolucional, j indica el número de canales de entrada al filtro, z indica el número de canales de salida del filtro y t indica el *pad* (o relleno) del filtro. Dense(i,j) indica una capa de neuronas densa (todas las neuronas conectadas con la capa anterior y con la capa siguiente) de i neuronas, conectada con j neuronas de la siguiente capa.

Tanto las capas de *pooling* como las capas de filtros convolucionales, tratarán los datos sobre en una dimensión (tamaño de ventana para pooling, tamaño del *kernel* convolucional para los filtros convolucionales o el tamaño del relleno en el caso del *pad* de los filtros convolucionales), ya que estamos utilizando audio, no imágenes.

Arquitecturas utilizadas:

- conv_1

- Conv(2, 1 => 4, 1)
- MaxPool(2)
- Conv(2, 4 => 8, 1)
- MaxPool(5)
- Conv(2, 8 => 16, 1)
- MaxPool(5)
- Dense(160,85)

- conv_2
  - Conv(2, 1 => 4, 1)
  - MaxPool(5)
  - Conv(2, 4 => 8, 1)
  - MaxPool(5)
  - Conv(2, 8 => 16, 1)
  - MaxPool(2)
  - Dense(160,85)
- conv_3
  - Conv(2, 1 => 4, 1)
  - MeanPool(5)
  - Conv(2, 4 => 8, 1)
  - MeanPool(2)
  - Conv(2, 8 => 16, 1)
  - MeanPool(5)
  - Dense(160,85)
- conv_4
  - Conv(2, 1 => 4, 1)
  - MeanPool(2)
  - Conv(2, 4 => 8, 1)
  - MeanPool(5)
  - Conv(2, 8 => 16, 1)
  - MaxPool(5)
  - Dense(160,85)
- conv_5

- Conv(3, 1 => 4, 1)
- MaxPool(5)
- Conv(3, 4 => 8, 1)
- MaxPool(2)
- Conv(3, 8 => 16, 1)
- MaxPool(5)
- Dense(160,85)

- conv_6
  - Conv(5, 1 => 4, 2)
  - MaxPool(2)
  - Conv(5, 4 => 8, 2)
  - MaxPool(5)
  - Conv(5, 8 => 16, 2)
  - MaxPool(5)
  - Dense(160,85)
- conv_7
  - Conv(2, 1 => 4, 1)
  - MaxPool(5)
  - Conv(2, 4 => 8, 1)
  - MaxPool(2)
  - Conv(2, 8 => 16, 1)
  - MaxPool(5)
  - Dense(160,85)
- conv_8
  - Conv(3, 1 => 4, 1)
  - MaxPool(2)
  - Conv(3, 4 => 8, 1)
  - MeanPool(5)
  - Conv(3, 8 => 16, 1)
  - MaxPool(5)
  - Dense(160,85)
- conv_9

- Conv(3, 1 => 8, 1)
- MaxPool(2)
- Conv(3, 8 => 16, 1)
- MeanPool(5)
- Conv(3, 4 => 8, 1)
- MeanPool(5)
- Dense(160,85)

- conv_{10}
  - Conv(3, 1 => 4, 1)
  - MeanPool(5)
  - Conv(3, 4 => 8, 1)
  - MeanPool(2)
  - Conv(3, 8 => 16, 1)
  - MaxPool(5)
  - Dense(160,85)

5.6.3. Discusión

Se puede observar en los resultados que hay un par de modelos con alto *F1-Score* y precisión, siendo el mejor modelo la red conv_1, que presenta la siguiente arquitectura:

- Conv(2, 1 => 4, 1)
- MaxPool(2)
- Conv(2, 4 => 8, 1)
- MaxPool(5)
- Conv(2, 8 => 16, 1)
- MaxPool(5)
- Dense(160,85)

Este mejor modelo obtiene un *F1-Score* de $0{,}95735 \pm 0{,}0161$ y una precisión de $0{,}96734 \pm 0{,}01078$. Estos valores son altos y, aunque no tan elevados como en aproximaciones anteriores donde se alcanzó un *F1-Score* mayor al 99 %, este sistema tiene la ventaje de haber sido entrenado sobre *samples* con una duración de 0,1 segundos.

Cabe destacar que hay varios modelos con valores de *F1-Score* y de precisión increíblemente bajos, incluso menores al 1 %. Creemos que esto se puede deber a que

somos inexpertos sobre este tipo de redes y no seleccionamos las arquitecturas con las que experimentar de la mejor manera. Además, muchas de las redes que peores resultados muestran tienen en común que se ha utilizado una capa de *pooling* de media de la ventana, en lugar de calcular el máximo de la ventana. De la misma manera, las redes que mejor se comportan son las que solamente tienen capas de *pooling* de máximos, por lo que sospechamos que otras arquitecturas con estas últimas capas de *pooling* serían un punto sobre el que seguir explorando configuraciones de las redes.

Así mismo, hay redes con una gran inestabilidad, como pueden ser las redes conv_3 y conv_5, que presentan una alta desviación típica. Esto quizá se podría mejorar con varias ejecuciones en cada fold, ya que parece que los resultados de estas dos redes han dependido mucho de si los pesos se han inicializado aleatoriamente en unos buenos valores o no.

Es también notable decir que hemos utilizado un tamaño del canal de los filtros reducido (el máximo tamaño es 16), debido a que con mayores tamaños el tiempo de entrenamiento se hacía excesivo. Quizás con más recursos, podríamos explorar configuraciones que alcanzasen filtros con mayor tamaño de canales y que pudiesen ofrecer mejores resultados.

En definitiva, esta técnica tiene sus ventajas, como puede ser que no sea necesario una extracción previa de características y se puedan ofrecer los datos *raw* (en crudo) a la red, ahorrando así mucho tiempo al usuario de tener hacer ingeniería de características. Pero a su vez, la desventaja es que los tiempos de entrenamiento de estas redes es excesivo, y perdemos el control sobre varias partes de la red, como puede ser la selección de características y poder observar y analizar qué características son las mas importantes. Por lo tanto, aunque estas redes puedan dar buenos resultados, funcionan como una caja aún mas negra que las redes de neuronas tradicionales, haciendo impracticable cualquier tipo de razonamiento que pueda explicar la predicción que realiza, al contrario que pasa con modelos, como por ejemplo, los árboles de decisión.

Se muestra la matriz de confusión en la tabla 41 para una ejecución del sistema con una separación del 20 % de los patrones para test. También se muestra en la figura 10 una gráfica del entrenamiento para esta ejecución.

Esta ejecución muestra muy buenos resultados, con un *F1-Score* del 99,154 % y una precisión del 99,445 %. Esto se debe a que hacer *cross validation* nos sirve para comparar distintos modelos, pero muestra unos resultados pesimistas del comportamiento real del sistema.

A la vista de estos resultados, podemos concluír que son satisfactorios y que el sistema funciona muy bien.

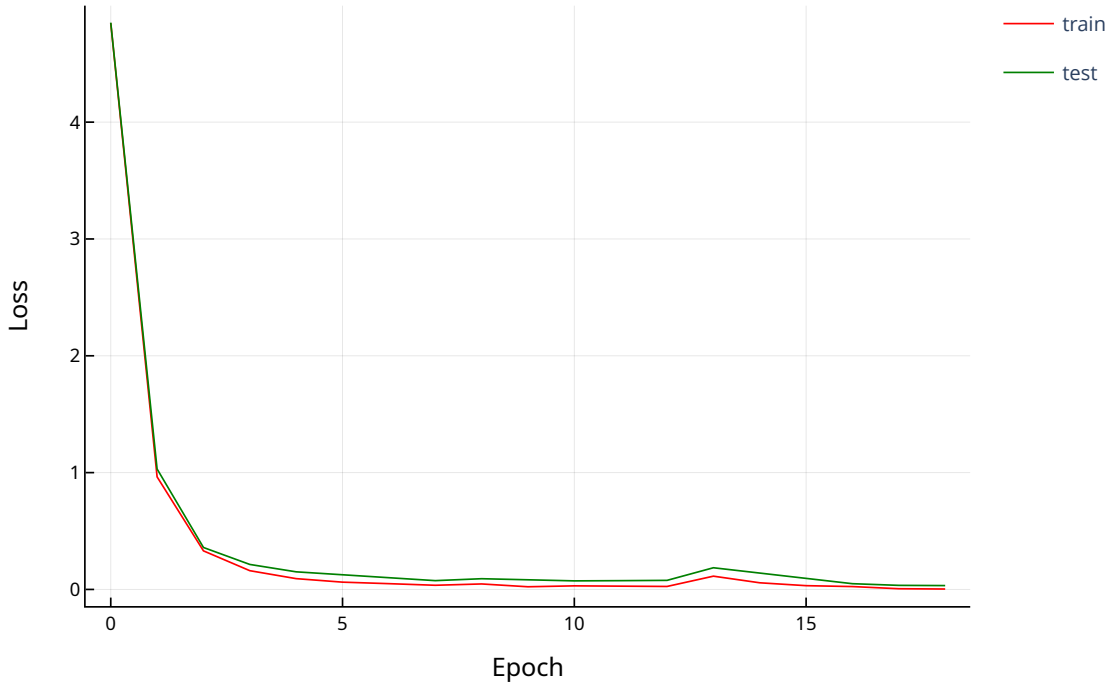


Figura 10: Entrenamiento red convolucional

Predicc.

Real	A#1	...	C#4	C#5	C#6	C#7	C1	...	D#3	D#4	D#5	...	D6	D7	E1	...	G7
A#1	27	...	0	0	0	0	0	...	0	0	0	...	0	0	0	...	0
...									...								
A4	0		0	0	0	2	0		0	0	0		0	0	0		0
A5	0	...	0	0	0	0	0	...	0	0	0	...	0	0	0	...	0
...									...								
C#3	0		0	0	0	0	0		0	0	0		0	0	0		0
C#4	0		20	0	0	0	0		0	0	0		0	0	0		0
C#5	0	...	0	8	0	0	0	...	0	0	0	...	0	0	0	...	0
C#6	0		0	0	11	0	0		0	0	0		0	0	0		0
C#7	0		0	0	0	22	0		0	0	0		0	0	0		0
...									...								
D#3	0	...	0	0	0	0	0	...	22	0	0	...	0	0	0	...	0
D#4	0		0	0	0	0	0		0	8	0		0	0	0		0
D#5	0		0	0	0	0	0		0	0	9		0	0	0		0
...									...								
D6	0		0	0	0	0	0		0	0	0		6	0	0		0
D7	0		0	0	0	0	0		0	0	0		0	4	0		0
E1	0		0	0	0	0	0		0	0	0		0	0	18		0
E2	0	...	0	0	0	0	0	...	0	0	0	...	0	0	0	...	0
E3	0		0	0	0	0	0		0	0	0		0	0	0		0
E4	0		0	0	0	0	0		0	1	0		0	0	0		0
...									...								
F7	0	...	0	0	0	0	0	...	0	0	0	...	0	2	0	...	0
G#1	0		0	0	0	0	0		0	0	0		0	0	0		0
...									...								
G#4	0		0	0	0	0	0		0	0	0		0	0	0		0
G#5	0	...	0	1	0	0	0	...	0	0	0	...	0	0	0	...	0
...									...								
G7	0	...	0	0	0	0	0	...	0	0	0	...	0	0	0	...	3

Tabla 41: Matriz de confusión mejor modelo

6. Conclusiones

A lo largo de la memoria hemos visto cómo, partiendo de los datos en crudo, se pueden aplicar distintas técnicas de aprendizaje automático a un problema concreto; realizando primero una extracción de características que representen a los datos de una forma más compacta y, posteriormente, aplicando modelos de aprendizaje automático para poder hacer predicciones sobre los mismos.

Se ha seguido una metodología basada en aproximaciones que, a nuestro parecer, ha sido muy útil, ya que nos ha permitido ir familiarizándonos con el problema a medida que íbamos incrementando su complejidad. Esto nos ha servido para comprender mejor todo lo que hacemos, además de conseguir muy buenos resultados.

El mayor problema que hemos tenido ha sido el elevado tiempo de entrenamiento que hemos necesitado para las últimas aproximaciones que, aunque tenga sentido ya que tenemos un número considerable de patrones, no deja de ser molesto para poder realizar experimentos. Nos limita, por ejemplo, en el hecho de no poder explorar muchas variantes de cada configuración, si, para cada una de ellas, tenemos que dedicarle 1h o más de entrenamiento (con *cross validation*). Además, como ya se ha mencionado, este tiempo de entrenamiento es aún más elevado si utilizamos técnicas de *deep learning*. Este problema lo hemos intentado paliar a lo largo de la quinta aproximación (Sección 5.5), reduciendo el número de características del sistema utilizando algoritmos de selección basados en información mútua. Sin embargo, aunque esto reduce la complejidad y con ella el tiempo de entrenamiento, creemos que disponiendo de más tiempo y recursos computacionales, dicho problema podría llegar a paliarse, aunque seguiría siendo el punto débil del sistema.

Pero no todo son problemas; estamos muy contentos con los resultados que hemos obtenido ante un problema que, aunque parezca sencillo, la posibilidad de ruido en los patrones puede hacer que sea complicado ofrecer predicciones acertadas. En especial, destacamos los sistemas basados en máquinas de soporte vectorial o *SVM* ya que, a la par que tienen un tiempo de entrenamiento mucho menor que otros sistemas, nos han ofrecido unos niveles de precisión muy elevados. También, hemos comprobado como, a medida que la complejidad del sistema iba aumentando, es decir, cuanto mayor es el número de clases, los modelos basados en árboles de decisión obtienen unos peores resultados. Los modelos basados en redes neuronales y en *kNN* ofrecen, en cambio, unos valores más constantes a lo largo de todas las iteraciones, aunque son las primeras las que alcanzan cotas de precisión más elevadas. Cabe destacar también que las *RNAs* son los sistemas que tardaban más tiempo con muchísima diferencia, por lo que, como ya repetimos varias veces en secciones anteriores, las hacen menos atractivas que otros sistemas como los *SVM*, sobre todo si no se cuenta con equipos lo suficientemente potentes (como es nuestro caso). Las redes convolucionales no muestran unos buenos resultados en general, aunque hay alguna en concreto que sí que funciona bastante bien, además que hay que tener en cuenta que en ellas utilizamos muestras de 0,1 segundos de los audios, por lo que en realidad estaban afrontando un problema más

complejo que el de los sistemas clásicos que utilizamos en aproximaciones anteriores. Entonces, pensamos que con una mayor experimentación de arquitecturas con estas redes convolucionales, podríamos obtener muy buenos resultados.

En definitiva, todo esto nos ha servido para comprender mucho mejor algunos modelos de aprendizaje automático, además de cómo se pueden aplicar dichos modelos de forma práctica en situaciones reales.

7. Trabajo futuro

Tras los buenos resultados de nuestro sistema, podemos concluír que podría ser utilizado en múltiples herramienteas como son las aplicaciones de afinación en dispostivos móviles o las aplicaciones de aprendizaje elemental para los más novatos que quieran, por ejemplo, practicar su oído musical.

También sería interesante avanzar de cara a poder producir cierta explicabilidad de las predicciones del sistema, ya que sería beneficioso para el aprendizaje del usuario el poder saber por qué cierto sonido se asocia a una nota en concreto.

Asimismo, también sería conveniente trabajar más sobre el tiempo de entrenamiento del sistema, ya que, por ejemplo, con modelos como pueden ser las redes convolucionales, el tiempo de entrenamiento es excesivo. Sería un gran aporte lograr una alta precisión con un sistema que sea energéticamente más eficiente y respetuoso con el medio ambiente, además de permitir que la utilización de estos modelos no requiera acceso a grandes recursos ni de tiempo ni computacionales para que, así, cualquier persona en su ordenador personal pueda entrenar, ejecutar el sistema y obtener resultados igualmente buenos.

Otra alternativa para abordar el problema del excesivo tiempo de entrenamiento y así poder utilizar modelos más complejos, sería el contar con equipos más potentes que nos ofreciesen una mayor capacadidad de cálculo.

Por otra parte, tomando como ejemplo trabajos como los de Klapuri, 2004, nuestro sistema podría ser una herramienta práctica a la hora de abordar problemas más complejos como puede ser la difícil tarea de la transcripción automática de música en tiempo real, ya que hemos visto que, ante muestras de muy baja duración (0,1s), nuestro sistema ofrece igualmente una muy buena precisión.

También cabe destacar que, teniendo en cuenta la eficacia de nuestro sistema y tal como dijimos en la introducción, este podría ser ampliado para ser capaz de clasificar varias notas que sean tocadas al mismo tiempo, o incluso ser capaz de reconocer acordes o tonalidades musicales. Esto sería un problema *Multi-label*, y tendría una mayor complejidad que el problema que hemos abordado nosotros.

La metodología que seguimos a lo largo de esta memoria también podría estar enfocada para otros instrumentos, dado que las características que usamos en los diferentes

modelos no son únicas para el piano y tienen la misma naturaleza en otros instrumentos como una guitarra, un bajo o un arpa.

En definitiva, las múltiples aplicaciones que tiene este sistema en diferentes tareas dentro del ámbito musical, lo hacen indiscutiblemente útil e interesante, tanto desde un marco científico como comercial. Aplicaciones como Shazam, Spotify, Tidal o YouTube podrían darle distintos usos, como el reconocimiento de canciones, *covers* o clasificaciones por géneros musicales, usando, adaptando o integrando nuestro sistema dentro de sus modelos de negocio.

8. Bibliografía

Referencias

Baevski, A., Hsu, W.-N., Conneau, A., & Auli, M. (2021). Unsupervised speech recognition. *Advances in Neural Information Processing Systems*, *34*.

Benetos, E., Dixon, S., Duan, Z., & Ewert, S. (2018). Automatic music transcription: An overview. *IEEE Signal Processing Magazine*, *36*(1), 20-30.

Chang, S., Lee, J., Choe, S. K., & Lee, K. (2017). Audio cover song identification using convolutional neural network. *arXiv preprint arXiv:1712.00166*.

Cortes, C., & Vapnik, V. (1995). Support-vector networks. *Machine learning*, *20*(3), 273-297.

Ding, C., & Peng, H. (2005). Peng, H.: Minimum Redundancy Feature Selection from Microarray Gene Expression Data. Journal of Bioinformatics and Computational Biology 3(2), 185-205. *Journal of bioinformatics and computational biology*, *3*, 185-205. https://doi.org/10.1142/S0219720005001004

Foo, S. W., & Wong, P. L. (1999). Recognition of piano notes. *IEEE International Conference on Information, Communications and Signal Processing (1999: Singapore)*.

Guo, G., Wang, H., Bell, D., Bi, Y., & Greer, K. (2003). KNN model-based approach in classification. *OTM Confederated International Conferences."On the Move to Meaningful Internet Systems"*, 986-996.

Hermo González, J. (2022). mRMR-enhanced. https://github.com/jorgehermo9/mrmr-enhanced

Indolia, S., Goswami, A. K., Mishra, S., & Asopa, P. (2018). Conceptual Understanding of Convolutional Neural Network- A Deep Learning Approach [International Conference on Computational Intelligence and Data Science]. *Procedia Computer Science*, *132*, 679-688. https://doi.org/https://doi.org/10.1016/j.procs.2018.05.069

Klapuri, A. (2004). *Signal processing methods for the automatic transcription of music*. Tampere University of Technology Finland.

Krogh, A. (2008). What are artificial neural networks? *Nature biotechnology*, *26*(2), 195-197.

Marolt, M., Kavcic, A., & Privosnik, M. (2002). Neural networks for note onset detection in piano music. *Proceedings of the 2002 International Computer Music Conference*.

Myles, A. J., Feudale, R. N., Liu, Y., Woody, N. A., & Brown, S. D. (2004). An introduction to decision tree modeling. *Journal of Chemometrics: A Journal of the Chemometrics Society*, *18*(6), 275-285.

Osmalsky, J., Embrechts, J.-J., Van Droogenbroeck, M., & Pierard, S. (2012). Neural networks for musical chords recognition. *Journées d'informatique musicale*, 39-46.

Solanki, A., & Pandey, S. (2019). Music instrument recognition using deep convolutional neural networks. *International Journal of Information Technology*, 1-10.