Reconocimiento automático de notas del piano

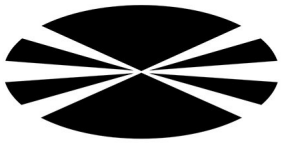
David Rodríguez Bacelar

Kevin Millán Canchapoma

Luca D'angelo Sabín

Jorge Hermo González

18 de abril de 2022



UNIVERSIDADE DA CORUÑA

Índice

1. 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 2, desarrollaremos las diferentes soluciones en la Sección 4, hablaremos sobre las conclusiones del trabajo en la Sección 5 y finalizaremos comentando las aplicaciones al mundo real en la Sección 6. También se podrá consultar las bibliografía utilizada en la Sección 3 y 7.

1.1. Glosario

- Sample: muestra de carácter musical
- 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 artificial.
- 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. 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 modelo.

2.1. 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.

2.2. 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 media de los *samples* es de aproximadamente 6 segundos.

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.

3. 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 artificiaes 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*.

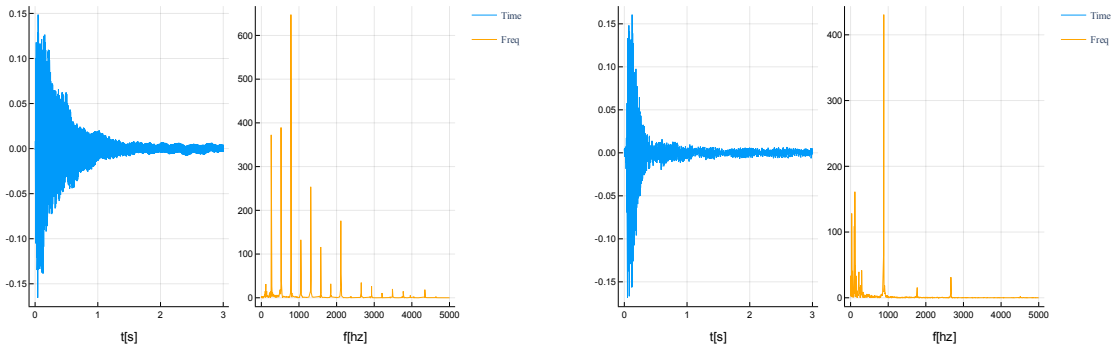
4. Desarrollo

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.

4.1. Primera aproximación

4.1.1. 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 1a y 1b 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 1: 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, nos 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 ruído que se salga de los valores típicos, por lo que una normalización entre mínimo y máximo no es lo adecuado.

Los parámetros de normalización para cada característica son los siguientes:
(Media ± Desviación Típica)

- E: 0,0045678928 ± 0,0035726614
- zero-crossing: 1797,2139 ± 1352,7441
- m1: 16,821161 ± 11,616611
- m2: 9,4079 ± 5,8349323
- m3: 14,607474 ± 11,515968
- m4: 3,8720663 ± 3,5469663
- m5: 2,3520973 ± 2,1539617
- m6: 0,48698702 ± 0,6487291
- m7: 0,7153551 ± 1,0973071

- m8: 0,40077275 ± 0,500489
- m9: 0,22367951 ± 0,3412378
- m10: 0,23234393 ± 0,5112222
- std1: 54,282375 ± 46,679535
- std2: 31,582827 ± 27,185955
- std3: 57,896194 ± 38,439426
- std4: 11,108885 ± 12,696077
- std5: 7,0041313 ± 6,732649
- std6: 0,52131796 ± 0,72755563
- std7: 2,7318008 ± 4,6750984
- std8: 0,9234557 ± 1,2807367
- std9: 0,47755784 ± 0,89846444
- std10: 0,5695597 ± 1,2890217
- max1: 832,8222 ± 779,30615
- max2: 567,7255 ± 505,1785
- max3: 901,9232 ± 548,2265
- max4: 192,80989 ± 237,47154
- max5: 98,938354 ± 103,55783
- max6: 4,392064 ± 6,0777674
- max7: 40,596066 ± 65,99385
- max8: 11,672165 ± 15,524432
- max9: 5,7904434 ± 10,440257
- max10: 5,6131926 ± 11,565159

Donde E indica la **energía de la señal**, zero-crossing indica el número de veces que la señal cruza el eje horizontal por segundo en el tiempo (**Zero-crossing/s**) mi indica la **media de la intensidad** en el intervalo i en frecuencia, $stdi$ indica la desviación típica de la intensidad en el intervalo i en frecuencia y $maxi$ indica el máximo de la intensidad en el intervalo i en frecuencia.

Se van a realizar experimentos sobre los siguientes modelos: RRNNAA, SVM, Árboles de decisión y kNN.

Para los experimentos, se va a realizar ***cross-validation*** con cada modelo, para así poder evaluar el rendimiento de cada uno, con independencia de la aleatoriedad que se pueda producir al seleccionar un subconjunto de patrones para test. 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*. Para obtener el valor de las métricas en cada fold, se computarán para el conjunto de test seleccionado en cada fold.

Además, 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.

4.1.2. Resultados

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

4.1.2.1 RRNNAA

Para el entrenamiento de las RRNNAA, 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 1

Tabla 1: Resultados RRNNA

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]	$0{,}9621 \pm 0{,}0974$	$0{,}97934 \pm 0{,}05699$

4.1.2.2 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 2

Tabla 2: 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)	$0{,}93642 \pm 0{,}07368$	$0{,}94119 \pm 0{,}07166$
(sigmoid, 3, 1, 1)	$0{,}71875 \pm 0{,}15823$	$0{,}77143 \pm 0{,}11289$
(poly, 5, 10, 0,1)	$0{,}99 \pm 0{,}03162$	$0{,}98571 \pm 0{,}04518$
(rbf, 3, 10, 0,1)	$0{,}76689 \pm 0{,}01542$	$0{,}62214 \pm 0{,}02026$
(sigmoid, 3, 10, 0,1)	$0{,}70059 \pm 0{,}1406$	$0{,}77768 \pm 0{,}06262$
(rbf, 3, 0,01, 100)	$1{,}0 \pm 0{,}0$	$1{,}0 \pm 0{,}0$
(rbf, 3, 100, 0,01)	$0{,}76689 \pm 0{,}01542$	$0{,}62214 \pm 0{,}02026$

4.1.2.3 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 3

4.1.2.4 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 4

Tabla 3: 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	$1{,}0 \pm 0{,}0$	$1{,}0 \pm 0{,}0$

Tabla 4: 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	$0{,}99091 \pm 0{,}02875$	$0{,}99286 \pm 0{,}02259$

4.1.3. Discusión

Los resultados han sido muy buenos, ya que en la mayoría de casos de 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 escogido en esta aproximación era muy sencillo y utilizamos un número elevado de características que puedan permitir separar las clases.

En lo referente a las características, han sido correctamente elegidas, ya que alcanzamos una precisión muy alta. Pero también se puede culpar al elevado número de características que algunos sistemas concretos no funcionen tan bien (por ejemplo, algunos SVM), ya que algunas de ellas pueden ser innecesarias o hasta contraproducentes en este sencillo problema. Sería buena opción intentar reducir el número de características para futuros problemas, pero hay que tener cuidado, ya que si aumentamos el número de clases a clasificar, podría darse el caso de haber eliminado una característica que sí que nos haría falta en un futuro, por lo que de momento es mejor no eliminar características.

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), ya que es ahí donde está la frecuencia de las notas C4 y A5, respectivamente. Las características que se corresponden a dichos intervalos son *m1, std1, max1* para el primer intervalo y *m3, std3, max3* 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. O 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 su capacidad de generalización.

En la figura 2 podemos ver un gráfico del entrenamiento de dicha rna, donde se puede ver el error cometido en los conjuntos de test, validación y entrenamiento para cada ciclo de entrenamiento. Este entrenamiento se ha realizado utilizando todos los patrones, separando un 20 % para el conjunto de test. Se puede observar que la curva de la gráfica desciende muy rápido, esto se puede deber a que tenemos un ratio de aprendizaje alto para este problema concreto. 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).

En la figura 5 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. Se utilizó el 80 % de los patrones restantes para el entrenamiento de la RNA, ya que el conjunto de entrenamiento no se debería utilizar para realizar predicciones.

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

Tabla 5: 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

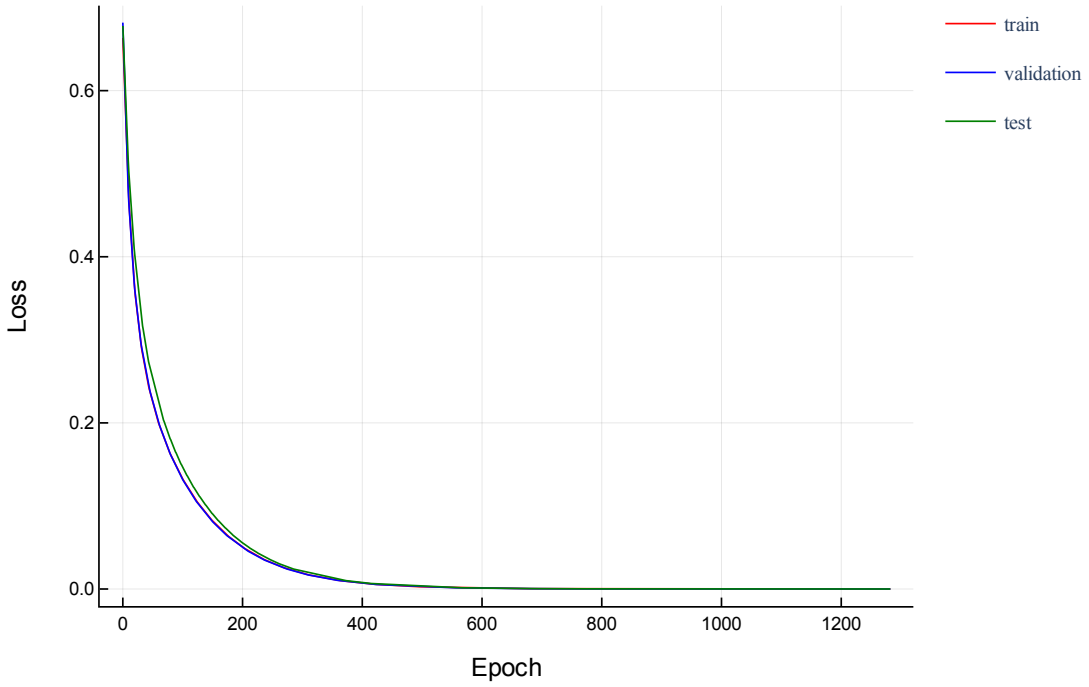


Figura 2: Entrenamiento RNA

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.

4.2. Segunda aproximación

Como hemos obtenido resultados muy buenos en la aproximación anterior, se va a aumentar la complejidad del problema, es decir, añadiremos más notas a ser clasificadas.

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

Intentaremos que el sistema funcione bien con las características que ya tenemos, pero se añadiría alguna más si vemos que el sistema no funciona bien.

5. Conclusiones

6. Trabajo futuro

7. 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*.

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

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

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

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.