Universidad Zaragoza

Trabajo Fin de Grado

Análisis armónico de archivos MIDI

Harmonic analysis of MIDI files

Autor

Jorge Bajén Gonzalo

Directores

Carlos Hernández Oliván

José Ramón Beltrán Blázquez

ESCUELA DE INGENIERÍA Y ARQUITECTURA

2022

AGRADECIMIENTOS

Quería aprovechar este espacio para agradecer tanto a José Ramón Beltrán Blázquez como a Carlos Hernández Oliván por su dedicación y apoyo mostrados en la realización de este trabajo, así como a la labor que están realizando en el departamento, dándole voz a la introducción de la música en las investigaciones de la universidad.

A mi familia, sobre todo a mis padres y hermanos por el apoyo incondicional y su habilidad para no soltarme nunca de la mano y mostrarme una confianza inspiradora.

Por último, me gustaría dar las gracias a mi fiel compañera, la música, ya que sin ella este trabajo nunca podría haber sido posible y me ha acompañado durante todas las etapas de mi vida y en especial a mi grupo HUE CO, con el que he aprendido a ser constante y perseguir aquello que te motiva a mirar hacia adelante.

RESUMEN

En este trabajo se tiene como objetivo la extracción de acordes desde archivos MIDI por medio de la utilización de inteligencia artificial. Para ello se ha utilizado como referencia una red neuronal modelada por Tsung-Ping y Su. Dicho modelo utiliza archivos en formato *.csv*, por lo que para el correcto funcionamiento de nuestro modelo, ha sido necesaria la creación de un algoritmo que transformara los datos incluido en un archivo MIDI en los datos de entrada del archivo empleado por los creadores de la red (archivos *.csv*). Una vez creado dicho algoritmo, se ha procedido al entrenamiento de la red, utilizando la misma base de datos que los creadores de la red. Tras la finalización del entrenamiento de la red se ha implementado un algoritmo para la predicción de acordes a través de los archivos MIDI. Los resultados obtenidos con dicha predicción han sido muy positivos, ya que los resultados de validación de nuestro modelo superan a los obtenidos por los creadores de la red en el modelo original.

Índice

Capítulo 1

Introducción y objetivos

1.1. Motivación y conceptos

En el mundo de la música son muchas las personas que piensan que está todo creado o escuchado. Lejos de esa afirmación se encuentra la base a abordar en este trabajo. Partiendo desde una perspectiva no manual podemos encontrar todo un mundo de investigación para la evolución de la música tal y como la conocemos hoy en día. Cierto es que la exploración de la música realizada de forma analógica es un campo que rara vez se trata de revisar, o implementar, ya sea por desconocimiento o por simple desapego hacia la idea de digitalizar algo tan humano. Precisamente este pensamiento es el que me impulsó para hacer este trabajo, en el que por medio de las inteligencias artificiales, se buscará extraer acordes de un archivo MIDI de manera automatizada, lo que facilitará a todo el mundo (sin importar su nivel de conocimientos musicales) una entrada hacia el campo de la composición musical.

La introducción de la Inteligencia Artificial (IA) [2] en el estudio y creación de partituras representó un avance que no se incorporó a la industria musical hasta comienzos del siglo XXI, con el objetivo de facilitar el acceso a la creación musical a aquellos usuarios sin conocimientos musicales.

Aunque dichas IAs (en su introducción en el ámbito musical) fueron utilizadas de manera primitiva para el trabajo de composición y/o entendimiento de los procesos musicales, [3] en la actualidad han sido utilizadas para profundizar en otros campos pertenecientes al desarrollo natural de nuestra sociedad, de manera que hoy en día forman parte de nuestra rutina.

Desde comienzos del siglo XXI existe un ámbito científico y tecnológico encargado del estudio de tecnologías musicales, conocido como Music Information Retrieval (MIR) [4]. La primera conferencia que se realizó bajo el ámbito del MIR fue en el año 2000, aunque el estudio musical a través de IAs ya fué utilizado previamente [5], [6].

La primera empresa en indagar en la utilización de IAs en el ámbito musical para su

comercialización de cara al público fué *Shazam*[1] (en el año 1999), la cuál determinaba qué canción estaba siendo reproducida por el usuario mediante la identificación sonora. El papel de las IAs en la comercialización fue creciendo con los años de manera notoria, tomando un papel realmente importante gracias a sus algoritmos, los cuales permitían a las discografías optimizar sus procesos de búsqueda en el descubrimiento de nuevos talentos, así como el avance de la personalización del catálogo musical ofrecido al consumidor por empresas como Spotify[2], una empresa sueca que crea listas personalizadas con las canciones relacionadas con los gustos personales de cada consumidor mediante IAs.

Respecto a la aportación de las IAs en el análisis armónico, podemos observar un gran número de avances, siendo los siguientes los más significativos: en el año 2005 se presentó el primer modelo que permitía extraer los primeros acordes de archivos musicales [7], pero el siguiente cambio realmente significativo no vino hasta 2016, cuando un grupo de investigadores de la empresa Google formó "Magenta"[3], para desarrollar, entre otras aplicaciones, métodos de composición musical y extracción de acordes mediante implementación de IAs.

1.2. Objetivos y alcance del proyecto

La música como la conocemos se concibe como una extensión del ser humano a través de la cuál se trata de expresar emociones. Por eso, existen distintas objeciones hacia la creación de la misma, mediante la implantación de métodos automáticos.

En este trabajo trataremos de facilitar la creación de dichos métodos mediante la extracción de acordes y escalas desde un archivo MIDI.

Los pasos seguidos para dicha extracción han sido:

- Recopilación de información para el estado del arte.
- Ajuste del algoritmo desarrollado por Tsung-Ping Chen and Li Su [1] [8].
- Entrenamiento y validación del modelo.
- Implementación código transcripción MIDI-CSV para la predicción armónica de archivos MIDI.
- Estudio del entrenamiento y evualuación de los resultados.

[1] `https://www.shazam.com/es/home`

[2] `https://www.google.com/search?q=spotify&oq=spotify&aqs=chrome..69i57j46i39i199i465j69i59j0i67l2j0i512j69i60l2.1434j0j7&sourceid=chrome&ie=UTF-8`

[3] `https://magenta.tensorflow.org/`

1.3. Estructura de la memoria

La memoria del trabajo se organiza en 6 capítulos. Éstos capítulos representan distintas secciones a tener en cuenta en el entendimiento del trabajo y deben de seguirse en orden para la total comprensión del mismo.

En este primer capítulo se hablará del contexto que rodea al trabajo, así como de los objetivos, motivación y alcance que se desean obtener.

En el segundo capítulo se introducirá brevemente el funcionamiento y funcionalidad de la detección armónica de acordes, así como sus posibles formas de detección.

En el tercer capítulo se describe el estado del arte empleado en este campo, para el cuál ha sido necesario desengranar los modelos empleados con anterioridad y sus funciones actuales.

En el cuarto capítulo se explica los modelos que han sido empleados para realizar los diferentes programas que han sido implementados para la detección de acordes musicales.

En el quinto capítulo se mostrarán los resultados experimentales obtenidos y se compararán estos con otros sistemas desarrollados del mismo campo.

Por último, en el capítulo 6 se mostrarán las conclusiones del trabajo, así como posibles vías de estudio que serían interesantes de cara a proseguir con la investigación e innovación en la creación musical mediante IAs.

Capítulo 2

Introducción a los archivos MIDI, detección armónica y familiarización con el Deep Learning

En este trabajo nos centraremos principalmente en el análisis de la armonía de un archivo MIDI. Para ello introduciremos los conceptos tanto de tonalidad, acorde y escala, con el objetivo de abordar el análisis armónico, así como el concepto de MIDI y sus diversas funcionalidades y características propias.

2.1. MIDI y archivo MIDI

Se entiende por MIDI (Musical Instrument Digital Interface) a un protocolo de interacción entre diferentes instrumentos y dispositivos (controladores, sintetizadores, instrumentos electrónicos, etc.) con objeto de simplificar y facilitar su comunicación en la elaboración musical. Fué inventado en 1981 por el ingeniero electrónico Dave Smith, y desde su aparición, supuso un cambio radical en la manera de componer música en todo el mundo.

Un archivo MIDI es un archivo que utiliza dicho protocolo, almacenando los sonidos musicales pertenecientes al archivo, e información relevante acerca de los mismos. Este tipo de archivos son de un peso considerablemente menor al de otros archivos de audio comúnmente utilizados (como pueden ser el wav o el mp3[1], archivos utilizados desde la década de los 90's).

[1] `https://curiosfera-historia.com/quien-invento-el-mp3-historia/`, último acceso: agosto 2022

2.1.1. Utilización, conectividad y mensajes MIDI

Un dispositivo con conectividad MIDI convierte una serie de instrucciones en sonido, gracias a que la información proporcionada nos informa de las notas contenidas para el instrumento, su duración, intensidad, altura, etc. Además de la información acerca de las notas, los archivos MIDI también nos dicen el tipo de instrumento para el que está elaborado dicho archivo, sin llegar a decirnos exactamente el timbre del instrumento, es decir, nos indican si el instrumento para el que han sido realizados es, por ejemplo, un piano, pero sin indicar el tipo de piano exacto para el que han sido elaborados (dicha información se almacena en los mensajes Program Change). Lo que no nos dice dicha información que se encuentra en el protocolo interno de los aparatos MIDI es el lugar de la escala armónica en que se debe de tocar, y en este problema se basa mi estudio.

La conectividad asociada a estos aparatos MIDI es muy variada. Puede haber casos muy sencillos en los que la interfaz a interpretar vaya directamente relacionada con el ordenador que contiene el sistema electrónico dedicado a la grabación y/o producción musical (DAW),y ocasiones en las que será necesario conectar otros dispositivos intermedios para transformar las señales de salida/entrada pertinentes.2.1[2]

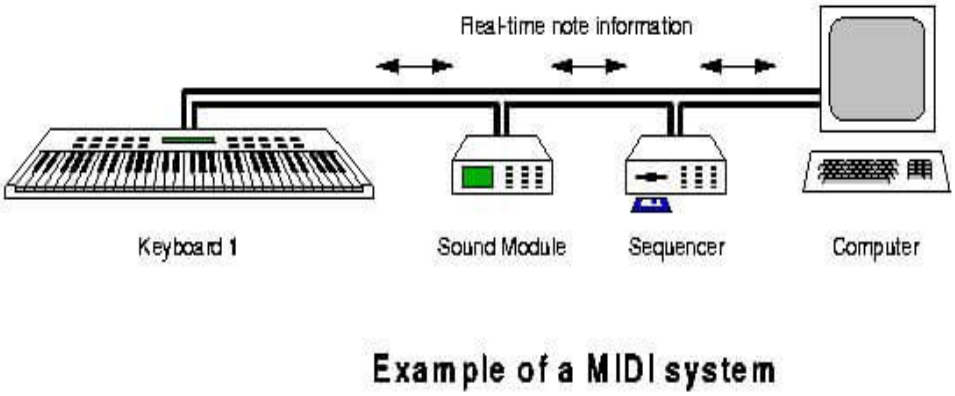

Figura 2.1: Esquema de un sistema MIDI para enviar señal desde un teclado a un ordenador

En cuanto al mensaje de los archivos MIDI [9] es muy importante tener en cuenta 2 tipos, los mensajes de canal (aquellos que únicamente se referencian a un canal) y los mensajes de sistema (cuando se referencian al sistema general).

El tipo de mensaje en el que más hincapié debemos hacer para comprender el funcionamiento de los archivos MIDI son los mensajes de canal, en concreto los mensajes MIDI Note-on y Note-off [10].

El Note-on sirve para para indicar el comienzo de una nota (más precisamente para

[2] `https://www.cavsi.com/preguntasrespuestas/que-es-midi/` último acceso: agosto 2022

indicar el comienzo de la pulsación en el aparato MIDI empleado), y el Note-off para indicar la finalización de la misma nota (o finalización de la pulsación).

Los datos incluidos en cada uno de los mensajes son:

- El número de nota (o *pitch*).
- El canal por el que está siendo reproducida la nota.
- La *velocity*, que actúa a modo de indicador de la potencia con la que ha sido pulsada la activación de la nota en cuestión.

Cada uno de los mensajes contiene 3 bytes, uno para cada uno de los datos. Con el primero de los bytes detectamos el canal MIDI, con el segundo identificamos el *pitch* de la nota a reproducir y con el tercero el *velocity* [11].

2.1.2. Datos contenidos en un fichero MIDI

Cada vez que una tecla de un instrumento MIDI es pulsada se crea una nota MIDI, también denominada evento MIDI. En cada uno de los eventos MIDI, encontramos almacenados 8 patrones diferenciadores, lo que nos permitirá obtener información suficiente como para (con el instrumento adecuado seleccionado) reproducir su información en forma de melodía.

Cada patrón caracteriza de manera diferente la lectura y el funcionamiento del MIDI:

- Un detector de pulsación, para detectar cuando el sonido debe comenzar/terminar de reproducirse.
- Detección de teclas pulsadas, para contabilizar y detectar múltiples teclas tocadas simultáneamente.
- Pace, o velocidad a la que se pulsan las teclas, es decir, fuerza con la que éstas son tocadas. Con ello se puede obtener la intensidad con la que se desea tocar el fragmento.
- Fuerza con la que una tecla se mantiene pulsada.
- Tempo en el que ha sido tocado el archivo, indicado en BPM, facilitándose así su posterior análisis y ajuste.
- *Panning*, o paneo, que nos indica el lugar por dónde el sonido debe de emerger, ya sea de forma frontal, como lateral(tanto por izquierda como por derecha).

– Modulación, nos permite ajustar el sonido de manera tonal, o modificar dicho tono para conseguir cambiar el timbre al sonido originalmente reproducido.

– Volumen de la pieza o archivo MIDI.

En la figura 2.2[3] podemos observar un archivo MIDI en su representación mediante un DAW.

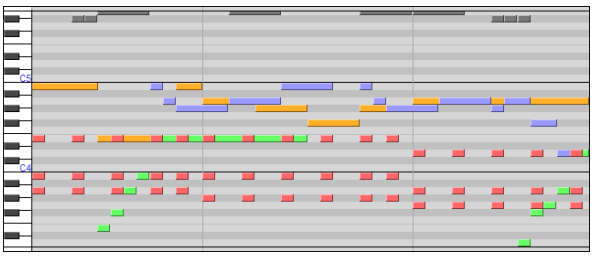

Figura 2.2: Archivo MIDI representado en un DAW

2.2. Detección armónica

La detección armónica o Automatic Chord Recognition (ACR) es un campo de estudio dentro del MIR que se encarga de estudiar y extraer la armonía de señales musicales, ya sean señales de audio o música simbólica (como los archivos MIDI).

2.2.1. Detección de Tonalidad

Para la detección de armonía existen varios factores a tener en cuenta. Lo primero en lo que debemos fijarnos es en la armadura de la partitura, con ella sabremos directamente que tonalidad es la que debemos tocar (en nuestro caso particular este paso no nos es útil debido a que no disponemos de la partitura), después miraríamos el acorde con el que concluye la pieza, ya que muchas piezas finalizan con su primer grado (el cuál incluye la armonía de la pieza), después debemos fijarnos en las progresiones armónicas presentes a lo largo de la pieza, ya que estas se suelen realizar como modificaciones del acorde de primer grado de la armonía principal, y ya por último nos quedaría fijarnos en los acordes de 7ª dominantes, ya que estos son (casi siempre) parte de la armonía. Existen algoritmos tradicionales que tratan de detectar la tonalidad de música simbólica. Estos algoritmos utilizan *key-profiles* para obtener la tonalidad [12] [6].

[3] `https://patrickdearteaga.com/es/formato-midi-en-videojuegos/` último acceso: agosto 2022

Para las IAs hay varias formas de detectar la tonalidad de una pieza, pero la más fiable e interesante en cuanto a metodología es el estudiado y elaborado por Daniele P. Radicioni and Roberto Esposito en 2007 [13].

2.2.2. Detección de acorde

Para poder detectar un acorde son indispensables conocimientos musicales, dado que sin ellos no nos será posible discernir entre las notas que estamos oyendo.

Los principales sonidos que debemos saber diferenciar son los sonidos de la mayor y tercera menor, (desarrollados en el anexo A) de esta forma podremos detectar acordes mayores y menor con soltura, únicamente practicando el método.

Aunque a día de hoy los músicos siguen descubriendo qué acordes están escuchando de manera automática y mental, ya existen aplicaciones que nos permiten sacar el acorde automático desde una pista de audio (hay programas como *Studio One* que permite la extracción directa de acordes con un comando propio de la plataforma), pero en nuestro caso queremos saber el acorde que se está tocando desde la información dada por un archivo MIDI.

2.2.3. Detección del ritmo

El ritmo o compás de una obra musical se detecta al igual que la tonalidad, es decir, lo podemos observar al inicio de la partitura indicado en forma numérica.

En nuestro caso, y como ya he indicado con anterioridad, no disponemos de dicha partitura para extraer el ritmo, pero al contrario que con la tonalidad, un archivo MIDI si posee información sobre la duración de las distintas notas o marcas relacionadas con las notas. Gracias a esto podemos deducir con mayor facilidad el compás de la pieza.

Para saber si el sistema de reconocimiento será o no eficiente, habrá que tener en cuenta la capacidad de detección exacta de acorde.

2.3. Introducción al Deep Learning

El Deep Learning, o aprendizaje profundo, se define como un conjunto de técnicas que pretenden simular el comportamiento del aprendizaje humano con el fin de aprender sobre ciertos campos del conocimiento.

A su vez, el DL es un tipo de Machine Learnining (ML), o aprendizaje automático, cuya característica que lo diferencia son las múltiples capas de conocimiento que utiliza para conseguir entrenar a la máquina. Estas capas se entrelazan formando una red neuronal, de la que la máquina a entrenar va sacando información y creando nuevas capas de manera automática. De ahí que el Deep Learning esté adquiriendo gran

importancia dentro del ML, ya que es capaz de obtener datos de salida específicos desde una entrada más general. [14] [15]

2.3.1. Redes Long-Short Time Memory (LSTM)

Dentro del ML existen multitud de redes neuronales, las cuales están más desarrolladas en el apéndice B, pero para nuestro trabajo, haremos incapié en un tipo de red neuronal recurrente (o RNN), denominada Long-Short Time Memory.

Una red LSTM [16] (la cuál podemos observar en la figura 2.3[4]) es una especificación de una RNN (Recurrent Neutral Network), redes que se sirven de la utilización de loops en cada uno de sus pasos para pasar la información de un paso a otro, capaz de aprender dependencias más grandes o largas.

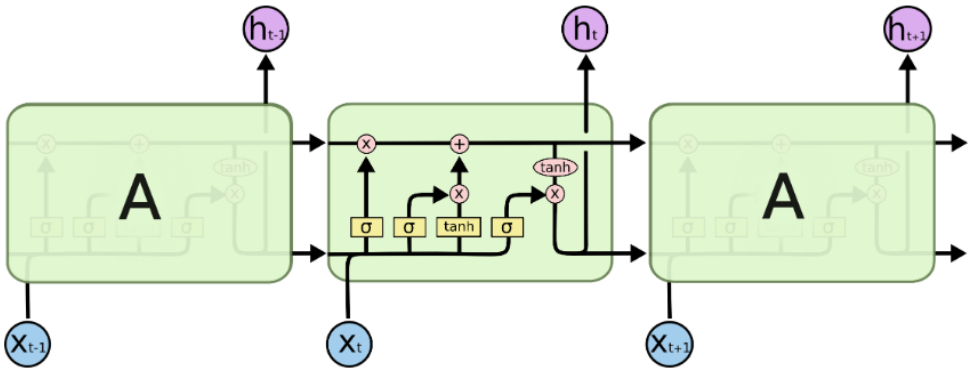

Figura 2.3: Procesos en una red LSTM

Estas redes neuronales se identifican por almacenar la información de sucesos previos para poder aprender sobre ellos. Esto se debe a que su estructura tiene cuatro niveles, en lugar de uno, y tres puertas reguladoras, las cuales filtran la información previa que deseamos dejar pasar, o el nivel de desarrollo que queremos que alcance en cada iteración de nuestra red, mediante valores entre 0 y 1.[16]

2.3.2. Multi-task Deep Learning

El aprendizaje multitarea se basa en la resolución de varias tareas de aprendizaje de manera simultánea.[17]

Para aprender de una manera simultánea, se utiliza el aprendizaje paralelo, de manera que lo que aprende de una tarea sirve al mismo tiempo para entrenar y aprender

[4] `http://colah.github.io/posts/2015-08-Understanding-LSTMs/` último acceso: agosto 2022

en las demás tareas, de una manera recíproca. Podemos observar el funcionamiento del modelo en la figura 2.4[5]

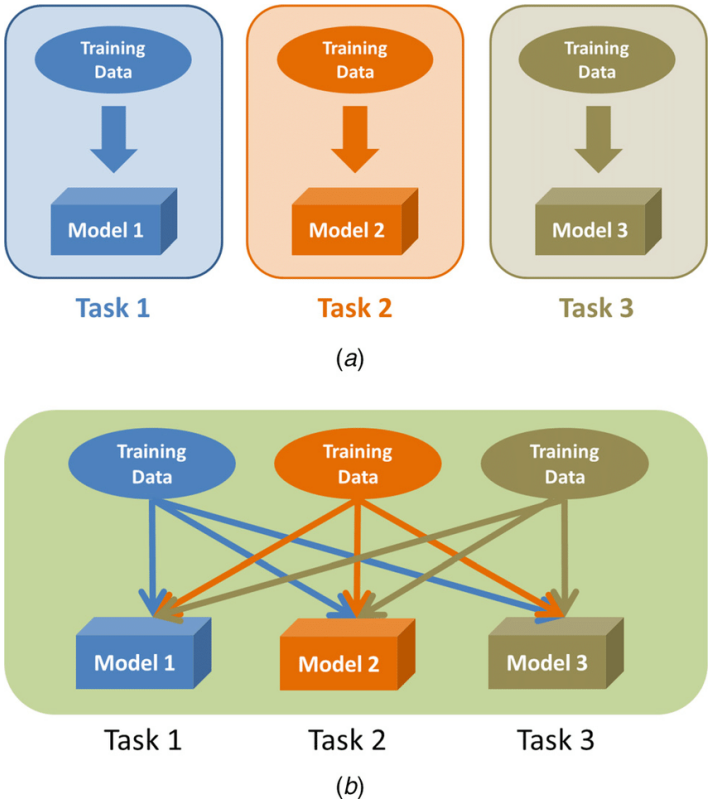

Figura 2.4: Funcionamiento de una red multi-task. [18]

[5]https://www.researchgate.net/figure/Difference-between-single-task-learning-and-multitask-learning-a-single-task-learning_fig2_307622018 último acceso: agosto 2022

Capítulo 3

Estado del arte

Para poder realizar este trabajo, es indispensable tener en cuenta los avances realizados en el análisis de archivos MIDI, cuyo objetivo es relacionarlos con instrumentos musicales. Para ello me he basado en varios trabajos realizados anteriormente. Los más importante son los trabajos realizados por Kristen Masada y Razvan Bunescu [19], y el estudio de Matthew McVicar [20], ambos realizados con objetivo de reconocer acordes a través de archivos MIDI ubicados en bases de datos.

3.1. Detección de Tonalidad

Para poder detectar sonidos con el objetivo de crear música de manera automática utilizando DL, es imprescindible poder detectar la tonalidad de el sonido a detectar, ya que sin el tono es imposible poder llegar a obtener los acordes del mismo.

Este concepto no ha pasado desapercibido en la historia de ACR, ya que el avance de las IAs ha tenido una gran importancia en la detección de la tonalidad de una partitura, permitiéndonos a día de hoy crear aplicaciones como *'Hum to search'*[1] [21], o *'Tararear para buscar'*, una herramienta desarrollada por Google[2] que nos permite identificar y reproducir una canción tarareando una parte de la misma, detectando la tonalidad mediante el entrenamiento de una red neuronal.

3.2. Reconocimiento y extracción de acordes

3.2.1. Definición de acorde

A lo largo de los años, han sido numerosas las definiciones que ha recibido el concepto de acorde en el ámbito musical. Para este trabajo el concepto se va a definir como la disposición vertical de 3 o más sonidos simultáneos, generando estos una

[1] `https://guidebooks.google.com/iphone/googleapp/hum-to-search-for-a-song` último acceso: agosto 2022.

[2] `https://about.google/intl/es_zz/` último acceso: agosto 2022

armonía. Estos sonidos pueden ser emitidos por uno o más instrumentos. Es por esto que la detección de acordes de forma manual es un trabajo complejo y que requiere de grandes conocimientos musicales previos. La alternativa a la identificación manual o personal no es otra que la implementación y entrenamiento de redes neuronales.

3.2.2. Transcripción del archivo MIDI

Para poder reconocer e identificar los acordes que se están interpretando en una melodía a través de su archivo MIDI, es indispensable reconocer primeramente qué notas y en qué orden y tempo están siendo utilizadas en el archivo. Al comienzo del estudio de los diferentes acordes para su reconocimiento, éstos se intentaban reconocer mediante la búsqueda de su primera nota, y las posibles secuencias de la misma. El problema de dicho análisis era la dificultad que se encontraba al intentar clasificar acordes complejos (como pueden ser acordes de novena, acordes disminuidos o acordes aumentados, entre otros).

3.2.3. Análisis de la armonía

Una vez tenemos claro qué notas y en que orden y tempo están siendo tocadas en nuestro archivo, podremos extraer la partitura musical que se está interpretando. De esta manera podremos proceder a detectar que armonías se están interpretando y al mismo tiempo detectar la escala del mismo. Para ello empezaremos separando nuestra partitura en pequeños fragmentos, en los cuales detectaremos el acorde predominante.

Estudio del ACR

En la primera experiencia con la extracción, se intentó detectar acordes analizando nota por nota ('a la fuerza'), empleando para ello algoritmos de Viterbi. Viendo que dicho algoritmo no era muy efectivo (como podremos observar en posteriores apartados de este mismo trabajo), y aprovechando la aparición en el mundo del Automatic Chord Recognition (ACR) del Deep Learning, por fin se pudo estudiar los acordes a través de dos métodos de mayor precisión: Bi-directional Transformer for Chord Recognition (BTC) and the Harmony Transformer (HT).

Para predecir los acordes se pueden utilizar la mezcla de modelos de identificación de notas, tales como segmentación de acordes (Degani, 2015 [22]), el *tempo* y la tonalidad (Zenz y Sauber, 2007 [23]), la línea seguida por un bajo (Yang y Magnusdottir, 2016 [3]), las notas fundamentales (Yang y Magnusdottir, 2016 [3]), la medida (Degani, 2015 [22]) y una progresión posterior del posible acorde ((Sheh, 2012 [24]);(Taemi, 2009 [25]);(Kwon y Deng, 2016 [26]);(Widmer y Korzeniowski, 2016 [27])). Lo que nos da

un estudio bastante válido del acorde en cuestión a analizar.

A la hora de abordar el estudio musical a través de acordes, tendremos en cuenta su representación simbólica, ya que a pesar de que en el estudio de acordes sí que podemos encontrar varias referencias (Bunescu Masada, 2022 [19]), a la hora de analizar su simbología nos encontramos que ésta no ha sido estudiada con detenimiento mediante redes neuronales (Chen Su, 2021 [1]).

3.3. Redes neuronales empleadas

A lo largo del estudio de los automatismos en el ACR, se emplearon distintas redes neuronales. Pese a que para este trabajo vamos a implementar una red LSTM, la cuál ya ha sido explicada previamente en la sección 2.3.1, existen varias redes neuronales utilizadas en el entorno de ACR, como el BTC o el HT

3.3.1. BTC Y HT

El BTC (Figura 3.1) utilizó un mecanismo de autoatención para capturar la dependencia a largo plazo en secuencias musicales y mostró su capacidad para segmentar secuencias de acordes; el HT (Figura 3.2) estimó las transiciones de acordes (o límites de acordes) y luego reconoció los acordes atendiendo a la secuencia informada por segmentación.los dos modelos han demostrado la eficacia del mecanismo de atención en el modelado de progresiones de acordes y han superado a otros modelos prometedores en investigaciones anteriores (Korzeniowski y Widmer, 2016 [27]). El BTC utiliza un codificador bidireccional mientras que el HT utiliza dos unidireccionales, uno en cada dirección (hacia delante y hacia atrás)(Chen y Su).

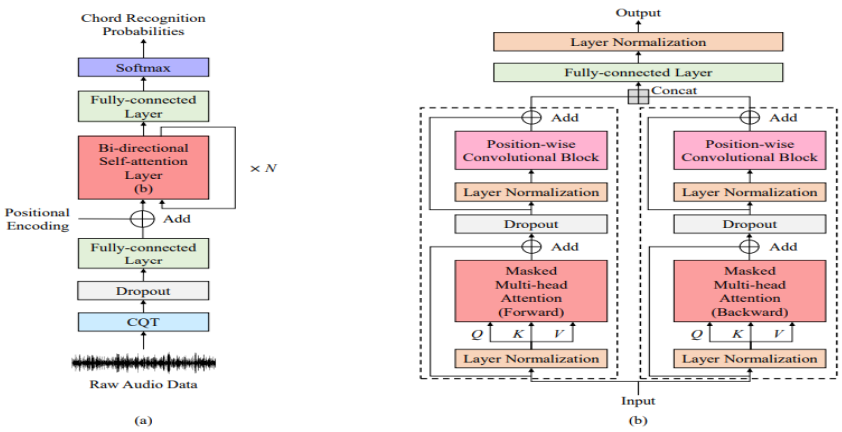

Figura 3.1: Representación de una red bidireccional [28]

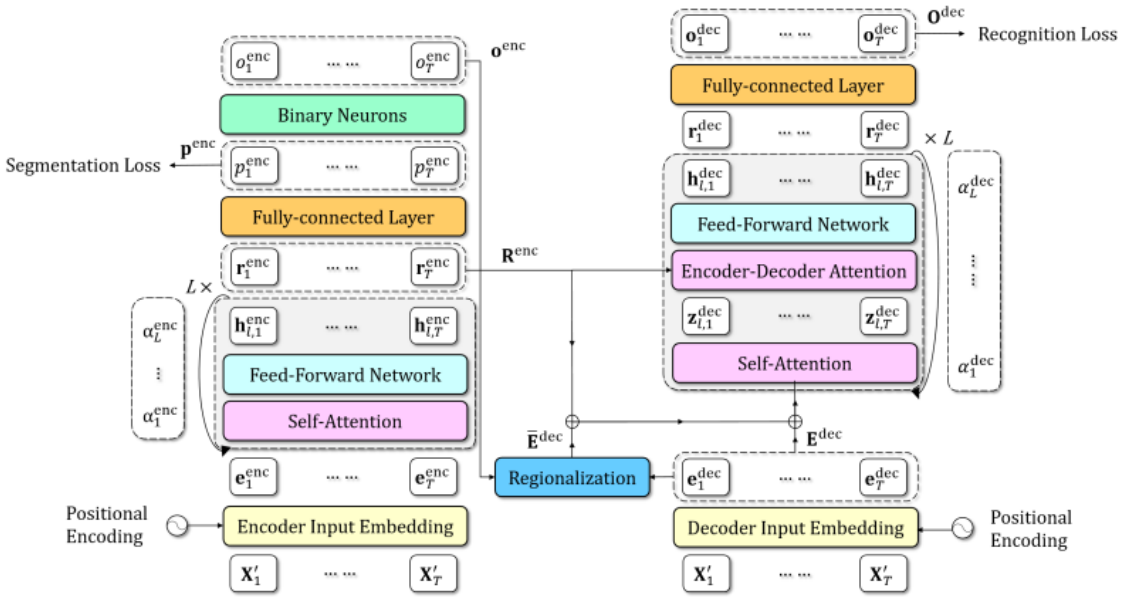

Figura 3.2: Representación de dos redes unidireccionales [8]

Capítulo 4

Descripción del Modelo

4.1. Modelos utilizados

Una vez explicados los modelos y apartados necesarios para la extracción de acordes desde un archivo MIDI, procederemos a explicar los modelos utilizados en este trabajo. El modelo implementado ha sido una red BLSTM (una red LSTM bidireccional). Para poder comprender como funcionan y como deben de ser tratados los archivos MIDI, ha sido necesario utilizar una librería realizada por Carlos Hernández[1].

4.1.1. Librerías y Frameworks utilizadas

Para la realización del proyecto ha sido necesario analizar y modificar un algoritmo, el cual ha sido evaluado en Anaconda[2], empleando Python 3[3] como lenguaje. El código ha sido implementado en Visual Studio Code[4].

Para la correcta implementación de dicho algoritmo ha sido necesario instalar un nuevo entorno en Anaconda Promt,el cual ha sido denominado LSTM, así como diferentes librerías de Python: Numpy[5], Pandas[6], Tensorflow[7]. El trabajo ha sido realizado en el programa Overleaf[8].

[1] `https://github.com/carlosholivan/musicaiz` último acceso: agosto 2022
[2] `https://docs.anaconda.com/anaconda/user-guide/getting-started/` último acceso: agosto 2022
[3] `https://www.python.org/downloads/` último acceso: agosto 2022
[4] `https://code.visualstudio.com/` último acceso: agosto 2022
[5] `https://numpy.org/` último acceso: agosto 2022
[6] `https://pandas.pydata.org/` último acceso: agosto 2022
[7] `https://www.tensorflow.org/install` último acceso: agosto 2022
[8] `https://www.overleaf.com/project/6243367e416f341a02af85ce` último acceso: agosto 2022

4.2. Descripción del modelo de Tsung-Ping Chen y Li Su [1]

El entrenamiento de la red está basado en una mejora de un algoritmo ya implementado por parte de los investigadores Chen Tsung-Ping y Li Su[9]. El algoritmo original utiliza una base de datos, que utiliza para transformar 32 archivos en formato *.csv* en un archivo en formato *.ckpt*.

Para el correcto funcionamiento del modelo ha sido necesario ajustar los parámetros con los utilizados en el trabajo, implementando un código que permitiese el paso de MIDI (el archivo que utilizamos en nuestra investigación, y por lo tanto objetivo de la misma) a *.csv*. Para lograrlo se han utilizando diversas funciones del código fuente del modelo (también modificadas) que serán explicadas con detenimiento más adelante.

En la figura 4.1[10] podemos observar una descripción de cómo funciona el entrenamiento del modelo paso por paso, desde el archivo MIDI, hasta la extracción del acorde y su comparación con el acorde original (validación y testeo).

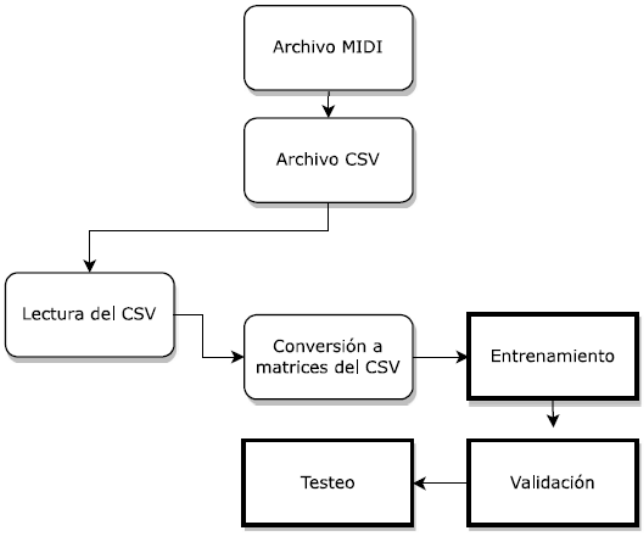

Figura 4.1: Diagrama descriptivo del funcionamiento del modelo.

[9] `https://github.com/Tsung-Ping/functional-harmony/tree/master/BPS_FH_Dataset` último acceso: agosto 2022

[10] `https://app.diagrams.net/?src=about` último acceso: agosto 2022

Previamente a la descripción de las modificaciones realizadas en el trabajo, es necesario explicar el modelo realizado por Tsung-Ping y Su[11] (la explicación de las pertinentes modificaciones del modelo será realizada posteriormente, en el apartado 4.3), el cuál nos permite entrenar una base de datos, basada en archivos en formato *.csv* de Ludwig van Bethoveen[12], para la extracción de acordes en base a varias redes neuronales.

En el modelo original, son empleadas 3 tipos de redes, una primera red Single Task Layer (STL), y 2 redes Multi-Task Layer (MTL). La primera de las 2 redes MTL emplea un único nivel de realización de tareas y la segunda incluye 2 niveles de realización de tareas. Para nuestro modelo únicamente analizaremos la red MTL con 1 nivel para la realización de tareas de entrenamiento, debido a que los resultados obtenidos por la red STL tienden a ser menos positivos que los de una red MTL, y con los 2 niveles de la red MTL, los resultados no mejoran, sino que únicamente varían.

4.2.1. Entrada de la red

La entrada de la red es una matriz $n \times m$ donde n es el número de notas a identificar y clasificar, y m es el valor temporal de una semifusa[13] en un compás. El número n hace referencia a las notas de entrada a la red (las filas de las que dispone el archivo de entrada), siendo estas las disponibles desde la nota C1 hasta la nota C6, es decir, un total de 5 escalas completas y la nota correspondiente a C de la escala de sexto orden. De forma que el total de notas disponibles son cada una de las 12 notas de cada escala (n, 4.2.1), lo que resulta de un total de 61 notas (al añadir la nota $C6$) disponibles a la entrada de la red.

En la figura 4.2[14] podemos observar una representación de la matriz de entrada de la red, dónde se dibuja la representación en formato MIDI de figuras(aclaración de las figuras disponible en el anexo A) como corcheas, negras o redondas (1/16,1/4 y 1 tempo respectivamente) .

En la ecuación 4.2.1 se hace referencia a cada una de las notas entre C1 y C6, las notas de posibles de entrada a la red.

[11] `https://github.com/Tsung-Ping/functional-harmony` último acceso: agosto 2022

[12] `https://github.com/Tsung-Ping/functional-harmony/tree/master/BPS_FH_Dataset` último acceso: agosto 2022

[13] `https://comamusical.com/semifusa/`

[14] `https://app.diagrams.net/?src=about` último acceso: agosto 2022

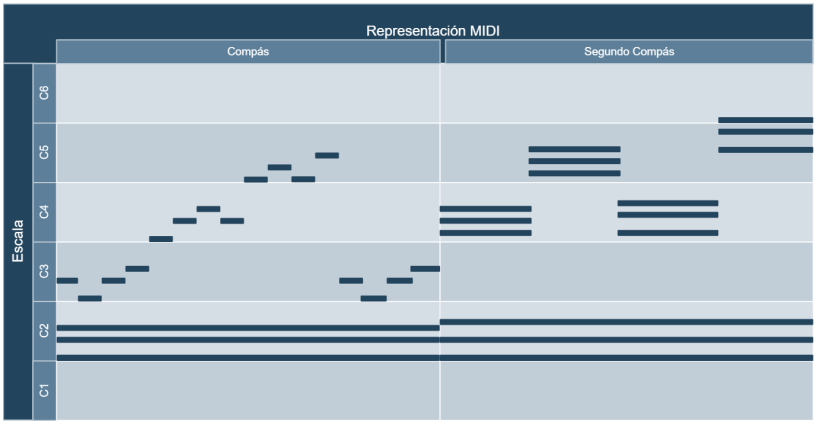

Figura 4.2: Representación de la matriz de entrada a la red.

$$n = C1, C1^{\#}, D1, D1^{\#}, E1, F1, F1^{\#}, G1, G1^{\#}, A1, A1^{\#}, B1, C2, C2^{\#},$$
$$D2, D2^{\#}, E2, F2, F2^{\#}, G2, G2^{\#}, A2, A2^{\#}, B2, C3, C3^{\#}, D3, D3^{\#}, E3,$$
$$F3, F3^{\#}, G3, G3^{\#}, A3, A3^{\#}, B3, C4, C4^{\#}, D4, D4^{\#}, E4, F4, F4^{\#}, G4, G4^{\#},$$
$$A4, A4^{\#}, B4, C5, C5^{\#}, D5, D5^{\#}, E5, F5, F5^{\#}, G5, G5^{\#}, A5, A5^{\#}, B5, C6$$

Los valores de entrada mencionados (n y m) nos permitirán saber en todo momento la nota que está siendo evaluada, así como su valor temporal (nos permitirá conocer si la nota equivale a una negra, una corchea, una redonda etc.). En la entrada podemos encontrar un total de 1024 parámetros de entrada para entrenar, lo que hace que podamos estudiar todas las obras musicales sin ningún problema, ya que las matrices de entrada tienen todas un tamaño menor.

La red analiza el contenido de cada uno de los archivos *.csv* para obtener la información pertinente sobre cada nota y sus respectivas características: la tonalidad, el primer y segundo grado, la condición y la inversión, características de las que hablaremos más adelante.

Los valores de cada archivo *.csv* los podemos observar en la figura 4.3 y son los siguientes:

- Inicio de la nota, nos indica en qué momento debe reproducirse el sonido de la nota en cuestión, y deben de ir ordenadas de la primera a la última.

- Número de nota o *pitch*, nos indica la nota en cuestión que debemos de tocar. La nota de referencia es el C1, cuyo valor en *pitch* es de 60.

- Número de tono morfético o *morphetic pitch*, nos indica el *pitch* de la nota en escala natural, es decir, en una escala diatónica (explicación en el anexo A), cuyas notas son C,D,E,F,G,A,B, sin incluir en ella notas con bemoles o sostenidos.

Disminuyendo el valor del *pitch* cuando las notas se representen en clave de sol y aumentando cuando la clave sea de fa. Para nuestro modelo consideraremos que este valor es idéntico al valor del *pitch*, ya que éste únicamente aporta información adicional, sin influir en el resultado del entrenamiento.

- La duración viene definida por la fracción de *ticks* (medida temporal de un MIDI), sobre la duración de un pulso dentro del compás (lo que equivale a la duración de una *negra*).

- El número de pentagrama o *staff number*, define el tipo de tonalidad en la que se está representando la nota. En caso de que dicha clave fuera de sol, el valor será 0. Cuando la clave representada sea la clave de fa, el valor de dicho parámetro será 0. Dado que no hay manera de hallar este parámetro desde un archivo MIDI, y que no influye en los resultados del entrenamiento, hemos optado por darle el valor de 0 para todas las notas.

- El número de compás, nos indica el compás al que pertenece cada nota, ordenado numéricamente de 0 al último compás. Se extrae directamente de cada instrumento perteneciente al MIDI.

En la figura 4.3[15] podemos observar una representación de los datos incluidos en cada archivo *.csv*.

```
1   -1.0,60,60,1.0,0,0
2   0.0,65,63,1.0,0,1
3   1.0,68,65,1.0,0,1
4   2.0,72,67,1.0,0,1
5   3.0,77,70,1.0,0,1
6   4.0,80,72,1.5,0,2
7   5.0,53,56,1.0,1,2
8   5.0,56,58,1.0,1,2
9   5.0,60,60,1.0,1,2
10  5.5,79,71,0.166666666667,0,2
11  5.66666666667,77,70,0.166666666667,0,2
12  5.83333333333,76,69,0.166666666667,0,2
13  6.0,53,56,1.0,1,2
14  6.0,56,58,1.0,1,2
15  6.0,60,60,1.0,1,2
16  6.0,77,70,1.0,0,2
```

Figura 4.3: Extracto de un archivo de entrada *.csv*.

[15]Figura capturada directamente del archivo

4.2.2. Arquitectura de la red

La red de Tsung-Ping Chen y Li Su [1] se basa en una red LSTM bidireccional (BLSTM) para la predicción armónica. la diferencia de dicha red con una LSTM corriente se basa en su memoria, ya que la red BLSTM recogerá información tanto de casos pasados de la red como de los casos futuros (es decir, la propia red retrocede para almacenar información de casos futuros en sucesos anteriores de cara a predecir sucesos futuros).

Para la realización del modelo, nos hemos basado en la red multi-task (MTL) Esta red está compuesta por los 5 niveles principales previamente expuestos (figura 4.1).

El modelo usa el Multi-task Deep Learning para predecir no sólo la tonalidad, sino los acordes y grados.

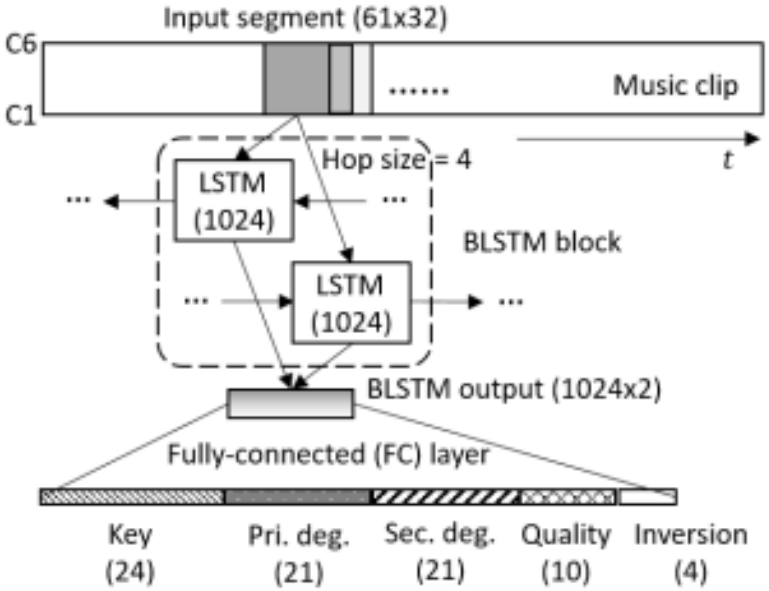

Figura 4.4: Funcionamiento de la red modelada por Tsung-Ping y Su [1]

Previamente a realizar las tareas de predicción, el modelo pretende transformar los datos de entrada introducidos de forma ordenada y numérica (como se ha desarrollado en el apartado previo), en una manera simbólica, es decir, se trata de realizar una traducción del código a una notación musical. Esto se realiza mediante una serie de funciones que recibe el archivo *chords* (cuyo almacenamiento podemos ver en la figura 4.6[16]) y devuelve la información de las notas en notación musical, etiquetándolas (en la figura 4.5 podemos observar una representación de como funcionaría la extracción de dichos archivos).

[16]extraído directamente de un archivo chords

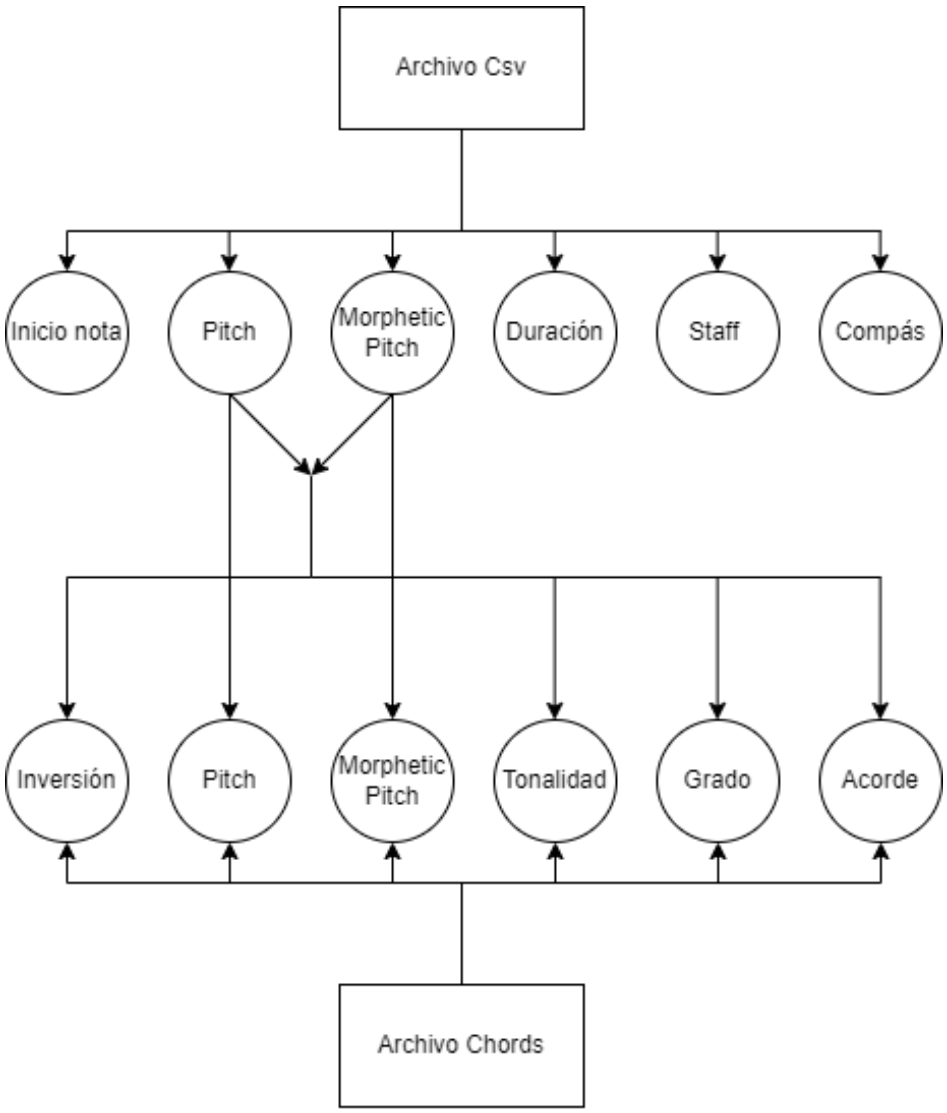

Figura 4.5: Diagrama explicativo del paso de *.csv* a *chords*

Para sacar cada uno de los valores identificadores de los acordes que aparecen como salida del modelo se tienen en cuenta 5 vectores diferentes. Estos son la tonalidad, el primer y segundo grado, su condición y su inversión.

Todos ellos vienen definidos en la red como una matriz 61x32 donde n es el número de notas disponibles (teniendo en cuenta un intervalo sonoro entre C1 y C6, un total de 5 escalas y una nota adicional) y m el valor de una fusa[17], equivalente a una redonda en un compás de 4x4 (debido a que la duración de una redonda sería 4 tiempos, y cada tiempo equivale a 8 ticks), o lo que es lo mismo, cada tempo se pueden reproducir 8 fusas.

[17] `https://comamusical.com/fusa/` último acceso: agosto 2022

20	24	f	5	D7	2	V43
24	26	f	1	m	1	i6
26	28	f	2	d	1	ii-6
28	31	f	5	D7	0	V
31	40	A-	3	m	0	iii
40	44	A-	4	M7	1	IV65
44	48	A-	2	m7	0	ii7
48	52	A-	5	D7	2	V43
52	56	A-	1	M	0	I
56	58	A-	2	m	1	ii6
58	60	A-	5/5	D7	1	V65/V
60	64	A-	5	M	0	V
64	66	A-	2	m	1	ii6

Figura 4.6: Extracto de un archivo *chords*.

Esto nos permitirá saber en todo momento la nota que está siendo evaluada, así como su valor temporal (nos permitirá conocer si la nota equivale a una negra, una corchea, una redonda etc.).

En la figura 4.6 podemos observar un fragmento de la nomenclatura de un archivo *chords*. Estos archivos contienen matrices de un tamaño $n \times m$, donde n equivale al número de notas que contiene el archivo, y m son las 7 características de cada una de las notas del mismo:

- Las primeras 2 columnas hacen referencia al comienzo y final de la nota a interpretar en formato *ticks*.

- En la tercera columna encontramos información referente a la tonalidad en la que debe de ser reproducida la nota.

- En la cuarta columna encontramos el grado de la nota, es decir, el lugar que ocupa dicha nota en el acorde de referencia de la tonalidad a reproducir.

- Para la quinta columna disponemos de información relevante al tipo de acorde que se representa con la nota, es decir, su condición (este concepto se desarrollará con más detenimiento a continuación).

- En la sexta columna analizaremos la inversión del acorde representado por la figura (el cuál, al igual que la condición del acorde, será analizado a continuación).

- En la última columna nos encontramos la notación numérica romana de la nota en cuestión. La notación numérica romana utiliza las siguientes reglas: cuando la nota es la nota tónica del acorde se representará con un I, cuando esta sea una nota super-tónica del acorde recibirá la por notación II, a una nota mediana se le denotara III, mientras que IV servirá para notas sub-dominantes, V para notas dominantes y VI y VII para notas sub-medianas y guías respectivamente.

En la tabla 4.1 podemos observar los valores que identifican a cada acorde:

- La tonalidad es un vector de 24 valores en donde van incluidos todos los posibles acordes, tanto mayores como menores. Para este trabajo definimos X como el vector de tonalidades que contiene el índice de cada tonalidad desde 0 a 23. En la ecuación 5.3 podemos observar todas las posibles tonalidades que puede detectar el modelo.

- Los grados identifican la posición de la raíz del acorde, los cuales pueden ser 7, es decir, identifica si el acorde está ordenado desde su tónica hasta la quinta. El primer grado (P) sirve para identificar la tónica, de manera temporal, mientras que el secundario (S) indica la posición temporal del fundamental en relación con la tónica.

- La condición viene definida por la distancia entre notas del acorde. Analiza las triadas y decide entre las 10 posibles (observables en C) condiciones: mayor (M), menor (m), disminuido (d), aumentado (a), séptima mayor (M7), séptima dominante (D7), séptima menor (m7), séptima disminuida (d7), séptima semi-disminuida (h7) sexta aumentada (a6).

- La inversión determina el orden de las notas en el acorde, si el orden es el original del acorde, su inversión será nula. Si la tercera nota del acorde es la base del acorde, será un acorde de primera inversión. Si la quinta nota es la base del acorde, éste será un acorde de segunda inversión. Por último, aquellos acordes que dispongan de séptima. y esta sea la base del acorde, estos acordes recibirán el nombre de tercera inversión. Esta característica viene dada por I.

$$X = \{X_C, X_{C\#}, X_D, X_{D\#}, X_E, X_F, X_{F\#}, X_G, X_{A\flat}, X_A, X_{B\flat}, X_B\} \tag{4.1}$$

$$C = \{M, m, a, d, M7, m7, D7, d7, h7, a6\} \tag{4.2}$$

$$P = \left\{C, C^{\#}, C^{\flat}, D, D^{\#}, D^{\flat}, E, E^{\#}, E^{\flat}, F, F^{\#}, F^{\flat}, G, G^{\#}, G^{\flat}, A, A^{\#}, A^{\flat}, B, B^{\#}, B^{\flat}\right\} \tag{4.3}$$

$$S = \left\{C, C^{\#}, C^{\flat}, D, D^{\#}, D^{\flat}, E, E^{\#}, E^{\flat}, F, F^{\#}, F^{\flat}, G, G^{\#}, G^{\flat}, A, A^{\#}, A^{\flat}, B, B^{\#}, B^{\flat}\right\} \tag{4.4}$$

$$I = \{0_{th}, 1_{th}, 2_{th}, 3_{th}\} \tag{4.5}$$

Vector	Dimensión	Etiqueta	Contenido
Tonalidad	24	X	24 Tonalidades, mayores o menores
Primer Grado	21	P	7 números romanos con 3 tipos (neutral, b o #)
Segundo Grado	21	S	7 números romanos con 3 tipos (neutral, b o #)
Condición	10	C	M,m,a,d,M7,m7,D7,d7,h7,a6
Inversión	4	I	Ninguna, primera, segunda, tercera

Tabla 4.1: Dimensión y contenido de las características de los acordes

4.2.3. Función de pérdidas

Una vez definido el funcionamiento del modelo, necesitamos utilizar una función de pérdidas para el correcto funcionamiento de la red, ya que será la herramienta que nos permita evaluar la veracidad de los datos entrenados por la red.

Para nuestro modelo, hemos utilizado una función de pérdidas de regresión, ya que el valor que deseamos hallar es un acorde, que lo identificamos con un valor continuo (ya que los acordes no varían su composición; si cambiara una nota del acorde, este pasaría a ser otro acorde).

En concreto se ha utilizado una función de pérdidas *L2* (o función de pérdida cuadrática), la cuál se mide como "el promedio de la diferencia al cuadrado entre las predicciones y las observaciones reales" [18] (ecuación 4.6). La particularidad de la función de pérdidas elegida se basa en el enfoque global de la detección de error, puesto que en lugar de enfocarse en hacia dónde se dirige el error, se preocupa de la desviación (en módulo) del error, tratando de disminuir la media de desviación lo máximo posible.

$$MSE = \left\{ \frac{\sum_{i=1}^{n} (y_i - \hat{y}_i{}^2)}{n} \right\} \tag{4.6}$$

[18] `https://sitiobigdata.com/2019/12/24/funciones-comunes-de-perdida-en-el-aprendizaje-automatico/`, último acceso: agosto 2022

4.2.4. Entrenamiento de la red

Para el entrenamiento de la red disponemos de la base de datos BPS-FH dataset[19] con 32 archivos *.csv*. Cada uno de los archivos contiene una obra musical de Beethoven[20]. A cada uno de estos archivos se le asigna una función de estudio, dividido en tres partes.

- La primera parte del entrenamiento se basa en el propio entrenamiento del modelo. Para esta tarea, la que más información necesita, utilizamos 11 archivos.
- Seguido al entrenamiento se realiza la validación de los archivos, para corroborar que el entrenamiento ha sido correcto. Se cuenta con 6 archivos en esta parte del entrenamiento.
- Por último, se realiza el testeo de los valores obtenidos, comparándolos con los ya conocidos, para lo que se cuenta con otros 6 archivos.

Podemos observar los archivos seleccionados para cada proceso del entrenamiento en la tabla **??**

4.3. Modificaciones del modelo e implementación de código del trabajo

Una vez desarrollado el modelo en el que se basa el trabajo, procedo a explicar las modificaciones realizadas sobre el mismo.

El entrenamiento de la red sería inviable sin la transcripción del archivo MIDI a archivo *.csv*, para posteriormente crear las notas que entran al modelo de entrenamiento. El proceso de transformación de la información del archivo está desarrollado gracias a la librería pandas[21], así como la librería de Carlos Hernández[22] [29] llamada *musanalisis*.

[19] `https://github.com/Tsung-Ping/functional-harmony/tree/master/BPS_FH_Dataset`, último acceso: agosto 2022

[20] `https://es.wikipedia.org/wiki/Ludwig_van_Beethoven`, último acceso: agosto 2022

[21] `https://pandas.pydata.org/` último acceso: agosto 2022

[22] `https://github.com/carlosholivan/musicaiz` último acceso: agosto 2022

Datos	Número de pieza
Entrenamiento	1,3,5,11,16,19,20,22,25,26,32
Validación	6,13,14,21,23,31
Testeo	8,12,18,24,27,28

Tabla 4.2: Relación de piezas para cada tarea.

Para la correcta extracción de los datos del archivo MIDI ha sido necesario implementar un código adicional al del modelo de Tsung-Ping y Su [1], que permitiera la correcta lectura de los datos de los archivos *.csv*. Para ello ha sido necesario crear 6 librerías (una por cada característica del archivo) para transcribir su información a las características necesarias del archivo *.csv*. De manera que cada una de las librerías se abra para cada nota, dentro de cada compás, incluidos en cada uno de los instrumentos pertenecientes al archivo MIDI. Así pues, al término del conversor, el archivo MIDI de origen es reemplazado por un archivo *.csv* de entrada en la red. Para que la lectura de los archivos *.csv* fuera correcta ha sido imprescindible introducir valores manuales, como el valor 0 del *staff number* para todas las notas, o el *morphetic pitch* de valor idéntico al *pitch*.

En la figura 4.7 podemos observar una simulación de cómo conseguiríamos determinar el acorde, mediante el entrenamiento del modelo, de una manera visual. Cada uno de los niveles nos muestra la manera en la que el modelo organiza y analiza la información recibida por el archivo.

Además del desarrollo de un conversor csv-MIDI, ha sido necesario implementar un algoritmo con el objetivo de que, a través de los datos del entrenamiento del modelo de Tsung-Ping, podamos predecir los acordes que van a ser tocados en obras posteriores, mediante la extracción de notas aleatorias de una obra cualquiera.

Para ello, ha sido necesario ha sido necesaria la utilización de las librerías mencionadas en la sección 4.1, así como la extracción de las funciones *load chord labels*, *segment pianorolls*, *prepare input data*,*split input data* y *load notes* del modelo de Tsung-Ping y Su, así como la librería musanalysis[23].

De modo que al ajustar los parámetros de longitud de red (ya que originalmente se entrenaba una base de datos[24], y ahora únicamente necesitaremos el archivo a predecir para comparar las predicciones obtenidas con los resultados correctos) e introducir los parámetros pertinentes para la comparación de la validación, el testeo y el entrenamiento de la red (todos ellos explicados con detenimiento en el apartado 5), obtendremos los porcentajes de acierto de la predicción de la red.

[23] `https://carlosholivan.github.io/musicaiz/` Último acceso: agosto 2022

[24] `https://github.com/Tsung-Ping/functional-harmony/tree/master/BPS_FH_Dataset` último acceso: agosto 2022

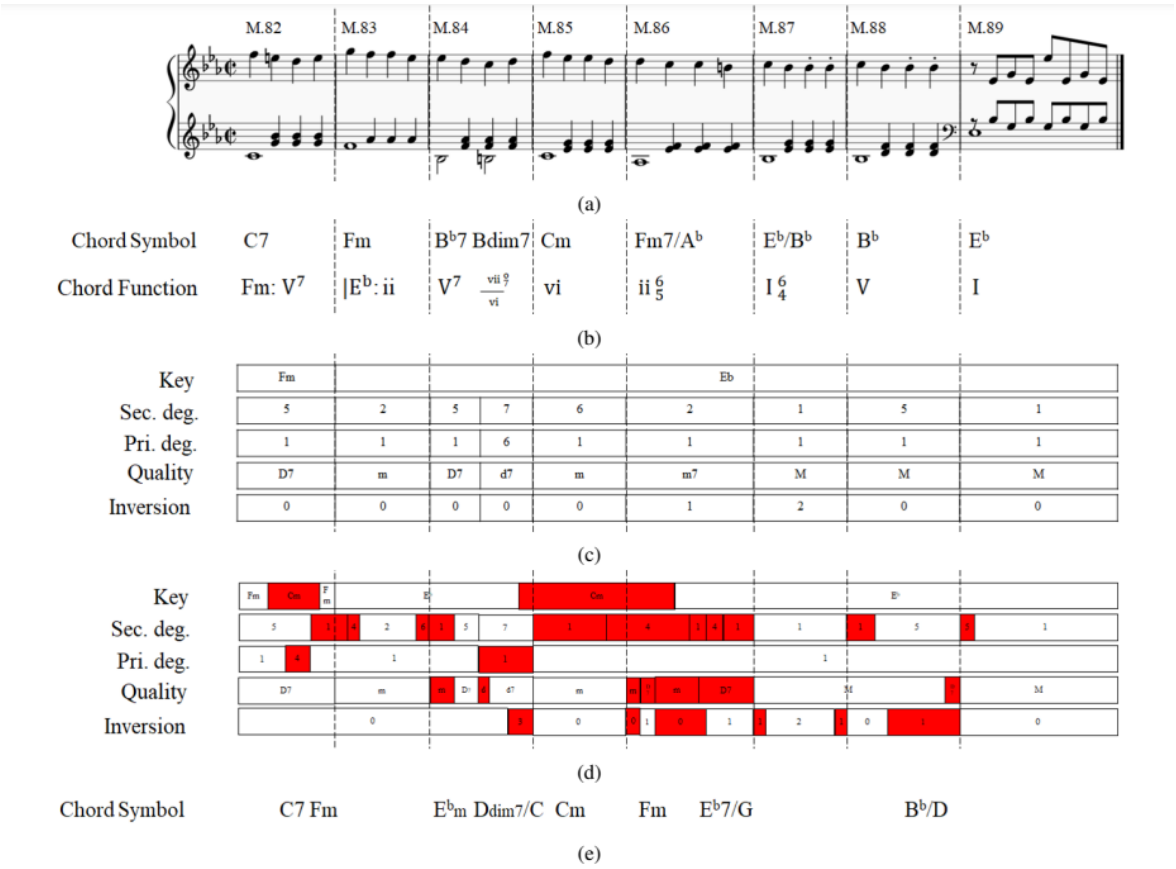

Figura 4.7: Representación de la denominación de acordes desde un extracto de partitura (a), pasando por el archivo *chords* (b), descomponiéndolo en los 5 niveles característicos de cada acorde (c), cuya representación real observamos en (d), para extraer el nombre del acorde (e). [1]

Capítulo 5

Experimentos y Resultados

5.1. Base de Datos

Como hemos comentado en el capítulo 4 la base de datos utilizada para entrenar y validar el modelo es la BPS-Harmony Dataset [1] [1].

Los archivo se separarán en 3 grupos, por un lado estarán los archivos que vamos a utilizar para el entrenamiento, por otro los correspondientes a la validación de y por otro los utilizados en el testeo. Su evaluación se realizará por separado una vez hallamos recorrido el código del modelo.

5.2. Predicción de Resultados

Para poder realizar la lectura y comprobación del modelo, se ha programado un conversor con el objetivo de convertir archivos MIDI a la representación que requiere la entrada de la red (archivos *.csv*), para ser capaces de hacer predicciones en este tipo de archivos (MIDI) (puesto a que son los más utilizados en la industria de la música para labores de producción musical, grabación etc).

Este algoritmo ha sido realizado utilizando funciones propias del modelo de Tsung-Ping y Su [1]. Ha sido necesario implementar también un algoritmo que permitiera el paso de archivo MIDI a las características propias de los archivos *.csv*, con el objetivo de introducir los datos del archivo *.csv* en el modelo. Una vez los datos del *.csv* son introducidos en el modelo, estos comienzan a analizarse, organizando las notas representativas de cada fila en los distintos acordes de los que puede formar parte. Posteriormente, se compara cada acorde extraído de los archivos *.csv* con los acordes reales del mismo (extraídos de la base de datos de la red).

[1] `https://github.com/Tsung-Ping/functional-harmony/tree/master/BPS_FH_Dataset`

5.3. Métricas de Evaluación

Las métricas de la evaluación[2] se utilizan para medir el rendimiento del modelo de predicción, asegurando que los resultados que obtenemos no son erróneos, basándose en la matriz de confusión (figura 5.1[3]). Los resultados que queremos obtener idealmente son únicamente los True Positive (TP).

Para el modelo existen tres métricas fundamentales de evaluación, precisión, recuerdo y exactitud.

- La precisión (o *precision*) hace referencia a la frecuencia con la que un sí es correcto. O lo que es lo mismo, el número de predicciones correctas en proporción con todas las predicciones que se han realizado ($PREC$).
- La exhaustividad (o *recall*) relaciona únicamente los verdaderos positivos, es decir, trata de identificar cuáles de los positivos son realmente verdaderos positivos. Su notación es *recall*.
- La exactitud (o *accuracy*) es la medida más directa de la calidad de los evaluadores. Su valor varía entre 0 y 1, siendo 1 el valor deseado y 0 el valor a evitar. Para nuestro trabajo la denominaremos ACC

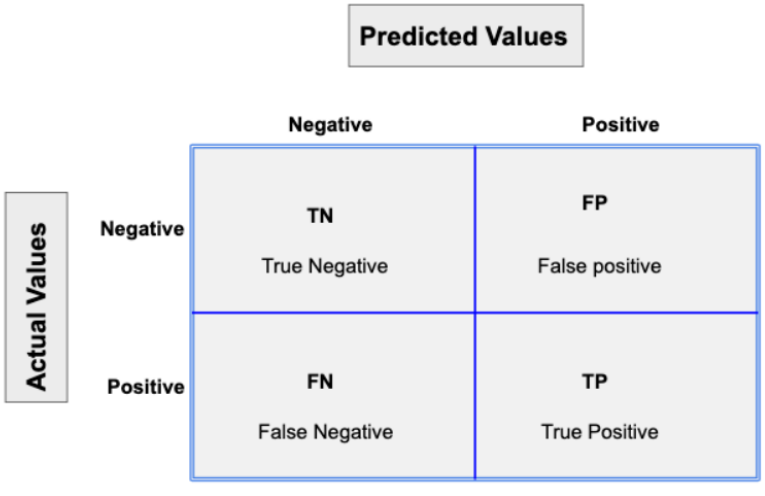

Figura 5.1: Matriz de confusión en la evaluación de resultados.

Todos los valores empleados en las fórmulas hacen referencia a la notación de la tabla perteneciente a la figura 5.1.

[2] https://www.datasource.ai/es/data-science-articles/metricas-de-evaluacion-de-modelos-en-el-aprendizaje-automatico

[3] https://www.datasource.ai/es/data-science-articles/metricas-de-evaluacion-de-modelos-en-el-aprendizaje-automatico último acceso: agosto 2022

$$PREC = \left\{ \frac{TP}{Predicciones_{Si}} \right\} \tag{5.1}$$

$$recall = \left\{ \frac{TP}{TP + FN} \right\} \tag{5.2}$$

$$ACC = \left\{ \frac{TP + TN}{TP + TN + FP + FN} \right\} \tag{5.3}$$

Para la ejecución y entrenamiento del modelo, vamos a basar nuestro estudio en la métrica de *accuracy* únicamente, ya que es la medida que mayor efectividad tiene y también la que mejor estima provee.

5.4. Experimentos

Se ha entrenado el modelo MTL con un sólo nivel de tareas específico, debido a que los resultados por este método deben de ser más óptimos que los obtenidos mediante una red Singular-Task Layer (STL), la cuál realizaría únicamente un cálculo o tarea cada vez, y la red MTL con 2 niveles de tareas específicos.

Los hiperparámetros elegidos para el modelo se encuentran en la figura 5.2 y han sido:

- El *batch-size*, que viene definido por el número de notas pertenecientes a cada uno de los csv (variará con el archivo a analizar).

- El espacio característico, o *feature size*, nos define los n espacios disponibles en el modelo para almacenar las variables. En nuestro modelo tomará un valor de 1952, ya que para valores superiores el modelo comenzaría a entrar en la zona de *overfitting* («*Es decir, nos encontramos en la situación que el modelo puede tener una baja tasa de error de clasificación para los datos de entrenamiento, pero no se generaliza bien a la población general de datos en los que estamos interesados en realidad*[4]»), por lo que los datos obtenidos de la predicción de resultados serían erróneos.

- El número de pasos en los que se realiza cada una de las iteraciones de entrenamiento del modelo es 64.

- El número de unidades ocultas (o *hidden units*) será de 1024. Estos niveles permiten a la red almacenar múltiples entradas en cada una de sus unidades, de manera que cuando se necesiten pueden ser sustraídas.

» [4] `https://torres.ai/datos-y-overfitting-keras-tensorflow/` último acceso: agosto 2022

– La tasa de aprendizaje (o *learning rate*) nos indica el tamaño de paso de cada iteración, siempre teniendo cuidado de no entrar en el mínimo de función de pérdida, lo cuál nos daría datos erróneos en las iteraciones. En nuestro modelo tomará un valor de 0.0004.

– Por último debemos de tener en cuenta la regulación de abandonos en las iteraciones, o *dropout rate*, un parámetro que nos permite ignorar datos aleatorios que nos encontremos en el modelo (en nuestro caso estos serían notas sueltas, que no perteneciesen a ningún acorde, sino que simplemente fueran notas de paso). En nuestro modelo tomará un valor de 0,5.

Aparte de los mencionados, se utilizarán también

```
tf.reset_default_graph()
network = MTL_BLSTM_RNNModel(feature_size=1952,
                             n_steps=64,
                             n_hidden_units=1024,
                             learning_rate=1e-4,
                             dropout_rate=0.5)
```

Figura 5.2: Declaración de hiperparámetros (extraída del propio código)

Para el entrenamiento de nuestro modelo hemos decidido realizar un total de 27 iteraciones o *epoch* («*El número de epoch es un hiperparámetro que define el número de veces que el algoritmo de aprendizaje funcionará en todo el conjunto de datos de entrenamiento*»[5]). Con cada una de las iteraciones el modelo irá aprendiendo sobre lo estudiado en las anteriores y será capaz de decidir por sí misma como afrontar la predicción de los acordes en la siguiente iteración. Decidimos que el número fuera 27 ya que fué el escogido en el modelo original, y para valores mayores el modelo comenzaba a automatizar las predicciones. Esto quiere decir, que en lugar de intentar comprender el funcionamiento del modelo, aprende el funcionamiento de los ejemplos dados como entrada de la red, lo que haría que para los ejemplos integrados en el modelo la predicción funcionara perfectamente, pero si cambiáramos los archivos a predecir, el modelo sería completamente incapaz de dar con una correcta predicción (es semejante al concepto *aprender de memoria* del cerebro humano, sin recapacitar sobre lo que se está aprendiendo, únicamente absorbiendo la información).

[5] `https://deepai.org/machine-learning-glossary-and-terms/epoch` último acceso: agosto 2022

5.5. Comparación de los resultados obtenidos

Una vez terminamos el entrenamiento, con las métricas y experimentos de evaluación ya definidos, debemos obtener los resultados de nuestro modelo y compararlos con los resultados que obtuvieron Tsung-Ping y Su [1] con el suyo.

Éstos resultados han sido evaluados en 3 paquetes , separando los archivos utilizados en el testeo, de los de la validación de los de el entrenamiento (obteniendo por lo tanto unos porcentajes de precisión en el entrenamiento del modelo, otros distintos para la validación del mismo y otros diferentes para el testeo), y comparando su porcentaje de acierto en tonalidad, primer y segundo grado, cualidad, inversión y precisión global con los modelos realizados por Tsung-Ping y Su. En este caso se han elegido dos modelos para la comparación, tanto el modelo STL (pese a que los resultados deberían de ser a priori peores que los que hallamos obtenido en nuestro modelo, debemos asegurarnos) como el modelo MTL con 1 nivel de realización de tareas (con el objetivo de igualarlos o mejorarlos).

Para la comparación del modelo se van a tener en cuenta los resultados obtenidos en el testeo del modelo, ya que son los que realmente podemos comparar con los obtenidos en el modelo de referencia.

En la tabla 5.1 se compararán los datos obtenidos por Tsung-Ping y Su en el testeo los modelos mencionados en el anterior párrafo con la precisión obtenida en nuestro proceso de testeo. Además, podemos observar los resultados de la validación y entrenamiento de nuestro modelo.

5.6. Discusión de los Resultados

Una vez obtenidos y guardados los resultados del entrenamiento del modelo en la tabla 5.1, debemos descifrar dichos datos para saber cómo de positivo o negativo a sido el modelo implementado.

Vamos ha realizar la comparación de dichos resultados:

– En la tabla 5.1 observamos una gran mejoría en el testeo de los resultados de nuestro modelo con respecto al de Tsung-Ping y Su. La tonalidad a sido probada

Tarea	**Modelo**	Tonalidad	Primer Grado	Segundo Grado	Cualidad	Inverión	Global
Chord Function	MTL-BLSTM-RNN with 1 task-specific layer [1]	68.48	50.49	10.96	62.31	60.04	25.53
Chord Function	STL-BLSTM-RNN [1]	67.06	48.31	9.38	61.87	57.95	23.57
Chord Function	Testeo de nuestro modelo	77.75	60.51	65.43	71.01	64.43	33.13
Chord Function	Validación de nuestro modelo	76.59	57.80	63.67	69.13	64.53	30.98
Chord Function	Entrenamiento de nuestro modelo	96.84	84.19	87.94	88.44	84.07	67.76

Tabla 5.1: Resultados del testeo, validación y entrenamiento de nuestro modelo y del testeo de los modelos originales de Tsung-Ping y Su

con una eficacia de 77.45 %, un 10.59 % mayor que en el modelo STL (lo cuál era previsible, porque como ya había dicho este modelo realiza únicamente una tarea cada vez, ralentizando el proceso), y un 9,27 % mayor que el modelo MTL, lo que es un resultado muy positivo para nuestro modelo. A su vez, podemos observar que el valor de la precisión en el testeo de nuestro modelo es de 33.13 %, un 7,6 % mayor que el modelo de referencia MTL. Éste es el parámetro global de acierto completo en el acorde, y una diferencia positiva tan grande nos indica que la implementación del modelo en cuanto al testeo ha sido muy positiva.

- En cuanto a la validación del modelo, podemos observar que ésta ha sido también positiva, puesto que todos los valores son superiores a los de los modelos de Tsung-Ping y Su, (pese a que como ya he mencionado anteriormente estos resultados son en cuanto al testeo, por lo que no son semejantes a los de valoración, el hecho de que sean superiores no deja de ser positivo).

- Podemos observar que tanto los valores de testeo como los de validación son superiores a los obtenidos en el testeo de las redes originales. Pero los valores de testeo son los que realmente nos dan una imagen de la efectividad que tiene el modelo a la hora de extraer acordes correctamente, mientras que los valores de validación únicamente nos indican la precisión con la que el modelo extrae un acorde de la red.

- De igual forma observamos que los resultados de entrenamiento son muy elevados, lo que nos indica que el modelo ha sido ejecutado correctamente. Siendo un 96.84 % en el entrenamiento de detección de la tonalidad y de hasta un 67.76 % en la detección global del acorde, por lo que podemos decir que en entrenamiento de la red es realmente capaz de determinar acordes de una manera muy completa.

  Estos valores son realmente superiores, pero ello no es una casualidad. El entrenamiento o (*training accuracy*) únicamente nos muestra el valor con el que el modelo entrena correctamente, es decir, es capaz de leer los archivos y transformarlos en una respuesta en el formato correcto de salida. Éstos datos únicamente nos muestran que nuestro modelo entrena correctamente, ya que son valores realmente elevados, pero los valores a tener en cuenta para la conclusión final del modelo (y por lo tanto los valores realmente relevantes del trabajo realizado) son los resultados extraídos del testeo, de la tabla 5.1.

- Estos últimos resultados extraídos del entrenamiento del modelo son realmente superiores a los de los modelos anteriores, y esto tiene una explicación. Éstos valores realmente no representan el grado de acierto en la determinación de

acorde de la red, sino que representan la probabilidad de que la red extraiga datos relativos a un acorde por cada fichero entregado a la misma, es decir, el porcentaje de segundo grado. Por ejemplo, nos indica que con un 87.94 % de probabilidad, el modelo extraerá un valor para el segundo grado del acorde que se está reproduciendo, pero no nos indica que esta predicción vaya a estar bien o mal, para ello nos deberemos de fijar en el testeo de los resultados del modelo (representada en la tabla5.1).

Capítulo 6

Conclusiones

A lo largo del desarrollo del trabajo se ha ido elaborando una posible solución para la detección de acordes de manera programada, mediante la implementación de modelos en ML. Como hemos podido observar, la utilización de estos métodos, pese a que no son nuevos, cada vez tiene más peso en la industria musical, siendo numerosas las empresas que a día de hoy se apoyan en estos métodos para consolidar sus proyectos futuros. De la mano del *ISMIR* [4] se han ido desarrollando diferentes modelos en la descomposición musical desde múltiples archivos (como pueden ser *mp3, wav* o el propio MIDI), consiguiendo cada vez mejores resutados en su estudio, tanto en la extracción de acordes, con en la automatización de la creación de los mismos, de manera que la comunidad va mejorando sus métodos de manera retroactiva. Pese a que el avance de la extracción musical cada vez son mayores, la mayor parte de los estudios se centran en la extracción a partir de otros archivos de audio, es por esto que nuestro trabajo supone un gran avanze para el análisis de un archivo MIDI. El estudio realizado en este trabajo permite mejorar la extracción de acordes procedentes de un archivo MIDI mediante la mejora de un modelo realizado por Tsung-Ping y Su. De manera que de los resultados de este trabajo se puede extraer que:

- El entrenamiento del modelo es positivo, lo que permitirá su utilización para futuros modelos.
- Los resultados de acierto en la detección de acorde son mejores que los del modelo tomado cómo referencia.

6.1. Líneas Futuras

Gracias a este trabajo, hemos podido mejorar la efectividad en la extracción de los acordes de un archivo MIDI, lo que ayudará a los futuros investigadores, creándoles una línea de desarrollo en la que pueden unir distintos tipos de archivos, y trasladar

información de uno a otro sin ningún problema, controlando en todo momento la información contenida en el mismo.

Es evidente que la aportación de este trabajo será influyente sobre todo en el campo de la composición musical, por lo que las líneas futuras se basarán sobre todo en ese campo. Algunas de ellas pueden ser: desarrollar un modelo automático de composición de acordes en concordancia con obras realizadas (se podría realizar a partir de la base de datos empleada en este trabajo, para crear obras en concordancia con las del artista empleado en la base de datos), automatización en la organización de canciones según acordes, como objetivo de desarrollar una inteligenica artificial capaz de componer canciones personalizadas a los gustos del oyente según acordes extraidos de sus canciones escuchadas, o generar redes que permitan la creación automática de instrumentales multi-instrumento. Los resultados obtenidos en este trabajo son muy positivos, pero sería interesante seguir profundizando en la robustez del sistema mediante un análisis robusto para sacar conclusiones más globales con otro tipo de bases de datos, mediante barridos paramétricos.

6.2. Valoración Personal

En lo personal, la elaboración de este trabajo ha supuesto un verdadero reto, a la hora de abordar un trabajo amplio en un campo (el musical) por el que tengo un gran interés.

Me gustaría hacer especial hincapié en los conocimientos obtenidos durante y previamente al trabajo, tanto para el entendimiento de redes neuronales, como la implementación de un modelo de aprendizaje profundo y la comprensión del lenguaje informático utilizados. Dado que en el grado de Ingeniería de Tecnologías Industriales, los conocimientos en estos ámbitos han sido desarrollados de una manera muy poco concisa, el poder realizar este proyecto me ha ayudado a poder desarrollar conocimientos sobre esta disciplina.

Pese a la necesidad de volcarme en desarrollar conocimiento en las múltiples áreas mencionadas y en programas desconocidos previamente por mi persona (como pueden ser *python*, *overloaf*, *Visual Studio Code*, *Anaconda*, *GitHub* etc.), la oportunidad de poder introducir mis conocimientos musicales en el mundo de la ingeniería ha sido una experiencia muy positiva de cara a mi desarrollo futuro. Gracias a este trabajo he podido vislumbrar diferentes caminos por los que unir los dos campos en los que me he ido adentrando a lo largo de mi vida.

Las puertas abiertas a raíz de la realización de este proyecto son innumerables, y sólo me queda agradecer a José Ramón Beltrán y Carlos Hernández Oliván por darme

la oportunidad de unir ambos caminos para un futuro ilusionante.

Capítulo 7

Bibliografía

[1] Tsung-Ping Chen and Li Su. Attend to chords: Improving harmonic analysis of symbolic music using transformer-based models. *Trans. Int. Soc. Music. Inf. Retr.*, 4(1):1–13, 2021.

[2] `https://promocionmusical.es/inteligencia-artificial-musica-herramientas-aplicaciones/`. Último acceso: agosto 2022.

[3] Li-Chia Yang and Alexander Lerch. On the evaluation of generative models in music. *Neural Computing and Applications*, 32(9):4773–4784, 2020.

[4] `https://ismir.net/`. Último acceso: agosto 2022.

[5] Warren S. McCulloch and Walter H. Pitts. A logical calculus of the ideas immanent in nervous activity. In Margaret A. Boden, editor, *The Philosophy of Artificial Intelligence*, Oxford readings in philosophy, pages 22–39. Oxford University Press, 1990.

[6] David Temperley. What's key for key? the krumhansl-schmuckler key-finding algorithm reconsidered. *Music Perception*, pages 65–100, 1999.

[7] MAP Neshadha Perera and SR Kodithuwakku. Music chord recognition using artificial neural networks. In *Proceedings of the International Conference on Information and Au-tomation*, pages 304–308, 2005.

[8] Tsung-Ping Chen and Li Su. Functional harmony recognition of symbolic music data with multi-task recurrent neural networks. In Emilia Gómez, Xiao Hu, Eric Humphrey, and Emmanouil Benetos, editors, *Proceedings of the 19th International Society for Music Information Retrieval Conference, ISMIR 2018, Paris, France, September 23-27, 2018*, pages 90–97, 2018.

[9] https://blog.landr.com/es/que-es-el-midi-la-guia-del-principiante-para-la-herramienta-musical-mas-poderosa/. Último acceso: agosto 2022.

[10] Soyuz. https://www.hispasonic.com/reportajes/protocolo-midi/13#section-3. Último acceso: agosto 2022.

[11] https://www.sweetwater.com/insync/note/. Último acceso: agosto 2022.

[12] Carol L Krumhansl and Edward J Kessler. Tracing the dynamic changes in perceived tonal organization in a spatial representation of musical keys. *Psychological review*, page 334, 1982.

[13] Daniele Paolo Radicioni and Roberto Esposito. Tonal harmony analysis: A supervised sequential learning approach. In Roberto Basili and Maria Teresa Pazienza, editors, *AI*IA 2007: Artificial Intelligence and Human-Oriented Computing, 10th Congress of the Italian Association for Artificial Intelligence, Rome, Italy, September 10-13, 2007, Proceedings*, volume 4733 of *Lecture Notes in Computer Science*, pages 638–649. Springer, 2007.

[14] Raúl Arrabales. https://www.xataka.com/robotica-e-ia/deep-learning-que-es-y-por-que-va-a-ser-una-tecnologia-clave-en-el-futuro-de-la-inteligencia-artificial. Último acceso: agosto 2022.

[15] "https://promocionmusical.es/inteligencia-artificial-musica-herramientas-aplicaciones/. Último acceso: agosto 2022.

[16] Schmidhuber. http://colah.github.io/posts/2015-08-Understanding-LSTMs/. Último acceso: agosto 2022.

[17] Yu Zhang and Qiang Yang. A survey on multi-task learning. *IEEE Transactions on Knowledge and Data Engineering*, pages 1–1, 2021.

[18] https://miro.medium.com/max/850/1*jdkVep6NTz8umOViKDPusA.png. Último acceso: agosto 2022.

[19] Kristen Masada and Razvan Bunescu. Chord recognition in symbolic music: A segmental crf model, segment-level features, and comparative evaluations on classical and popular music. 2018.

[20] Matthew McVicar. A machine learning approach to automatic chord extraction. Tesis doctoral, University of Bristol, 2010.

[21] José García. https://www.xataka.com/aplicaciones/nanana-a-tu-cancion-favorita-asi-funciona-inteligencia-artificial-google-que-encuentra-canciones-solo-tatarearlas. Último acceso: agosto 2022.

[22] Alessio Degani, Marco Dalai, Riccardo Leonardi, and Pierangelo Migliorati. Harmonic change detection for musical chords segmentation. In *2015 IEEE International Conference on Multimedia and Expo, ICME 2015, Turin, Italy, June 29 - July 3, 2015*, pages 1–6. IEEE Computer Society, 2015.

[23] Veronika Zenz and Andreas Rauber. Automatic chord detection incorporating beat and key detection. In *2007 IEEE International Conference on Signal Processing and Communications*, pages 1175–1178. IEEE, 2007.

[24] Alexander Sheh and Daniel P. W. Ellis. Chord segmentation and recognition using em-trained hidden markov models. In *ISMIR 2003, 4th International Conference on Music Information Retrieval, Baltimore, Maryland, USA, October 27-30, 2003, Proceedings*, 2003.

[25] Taemin Cho and Juan P Bello. Real-time implementation of hmm-based chord estimation in musical audio. In *Proceedings of the International Computer Music Conference (ICMC)*, pages 16–21, 2009.

[26] Jun-qi Deng and Yu-Kwong Kwok. A hybrid gaussian-hmm-deep learning approach for automatic chord estimation with very large vocabulary. In Michael I. Mandel, Johanna Devaney, Douglas Turnbull, and George Tzanetakis, editors, *Proceedings of the 17th International Society for Music Information Retrieval Conference, ISMIR 2016, New York City, United States, August 7-11, 2016*, pages 812–818, 2016.

[27] Filip Korzeniowski and Gerhard Widmer. A fully convolutional deep auditory model for musical chord recognition. *CoRR*, abs/1612.05082, 2016.

[28] Jonggwon Park, Kyoyun Choi, Sungwook Jeon, Dokyun Kim, and Jonghun Park. A bi-directional transformer for musical chord recognition. *CoRR*, abs/1907.02698, 2019.

[29] Carlos Hernández Oliván. https://carlosholivan.github.io/musicaiz/. Último acceso: agosto 2022.

[30] http://www.tonalsoft.com/pub/pitch-bend/pitch.2005-08-31.17-00.aspx. Último acceso: agosto 2022.

[31] https://planetamusik.com/blog/archivo-midi-musica/. Último acceso: agosto 2022.

[32] H. Farnsworth. What-if machine analysis and design. *IEEE Transactions on quantum neuroscience electronics*, 3 031.

[33] Alessio Degani. *Harmonic Change Detection for MusicalESTO NO ES ASI Chords Segmentation*. Anaconda, 2015.

[34] Tomas Rocher Matthias Robine Pierre Hanna Robert Strandh. Dynamic chord analysis for symbolic music. *Proceedings of the International Computer Music Conference*, 2009.

[35] Wenchang Yang and Gudrun Magnusdottir. yang 2016, 10 2016.

[36] Tomas Rocher Matthias Robine Pierre Hanna Robert Strandh. Dynamic chord analysis for symbolic music. *Proceedings of the International Computer Music Conference*, 2009.

[37] Usroastserie-team. Cómo detectar acordes en la música. *Usroastserie*, 2022.

[38] David Meredith. Computing pitch names in tonal music: A comparative analysis of pitch spelling algorithms. Trabajo fin de grado, St. Anne's College, University of Oxford, 2007.

[39] https://www.cavsi.com/preguntasrespuestas/que-es-midi/. Último acceso: agosto 2022.

[40] Planeta Musik [online User]. Qué es un midi y cómo usarlo. *Planeta Musik*, 2017.

[41] Salvador Govea. https://soundcheck.com.mx/un-acercamiento-al-lenguaje-midi-segunda-parte/. Último acceso: agosto 2022.

[42] Carlos Hernandez-Olivan, Jorge Abadias Puyuelo, and Jose R. Beltran. Subjective evaluation of deep learning models for symbolic music composition, 2022.

[43] Carlos Hernandez-Olivan and José Ramón Beltrán. Music composition with deep learning: A review. *CoRR*, abs/2108.12290, 2021.

[44] Ingo Lütkebohle. http://aiweb.techfak.uni-bielefeld.de/content/bworld-robot-control-software/. Último acceso: junio 2022].

[45] Shulei Ji, Jing Luo, and Xinyu Yang. A comprehensive survey on deep music generation: Multi-level representations, algorithms, evaluations, and future directions. *CoRR*, abs/2011.06801, 2020.

[46] Engineering National Academies of Sciences, Medicine, et al. Reproducibility and replicability in science. 2019.

[47] Livio Grasso. *Encuestas. Elementos para su diseño y análisis.* Editorial Brujas, 2006.

[48] La textura de la música, abc. `https://www.abc.com.py/edicion-impresa/suplementos/escolar/la-textura-en-la-musica-1253837.html`. Último acceso: junio 2022.

[49] Definición de inteligencia artificial según lexico, diccionario de oxford. `https://www.lexico.com/definition/artificial_intelligence/`. Último acceso: junio 2022.

[50] Definición de deep learning según lexico, diccionario de oxford. `https://www.lexico.com/definition/deep_learning`. Último acceso: junio 2022.

[51] Stuart J. Russell and Peter Norvig. *Artificial Intelligence: A Modern Approach (4th Edition).* Pearson, 2020.

[52] Yoshua Bengio, Yann LeCun, and Geoffrey E. Hinton. Deep learning for AI. *Commun. ACM*, 64(7):58–65, 2021.

[53] Yann LeCun, Yoshua Bengio, and Geoffrey E. Hinton. Deep learning. *Nat.*, 521(7553):436–444, 2015.

[54] Lluís A. Belanche Muñoz. Feed-forward artificial neural network basics. In Juan R. Rabuñal, Julian Dorado, and Alejandro Pazos, editors, *Encyclopedia of Artificial Intelligence (3 Volumes)*, pages 639–646. IGI Global, 2009.

[55] I.A Basheer and M Hajmeer. Artificial neural networks: fundamentals, computing, design, and application. *Journal of Microbiological Methods*, 43(1):3–31, 2000. Neural Computting in Micrbiology.

[56] Muneyuki Unehara, Koichi Yamada, and Takuma Shimada. Subjective evaluation of music with brain wave analysis for interactive music composition by IEC. In *2014 Joint 7th International Conference on Soft Computing and Intelligent Systems (SCIS) and 15th International Symposium on Advanced Intelligent*

Systems (ISIS), Kita-Kyushu, Japan, December 3-6, 2014, pages 66–70. IEEE, 2014.

[57] Anders Krogh. What are artificial neural networks? *Nature biotechnology*, 26(2):195–197, 2008.

[58] Rnas: qué son y cómo se entrenan. `https://www.xeridia.com/blog/redes-neuronales-artificiales-que-son-y-como-se-entrenan-parte-`. Último acceso: junio 2022.

[59] Shifei Ding, Hui Li, Chunyang Su, Junzhao Yu, and Fengxiang Jin. Evolutionary artificial neural networks: a review. *Artificial Intelligence Review*, 39(3):251–260, 2013.

[60] ¿cómo aprenden las redes neuronales? `https://docs.microsoft.com/es-es/archive/msdn-magazine/2019/april/artificially-intelligent-how-do-neural-networks-learn`. Último acceso: junio 2022.

[61] Sagar Sharma, Simone Sharma, and Anidhya Athaiya. Activation functions in neural networks. *towards data science*, 6(12):310–316, 2017.

[62] Diederik P. Kingma and Max Welling. Auto-encoding variational bayes. In Yoshua Bengio and Yann LeCun, editors, *2nd International Conference on Learning Representations, ICLR 2014, Banff, AB, Canada, April 14-16, 2014, Conference Track Proceedings*, 2014.

[63] Tutorial - what is a variational autoencoder? `https://jaan.io/what-is-variational-autoencoder-vae-tutorial/`. Último acceso: junio 2022.

[64] Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron C. Courville, and Yoshua Bengio. Generative adversarial nets. In Zoubin Ghahramani, Max Welling, Corinna Cortes, Neil D. Lawrence, and Kilian Q. Weinberger, editors, *Advances in Neural Information Processing Systems 27: Annual Conference on Neural Information Processing Systems 2014, December 8-13 2014, Montreal, Quebec, Canada*, pages 2672–2680, 2014.

[65] Redes neuronales recurrentes. `https://torres.ai/redes-neuronales-recurrentes/`. Último acceso: junio 2022.

[66] Alex Graves, Abdel-rahman Mohamed, and Geoffrey E. Hinton. Speech recognition with deep recurrent neural networks. In *IEEE International*

Conference on Acoustics, Speech and Signal Processing, ICASSP 2013, Vancouver, BC, Canada, May 26-31, 2013, pages 6645–6649. IEEE, 2013.

[67] ¿cómo aprenden las redes neuronales? `https://blogs.nvidia.com/blog/2022/03/25/what-is-a-transformer-model/`. Último acceso: junio 2022.

[68] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. In Isabelle Guyon, Ulrike von Luxburg, Samy Bengio, Hanna M. Wallach, Rob Fergus, S. V. N. Vishwanathan, and Roman Garnett, editors, *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, pages 5998–6008, 2017.

[69] Neural networks. `https://ml4a.github.io/ml4a/neural_networks/`. Último acceso: junio 2022.

[70] Qué son las redes neuronales y sus funciones. `https://www.atriainnovation.com/que-son-las-redes-neuronales-y-sus-funciones/`. Último acceso: junio 2022.

[71] Wikipedia. Motivo (música) — Wikipedia, the free encyclopedia. `http://es.wikipedia.org/w/index.php?title=Motivo%20(m%C3%BAsica)&oldid=130921777`, 2022. [Último acceso: junio 2022].

[72] Terms that describe texture. `https://courses.lumenlearning.com/musicappreciation_with_theory/chapter/monophony/`. Último acceso: junio 2022.

[73] Musica fuerte, auriculares, ipods - constante ruido en sus oÍdos. `https://www.widex.cl/es-cl/sobre-perdida-auditiva/audifonos-musica-y-ruido`. Último acceso: junio 2022.

[74] Christoph Wolff. *Johann Sebastian Bach: the learned musician*. WW Norton & Company, 2001.

[75] Jean-Pierre Briot, Gaëtan Hadjeres, and François-David Pachet. *Deep Learning Techniques for Music Generation*. Springer, 2020.

[76] Cheng-Zhi Anna Huang, Ashish Vaswani, Jakob Uszkoreit, Noam Shazeer, Ian Simon, Curtis Hawthorne, Andrew M. Dai, Matthew D. Hoffman, Monica Dinculescu, and Douglas Eck. Music transformer, 2018.

[77] Walter Piston. Harmony. Technical report, 1941.

[78] David Heckerman, David Maxwell Chickering, Christopher Meek, Robert Rounthwaite, and Carl Myers Kadie. Dependency networks for inference, collaborative filtering, and data visualization. *J. Mach. Learn. Res.*, 1:49–75, 2000.

[79] Olof Mogren. C-rnn-gan: Continuous recurrent neural networks with adversarial training, 2016.

[80] Ching-Hua Chuan and Dorien Herremans. Modeling temporal tonal relations in polyphonic music through deep networks with a novel image-based representation. In Sheila A. McIlraith and Kilian Q. Weinberger, editors, *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, (AAAI-18), the 30th innovative Applications of Artificial Intelligence (IAAI-18), and the 8th AAAI Symposium on Educational Advances in Artificial Intelligence (EAAI-18), New Orleans, Louisiana, USA, February 2-7, 2018*, pages 2159–2166. AAAI Press, 2018.

[81] `https://sitiobigdata.com/2019/12/24/funciones-comunes-de-perdida-en-el-aprendizaje-automatico/`. Último acceso: agosto 2022.

Lista de Figuras

Lista de Tablas

Anexos

Anexos A

Conceptos musicales básicos

Para poder comprender este trabajo es necesario tener claros ciertos conceptos musicales, ya que sin ellos será complicado entender el nexo de unión entre lo que queremos obtener y el métedo empleado para dicho fin.

Para comenzar a adentrarse en el mundo de la música lo más importante es comprender qué es una nota y los tipos de notación existentes para las mismas.

A.1. Notas y sus nomenclaturas

«*Se denomina nota musical al sonido que se produce a través de una vibración cuya frecuencia es constante. Puede decirse, por lo tanto, que una nota es un sonido con una cierta frecuencia.*[1]» Para diferenciar cada nota de las demás, necesitamos darles un nombre específico, para lo cuál se diferencia entre 7 tipos de notas en una escala diatónica (este término lo explicaremos más adelante), pero el nombre difiere entre dos tipos de notaciones, la notación anglosajona (A.2) y la latina (A.1).

$$Latina = \{Do, Re, Mi, Fa, Sol, La, Si\} \tag{A.1}$$

$$Anglosajona = \{C, D, E, F, G, A, B\} \tag{A.2}$$

Para entender la función de introducir una nomenclatura para diferenciar las notas, es primordial explicar el concepto pentagrama, ya que dependiendo del lugar que ocupa cada nota en el mismo, varía su nombre.

» [1] `https://definicion.de/nota-musical/` último acceso: agosto 2022

En cuanto a la duración de las notas también existen diferentes figuras, dependiendo de si su figura en un circulo negro, poseen puntos de anotación o si las notas hacen referencia a silencios, existen numerosas notas. En la figura A.1[2], podemos encontrar las distintas figuras (desde la redonda hasta le semifusa) relevantes para el trabajo.

NOMBRE	FIGURA	SILENCIO	VALOR
Redonda			4 Tiempos
Blanca			2 Tiempos
Negra			1 Tiempo
Corchea			1/2 Tiempo
Semicorchea			1/4 Tiempo
Fusa			1/8 Tiempo
Semifusa			1/16 Tiempo

Figura A.1: Representación de las diferentes figuras musicales.

A.1.1. Pentagrama

«*Se conoce como pentagrama a una modalidad de notación musical que se basa en una estructura compuesta por cinco rectas ubicadas de manera paralela y a una misma distancia de separación.*[3]»

Así pues, cada una de las líneas de dicho pentagrama supondrá una nota distinta (como podemos observar en la figura A.3[4]). Pero estas notas también variarán en función de la clave en la que estén escritas, que pueden ser: la clave de sol, la clave de

[2] `https://badpuarevistamusical.wordpress.com/2018/07/10/figuras-musicales/` último acceso: agosto 2022

» [3] `https://definicion.de/?s=pentagrama` último acceso: agosto 2022

[4] `https://mcarmenfer.wordpress.com/2011/02/21/acordes-nomenclatura-anglosajona/` último acceso: agosto 2022

fa y la clave de do (sus representaciones se pueden observar en la figura A.2[5]).

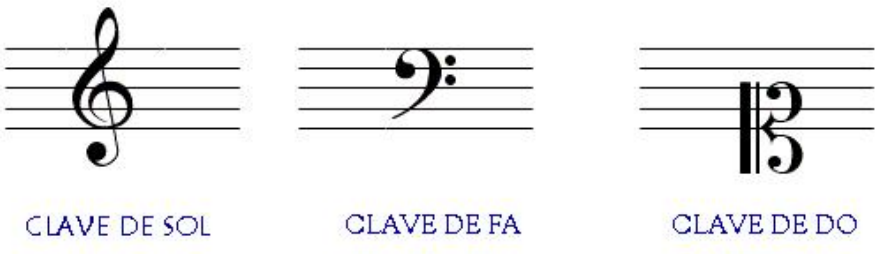

Figura A.2: Representación de las diferentes claves, do, sol y fa.

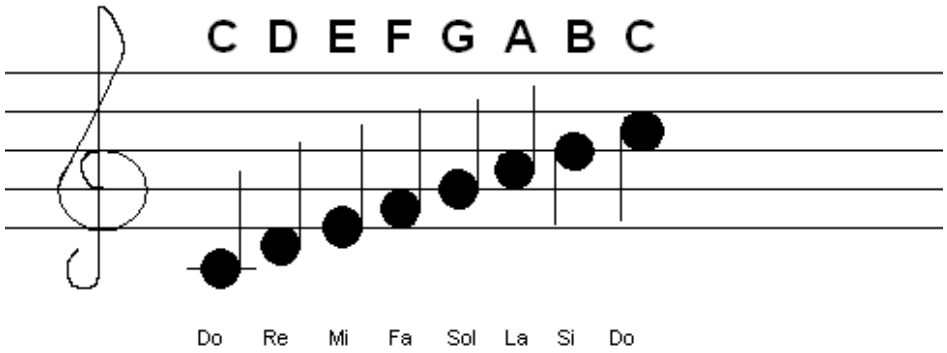

Figura A.3: Representación de las nomenclaturas latina y anglosajona en un pentagrama en clave de sol.

Una vez definidas las 7 notas pertenecientes a una escala diatónica toca definir el concepto de *escala*: «*Se conoce como escala musical a los sonidos consecutivos que se suceden de forma regular, ya sea en sentido ascendente o descendente, y que están relacionados a un único tono que es el que da nombre a la escala*[6]» Así pues, una escala diatónica será aquella que carezca de tonalidad, es decir, sus notas no sufrirán variaciones de semitonos.

Las modificaciones en semitono de las notas son de tres tipos: sostenido #, bemol ♭ y becuadro ♮, omo podemos observar en la figura A.4[7].

[5] `https://www.unprofesor.com/musica/que-son-las-claves-musicales-3481.html` último acceso: agosto 2022

» [6] `https://definicion.de/escala/` último acceso: agosto 2022

[7] `https://www.youtube.com/watch?v=JGWsAth5gFk` último acceso: agosto 2022

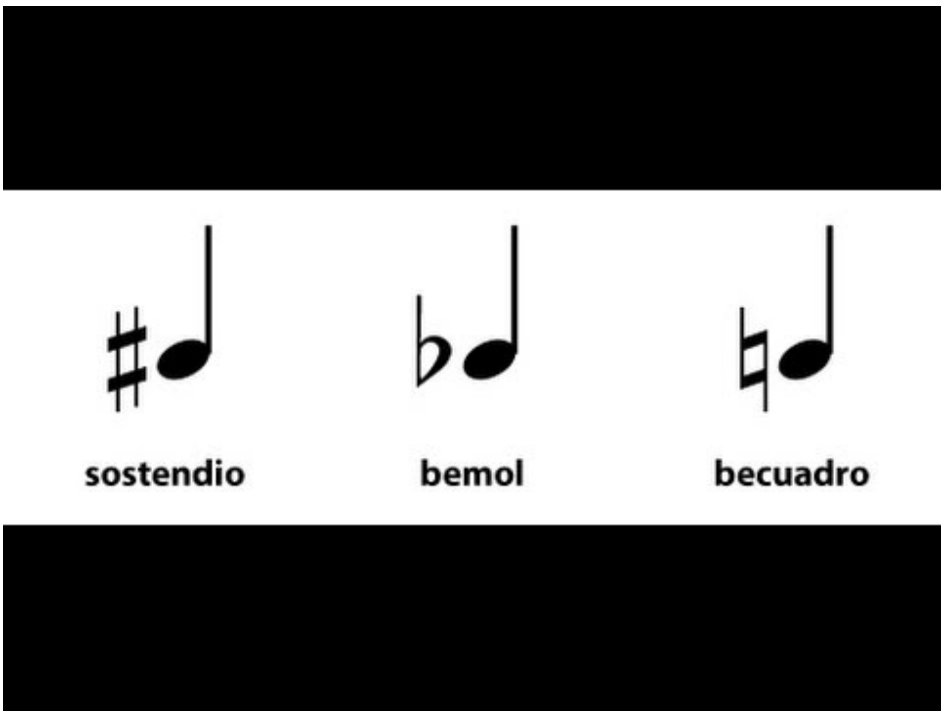

Figura A.4: Representación de las diferentes modificaciones de semitono: sostenido, bemol y becuadro.

Así pues, una vez definidas las variaciones de semitono de las distintas notas, podemos proceder a definir el concepto de tonalidad.

A.1.2. Tonalidad

La tonalidad viene definida en cada partitura, y hace referencia a la cantidad de notas que están alteradas (ya sea mediante bemoles como por sostenidos o becuadros). En la figura A.5[8] podemos observar las diferentes tonalidades en función de los sostenidos o bemoles incluidos en la armadura (parte de la partitura en dónde se define la tonalidad).

[8] `https://www.unprofesor.com/musica/las-tonalidades-musicales-definicion-y-caracteristicas-3525.html` último acceso: agosto 2022

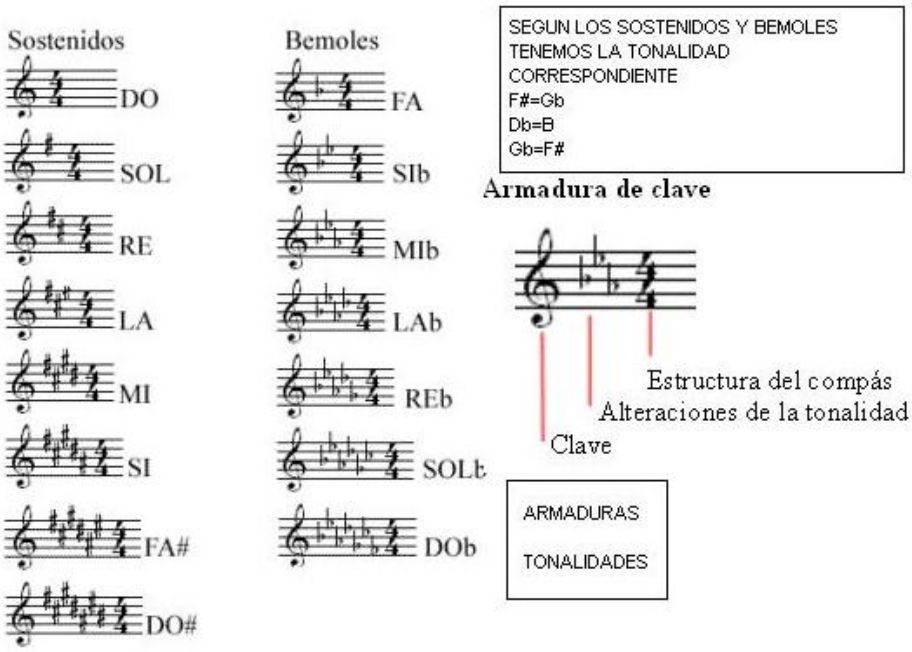

Figura A.5: Representación de las diferentes tonalidades.

A.2. Acorde

Una vez conocida la tonalidad en la que se reproducen los sonidos musicales, es necesario saber la disposición con la que se tocan las notas, ya que éstas, para reproducir un sonido acorde para el oído del oyente, deben de organizarse, y esa precisamente es la función de los acordes.

«*Un acorde consiste en un conjunto de dos o más notas diferentes y que constituyen una unidad armónica.*[9]»

Ésta definición da lugar a una gran cantidad de diferentes acordes, pero los más importantes los dividiremos de momento en acordes mayores y acordes menores.

A.2.1. Terceras, quintas y séptimas

Pero antes de proceder a la explicación de los diferentes acordes debemos aclarar que son las terceras, las quintas y las séptimas:

- «*Se denomina tercera al intervalo de tres grados entre dos notas de la escala musical* [10]».

- «*Se denomina quinta al intervalo de cinco grados entre dos notas de la escala musical* [11]».

[9] `https://es.wikipedia.org/wiki/Acorde` último acceso: agosto 2022
[10] `https://es.wikipedia.org/wiki/Tercera` último acceso: agosto 2022
[11] `https://es.wikipedia.org/wiki/Quinta` último acceso: agosto 2022

– «*Se denomina séptima al intervalo de siete grados entre dos notas de la escala musical* [12]».

En las figuras A.6[13], A.7[14] y A.8[15] podemos observar las representaciones de los intervalos de tercera , quinta y séptimas en un pentagrama.

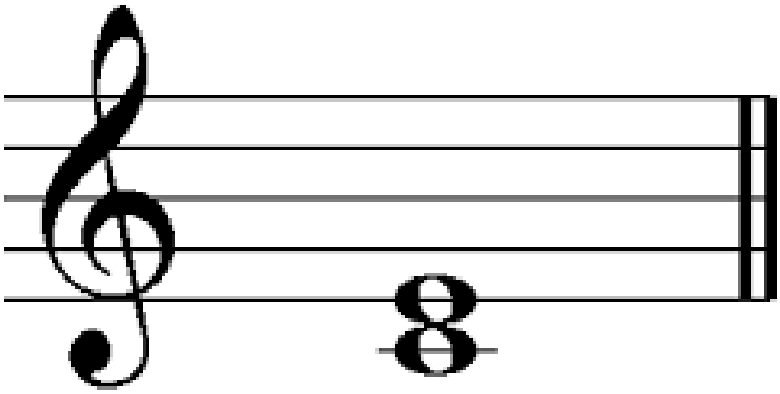

Figura A.6: Representación de un intervalo de tercera.

» [12]https://es.wikipedia.org/wiki/S\unhbox\voidb@x\bgroup\let\unhbox\voidb@x\setbox\@tempboxa\hbox{e\global\mathchardef\accent@spacefactor\spacefactor}\let\begingroup\let\typeout\protect\begingroup\def\MessageBreak{Ω(Font)}\let\protect\immediate\write\m@ne{LaTeXFontInfo:oninputline94.}\endgroup\endgroup\relax\let\ignorespaces\relax\accent19e\egroup\spacefactor\accent@spacefactor\futurelet\@let@token\protect\penalty\@M\hskip\z@skipptima último acceso: agosto 2022

[13]https://es.wikipedia.org/wiki/Tercera último acceso: agosto 2022

[14]https://es.wikipedia.org/wiki/Quinta último acceso: agosto 2022

[15]https://es.wikipedia.org/wiki/S\unhbox\voidb@x\bgroup\let\unhbox\voidb@x\setbox\@tempboxa\hbox{e\global\mathchardef\accent@spacefactor\spacefactor}\let\begingroup\let\typeout\protect\begingroup\def\MessageBreak{Ω(Font)}\let\protect\immediate\write\m@ne{LaTeXFontInfo:oninputline96.}\endgroup\endgroup\relax\let\ignorespaces\relax\accent19e\egroup\spacefactor\accent@spacefactor\futurelet\@let@token\protect\penalty\@M\hskip\z@skipptima último acceso: agosto 2022

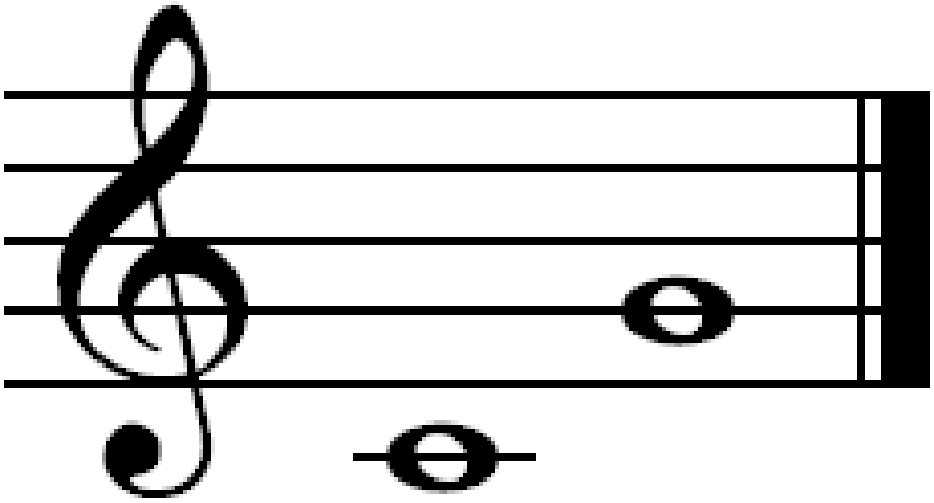

Figura A.7: Representación de un intervalo de quinta.

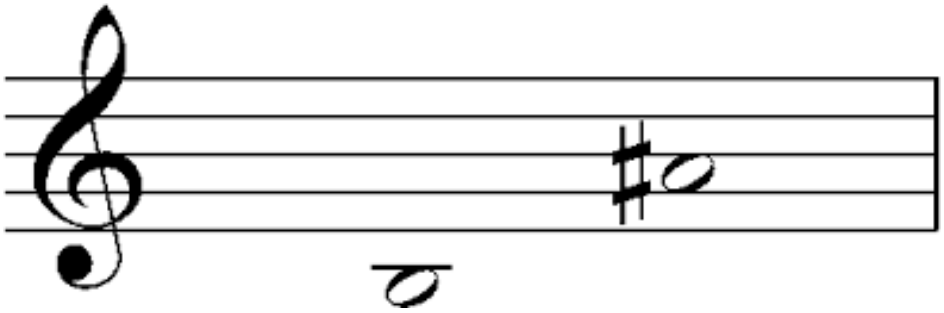

Figura A.8: Representación de un intervalo de séptima.

Una vez definidos los distintos intervalos principales presentes en los acordes, podemos proceder a analizar los distintos tipos de acordes (aquellos que con más frecuencia aparecen en las obras musicales).

Como ya habíamos comentado, estos acordes se dividen fundamentalmente en acordes mayores y menores. Por lo general, los acordes suelen ser triadas, es decir, tres notas separadas entre sí por 2 tonos o tono y medio.

- Los acordes mayores mayores se diferencian por su tercera principal, es decir, la diferencia de tonos entre su nota fundamental (o su nota primera del acorde) y su tercera, la cuál debe ser mayor (la distancia en tonos de la fundamental a su tercera deben de ser 2 tonos).

- Los acordes menores también se diferencian por su tercera principal, la cuál debe de ser de tono y medio (medio tono menor al acorde mayor).

En la figura A.9[16], podemos observar distintos acordes, tanto mayores como menores.

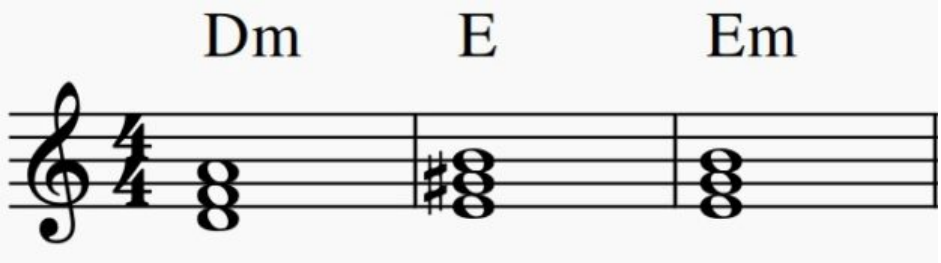

Figura A.9: Representación de tres acordes: Re menor, Mi mayor y Mi menor.

A.2.2. Otros tipos de acordes

Los acordes principales son los mayores y menores, pero la cantidad de acordes que se emplean en la creación musical son infinitos. En este apartado analizaremos los acordes disminuidos, aumentados y con séptima.

- Los acordes aumentados son aquellos a los que siendo un acorde mayor, se les aumenta en medio tono la quinta (quedando así una quinta aumentada), en la figura A.10[17] podemos observar un ejemplo.

- Los acordes disminuidos, por el contrario, so aquellos que siendo menores, se les disminuye medio tono la quinta (quedando así una quinta disminuida), en la figura A.11[18] podemos observar un ejemplo.

- Los acordes con séptima son aquellos acordes que indiferentemente de ser mayores o menores, poseen 4 notas, siendo la cuarta una tercera con respecto de la tercera nota (previamente denominada quinta), quedando así una séptima con respecto de la nota fundamental, en la figura A.12[19] podemos observar un ejemplo.

[16] `https://cresciente.net/teoria-musical-basica/que-es-un-acorde/` último acceso: agosto 2022

[17] `https://www.unprofesor.com/musica/que-son-las-claves-musicales-3481.html` último acceso:agosto 2022

[18] `https://comamusical.com/acordes-disminuidos/` último acceso:agosto 2022

[19] `https://es.wikipedia.org/wiki/Acorde_de_s%C3%A9ptima` último acceso:agosto 2022

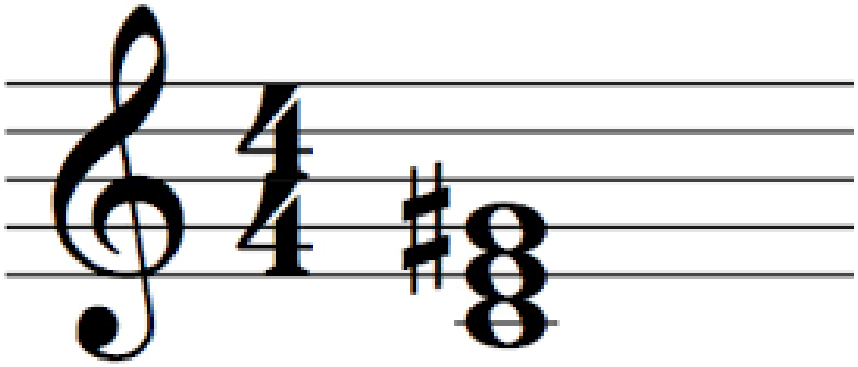

Figura A.10: Representación de un acorde de do mayor aumentado

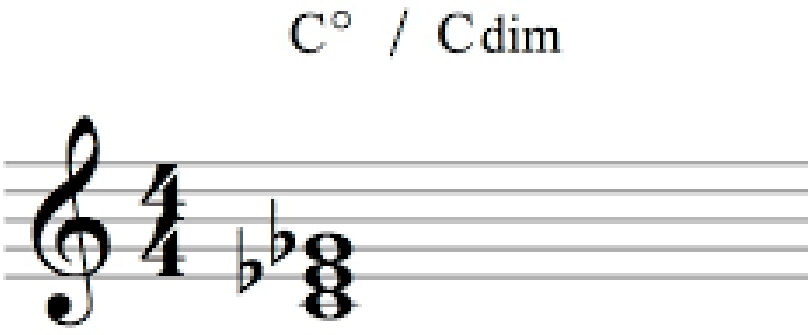

Figura A.11: Representación de un acorde de re menor disminuido

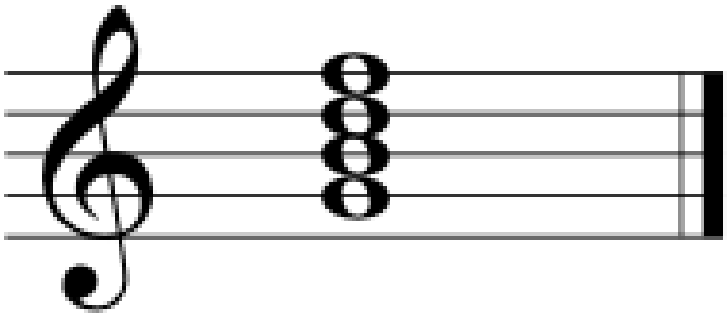

Figura A.12: Representación de un acorde de sol con séptima dominante

Anexos B

Arquitecturas de Redes Neuronales (RRNN) en Deep Learning

B.1. Introducción a las Redes Neuronales

Una red neuronal viene definida por neuronas, cuyo funcionamiento[1] se basa en recibir unos valores numéricos de entrada, combinarlos, y devolver un valor numérico de salida. Estas neuronas forman a su vez capas (o niveles), los cuáles a su vez forman redes que unidas a un algoritmo, una entrada y una salida, forman un sistema neuronal (en la figura B.1, podemos observar representados cada uno de los componentes nombrados).

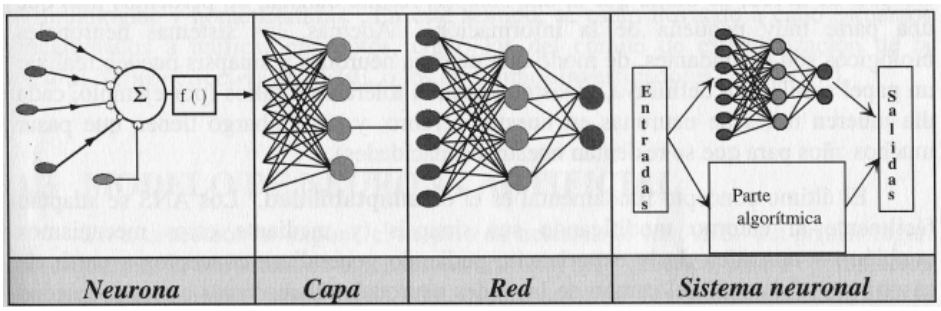

Figura B.1: Representación de las diferentes partes que definen un sistema neuronal

Para poder entender el funcionamiento de un sistema neuronal es indispensable definir la función de activación: función que transmite la información generada por la combinación lineal de los pesos y las entradas, es decir son la manera de transmitir la información por las conexiones de salida[2]. Éstas funciones de activación pueden ser de distintos tipos, como observamos en la figura B.2[3]

[1] `https://www.futurespace.es/redes-neuronales-y-deep-learning-capitulo-2-la-neurona/` último acceso: agosto 2022

[2] `https://aiofthings.telefonicatech.com/recursos/datapedia/funcion-activacion` último acceso: agosto 2022

[3] `https://www.futurespace.es/redes-neuronales-y-deep-learning-capitulo-2-la-neurona/` último acceso: agosto 2022

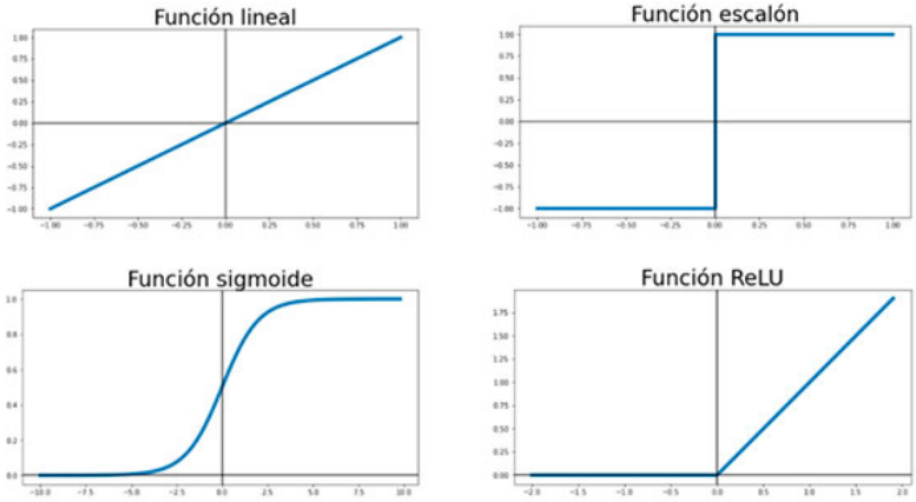

Figura B.2: Representación de las diferentes funciones de activación

Una vez analizadas las funciones de activación, queda analizar las funciones de pérdidas de la red.

Una función de pérdida es una función que evalúa la desviación entre las predicciones realizadas por la red neuronal y los valores reales de las observaciones utilizadas durante el aprendizaje[4]. Las diferentes funciones de pérdidas se pueden observar en la figura B.3[5]

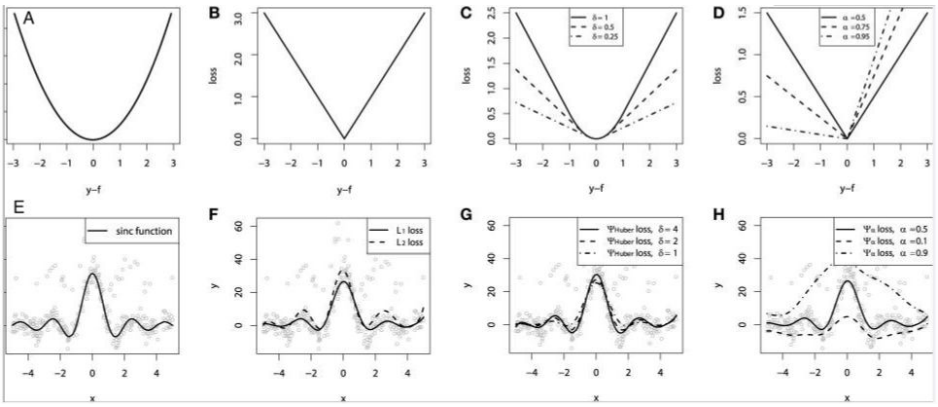

Figura B.3: Representación de las diferentes funciones de pérdidas

B.2. Tipos de redes neuronales

Una vez introducido el funcionamiento de una red neuronal, será importante definir los tipos de redes que existen. Las redes pueden ser diferenciadas por varias clases: según el número de capas, según el tipo de conexiones y según el grado de conexiones.

[4] `https://www.ediciones-eni.com/open/mediabook.aspx?idR=8dd2ca32769cb24b49648b15ef8e777e` último acceso: agosto 2022

[5] `https://programmerclick.com/article/23511327338/` último acceso: agosto 2022

B.2.1. Redes neuronales según el número de capas

Teniendo en cuenta el número de capas presentes en una red neuronal, éstas pueden ser separadas en 2 grandes grupos: redes monocapas (dónde la capa de entrada se conecta directamente con la salida)[6] y redes multicapa, o *perceptrón multicapa*[7]. En la figura B.4 podemos observar una representación de una red neuronal multicapa.

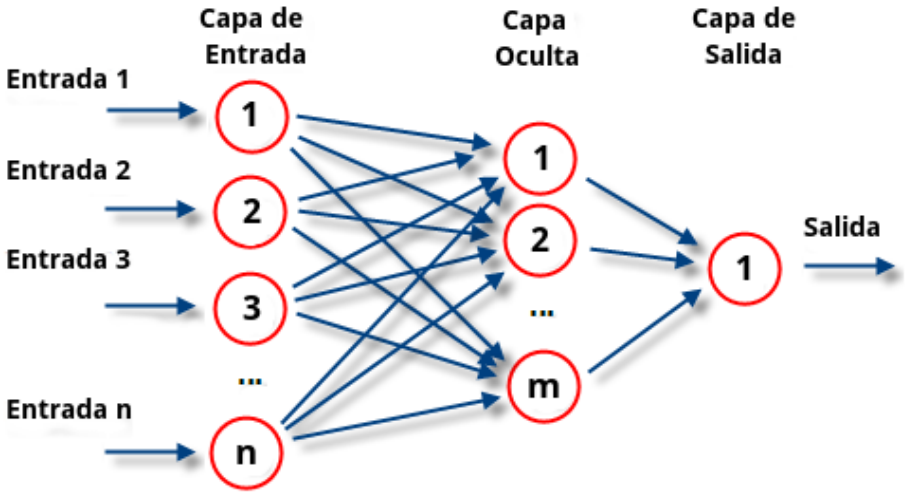

Figura B.4: Representación de las múltiples capas en una red neuronal multicapa

B.2.2. Redes neuronales según el tipo de conexiones

Según su conexión, las redes neuronales se dividen en redes neuronales no recurrentes y redes neuronales recurrentes (RNN).

- Las redes neuronales no recurrentes[8] carecen de memoria, es decir, no almacenan la información entre capas, lo que las hace menos utilizadas.

- Las redes neuronales recurrentes, al contrario que las no recurrentes, sí que almacenan la información obtenida en cada una de sus capas para la capa siguiente, esto hace de estas redes, las más utilizadas. En la figura B.5[9] podemos observar una representación del funcionamiento de una RNN.

[6] `https://www.threepoints.com/blog/que-son-las-redes-neuronales` último acceso: agosto 2022

[7] `https://es.wikipedia.org/wiki/Perceptr%C3%B3n_multicapa` último acceso: agosto 2022

[8] `https://www.threepoints.com/blog/que-son-las-redes-neuronales` último acceso: agosto 2022

[9] `https://torres.ai/redes-neuronales-recurrentes/` último acceso: agosto 2022

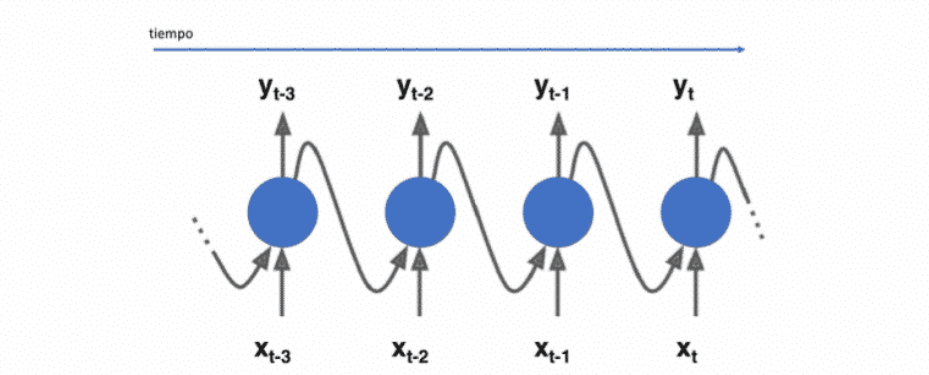

Figura B.5: Representación del funcionamiento de una RNN

B.2.3. Redes neuronales según el grado de conexiones

Otro de los factores a tener en cuenta en la clasificación de redes neuronales es el grado de sus conexiones, ya que dependiendo de si éstas están totalmente o parcialmente conectadas, la red se parecerá más o menos a un cerebro humano.

- En las redes totalmente conectadas, la totalidad de las neuronas están conectadas entre ellas, lo que hará que su comportamiento se asemeje más al del cerebro humano.

- En las redes parcialmente conectadas, únicamente una parte de las neuronas estarán conectadas entre sí, lo cuál hace que se asemeje menos a un cerebro humano que una red totalmente conectada.

Una vez definidos los tipos de redes neuronales, pasamos a explicar unas de las redes neuronales más utilizada e importantes, las redes neuronales convolucionales.

B.2.4. Redes neuronales convolucionales

Las Redes neuronales convolucionales son un tipo de redes neuronales artificiales donde las «neuronas» corresponden a campos receptivos de una manera muy similar a las neuronas en la corteza visual primaria (V1) de un cerebro biológico. Este tipo de red es una variación de un perceptrón multicapa, sin embargo, debido a que su aplicación es realizada en matrices bidimensionales, son muy efectivas para tareas de visión artificial, como en la clasificación y segmentación de imágenes, entre otras aplicaciones[10]. En la figura B.6[11] podemos observar una representación del funcionamiento de una red neuronal convolucional.

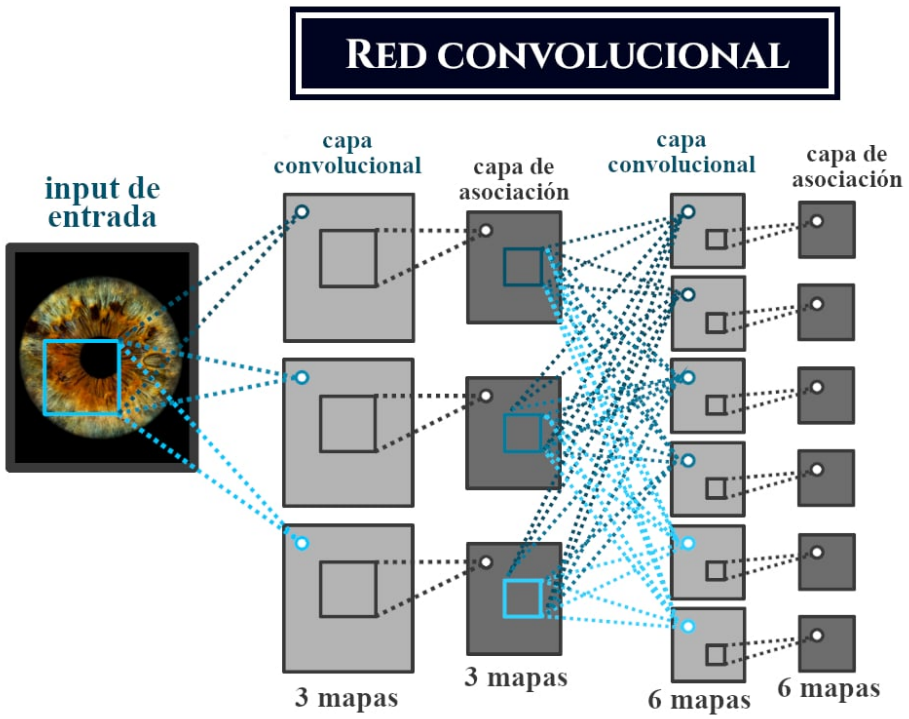

Figura B.6: Representación del funcionamiento de una red neuronal convolucional

[10] https://www.juanbarrios.com/redes-neurales-convolucionales/#Definiciones último acceso: agosto 2022

[11] https://lamaquinaoraculo.com/computacion/redes-neuronales-convolucionales/ último acceso: agosto 2022