UNIVERSIDADE DA CORUÑA

APRENDIZAJE AUTOMÁTICO 23/24

Identificación de notas musicales

Autores:
Rafael de Almeida Leite
David Fernández Mato
Dario Gjergo
José Luis López Portelinha
Pablo Pajón Area

Fecha: *A Coruña, 5 de mayo de 2024*

Índice

Capítulos **Página**

1. Introducción

La música forma parte de la vida cotidiana de cualquier persona. Desde los instrumentos paleolíticos, hasta la aparición del *streaming*, pasando por las composiciones griegas [Wik24f] o los ilustres compositores del Clasicismo [Wik24h], el impacto histórico y la inherencia de la música al ser humano es ciertamente innegable. La industria musical es un sector en constante crecimiento [IFP23], con un alcance global y del que forman parte miles de personas, por lo que se trata de un campo de estudio perfecto para nuestro trabajo, que se encuentra en la intersección entre la tecnología y el arte.

La música se ha postulado como una figura con gran peso, que ha moldeado la sociedad y la cultura a lo largo de los siglos. Desde los primeros registros de sonidos hasta la era digital actual, ha sido una expresión fundamental de la humanidad. Sin embargo, en la era de la información y la tecnología, el análisis y la comprensión de la música han evolucionado drásticamente. La posibilidad de reconocer y comprender las notas musicales a partir de las ondas sonoras ha desencadenado un gran cambio revolucionario en la forma en que interactuamos con la música y su industria.

El reconocimiento de notas musicales a partir de archivos de audio es un problema complejo que ha recibido considerable atención en los campos de la ingeniería de señales, la música computacional y el aprendizaje automático. En esencia, implica la capacidad de analizar una señal de audio, extraer características relevantes y asociarlas con notas musicales específicas.

El proceso comienza con la captura de la señal de audio, que puede contener una variedad de fuentes sonoras, desde instrumentos individuales hasta grabaciones de conjuntos completos. En el caso de archivos de audio en formato *.wav*, se conserva una representación digital de la señal de audio, lo que proporciona una base sólida para el análisis posterior.

El siguiente paso es el pre-procesamiento de la señal, que puede incluir la normalización de la amplitud, la eliminación de ruido y la segmentación en tramos más pequeños, para un análisis más detallado. Una vez que la señal ha sido preparada, se procede a extraer las características relevantes, como la frecuencia fundamental, la intensidad o la duración de las notas.

En el corazón de este proceso se encuentra la relación entre las características extraídas y las notas musicales correspondientes. Esta relación puede modelarse mediante técnicas de aprendizaje automático, como las redes neuronales, que son capaces de aprender patrones complejos en los datos y generar predicciones precisas una vez entrenadas adecuadamente.

Sin embargo, el reconocimiento preciso de notas musicales presenta varios desafíos. Por un lado, la presencia de ruido en la señal de audio puede interferir con la detección de las notas. Además, la variabilidad en la interpretación musical, incluidas las variaciones en la entonación y el timbre, puede dificultar la asociación exacta entre las características del sonido y las notas musicales.

A pesar de estos desafíos, el desarrollo de sistemas de reconocimiento de notas musicales tiene importantes aplicaciones prácticas. Por ejemplo, en el ámbito

educativo, estos sistemas pueden utilizarse como herramientas de aprendizaje para estudiantes de música, ayudándoles a desarrollar su habilidad auditiva y mejorar su comprensión de la teoría musical. Además, en el contexto de la producción musical, estas tecnologías pueden agilizar el proceso de composición y grabación, permitiendo a los músicos capturar y explorar ideas musicales de manera más eficiente.

Por todo esto, el reconocimiento de notas musicales a partir de archivos de audio es un campo emocionante, que une elementos de la ingeniería de señales con el procesamiento de datos y la teoría musical.

2. Descripción del problema

El desafío inicial en este proyecto radica en el desarrollo de un sistema inteligente capaz de discernir entre las distintas notas que puede producir un instrumento musical de cuerda o de viento. Esta elección se fundamenta en las características de estos instrumentos, cuyas notas tienden a tener una mayor duración en comparación con otras familias de instrumentos, como los de percusión, lo que facilita su análisis y clasificación.

La base de datos utilizada para este propósito se obtiene de un trabajo conjunto de miembros de las universidades norteamericanas de California, Nueva York, Columbia y Boston, además del Conservatorio de París [Car20], y que nos proporciona un conjunto de casi tres mil audios en formato *.wav* con cada una de las notas de catorce instrumentos distintos. Para nuestro trabajo, centraremos nuestra atención en cuatro de ellos.

Los archivos utilizados tienen duraciones de entre cuatro y diez segundos, y cada uno captura el sonido grabado de una nota correspondiente a un violín, una viola, un violonchelo o un contrabajo. Además de los archivos *.wav*, se incluyen también una serie de archivos *.csv* que detallan el nombre y la duración de cada nota con respecto a la muestra original. Es importante destacar que estos archivos *.csv* son generados internamente, partiendo del conocimiento previo de que todos las segmentos de un audio corresponden a una única nota musical. En lo que respecta a las características comunes concretas que comparten todos los archivos de audio, cabe recalcar que todos han sido grabados en el mismo ambiente, utilizando el mismo instrumento y el mismo micrófono; es importante tener en cuenta estas limitaciones a la hora de enfrentarse al problema.

Con el objetivo de simplificar el análisis y garantizar la coherencia de los datos, nos centramos exclusivamente en los archivos de sonido *.wav* con una frecuencia de muestreo de 44100Hz. Además, para facilitar la extracción de características relevantes, dividimos cada archivo de audio en ventanas de 1.486 segundos de duración, pues representa una ventana con un solapamiento del 25 % de la duración total. Durante este proceso, trabajamos con la amplitud de cada muestra, la cual, junto con la frecuencia de muestreo, será utilizada en la Transformada Rápida de Fourier (FFT) [Wik24g] para extraer la media y la mediana de la señal en frecuencia entre varias frecuencias determinadas. Estos valores serán utilizados para evaluar y vali-

dar los clasificadores diseñados para este propósito, permitiéndonos así determinar la eficacia y precisión del sistema en la tarea de reconocimiento de notas musicales.

La métrica que utilizaremos para comparar modelos y configuraciones será la métrica de precisión, ya que nos proporciona el número de predicciones correctas realizadas por el modelo sobre el total de predicciones realizadas. Si bien todas las métricas tienen su importancia, mayor o menor, creemos que la precisión es la métrica más significativa de todas, ya que es la variable más representativa del funcionamiento del modelo, y es fácilmente interpretable y comparable.

Además, consideramos interesante incluir en nuestras figuras la métrica de *F1-score*, que es la media harmónica entre sensibilidad (probabilidad de clasificar como positivos los elementos positivos) y especificidad (probabilidad de clasificar como negativos los elementos negativos), y que se utiliza habitualmente para problemas de clasificación con distribuciones de clases desbalanceadas [Jas21].

3. Análisis bibliográfico

Tras la búsqueda y análisis de múltiples trabajos sobre el tema que nos atañe, hemos encontrado similitudes con el nuestro en algunos de ellos, aunque gran parte de ellos utilizan el piano como instrumento principal. La publicación que presenta Sayali Gadre [Say22] es especialmente significativa, ya que resume a la perfección el método seguido para el procesamiento de señales digitales y la extracción de sus notas musicales, mediante la división de las canciones en ventanas, analizando cada una de ellas por separado, y la aplicación individual de la transformada rápida de Fourier (o FFT, por sus siglas en inglés) [Wik24g]. Una explicación de las FFT, además de su aplicación y uso en la automatización de la transcripción de notas musicales, puede encontrarse en el documento presentado por Mohamed Karioun y Simon Tihon [Moh18].

El uso de las FFT es definitivamente popular para la resolución de la tarea, dada la gran cantidad de trabajos que hacen referencia a su uso, con ejemplos tan recientes como los documentos presentados por Leo Prasanth [Leo23] y Kelvin A. Minor [Kel22], con menos de dos años de antigüedad.

Después, son muchos los trabajos que apuestan por el uso de Python para implementar redes neuronales con un nivel alto de abstracción, como la tesis presentada por Manuel Mínguez Carretero [Man18], que hace uso de *Keras*, una API de Python que ahorra al usuario las complejas operaciones tras una red neuronal. Aproximaciones similares pueden observarse en trabajos como los pertenecientes a Chiheb Trabelsi y Olexa Bilaniuk [Chi18], o a Bhuwan Bhattarai y Joonwhoan Lee [Bhu23], en el que se mencionan los usos de librerías como *Librosa*. En este trabajo, de nuevo, los autores destacan el uso de la FFT y la consideran necesaria, ya que "juega un papel crucial en la transcripción de música mediante análisis de señales", como ya hemos mencionado previamente.

Por otro lado, y con un método completamente diferente al nuestro, se encuentran documentos como el perteneciente a Potcharapol Suteparuk [Pot06], en el que la

aproximación a la detección de notas se realiza mediante el análisis de grabaciones en vídeo, con un resultado mucho menos preciso que en cualquiera de los ejemplos anteriores. El autor menciona varios problemas relacionados con la naturaleza física del experimento, un factor limitante que no se encuentra presente en nuestro método de transcripción de notas.

Dados estos precedentes, podemos presumir la viabilidad técnica de desarrollar una solución al problema, aunque cabe resaltar que una parte sustancial de la literatura sobre problemas similares recibe *addendums* de manera habitual, por lo que se presume la constante evolución de las soluciones planteadas a este problema, y la dificultad de conseguir un resultado de la más alta calidad. Además, nosotros haremos uso del lenguaje de programación Julia, pensado específicamente para ciencia de datos y aprendizaje automático, por lo que bate en todas las métricas de rendimiento y velocidad relevantes a Python, que es el lenguaje más común de los trabajos expuestos en estas líneas.

4. Desarrollo

4.1. Aproximación 1

4.1.1. Descripción

Para comenzar, se han decidido realizar varias aproximaciones del problema. La primera aproximación acota el problema hasta su forma más sencilla, que es distinguir entre dos notas claramente diferentes. Con este objetivo, se han seleccionado dos notas como prueba, A4 y C5, por lo que tenemos un total de dos estados en este caso.

Para empezar, se genera un *input* que consiste en dividir los archivos de audio en ventanas con solapamiento, como se ha mencionado anteriormente. Esto se hace para aumentar el número de muestras dividiendo el archivo, poniendo en las ventanas una duración de potencia de dos muestras, y aumentando el número de datos mediante el solapamiento, que es especialmente importante en estas etapas iniciales del problema, donde no hay un gran volumen de datos. Teniendo en cuenta esto, se obtienen un total de 176 y 171 patrones para sus respectivas clases.

Una vez se ha dividido el archivo en ventanas para facilitar la interpretación de los datos, se utiliza la Transformada de Fourier Rápida, o FFT [Wik24g], motivo por el cual las muestras de las ventanas tienen duración de potencia de dos muestras, debido al funcionamiento interno de la misma. La FFT se utiliza para obtener, de todas las ventanas, la media y la desviación típica de la señal en todas las frecuencias.

Una vez tenemos esto, ya tenemos el *input* de nuestra primera aproximación, que son esos 2 valores por cada una de las ventanas. En lo que respecta al *target*, se ha decidido que, dadas las características de este problema, lo mejor será hacer un problema de clasificación, en el que cada categoría es una nota. Con este fin, hemos decidido hacer un *target* manualmente, en el cual partiendo del hecho de que cada nota se almacena en una carpeta separada se itera sobre las carpetas de notas y en

una matriz de ceros y unos se pone a 1.

Una vez llegados a este punto, entrenaremos cuatro tipos de modelos: redes de neuronas artificiales (por sus siglas, RNA), máquinas de vectores de soporte (SVM), árboles de decisión y vecinos más cercanos (kNN).

En el caso de las RNA, el siguiente paso es definir una red neuronal con los *inputs* y *targets* anteriormente mencionados divididos en tres conjuntos seleccionados aleatoriamente: test (10 %), validación (20 %) y entrenamiento (el restante 70 %). Esto se hace para comprobar cómo funciona el modelo con nuevos datos. En lo que al entrenamiento respecta, como se ha mencionado antes, utilizaremos una capa de entrada con dos *inputs*, dos capas intermedias de 4 y 3 neuronas respectivamente, y una de salida con otras dos, que utiliza la función *softmax* [Wik24i] para que dé salidas correspondientes al dominio de nuestro problema. Debido a la naturaleza del problema de clasificación, y teniendo en cuenta que se desea expandir a más notas, se utilizará la función de error *cross-entropy* [Wik24d], para medir la discrepancia entre las predicciones del modelo y los valores reales. Cabe recalcar que pese a no ser esta la forma más óptima de clasificar dos clases, se hace así con previsión a un aumento de las mismas.

Como es una RNA, el entrenamiento se debe realizar varias veces sobre el conjunto de datos de entrenamiento, en nuestro caso ponemos dos posibles condiciones: llegar a un error *cross-entropy* [Wik24d] de 0.5 o que itere más de 120 (esta última suele ser la que finaliza el entrenamiento). Para finalizar, se comprobarán los datos del conjunto de test para ver el éxito del entrenamiento, y con este fin, por cada ventana seleccionamos el valor más alto y lo ponemos a 1, mientras que el resto serán 0, para comparar (*output* y *target* con mismo formato, para poder compararlos). Además, debido a que una clase es la inversa de la otra, solo se mandará el valor de una clase de resultados y *target* por simplicidad, aprovechando que no se pierde información.

En lo que respecta a SVM, árboles de decisión (basados en el algoritmo *CART* [Wik24b]) y kNN, se ha decidido dividir en dos conjuntos también seleccionados aleatoriamente, test (10 %) y entrenamiento (el restante 90 %), para así, como en el caso anterior, poder comprobar cómo reacciona el modelo ante nuevos datos.

Con el fin de transformar los datos para los modelos, se cambia la matriz anteriormente descrita de ceros y unos por un array, que tiene en cada valor la posición de la matriz en la que estaría el uno, ya que es necesario para este tipo de modelos. La configuración de dichos modelos se muestra en las tablas del siguiente apartado, justo encima de los resultados de cada uno de ellos, visualizando así de manera conjunta parámetros y resultados, y facilitando su interpretación.

Para finalizar, se vuelve a convertir los *targets* y resultados al formato previo semejante al de RNA, se obtienen los valores resultantes de comparar la salida y el *target* del conjunto test, y se calcula su matriz de confusión [Wik24c] y la precisión, entre otros valores de menor importancia. Igual que con las RNA, solo se mandará el valor de una clase de resultados y *target* por simplicidad aprovechando que no se pierde información.

Destacar que en esta primera etapa de la aproximación hemos decidido no im-

plementar el *cross validation*, para comprobar en un principio la correcta implementación del resto de los procesos.

4.1.2. Resultados

Los siguientes conjuntos representan una ejecución promedio del código con los valores de test, en la que los resultados son una representación aproximada de todos los valores obtenidos.

	RNA 1	RNA 2	RNA 3	RNA 4
Capa oculta	1	2	3	4
Matriz de confusión	11 6 6 10	12 6 9 7	9 14 2 13	14 1 20 1
Precisión	0.64	0.56	0.58	0.42
F1-score	0.63	0.48	0.62	0.09
Error RNA (cross-entropy)	0.65	0.68	0.68	0.70

	RNA 5	RNA 6	RNA 7	RNA 8
Capa oculta 1	2	3	4	4
Capa oculta 2	1	2	2	3
Matriz de confusión	16 0 18 2	8 6 2 14	9 12 7 10	11 10 6 14
Precisión	0.50	0.73	0.50	0.61
F1-score	0.18	0.78	0.51	0.64
Error RNA (cross-entropy)	0.69	0.65	0.72	0.71

Cuadro 1: Resultados de las arquitecturas de RNA para la aprox. 1.

Vemos los resultados de RNA (cuadro 1) para una capa oculta de una, dos, tres y cuatro neuronas respectivamente y, a continuación, con dos capas de dos/una, tres/dos, cuatro/dos y, por último, cuatro/tres neuronas. Mostramos las matrices de confusión de cada arquitectura, y debajo se enseñan los valores de las métricas de precisión y *F1-score*. Podemos observar valores de entre 0.42 y 0.73, que se corresponden con arquitecturas de una capa oculta (4) y dos capas ocultas (3 y 2). Sin embargo, no podemos concluir que tener dos capas asegure una mejor precisión. El F1-score también es extremadamente irregular, con valores que fluctúan en un rango muy amplio.

Además de las métricas mencionadas, también incluimos el error *cross-entropy*, que estará presente en todos los cuadros relativos a RNA. La evolución de dicho error durante el entrenamiento de RNA ha sido aceptable, no percibimos datos sospechosos. Podemos verlo en la figura 1.

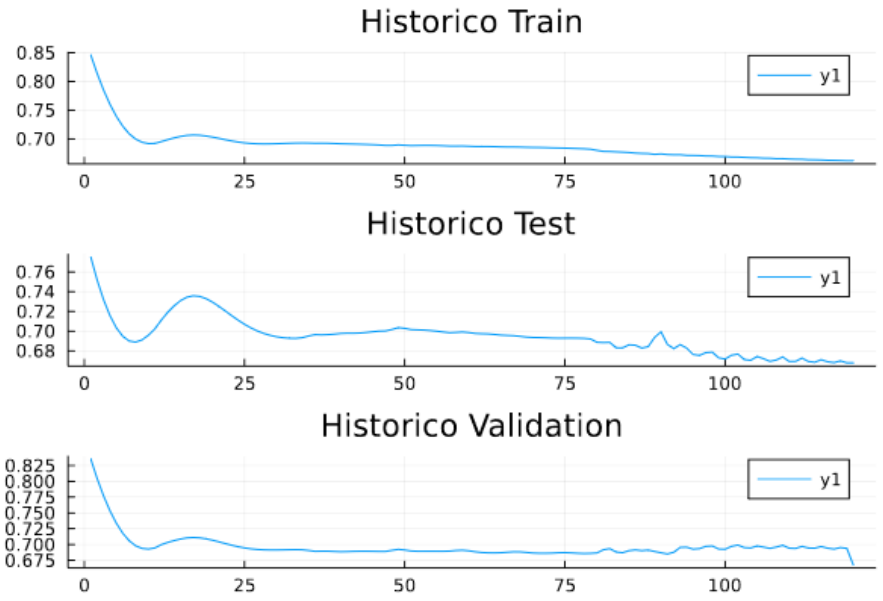

Figura 1: Gráfico del histórico de entrenamiento de RNA en la aprox. 1.

	SVM 1	SVM 2	SVM 3	SVM 4	SVM 5	SVM 6	SVM 7	SVM 8
Kernel	sigmoid	rbf	linear	sigmoid	rbf	linear	linear	rbf
Grado del kernel	1	1	1	4	4	4	2	2
Gamma del kernel	1	1	1	2	2	2	3	3
C	3	3	3	3	3	3	1	1
Matriz de confusión	1 25 0 16	8 8 11 14	10 6 6 7	1 16 1 15	9 8 3 6	13 2 4 7	11 11 4 5	7 5 10 9
Precisión	0.40	0.54	0.59	0.48	0.58	0.77	0.52	0.52
F1-score	0.56	0.60	0.54	0.64	0.52	0.70	0.40	0.55

Cuadro 2: Resultados de las arquitecturas de SVM para la aprox. 1.

Para SVM (cuadro 2), tenemos una comparativa entre los modelos con kernels sigmoidales, rbf o lineales, con variaciones en los valores de gamma y C (grado uno, gamma uno y C tres; grado cuatro, gamma dos y C tres; grado dos, gamma tres y C uno). Incluimos también las matrices de confusión. Observamos una serie de valores bastante irregulares entre las arquitecturas, si bien la 7 (lineal con valores 4, 2 y 3) destaca por sus buenos resultados.

	Á. d. 1	Á. d. 2	Á. d. 3	Á. d. 4	Á. d. 5	Á. d. 6
Profundidad máxima	1	2	3	4	5	6
Matriz de confusión	8 12 5 9	9 11 3 12	5 10 4 15	4 9 5 14	9 10 6 15	6 9 7 13
Precisión	0.50	0.61	0.59	0.56	0.60	0.54
F1-score	0.51	0.67	0.68	0.67	0.65	0.62

Cuadro 3: Resultados de las arquitecturas de árbol de decisión para la aprox. 1.

Tenemos los resultados para las métricas de los árboles de decisión (cuadro 3), junto a las matrices de confusión para cada una de ellas. La profundidad máxima de los árboles aumenta de uno a seis, aunque las métricas de precisión y F1-score parecen ser resistentes a este hecho, y no cambian significativamente.

	kNN 1	kNN 2	kNN 3	kNN 4	kNN 5	kNN 6
Número de vecinos	1	2	3	4	5	6
Matriz de confusión	5 3 12 12	3 6 6 16	12 5 8 10	9 10 3 12	12 9 5 13	6 7 2 12
Precisión	0.53	0.61	0.63	0.62	0.64	0.67
F1-score	0.61	0.72	0.61	0.65	0.65	0.73

Cuadro 4: Resultados de las arquitecturas de kNN para la aprox. 1.

Muestra de los resultados para las arquitecturas del modelo de kNN (cuadro 4), y en la que el número de vecinos más cercanos de las arquitecturas aumenta de uno a seis. Se muestran precisión y *F1-score*, al igual que en las tablas anteriores, y las matrices de confusión, que abordaremos solo en esta aproximación (se explicará más adelante por qué). En este caso, notamos una ligera tendencia ascendente de los resultados de las métricas a medida que aumenta el número de vecinos.

4.1.3. Discusión

Como hemos mencionado en apartados previos, consideramos que la precisión es la métrica más significativa y, por tanto, la que tendremos más en cuenta para extraer conclusiones. Mediante la observación de los datos expuestos sobre estas líneas y los resultados obtenidos, podemos constatar que, potencialmente, el mejor modelo para nuestro caso es el SVM, con una precisión bastante alta, llegando a máximos de 0.80. El F1-score de 0.70 acompaña la apreciación positiva de esta arquitectura.

A pesar de que también es la más inestable en lo que a resultados se refiere, la arquitectura RNA también presenta resultados destacables. Como comentábamos previamente, las características inherentes a las redes neuronales son causa de la alta variación de los valores, que cambian a medida que se realizan ejecuciones del código. Sin embargo, las sucesivas ejecuciones arrojan datos aceptables.

Es menos ideal el caso del algoritmo kNN, que cuenta con una precisión más baja, aunque su valor F1 es positivo y se encuentra en línea con el resto de arquitecturas.

Por último, el método de los árboles de decisión cuenta con valores aceptables en sus métricas, si bien podría necesitar más refinamiento en todos sus campos. Se trata de la arquitectura con peores estadísticas, aunque no por ello son malos resultados.

En el cuadro 5 podemos ver los resultados de la mejor versión de cada una de las cuatro arquitecturas consideradas para el presente trabajo:

	RNA	SVM	Á. d.	kNN
Precisión	0.73	0.77	0.61	0.67
F1-score	0.78	0.70	0.67	0.73

Cuadro 5: Comparación entre los mejores resultados de las métricas de la aprox. 1.

Con todo esto, creemos que partimos de una buena base para desarrollar el resto de aproximaciones de nuestro proyecto. Los resultados han sido aceptables de manera general, y no se detectan fallos en la implementación del código. Estimamos que el progreso realizado es positivo, aunque tal vez los datos podrían ser mejorables en algunos casos. Por lo tanto, se considera conveniente ampliar la segunda aproximación, y poder discernir entre más notas a la vez.

4.2. Aproximación 2

4.2.1. Descripción

Una vez hemos establecido una primera aproximación sencilla, hemos decidido expandir el dominio de nuestro problema y qué datos tiene en cuenta. Para empezar, cambiamos la forma en la que se calculan los *inputs*. Anteriormente se calculaban mirando en la *FFT* [Wik24g] la media y la desviación estándar de toda la banda de frecuencias, dando resultados subóptimos a la hora de distinguir notas. Por lo tanto, teniendo en cuenta el ampliado dominio del problema, estos valores, en lugar de ser para toda la banda de frecuencias, se especifican para la frecuencia de cada nota a identificar $\pm 5\,\%$. Esto hace que nuestro *valores por ventana* contenga estos dos valores por cada nota y así, si hay 3 notas, tendrán la media y la desviación estándar de esas 3 notas, dando 6 *valores por ventana*. Esto se hace con el objetivo de aprovechar la clara relación que hay entre una y el valor que devuelve esa nota para la *FFT*.

Por lo tanto, visto que en teoría nuestra nuestra precisión debería aumentar tras aprovechar la relación anteriormente nombrada, hemos decidido aumentar el dominio de nuestra aproximación de 2 notas a un cuarto de las notas, para todos los

instrumentos disponibles. Así, de las 22 notas que contienen estos datos, se obtendrán 44 *valores por ventana* y 22 *clases* para su *target*, ya que seguimos entendiendo el problema como un problema de clasificación en el que cada clase sigue siendo una nota. Como ya se ha programado la anterior aproximación como una multiclase en su mayoría, apenas se han tenido que hacer modificaciones, más allá de cómo se calculan los datos y cómo presentar y calcular la matriz de confusión [Wik24c], aunque debido a su volumen no se mostrará más en el apartado de resultados. Ya que hemos expandido nuestros datos, decidimos implementar la desviación típica a los datos calculados sobre los resultados. Además, como ya hemos completado una primera instancia del código en la primera aproximación, implementamos ahora la *cross-validation*, para obtener una representación de los datos más fidedigna; esta aproximación será ejecutada con un *k-fold* de 10. A mayores, creamos un diagrama de caja [Wik24a] para facilitar la comprensión de los resultados. Cabe destacar que, ahora que hemos expandido el número de datos nuestra base de datos, no está nivelada, aunque ahora aumentaremos las ventanas sobre las que entrenamos a valor. La distribución de valores queda reflejada en la figura 2:

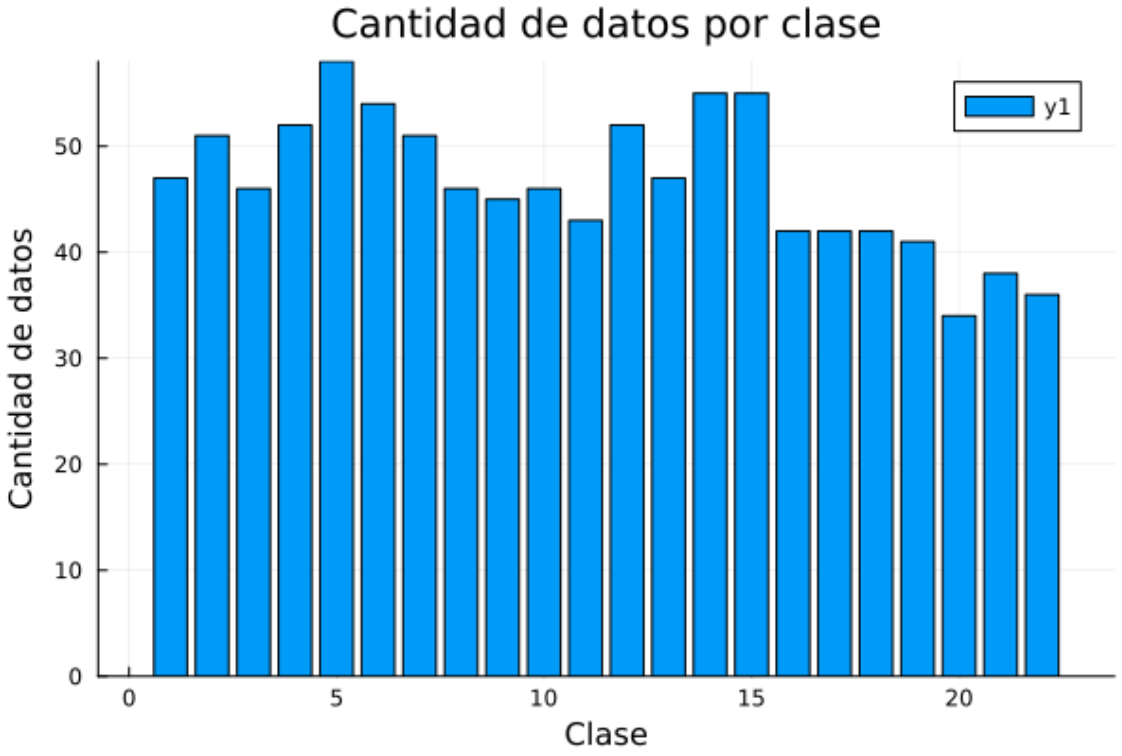

Figura 2: Distribución de valores del dataset en la aprox. 2. por clase

Como hemos mencionado anteriormente, utilizamos la FFT para analizar la similitud de la onda de una nota con una frecuencia dada, y devolverá un valor mayor cuanto mayor sea la similitud. Entonces, cogemos un margen de $\pm 5\,\%$ sobre la frecuencia de la nota, y dado ese conjunto de valores calculamos su FFT. Del conjunto de valores resultante, calculamos la media y desviación típica.

Por ejemplo, en la figura 3, podemos observar que la frecuencia de la nota de entrada es de 1000 Hz. Por tanto, cogemos el margen previamente dicho, calculamos la FFT sobre estos valores del margen (950 - 1050 Hz), y con ellos obtenemos la media y desviación típica de este conjunto. Este proceso se repite por cada nota que puede identificar nuestro sistema en cada ventana. Así, el número de valores de cada ventana será el número de notas que nuestro sistema puede clasificar, multiplicado por 2; esto se corresponde con la media y la desviación típica de cada nota.

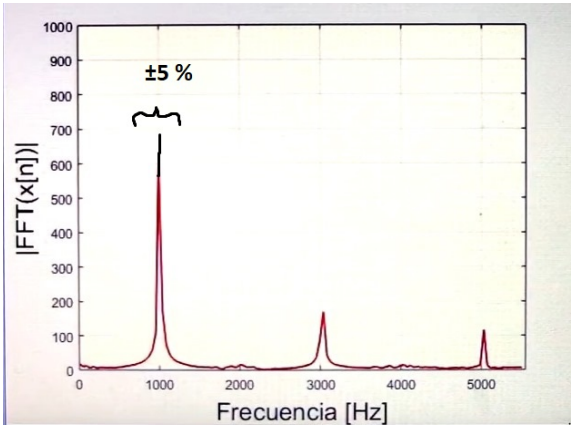

Figura 3: Valores de la FFT para la onda de una nota musical.

4.2.2. Resultados

Los siguientes conjuntos representan una ejecución promedio del código con los valores de test, en la que los resultados son una representación aproximada de todos los valores obtenidos.

Como novedad, y de aquí en adelante, incluiremos en todas las tablas la métrica de la desviación típica de la precisión, medida que utilizaremos para cuantificar la dispersión del conjunto de valores de esta métrica que retorne la ejecución continuada de nuestro código.

	RNA 1	RNA 2	RNA 3	RNA 4	RNA 5	RNA 6	RNA 7	RNA 8
Capa oculta 1	4	8	16	64	50	64	120	150
Capa oculta 2	-	-	-	-	20	32	110	120
Precisión	0.50	0.92	0.98	0.99	0.98	0.98	0.98	0.99
F1-score	0.48	0.92	0.98	0.99	0.99	0.99	0.98	0.99
Desv. típ. precisión	0.21	0.09	0.04	0.03	0.04	0.03	0.04	0.03
Error RNA (cross-entropy)	1.31	0.39	0.11	0.09	0.10	0.10	0.11	0.08

Cuadro 6: Resultados de las arquitecturas de RNA.

Vemos los resultados de RNA (cuadro 6) para una capa oculta de cuatro, ocho, dieciséis y sesenta y cuatro neuronas respectivamente y, a continuación, con dos capas de 50/20, 64/32, 120/110 y, por último, 150/120 neuronas. Se enseñan los valores de las métricas de precisión y *F1-score*, desviación típica de la precisión, y el

error cross-entropy. Valores mayores de neuronas en las capas, tanto con una como con dos, parecen arrojar mejores resultados de las métricas relevantes, con máximos de hasta 0.99 en ambas. Las desviaciones son bajas en casi todos los casos.

La evolución de nuestro error *cross-entropy* durante el entrenamiento de RNA ha sido buena, y no se perciben datos fuera de lo normal. Podemos verlo en la figura 3.

Figura 4: Gráfico del histórico de entrenamiento de RNA para la aprox. 2.

	SVM 1	SVM 2	SVM 3	SVM 4	SVM 5	SVM 6	SVM 7	SVM 8
Kernel	rbf	rbf	sigmoid	sigmoid	poly	poly	linear	linear
Grado del kernel	3	3	20	5	3	4	20	60
Gamma del kernel	8	2	7	3	2	3	15	9
C	7	1	4	2	1	5	9	7
Precisión	0.02	0.03	0.02	0.03	0.96	0.96	0.98	0.99
F1-score	0.01	0.01	0.00	0.01	0.96	0.96	0.98	0.99
Desv. típ. precisión	0.30	0.30	0.30	0.30	0.06	0.06	0.04	0.03

Cuadro 7: Resultados de las arquitecturas de SVM para la aprox. 2.

Se trata de una tabla comparativa entre los modelos SVM (cuadro 7) con kernels sigmoidales, rbf, polinomiales o lineales, con variaciones en los valores de gamma y C. También en este caso, precisión, su desviación típica y el *F1-score*. Las arquitecturas lineales y polinoniales albergan unos resultados muy superiores a las del resto, con valores que superan precisiones de 0.90 en todos los casos, teniendo en cuenta su desviación típica.

	Á. d. 1	Á. d. 2	Á. d. 3	Á. d. 4	Á. d. 5	Á. d. 6
Profundidad máxima	2	10	20	50	100	120
Precisión	0.14	0.83	0.95	0.95	0.94	0.94
F1-score	0.13	0.84	0.95	0.95	0.94	0.94
Desv. típ. precisión	0.28	0.12	0.07	0.07	0.07	0.07

Cuadro 8: Resultados de las arquitecturas de árbol de decisión para la aprox. 2.

Tenemos los resultados para las métricas de los árboles de decisión (cuadro 8), junto a las matrices de confusión para cada una de ellas. La profundidad máxima de los árboles va de dos a diez, veinte, cincuenta, cien y, al final, ciento veinte. Parece que una mayor profundidad máxima se traduce en mejores resultados para todas las métricas, con máximos de 0.95 de precisión y *F1-score* a partir de una profundidad de 20 (profundidades menores consiguen resultados más pobres).

	kNN 1	kNN 2	kNN 3	kNN 4	kNN 5	kNN 6
Número de vecinos	1	2	20	80	200	1000
Precisión	0.98	0.98	0.94	0.89	0.80	0.21
F1-score	0.98	0.98	0.94	0.89	0.80	0.17
Desv. típ. precisión	0.04	0.04	0.07	0.10	0.13	0.27

Cuadro 9: Resultados de las arquitecturas de kNN para la aprox. 2.

Por último, mostramos los resultados para las arquitecturas del modelo de kNN (cuadro 9), y en la que el número de vecinos más cercanos de las arquitecturas aumenta de uno a dos, veinte, ochenta, doscientos y mil. Se muestran precisión, su desviación, y el *F1-score*, al igual que en las tablas anteriores. Aumentar el número de vecinos parece actuar en detrimento de las métricas, que ofrecen resultados más flojos. Con dos vecinos, la precisión se mantiene en 0.98, así como el F1-score. A partir de ahí, ambas bajan en similar proporción (redondeada a dos decimales).

4.2.3. Discusión

Con los resultados en el caso de las RNA, se observa que la arquitectura 4 con 64 neuronas en la primera capa oculta y 50 en la segunda, junto con la arquitectura 6 con 64 y 32 neuronas respectivamente, obtienen un rendimiento destacado. Tanto en precisión como en *F1-score*, estas arquitecturas alcanzan valores superiores al 0.98, con desviaciones típicas relativamente bajas, lo que sugiere una consistencia en el rendimiento del modelo.

Los modelos SVM muestran variabilidad en su rendimiento según el tipo de kernel utilizado. El kernel polinomial logra buenos resultados, con precisión y *F1-score* cercanos a 0.98, mientras que el kernel lineal produce ligeramente aún mejores resultados, con una precisión y *F1-score* de 0.98 y 0.99 respectivamente. Además, la desviación típica indica una consistencia en el rendimiento de estos modelos.

Los árboles de decisión muestran una mejora en el rendimiento a medida que aumenta la profundidad máxima del árbol, con una precisión y *F1-score* máximos de 0.95 y 0.95 respectivamente para una profundidad de 20. Sin embargo, la desviación típica indica una mayor variabilidad en el rendimiento de estos modelos en comparación con otros.

En el caso de kNN, se observa que el rendimiento del modelo varía significativamente con el número de vecinos considerados. Con un vecino, kNN alcanza una precisión y *F1-score* de 0.98, pero con un aumento en el número de vecinos, la precisión disminuye considerablemente. Además, la desviación típica aumenta notablemente a medida que aumenta el número de vecinos, lo que indica una mayor variabilidad en el rendimiento del modelo.

Aquí podemos ver estos resultados, que serían los mejores para cada una de las cuatro arquitecturas, en el cuadro 10 y la figura 4:

	RNA	SVM	Á. d.	kNN
Precisión	0.99	0.99	0.95	0.98
F1-score	0.99	0.99	0.95	0.98
Desv. típ. precisión	0.03	0.03	0.07	0.04

Cuadro 10: Comparación entre los mejores resultados de las métricas de la aprox. 2.

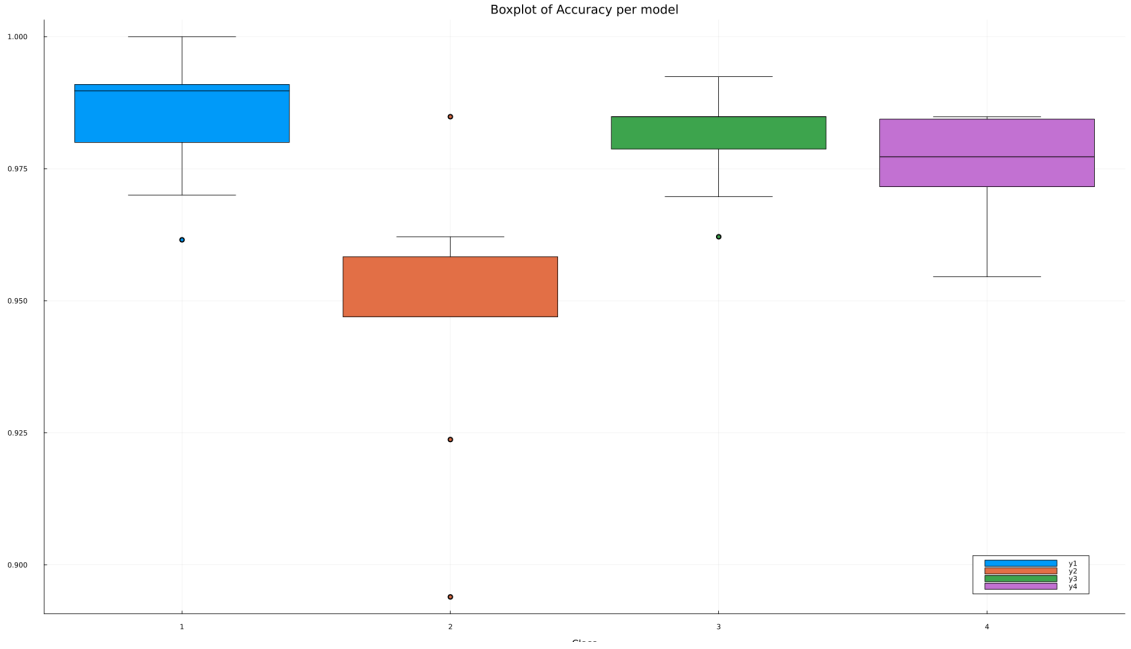

Figura 5: Diagrama de caja de la precisión alcanzada por los modelos RNA, árbol de decisión, SVM y kNN, en este orden, con las arquitecturas que mejores resultados obtienen para la aprox. 2.

Tal y como podemos ver, nuestro nuevo método para extraer datos resulta altamente efectivo en lo que respecta a este problema. Por lo tanto, es conveniente continuar explorando en esta línea, aumentando el dominio de nuestros datos y comprobando si es efectivo en todos los casos.

4.3. Aproximación 3

4.3.1. Descripción

Una vez que hemos comprobado el correcto funcionamiento de nuestra segunda aproximación, vamos a aumentar aún más nuestro dominio, utilizando aproximadamente la mitad de las notas disponibles en la base de datos para todos los instrumentos. Es así como obtenemos 42 clases, que son las notas que vamos a distinguir y que según el método de la aproximación pasada generan 84 *targets*. Más allá de este matiz, se mantiene el mismo funcionamiento a la hora de realizar la aproximación. La distribución de valores, por tanto, resulta de la siguiente manera (figura 5):

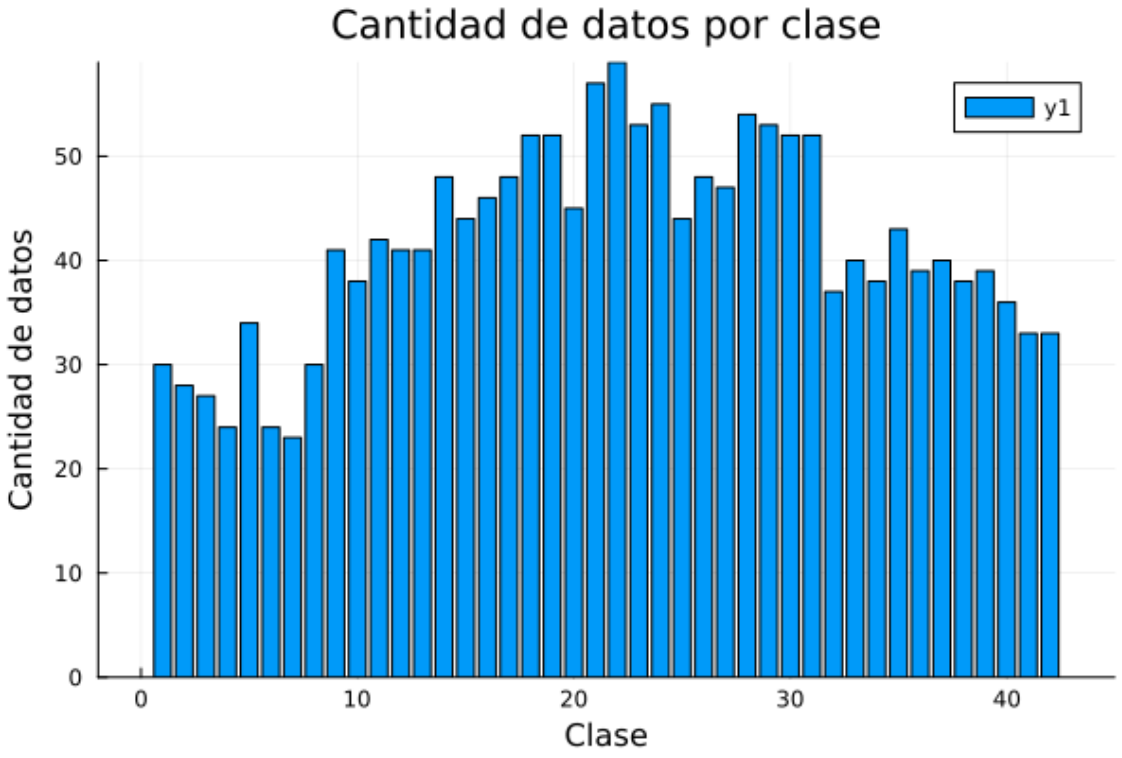

Figura 6: Distribución de valores del dataset en la aprox. 3. por clase

4.3.2. Resultados

Los siguientes conjuntos representan una ejecución promedio del código con los valores de test, en la que los resultados son una representación aproximada de todos los valores obtenidos.

Cabe recordar que, además de las métricas mencionadas en el segundo apartado de esta memoria, enseñamos también la desviación típica de la precisión.

	RNA 1	RNA 2	RNA 3	RNA 4	RNA 5	RNA 6	RNA 7	RNA 8
Capa oculta 1	4	8	16	64	50	64	120	150
Capa oculta 2	-	-	-	-	20	32	110	120
Precisión	0.36	0.85	0.97	0.98	0.98	0.98	0.98	0.98
F1-score	0.32	0.85	0.97	0.98	0.99	0.98	0.98	0.98
Desv. típ. precisión	0.17	0.08	0.04	0.03	0.03	0.03	0.03	0.03
Error RNA (cross-entropy)	-	0.73	0.15	0.10	0.09	0.10	0.09	0.09

Cuadro 11: Resultados de las arquitecturas de RNA para la aprox. 3.

Vemos los resultados de RNA (cuadro 11) para una capa oculta de cuatro, ocho, dieciséis y sesenta y cuatro neuronas respectivamente y, a continuación, con dos capas de 50/20, 64/32, 120/110 y, por último, 150/120 neuronas. Se enseñan los valores de las métricas de precisión y *F1-score*, desviación típica de la precisión, y el error cross-entropy. En similitud con la anterior aproximación, obtenemos los mejores resultados de precisión y F1-score con dos capas de 50 y 20 neuronas, además de un error bajo.

La evolución del error *cross-entropy* durante el entrenamiento de RNA ha sido aceptable, no vemos datos erróneos. Podemos verlo en la figura 6.

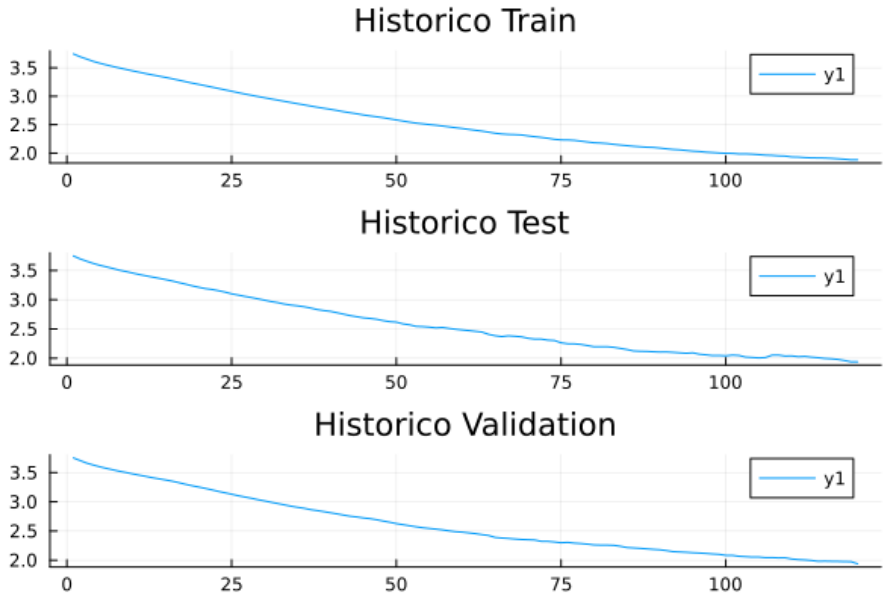

Figura 7: Gráfico del histórico de entrenamiento de RNA para la aprox. 3.

	SVM 1	SVM 2	SVM 3	SVM 4	SVM 5	SVM 6	SVM 7	SVM 8
Kernel	rbf	rbf	sigmoid	sigmoid	poly	poly	linear	linear
Grado del kernel	3	3	20	5	3	4	20	60
Gamma del kernel	8	2	7	3	2	3	15	9
C	7	1	4	2	1	5	9	7
Precisión	0.02	0.01	0.02	0.02	0.96	0.95	0.99	0.99
F1-score	0.00	0.00	0.00	0.01	0.96	0.95	0.99	0.99
Desv. típ. precisión	0.22	0.22	0.22	0.22	0.05	0.05	0.02	0.02

Cuadro 12: Resultados de las arquitecturas de SVM para la aprox. 3.

Se trata de una tabla comparativa entre los modelos SVM (cuadro 12) con kernels sigmoidales, rbf, polinomiales o lineales, con variaciones en los valores de gamma y C. También en este caso, precisión, su desviación típica y el *F1-score*. Mostramos el comportamiento del modelo en función de los hiperparámetros usados, y observamos que, de nuevo, los kernels polinomial y lineal obtienen los mejores resultados, de manera clara.

	Á. d. 1	Á. d. 2	Á. d. 3	Á. d. 4	Á. d. 5	Á. d. 6
Profundidad máxima	2	10	20	50	100	120
Precisión	0.08	0.66	0.92	0.91	0.91	0.92
F1-score	0.06	0.66	0.92	0.91	0.91	0.93
Desv. típ. precisión	0.21	0.13	0.06	0.06	0.06	0.06

Cuadro 13: Resultados de las arquitecturas de árbol de decisión para la aprox. 3.

Tenemos los resultados para las métricas de los árboles de decisión (cuadro 13), junto a las matrices de confusión para cada una de ellas. La profundidad máxima de los árboles va de dos a diez, veinte, cincuenta, cien y, al final, ciento veinte. Observamos que, aparentemente, una mayor profundidad resulta en mejores cifras de métricas: a partir de la arquitectura 3, obtenemos las mejores.

	kNN 1	kNN 2	kNN 3	kNN 4	kNN 5	kNN 6
Número de vecinos	1	2	20	80	200	1000
Precisión	0.98	0.97	0.91	0.82	0.73	0.40
F1-score	0.98	0.97	0.91	0.82	0.71	0.37
Desv. típ. precisión	0.03	0.04	0.06	0.09	0.11	0.17

Cuadro 14: Resultados de las arquitecturas de kNN para la aprox. 3.

Muestra de los resultados para las arquitecturas del modelo de kNN (cuadro 14), y en la que el número de vecinos más cercanos de las arquitecturas aumenta de uno a dos, veinte, ochenta, doscientos y mil. Se muestran precisión, su desviación y *F1-score*, al igual que en las tablas anteriores. De nuevo, un mayor número de vecinos implica unas métricas con peores números.

4.3.3. Discusión

Tras evaluar los resultados obtenidos, y comenzando con los de RNA, se observa que las arquitecturas a partir de 16 neuronas en la primera capa oculta muestran un rendimiento destacado, alcanzando precisión y *F1-score* superiores al 0.97. Añadir una segunda capa oculta, en nuestro ejemplo de 20 o más neuronas, apenas afecta a los resultados de las métricas propuestas.

Los modelos SVM exhiben una variabilidad en el rendimiento dependiendo del tipo de kernel y sus parámetros. Se observa que el kernel lineal y polinomial logran los mejores resultados con una precisión y *F1-score* altísimas, de incluso 0.99. Además, la desviación típica para estos modelos es baja, lo que indica una consistencia en el rendimiento.

Los árboles de decisión muestran una mejora en la precisión y el *F1-score* a medida que aumenta la profundidad máxima del árbol, igual que sucedía en la anterior aproximación. A partir de una profundidad máxima de 20 alcanzamos los mejores resultados posibles, con desviaciones típicas por debajo de 0.07.

Por último, en cuanto a kNN, se observa que el rendimiento del modelo disminuye drásticamente a medida que aumenta el número de vecinos considerados (también, igual que en previas aproximaciones). Con un vecino, kNN alcanza una precisión y *F1-score* de 0.98, pero con un aumento en el número de vecinos, la precisión decrece significativamente, especialmente a partir de los 20. Además, la desviación típica aumenta con el número de vecinos, lo que indica una mayor variabilidad en el rendimiento del modelo.

Disponemos de un cuadro 15 y una figura 7 comparativa para visualizar de manera conjunta los mejores resultados para cada una de las cuatro arquitecturas:

	RNA	SVM	Á. d.	kNN
Precisión	0.98	0.99	0.92	0.98
F1-score	0.98	0.99	0.93	0.98
Desv. típ. precisión	0.03	0.02	0.06	0.03

Cuadro 15: Comparación de los mejores resultados de las métricas para la aprox. 3.

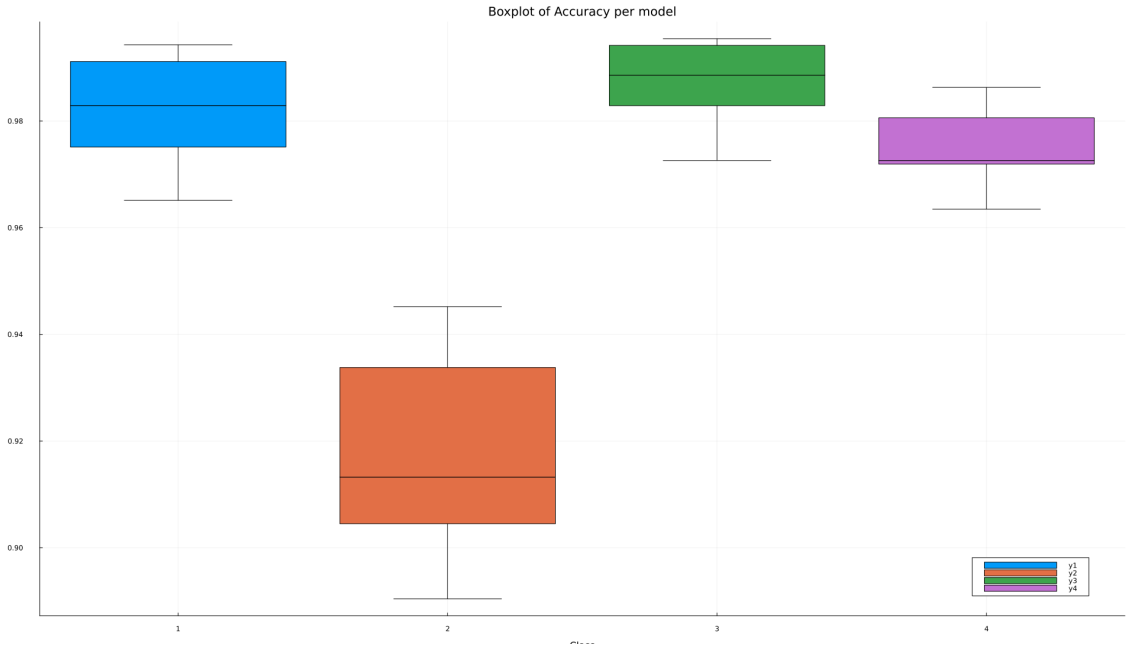

Figura 8: Diagrama de caja de la precisión alcanzada por los modelos RNA, árbol de decisión, SVM y kNN, en este orden, con las arquitecturas que mejores resultados obtienen para la aprox. 3.

A semejanza de la aproximación anterior, los modelos de RNA, SVM y kNN vuelven a destacar, en detrimento de los árboles de decisión que, en cualquier caso, consiguen unas cifras nada desdeñables.

Como se puede observar, nuestro modo de enfrentar el problema se mantiene efectivo y, por lo tanto, consideramos que hay que expandir el dominio del problema para continuar avanzando en esta dirección, y así aumentar la complejidad del problema, completando una nueva aproximación.

4.4. Aproximación 4

4.4.1. Descripción

Ahora que hemos comprobado la aparente eficiencia en nuestro proceso de encontrar y asociar los valores de entradas y las salidas, hemos decidido añadir todos los valores del *dataset*, con el fin de ver cómo se comporta nuestro modelo frente a pistas de varios instrumentos de diferentes familias, como pueden ser cuerda frotada o viento metal, con todas sus notas. De esta manera, la nueva distribución de datos cuenta con un total de 2913 ventanas sobre las que entrenar. Por lo demás, mantenemos la estructura de la anterior aproximación.

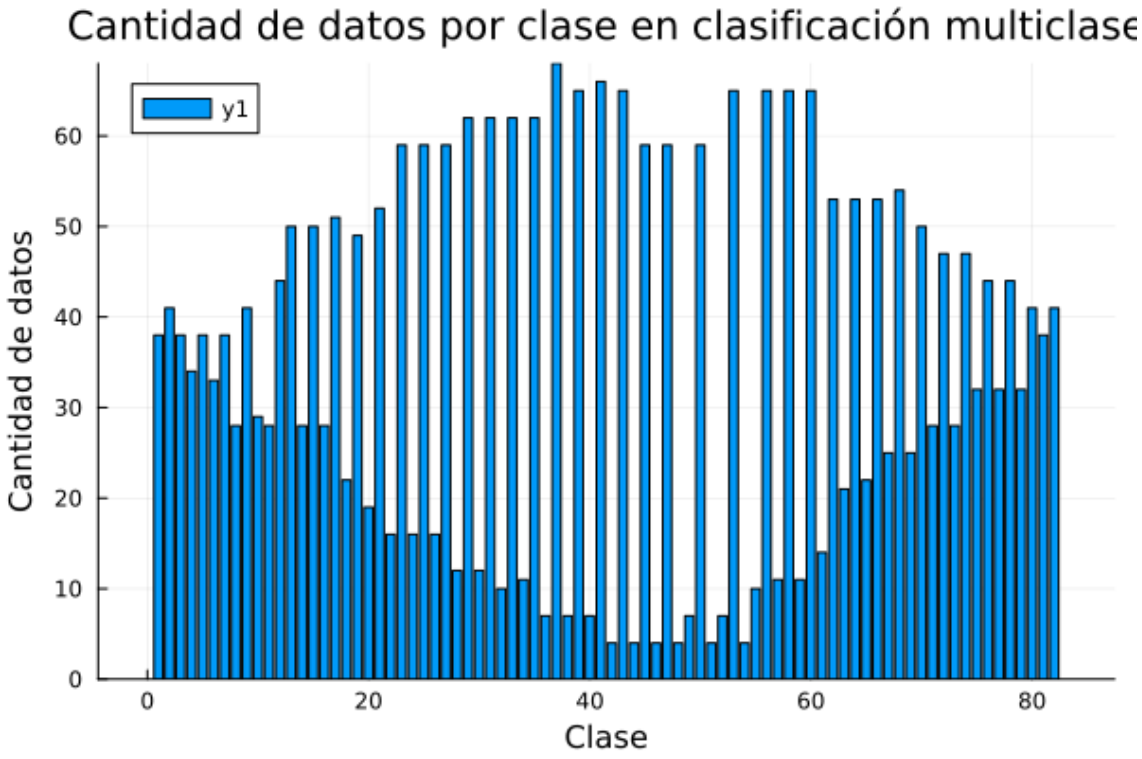

Figura 9: Distribución de valores del dataset en la aprox. 4. por clase

4.4.2. Resultados

Los siguientes conjuntos representan una ejecución promedio del código con los valores de test, en la que los resultados son una representación aproximada de todos los valores obtenidos.

	RNA 1	RNA 2	RNA 3	RNA 4	RNA 5	RNA 6	RNA 7	RNA 8
Capa oculta 1	4	8	16	64	50	64	120	150
Capa oculta 2	-	-	-	-	20	32	110	120
Precisión	0.25	0.74	0.93	0.98	0.96	0.98	0.98	0.98
F1-score	0.20	0.73	0.93	0.97	0.96	0.98	0.97	0.98
Desv. típ. precisión	0.13	0.08	0.04	0.02	0.03	0.02	0.02	0.02
Error RNA (cross-entropy)	2.58	1.24	0.38	0.11	0.17	0.12	0.12	0.09

Cuadro 16: Resultados de las arquitecturas de RNA para la aprox. 4.

Vemos la tabla de las métricas RNA (16) para una capa oculta de cuatro, ocho, dieciséis y sesenta y cuatro neuronas respectivamente y, a continuación, con dos capas de 50/20, 64/32, 120/110 y, por último, 150/120 neuronas. Se enseñan los valores de las métricas de precisión y *F1-score*, desviación típica de la precisión, y el error cross-entropy. Los resultados vuelven a ser similiares a los de la anterior aproximación, con precisiones y *F1-scores* por encima de 0.90 de manera consistente (a partir de una capa de dieciséis neuronas).

La evolución del error *cross-entropy* durante el entrenamiento de RNA ha sido normal, y no vemos datos sospechosos. Podemos visualizarlo en la figura 25.

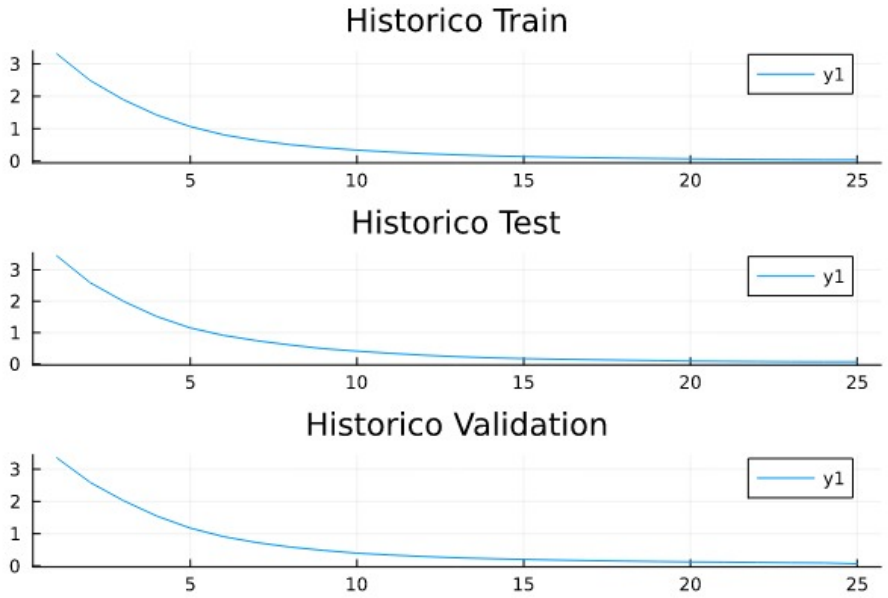

Figura 10: Gráfico del histórico de entrenamiento de RNA para la aprox. 4.

	SVM 1	SVM 2	SVM 3	SVM 4	SVM 5	SVM 6	SVM 7	SVM 8
Kernel	rbf	rbf	sigmoid	sigmoid	rbf	linear	linear	linear
Grado del kernel	20	3	20	5	5	5	20	60
Gamma del kernel	8	2	7	3	3	3	15	9
C	7	1	4	2	2	2	9	7
Precisión	0.01	0.01	0.01	0.01	0.02	0.98	0.98	0.98
F1-score	0.00	0.00	0.00	0.00	0.00	0.99	0.98	0.98
Desv. típ. precisión	0.16	0.15	0.16	0.16	0.15	0.02	0.02	0.02

Cuadro 17: Resultados de las arquitecturas de SVM para la aprox. 4.

Se trata de una tabla comparativa entre los modelos SVM (cuadro 17) con kernels sigmoidales, rbf y lineales, con variaciones en los valores de gamma y C. También en este caso, precisión, su desviación típica y el *F1-score*. Mostramos el comportamiento del modelo en función de los hiperparámetros usados, y vemos que los kernels lineales vuelven a sacar una gran ventaja en todos los campos.

	Á. d. 1	Á. d. 2	Á. d. 3	Á. d. 4	Á. d. 5	Á. d. 6
Profundidad máxima	2	10	20	50	100	120
Precisión	0.06	0.58	0.77	0.86	0.86	0.86
F1-score	0.04	0.57	0.77	0.86	0.86	0.86
Desv. típ. precisión	0.15	0.10	0.08	0.06	0.06	0.06

Cuadro 18: Resultados de las arquitecturas de árbol de decisión para la aprox. 4.

Tenemos los resultados para las métricas de los árboles de decisión (cuadro 18), junto a las matrices de confusión para cada una de ellas. La profundidad máxima de los árboles va de dos a diez, veinte, cincuenta, cien y, al final, ciento veinte. De nuevo, a mayor profundidad, mejores resultados. La precisión, de 0.86 para el mejor caso, se encuentra un poco por debajo de los resultados de la aproximación anterior. Por otro lado, profundidades bajas parecen devolver resultados muy pobres.

	kNN 1	kNN 2	kNN 3	kNN 4	kNN 5	kNN 6
Número de vecinos	1	2	20	80	200	1000
Precisión	0.97	0.96	0.89	0.79	0.66	0.46
F1-score	0.97	0.95	0.88	0.77	0.61	0.39
Desv. típ. precisión	0.03	0.03	0.05	0.07	0.09	0.11

Cuadro 19: Resultados de las arquitecturas de kNN para la aprox. 4.

Muestra de los resultados para las arquitecturas del modelo de kNN (cuadro 19), y en la que el número de vecinos más cercanos de las arquitecturas aumenta de uno a dos, veinte, ochenta, doscientos y mil. Se muestran precisión y *F1-score*, al igual que en las tablas anteriores. Vemos que el comportamiento del modelo es mejor con arquitecturas de menor número de vecinos. Las precisiones bajan hasta los 0.46 en el peor de los casos, si bien alcanzan 0.96 en los mejores.

4.4.3. Discusión

Los resultados de la presente aproximación nos recuerdan a los de las dos previas, dadas las grandes similitudes entre sus valores.

En el caso de las RNA, volvemos a observar que a partir de una capa oculta de 16 neuronas, el modelo rinde con una precisión superior al 93 por ciento. Además, se evidencia una disminución en el error RNA (*cross-entropy*) a medida que se aumenta la complejidad de la red neuronal, lo que sugiere una mejora en la capacidad de generalización.

Por otro lado, los modelos SVM vuelven a destacar en sus arquitecturas con kernel lineal, con una precisión y *F1-score* de 0.98 en todas sus configuraciones. Además, la desviación típica de la precisión es muy baja, indicando una consistencia en el rendimiento del modelo.

Los árboles de decisión, por su parte, exhiben de nuevo un aumento en la precisión y el *F1-score* a medida que se aumenta la profundidad máxima del árbol. La profundidad máxima de 50 marca el mejor rendimiento posible, con una precisión y *F1-score* de 0.86, igual que las siguientes arquitecturas. Sin embargo, la desviación típica de la precisión indica una mayor variabilidad en el rendimiento de estos modelos en comparación con los anteriores.

Finalmente, el modelo kNN vuelve a mostrar una disminución en el rendimiento a medida que se aumenta el número de vecinos considerados. Con un vecino, kNN alcanza una precisión y *F1-score* de 0.97, pero con un aumento en el número de vecinos, la precisión disminuye significativamente. La desviación típica de la precisión

en el mejor de los casos se asemeja más a los dos primeros modelos, rindiendo de manera similar, y mejor que los árboles de decisión.

Podemos ver estos resultados, que serían los mejores para cada una de las cuatro arquitecturas, mediante el siguiente cuadro 20 y la figura 10:

	RNA	SVM	Á. d.	kNN
Precisión	0.98	0.98	0.86	0.97
F1-score	0.98	0.98	0.86	0.97
Desv. típ. precisión	0.09	0.02	0.06	0.03

Cuadro 20: Comparación de los mejores resultados de las métricas para la aprox. 4.

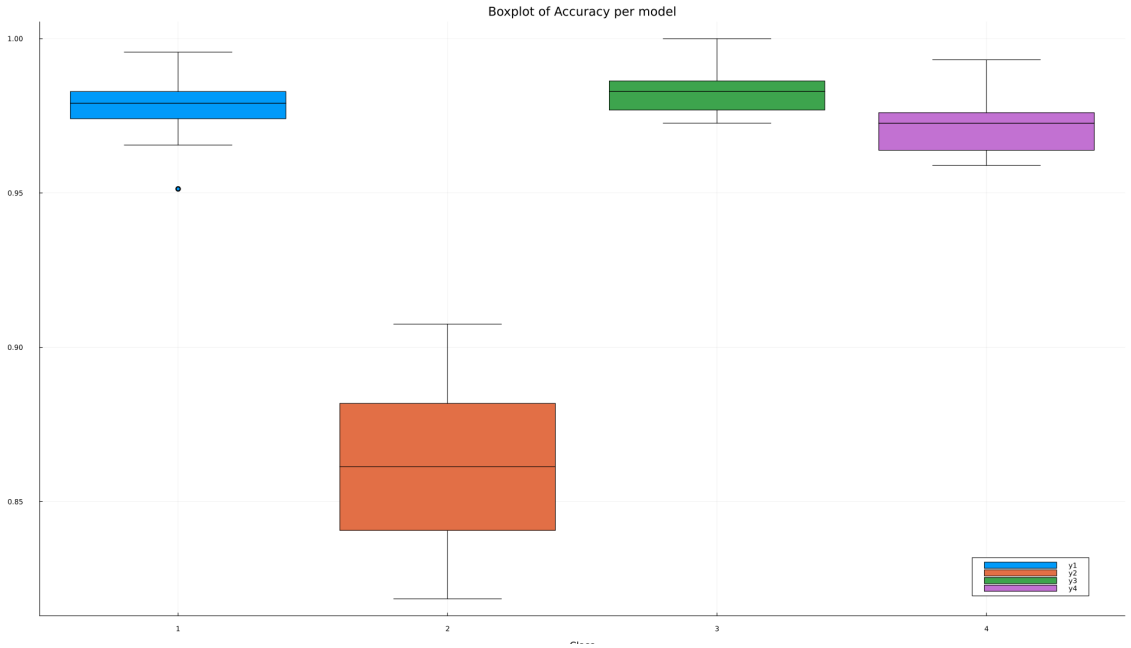

Figura 11: Diagrama de caja de la precisión alcanzada por los modelos RNA, árbol de decisión, SVM y kNN, en este orden, con las arquitecturas que mejores resultados obtienen para la aprox. 4.

Como podemos observar de nuevo, nuestra forma de abordar el problema parece resolver bien el problema, independientemente del número de notas que clasifiquemos. Por tanto, consideramos este resultado satisfactorio, y consideramos que no es necesario hacer más pruebas.

4.5. Aproximación 5: Deep-learning

4.5.1. Descripción

Ahora que hemos llegado a una conclusión aceptable con nuestras aproximaciones anteriores, hemos decido usar *deep-learning* [Wik24e] con los datos para resolver el problema. Ahora el *input* será la propia muestra del *.wav*, que aun así se seguirá procesando por ventanas para tener un *input* de tamaño fijo para el modelo. Debido al aumento del tamaño de los *inputs*, de 82 valores por ventana, y 65535 ventanas, se decide disminuir drásticamente el número de datos que se van a utilizar, a cinco archivos de audio *.wav* para 10 notas, con lo que el número de clases cambia a diez. Este aumento del tamaño del *input* ocurre debido a que el modelo requiere los datos sin procesar, ya que esta acción la realiza el propio modelo.

En lo que respecta a la red, se ejecuta un bucle que entrena la red hasta que el conjunto de entrenamiento llega a un 90 % precisión o no mejora durante 10 ciclos y la función de error.

En cuanto a la distribución utilizada, no tenemos conjunto de validación, solo de test, que corresponde al 20 % de los datos iniciales. El 80 % restante es para el conjunto de entrenamiento. Por otro lado, al final tenemos 328 patrones, es decir, el número de ventanas que utilizamos para esta aproximación.

4.5.2. Resultados

Realizamos un total de diez arquitecturas distintas para el modelo RNA en *deep-learning*, y cuyos datos podemos consultar bajo estas líneas. Tenemos columnas para las capas de cada arquitectura de RNA convolucional, y otras dos con las precisiones del conjunto de entrenamiento y de test.

Observamos que la distribución de clases es relativamente equilibrada, mucho más que en las aproximaciones anteriores. Por eso, consideramos que la métrica de *F1-score* no es especialmente relevante para esta, y la omitimos de los resultados. Nos centraremos en la precisión, que es aún más importante si cabe.

	Capa 1	Capa 2	Capa 3	Capa Dense	Prec. conj. entrenamiento	Prec. conj. test
Arq. 1	1=>16	16=>32	32=>32	262144	78.47 %	28.79 %
Arq. 2	1=>16	16=>32		524288	78.69 %	30.30 %
Arq. 3	1=>16	16=>32	32=>64	524288	18.49 %	13.64 %
Arq. 4	1=>1			32768	79.40 %	25.76 %
Arq. 5	1=>8			262144	79.09 %	27.27 %
Arq. 6	1=>32			1048576	39.09 %	9.09 %
Arq. 7	1=>8	8=>16		262144	80.03 %	27.27 %
Arq. 8	1=>32	32=>64		1048576	18.81 %	31.82 %
Arq. 9	1=>16			524288	40.11 %	28.79 %
Arq. 10	1=>8	8=>16	16=>32	262144	74.72 %	37.88 %

Cuadro 21: Resultados de las arquitecturas de RNA convolucionales para la aprox. 5.

Mostramos en el cuadro 21 los resultados para diez arquitecturas distintas para

RNA convolucionales, con cuatro columnas para las distintas capas (con datos que cambiamos) de la misma, y los resultados de precisión para cada uno de los conjuntos, es decir, para entrenamiento y test.

Las columnas de las capas 1, 2 y 3 muestran los canales de entrada y salida, respectivamente. Si hay dos o más, tendremos capas de *pooling* entre ellas, con tamaño de ventana 2, y que no se muestran en el cuadro por brevedad y evitar redundancia. La capa Dense, por su parte, hace referencia a la red neuronal densa de nuestro modelo, y en ella se muestra el número de características de entrada (por ejemplo, en el décimo caso, 262144). La salida para todas las arquitecturas es de 10, que tampoco se muestra en el cuadro.

Por último, la función de activación de la última capa es la *softmax*, que se utiliza comúnmente en las tareas de clasificación para convertir las salidas de la capa anterior en probabilidades.

4.5.3. Discusión

Tras observar los resultados para la precisión en los conjuntos de test, podemos ver que dicha métrica se encuentra muy lejos de alcanzar el gran rendimiento de las aproximaciones previas. Con precisiones tan bajas como del 9 por ciento, y llegando a máximos de 38, nos quedamos lejos de las marcas conseguidas con la aproximación 4, en las que además se consideran muchos más datos.

Las mejores arquitecturas podrían parecer estar ligeramente relacionadas con un mayor número de capas, siendo las de tres las que mejores resultados potenciales alcanzan, en general. Sin embargo, las diferencias son escasas. El peor resultado lo tenemos en la arquitectura 6, con una capa de un canal de entrada y dieciséis de salida, y que resulta en una pobre precisión del nueve por ciento. Sin embargo, la segunda peor precisión la marca la arquitectura 3, que tiene tres capas. Por tanto, no podemos concluir definitivamente que los resultados de las métricas se vean afectados por las capas. En general, observamos bastante irregularidad en las precisiones resultantes.

En resumen, es por esto que no consideramos exitosos los resultados de esta aproximación, si bien creemos que hemos aplicado correctamente las características de las RNA convolucionales a nuestro problema.

5. Conclusiones

Tras la realización de las cinco aproximaciones, y teniendo en cuenta todos los resultados obtenidos para las métricas que estimamos en su momento, consideramos que los números son muy buenos, y se cumple con el objetivo marcado al inicio de este documento. El problema tratado se resuelve de manera satisfactoria para la gran mayoría de los casos, con unos datos de precisión excelentes, y poca variación en los mismos. Por tanto, creemos que el sistema inteligente es viable para la resolución de la problemática en cuestión, además de su posible aplicación en entornos similares (que trataremos en la siguiente sección, relativa al trabajo futuro).

Entendimos desde el principio el entorno como un problema de clasificación, generando un *input* consistente en la división de los archivos de audio en ventanas con cierto grado de solapamiento, y utilizando la transformada de Fourier rápida para obtener datos de las señales en todas las frecuencias.

Si bien el primer acercamiento estableció las bases del sistema, presentando los cuatro modelos de entrenamiento y sus características destacadas, y produjo las peores métricas para dichos modelos (aceptables, aún así), las aproximaciones segunda, tercera y cuarta se centraron en la ampliación de la base de datos disponible para la ejecución, además de realizar las pertinentes modificaciones en el código fuente para su gestión, y de la optimización del proceso al completo, produciendo los excelentes resultados de los que disponemos. Tras la visualización de estos datos presentados, amén de las métricas estimadas y arquitecturas probadas, podemos resaltar la fortaleza de los modelos de RNA y SVM, por ese orden, que desde la primera aproximación destacaban en precisión y su desviación típica, más alta y más baja de lo normal respectivamente.

La revisión de los resultados parece determinar que la mejor aproximación fue la cuarta, en la que el modelo de RNA con dos capas ocultas de 64 y 32 neuronas, resultó en una excelente precisión del 98 %, con una desviación típica de solo 0.02. Además, el modelo SVM obtuvo similares estadísticas para un kernel lineal de grado 5, gamma 3 y C igual a 2. Eso sí, la falta de un test estadístico que pueda confirmarlo, hace que no tengamos estos números como afirmaciones absolutas.

Debe comentarse también que los resultados de la quinta aproximación, o de *deep-learning*, y consistente en la introducción de redes de neuronas convolucionales, arroja peores resultados que las aproximaciones anteriores en las métricas consideradas más útiles, y por tanto no es considerada la mejor, pese a ser la más reciente.

Para cerrar este apartado, estimamos que la mayor dificultad atravesada en la realización de nuestro trabajo sería relativa a la parte del procesamiento de los datos en el código, es decir, la conversión de dichos datos en formatos aceptables para los modelos, ya que ocurrió al comienzo de la práctica, y había que tener una amplia gama de factores en cuenta (división de ventanas, solapamiento, etc.). Por fortuna, el desarrollo del resto de aspectos de la práctica fluyó con relativa comodidad, mediante trabajo semanal.

6. Trabajo futuro

A lo largo del desarrollo de este sistema inteligente, y mediante iteraciones que mejoraban el rendimiento del código, hemos alcanzado una resolución al problema inicial planteado que estimamos ampliamente satisfactoria, y que permite discernir notas musicales de varios tipos de instrumentos.

La resolución de esta problemática nos ha parecido interesante desde el mismo inicio ya que, por ejemplo, puede proporcionar un medio para distinguir las notas de una composición musical a aquellas personas que no tienen un oído absoluto, o

a las que empezando a adentrarse en el campo de la música. Consideramos que la aplicación más útil se realizaría precisamente en el campo de la educación.

Se nos ocurre también que este sistema podría ser integrado en un conjunto de herramientas más grande, asistidas por inteligencia artificial, para músicos. Así, un programa de generación de canciones, por ejemplo, puede saber qué notas musicales se están reproduciendo, y así actuar en consecuencia. Otro uso en línea con este sería el de una composición asistida, en la que el usuario podría aportar sus ideas y recibiría sugerencias de armonización para continuar, por ejemplo.

Otra aplicación evidente de nuestro trabajo sería su uso como afinador de instrumentos. La detección de notas del sistema permitiría saber en todo momento a la persona que lo utiliza qué nota se está tocando, y así tomar las medidas correspondientes. Además, podrían añadirse formas de detectar y evaluar la calidad del sonido, o la misma sugerencia de ajustes precisos para su consecución, que asistan al usuario en su objetivo.

En lo que respecta a otros entornos, pensamos que el tratamiento que realizamos de los datos en este trabajo podría ser reutilizado para otros problemas relacionados con el audio. Por ejemplo, podríamos tener sistemas inteligentes de clasificación de sonidos relativos a los videojuegos, en los que se procesarían audios pertenecientes a distintas sagas y entregas de este medio de entretenimiento, y se extraería el videojuego al que pertenecen.

Referencias

[Bhu23] Bhuwan Bhattarai y Joonwhoan Lee. A Comprehensive Review on Music Transcription. https://www.mdpi.com/2076-3417/13/21/11882, 2023. En inglés.

[Car20] Carmine Emanuele, Daniele Ghisi, Vincent Lostanlen, Fabien Lévy, Joshua Fineberg, Yan Maresz. TinySOL: an audio dataset of isolated musical notes. https://zenodo.org/records/3685367, 2020. En inglés.

[Chi18] Chiheb Trabelsi y Olexa Bilaniuk. Deep Complex Networks. https://paperswithcode.com/paper/deep-complex-networks/review/?hl=22615, 2018. En inglés.

[IFP23] IFPI. Global Music Report. https://globalmusicreport.ifpi.org/, 2023. En inglés.

[Jas21] Jason Brownlee. Tour of Evaluation Metrics for Imbalanced Classification. https://machinelearningmastery.com/tour-of-evaluation-metrics-for-imbalanced-classification/, 2021. En inglés.

[Kel22] Kelvin A. Minor. Automatic Music Transcription Using Fourier Transform for Monophonic and Polyphonic Audio File. https://iieta.org/journals/isi/paper/10.18280/isi.270413, 2022. En inglés.

[Leo23] Leo Prasanth. A Framework for Piano Notes Analysis Using Digital Signal Processing. https://ijcrt.org/papers/IJCRT2302528.pdf, 2023. En inglés.

[Man18] Manuel Mínguez Carretero. Automatic Music Transcription Using Neural Networks. https://rua.ua.es/dspace/bitstream/10045/76991/1/Automatic_music_transcription_using_neural_networks_MINGUEZ_CARRETERO_MANUEL.pdf, 2018. En inglés.

[Moh18] Mohamed Karioun y Simon Tihon. Deep Learning in Automatic Piano Transcription. https://dial.uclouvain.be/memoire/ucl/en/object/thesis%3A17184/datastream/PDF_01/view, 2018. En inglés.

[Pot06] Potcharapol Suteparuk. Detection of Piano Keys Pressed in Video. https://stacks.stanford.edu/file/druid:bf950qp8995/Suteparuk.pdf, 2006. En inglés.

[Say22] Sayali Gadre. Piano Notes Recognition using Digital Signal Processing. https://ieeexplore.ieee.org/abstract/document/10007478, 2022. En inglés.

[Wik24a] Wikipedia. Box plot. https://en.wikipedia.org/wiki/Box_plot, 2024. En inglés.

[Wik24b] Wikipedia. CART. https://es.wikipedia.org/wiki/Aprendizaje_basado_en_%C3%A1rboles_de_decisi%C3%B3n, 2024.

[Wik24c] Wikipedia. Confusion matrix. https://en.wikipedia.org/wiki/Confusion_matrix, 2024. En inglés.

[Wik24d] Wikipedia. Cross-entropy. https://en.wikipedia.org/wiki/Cross-entropy, 2024. En inglés.

[Wik24e] Wikipedia. Deep learning. https://en.wikipedia.org/wiki/Deep_learning, 2024. En inglés.

[Wik24f] Wikipedia. Epitafio de Sícilo. https://es.wikipedia.org/wiki/Epitafio_de_S%C3%ADcilo, 2024. Composición musical completa más antigua que se conserva.

[Wik24g] Wikipedia. Fast Fourier transform. https://en.wikipedia.org/wiki/Fast_Fourier_transform, 2024. En inglés.

[Wik24h] Wikipedia. Música del Clasicismo. https://es.wikipedia.org/wiki/M%C3%BAsica_del_Clasicismo, 2024.

[Wik24i] Wikipedia. Softmax function. https://en.wikipedia.org/wiki/Softmax_function, 2024. En inglés.