eman ta zabal zazu

Universidad del País Vasco
Euskal Herriko Unibertsitatea

INFORMATIKA FAKULTATEA
FACULTAD DE INFORMÁTICA

Trabajo de Fin de Máster

Facultad de Informática

Desarrollo de una nueva herramienta para la composición musical mediante el uso de la voz

Javier Aguirre Herrero

Dirección
Basilio Sierra
Izaro Goienetxea

12-07-2022

Abstract

El desarrollo de una nueva herramienta para la composición musical mediante el uso de la voz es un proyecto de fin de máster y de emprendimiento con el objetivo de mejorar la interacción humano-computadora en la intersección de los campos de la música y ciencias de la computación.

En este trabajo de final de máster se introduce la motivación de este proyecto y se describe la amplia necesidad en la industria de la música por una solución para poder componer en cualquier momento y lugar de una forma intuitiva. Posteriormente se presentan los principales aspectos teóricos como los coeficientes cepstrales del sonido y algoritmos a utilizar para hacer predicciones como kNN y las redes neuronales convolucionales.

A continuación, se describe el pricipal reto tecnológico a la hora de desarrollar el proyecto y se procede a todo lo relativo a los sistemas de reconocimiento de comandos de voz y de frecuencia. A continuación se explica la integración de dichos algoritmos con el sistema y el desarrollo de un sistema de reconocimiento de frecuencias en tiempo real con metrónomo.

Se puede concluir que el principal reto tecnológico ha sido resuelto. Además se ha conseguido un sistema de reconocimiento de comandos de voz con un $92,5\,\%$ de precisión con un peso menor a $0,5$ Mb y una inferencia instantánea. Además se ha integrado un sistema de reconocimiento de frecuencia y se ha desarrollado un sistema preliminar de reconocimiento de frecuencias a tiempo real con metrónomo.

Índice de contenidos

Índice de figuras

CAPÍTULO 1

Introducción

Este trabajo de fin de máster nace de la necesidad de desarrollar una herramienta que permita a músicos y compositores guardar y componer música en cualquier dispositivo mediante el uso de la voz así evitando comprar otros dispositivos y tener una comunicación más intuitiva con la tecnología.

Actualmente es un proyecto de emprendimiento basado en innovación con el objetivo de salir a mercado y es por ello que no es un trabajo puramente científico con el objetivo de superar al estado del arte o de abrir un nuevo campo de investigación sino más bien un trabajo orientado a la ingeniería y desarrollo de un producto en el que la investigación juega un papel importante y que de esta manera pueda ayudar a muchos músicos independientemente de donde se encuentren.

El trabajo empieza con una motivación sobre por qué este proyecto y qué necesidad hay en el mercado además de comentar la proyección del mismo. A continuación se explican los aspectos teóricos necesarios para comprender la terminología que se expone a lo largo del trabajo. Posteriormente se explica uno de los pricipales retos tecnológicos que reside en la integración de todas las partes del sistema para evitar hacer código innecesario y poder escalarlo fácilmente de cara a futuro. Por otro lado, se describe la creación e implementación de los dos sistemas de reconocimiento: uno de voz y uno de frecuencia, así como también su implementación para que funcione en tiempo real. Finalmente, se describe un último sistema en desarrollo que contempla la detección de frecuencias en tiempo real con un metrónomo.

Para concluir tambien se expone el trabajo futuro a realizar y la perspectiva de cara a salir a mercado.

CAPÍTULO 2

Motivación

Para entender la motivación detrás de este trabajo de fin de máster primero debemos entender cómo es el día a día de cualquier compositor o arreglista y cuales son sus necesidades. Al igual que entender mis inquietudes personales, las cuales me han llevado a convertirlo también en un proyecto de emprendimiento.

2.1. Composición musical en el siglo XXI

A día de hoy, gracias a la informática, los compositores apenas componen música a papel como lo harían los compositores clásicos como Bach, Mozart, Beethoven y sus contemporáneos. Antaño solamente los compositores más expertos eran capaces de escribir música para orquesta, ya que tenían la capacidad de escuchar toda una partitura en su cabeza.

Los motivos de utilizar música digitalizada son varios. Por un lado, el compositor, gracias a los softwares de edición de partituras puede escuchar directamente el feedback del software a tiempo real y escuchar cómo está quedando su obra a cada instante con un simple click en el botón de "play". Además, tener una partitura digitalizada permite extraer las partichelas de una partitura general. Esto significa que un compositor puede componer una partitura general de orquesta en la que hay 30 líneas musicales sonando al mismo tiempo y puede extraer automáticamente cada voz para entregarla al interprete correspondiente. Por ejemplo, en una partitura general, se pueden obsevar todos los instrumentos de una orquesta, pero el violinista solamente necesita su parte. A día de hoy dicha extracción se hace de manera automática con estos softwares.

Además, existe la posibilidad de extraer una partitura en formato musicXML (que es la descripción de una partitura en formato XML) y hacer con ello cualquier tarea que se desee. Por ejemplo, importarlo en otro tipo de softwares como los DAW (Digital Audio Workstation) que son secuenciadores que permiten editar el sonido de una manera profesional con instrumentos virtuales, que cuesta distinguir de los reales, para producir directamente la música prescindiendo de interpretes reales.

En definitiva, los editores de partituras llegaron hace mucho tiempo para quedarse y son la herramienta, junto con los DAW, que se utiliza por defecto en la industria musical.

2.2. Objetivo del proyecto de emprendimiento

Para poder entender el objetivo de este trabajo, es imprescindible entender cómo se utilizan estos softwares y los problemas que tienen a día de hoy. Lo primero que hay que saber es cómo un compositor pasa de tener una idea musical a escribirla.

2.2.1. Editores de partitura actuales

A día de hoy se introducen notas mediante el teclado del ordenador y el ratón, se seleccionan las longitudes de nota y se introducen en el pentagrama una a una. Sin embargo, se puede observar que no es nada práctico. Cuando a un compositor se le ocurre una idea musical quiere transcribirla lo antes posible antes de que se le olvide, y en todo este proceso muchas veces ocurre. Además, el tiempo que conlleva su escritura se vuelve muy largo. A día de hoy este problema está solucionado, prácticamente todo compositor dispone de un piano digital que conecta al ordenador para componer de una manera más veloz. De esta forma, si un compositor quiere escribir la nota "sol", en lugar del proceso explicado previamente, toca en su piano digital la tecla de *sol* y automáticamente se imprime en pantalla. En caso de querer escribir *sol, la y si*, solamente hay que tocar esas tres notas en el piano digital y se escribirán. El ahorro de tiempo que supone es tremendo.

Ahora que se ha descrito el proceso de composición y se entiende la importancia del piano digital, veamos qué problemas tiene y qué me ha llevado a emprender con este proyecto. Para ello considero importante entender mi trayectoría y describir cómo he vivido este problema en primera persona.

2.2.2. El problema de no disponer de un piano digital

Empecé en la música desde la edad de los 5 años y he cursado grado elemental y profesional de música. Posteriormente, al acabar a la edad de 18 años empecé a componer de manera regular. A la hora de irme de intercambio universitario a la universidad de $Sungkyunkwan$ en Corea del Sur, me di cuenta de que no podía componer. En 6 meses no pude componer una sola partitura debido a que no disponía de un piano digital. Busqué alternativas en el mercado y no había nada decente disponible para componer de una manera ágil fuera del estudio. Había pianos enrollables y plegables en el mercado, pero tienden a fallar y además no disponía de espacio para desplegarlos. De modo que la mejor solución fue no componer hasta la vuelta.

2.2.3. Solución propuesta

Tras lo descrito en el parrafo anterior, pensé que de alguna manera se podría componer de forma veloz sin necesidad de más dispositivos. Ahí es cuando surgió la idea de usar la voz. Es algo que todos tenemos, no ocupa espacio, y permitiría componer en cualquier lugar, tanto con el móvil como tablets y ordenadores. Es un sistema de composición que permitiría la composición tanto por medio de comandos de voz como mediante el tarareado, permitiría además una nueva forma de componer no solo como sustituto del piano digital, sino como una nueva herramienta de improvisación, más intuitiva y de menor coste. Solamente habría que comprar una aplicación y cualquier persona incluso sin conocimientos podría componer. Además, se resolvería un problema muy extendido entre compositores como es guardar ideas musicales en cualquier lugar. En definitiva una herramienta que me ayudaría a mí y a

todos los músicos y compositores a componer con un menor coste, en cualquier lugar y de manera mucho más intuitiva.

Me gustaría añadir que al ser un proyecto de emprendimiento, se han realizado estudios de mercado que avalan la amplia necesidad del mismo. Por ejemplo, solamente la cantidad de compositores amateurs registrados en una página web de composición consta de más de 6,000,000 de usuarios registrados y es importante destacar que no se trata de uno de los principales softwares de composición. Sumando los usuarios de los principales editores y la cantidad de músicos que también podrían utilizar el software para otros fines como la improvisación, el tamaño es inmenso. Es por ello que diversas instituciones han avalado el proyecto de emprendimiento. Por un lado, he obtenido la *Beca de Bidasoa Activa de emprendimiento*, *Beca de la UPV/EHU de emprendimiento*, *Premio Manuel Laborde a mejor proyecto en la categoría de "Tesis de Grado, Máster y Tésis Doctorales"* y el *Premio Irun ekintzan a mejor proyecto de innovación de la ciudad de Irun*. Además, cuento con el apoyo de *BIC Gipuzkoa* para financiar el proyecto de cara a futuro.

2.3. Metodología de un proyecto de emprendimiento

En todo proyecto de emprendimiento es importante no hacer un gran desarrollo para luego sacarlo a mercado, ya que si el mercado no responde de forma deseada, todo el tiempo y recursos habrán sido desaprovechados. Es por ello que este proyecto se basa en la *metodologia lean* de Eric Ries [1] donde ocurre un ciclo de aprendizaje, construcción y medición a cada paso que se da. En este caso se va a desarrollar un producto mínimo viable para probarlo en el mercado, y, dependiendo de la respuesta obtenida, continuar o pivotar.

Es decir, en caso de querer fabricar y vender coches, plantear desarrollar directamente un coche es algo que llevaría muchísimo tiempo y dinero, además, debido a la alta competencia y a la probablemente pobre calidad del producto en el momento de salir al mercado lo más seguro es que sea un desastre. Es por ello que lo primero debería ser desarrollar un rodamiento e intentar venderlo. En caso de éxito, desarrollar diferentes partes y hacer el mismo proceso hasta poder tener los recursos de desarrollar un coche sin correr un riesgo tan grande.

A este procedimiento, en emprendimiento se le llama PMV (Producto Mínimo Viable). Por ello, en este apartado se desglosará el proyecto en PMV y el producto final. El PMV será una aplicación para dispositivos móviles, que permita guardar ideas al vuelo en cualquier momento y lugar. Para ello, el usuario tendrá la opción de tararear a tiempo real al sonido de una claqueta y el software deberá transcribir lo dicho. Además, para poder corregir posibles fallos o añadir nuevas notas, se usarán comandos de voz como *do, re, mi, fa, sol, la y si.*

El producto final será una aplicación para tablets y ordenadores que esté contectada con la aplicación descrita para móviles que permita seguir desarrollando las ideas que al compositor se le ocurran durante el día, permitiendo de esta manera escribir partituras mucho más complejas.

El desarrollo del PMV está pensado para que todo el código que se cree se pueda reutilizar para crear el producto final.

2.4. Objetivo del trabajo de fin de máster

Teniendo en cuenta todo lo descrito con anterioridad: los objetivos personales, el estudio de mercado realizado y la metodología de trabajo, el objetivo de este trabajo de fin de máster es, por un lado, desarrollar un entorno en el que todo el código que se programe sirva para cualquier dispositivo (ya sean móviles, tablets u ordenadores). Por otro lado, crear un sistema de reconocimiento de comandos de voz y un sistema de reconocimiento de frecuencias (para el tarareo), así como tambien funcionalidades básicas de la aplicación.

En resumen, el reto principal reside en desarrollar el entorno de programación y hacer la integración de los algoritmos con el entorno como veremos posteriormente.

CAPÍTULO 3

Aspectos Teóricos

Para poder desarrollar los sistemas de reconocimiento tanto de frecuencia como de comandos de voz es importante entender la tecnología detrás de ambos sistemas. Es por ello que se ha revisado el estado del arte y se ha hecho una evaluación de las herramientas disponibles para determinar las técnicas a utilizar.

Los motivos detrás de dicha elección se describen en el siguiente capítulo, mientras que en este se procederá a la descripción de los mismos.

Por un lado, se deben entender los algoritmos de predicción y los elementos que van a permitir que estas predicciones puedan ser realizadas. De modo que empezaremos describiendo los MFCC (coeficientes del sonido) que serán determinantes para poder capturar información relevante del audio. Posteriormente los algoritmos kNN y CNN para realizar predicciones sobre dichos coeficientes.

3.1. MFCC

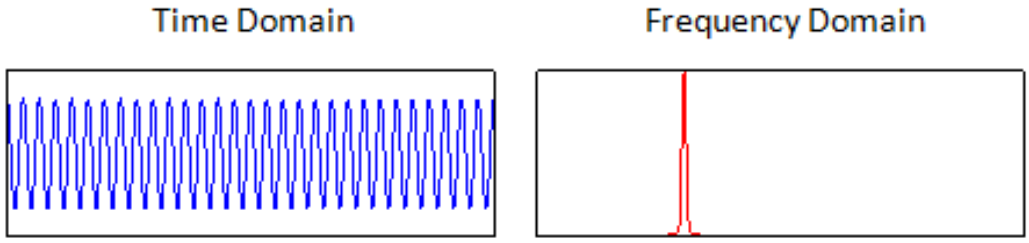
Para poder entender el significado de MFCC vamos a descomponer el acrónimo. MFCC o Mel Frequency Cepstral Coefficients en castellano se refiere a los Coeficientes Cepstrales de la Frecuencia de Mel. Sin embargo, es posible que no se entienda bien qué son los coeficientes cepstrales y es por ello que primero trataremos de entender el origen de la palabra cepstral.

La palabra cepstral no es más que la palabra *spectral* pero con las primeras cuatro letras invertidas. Es por ello que es importante entender que es el espectro del sonido. El espectro de un sonido es definido como la representación de la distribución de energía sonora de dicho sonido en función de la frecuencia. El espectro es importante porque la percepción auditiva del sonido es de naturaleza predominantemente espectral.

De esta manera se puede definir el cepstro como la información de la tasa de cambio en las bandas espectrales. Esto se podria entender como el espectro del espectro y es por ello que para su abreviación se decidió simplificar con la palabra cepstro. En el análisis convencional de las señales temporales, cualquier componente periódico (por ejemplo, los ecos) aparece como picos en el espectro de frecuencias correspondiente (es decir, el espectro

de Fourier, que se obtiene aplicando una transformada de Fourier a la señal temporal). Esto puede verse en la Figura 3.1.

Figura 3.1 Ejemplo de la transformación de una frecuencia del dominio temporal al dominio de frecuencia.



Al tomar el logaritmo de la magnitud de este espectro de Fourier y, a continuación, volver a tomar el espectro de este logaritmo mediante una transformación del coseno, observamos un pico allí donde hay un elemento periódico en la señal temporal original. Dado que aplicamos una transformación sobre el propio espectro de frecuencias, el espectro resultante no está ni en el dominio de la frecuencia ni en el del tiempo, y de ahí que Bogert et al. [2] decidieran llamarlo dominio de la quefrency o cuefrencia. Y como se ha explicado previamente a este espectro del logaritmo del espectro de la señal temporal lo denominaron cepstrum o cepstro.

El Cepstro se introdujo por primera vez para caracterizar los ecos sísmicos resultantes de los terremotos.

Por otro lado, el tono es una de las características de una señal de voz y se mide como la frecuencia de la señal. La escala Mel es una escala que relaciona la frecuencia percibida de un tono con la frecuencia real medida. La escala de Mel se basa en la frecuencia para ajustarse más a lo que el oído humano puede oír (los seres humanos identifican mejor los pequeños cambios en el habla en las frecuencias más bajas). Esta escala se ha obtenido a partir de conjuntos de experimentos con sujetos humanos. Vamos a proceder a hacer una explicación intuitiva de lo que capta la escala mel.

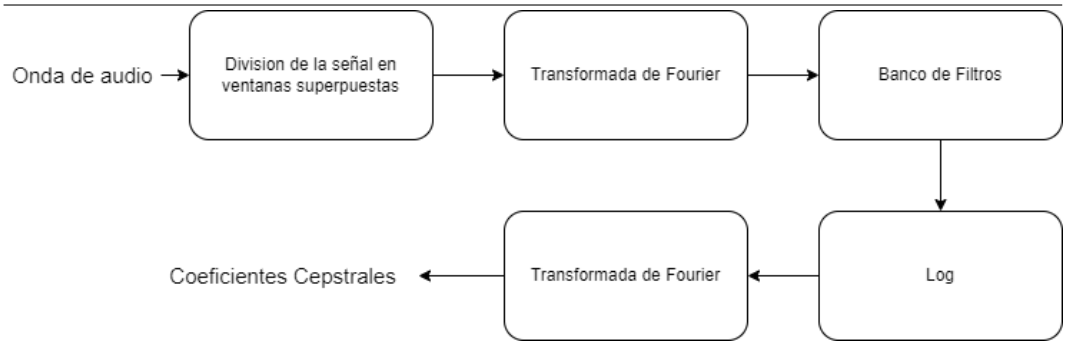
El rango de la audición humana es de 20Hz a 20kHz. Imaginando una melodía a 300 Hz, esto sonaría algo así como el tono de marcación estándar de un teléfono fijo. Ahora imaginando una melodía a 400 Hz sería un tono de marcación un poco más alto. Ahora imagina una señal de 900 Hz similar al sonido de retroalimentación de un micrófono y un sonido de 1kHz. La distancia percibida entre estos dos sonidos puede parecer mayor que la de los dos primeros, aunque la diferencia real sea la misma (100 Hz). La escala Mel trata de captar estas diferencias. Una frecuencia medida en Hertz (f) puede convertirse a la escala Mel mediante la siguiente fórmula:

$$Mel(f) = 2595 * log(1 + \frac{f}{700})$$

Cualquier sonido generado por el ser humano viene determinado por la forma de su tracto vocal (incluida la lengua, los dientes, etc.). Si esta forma puede determinarse correctamente, cualquier sonido producido puede representarse con precisión. La envolvente del espectro de potencia temporal de la señal de voz es representativa del tracto vocal y la MFCC (que no es más que los coeficientes que componen el cepstrum de frecuencias Mel)

representa con precisión esta envolvente. El diagrama de bloques de la Figura 3.2 es un resumen paso a paso de cómo se llega a los MFCC.

Figura 3.2 El diagrama de bloques es un resumen paso a paso de cómo se llega a los MFCC.



Una vez explicados los MFCC, el siguiente paso es entender cuales son los algoritmos que se han utilizado para hacer las predicciones. En este caso son algoritmos de aprendizaje supervisado.

El aprendizaje supervisado es un campo dentro del aprendizaje automático que para hacer predicciones, supone que partimos de un conjunto de datos etiquetado previamente, es decir, conocemos el valor del atributo objetivo para el conjunto de datos que disponemos.

En otras palabras tendremos cada uno de los audios con una etiqueta (*do, re, mi, fa, sol, la o si*) y dicho algoritmo aprenderá a base de ejemplos. Es una tarea supervisada, ya que cada audio debe ser etiquetado de forma manual para posteriormente darsela al algoritmo para su entrenamiento. Es importante aclarar que no son notas cantadas sino comandos de voz hablados.

3.2. KNN

A la hora de hacer las predicciones utilizando los MFCCs el primer algoritmo a implementar es kNN o k-Nearest-Neighbor. K-Nearest-Neighbor es un algoritmo supervisado que puede usarse para clasificar nuevas muestras (valores discretos) o para predecir (regresión, valores continuos). Sirve para clasificar valores buscando los puntos de datos “más similares” (por cercanía) aprendidos en la etapa de entrenamiento y haciendo conjeturas de nuevos puntos basado en esa clasificación.

En K-Nearest Neighbor la “K” significa la cantidad de “puntos vecinos” que tenemos en cuenta en las cercanías para clasificar los “n” grupos.

Los pasos a seguir para crear un algoritmo de kNN son los siguientes:

- Calcular la distancia entre el ítem a clasificar y el resto de ítems del dataset de entrenamiento.
- Seleccionar los “k” elementos más cercanos (con menor distancia, según la función que se use)
- Realizar una “votación de mayoría” entre los k puntos: los de una clase/etiqueta que «dominen» decidirán su clasificación final.

En nuestro caso será utilizado para recibir los coeficientes cepstrales de mel y clasificar los audios en una de las categorías correspondientes.

3.3. CNN

Así como kNN es un algoritmo bastante básico, la segunda aproximación será un acercamiento algo más complejo mediante aprendizaje profundo con redes neuronales convolucionales.

La CNN es un tipo de red neuronal artificial basada en el córtex visual del ojo humano para identificar distintas características en las entradas que en definitiva hacen que pueda identificar objetos y "ver". Para ello, la CNN contiene varias capas ocultas especializadas y con una jerarquía: esto quiere decir que las primeras capas pueden detectar líneas, curvas y se van especializando hasta llegar a capas más profundas que reconocen formas complejas como un rostro o la silueta de un animal.

Un aspecto esencial a entender es el motivo de por qué utilizar CNN, que es un algoritmo típico de visión por computador para predicción de sonidos. El motivo principal es que los MFCCs se pueden representar forma de matriz creando una imagen. Tendríamos 13 coeficientes en la vertical de la matriz y 31 en la horizontal, que corresponderían a la línea temporal, ya que se cogen 31 muestras en 1 segundo de comando de voz. Es por ello que usar CNNs es una buena opción, ya que en el fondo se está trabajando información que se podría representar como imágenes.

Para entenderlo mejor comencemos por un ejemplo. La red toma como entrada los píxeles de una imagen. Si tenemos una imagen con apenas 28×28 píxeles de alto y ancho, eso equivale a 784 neuronas. Y eso es si sólo tenemos 1 color (escala de grises). Si tuviéramos una imagen a color, necesitaríamos 3 canales (red, green, blue) y entonces usaríamos 28x28x3 = 2352 neuronas de entrada. Esa es nuestra capa de entrada. Para continuar con el ejemplo, supondremos que utilizamos la imagen con un solo color.

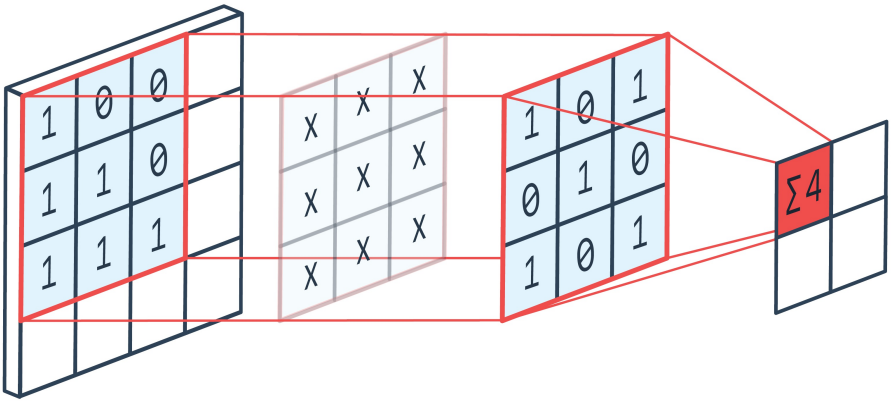
Ahora comienza el procesado distintivo de las CNN. Es decir, las llamadas convoluciones: Estas consisten en tomar grupos de píxeles cercanos de la imagen de entrada e ir operando matemáticamente (producto escalar) contra una pequeña matriz que se llama kernel. Ese kernel supongamos de tamaño 3×3 pixels recorre todas las neuronas de entrada (de izquierda-derecha, de arriba-abajo) y genera una nueva matriz de salida, que en definitiva será nuestra nueva capa de neuronas ocultas.

No aplicaremos un solo kernel, sino que tendremos muchos kernel (su conjunto se llama filtros). Por ejemplo en esta primera convolución podríamos tener 32 filtros, con lo cual realmente obtendremos 32 matrices de salida (este conjunto se conoce como "feature mapping"), cada una de 28x28x1, dando un total del 25.088 neuronas para la primera capa oculta de neuronas.

A medida que vamos desplazando el kernel, vamos obteniendo una nueva imagen filtrada por el kernel. En esta primera convolución, siguiendo con el ejemplo anterior, es como si obtuviéramos 32 imágenes filtradas nuevas. Estas imágenes nuevas lo que están "dibujando" son ciertas características de la imagen original. Esto ayudará en el futuro a poder distinguir un objeto de otro. Este procedimiento de puede ver en la Figura 3.3.

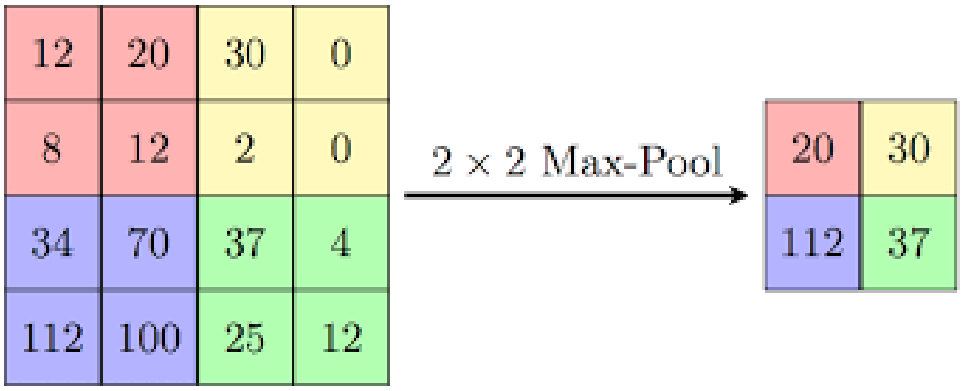
Ahora viene un paso en el que reduciremos la cantidad de neuronas antes de hacer

Figura 3.3 En la imagen se describe el ejemplo de una convolución. Un kernel esta pasando por los primeros 9 valores y rellenando un píxel de la reconstrucción.



una nueva convolución. El motivo reside en que si hiciéramos una nueva convolución a partir de esta capa, el número de neuronas de la próxima capa sería enorme. Para reducir el tamaño de la próxima capa de neuronas haremos un proceso de subsampling en el que reduciremos el tamaño de nuestras imágenes filtradas pero en donde deberán prevalecer las características más importantes que detectó cada filtro. Hay diversos tipos de subsampling pero en este caso y a lo largo del trabajo de fin de máster se utilizará el max pooling como se describe en la Figura 3.4.

Figura 3.4 Ejemplo de max pooling donde por cada cuatro píxeles, el píxel de mayor valor es extraido.



La primera convolución es capaz de detectar características primitivas como lineas o curvas. A medida que hagamos más capas con las convoluciones, los mapas de características serán capaces de reconocer formas más complejas, y el conjunto total de capas de convoluciones podrá “ver”.

Para terminar, tomaremos la última capa oculta a la que hicimos subsampling, que se dice que es tridimensional por tomar la forma -en nuestro ejemplo- 3x3x128 (alto,ancho,mapas) y la aplanamos, esto es que deja de ser tridimensional, y pasa a ser una capa de neuronas tradicionales, de las que ya conocíamos. Por ejemplo, podríamos aplanar (y conectar) a una

nueva capa oculta de 100 neuronas feedforward.

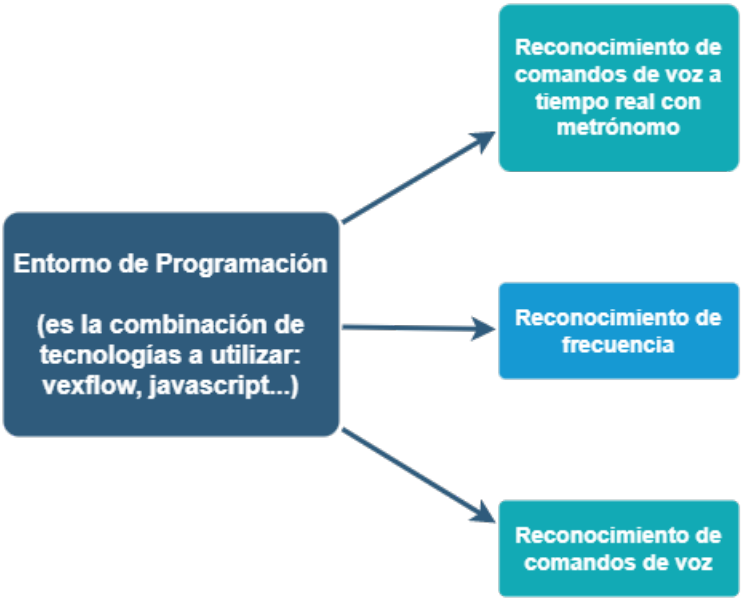
Entonces, a esta nueva capa oculta tradicional, le aplicamos una función llamada Softmax que conecta contra la capa de salida final que tendrá la cantidad de neuronas correspondientes con las clases que estamos clasificando. Si clasificamos perros y gatos, serán dos neuronas. Si es el dataset Mnist numérico serán 10 neuronas de salida. Si clasificamos coches, aviones o barcos serán 3, etc.

CAPÍTULO 4

Desarrollo de Software Inicial

El proyecto comienza en junio de 2021 en bidasoa activa y el primer reto consiste en crear el entorno necesario para posteriormente poder crear e integrar los sistemas de inteligencia artificial necesarios. Es por ello que uno de los retos más grandes e importantes es de ingenieria de software. Es un problema muy complejo, ya que hay muchas piezas a integrar y no precisamente de fácil integración, ya que cada una es bastante específica. En caso de conseguirlo su resolución conlleva la escalabilidad del sistema y un paso muy grande hacia obtener un producto mínimo viable. En la Figura 4.1 se puede observar un esquema de lo que se espera desarrollar y cómo los sistemas de inteligencia artificial y reconocimiento de frecuencias están unidos al entorno de desarrollo o sistema.

Figura 4.1 El entorno de programación compuesto por vexflow, javascript y demás tecnologías que se ocultarán como secreto empresarial y su conexión con los sistemas de reconocimiento de comandos de voz, reconocimiento de frecuencia y reconocimiento de frecuencia a tiempo real.



4.1. Entorno de programación

Los siguientes puntos son los requerimientos del entorno a desarrollar y el motivo por el que es un reto de alta complejidad.

- Todo lo codificado debe poder **funcionar** en **moviles, tablets y ordenadores**.
- Todo el código generado debe ser **directamente transferible** sin muchas complicaciones a cualquier dispositivo para poder actualizar la aplicación de forma rápida y poder evaluar el comportamiento de los usuarios.
- Para no reinventar la rueda y poder imprimir notas, pentagramas y elementos musicales en pantalla, es imprescindible utilizar alguna **librería de impresión musical** ya existente.
- Todo el sistema debe ser compatible con la **grabación de microfono utilizando el mismo método en todos los dispositivos**.
- Debe tener un **sistema de inteligencia artificial** que funcione en todos los dispositivos **sin conexión a internet, rápido y de poco peso**.
- Se debe **poder integrar** un **sistema de reconocimiento de frecuencias** para el sistema de tarareado.
- La **aplicación** con todo lo descrito anteriormente debe ser **muy rápida**, ya que si no el flujo de composición se rompería. En otras palabras, el sistema debe hacer inferencias de forma veloz para que el compositor no tenga que esperar entre comandos.
- Debe tener la posibilidad de **mandar partituras de un dispositivo a otro**.

Se puede observar que el reto no es para nada trivial; de hecho es muy complejo. La integración de todos estos sistemas ha durado dos meses en los que se ha trabajado a jornada parcial en el proyecto.

Las diferentes piezas tecnológicas elegidas para crear el sistema completo han sido las siguientes: **React Native** debido a la versatilidad que ofrece tanto por la capacidad de desarrollo multiplataforma como por la facilidad de utilizar el mismo código a distintos dispositivos. Además react garantiza la rapidez de la aplicación, ya que hace un uso muy eficiente del DOM web. Es un sistema en el que se basan la mayoría de aplicaciónes modernas como Instagram, Facebook y demás. Por otro lado, la librería de impresión de pentagramas utilizada es **Vexflow**, una librería desarrollada en 12 años por más de 70 ingenieros. **Node.js** ha sido elegido como la pieza clave para la gestión de los datos en la nube.

Después de dos meses de integración a jornada parcial, uno de los engranajes imprescindibles, al no ser compatible, hace que no se pueda continuar el desarrollo y todo el código debe ser deshechado. Aparentemente vexflow, a pesar de estar desarrollado también para react native, está en una fase muy temprana y la integración total resulta imposible.

Por estos motivos y gracias a uno de los mayores contribuidores de la librería vexflow que decidió ayudar en la elección de las piezas tecnológicas a integrar, en los cuatro meses posteriores trabajando de nuevo a jornada parcial se ha podido conseguir un sistema uniendo todos los requisitos.

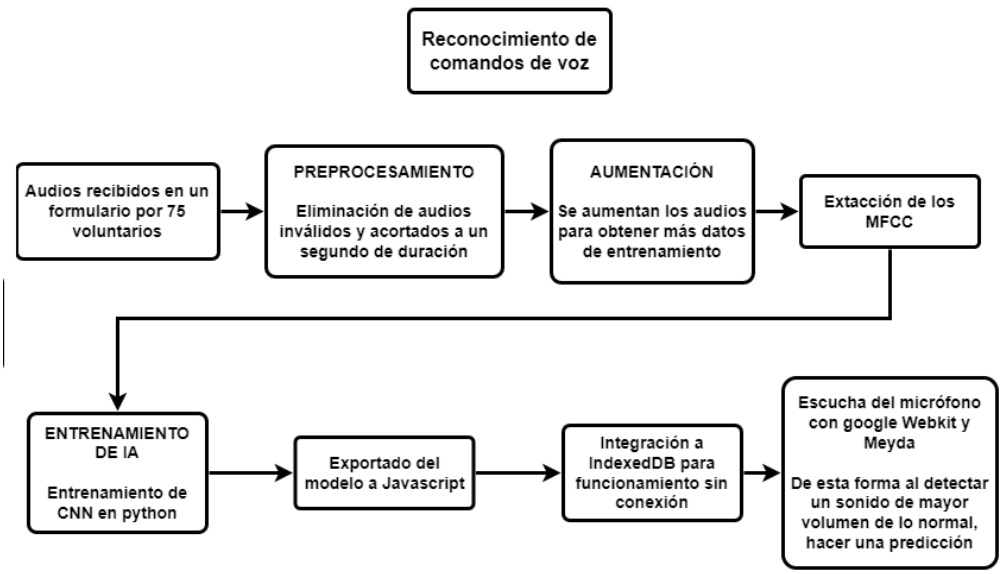
Es por ello que no mencionaré la totalidad de las tecnologías empleadas, ya que dada la complejidad, será tratado como secreto empresarial. Posteriormente, alguna de las tecnologías será mencionada pero no la totalidad de ellas.

Después de 6 meses de desarrollo trabajando en el proyecto 4 horas diarias, la parte más complicada ha sido resuelta.

4.2. Sistemas de reconocimiento de comandos y frecuencia

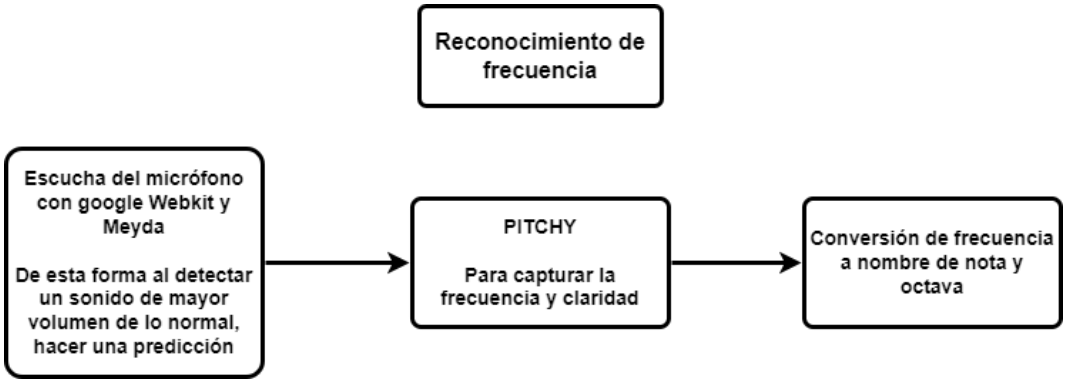
Como se ha descrito previamente, el entorno de programación tiene ciertos requisitos que se extienden a los sistemas de reconocimiento de comandos y de frecuencia. En las siguientes figuras se describe el esquema interno de cada uno de los sistemas de reconocimiento. En la Figura 4.2 se describe el sistema de reconocimiento de comandos de voz.

Figura 4.2 El sistema de reconocimiento de comandos de voz comienza por una muestra de 75 audios, un preprocesamiento, aumentación, extracción de los MFCC, entrenamiento de la red neuronal en python, extracción a javascript y su correspondiente integración.



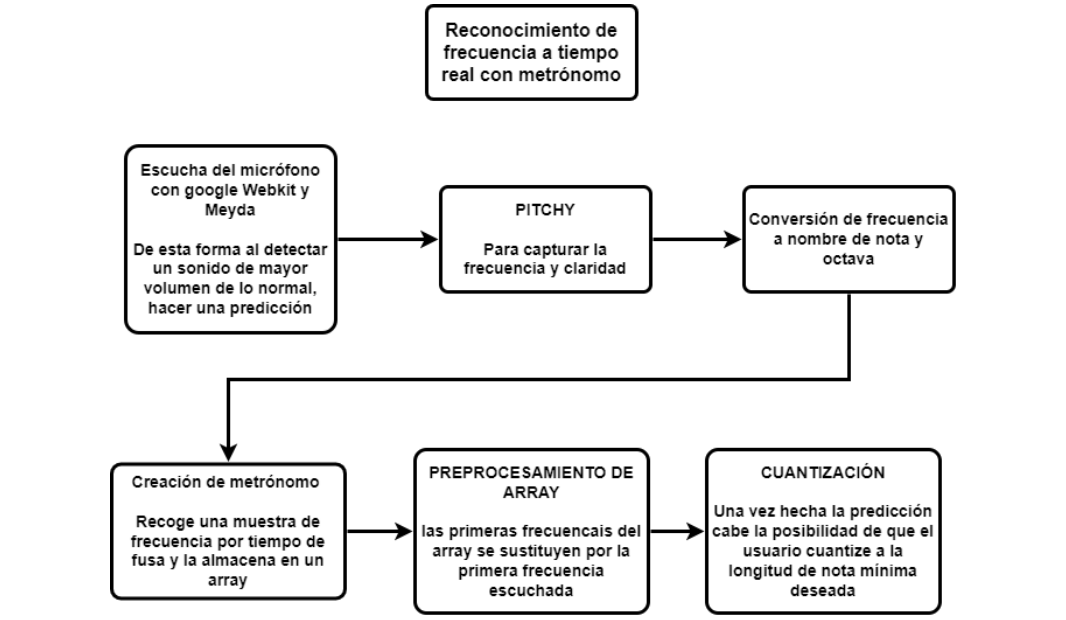
En la Figura 4.3 se describe el sistema de reconocimiento de frecuencias.

Figura 4.3 El reconocimiento de frecuencias comienza por la lectura de los audios en la aplicación, pitchy los procesa y se convierten a la nota y octava correspondiente para la posterior impresión en el pentagrama.



Finalmente, en la Figura 4.4 se describe el sistema de reconocimiento a tiempo real con metrónomo.

Figura 4.4 El sistema de reconocimiento de frecuencias a tiempo real con metronomo tiene el mismo funcionamiento que el sistema de reconocimiento de frecuencias pero con la creación de metrónomo y el procesamiento de array correspondiente.



Estos esquemas serán de gran utilidad en los capítulos posteriores a la hora de entender mejor cada uno de los sistemas ya que hay un capítulo reservado para cada uno de ellos.

CAPÍTULO 5

Preparación y obtención de los datos

Una vez desarrollado el entorno, la primera tarea a realizar consiste en desarrollar un sistema de inteligencia artificial capaz de diferenciar entre do, re, mi, fa, sol, la, si y $ruido$. Ahora, es importante mencionar que esta tarea no es nada trivial por varios requerimientos del sistema descritos previamente. De todos los requerimientos descritos nos centraremos en los que el sistema de inteligencia artificial debe cumplir.

- El sistema de inteligencia artificial debe ser un algoritmo que se ejecute en el propio móvil para poder componer sin conexión.
- El sistema debe pesar poco para poder entrar en el cualquier dispositivo.
- El sistema debe poder funcionar de forma veloz en cualquier dispositivo.
- El sistema debe ser lo más preciso posible.
- El sistema debe poder integrarse en javascript.

Por estos motivos, herramientas como *IBM Watson* o *Google Webkit for Speech Recognition* no son buenas opciones, ya que por un lado se ejecutan en la nube y por otro el coste de cómputo del tratamiento de todo el lenguaje natural es innecesario y solamente ralentizarían la predicción. De modo que la mejor opción es crear un modelo de inteligencia artificial desarrollado específicamente para la tarea.

Además, hay que hacer frente a otro problema. A día de hoy no existe un solo dataset de los siguientes comandos: do, re, mi, fa, sol, la, si. Por ello, que antes de poder proceder a construir el modelo de inteligencia artificial de alguna forma se debe conseguir un dataset con dichos comandos.

5.1. Obtención del dataset

Se ha recolectado un total de 75 muestras de dos segundos de duración. Por un lado 40 personas con origen español y con una distribución geografica bien distribuida. Por otro lado, para poder obtener muestras de paises latinoamericanos se ha distribuido una encuesta por medio de la plataforma reddit y se han obtenido 35 muestras de gente latinoamericana.

Para una buena obtención de datos lo ideal es poder replicar el entorno en el que se ejecutará el sistema; en otras palabras, lo ideal sería que los audios sean grabados en los dispositivos en los que el sistema se ejecutaría. Para ello se ha desarrollado un formulario para recoger los diferentes comandos de voz en el que se graba desde el propio dispositivo del usuario en un formato de dos segundos por audio.

Pensando en el desarrollo futuro, no solamente se han recogido los comandos mencionados previamente, sino que cada uno de los formularios contiene los siguientes comandos: *do, re, mi, fa, sol, la, si, silencio, borrar, sostenido, bemol, puntillo, acorde, arriba, abajo, atras, adelante, editar, grabar, play, pausa, stop, ligar, segunda, tercera, cuarta, quinta, sexta, septima, octava.*

Es importante que para obtener una buena muestra, dichas grabaciones sean independientes e identicamente distribuidas (I.I.D) para que puedan representar de la manera menos sesgada posible a la población. Es por ello que las muestras recogidas son de personas desde los 20 hasta los 75 años, tanto hombres como mujeres y de diferentes paises hispanohablantes.

5.2. Preparación de los datos

Una vez obtenidos los datos hay que preprocesarlos para que el modelo pueda hacer las mejores inferencias posibles. Para ello, la primera tarea realizada ha consistido en escuchar los audios uno a uno para poder filtrar los datos válidos de los no válidos. Posteriormente se ha utilizado google colab (una plataforma desarrollada por google para la ejecución de código python en la nube) para el preprocesamiento.

La primera tarea consiste en homogeneizar las frecuencias de todos los audios para que a la hora de pasarle los datos a nuestro algoritmo el input siempre sea el mismo. A la hora de elegir un sample rate es importante que no sea demasiado grande para evitar demasiado peso en la aplicación e igualmente con tamaños muy pequeños ya que se puede perder la calidad. De modo que el sample rate al que se han tranformado los audios es a 16000 Hz.

Posteriormente, en lugar de obtener audios de dos segundos para poder hacer inferencias más rápidas se ha decidido acortar los audios a un segundo. Para ello se ha creado un algoritmo para detectar el silencio o silencio relativo, ya que en muchos casos hay ruido de fondo, y cortar todos los audios automáticamente.

Posteriormente, todos los audios se han comprobado de forma manual para verificar que no hay ningún audio cortado.

5.3. Data augmentation

En general, cuantos más datos tengan, los algoritmos aprenden mejor. Es por ello que cuantos más audios sean recolectados mejor será la precisión del algoritmo. Para lograr este objetivo, los audios serán aumentados. Los siguientes tipos de aumentación de audios serán utilizados:

- Ruido blanco mezclado al audio original
- Cambio de pitch
- Cambio de pitch + Ruido blanco añadido al audio original

Toda la aumentación de datos se ha hecho en python y posteriormente descargado para poder entrenar en javascript con dichos audios.

En total disponiendo de 75 grabaciones por comando y tres aumentaciones diferentes por audio se han obtenido un total de 300 muestras por comando de voz.

CAPÍTULO 6

Desarrollo del Sistema de Inteligencia Artificial

Una vez obtenidos los datos y procesarlos para alimentar un modelo, el siguiente paso consiste en definir el preprocesamiento necesario para posteriormente pasarle los datos al modelo.

En este caso, al tener la limitación de tener que utilizar javascript, no se podrán utilizar todos los recursos disponibles en la literatura sino los disponibles en javascript.

A continuación, se describen los recursos disponibles en javascript y los problemas principales.

6.1. Integración con javascript

El principal problema observado a la hora de desarrollar un sistema de reconocimiento de comandos de voz para javascript, es que es complicado aplicar la tecnología del estado del arte. Al revisar el estado del arte [3] [4] podemos observar que la mayor precisión se obtiene mediante el uso de espectrogramas del sonido y posteriormente aplicando redes neuronales convolucionales o algoritmos como kNN [5] .

El problema reside en que para javascript apenas hay desarrollada una librería aceptable que permita crear espectrogramas del sonido y guardarlos. Es por ello que las dos únicas opciones son desarrollar una libreria que sea capaz de hacerlo o tratar de buscar otro método. Dado que desarrollar una librería podría llevar mucho tiempo, es importante revisar la literatura y ver si a pesar de no conseguir resultados del estado del arte se pueden obtener buenos resultados. Es importante tener en cuenta que el objetivo no es tener una precisión del 100 %, sino tratar de buscar un sistema pueda resolver la tarea con buena precisión y con los requisitos previamente descritos.

Al buscar en la literatura se puede observar cómo previo a la creación de espectrogramas muchos papers se centraban en utilizar los coeficientes cepstrales del sonido tambien conocidos como MFCC [6], [7], [8], [9]. Es por ello que finalmente el descubrimiento de la librería Meyda [10] fue clave para el desarrollo del proyecto.

Meyda es una librería creada por Rawlinson et al. [10] y se puede integrar a javascript de forma sencilla.

Una vez obtenida una forma de procesar los datos, la siguiente tarea consiste en definir el algoritmo o modelo a utilizar para poder hacer inferencias. Gran cantidad de papers [5] [8] [11] [9] a la hora de utilizar los coeficientes cepstrales utilizan el conocido algoritmo de k Nearest Neighbours (kNN) y los resultados obtenidos son muy buenos. Es por ello que el siguiente paso consiste en montar y probar dicho algoritmo.

6.2. kNN

De nuevo es importante tener en cuenta que al estar desarrollando el proyecto en javascript los recursos son limitados. Es por ello que los pasos a seguir son: obtener una librería de kNN para javascript y extraer los MFCC en javascript.

El primer paso consiste en crear un código para extraer todos los MFCC de los audios a clasificar y extraerlos en un archivo excel para posteriormente poder utilizarlos en un kNN. El excel consiste en tener en cada línea todos los coeficientes cepstrales y al final el número de la clase a la que pertenece el audio.

La extracción en un excel es importante ya que en javascript no hay librerías de kNN que permitan hacer cross validation en el propio algoritmo y es por ello que esto facilita la segmentación en bloques de train y test y de esta manera programar nosotros mismos el cross-validation.

Una vez tenemos los datos y el algoritmo procedemos a desarrollar el código incluyendo cross validation y probando diferentes tipos de ventaneado y de tamaño de buffer para alimentar el algoritmo. En concreto se ha probado con ventaneado de $hanning$, $hamming$, $blackman$ y $sine$. Y los diferentes tamaños de buffer que indican en cuántos bloques se separará el audio. Para más información sobre los tipos de ventanas el paper de Podder et al. [12] introduce y hace un análisis comparativo entre dichas ventanas.

Los resultados obtenidos son los siguientes:

Utilizando un tamaño de buffer de 256, se obtiene la tabla 6.1. El tamaño de buffer indica en en cuantas secciones en la que se dividirá el audio para así extraer los MFCC, es decir, por cada 256 valores del audio se extraeran 13 coeficientes de MFCC.

k	Precisión Hanning	Precisión Hamming	Precisión Blackman	Precisión Sine
1	0,52	0,52	0,56	0,51
2	0,54	0,54	0,54	0,5
3	0,52	0,53	0,55	0,5
4	0,52	0,51	0,54	0,46
5	0,5	0,49	0,53	0,45
6	0,51	0,50	0,52	0,46
7	0,50	0,49	0,52	0,44
8	0,51	0,50	0,52	0,46
9	0,50	0,49	0,51	0,48

Tabla 6.1 Tabla de resultados de kNN con un tamaño de buffer de 256.

La tabla 6.2 se obtiene con un tamaño de buffer de 512. Se puede observar como la mayor puntuación se obtiene mediante Hamming y es de 53 %.

k	Precisión Hanning	Precisión Hamming	Precisión Blackman	Precisión Sine
1	0,49	0,50	0,48	0,48
2	0,5	0,51	0,48	0,5
3	0,48	0,51	0,46	0,46
4	0,49	0,51	0,46	0,46
5	0,50	0,53	0,47	0,48
6	0,5	0,51	0,47	0,48
7	0,49	0,49	0,47	0,47
8	0,46	0,47	0,47	0,47
9	0,48	0,48	0,46	0,48

Tabla 6.2 Tabla de resultados de kNN con un tamaño de buffer de 512.

Para acabar, la tabla 6.3 corresponde con el tamaño de buffer de 1024. De nuevo, la mayor puntuación se obtiene con Hamming y es de 55 %.

k	Precisión Hanning	Precisión Hamming	Precisión Blackman	Precisión Sine
1	0,51	0,55	0,51	0,54
2	0,51	0,55	0,50	0,54
3	0,51	0,53	0,51	0,52
4	0,50	0,52	0,50	0,53
5	0,48	0,51	0,5	0,51
6	0,49	0,52	0,5	0,51
7	0,48	0,49	0,5	0,52
8	0,48	0,48	0,48	0,52
9	0,50	0,51	0,48	0,51

Tabla 6.3 Tabla de resultados de kNN con un tamaño de buffer de 1024.

La tabla 6.4 corresponde con el resumen de los mejores resultados de las tablas anteriores.

Tamaño de buffer	Hanning	Hamming	Blackman	Sine
Tamaño de buffer de 256	0,54	0,54	0,56	0,51
Tamaño de buffer de 512	0,50	0,53	0,48	0,5
Tamaño de buffer de 1024	0,51	0,55	0,51	0,54

Tabla 6.4 Resumen de tablas de resultados de kNN. El mejor resultado se obtiene con blackman con un 56 % de precisión.

Se puede observar en la última tabla cómo el mejor resultado se obtiene con Blackman con un 56 % de precisión y un tamaño de buffer de 256. Los resultados distan mucho del ideal, la mejor puntuación es de un 56 %. Para asegurarse tambien se ha ejecutado kNN en R desde la librería caret para corroborar que no es problema de la implementación del algoritmo en javascript. En R el accuracy máximo obtenido ha sido del 54 %.

La hipótesis principal de este mal funcionamiento en diferencia a la mayoría de papers es que probablemente el ruido de fondo sumado con la poca cantidad de datos haga difícil para kNN diferenciar los audios. Es por ello que se puede tratar de hacer extracción de características de manera manual o tratar de cambiar de algoritmo para que lo haga de manera automática. Para solventar ese problema, utilizaremos redes neuronales convolucionales.

Otra posibilidad es que kNN al no ser un algoritmo especialmente desarrollado para el análisis de series temporales no las esté teniendo en cuenta. Ciertos papers [7] proponen utilizar otras técnicas como el dynamic time warping para poder comparar las series temporales utilizando el mismo kNN.

6.3. Conclusión de KNN

Dado que kNN ha obtenido malos resultados, la hipótesis de estos resultados se basa en que a pesar de ser un buen clasificador, el ruido de fondo puede ser un problema. Es bien sabido que las redes neuronales convolucionales son capaces de hacer lo que se conoce como feature extraction de forma automática y de esa forma tratar de distinguir con mayor precisión lo que sería el ruido y el comando de voz. Es un problema que se ha descrito en [13] [3] y aparentemente con buen resultado.

Por otro lado, dado que otro de los potenciales problemas es que no solamente hace falta extraer las características descritas previamente sino también que se entienda el contexto, es decir, la serie temporal. En cierta medida, las redes neuronales convolucionales lo consiguen como se describe en [13] [3].

Sin embargo, de cara a futuro se deberían explorar mejores formas de explotar el análisis de los MFCC como serie temporal ya que información valiosa se perdería de otra manera. Es algo que en el paper de Douglas et al. [3] se describe. De hecho, en el paper se utilizan biLSTM para capturar la serie temporal en ambas direcciónes.

Otros papers también referencian la alta precisión que se puede obtener utilizando MFCCs con redes neuronales convolucionales [14], [6], [15] y es por todos estos motivos que se ha decidido proceder a su implementación.

Para finalizar, añadir que en caso de funcionar de forma correcta puede ser interesante utilizar LPC o linear predictive coding dado que en combinación con MFCCs se ha visto [14] que el funcionamiento es muy bueno para audios de mala calidad.

6.4. CNN

Por los motivos expuestos previamente, las redes neuronales convolucionales son un tipo de algoritmo prometedor para la solución de este problema. Es por ello que se va a proceder a la explicación tanto de la obtención de los MFCC como a la explicación de la arquitectura concreta utilizada.

El procedimiento consiste en desarrollar el modelo en python y exportarlo a javascript mediante tensorflow.js

6.4.1. Obtención de los MFCC

Para la extracción de los MFCC disponemos de dos herramientas: Meyda, la herramienta desarrollada en javascript y Librosa, la librería desarrollada para python. Teniendo en cuenta que la red neuronal convolucional será desarrollada en python y posteriormente extraida a javascript, lo lógico, sería utilizar librosa ya que se podría hacer todo en el mismo código, tanto la obtención de los MFCC como el entrenamiento del modelo.

Sin embargo, a la hora de hacer predicciones en la aplicación final, éstas se harán mediante Meyda ya que es la única opción disponible en javascript. De modo que las dos opciones disponiblen son: extraer MFCCs desde Librosa y luego hacer predicciones en la aplicación con Meyda o hacer todo con Meyda.

La elección final ha sido realizar todo con Meyda ya que, a veces, puede haber discrepancias a la hora de extraer los componentes del sonido desde diferentes librerías y podría implicar tener un modelo sesgado.

Este procedimiento, implica el desarrollo de un código para extraer todos los MFCC de los audios a clasificar utilizando javascript y extraerlos en un archivo excel para posteriormente poder importarlos desde python.

6.4.2. Arquitectura de CNN

A la hora de desarrollar la arquitectura de la CNN se debe tener en cuenta que el modelo debe hacer inferencias rápido y debe pesar poco para poder integrarla en todo tipo de dispositivos.

Es por ello que la arquitectura elegida no debe ser grande y muchos modelos del estado del arte deben ser descartados como ResNet, MobileNet... Muchos de estos modelos tienen una cantidad muy grande de parametros, lo que los hace pesados y con una latencia que para el objetivo final no es la ideal.

La arquitectura utilizada es mucho más pequeña y consiste en las siguientes capas como se puede observar en la Figura 6.1. Una capa de convolución 2D con 32 filtros de 4x4, seguido de batch normalization, de nuevo se repite el mismo proceso un max pooling con un tamaño de pool de 2x2 y un dropout de 0,2. Todo lo explicado hasta ahora se repite de la misma manera pero con un tamaño de filtro de 3x3. A continuación, una capa convolucional de 128 filtros de 3x3 y max pooling y dropout iguales que los descritos previamente. Para acabar, se realiza una operación de aplanamiento o flatten y se crea una capa Dense de 128 neuronas, una capa de dropout de 0.5 y finalmente una ultima capa de 8 neuronas con softmax que serán las predictoras de *do, re, mi, fa, sol, la, si y ruido.*

6.4.3. Resultados

La arquitectura desarrollada es una arquitectura más pequeña y rápida que los modelos del estado del arte como ha sido explicado previamente. De hecho, pesa menos que una imagen ($0,5$ megabytes) y hace inferencias muy rápidas incluso en móviles con procesadores lentos.

Hay que tener en cuenta que debido a las limitaciones de javascript se esta trabajando con MFCCs en lugar de espectrogramas y con redes neuronales convolucionales. En la

Figura 6.1 Arquitectura de la red neuronal convolucional.

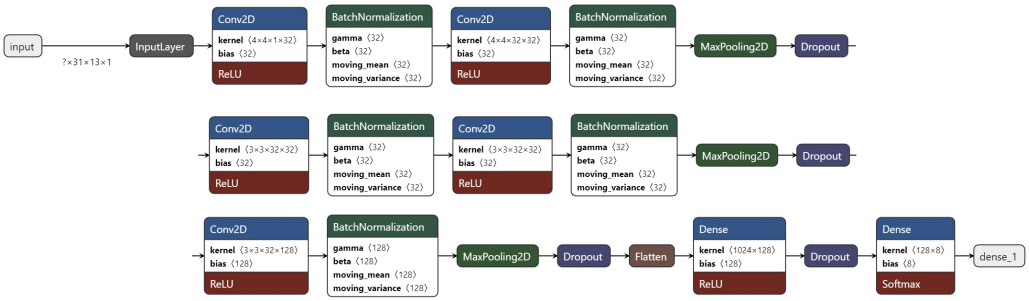
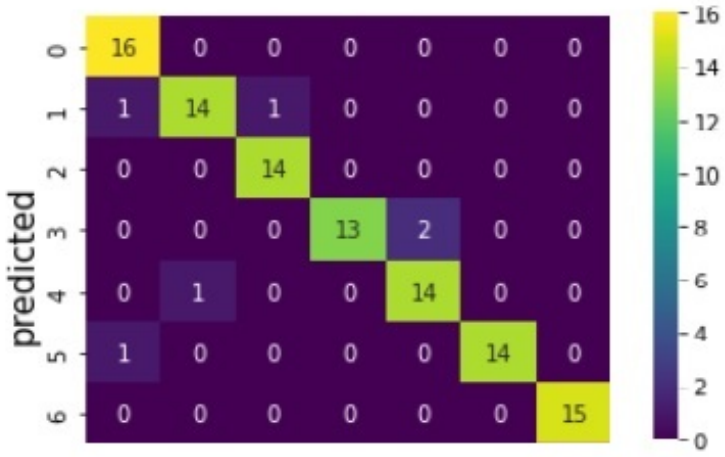


Figura 6.2 se puede observar la matriz de confusión en la que se indica el número de aciertos y fallos de la red neuronal asi como tambien la precisión.

El accuracy obtenido en el test set es de $92,5\,\%$.

Figura 6.2 Matriz de confusión en el test set.



Se puede confirmar que las redes neuronales convolucionales como se ha descrito en la literatura son efectivas a la hora de clasificar comandos de voz.

CAPÍTULO 7

Desarrollo del Sistema de Reconocimiento de Frecuencias

Para realizar el reconocimiento de frecuencias al igual que en los comandos de voz, el primer paso debe ser investigar que hay realizado en la literatura, y, en caso de que haya algún tipo de código libre, pasar a implementarlo.

Es importante recalcar que este trabajo de fin de máster además de ser un trabajo de investigación es además un reto de desarrollo e integración. Es por ello que encontramos en la literatura a "pitchy"que es una herramienta basada en el paper *a smarter way to find pitch* [16].

7.1. Funcionamiento

Pitchy permite el reconocimiento de frecuencias a tiempo real. Aparte de obtener la frecuencia cada número de milisegundos especificado por el usuario, también cuenta con la posibilidad de obtener el grado de incertidumbre de la frecuencia escuchada. De esta forma se pueden filtrar solamente frecuencias con una alta tasa de confianza.

Lo que en el paper llama MPM o McLeod Pitch Method funciona sin utilizar un filtro de paso bajo, por lo que puede trabajar con sonidos con altas frecuencias armónicas, como como un violín, y puede mostrar cambios de tono de forma fiable. El MPM funciona bien sin ningún tipo de procesamiento posterior para corregir el tono. El postprocesamiento es un requisito habitual en otros detectores de tono. Esto es algo importante, ya que agiliza el análisis y de cara a la velocidad de la aplicación es un aspecto clave.

El sistema dispone de una opción para igualar los niveles de la señal a la sensibilidad del oído interno. Se utilizan curvas de ecualización estándar, que tienden a reducir las frecuencias bajas que no se perciben bien en relación con las frecuencias alrededor de 3700 Hz que se escuchan mejor.

Esto ayuda a pasar de una estimación de la frecuencia fundamental hacia algo más correlacionado con el tono. Los algoritmos de afinación existentes que utilizan el dominio de Fourier sufren una fuga espectral. Sin embargo, Pitchy soluciona este problema. Para

entender la fuga espetral y cómo Pitchy lo soluciona podemos decir que si una señal pasa a través de un sistema lineal, la salida no tendrá ninguna frecuencia que no estuviera presente en la entrada. Sin embargo, si el sistema no es lineal, la salida tendrá nuevas frecuencias que no están presentes en la entrada original. Estas nuevas frecuencias son la fuga espectral. Esto se debe a que la ventana elegida en los datos no siempre contiene un número entero de periodos de la señal.

La solución habitual a esto es reducir la fuga utilizando una función de ventana, suavizando los datos en los bordes de la ventana. Un problema similar ocurre en algunos métodos del dominio del tiempo, como la autocorrelación, donde una ventana que contiene un número fraccionario de periodos, produce máximos en ubicaciones variables dependiendo de la fase de la entrada. Sin embargo, MPM introduce un método de normalización que se ve menos afectado por los problemas de bordes, al tener en cuenta términos en cada lado de la correlación por separado.

Como conclusión podemos decir que *pitchy* es la implementación de MPM que es un algoritmo efectivo, de procesamiento rápido, sin mucha pérdida de información y fiable. Es por estos motivos que se ha decidido implementarlo.

CAPÍTULO 8

Integración con el Sistema

En los dos capitulos previos se han presentado los algoritmos de reconocimiento de comandos de voz y de reconocimiento de frecuencias (en este caso solamente se ha utilizado uno existente). Ahora, debido a que el objetivo del proyecto es poder integrarlo en todo tipo de dispositivos, se describirá la implementación con el sistema desarrollado.

8.1. Integración del sistema de comandos de voz

El objetivo de la integración consiste en hacer predicciones cada vez que se escuche un sonido por encima del umbral. En caso de que lo escuchado sea ruido, se clasificará como ruido y en caso de que sea uno de los comandos se imprimirá la nota correspondiente en pantalla.

Como se ha comentado previamente, todos los sistemas deben de funcionar sin conexión a internet e integrarse en javascript. Es por ello que mediante Tensorflow.js se convierte y descarga el modelo a dos archivos. Un fichero JSON y uno BIN. El fichero JSON contiene la arquitectura de la red neuronal mientras que el BIN contiene los pesos de cada neurona tras haber realizado el entrenamiento previamente en python.

Teniendo en cuenta que tenemos una plataforma web, para poder integrar el sistema sin conexión, se ha utilizado indexedDB que permite, una vez descargado un modelo desde HTTP, guardarlo en la caché del navegador y de esta forma poder acceder sin conexión al modelo.

Una vez tenemos el modelo cargado el siguiente paso es capturar el micrófono en tiempo real y poder realizar inferencias. Es imprescindible que en todos los navegadores y dispositivos funcione de la misma manera para evitar desarrollo innecesario. Es por ello que se utiliza *webkit* para capturar el microfono en cualquier navegador (Safari, Chrome...).

Posteriormente abrimos un stream de audio para que toda la información de audio pase de forma continua. Teniendo este stream, debemos convertirlo a MFCCs para poder hacer las predicciones. Es por ello que utilizaremos Meyda para extraer 13 componentes del sonido a 16000 Hz y con un tamaño de bloques de 512.

Además tambien utilizaremos una función de ayuda que dispone Meyda para obtener

los RMS de cada audio escuchado (el volumen del audio). De esta forma las inferencias se realizarán cada vez que el segmento de audio escuchado supere un cierto umbral de volumen.

8.2. Integración del sistema de frecuencias

El procedimiento es muy similar al anterior. Se utiliza Meyda y los RMS para la detección de cuándo se ha cruzado el umbral para hacer predicciones. En este caso, no hace falta alojar ningún modelo en indexedDB debido a que es un algoritmo.

La diferencia principal, es que en lugar de computar los RMS se hace una llamada a pitchy cada "N"milisegundos y se obtienen muestras de la frecuencia escuchada. Para añadir una capa de fiabilidad, se establece el umbral de confianza que tiene pitchy para evitar capturar ruidos indeseados que no son claros.

Una vez obtenidas las frecuencias tendremos un array con una serie de las mismas. Para proceder a la predicción se obtiene la moda de la secuencia y de esta forma podemos garantizar que la frecuencia escuchada es la correcta.

Posteriormente se hace un parser para establecer a que nota y octava pertenece cada frecuencia y de esta manera poder imprimirla en pantalla.

CAPÍTULO 9

Desarrollo del Sistema en Tiempo Real con Metrónomo

Finalmente, la tercera de las funcionalidades es poder realizar las predicciones en tiempo real con metrónomo. Esto lo que permite es poder tararear al dispositivo con un tempo establecido y que el sistema haga una predicción tanto de la frecuencia como del ritmo. Sería como si una persona que se dedica a transcribir de manera profesional escribe por ti lo que se ha tarareado.

9.1. Desarrollo de sistema de reconocimiento de frecuencias en tiempo real con metrónomo

Suponiendo que el usuario elige un tempo de $4/4$ lo que significa que hay cuatro tiempos que se repiten ciclicamente a la misma velocidad: 1, 2, 3, 4, 1, 2, 3, 4, 1 ... La persona tararea y el sistema debe ser capaz de desglosar no solamente las frecuencias escuchadas, sino también el tempo de cada una de ellas. Por ejemplo, una negra tiene una duración de uno de los tiempos descritos previamente, una corchea la mitad, una semicorchea la mitad de la corchea y finalmente una fusa la mitad de la semicorchea. Esto descrito de otra manera sería, si una negra vale 1, la corchea 1/2, la semicorchea 1/4 y la fusa 1/8. De modo que en un tiempo entran 8 fusas o 4 semicorcheas o 2 corcheas o una negra. Para hacer predicciones en pantalla aparecerá el tempo.

Es decir, los números 1, 2, 3 y 4 secuencialmente y el usuario tarareará. Posteriormente, para la escritura de lo tarareado se recoge una muestra por tempo de fusa en un array. En otras palabras, cada uno de los tempos se recogen 8 muestras. A la hora de imprimir en pantalla lo escuchado la metodología es sencilla. Cada uno de los elementos del array, al ser una nota, se apilan las iguales para escribir la duración correspondiente, es decir, en caso de tener 8 fusas guardadas en el array que corresponden a 440Hz se puede saber que en verdad la duración real es de una negra y corresponde a la nota "la". Agrupando de esta manera todas las notas por grupos, se puede escribir lo tarareado sin necesidad de inteligencia artificial.

9.2. El problema del procedimiento actual

El principal problema de lo descrito anteriormente que se basa en tomar muestras e ir apilandolas para descubrir la longitud de cada nota reside en que las personas no somos perfectas. Una persona a la hora de tararear, se puede ir un poco de tempo y la traducción resulta en desastre. Además un pequeño desafine y la nota que querías cantar no es predicha de forma correcta.

Es por ello, que de cara a futuro se debría de montar un sistema de inteligencia artificial que trate de comprender qué ha querido decir cada interprete. Para ello se podría seguir utilizando el sistema actual, pero obteniendo tambien la corrección de cada usuario. De esta forma, tendriamos muestras de del array predicho y del real. Entrenando un modelo de inteligencia artificial, se podría predecir lo que en verdad se ha querido decir.

De esta forma un usuario tararearía, y al momento previo a la impresion pasar el resultado por un modelo de inteligencia artificial que pueda limpiar el array dejandolo como el usuario en verdad tenía en mente.

CAPÍTULO 10

UI/UX y Base de Datos con Firebase

En cuanto al desarrollo de la interfaz se refiere es importante tener un desarrollo básico y funcional para poder plantear un primer lanzamiento de la aplicación a modo de prueba. De esta forma se podrá evaluar la comodidad de la interfaz, la intuitividad y en caso de ser óptima, proceder a mejorar el diseño de manera que esté orientado al mercado.

En este capítulo, a la vez de explicar el diseño de la interfaz, tambien se describe el funcionamiento de la base de datos firebase de google conectada a la aplicación.

10.1. Login

La primera funcionalidad básica es poder registrarse e iniciar sesión, es por ello que de momento utilizaremos el inicio de sesión de google con OAuth 2.0, que es un sistema de autenticación seguro de google y que firebase proporciona. La interfaz se describe en la izquierda de la Figura 10.1.

10.2. Menú principal

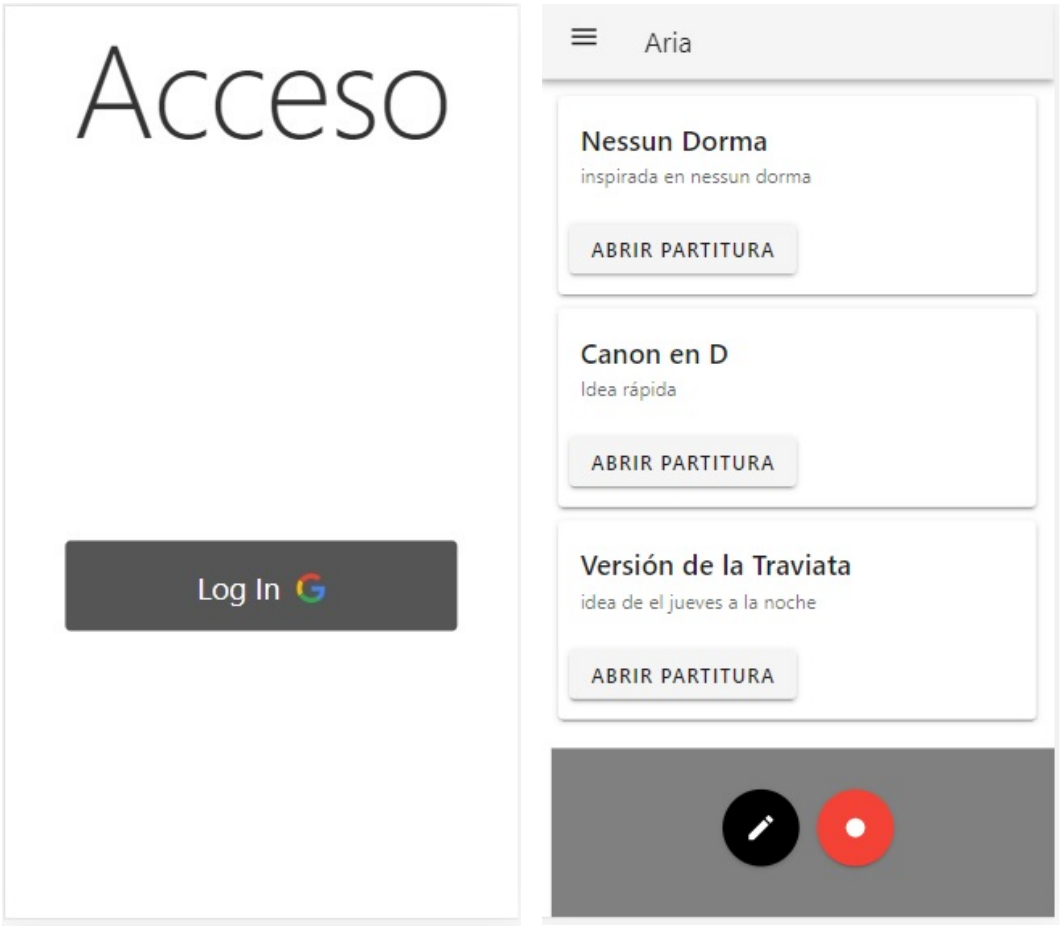
Para continuar, una vez se registra e inicia sesión un usuario, se abre la ventana de menú principal. La ventana consiste en las diferentes melodias grabadas por el usuario, las cuales están guardadas en firebase y se descargan de la nube al momento de abrir el menú. De modo que hay una llamada al servidor de la que se extraen los datos y se imprimen en pantalla.

Por otro lado, también en la parte superior izquierda hay un botón que abre el *drawer* o pantalla deslizante y en la parte inferior están las opciones de entrar al editor de forma normal o mediante grabación a tiempo real. El menú principal se puede ver en la derecha de la Figura 10.1.

10.3. Drawer

Como se ha descrito previamente, en la parte superior izquierda del menú, el botón activa el drawer que se puede ver en la Figura 10.2. Éste permite cerrar sesión al usuario

Figura 10.1 Login en la izquierda y menú principal a la derecha.



actual.

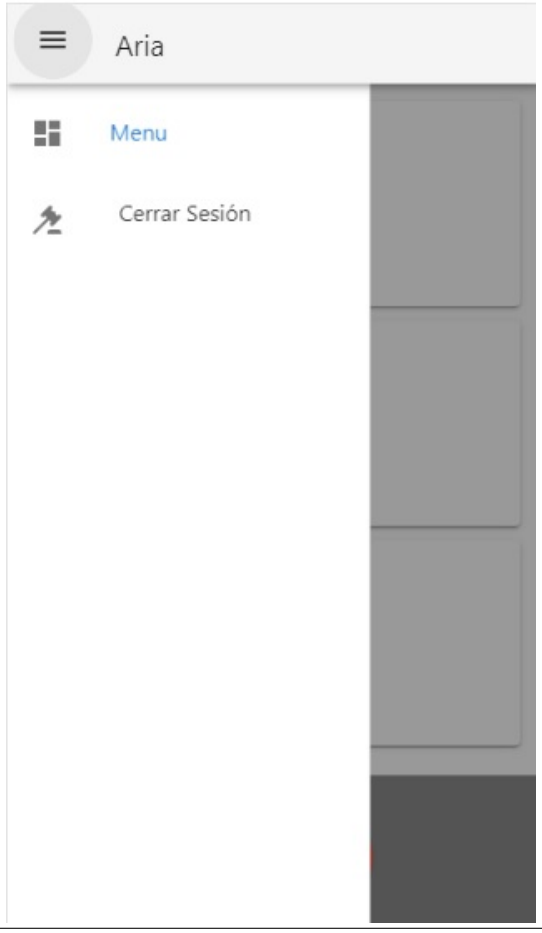
10.4. Editor

El editor es la pantalla más importante de la aplicación, donde el usuario pasará la mayor parte del tiempo. Consiste en la parte izquierda de botones describiendo las diferentes longitudes de nota: *redonda, blanca, negra, corchea* y *semicorchea*. En el momento de tocar uno de los botones la aplicación empieza a escuchar al usuario y empieza el reconocimiento de notas (*do, re, mi, fa, sol, la* y *si*). El usuario puede cambiar la longitud de notas a su parecer y se imprimirá la nota dicha con la longitud seleccionada.

Este proceso acelera mucho la velocidad de escritura respecto al resto de aplicaciones móviles.

Por otro lado, el editor dispone de otros tres botones en la parte inferior derecha de la pantalla, como se pueden ver en la Figura 10.3, donde el botón gris activa el drawer de la Figura 10.4 que permite habilitar el uso del reconocimiento de frecuencia en lugar del reconocimiento de comandos. El botón verde habilita el reconocimiento en tiempo real y dependiendo del *tempo* seleccionado por el usuario aparecerán una serie de números en pantalla cíclicamente describiendo a la velocidad a la que se debe tararear.

Figura 10.2 Drawer del menú principal.



Por último está el botón rojo con la equis que permite terminar la escucha del dispositivo.

10.5. Guardar canción

Finalmente, hay un botón encima del editor que permite guardar canciones en el servidor. Esta funcionalidad permite dar un nombre y descripción a la partitura y guardarla en firebase como se puede ver en la Figura 10.5.

Figura 10.3 editor.

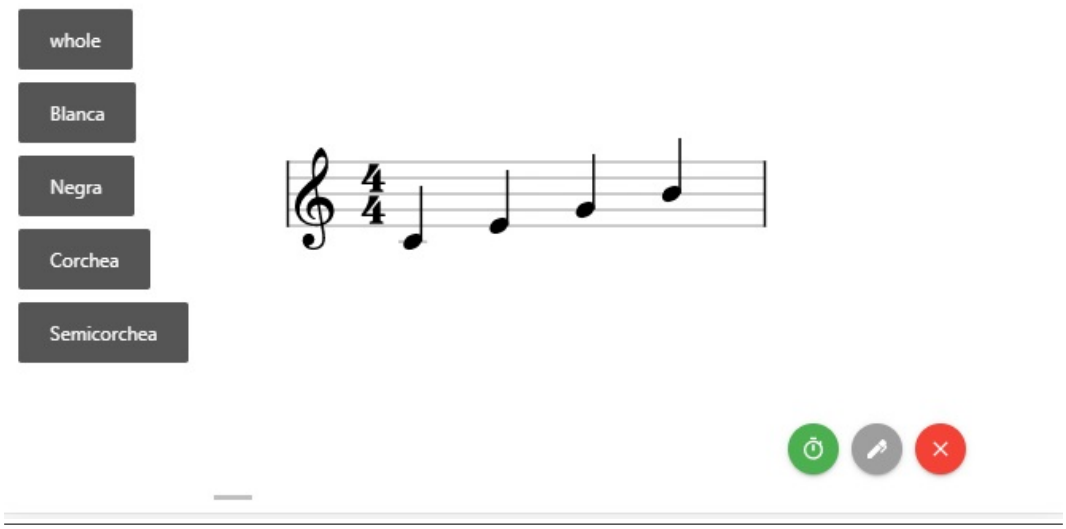


Figura 10.4 drawer del editor con opción a activar modo frecuencia.

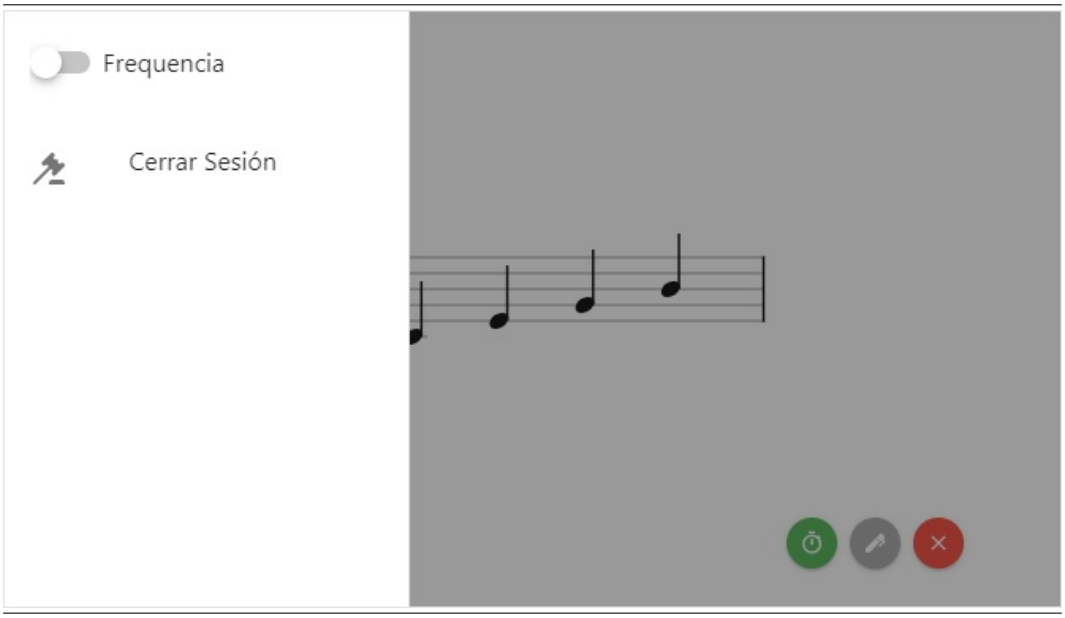
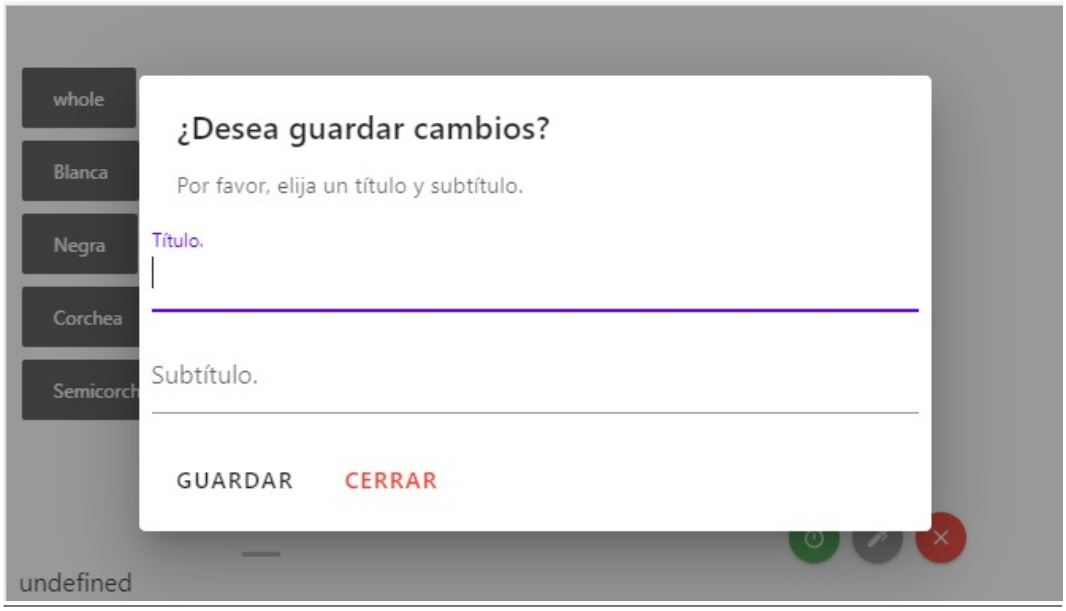


Figura 10.5 guardar partitura.



CAPÍTULO 11

Trabajo Futuro

A continuación, recapitularemos lo trabajado hasta ahora. Por un lado se ha realizado la integración de todas las piezas de software satisfactoriamente, y por otro, un sistema de reconocimiento de comandos de voz con un $92,5\,\%$ de precisión. Se ha implementado pitchy gracias a la contribución de McLeod et al. [16] y un sistema preliminar de reconocimiento de tarareo en tiempo real.

Es por ello que las tareas a mejorar a parte de todo lo respectivo al emprendimiento son la mejora de la precisión del reconocimiento de comandos de voz y la precisión del sistema de frecuencias en tiempo real.

11.1. Mejora del reconocimiento de comandos de voz

En lo que respecta al reconocimiento de comandos de voz hay varias vías de investigación posibles dependiendo de las tareas a realizar.

La primera opción consiste en continuar con el trabajo actualmente implementado y tratar de mejorar el accuracy. Para ello, la primera de las opciones es tratar de limpiar mejor los audios para evitar el exceso de ruido. Por otro lado, utilizar más técnicas de aumentación de datos también puede resultar efectivo, asi como la propia obtención de nuevas muestras de audio. Por otro lado, en cuanto a la arquitectura, la literatura previamente descrita muestra la importancia de entender los datos en un audio de una manera secuencial. Es por ello que algoritmos efectivos en el reconocimiento de series temporales pueden ser muy buena opción, como la combinación previamente descrita de biLSTM.

Por otro lado, otra de las opciones muy prometedoras es la descrita en el artículo de Chowdhury et al. [14], ya que con lo actualmente implementado solamente añadiendo LPC features y 1d triplet loss se obtienen muy buenos resultados con CNN en señales de audio degradadas. Esto es algo que no solamente el dataset actual tiene, sino que cada usuario al utilizar la aplicación es potencialmente un usuario con una señal de mala calidad, por lo que la implementación puede ser muy provechosa.

11.2. Mejora del sistema de reconocimiento de frecuencia en tiempo real con metrónomo

Por otro lado, como se ha descrito previamente, los humanos no somos perfectos y es por ello que los tarareos tampoco. Ni en lo correspondiente al eje vertical ni en el horizontal de la partitura se puede esperar una precisión muy alta es decir ni en el ritmo ni en la frecuencia ya que, a veces, hay desafinaciones y otras veces fallos a la hora de ir a tiempo.

Es por ello que la investigación y desarrollo de un sistema que sea capaz de interpretar lo que se ha intentado tararear resulta crucial. Esto podría realizarse con algo similar a lo que en procesamiento del lenguaje natural corresponde a un modelo del lenguaje. Es decir, una vez una persona habla y un sistema de ASR (automatic speech recognition) recibe la señal, normalmente el resultado de la traducción no es perfecto. Son luego los modelos de lenguaje los que transforman lo recibido en algo con mayor sentido utilizando diferentes técnicas.

Un buen camino puede ser la utilización de modelos estadísticos para así tratar de corregir todo lo que escape a un cierto umbral. Como trabajo previo, un buen punto de partida es lo que a dia de hoy se utiliza para la creación de música automática [17] [18]. Estos modelos estadísticos se entrenan en base a partituras previas y aprenden los patrones necesarios para mantener la lógica y coherencia en la creación de automática de música.

Una buena opción es utilizar dichos modelos para a la hora de recibir un stream de audio y decodificarlo, tratar de corregir las imperfecciones utilizando estos modelos.

11.3. El camino de emprendimiento restante

Aparte de todo lo descrito con anterioridad, sin duda el reto más grande de todos es el de emprender. No solamente va a hacer falta que lo descrito en este trabajo funcione a la perfección sino que el mercado lo acoja para poder ir construyendo sobre lo que será este primer peldaño. Una vez el producto esté disponible toda la gestión de marketing y de empresa será el segundo peldaño y en caso de que se confirme la viabilidad o a lo que en el mundo del emprendimiento se le llama product-market fit se podrá proceder a la validación del modelo de negocio o business model fit.

En caso de lo contrario, se deberá pivotar para poder encontrar lo que al final acabe dando resultado. Ese pivotaje puede ser de modelo de negocio, de producto, de marketing... pero en cualquier caso seguir buscando cómo poder aprovechar lo realizado hasta la fecha para aportar el máximo valor posible. En el peor de los casos habrá que pivotar en cuanto a producto lo que no significa que todo el trabajo deba ser deshechado.

Esto último es muy importante, ya que, antes de acabar me gustaría mencionar que todo este trabajo de fin de máster está pensado para la máxima reutilización posible. En caso de que el producto no encuentre su lugar, con los mismos sistemas de inteligencia artificial se puede desarrollar software para que guitarristas puedan transcribir a tablatura de forma rápida en cualquier lugar o incluso renovar la educación musical con la opción de un autoasesoramiento al momento por parte de una inteligencia artificial. En cualquier caso, este trabajo es el primer peldaño de algo que espero llegue a mucha gente y que dure mucho tiempo.

Bibliografía

[1] Eric Ries and Bartosz Sałbut. El método lean startup. 2012. Ver página 5.

[2] Bruce P Bogert. The quefrency alanysis of time series for echoes; cepstrum, pseudo-autocovariance, cross-cepstrum and saphe cracking. *Time series analysis*, pages 209–243, 1963. Ver página 8.

[3] Douglas Coimbra de Andrade, Sabato Leo, Martin Loesener Da Silva Viana, and Christoph Bernkopf. A neural attention model for speech command recognition. *arXiv preprint arXiv:1808.08929*, 2018. Ver páginas 21, 24.

[4] Brian McMahan and Delip Rao. Listening to the world improves speech command recognition. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32, 2018. Ver página 21.

[5] Roman Vygon and Nikolay Mikhaylovskiy. Learning efficient representations for keyword spotting with triplet loss. In *International Conference on Speech and Computer*, pages 773–785. Springer, 2021. Ver páginas 21, 22.

[6] Pialy Barua, Kanij Ahmad, Ainul Anam Shahjamal Khan, and Muhammad Sanaullah. Neural network based recognition of speech using mfcc features. In *2014 international conference on informatics, electronics & vision (ICIEV)*, pages 1–6. IEEE, 2014. Ver páginas 21, 24.

[7] Muhammad Imtiaz and Gulistan Raja. Isolated word automatic speech recognition (asr) system using mfcc, dtw knn. pages 106–110, 11 2016. Ver páginas 21, 24.

[8] Mahdi Shaneh and Azizollah Taheri. Voice command recognition system based on mfcc and vq algorithms. *World Academy of Science, Engineering and Technology*, 57:534–538, 2009. Ver páginas 21, 22.

[9] Muhammad Atif Imtiaz and Gulistan Raja. Isolated word automatic speech recognition (asr) system using mfcc, dtw & knn. In *2016 asia pacific conference on multimedia and broadcasting (APMediaCast)*, pages 106–110. IEEE, 2016. Ver páginas 21, 22.

[10] Hugh Rawlinson, Nevo Segal, and Jakub Fiala. Meyda: an audio feature extraction library for the web audio api. In *The 1st web audio conference (WAC). Paris, Fr*, 2015. Ver páginas 21, 22.

[11] Lina Tarek Benamer and Osama AS Alkishriwo. Database for arabic speech commands recognition. 2020. Ver página 22.

[12] Prajoy Podder, Tanvir Zaman Khan, Mamdudul Haque Khan, and M Muktadir Rahman. Comparative performance analysis of hamming, hanning and blackman window. *International Journal of Computer Applications*, 96(18), 2014. Ver página 22.

[13] Shakil Ahmed Sumon, Joydip Chowdhury, Sujit Debnath, Nabeel Mohammed, and Sifat Momen. Bangla short speech commands recognition using convolutional neural networks. In *2018 International Conference on Bangla Speech and Language Processing (ICBSLP)*, pages 1–6, 2018. Ver página 24.

[14] Anurag Chowdhury and Arun Ross. Fusing mfcc and lpc features using 1d triplet cnn for speaker recognition in severely degraded audio signals. *IEEE Transactions on Information Forensics and Security*, 15:1616–1629, 2020. Ver páginas 24, 39.

[15] Ovishake Sen, Pias Roy, et al. A convolutional neural network based approach to recognize bangla spoken digits from speech signal. In *2021 International Conference on Electronics, Communications and Information Technology (ICECIT)*, pages 1–4. IEEE, 2021. Ver página 24.

[16] Philip McLeod and Geoff Wyvill. A smarter way to find pitch. In *ICMC*, volume 5, pages 138–141, 2005. Ver páginas 27, 39.

[17] Dorien Herremans, Kenneth Sörensen, and Darrell Conklin. Sampling the extrema from statistical models of music with variable neighbourhood search. 2014. Ver página 40.

[18] Izaro Goienetxea, Iñigo Mendialdua, Igor Rodríguez, and Basilio Sierra. Statistics-based music generation approach considering both rhythm and melody coherence. *IEEE Access*, 7:183365–183382, 2019. Ver página 40.