
UNIVERSIDAD AUTÓNOMA DE GUADALAJARA

CON RECONOCIMIENTO DE VALIDEZ OFICIAL DE ESTUDIOS DE LA SECRETARÍA DE
EDUCACIÓN PÚBLICA SEGÚN ACUERDO No.158 DE FECHA 17 DE JULIO DE 1991

POSTGRADO E INVESTIGACIÓN



Uso de Deep Learning para composición musical

TRABAJO DE INVESTIGACIÓN

QUE PRESENTA

EFRAIN ADRIAN LUNA NEVAREZ

PARA OBTENER EL GRADO DE

MAESTRÍA EN CIENCIAS COMPUTACIONALES

DIRECTOR Y ASESOR:

MTRO. JUAN ANTONIO VEGA FERNANDEZ

GUADALAJARA, JALISCO. AGOSTO 2018

Número de registro en el sistema de investigación

Dedicatoria

Este trabajo se lo dedico a todos los músicos que sean apasionados por la tecnología, y ven en ella una forma para seguir evolucionando musicalmente.

Agradecimientos

Primeramente agradezco a Dios por permitirme haber concluido un ciclo mas en mi formación profesional, cada paso que doy siempre es gracias a él.

Agradezco a mi madre cuya fuerza de voluntad fue capaz de sacar adelante a una familia completa, ella me enseñó el valor del trabajo y del estudio. Gracias a ella que me brindo un consejo cuando lo necesite, un hombro de apoyo cuando flaqueaba y una reprimenda cuando no estaba asciendo las cosas correctamente. Gracias a su amor incondicional soy quien soy ahora.

A mi novia que estuvo conmigo en épocas difíciles en mi vida, cuando pensé que no podría terminar con mis estudios por todas las responsabilidades que había adquirido, ella siempre estuvo ahí para animarme a seguir adelante.

Mis hermanas las cuales siempre me dieron palabras de aliento y siempre estuvieron preocupadas por mi, en todo momento, agradezco su apoyo incondicional.

Al CONACYT porque me brindo el apoyo económico para poder terminar este grado de estudio.

A mis maestros que tuvieron la suficiente paciencia y dedicación para transmitirnos parte de su conocimiento, gracias a ellos pude concluir con este grado, ganando el suficiente conocimiento para poder aplicarlo en el campo laboral.

A mi Asesor el Maestro Juan Antonio Vega Fernández, el cual me guió a lo largo de esta tesis para realizar un trabajo que pueda dejar algo de conocimiento a la comunidad, agradezco la paciencia que me tuvo y el tiempo que me dedico para la revisión de este proyecto.

A mis compañeros de generación los cuales me enseñaron muchísimas cosas, fue muy enriquecedor contar con personas que a pesar de que habían vivido cosas muy diferentes, tenían en la mente seguir superándose escolarmente y profesionalmente hablando.

Finalmente agradezco a la UAG, escuela que me dejo tanto aprendizaje y vivencias. A todo el personal administrativo de esta institución que me guiaron a travez de todos los tramites necesarios para concluir esta etapa.

Resumen

En este trabajo, se analizaron 3 diferentes tipos de arquitecturas de redes de Deep Learning, para que sirvieran de apoyo a la hora de crear composiciones musicales.

Estas redes usaron capas de neuronas LSTM, las cuales nos permitieron trabajar con secuencias de datos.

Las redes descritas en este proyecto fueron entrenadas para reconocer y generar secuencias para los instrumentos de guitarra y bajo eléctrico.

El proceso de entrenamiento se logro usando una base de datos de archivos MIDI, los cuales son de genero rock y pop, por lo tanto, el aprendizaje de estas redes únicamente sera para reconocer estos géneros musicales.

Las arquitecturas usadas fueron entrenadas para reconocer tonalidades de las canciones y en base a ellas generar nuevas piezas musicales.

Con el fin de validar que las composiciones generadas tuvieran la tonalidad deseada se utilizo el algoritmo de Krumhansl-Schmuckler.

Tabla de Contenido

| | |
|--|-----------|
| Lista de Figuras | IX |
| Lista de Tablas | XI |
| 1. Introducción | 1 |
| 1.1. Descripción del Problema | 1 |
| 1.2. Objetivos | 1 |
| 1.2.1. Objetivo General | 1 |
| 1.2.2. Objetivos Específicos | 2 |
| 1.3. Justificación | 2 |
| 1.4. Delimitación | 2 |
| 1.5. Organización de la Tesis | 3 |
| 2. Inteligencia Artificial | 5 |
| 2.1. Redes neuronales artificiales | 7 |
| 2.1.1. Aprendizaje de las redes neuronales | 9 |
| 2.1.2. Deep Learning | 11 |
| 2.1.3. Redes neuronales recurrentes | 11 |
| 2.1.3.1. LSTM | 13 |
| 2.1.4. Optimizadores | 14 |
| 3. Teoría musical | 15 |
| 3.1. Composición musical | 16 |
| 3.1.1. Notas musicales | 16 |
| 3.1.2. Claves musicales | 18 |
| 3.1.3. Compases y tiempo | 19 |

| | | |
|-----------|--|-----------|
| 3.1.4. | Escalas | 21 |
| 3.1.5. | Armaduras | 22 |
| 3.1.6. | Tonalidades | 23 |
| 3.1.7. | Melodías y Armonías | 24 |
| 3.2. | Formato MIDI | 24 |
| 3.3. | Algoritmo Krumhansl-Schmuckler | 27 |
| 4. | Desarrollo | 29 |
| 4.1. | Datos | 29 |
| 4.2. | Arquitectura de la aplicación | 30 |
| 4.2.1. | UI | 31 |
| 4.2.2. | Preprocesamiento | 32 |
| 4.2.3. | Red neuronal | 33 |
| 4.2.4. | Entrenamiento | 34 |
| 4.2.5. | Generación | 35 |
| 4.2.6. | Analizador | 37 |
| 5. | Resultados | 39 |
| 5.1. | Etapas de entrenamiento | 39 |
| 5.1.1. | Arquitectura 1 | 40 |
| 5.1.1.1. | Guitarra | 40 |
| 5.1.1.2. | Bajo | 42 |
| 5.1.2. | Arquitectura 2 | 44 |
| 5.1.2.1. | Guitarra | 44 |
| 5.1.2.2. | Bajo | 46 |
| 5.1.3. | Arquitectura 3 | 48 |
| 5.1.3.1. | Guitarra | 48 |
| 5.1.3.2. | Bajo | 50 |
| 5.1.4. | Comparativa de las arquitecturas | 52 |
| 5.1.4.1. | Guitarra | 52 |
| 5.1.4.2. | Bajo | 53 |
| 5.2. | Etapas de generación | 55 |
| 5.2.1. | Arquitectura 1 | 55 |
| 5.2.2. | Arquitectura 2 | 55 |
| 5.2.3. | Arquitectura 3 | 55 |

TABLA DE CONTENIDO

VII

5.2.4. Comparativa de las arquitecturas

55

5.3. Etapa de validación

55

5.3.1. Arquitectura 1

55

5.3.2. Arquitectura 2

57

5.3.3. Arquitectura 3

58

5.3.4. Comparativa de las arquitecturas

60

6. Conclusiones

63

6.1. Trabajos futuros

64

Bibliografía

67

Lista de Figuras

| | |
|--|----|
| 2.1. Ramas de la inteligencia artificial | 5 |
| 2.2. Red neuronal biológica | 7 |
| 2.3. Estructura general de una neurona | 8 |
| 2.4. Neurona Artificial | 8 |
| 3.1. Relacion de tonos y semitonos | 17 |
| 3.2. Pentagrama | 17 |
| 3.3. Clave de Sol en 2da. Linea | 18 |
| 3.4. Clave de Fa en 4ta. Linea | 18 |
| 3.5. Clave de Do en 4ta. y 3ra. linea | 19 |
| 3.6. Nombre de las notas en un pentagrama con clave de Sol | 19 |
| 3.7. Duración de las notas musicales | 20 |
| 3.8. Valor relativo de las notas | 20 |
| 3.9. Compas | 20 |
| 3.10. Tiempos fuertes y débiles | 21 |
| 3.11. Escala de Do Mayor | 21 |
| 3.12. Escala de Sol Mayor | 21 |
| 3.13. Escala de La menor | 22 |
| 3.14. Armaduras relativas con sostenidos | 23 |
| 3.15. Armaduras relativas con bemoles | 23 |
| 3.16. Melodía y Armonía | 24 |
| 3.17. Extracto de un formato MIDI | 26 |
| 3.18. Partitura correspondiente al extracto MIDI | 26 |
| 3.19. Instrumentos en formato MIDI | 27 |
| 4.1. Arquitectura de la aplicación | 30 |

| | |
|--|----|
| 4.2. Arquitectura de las redes usadas | 33 |
| 4.3. Flujo de entrenamiento | 35 |
| 4.4. Flujo de generación de notas | 36 |
| 5.1. Grafica de error para guitarra en arquitectura 1 | 40 |
| 5.2. Grafica de aprendizaje para guitarra en arquitectura 1 | 41 |
| 5.3. Grafica de error para bajo en arquitectura 1 | 42 |
| 5.4. Gráfica de aprendizaje para bajo en arquitectura 1 | 43 |
| 5.5. Grafica de error para guitarra en arquitectura 2 | 44 |
| 5.6. Gráfica de aprendizaje para guitarra en arquitectura 2 | 45 |
| 5.7. Grafica de error para bajo en arquitectura 2 | 46 |
| 5.8. Grafica de aprendizaje para bajo en arquitectura 2 | 47 |
| 5.9. Gráfica de error para guitarra en arquitectura 3 | 48 |
| 5.10. Grafica de aprendizaje para guitarra en arquitectura 3 | 49 |
| 5.11. Gráfica de error para bajo en arquitectura 3 | 50 |
| 5.12. Gráfica de aprendizaje para bajo en arquitectura 3 | 51 |
| 5.13. Gráfica de validación para guitarra en arquitectura 1 | 56 |
| 5.14. Gráfica de validación para bajo en arquitectura 1 | 56 |
| 5.15. Gráfica de validación para guitarra en arquitectura 2 | 57 |
| 5.16. Gráfica de validación para bajo en arquitectura 2 | 58 |
| 5.17. Gráfica de validación para guitarra en arquitectura 3 | 59 |
| 5.18. Gráfica de validación para bajo en arquitectura 3 | 59 |
| 5.19. Gráfica comparativa de arquitecturas para guitarra | 60 |
| 5.20. Gráfica comparativa de arquitecturas para bajo | 61 |

Lista de Tablas

| | |
|--|----|
| 4.1. Familias de guitarra y bajo. | 32 |
| 5.1. Valores de error para guitarra en arquitectura 1. | 40 |
| 5.2. Valores de aprendizaje para guitarra en arquitectura 1. | 41 |
| 5.3. Valores de error para bajo en arquitectura 1. | 42 |
| 5.4. Valores de aprendizaje para bajo en arquitectura 1. | 43 |
| 5.5. Valores de error para guitarra en arquitectura 2. | 44 |
| 5.6. Valores de aprendizaje para guitarra en arquitectura 2. | 45 |
| 5.7. Valores de error para bajo en arquitectura 2. | 46 |
| 5.8. Valores de aprendizaje para bajo en arquitectura 2. | 47 |
| 5.9. Valores de error para guitarra en arquitectura 3. | 48 |
| 5.10. Valores de aprendizaje para guitarra en arquitectura 3. | 49 |
| 5.11. Valores de error para bajo en arquitectura 3. | 50 |
| 5.12. Valores de aprendizaje para bajo en arquitectura 3. | 51 |
| 5.13. Comparativa del error en la fase de entrenamiento para guitarra . . . | 52 |
| 5.14. Comparativa del error en la fase de validación de entrenamiento para guitarra | 52 |
| 5.15. Comparativa del aprendizaje en la fase de entrenamiento para guitarra | 53 |
| 5.16. Comparativa del aprendizaje en la fase de validación de entrenamiento para guitarra | 53 |
| 5.17. Comparativa del error en la fase de entrenamiento para bajo | 53 |
| 5.18. Comparativa del error en la fase de validación de entrenamiento para bajo | 54 |
| 5.19. Comparativa del aprendizaje en la fase de entrenamiento para bajo . | 54 |
| 5.20. Comparativa del aprendizaje en la fase de validación de entrenamiento para bajo | 54 |

| | |
|--|----|
| 5.21. Valores de validación para la arquitectura 1. | 57 |
| 5.22. Valores de validación para la arquitectura 2. | 58 |
| 5.23. Valores de validación para la arquitectura 3. | 60 |
| 5.24. Comparativa de arquitecturas en la fase de validación para guitarra. . | 61 |
| 5.25. Comparativa de arquitecturas en la fase de validación para bajo. . . . | 62 |

Capítulo 1

Introducción

La música es el idioma universal, no importa en qué lugar físico nos encontremos, todos alguna vez hemos escuchado alguna canción que se nos queda grabada en la cabeza por mucho tiempo. En ocasiones nos preguntamos cómo es que el autor compuso esa canción.

Si bien es cierto el componer una canción es un arte, las canciones se pueden interpretar en forma matemática, esto nos abre la posibilidad de usar algoritmos para la creación de nuevas canciones.

1.1. Descripción del Problema

El tiempo requerido para la creación de una nueva canción es muy variado, ya que depende del género, el número de instrumentos y por supuesto la imaginación del compositor. Sin embargo si pudiéramos usar un algoritmo de aprendizaje en una computadora, esto nos permitiría generar nuevas canciones de una manera más sencilla y tomando mucho menos tiempo.

Existen compositores que a lo largo del tiempo no les gusta explorar nuevas formas o ritmos musicales y eso ocasiona que sus composiciones sean muy similares entre si.

1.2. Objetivos

1.2.1. Objetivo General

Creación de un programa que use algoritmos para servir de apoyo para composición musical.

1.2.2. Objetivos Específicos

- 1.- Implementar una red neuronal de Deep Learning que sea capaz de crear melodías y armonías musicales en guitarra.
- 2.- Implementar una red neuronal de Deep Learning que sea capaz de crear melodías musicales en bajo eléctrico.
- 3.- Conjuntrar la salida de las redes neuronales para generar un solo archivo de audio.
- 4.- Verificar la Tonalidad de la canción resultante.

1.3. Justificación

El uso de algoritmos de Deep Learning en la música no es algo nuevo, sin embargo, existen muchas áreas de oportunidad en cuestión de la implementación de estos algoritmos.

La mayoría de trabajos en esta área se basan en sonidos melódicos y no abarcan tan a fondo el campo armónico. Lo que se pretende en esta investigación es tener una manera fácil y rápida para la creación de nuevas composiciones musicales partiendo ya sea por melodías o armonías musicales de uno o varios instrumentos.

Un algoritmo eficaz para composición es aquel que es capaz de aprender a componer música partiendo de tres fundamentos musicales: la tonalidad, el ritmo y la cuadratura, el algoritmo que se pretende realizar en esta investigación evaluará estas variables y las tratará de generalizar para poder crear música a partir de una simple melodía o una armonía.

1.4. Delimitación

Este proyecto se centrará en crear melodías musicales de guitarra y bajo eléctrico por lo que no se analizarán otros instrumentos.

Las nuevas canciones creadas por estos algoritmos no buscan ser ideas finales de composición, la única intención es crear una serie de ideas para nuevas canciones.

La base de datos usada para entrenar a la red neuronal consta de canciones de género rock y pop unicamente, no se entrenara la red para reconocer y generar canciones de otros géneros musicales.

1.5. Organización de la Tesis

A continuación se muestra la descripción de cada capítulo:

- **Capítulo 1.-** En el primer capítulo se ve la descripción del problema, objetivos generales y justificación del proyecto.
- **Capítulo 2.-** Este capítulo contiene marco teórico del proyecto, en el cual se describen los conceptos generales usados en el proyecto.
- **Capítulo 3.-** En este capítulo se abordara el desarrollo del proyecto así como los resultados obtenidos.
- **Capítulo 4.-** En este último capítulo se ven las conclusiones del proyecto.

Inteligencia Artificial

En computación el término Inteligencia Artificial se define como la facultad de razonamiento de un programa informático, este agente racional tiene la capacidad de percibir su entorno y lleva a cabo acciones para maximizar el éxito en una tarea asignada.

Cuando una maquina es capaz de imitar las funciones cognitivas del ser humano para aprender y resolver problemas se puede decir que la maquina posee inteligencia artificial.

Dentro de la inteligencia artificial tenemos varias ramas que se enfocan en la resolución de problemas aplicando un principio muy específico de la inteligencia.

Las ramas de la inteligencia artificial se pueden dividir en áreas clásicas y áreas de vanguardia de acuerdo con la época cuando surgieron:

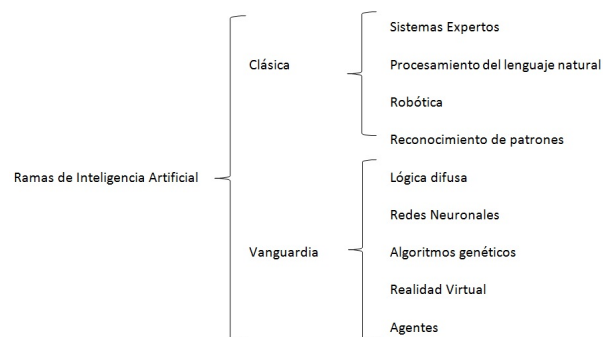


Figura 2.1: Ramas de la inteligencia artificial

Sistemas expertos.- son conocidos también como sistemas de conocimientos y estos programas informáticos aplican el proceso de razonamiento de un humano experto en la materia en la solución de problemas específicos. El modo de procesamiento

de estos sistemas es en base a una gran base de datos y utilizando una heurística avanzada para la determinación de las posibles soluciones a un problema.

Procesamiento del lenguaje natural.- estos son sistemas capaces de reconocer, procesar y en cierto punto emular la comunicación humana, estos sistemas buscan dejar el uso de lenguajes de programación o conjunto de comandos, para procesar el lenguaje humano natural. Para procesar el lenguaje es necesario dividirlo, primero se obtiene la comprensión del lenguaje natural, el cual investiga los métodos para que una computadora sea capaz de entender las instrucciones de este lenguaje, la segunda etapa consiste en la generación de lenguaje natural, aquí es donde la maquina intenta comunicarse en el lenguaje humano.

Robótica.- un robot es un dispositivo programado para realizar una tarea en específico. Se dice que un robot está adquiriendo inteligencia artificial si es capaz de responder a cambios en su entorno en lugar de seguir instrucciones programadas con anterioridad.

Reconocimiento de patrones.- esta es la parte de la inteligencia artificial encargada del procesamiento visual de un entorno, las imágenes son captadas por cámaras y posteriormente procesadas para el reconocimiento de patrones del entorno.

Lógica difusa.- es una forma matemática de representar el lenguaje natural, y el principio es generalizar la lógica clásica haciendo que las variables tomen valores lingüísticos de verdad.

Redes neuronales.- estas redes tratan de emular el comportamiento de las redes neuronales biológicas que poseen los humanos en su cerebro, partiendo de una neurona artificial y conectándola con otras para crear sinapsis entre ellas.

Algoritmos genéticos.- estos algoritmos son capaces de mutar para producir mejores respuestas a un entorno, estos sistemas tratan de imitar el proceso de selección natural en el cual al ir mutando algunos genes se van obteniendo sistemas más capaces para un entorno. Su función es seleccionar de una población de soluciones candidatas e intentar producir nuevas generaciones de soluciones las cuales se buscan que sean mejores que las anteriores.

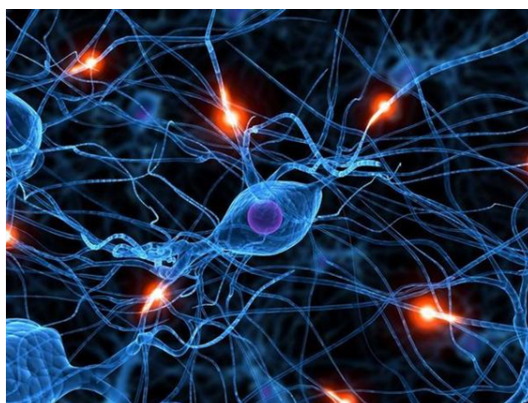
Realidad virtual.- es la recreación de un mundo artificial en tiempo real que puede ser captado por distintos canales sensoriales del espectador. Agentes.- estos son pequeños programas que actúan como espías observando las acciones comúnmente realizadas por el usuario, estas acciones son almacenadas y registradas, si en dado caso se llega a dar una anomalía el programa lanzara una alerta y dará una serie de

soluciones.

2.1. Redes neuronales artificiales

Las redes neuronales artificiales son un conjunto de neuronas creadas artificialmente para el desarrollo de inteligencia artificial en una computadora.

Estas redes están basadas en las redes biológicas del cerebro humano, modelando todos los factores biológicos de las neuronas.



"Jiménez, J., Pensamientos habituales y desarrollo personal. [Red neuronal]. Recuperado de <http://poderpersonalmexico.com/tag/redes-neuronales>"

Figura 2.2: Red neuronal biológica

Debido a su diseño las redes son capaces de aprender de la experiencia, de generalizar de casos anteriores a nuevos casos, de abstraer características esenciales a partir de entradas que representan en ocasiones información irreverente.

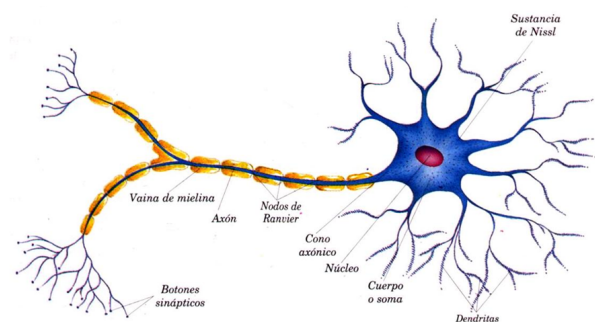
La capacidad de aprendizaje adaptativo es una característica fundamental de las redes neuronales y les permiten llevar a cabo ciertas tareas mediante un entrenamiento previo, pueden aprender a diferenciar patrones y generalizar a partir de estos. Son considerados sistemas dinámicos ya que son capaces de adaptarse a nuevas condiciones de entrada.

Tienen una alta tolerancia a fallos ya que son capaces de detectar patrones aun cuando estos patrones posean ruido, distorsión o simplemente están incompletos. Estos programas son capaces de seguir funcionando incluso si parte de la red presente fallas.

La información se almacena de forma distribuida en las conexiones de las neuronas, provocando redundancia de información, es decir se guardara sus valores en base a

la función de activación que posee cada neurona, de esta manera si una neurona es destruida o presenta fallas, las otras neuronas podrán aprender la información de la neurona que fallo.

Las neuronas humanas poseen diferentes secciones:



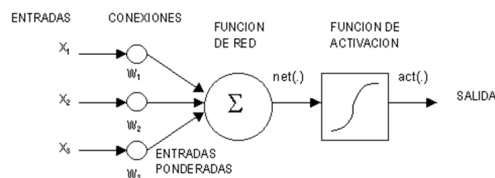
"Autor anónimo, Cuadros sinópticos y mapas conceptuales sobre las neuronas [Estructura general de una neurona]. Recuperado de <http://cuadrocomparativo.org/cuadros-sinopticos-sobre-las-neuronas>"

Figura 2.3: Estructura general de una neurona

En una neurona artificial se busca la emulación de las principales secciones de una neurona las cuales son:

- **Cuerpo.-** Se encarga de producir un impulso eléctrico en base a las entradas de la neurona.
- **Dendritas.-** son filamentos capaces de crear conexiones con otras neuronas.
- **Axón.-** Es el encargado de transmitir el impulso eléctrico generado por el cuerpo.

A continuación se muestra una imagen de como luciría una neurona artificial:



"Autor anónimo, Introducción a las redes neuronales artificiales [Neuronal artificial]. Recuperado de <http://www.rma.50webs.com/>"

Figura 2.4: Neurona Artificial

Esta neurona posee las siguientes secciones:

- **X1, X2, ..., Xn**.- Son las entradas de la neurona.
- **W1, W2, ..., Wn**.- Pesos específicos que tendrá cada entrada, esto hace que las entradas no valgan lo mismo ponderadamente.
- **Función de red**.- Esta es una función de sumatoria de las entradas.
- **Función de activación**.- Si la suma de las entradas es mayor o igual que el umbral definido por esta función se tendrá una señal a la salida.

La salida de la neurona viene dada por esta ecuación:

$$y_j = f\left(\sum_{i=1}^n (W_{ij}x_i + \theta_j)\right) \quad (2.1.1)$$

Una red neuronal no es más que la interconexión de varias neuronas artificiales, en la cual podemos identificar al menos tres secciones:

- **Capa de entrada**.- En esta capa se procesan todas las entradas, y si estas entradas son capaces de excitar las neuronas de esta capa se producirá una señal de salida.
- **Capa oculta**.- En esta capa se encuentran las neuronas encargadas del aprendizaje de la red.
- **Capa de salida**.- Esta neurona o neuronas de salida tendrán la salida del sistema.

La forma en que las redes aprenden es mediante la modificación de los pesos de las entradas.

Las redes neuronales artificiales han ido evolucionando con el paso del tiempo, hoy en día existen muchos modelos de redes neuronales, todas ellas con ventajas y desventajas si son comparadas entre ellas.

2.1.1. Aprendizaje de las redes neuronales

El procedimiento utilizado para llevar a cabo el proceso de aprendizaje en una red neuronal se denomina entrenamiento.

El problema de aprendizaje en las redes neuronales se formula en términos de la minimización de la función de error (o pérdida) asociada.

Normalmente, esta función está compuesta por dos términos, uno que evalúa cómo se ajusta la salida de la red neuronal al conjunto de datos de que disponemos, y que se denomina término de error, y otro que se denomina término de regularización, y que se utiliza para evitar el sobreaprendizaje por medio del control de la complejidad efectiva de la red neuronal.

Por supuesto, el valor de la función de error depende por completo de los parámetros de la red neuronal: los pesos sinápticos entre neuronas, y los bias asociados a ellas, que, como suele ser ya habitual, se pueden agrupar adecuadamente en un único vector de peso de la dimensión adecuada, que denotaremos por w . En este sentido, podemos escribir $f(w)$ para indicar que el valor del error que comete la red neuronal depende de los pesos asociados a la misma. Con esta formalización, nuestro objetivo es encontrar el valor w^* para el que se obtiene un mínimo global de la función f , convirtiendo el problema de aprendizaje en un problema de optimización.

En general, la función de error es una función no lineal, por lo que no disponemos de algoritmos sencillos y exactos para encontrar sus mínimos. En consecuencia, tendremos que hacer uso de una búsqueda a través del espacio de parámetros que, idealmente, se aproxime de forma iterada a un (error) mínimo de la red para los parámetros adecuados.

De esta forma, se comienza con una red neuronal con algún vector inicial de parámetros (a menudo elegido al azar), a continuación se genera un nuevo vector de parámetros, esperando que con ellos la función de error se reduzca (aunque dependiendo del método elegido, no es obligatorio, y temporalmente se puede admitir un empeoramiento del error siempre y cuando conduzca a una disminución posterior más acusada). Este proceso se repite, normalmente, hasta haber reducido el error bajo un umbral tolerable, o cuando se satisfaga una condición específica de parada.

El Descenso del Gradiente es el algoritmo de entrenamiento más simple y también el más extendido y conocido. Solo hace uso del vector gradiente, y por ello se dice que es un método de primer orden.

Este método para construir el punto w_{i+1} a partir de w_i se traslada este punto en la dirección de entrenamiento $d_i = -g_i$. Es decir:

$$w_{i+1} = w_i - g_i v_i \quad (2.1.2)$$

Donde el parámetro v se denomina tasa de entrenamiento, que puede fijarse a priori o calcularse mediante un proceso de optimización unidimensional a lo largo de la dirección de entrenamiento para cada uno de los pasos (aunque esta última opción es preferible, a menudo se usa un valor fijo, $v_i = v$ con el fin de simplificar el proceso).

Aunque es muy sencillo, este algoritmo tiene el gran inconveniente de que, para funciones de error con estructuras con valles largos y estrechos, requiere muchas iteraciones. Se debe a que, aunque la dirección elegida es en la que la función de error disminuye más rápidamente, esto no significa que necesariamente produzca la convergencia más rápida.

Por ello, es el algoritmo recomendado cuando tenemos redes neuronales muy grandes, con muchos miles de parámetros, ya que sólo almacena el vector gradiente (de tamaño n).

2.1.2. Deep Learning

Deep Learning usa redes neuronales con muchas capas para lograr aprendizajes más complejos. Este comportamiento asemeja la forma en que el cerebro humano toma decisiones, el cual usa la interconexión de varias capas de neuronas para realizar actividades complejas. Dentro de las redes de Deep Learning se tienen 2 tipos muy usados en la actualidad:

- **Redes convolucionales (CNN).**- Este tipo de redes usa la convolución en varias de sus capas para lograr el procesamiento de parámetros que se pueden representar en un espacio R^2 , un ejemplo claro de esto son las imágenes y vídeos, por lo tanto si se quiere hacer una clasificación o reconocimiento de imágenes, este tipo de redes nos proporcionan una buena herramienta de procesamiento.
- **Redes recurrentes (RNN).**- Este tipo de redes son muy usadas cuando se busca analizar una secuencia de datos, estas redes poseen memoria y una retroalimentación de la salida a la entrada.

2.1.3. Redes neuronales recurrentes

La idea detrás de las RNN es hacer uso de la información secuencial. En una red neuronal tradicional suponemos que todas las entradas (y salidas) son independientes entre sí. Pero para muchas tareas eso es una muy mala idea. Si quieres predecir la

siguiente palabra en una oración, es mejor que conozcas qué palabras vienen antes. Las RNN se llaman recurrentes porque realizan la misma tarea para cada elemento de una secuencia, y la salida depende de los cálculos previos. Otra forma de pensar acerca de las RNN es que tienen una "memoria" que captura información sobre lo que se ha calculado hasta ahora. En teoría, los RNN pueden hacer uso de la información en secuencias arbitrariamente largas, pero en la práctica se limitan a mirar hacia atrás solo unos pocos pasos.

La decisión de una red recurrente alcanzada en el paso de tiempo $t-1$ afecta la decisión que alcanzará un momento más tarde en el paso de tiempo t . Entonces, las redes recurrentes tienen dos fuentes de entrada, el presente y el pasado reciente, que se combinan para determinar cómo responden a los datos nuevos, de forma similar a como lo hacemos en la vida.

Esa información secuencial se conserva en el estado oculto de la red recurrente, que logra abarcar muchos pasos de tiempo a medida que avanza para afectar el procesamiento de cada nuevo ejemplo. Está encontrando correlaciones entre eventos separados por muchos momentos, y estas correlaciones se llaman "dependencias a largo plazo", porque un evento en el tiempo depende de, y es una función de, uno o más eventos que vinieron antes. Una forma de pensar acerca de las RNN es esta: son una forma de compartir pesos a lo largo del tiempo.

Así como la memoria humana circula invisiblemente dentro de un cuerpo, afectando nuestro comportamiento sin revelar su forma completa, la información circula en los estados ocultos de las redes recurrentes.

Describiremos el proceso de llevar la memoria hacia adelante matemáticamente:

$$h_t = \phi(Wx_t + Uh_{t-1}) \quad (2.1.3)$$

El estado oculto en el paso de tiempo t es h_t . Es una función de la entrada al mismo tiempo paso x_t , modificada por una matriz de ponderación W (como la que usamos para las redes feedforward) agregada al estado oculto del paso de tiempo anterior h_{t-1} multiplicado por su propio estado oculto matriz U de estado oculto, también conocida como matriz de transición y similar a una cadena de Markov. Las matrices de peso son filtros que determinan la importancia de acuerdo tanto con la entrada actual como con el estado oculto pasado. El error que generan volverá a través de la propagación inversa y se usará para ajustar sus ponderaciones hasta que el error no pueda bajar más.

Debido a que este ciclo de retroalimentación ocurre en cada paso de la serie, cada estado oculto contiene rastros no solo del estado oculto anterior, sino también de todos los que precedieron a h_{t-1} mientras la memoria pueda persistir.

2.1.3.1. LSTM

A mediados de los años 90, los investigadores alemanes Sepp Hochreiter y Juergen Schmidhuber propusieron una variación de la red recurrente con las denominadas unidades de memoria a largo plazo, o LSTM, como una solución al problema del gradiente de fuga.

Los LSTM ayudan a preservar el error que se puede volver a propagar a través del tiempo y las capas. Al mantener un error más constante, permiten que las redes recurrentes continúen aprendiendo durante muchos pasos de tiempo (más de 1000), abriendo así un canal para vincular causas y efectos de forma remota. Este es uno de los desafíos centrales para el aprendizaje automático y la IA, ya que los algoritmos se enfrentan con frecuencia a entornos en los que las señales de recompensa son dispersas y diferidas, como la vida misma.

Los LSTM contienen información fuera del flujo normal de la red recurrente en una celda cerrada. La información puede almacenarse, escribirse o leerse desde una celda, al igual que los datos en la memoria de una computadora. La célula toma decisiones sobre qué almacenar y cuándo permitir las lecturas, escrituras y borraduras, a través de puertas que se abren y cierran. Sin embargo, a diferencia del almacenamiento digital en computadoras, estas puertas son análogas, implementadas con la multiplicación de elementos por sigmoides, que están todas en el rango de 0-1. Siendo Analógica tiene la ventaja sobre digital de ser diferenciable y, por lo tanto, adecuado para la propagación inversa.

Esas puertas actúan sobre las señales que reciben, y de forma similar a los nodos de la red neuronal, bloquean o transmiten información en función de su fuerza e importación, que filtran con sus propios conjuntos de ponderaciones. Esos pesos, como los pesos que modulan los estados de entrada y ocultos, se ajustan a través del proceso de aprendizaje de redes recurrentes. Es decir, las células aprenden cuándo permiten que los datos entren, salgan o se eliminen a través del proceso iterativo de hacer conjeturas, volver a propagar el error y ajustar los pesos mediante el descenso del gradiente.

2.1.4. Optimizadores

Capítulo 3

Teoría musical

La teoría musical es el estudio de los elementos que conforman la música. En esta teoría se analizan todos los sonidos involucrados para la creación, análisis y composición musical.

La música es un arte que se basa en 2 elementos:

- Los sonidos
- Los silencios

Los sonidos tienen diferentes propiedades las cuales se describen a continuación:

- **Altura.-** un sonido puede ser agudo, medio o grave, dependiendo de la altura de su nota.
- **Duración.-** un sonido debe de tener una duración la cual se expresa en unidades de tiempo.
- **Intensidad.-** esto se refiere al volumen del sonido, puede ser débil o fuerte.
- **Timbre.-** se le conoce como timbre o color del sonido a como un sonido con la misma nota suena diferente dependiendo del instrumento usado para su interpretación.

Los silencios a su vez su única propiedad intrínseca es la duración, es decir en un silencio lo único que se mide es la duración del mismo.

3.1. Composición musical

La composición musical esta catalogado como un arte que tiene como objeto crear nuevas piezas musicales.

Un músico puede optar por varios caminos para la creación de su obra, existen músicos que se basan en su simple sentido común y crean canciones líricamente, sin embargo el proceso formal de composición involucra todos los conceptos musicales básicos de la teoría musical, los cuales se describirán a continuación.

3.1.1. Notas musicales

Las notas es un sistema que se usa para la representación de los diferentes sonidos en la música. En el mundo occidental se usa un sistema de 12 notas, los cuales pueden ser repetidos con diferentes alturas para generar una gama muy amplia de sonidos. Dentro de este sistema de 12 notas tenemos las notas naturales y las notas con alteraciones. Las notas naturales son:

Do, Re, Mi, Fa, Sol, La, Si (7 notas)

Las alteraciones no son más que agregar o quitar medios tonos a una nota, para eso se usan los siguientes símbolos:

- **# (sostenido).**- se agrega 1/2 tono a una nota.
- **x (doble sostenido).**- Se agrega 1 tono a una nota.
- **b (bemol).**- se disminuye 1/2 tono a una nota.
- **bb (doble bemol).**- Se disminuye 1 tono a una nota.

Usando los símbolos anteriores podemos definir las siguientes notas con alteraciones:

Do#, Re#, Fa#, Sol#, La# (5 notas)

Como se puede observar tanto Mi y Si no tienen sonidos con alteraciones ascendentes, ya que los sonidos producidos por estas alteraciones son igual al de las notas consecutivas, a este tipo de sonidos iguales se les conoce como notas enarmónicas.

Por ejemplo:

- Mi# sonaría exactamente igual que un Fa.

- Si \sharp sonaría exactamente igual que un Do.

También las notas con alteraciones pueden ser representadas usando el símbolo de bemol (b), es decir restando medio tono a una nota. En este caso se tendrían las notas con alteraciones de la siguiente manera:

Re \flat , Mi \flat , Sol \flat , La \flat , Si \flat (5 notas)

Como se mencionó anteriormente las notas que en sonido son exactamente iguales pero en nomenclatura son diferentes se conocen como notas enarmónicas, de tal manera que las 5 notas con alteraciones que se describieron se pueden hacer una comparación del sonido de las mismas, siendo así se tiene:

- Do \sharp suena exactamente igual que Re \flat .
- Re \sharp suena exactamente igual que Mi \flat .
- Fa \sharp suena exactamente igual que Sol \flat .
- Sol \sharp suena exactamente igual que La \flat .
- La \sharp suena exactamente igual que Si \flat .

Podemos ver la relación de tonos y semitonos (1/2 tonos) en la siguiente figura:

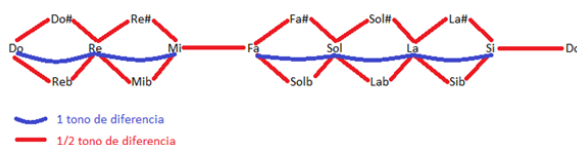


Figura 3.1: Relacion de tonos y semitonos

Como se puede observar entre Mi y Fa existe un semitono, al igual que entre Si y Do.

El pentagrama es un sistema de cinco líneas y cuatro espacios para escribir música:

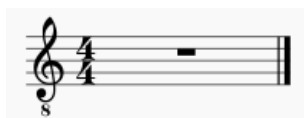


Figura 3.2: Pentagrama

Este sistema puede tener diferentes elementos dentro de los principales tenemos:

- Clave.
- Compás.
- Armadura.
- Notas.

3.1.2. Claves musicales

Las claves musicales se usan para darle nombre y altura a las notas musicales dentro de un pentagrama.

Las claves más usadas en la música son: la clave de Sol, la clave de Fa y la clave de Do.

La clave de Sol se usa en instrumentos agudos como la guitarra, violín, entre otros. Esta clave normalmente se escribe empezando en la segunda línea del pentagrama para formar una especie de G. El hecho de que esta clave se escriba en la segunda línea establece que esa línea será llamada como el nombre de la clave, en este caso Sol:

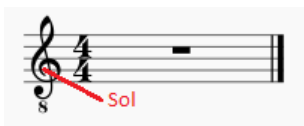


Figura 3.3: Clave de Sol en 2da. Línea

La clave de Fa es usada en instrumentos más graves, tales como el contrabajo, el bajo, etc. Normalmente se escribe empezando en la cuarta línea, de tal manera que esta línea tomara el nombre de Fa:

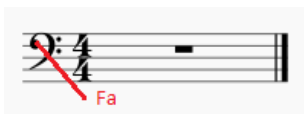


Figura 3.4: Clave de Fa en 4ta. Línea

La clave de Do es usada comúnmente para las voces, y esta clave se escribe en diferentes líneas dependiendo del timbre de la voz, para un tenor se usa la clave de Do en 4ta línea, mientras que para un soprano normalmente la clave se usa en 3ra o 2da línea:



Figura 3.5: Clave de Do en 4ta. y 3ra. línea

A partir de estas claves se le puede poner nombre a las diferentes líneas y espacios del pentagrama, por ejemplo en la clave de sol en segunda línea, el pentagrama quedaría de la siguiente forma:

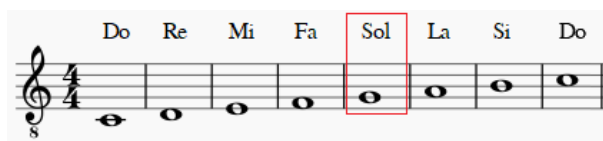


Figura 3.6: Nombre de las notas en un pentagrama con clave de Sol

3.1.3. Compases y tiempo

El compás se puede definir como la unidad métrica de la música, es la que nos dice que tiempo llevara la canción.

Existen una infinidad de compases los cuales podemos clasificar en dos grandes grupos:

- Compases regulares.
- Compases irregulares.

Los compases regulares están regidos por formas regulares las cuales dictan el tiempo, mientras que en los compases irregulares hay que determinar la base de tiempo de forma indirecta.

Cada nota puede tener un valor de tiempo y este valor estará determinado por el compás que se esté utilizando, por ejemplo en un compás de 4/4 las figuras musicales tendrán los siguientes valores:















| NOMBRE | FIGURA | SILENCIO | VALOR/PULSOS |
|-------------|---|---|--------------|
| 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 |

"Barraza, D. (2012), Figuras Musicales [Duración de las notas musicales]. Recuperado de <https://musicateoria.wordpress.com/figuras-musicales/>"

Figura 3.7: Duración de las notas musicales

Como se puede observar para cada figura musical existe su silencio correspondiente, el cual tendrá el mismo valor solo que este caso no se producirá sonido alguno.

Podemos tener también los valores relativos de estas notas respecto a otras notas:

| | |  |  |  |  |  |  |  |
|-------------|---|---|---|---|---|---|---|---|
| REDONDA |  | 1 | 2 | 4 | 8 | 16 | 32 | 64 |
| BLANCA |  | $\frac{1}{2}$ | 1 | 2 | 4 | 8 | 16 | 32 |
| NEGRA |  | $\frac{1}{4}$ | $\frac{1}{2}$ | 1 | 2 | 4 | 8 | 16 |
| CORCHEA |  | $\frac{1}{8}$ | $\frac{1}{4}$ | $\frac{1}{2}$ | 1 | 2 | 4 | 8 |
| SEMICORCHEA |  | $\frac{1}{16}$ | $\frac{1}{8}$ | $\frac{1}{4}$ | $\frac{1}{2}$ | 1 | 2 | 4 |
| FUSA |  | $\frac{1}{32}$ | $\frac{1}{16}$ | $\frac{1}{8}$ | $\frac{1}{4}$ | $\frac{1}{2}$ | 1 | 2 |
| SEMIFUSA |  | $\frac{1}{64}$ | $\frac{1}{32}$ | $\frac{1}{16}$ | $\frac{1}{8}$ | $\frac{1}{4}$ | $\frac{1}{2}$ | 1 |

"Gonzalez, T. (2015), El valor de las notas musicales en tiempo [Valor relativo de las notas]. Recuperado de <http://www.tucucu.com/2015/05/25/el-valor-de-las-notas-musicales-en-tiempo/>"

Figura 3.8: Valor relativo de las notas

De tal manera que en un compás tendremos dos datos los cuales se describen a continuación:

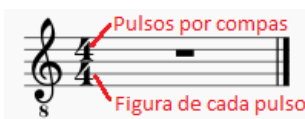


Figura 3.9: Compas

Se puede ver que este compas tendrá 2 pulsos y el valor de cada pulso será de un tiempo de negra, ya que el valor relativo de la negra respecto a la redonda es de $1/4$.

Otro punto a considerar en los compases es que estos tienen tiempos fuertes, semifuertes y débiles. La cuadratura de una armonía usa estos tiempos para indicar

los cambios que se deben de hacer.

Este es un ejemplo de los tiempos en compases de 4/4, 3/4 y 2/4:



Figura 3.10: Tiempos fuertes y débiles

3.1.4. Escalas

Las escalas son una serie de notas musicales que siguen un orden establecido por intervalos desde una nota base. En el mundo de la música hay una infinidad de escalas pero todas parten de la escala mayor de Do:

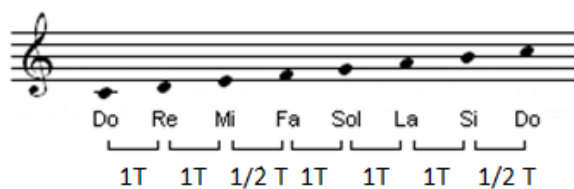


Figura 3.11: Escala de Do Mayor

En la escala mayor se tiene los siguientes intervalos: tono, tono, semitono, tono, tono, tono, semitono, si esta fórmula la aplicamos con las otras notas musicales podremos construir todas las escalas mayores.

Por ejemplo la escala de sol mayor quedaría de la siguiente manera:

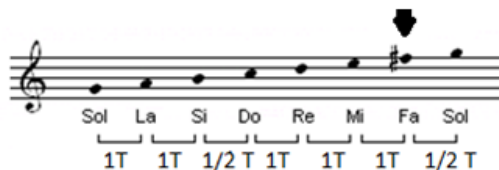


Figura 3.12: Escala de Sol Mayor

Para poder completar el tono completo de la fórmula de la escala mayor entre Mi y Fa se tuvo que poner una alteración en la nota de Fa.

La única escala mayor que no posee alteraciones es la escala mayor de Do, de ahí en más todas las demás escalas mayores tendrán al menos una alteración.

Las escalas menores parten de la escala mayor obteniendo su sexta nota y siguiendo las mismas notas de la escala mayor. Estas escalas menores se les conocen como menores relativas, ya que se basan en una escala mayor para su formación.

Por ejemplo la escala relativa de Do mayor sería La menor y esta escala posee exactamente las mismas notas que la escala mayor solamente que empieza desde la nota de La:

En este caso las notas son: La, Si, Do, Re, Mi, Fa, Sol, La.

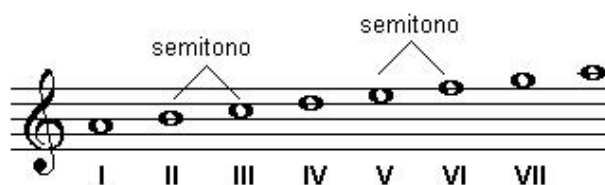


Figura 3.13: Escala de La menor

Esta relatividad puede ser usada con todas las escalas mayores para obtener sus relativas menores.

A pesar que las escalas mayores y menores poseen las mismas notas no deben ser nunca confundidas ya que el sonido final producido es muy diferente, mientras las escalas mayores se usan para canciones se puede decir hasta cierto punto alegres, las escalas menores normalmente acompañan melodías melancólicas, esto no es en todos los casos.

3.1.5. Armaduras

Todas las alteraciones de una escala se pueden juntar al inicio del pentagrama para dar lugar a lo que se conoce como armaduras.

Estas armaduras indicaran que todas las notas que poseen alteraciones las mantendrán a lo largo de toda la canción.

Debido a que las escalas mayores y sus relativas menores poseen las mismas alteraciones por lo tanto comparten también la misma armadura:

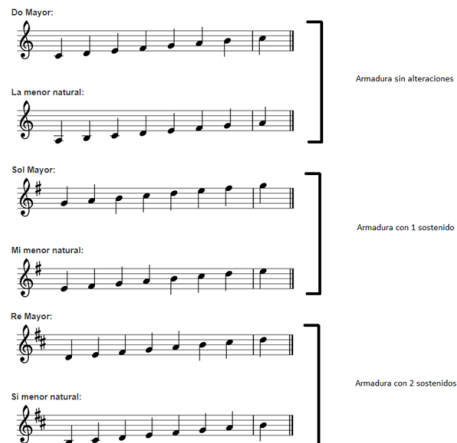


Figura 3.14: Armaduras relativas con sostenidos

También podemos tener armaduras con bemoles:

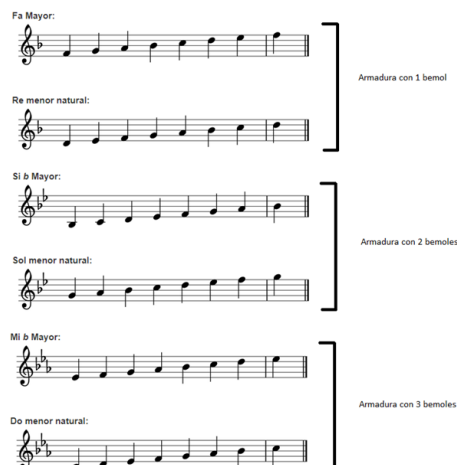


Figura 3.15: Armaduras relativas con bemoles

3.1.6. Tonalidades

La tonalidad de una canción se basa en la escala base de la canción y esta la podemos averiguar viendo la armadura que posee la canción, aunque como se vio anteriormente las escalas mayores y sus relativas poseen la misma armadura, así que antes de definir la tonalidad en base a la armadura también se debe de hacer un análisis de la interacción de las notas en la canción.

Dentro de las tonalidades tenemos tonalidades mayores y menores, por ejemplo si vemos un pentagrama el cual no posee alteraciones podríamos asumir que la canción

esta en Do mayor o en La menor, el siguiente paso sería ver la interacción de las notas en la canción para determinar correctamente la tonalidad.

3.1.7. Melodías y Armonías

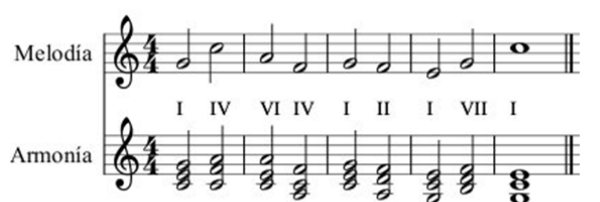
Una melodía es una sucesión de notas de forma ascendente o descendente que llevan cierta cordura.

Las melodías suelen estar formadas por frases y generalmente se repiten a lo largo de una canción variando algunas notas intermedias. En este caso se trata de una sucesión de notas que no son tocadas al mismo tiempo sino que una nota es tocada después de la otra.

Las armonías son una conjunción de sonidos tocados al mismo tiempo, normalmente la sucesión de varios acordes armónicos está ligado directamente con la melodía de la canción.

La armonía ha cambiado considerablemente desde la época de la música barroca hasta la música moderna, anteriormente para generar sistemas armónicos se usaban varios instrumentos tocando una nota en específico y la conjunción de todos los sonidos daba como resultado la armonía, actualmente la armonía es creada a partir de acordes de instrumentos que puedan generar este tipo de condiciones.

Esta es una comparativa para diferenciar entre lo que sería una melodía y una armonía:



"Wikipedia (2011), Composición musical [Melodía vs Armonía]. Recuperado de https://es.wikipedia.org/wiki/Composici%C3%B3n_musical"

Figura 3.16: Melodía y Armonía

3.2. Formato MIDI

MIDI (Interfaz digital de instrumentos musicales) es un estándar técnico que describe un protocolo, una interfaz digital y conectores para la interoperabilidad entre varios instrumentos musicales electrónicos, software y dispositivos. MIDI transmite

mensajes de eventos que especifican información de notas (como tono y velocidad), así como señales de control para parámetros (como volumen, vibrato y señales de reloj). Hay cinco tipos de mensajes y aquí solo consideramos el tipo de Canal de voz, que transmite datos de rendimiento en tiempo real a través de un solo canal. Dos mensajes importantes son:

- **Note on.-** Indica que una nota debe ser reproducida. Contiene la información de estado (número de canal, especificado por un entero dentro de [0 15] y dos valores de datos: un número de nota MIDI (el tono de la nota, un entero dentro de [0 127]) y una velocidad (que indica cómo la nota es reproducida, un entero dentro de [0 127]). Un ejemplo es ¡Note on, 0, 60, 50! que interpreta como: ^{en} el canal 1, comienza a reproducir un C medio con una velocidad de 50”.
- **Note off.-** Indica que una nota termina. En esa situación, la velocidad indica qué tan rápido se libera la nota. Un ejemplo es ¡Note off, 0, 60, 20! que interpreta como: ^{en} el canal 1, deja de reproducir un C medio con una velocidad de 20”.

Cada evento de nota está realmente incrustado en un fragmento de pista, una estructura de datos que contiene un valor de tiempo delta que especifica la información de temporización y el evento en sí. Un valor de tiempo delta representa la posición de tiempo, como un valor absoluto, del evento y podría representar:

- **Metrical time.-** Representa el numero de pulsos desde el comienzo. Una referencia llamada división y es definida en el encabezado del archivo, especificando cuantos pulsos por nota de cuarto.
- **time-code-based time.-** Representa el tiempo relacionado con horas, minutos y segundos de la canción.

Un ejemplo de un extracto de un archivo MIDI y su partitura correspondiente es mostrado en las siguientes figuras. La división ah sido puesta en 384 pulsos por cuarto de nota lo cual corresponde a 96 pulsos por un octavo de nota:

```

2, 96, Note_on_c, 0, 60, 90
2, 192, Note_off_c, 0, 60, 0
2, 192, Note_on_c, 0, 62, 90
2, 288, Note_off_c, 0, 62, 0
2, 288, Note_on_c, 0, 64, 90
2, 384, Note_off_c, 0, 64, 0
2, 384, Note_on_c, 0, 65, 90
2, 480, Note_off_c, 0, 65, 0
2, 480, Note_on_c, 0, 62, 90
2, 576, Note_off_c, 0, 62, 0

```

Figura 3.17: Extracto de un formato MIDI



Figura 3.18: Partitura correspondiente al extracto MIDI

Dentro del formato MIDI se pueden representar en cada track un instrumento diferente e independiente de los demás, esto permite una fácil manipulación de los diferentes instrumentos. A continuación se muestra una tabla de todos los instrumentos que pueden ser representados en este formato:

| PIANO | CHROM. PERCUS. | ORGAN | GUITAR |
|--|---|--|--|
| 01 - Acoustic Grand 02 - Bright Acoustic 03 - Electric Grand 04 - Honky-Tonk 05 - Electric Piano 1 06 - Electric Piano 2 07 - Harpsichord 08 - Clavinet | 09 - Celesta 10 - Glockenspiel 11 - Music Box 12 - Vibraphone 13 - Marimba 14 - Xylophone 15 - Tubular Bells 16 - Dulcimer | 17 - Drawbar Organ 18 - Percussive Organ 19 - Rock Organ 20 - Church Organ 21 - Reed Organ 22 - Accoridan 23 - Harmonica 24 - Tango Accordion | 25 - Nylon String Guitar 26 - Steel String Guitar 27 - Electric Jazz Guitar 28 - Electric Clean Guitar 29 - Electric Muted Guitar 30 - Overdriven Guitar 31 - Distortion Guitar 32 - Guitar Harmonics |
| BASS | SOLO STRINGS | ENSEMBLE | BRASS |
| 33 - Acoustic Bass 34 - Electric Bass(finger) 35 - Electric Bass(pick) 36 - Fretless Bass 37 - Slap Bass 1 38 - Slap Bass 2 39 - Synth Bass 1 40 - Synth Bass 2 | 41 - Violin 42 - Viola 43 - Cello 44 - Contrabass 45 - Tremolo Strings 46 - Pizzicato Strings 47 - Orchestral Strings 48 - Timpani | 49 - String Ensemble 1 50 - String Ensemble 2 51 - SynthStrings 1 52 - SynthStrings 2 53 - Choir Aahs 54 - Voice Oohs 55 - Synth Voice 56 - Orchestra Hit | 57 - Trumpet 58 - Trombone 59 - Tuba 60 - Muted Trumpet 61 - French Horn 62 - Brass Section 63 - SynthBrass 1 64 - SynthBrass 2 |
| REED | PIPE | SYNTH LEAD | SYNTH PAD |
| 65 - Soprano Sax 66 - Alto Sax 67 - Tenor Sax 68 - Baritone Sax 69 - Oboe 70 - English Horn 71 - Bassoon 72 - Clarinet | 73 - Piccolo 74 - Flute 75 - Recorder 76 - Pan Flute 77 - Blown Bottle 78 - Skakuhachi 79 - Whistle 80 - Ocarina | 81 - Square Wave 82 - Saw Wave 83 - Syn. Calliope 84 - Chiffer Lead 85 - Charang 86 - Solo Vox 87 - 5th Saw Wave 88 - Bass& Lead | 89 - Fantasia 90 - Warm Pad 91 - Polysynth 92 - Space Voice 93 - Bowed Glass 94 - Metal Pad 95 - Halo Pad 96 - Sweep Pad |
| SYNTH EFFECTS | ETHNIC | PERCUSSIVE | SOUND EFFECTS |
| 97 - Ice Rain 98 - Soundtrack 99 - Crystal 100 - Atmosphere 101 - Brightness 102 - Goblin 103 - Echo Drops 104 - Star Theme | 105 - Sitar 106 - Banjo 107 - Shamisen 108 - Koto 109 - Kalimba 110 - Bagpipe 111 - Fiddle 112 - Shanai | 113 - Tinkle Bell 114 - Agogo 115 - Steel Drums 116 - Woodblock 117 - Taiko Drum 118 - Melodic Tom 119 - Synth Drum 120 - Reverse Cymbal | 121 - Guitar Fret Noise 122 - Breath Noise 123 - Seashore 124 - Bird Tweet 125 - Telephone Ring 126 - Helicopter 127 - Applause 128 - Gunshot |

"María Quintanilla (2010). Protocolo general MIDI [Instrumentos en formato MIDI]. Recuperado de <http://cpms-acusticamusal.blogspot.com/2010/05/>"

Figura 3.19: Instrumentos en formato MIDI

3.3. Algoritmo Krumhansl-Schmuckler

Capítulo 4

Desarrollo

En este proyecto se busca que las redes neuronales sean capaces de aprender las diferentes tonalidades de las canciones, y que puedan crear nuevas composiciones que tengan sentido musicalmente hablando, tomando como base una canción de entrada en cierta tonalidad.

Esto será de gran ayuda para un músico ya que podrá introducir sus propias canciones y en base a las notas de sus canciones la red será capaz de generar música completamente nueva en la misma tonalidad de la canción de entrada. Estas canciones en definitiva no son composiciones finales, el músico podrá identificar las frases que sean de su agrado y continuar con la composición.

4.1. Datos

La base de datos de archivos MIDI que se utilizó en este proyecto se llama "Clean MIDI subset" la cual se usó en la tesis doctoral [1]. Esta base de datos contiene más de 17,000 canciones en formato MIDI, estas en su mayoría del género Pop y Rock, por lo que la red utilizada en este proyecto fue entrenada para asimilar este tipo de estilos.

Para el procesamiento de los archivos MIDI se utilizó una librería de software libre llamada Music21, la cual nos permite de una manera fácil trabajar con estos archivos, sin embargo se realizó una adaptación para que funcionara correctamente con diferentes sonidos de guitarra y bajo eléctrico. Esta librería es compatible con Python3 por lo que nuestro programa está hecho en este lenguaje de programación.

Los archivos MIDI de esta base de datos poseen varios Tracks, cada uno con un instrumento diferente, de los cuales en este proyecto solamente se tomarán en cuenta

los Tracks de guitarra y bajo eléctrico, los demás Tracks serán ignorados.

Cada canción tiene una duración promedio de 3 minutos, por lo que la cantidad de notas que procesa la red es bastante considerable.

4.2. Arquitectura de la aplicación

Este proyecto posee varios módulos, cada uno funciona independientemente de los otros, sin embargo existen interconexiones entre ellos para el paso de información, esto nos permite tener una buena modulación del código y realizar cambios en un módulo sin afectar a los otros.

Estos módulos se pueden observar en la siguiente figura:

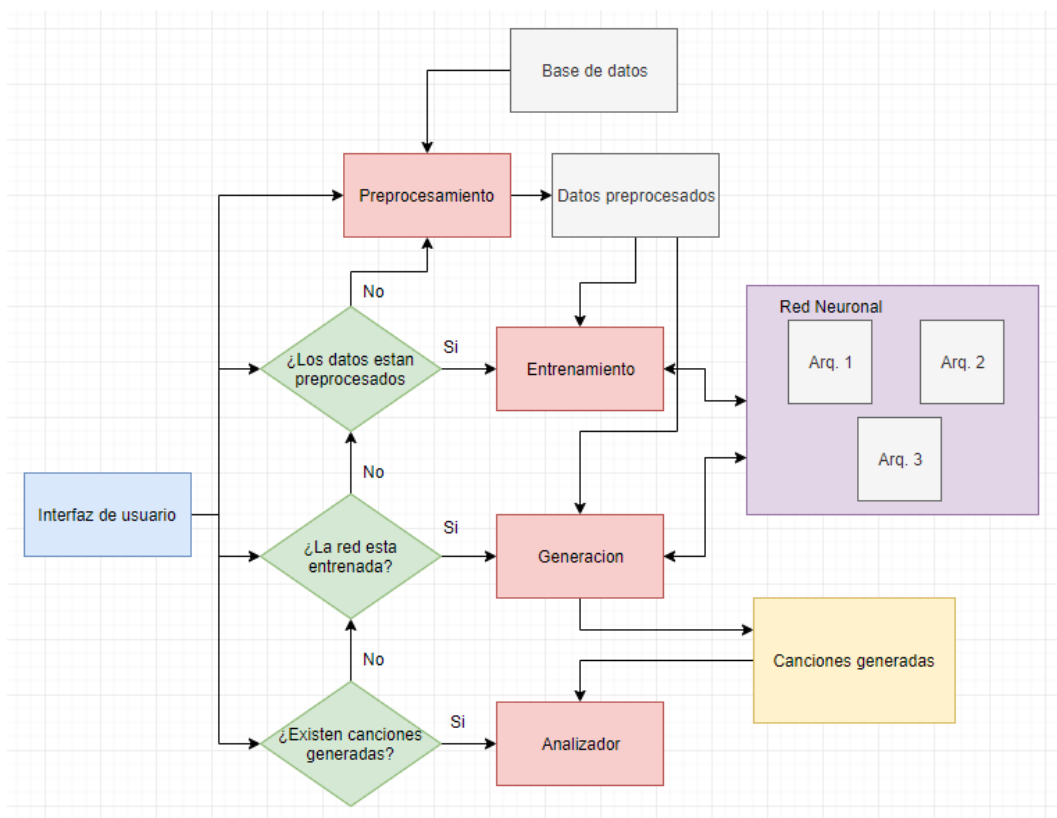


Figura 4.1: Arquitectura de la aplicación

Dentro de esta arquitectura tenemos módulos activos y módulos pasivos. Los módulos activos son aquellos que reciben y procesan información, mientras que los elementos pasivos solamente son para almacenar o consultar información.

Los elementos activos que se observan son:

- UI.
- Preprocesamiento.
- Red neuronal.
- Entrenamiento.
- Generación.
- Analizador.

Mientras que los elementos pasivos son:

- Base de datos.
- Datos preprocesados.
- Canciones generadas.

A continuación se describirán cada modulo del sistema.

4.2.1. UI

Este modulo es una pequeña interfaz de usuario de consola basada en menus, la cual nos permite acceder a los otros modulos del sistema.

El menu principal contiene el acceso a los siguientes modulos:

- 1.- Preprocesamiento.
- 2.- Entrenamiento.
- 3.- Generación.
- 4.- Analizador.

Como se puede observar en la figura 4.1 hay algunos modulos que para poderse ejecutar es necesario ejecutar otro modulo primero, con el fin de obtener la información necesaria para este modulo. El único modulo que no tiene condicionantes es el de Preprocesamiento, por lo tanto es el primer modulo que se debería de ejecutar en el programa.

4.2.2. Preprocesamiento

En este modulo se realiza el preprocesamiento de la base de datos, esto es con el fin de extraer únicamente los Tracks de guitarra y bajo de los archivos MIDI.

Como se menciona anteriormente esta base de datos tiene archivos de musica pop y rock, sin embargo, no todos los archivos son validos para ser procesados por este programa, debido a que algunos no contienen informacion completa en sus Tracks para su correcto procesamiento usando la libreria Music21.

La libreria Music21 esta enfocada principalmente para hacer reconocimiento de pianos y sus derivados, por lo tanto se tuvo que realizar un reacondicionamiento de esta libreria para el correcto funcionamiento con los instrumentos de guitarra y bajo.

El modo en que se puede reconocer si un track es de guitarra o de bajo es en base a un evento llamado "Program Change", este evento no lo poseen todos los Tracks MIDI por lo que en ocasiones es complicado determinar cual es el instrumento que se esta ejecutando, en estos casos el sintetizador MIDI es el encargado de asignarle un instrumento por defecto, el cual generalmente es el piano.

Estos eventos de "Program Change" estan agrupados en familias, de esta manera es mas facil su identificación, como se ah mencionado a nosotros unicamente nos interesa la familia de guitarra y bajo, por lo tanto los unicos "Program Change" que estaremos buscando son:

| Program Change | Familia |
|----------------|----------|
| 25-32 | Guitarra |
| 33-40 | Bajo |

Tabla 4.1: Familias de guitarra y bajo.

Este elemento de "Program Change" no es una propiedad inherente de los Tracks, por lo que hay que hacer una busqueda en todos los eventos de cada Track.

Normalmente encontraremos varios Tracks que contengan sonidos de guitarra o bajo en un archivo MIDI y esto se debe que para tocar por ejemplo una guitarra con distorsion se utiliza un Track, y para un sonido limpio se utiliza otro Track. Tambien existen casos en los cuales aun y cuando se trate del mismo sonido las notas están exparsidas en varios Tracks, esto depende de como fue que se creo el archivo MIDI.

Es por estas circunstancias la importancia de tener este modulo, ya que no so-

lamente se encarga de separar los Tracks de guitarra y bajo en archivos separados, sino que tambien valida la integridad de los archivos MIDI y descarta los Tracks que tienen muchos silencios y muy pocas notas activas.

Para tener un procesamiento mas rapido a la hora del proceso de entrenamiento o generacion musical, se realiza la exportacion de las notas y silencios a un archivo de manera serializada, de esta manera nos evitamos estar procesando los archivos MIDI y solamente deserializamos el archivo exportado.

Como se observa en la figura 4.1 la salida de este modulo seran los archivos preprocesados los cuales son necesarios para realizar la parte de entrenamiento.

4.2.3. Red neuronal

En este proyecto se estan utilizando 3 tipos de arquitecturas diferentes para validar cual de las 3 arroja mejores resultados:

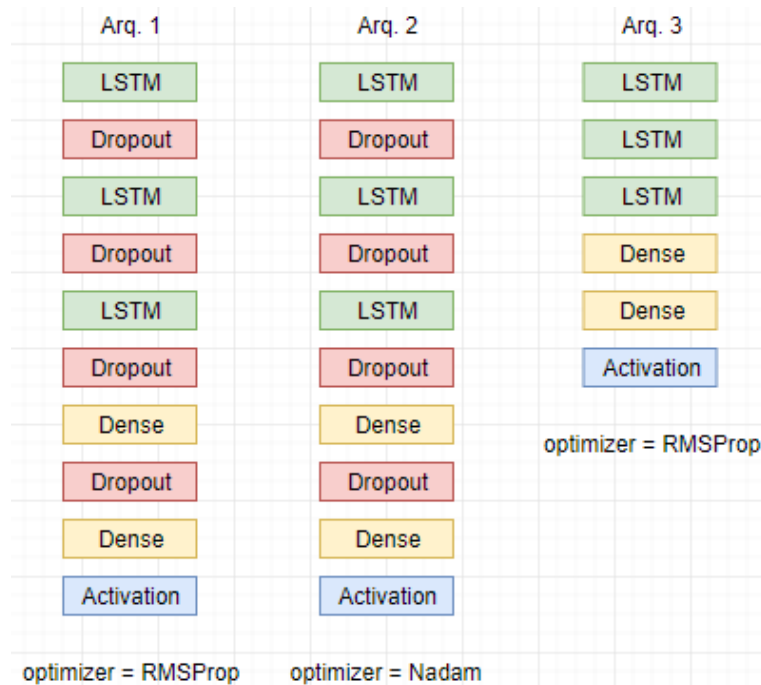


Figura 4.2: Arquitectura de las redes usadas

Como se puede observar las 3 redes poseen 3 capas de LSTM sin embargo en las 2 primeras se estan usando unas capas de Dropout intermedias, estas capas realizan la funcion de poner en 0 aleatoriamente algunas entradas para prevenir el sobre en-

trenamiento, sin embargo en la ultima arquitectura, se eliminaron estas capas para verificar el comportamiento que podría tener una red así.

Otro de los aspectos a verificar es el mejor optimizador, en este caso se esta usando un optimizador RMSProp en la primera y tercera arquitectura. Este optimizador funciona calculando dividiendo la tasa de aprendizaje por un promedio decreciente exponencial de gradientes cuadrados:

$$E[g^2]_t = 0.9E[g^2]_{t-1} + 0.1g_t^2 \quad (4.2.1)$$

$$\theta_{t+1} = \theta_t - \frac{\eta}{\sqrt{E[g^2]_t + \epsilon}} g_t \quad (4.2.2)$$

En la figura 4.2 se observa que la primera y segunda red son prácticamente iguales, lo único que las diferencia es que la segunda usa un optimizador Nadam, la cual es una combinación de RMSProp, Momentum y Nesterov:

$$\theta_{t+1} = \theta_t - \frac{\eta}{\sqrt{\hat{v}_t} + \epsilon} (\beta_1 \hat{m}_t + \frac{(1 - \beta_1)g_t}{1 - \beta_1^t}) \quad (4.2.3)$$

Las ultimas capas de las redes son densas las cuales interconectan todas las neuronas de una capa con otra, esto nos permite mejorar la tasa de aprendizaje.

Finalmente la capa de activación es por medio de una función softmax o función exponencial normalizada, la cual es una generalización de la función logística:

$$\sigma(z)_j = \frac{e^{z_j}}{\sum_{k=1}^K e^{z_k}} \quad \text{para } j = 1, \dots, K \quad (4.2.4)$$

4.2.4. Entrenamiento

En este modulo se realiza toda la parte del entrenamiento. Es necesario seleccionar el tipo de arquitectura que se usara para el entrenamiento, asi como también el instrumento a entrenar, es importante mencionar que cada instrumento es entrenado por separado.

Se toman los datos preprocesados (archivos exportados con las notas) los cuales son deserializados para obtener un arreglo únicamente con las notas y silencios, con este arreglo se empiezan a generar secuencias de 100 notas y silencios. Estas secuencias son insertadas en forma de matriz a una funcion de mapeo, la cual les asignara un valor

único a cada nota, esto es debido que la red neuronal solamente reconoce numeros. Finalmente esta matriz es normalizada para poder ser procesada por la red LSTM.

En la siguiente figura podemos observar como es el flujo de entrenamiento:



Figura 4.3: Flujo de entrenamiento

En este proyecto las diferentes arquitecturas fueron entrenadas con 200 épocas para cada instrumento, obteniendo buenos resultados de salida. Es posible entrenar usando más épocas sin embargo el poder de procesamiento nos limita un poco a esto, ya que para poder entrenar un instrumento lleva aproximadamente 2 semanas y media por cada red. Se está usando un servidor de Amazon con tarjetas NVIDIA para realizar todo el proceso de entrenamiento y aun así el tiempo de entrenamiento es bastante considerable.

En cada época se verifica si la red está teniendo una mejora en su aprendizaje, si es así se guarda un archivo con todos los pesos de las neuronas, con esto nos aseguramos de siempre guardar los mejores valores de pesos.

Además de este archivo de pesos, también se guarda un histórico con los valores de aprendizaje y de disminución del error, para poder graficarlo después y ver cómo fue evolucionando nuestra red.

De todo el conjunto de entrenamiento se usa un 75 % para la fase de entrenamiento y el otro 25 % para la fase de validación, de esta manera tenemos un gran conjunto de validación.

Los datos son introducidos a la red en forma de lotes, cada uno de tamaño 64 y la actualización de los pesos de la red se realiza después de que cada lote sea procesado.

4.2.5. Generación

Este es el módulo encargado de la generación de nuevas canciones en base a una canción de entrada, y una red entrenada para guitarra y bajo.

El usuario puede elegir con cual arquitectura realizar la generación musical, sin embargo, es importante mencionar que si no se tiene entrenada la arquitectura tanto para guitarra como para bajo no podrá ser usada.

La forma en que funciona este modulo es tomando una cancion de la base de datos preprocesados (archivos exportados con las notas), este archivo es deserializado para obtener un arreglo de notas y silencios, con el cual se generaran secuencias de 100 notas y silencios, tal como en el caso del modulo de entrenamiento son puestas en una matriz para posteriormente pasarlas por una función de mapeo.

Las secuencias son insertadas a la red una por una y usando la funcionalidad de prediccion de la red, se obtiene una secuencia de salida con las ponderaciones de la proximidad a la siguiente nota. Esta secuencia se analiza y se toma el numero mayor para descubrir cual es la siguiente nota generada. Este proceso se repite hasta tener el numero de notas deseadas por el usuario.

En esta figura se puede visualizar mas fácil este procedimiento:

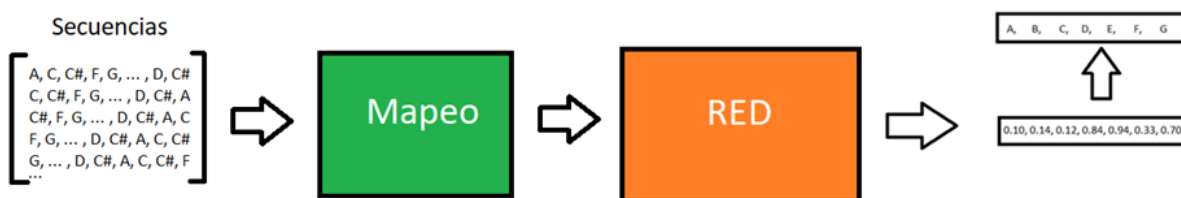


Figura 4.4: Flujo de generación de notas

El modulo generara la salida de guitarra y la de bajo, para posteriormente juntarlas en una sola cancion y el usuario podrá elegir si desea que la salida posea los mismos instrumentos de la canción base.

La canción de salida deberá tener la misma tonalidad de la canción base, esto no siempre se puede cumplir ya que la red no tiene un 100 % de eficacia, sin embargo, los resultados obtenidos en este proyecto fueron bastante alentadores llegando a un máximo de 75 % aproximadamente.

Las canciones son guardadas con un numero genérico seguido de una numeración, esto es para poder generar muchas canciones sin preocuparnos de que se puedan sobrescribir canciones anteriormente creadas.

4.2.6. Analizador

Finalmente el modulo de analizador como su nombre lo dice es el encargado de analizar las canciones generadas por el programa. Si no se tienen canciones generadas por el programa este modulo no podrá ser usado.

Este modulo contiene 3 utilerias:

- Visor.
- Reproductor.
- Analizador.

El visor nos permite visualizar una cancion en forma de tablatura en el programa que tengamos configurado por defecto en Music21, esto nos permite analizar las alturas y los intervalos entre las notas, así como también visualizar como se entrelazan la parte de guitarra con el bajo en la canción generada.

Es posible realizar modificaciones a las canciones creadas usando el visor, y verificar como se escucha con estas modificaciones. En este caso se tiene configurado MuseScore con Music21, el cual es un programa gratis para crear, reproducir e imprimir partituras.

El reproductor permite reproducir las canciones generadas en el reproductor por defecto instalado en Windows. Es importante mencionar que cada reproductor tiene su propio sonido a la hora de reproducir los archivos MIDI.

El analizador utiliza el algoritmo de Krumhansl-Schmuckler para realizar el calculo de la tonalidad de la canción generada. Este algoritmo retorna un arreglo con las posibles tonalidades de la canción, siendo el primer elemento la tonalidad con mayor probabilidad.

Para el calculo de la tonalidad se toma en cuenta las notas de las escalas para las diferentes tonalidades. Como se vio en el capitulo de Teoría musical, cada tonalidad tiene sus propias alteraciones, sin embargo existen tonalidades relativas, las cuales poseen exactamente las mismas alteraciones, por lo que en ocasiones este algoritmo determina que la tonalidad de la canción es una tonalidad relativa, en este caso tomamos como un calculo correcto, ya que para determinar mas a fondo si es una tonalidad relativa o no, es necesario analizar las notas dominantes de las canciones, así como las funciones tonales.

Una vez obtenida la tonalidad con este algoritmo, se compara contra la tonalidad con la que fue creada esta canción, si son iguales, lo catalogamos como un caso exitoso,

de esta manera podemos verificar si nuestra red en realidad esta aprendiendo a crear canciones con diferentes tonalidades.

Capítulo 5

Resultados

En esta sección se presentaran los resultados obtenidos de los diferentes experimentos que se realizaron en este proyecto. Estos experimentos se dividieron en 3 secciones:

- Etapa de entrenamiento.
- Etapa de generación.
- Etapa de validación.

En cada etapa se intenta evaluar a los diferentes instrumentos de la manera mas objetiva posible.

5.1. Etapa de entrenamiento

En la etapa de entrenamiento se usaron las 3 diferentes arquitecturas de redes neuronales de manera independiente, es decir, primeramente se entreno con una arquitectura y se guardaron los mejores valores de los pesos, para posteriormente realizar el mismo procedimiento con las otras arquitecturas.

Este entrenamiento fue realizado usando toda la base de datos preprocesados, los cuales tomaron en cuenta únicamente los archivos MIDI que contaran con al menos un Track de guitarra y uno de bajo.

Esta base de datos fue segmentada en 2 partes, dejando el 75 % de los datos para entrenamiento y el restante 25 % para la validación.

Las redes se entrenaron durante 200 epocas para cada instrumento, es decir, primero se entreno la red para que generara patrones de guitarra y luego se entreno para bajo, por lo tanto se tienen resultados de los 2 procesos.

5.1.1. Arquitectura 1

Esta arquitectura posee 3 capas de redes LSTM con capas de Dropout intermedias, así como también unas capas Densas al final y un optimizador RMSProp.

5.1.1.1. Guitarra

En la Figura 5.1 se muestra como fue disminuyendo el error usando esta arquitectura para guitarra.

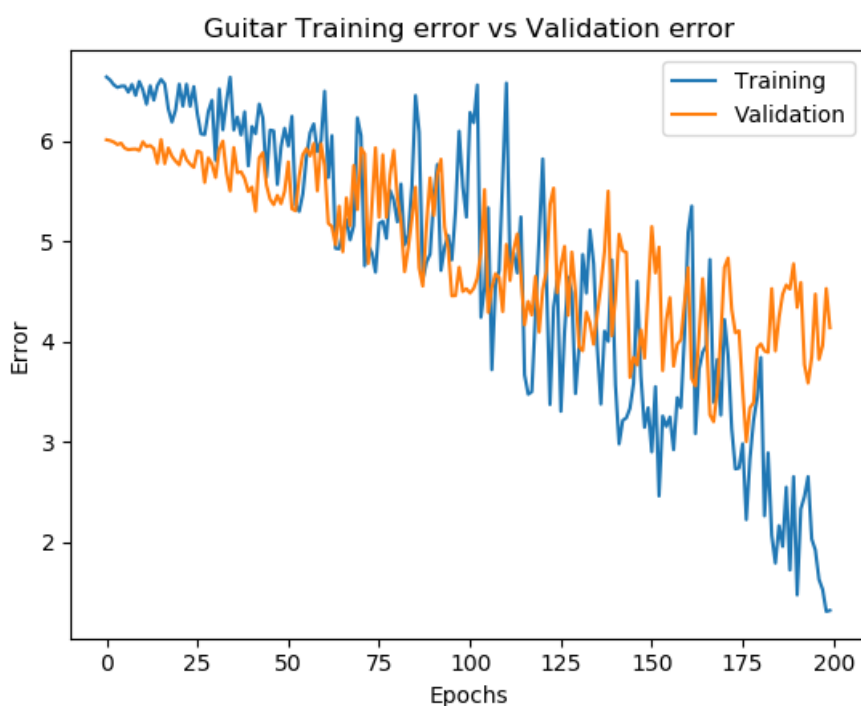


Figura 5.1: Grafica de error para guitarra en arquitectura 1

En la Tabla 5.1 se puede observar que la tendencia del error es a la baja, llegando a valores de 1.3161 en la fase de entrenamiento y de 4.1383 en la fase de validación, y aunque por la epoca 100 pareciera que el error va a tender a subir, no es así al final.

| Fase | Valor inicial | Valor final | Valor mínimo | Valor máximo |
|---------------|---------------|-------------|--------------|--------------|
| Entrenamiento | 6.6410 | 1.3161 | 1.3055 | 6.6410 |
| Validación | 6.0144 | 4.1383 | 3.0009 | 6.0167 |

Tabla 5.1: Valores de error para guitarra en arquitectura 1.

En la Figura 5.2 se muestra como fue aprendiendo la red con esta arquitectura para guitarra.

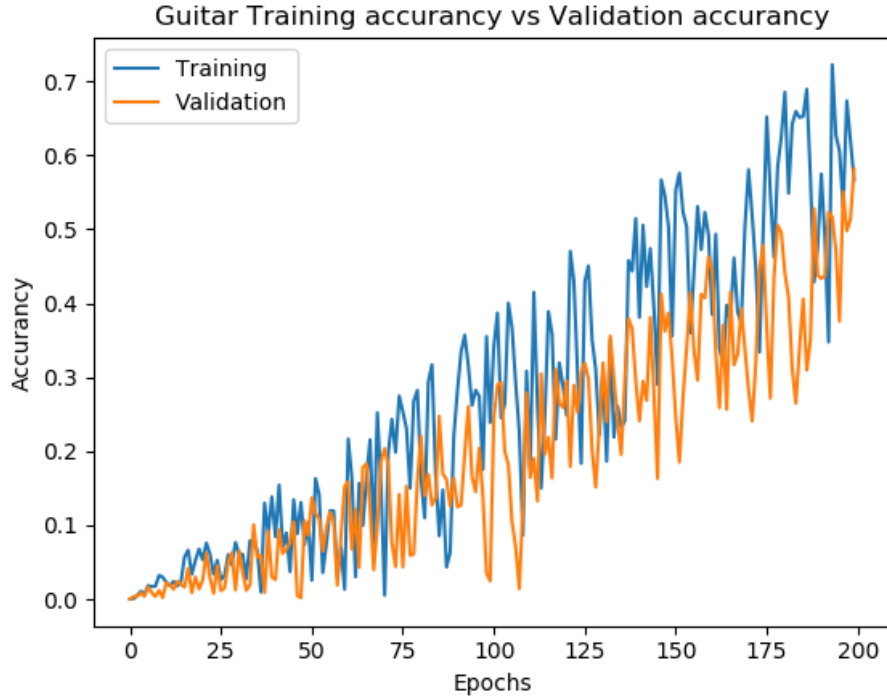


Figura 5.2: Grafica de aprendizaje para guitarra en arquitectura 1

En la Tabla 5.2 se observa que para la fase de entrenamiento se llego a un valor final de 56.66 % y en la fase de validación a un 58.18 %, este porcentaje podría parecer bajo, sin embargo se tuvieron máximos en algunas épocas de hasta 72.26 % y 58.18 % respectivamente.

El resultado final de esta arquitectura es bastante bueno, ya que la fase de validación se comporto de manera muy lineal en comparación de la fase de entrenamiento, obteniendo un valor muy parecido al final.

| Fase | Valor inicial | Valor final | Valor mínimo | Valor máximo |
|---------------|---------------|-------------|--------------|--------------|
| Entrenamiento | 0.0001 | 0.5666 | 0.0001 | 0.7226 |
| Validación | 0.0001 | 0.5818 | 0.0001 | 0.5818 |

Tabla 5.2: Valores de aprendizaje para guitarra en arquitectura 1.

5.1.1.2. Bajo

En la Figura 5.3 se muestra como fue disminuyendo el error usando esta arquitectura para bajo.

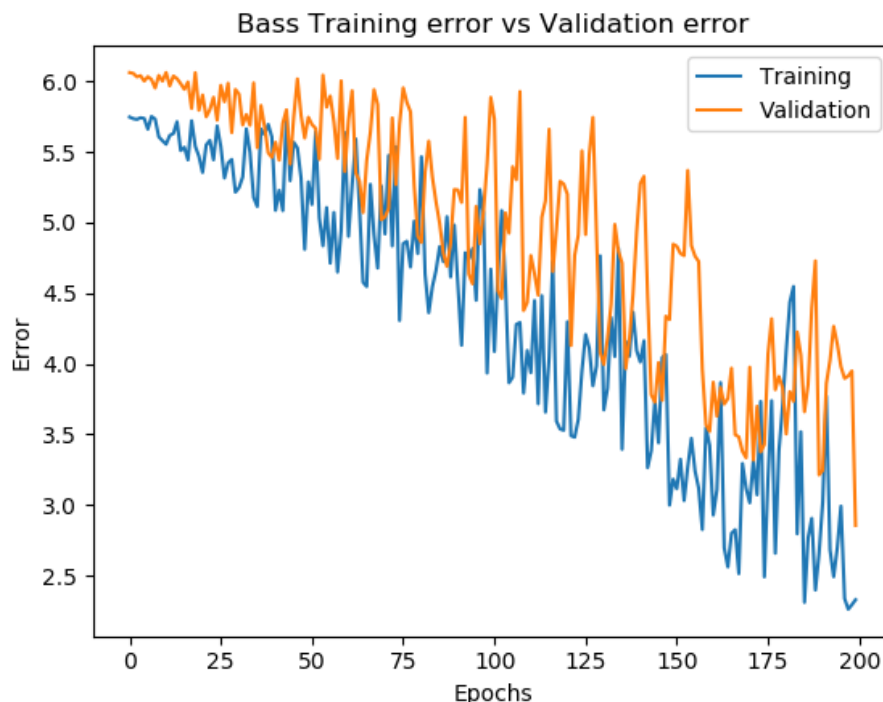


Figura 5.3: Grafica de error para bajo en arquitectura 1

En la Tabla 5.3 se puede observar que el error llego a valores de 2.3281 en la fase de entrenamiento y de 2.8547 en la fase de validación.

Despues de la epoca 150 parece que el error va a volver a subir tanto en la fase de entrenamiento como en la de validación, sin embargo al pasar las epocas se recupera la tendencia a la baja.

| Fase | Valor inicial | Valor final | Valor mínimo | Valor máximo |
|---------------|---------------|-------------|--------------|--------------|
| Entrenamiento | 5.7464 | 2.3281 | 2.2614 | 5.7531 |
| Validación | 6.0615 | 2.8547 | 2.8547 | 6.0620 |

Tabla 5.3: Valores de error para bajo en arquitectura 1.

En la Figura 5.4 se muestra como fue aprendiendo la red con esta arquitectura para bajo.

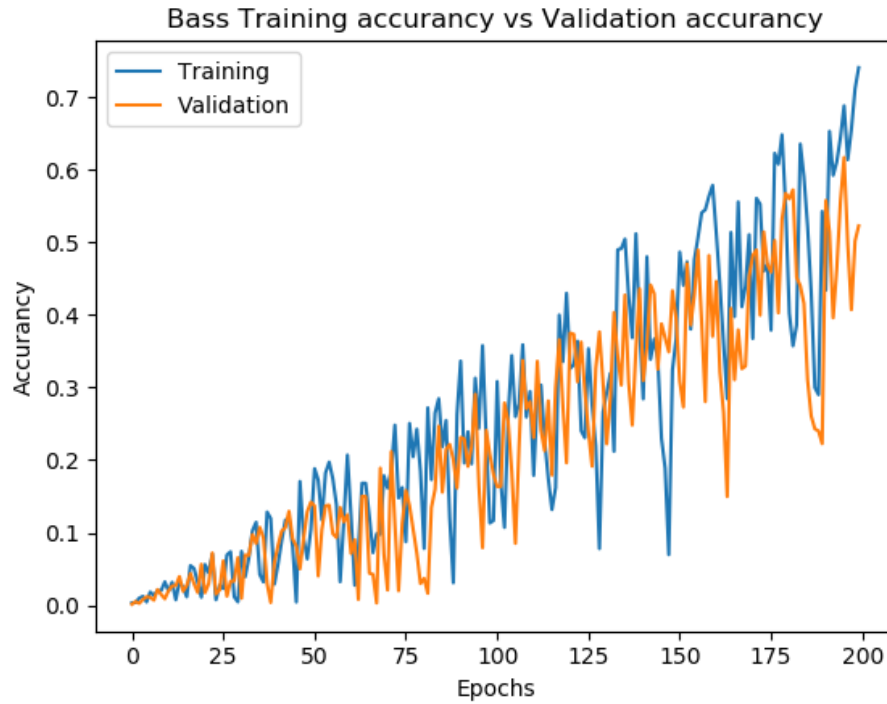


Figura 5.4: Gráfica de aprendizaje para bajo en arquitectura 1

En la Tabla 5.4 se observa que para la fase de entrenamiento se llegó a un valor final de 74.08 % y en la fase de validación a un 52.28 %.

Al igual que la guitarra, el bajo tuvo un excelente aprendizaje con esta arquitectura, a pesar de que tuvo algunas caídas bastante pronunciadas, cerca de la época 150.

El valor máximo obtenido es bastante similar al valor máximo para la guitarra con esta misma arquitectura.

| Fase | Valor inicial | Valor final | Valor mínimo | Valor máximo |
|---------------|---------------|-------------|--------------|--------------|
| Entrenamiento | 0.0034 | 0.7408 | 0.0034 | 0.7408 |
| Validación | 0.0014 | 0.5228 | 0.0014 | 0.6170 |

Tabla 5.4: Valores de aprendizaje para bajo en arquitectura 1.

5.1.2. Arquitectura 2

Esta arquitectura es muy similar a la numero 1, posee las mismas capas de LSTM, Dropout y Densas, sin embargo esta arquitectura usa un optimizador Nadam.

5.1.2.1. Guitarra

En la Figura 5.5 se muestra como fue disminuyendo el error usando esta arquitectura para guitarra. Se puede observar que en la fase de validación el error disminuyo mas lentamente que en la fase de entrenamiento.

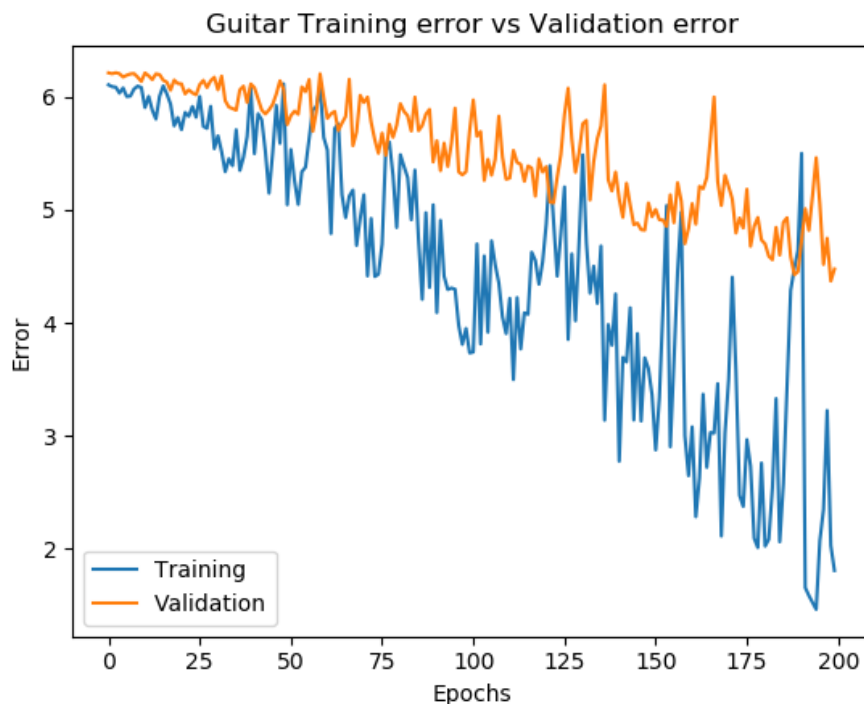


Figura 5.5: Grafica de error para guitarra en arquitectura 2

En la Tabla 5.5 se observa que el error disminuyo a niveles de 1.8103 y 4.4766 en las fases de entrenamiento y validación respectivamente.

| Fase | Valor inicial | Valor final | Valor mínimo | Valor máximo |
|---------------|---------------|-------------|--------------|--------------|
| Entrenamiento | 6.1054 | 1.8103 | 1.4658 | 6.1138 |
| Validación | 6.2114 | 4.4766 | 4.3697 | 6.2126 |

Tabla 5.5: Valores de error para guitarra en arquitectura 2.

En la Figura 5.6 se muestra como fue aprendiendo la red con esta arquitectura para guitarra. Se tuvieron grandes caídas en la fase de validación a partir de la época 150.

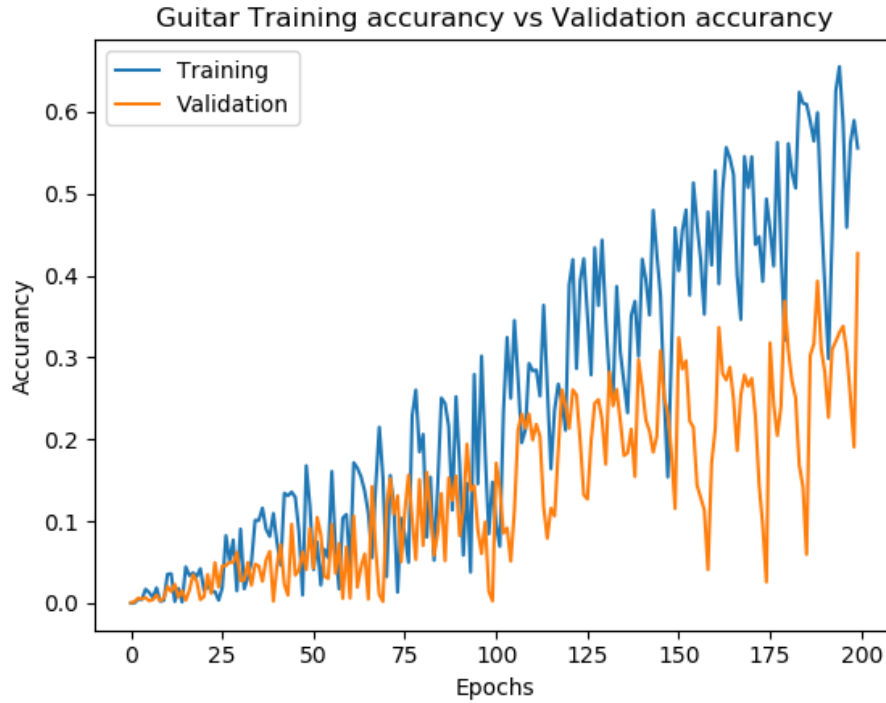


Figura 5.6: Gráfica de aprendizaje para guitarra en arquitectura 2

En la Tabla 5.6 se observa que para la fase de entrenamiento se llegó a un valor final de 55.58% y en la fase de validación a un 42.73%, obteniendo valores máximos en algunas épocas de hasta 65.55% y 42.73% respectivamente.

El desempeño de esta red para guitarra es un poco menor al de la arquitectura 1, a pesar de que el optimizador utiliza un algoritmo de momento para salir de mínimos locales.

| Fase | Valor inicial | Valor final | Valor mínimo | Valor máximo |
|---------------|---------------|-------------|--------------|--------------|
| Entrenamiento | 0.0000 | 0.5558 | 0.0000 | 0.6555 |
| Validación | 0.0007 | 0.4273 | 0.0007 | 0.4273 |

Tabla 5.6: Valores de aprendizaje para guitarra en arquitectura 2.

5.1.2.2. Bajo

En la Figura 5.7 se muestra como fue disminuyendo el error usando esta arquitectura para bajo. El error desciende de una forma mas lineal que en el caso de la guitarra.

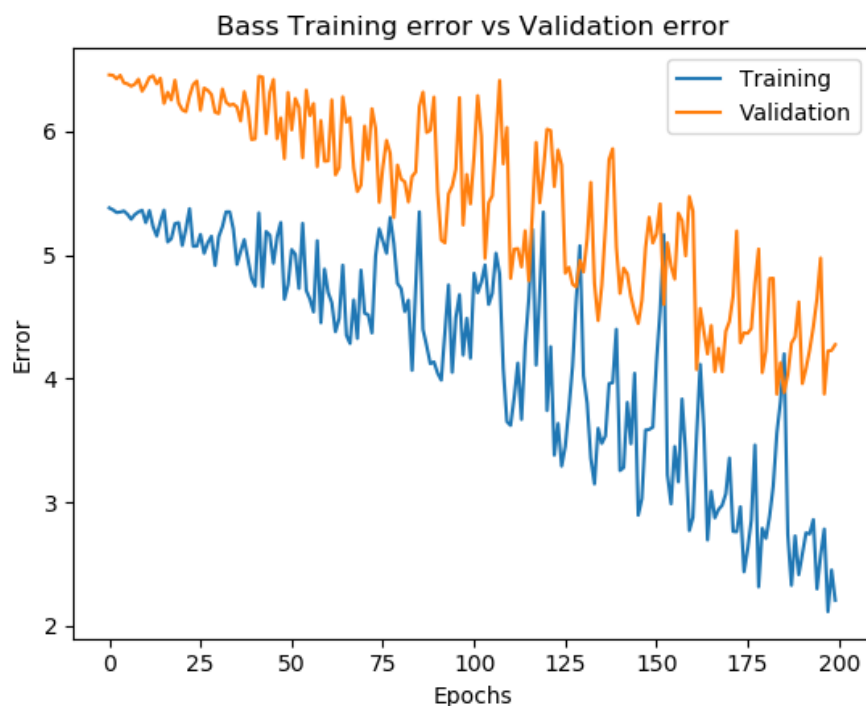


Figura 5.7: Grafica de error para bajo en arquitectura 2

En la Tabla 5.7 se observa que el error disminuyo a niveles de 2.2061 y 4.2744 en las fases de entrenamiento y validación respectivamente, sin embargo se tuvieron valores mínimos de 2.1150 y 3.8736 en épocas anteriores.

| Fase | Valor inicial | Valor final | Valor mínimo | Valor máximo |
|---------------|---------------|-------------|--------------|--------------|
| Entrenamiento | 5.3796 | 2.2061 | 2.1150 | 5.3796 |
| Validación | 6.4556 | 4.2744 | 3.8736 | 6.4556 |

Tabla 5.7: Valores de error para bajo en arquitectura 2.

En la Figura 5.8 se muestra como fue aprendiendo la red con esta arquitectura para bajo. La fase de validación no tuvo resultados tan buenos como la fase de entrenamiento.

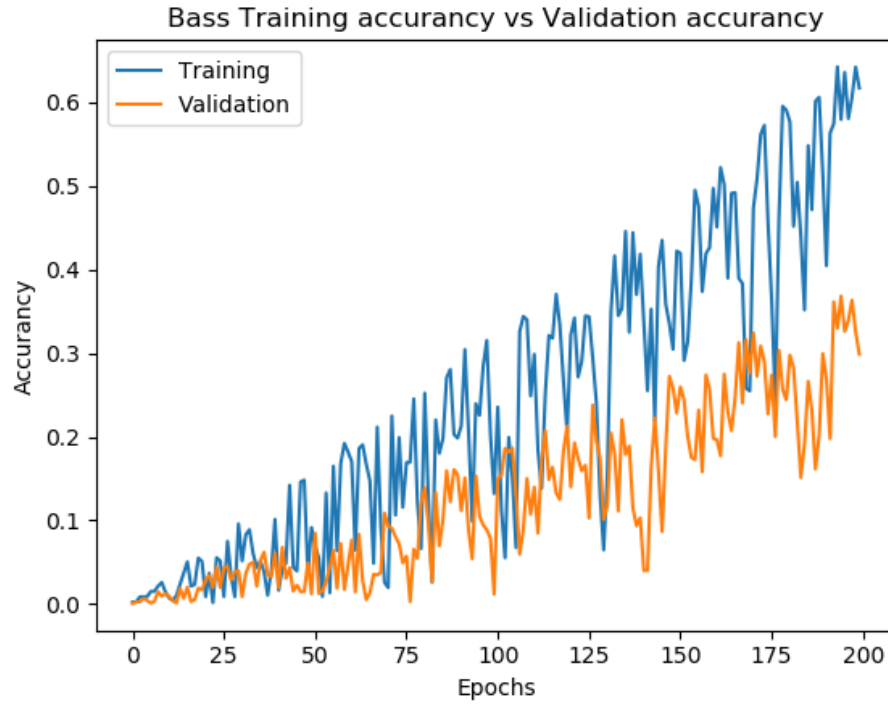


Figura 5.8: Grafica de aprendizaje para bajo en arquitectura 2

En la Tabla 5.8 se observa que para la fase de entrenamiento se llegó a un valor final de 61.70 % y en la fase de validación a un 29.88 %, obteniendo valores máximos en algunas épocas de hasta 64.26 % y 36.83 % respectivamente.

El rendimiento en la fase de validación es de más de un 50 % menor que en la fase de entrenamiento. Esto demuestra que la red no está aprendiendo a la velocidad deseada.

La guitarra obtuvo mejores resultados de aprendizaje que el bajo en la fase de validación siendo aproximadamente un 15 % mayor.

| Fase | Valor inicial | Valor final | Valor mínimo | Valor máximo |
|---------------|---------------|-------------|--------------|--------------|
| Entrenamiento | 0.0024 | 0.6170 | 0.0017 | 0.6426 |
| Validación | 0.0008 | 0.2988 | 0.0008 | 0.3683 |

Tabla 5.8: Valores de aprendizaje para bajo en arquitectura 2.

5.1.3. Arquitectura 3

En esta arquitectura se quitaron las capas de Dropout, por lo tanto la red es menos profunda. Al igual que en la arquitectura 1 se esta usando un optimizador RMSProp.

5.1.3.1. Guitarra

En la Figura 5.9 se muestra como fue cambiando el error usando esta arquitectura para guitarra. Se puede observar que el error va creciendo conforme las epocas pasan.

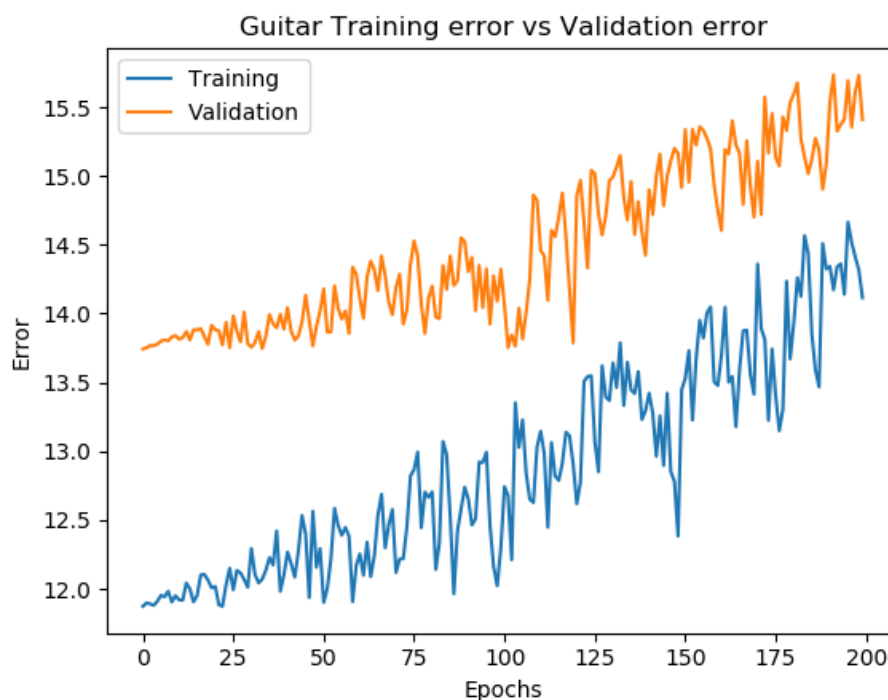


Figura 5.9: Gráfica de error para guitarra en arquitectura 3

En la Tabla 5.9 se puede observar que la tendencia del error es a la alza, llegando a valores de 14.1156 en la fase de entrenamiento y de 15.4095 en la fase de validación.

| Fase | Valor inicial | Valor final | Valor mínimo | Valor máximo |
|---------------|---------------|-------------|--------------|--------------|
| Entrenamiento | 11.8742 | 14.1156 | 11.8739 | 14.6645 |
| Validación | 13.7430 | 15.4095 | 13.7430 | 15.7345 |

Tabla 5.9: Valores de error para guitarra en arquitectura 3.

En la Figura 5.10 se muestra como fue aprendiendo la red con esta arquitectura para guitarra. A pesar de mostrar un comportamiento a la alza, los valores finales de esta grafica no son tan alentadores.

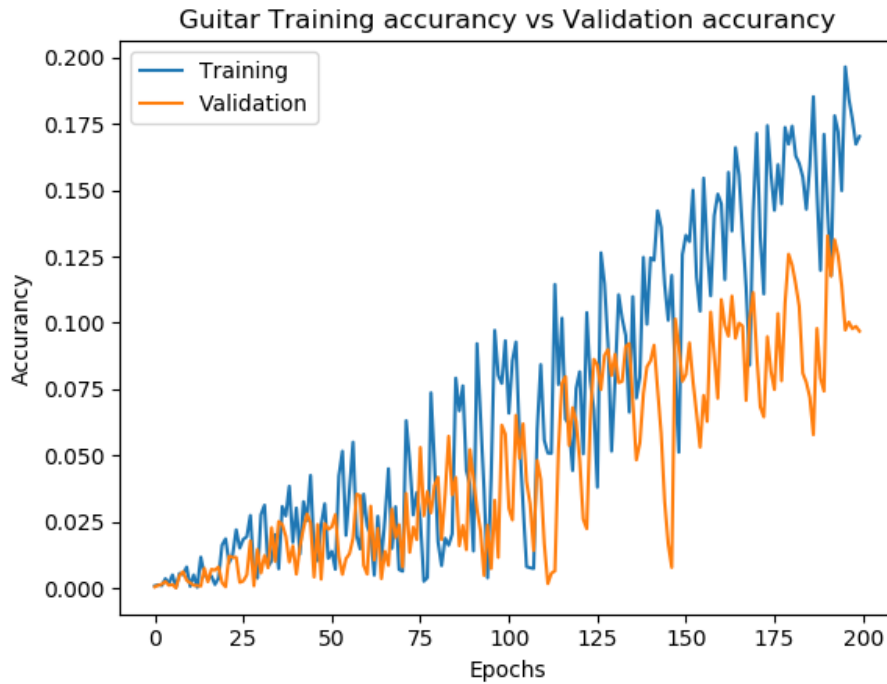


Figura 5.10: Grafica de aprendizaje para guitarra en arquitectura 3

En la Tabla 5.10 se observa que para la fase de entrenamiento se llego a un valor final de 17.02 % y en la fase de validación a un 9.69 %, obteniendo maximos de hasta 19.64 % y 13.28 % respectivamente.

Estos valores son bastante bajos y no garantizan que con el transcurso de las epocas esta arquitectura pueda seguir mejorando.

| Fase | Valor inicial | Valor final | Valor mínimo | Valor máximo |
|---------------|---------------|-------------|--------------|--------------|
| Entrenamiento | 0.0010 | 0.1702 | 0.0004 | 0.1964 |
| Validación | 0.0006 | 0.0969 | 0.0002 | 0.1328 |

Tabla 5.10: Valores de aprendizaje para guitarra en arquitectura 3.

5.1.3.2. Bajo

En la Figura 5.11 se muestra como fue cambiando el error usando esta arquitectura para bajo. Se puede observar un crecimiento del error conforme las épocas pasan.

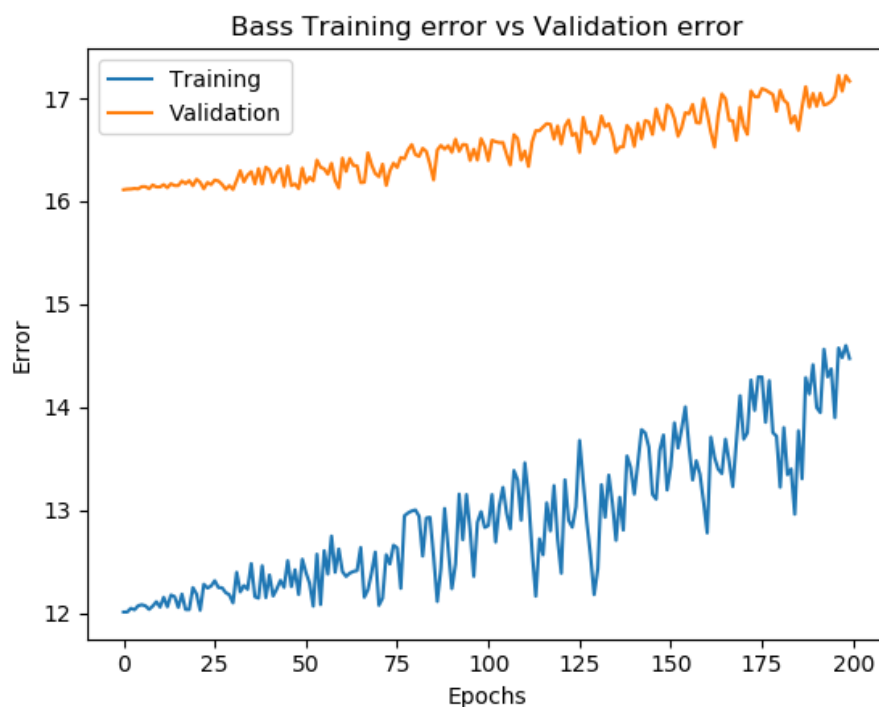


Figura 5.11: Gráfica de error para bajo en arquitectura 3

En la Tabla 5.11 se puede observar que la tendencia del error es a la alza, llegando a valores de 14.4731 en la fase de entrenamiento y de 17.1656 en la fase de validación.

| Fase | Valor inicial | Valor final | Valor mínimo | Valor máximo |
|---------------|---------------|-------------|--------------|--------------|
| Entrenamiento | 12.0105 | 14.4731 | 12.0105 | 14.5983 |
| Validación | 16.1120 | 17.1656 | 16.1120 | 17.2226 |

Tabla 5.11: Valores de error para bajo en arquitectura 3.

En la Figura 5.12 se muestra como fue aprendiendo la red con esta arquitectura para bajo. Esta grafica muestra que el bajo va aprendiendo pero la tasa de aprendizaje es bastante baja.

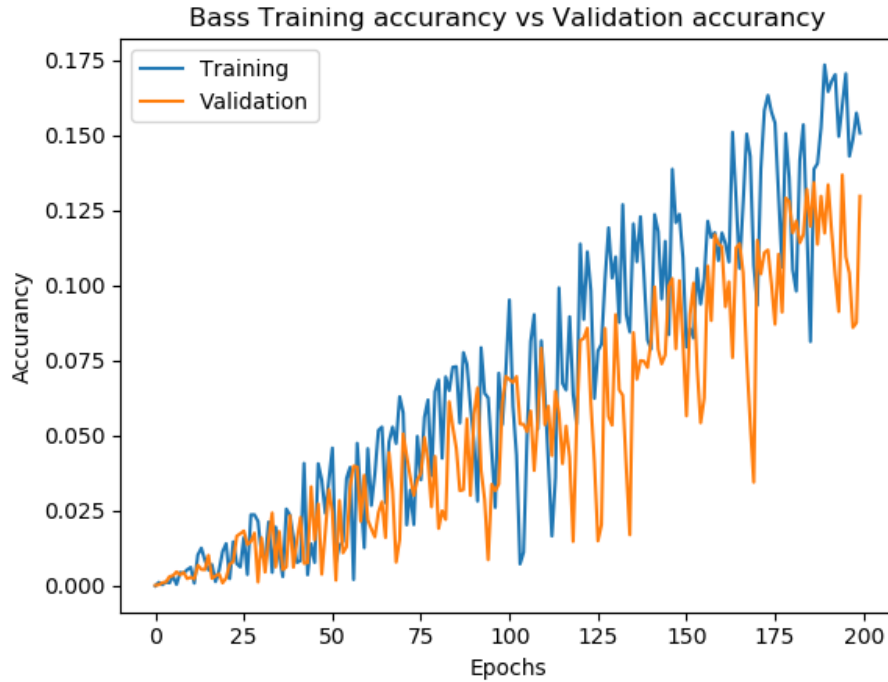


Figura 5.12: Gráfica de aprendizaje para bajo en arquitectura 3

En la Tabla 5.12 se observa que para la fase de entrenamiento se llegó a un valor final de 15.07 % y en la fase de validación a un 12.97 %, obteniendo máximos de hasta 17.34 % y 13.68 % respectivamente.

El rendimiento de esta red para bajo es menor al rendimiento para guitarra, sin embargo la tasa de aprendizaje es bastante parecida.

Es interesante ver que de la época 75 a la 135 la tasa de aprendizaje se mantiene oscilando entre los mismos valores, alcanzando por algunas épocas valores mayores.

Los valores obtenidos por esta red no son alentadores por lo que no podemos asegurarnos de que si entrenamos por más épocas esta red, sea capaz de alcanzar valores superiores al 70 %.

| Fase | Valor inicial | Valor final | Valor mínimo | Valor máximo |
|---------------|---------------|-------------|--------------|--------------|
| Entrenamiento | 0.0000 | 0.1507 | 0.0000 | 0.1734 |
| Validación | 0.0002 | 0.1297 | 0.0002 | 0.1368 |

Tabla 5.12: Valores de aprendizaje para bajo en arquitectura 3.

5.1.4. Comparativa de las arquitecturas

A continuación se presenta una comparativa de las diferentes arquitecturas, tomando en cuenta los valores descritos anteriormente.

5.1.4.1. Guitarra

En la tabla 5.13 vemos que en la fase de entrenamiento la arquitectura que mejor redujo el error fue la arquitectura 1 con un valor final de 1.3161, la arquitectura 2 dio un buen desempeño reduciendo el error hasta 1.8103 y finalmente, la arquitectura 3 presento el peor desempeño disminuyendo este valor.

| Arquitectura | Valor inicial | Valor final | Valor mínimo | Valor máximo |
|--------------|---------------|-------------|--------------|--------------|
| 1 | 6.6410 | 1.3161 | 1.3055 | 6.6410 |
| 2 | 6.1054 | 1.8103 | 1.4658 | 6.1138 |
| 3 | 11.8742 | 14.1156 | 11.8739 | 14.6645 |

Tabla 5.13: Comparativa del error en la fase de entrenamiento para guitarra

En la tabla 5.14 vemos que en la fase de validación la arquitectura que mejor redujo el error fue la arquitectura 1 con un valor final de 4.1383, la arquitectura 2 dio un buen desempeño reduciendo el error hasta 4.4766 y finalmente, la arquitectura 3 presento el peor desempeño disminuyendo este valor.

| Arquitectura | Valor inicial | Valor final | Valor mínimo | Valor máximo |
|--------------|---------------|-------------|--------------|--------------|
| 1 | 6.0144 | 4.1383 | 3.0009 | 6.0167 |
| 2 | 6.2114 | 4.4766 | 4.3697 | 6.2126 |
| 3 | 13.7430 | 15.4095 | 13.7430 | 15.7345 |

Tabla 5.14: Comparativa del error en la fase de validación de entrenamiento para guitarra

En la tabla 5.15 vemos que en la fase de entrenamiento la arquitectura que mejor aprendió fue la arquitectura 1 con un valor final de 56.66 %, seguido muy de cerca por la arquitectura 2 con un 55.58 % y finalmente, la arquitectura 3 presento el peor desempeño en este concepto.

| Arquitectura | Valor inicial | Valor final | Valor mínimo | Valor máximo |
|--------------|---------------|-------------|--------------|--------------|
| 1 | 0.0001 | 0.5666 | 0.0001 | 0.7226 |
| 2 | 0.0000 | 0.5558 | 0.0000 | 0.6555 |
| 3 | 0.0010 | 0.1702 | 0.0004 | 0.1964 |

Tabla 5.15: Comparativa del aprendizaje en la fase de entrenamiento para guitarra

En la tabla 5.16 vemos que en la fase de validación la arquitectura que mejor aprendió fue la arquitectura 1 con un valor final de 58.18 %, en segundo lugar tenemos a la arquitectura 2 con un 42.73 % y finalmente, la arquitectura 3 presento el peor desempeño en este concepto.

| Arquitectura | Valor inicial | Valor final | Valor mínimo | Valor máximo |
|--------------|---------------|-------------|--------------|--------------|
| 1 | 0.0001 | 0.5818 | 0.0001 | 0.5818 |
| 2 | 0.0007 | 0.4273 | 0.0007 | 0.4273 |
| 3 | 0.0006 | 0.0969 | 0.0002 | 0.1328 |

Tabla 5.16: Comparativa del aprendizaje en la fase de validación de entrenamiento para guitarra

5.1.4.2. Bajo

En la tabla 5.17 vemos que en la fase de entrenamiento la arquitectura que mejor redujo el error fue la arquitectura 2 con un valor final de 2.2061, la arquitectura 1 dio un buen desempeño reduciendo el error hasta 2.3281 y finalmente, la arquitectura 3 presento el peor desempeño disminuyendo este valor.

| Arquitectura | Valor inicial | Valor final | Valor mínimo | Valor máximo |
|--------------|---------------|-------------|--------------|--------------|
| 1 | 5.7464 | 2.3281 | 2.2614 | 5.7531 |
| 2 | 5.3796 | 2.2061 | 2.1150 | 5.3796 |
| 3 | 12.0105 | 14.4731 | 12.0105 | 14.5983 |

Tabla 5.17: Comparativa del error en la fase de entrenamiento para bajo

En la tabla 5.18 vemos que en la fase de validación la arquitectura que mejor redujo el error fue la arquitectura 1 con un valor final de 2.8547, la arquitectura 2

no redujo el error como era de esperarse quedando con un valor final de 4.2744 y finalmente, la arquitectura 3 presento el peor desempeño disminuyendo este valor.

| Arquitectura | Valor inicial | Valor final | Valor mínimo | Valor máximo |
|--------------|---------------|-------------|--------------|--------------|
| 1 | 6.0615 | 2.8547 | 2.8547 | 6.0620 |
| 2 | 6.4556 | 4.2744 | 3.8736 | 6.4556 |
| 3 | 16.112 | 17.1656 | 16.112 | 17.2226 |

Tabla 5.18: Comparativa del error en la fase de validación de entrenamiento para bajo

En la tabla 5.19 vemos que en la fase de entrenamiento la arquitectura que mejor aprendió fue la arquitectura 1 con un valor final de 74.08 %, seguido por la arquitectura 2 con un 61.70 % y finalmente, la arquitectura 3 presento el peor desempeño en este concepto.

| Arquitectura | Valor inicial | Valor final | Valor mínimo | Valor máximo |
|--------------|---------------|-------------|--------------|--------------|
| 1 | 0.0034 | 0.7408 | 0.0034 | 0.7408 |
| 2 | 0.0024 | 0.6170 | 0.0017 | 0.6426 |
| 3 | 0.0000 | 0.1507 | 0.0000 | 0.1734 |

Tabla 5.19: Comparativa del aprendizaje en la fase de entrenamiento para bajo

En la tabla 5.20 vemos que en la fase de validación la arquitectura que mejor aprendió fue la arquitectura 1 con un valor final de 52.28 %, en segundo lugar tenemos a la arquitectura 2 con un 29.88 % y finalmente, la arquitectura 3 presento el peor desempeño en este concepto.

| Arquitectura | Valor inicial | Valor final | Valor mínimo | Valor máximo |
|--------------|---------------|-------------|--------------|--------------|
| 1 | 0.0014 | 0.5228 | 0.0014 | 0.6170 |
| 2 | 0.0008 | 0.2988 | 0.0008 | 0.3683 |
| 3 | 0.0002 | 0.1297 | 0.0002 | 0.1368 |

Tabla 5.20: Comparativa del aprendizaje en la fase de validación de entrenamiento para bajo

5.2. Etapa de generación

En esta etapa se evaluara la salida de las redes desde un concepto musical.

5.2.1. Arquitectura 1

5.2.2. Arquitectura 2

5.2.3. Arquitectura 3

5.2.4. Comparativa de las arquitecturas

5.3. Etapa de validación

Durante esta etapa se evaluaron 100 canciones creadas usando las diferentes arquitecturas y aplicandoles el algoritmo de Krumhansl-Schmuckler para determinar la tonalidad final de las notas de cada instrumento.

Cada composición es hecha a partir de una cancion base, esta cancion posee una tonalidad ya establecida y la red debe de ser capaz de crear una nueva composición tomando como base las notas de esta cancion, la nueva pieza debería tener la misma tonalidad que la canción base, esto nos indica que nuestra red es capaz de identificar tonalidades y de crear nuevas cosas en base a estas.

La salida de cada instrumento es generado de manera independiente usando el Track correspondiente de la cancion base.

5.3.1. Arquitectura 1

Para la arquitectura 1 se obtuvieron los siguientes resultados:

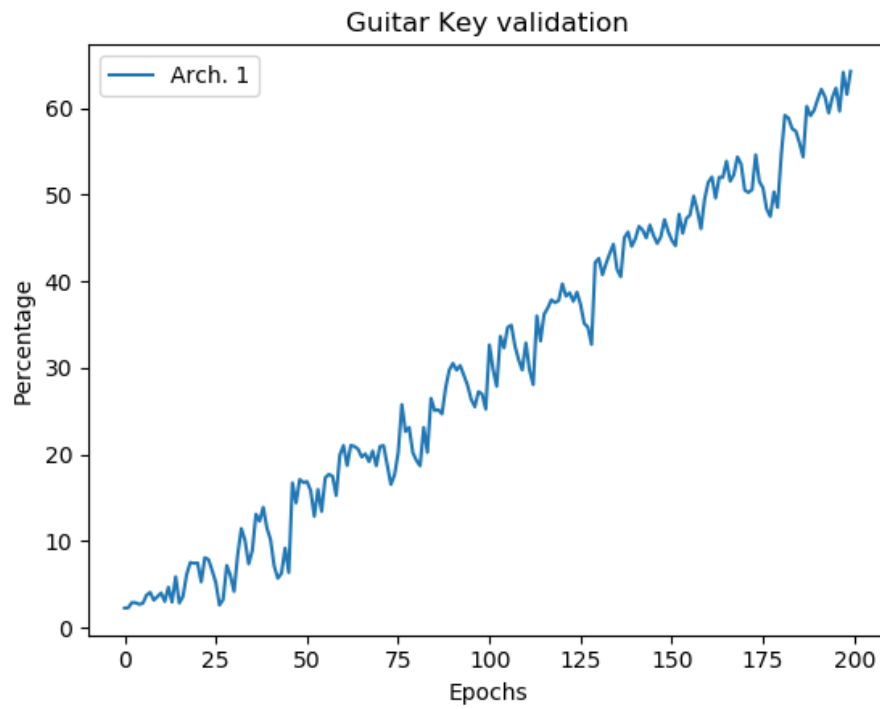


Figura 5.13: Gráfica de validación para guitarra en arquitectura 1

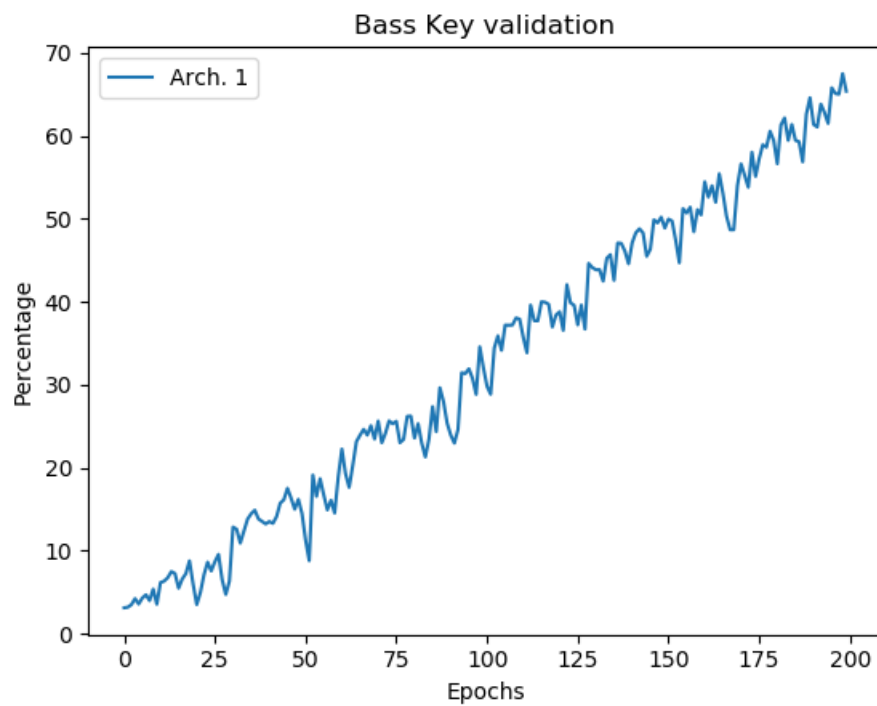


Figura 5.14: Gráfica de validación para bajo en arquitectura 1

En las figuras 5.13 y 5.14 se puede observar que conforme van pasando las épocas, el número de canciones creadas que poseen la misma tonalidad que la canción base va en aumento.

En la tabla 5.21 vemos que los valores alcanzados por esta arquitectura son de 64.26 % para guitarra y de 65.38 % para bajo, teniendo unos máximos de 64.26 % y 67.50 % respectivamente.

| Instrumento | Valor inicial | Valor final | Valor mínimo | Valor máximo |
|-------------|---------------|-------------|--------------|--------------|
| Guitarra | 2.2510 | 64.2632 | 2.2510 | 64.2632 |
| Bajo | 3.1204 | 65.3834 | 3.1204 | 67.5041 |

Tabla 5.21: Valores de validación para la arquitectura 1.

5.3.2. Arquitectura 2

Para la arquitectura 2 se obtuvieron los siguientes resultados:

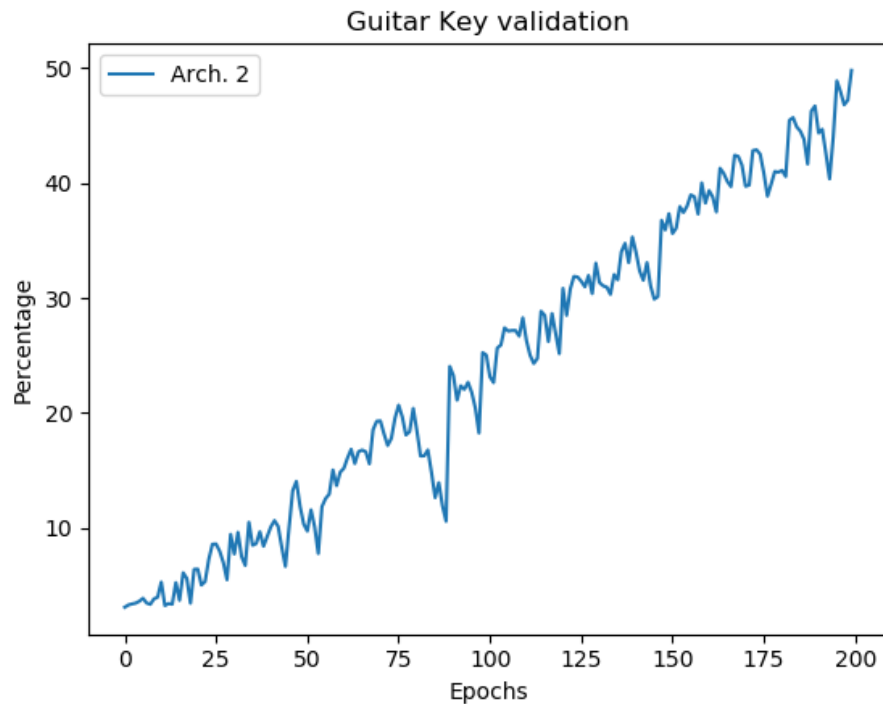


Figura 5.15: Gráfica de validación para guitarra en arquitectura 2

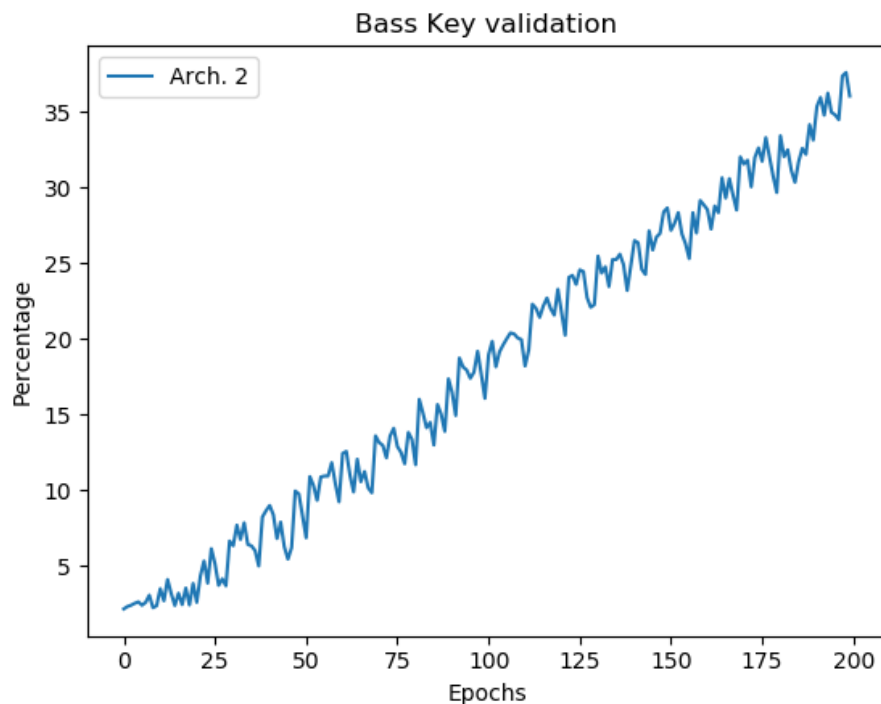


Figura 5.16: Gráfica de validación para bajo en arquitectura 2

En las figuras 5.15 y 5.16 se puede observar que conforme van pasando las épocas, el número de canciones creadas que poseen la misma tonalidad que la canción base va en aumento, sin embargo el crecimiento en las canciones de bajo es mucho más lento en comparación a la guitarra.

En la tabla 5.22 vemos que los valores alcanzados por esta arquitectura son de 49.81 % para guitarra y de 36.07 % para bajo, teniendo unos máximos de 49.81 % y 37.62 % respectivamente.

| Instrumento | Valor inicial | Valor final | Valor mínimo | Valor máximo |
|-------------|---------------|-------------|--------------|--------------|
| Guitarra | 3.1008 | 49.8122 | 3.1008 | 49.8122 |
| Bajo | 2.1270 | 36.0719 | 2.1270 | 37.6297 |

Tabla 5.22: Valores de validación para la arquitectura 2.

5.3.3. Arquitectura 3

Para la arquitectura 3 se obtuvieron los siguientes resultados:

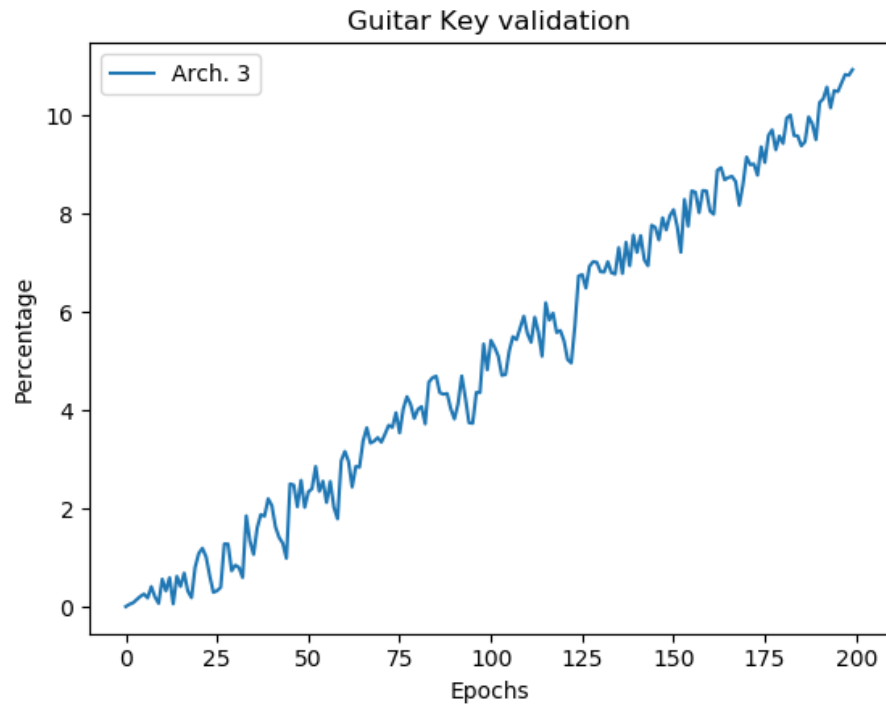


Figura 5.17: Gráfica de validación para guitarra en arquitectura 3

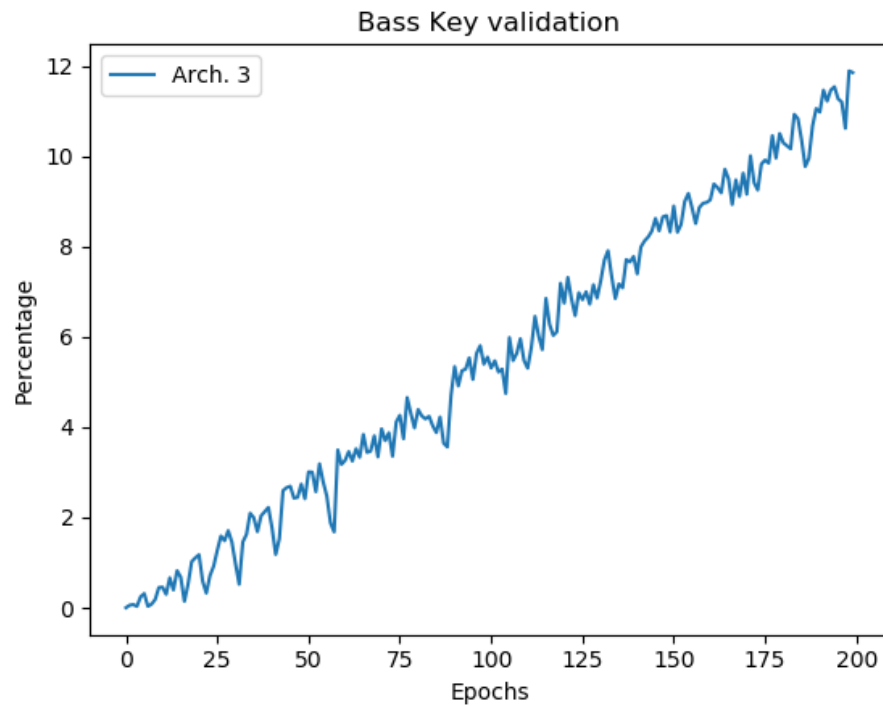


Figura 5.18: Gráfica de validación para bajo en arquitectura 3

En las figuras 5.17 y 5.18 se puede observar que conforme van pasando las épocas, el número de canciones creadas que poseen la misma tonalidad que la canción base va en aumento, sin embargo el porcentaje final es bastante bajo después de 200 épocas.

En la tabla 5.23 vemos que los valores alcanzados por esta arquitectura son de 10.94 % para guitarra y de 11.85 % para bajo, teniendo unos máximos de 10.94 % y 11.88 % respectivamente.

| Instrumento | Valor inicial | Valor final | Valor mínimo | Valor máximo |
|-------------|---------------|-------------|--------------|--------------|
| Guitarra | 0.0021 | 10.9422 | 0.0021 | 10.9422 |
| Bajo | 0.0022 | 11.8519 | 0.0022 | 11.8865 |

Tabla 5.23: Valores de validación para la arquitectura 3.

5.3.4. Comparativa de las arquitecturas

Podemos realizar una comparativa de estas arquitecturas si se grafican juntas.

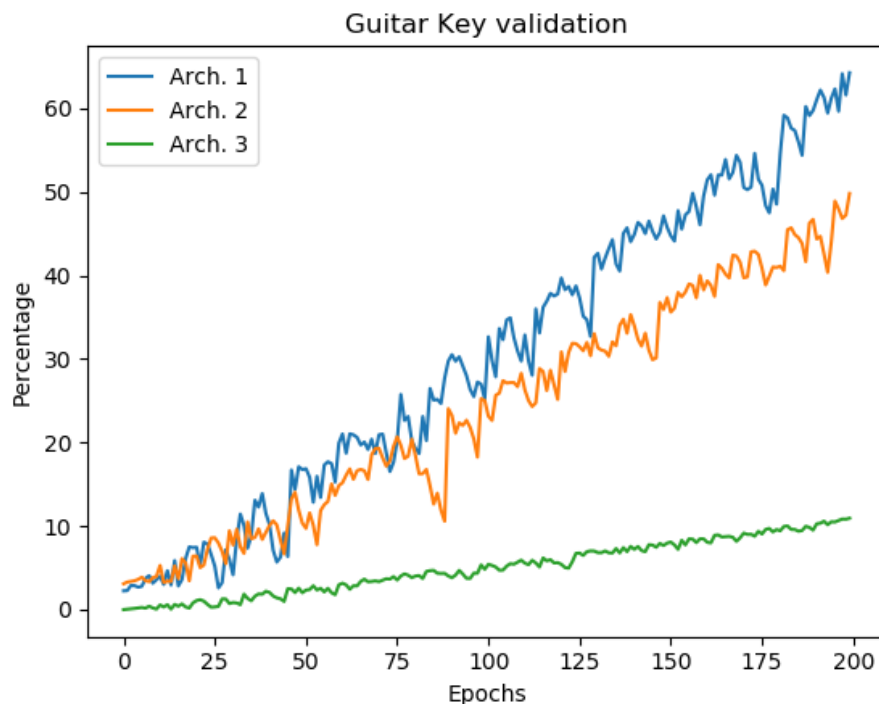


Figura 5.19: Gráfica comparativa de arquitecturas para guitarra

En la figura 5.19 observamos que la arquitectura que tuvo mejores resultados fue

la arquitectura 1, seguido de la arquitectura 2. El porcentaje de la arquitectura 3 es bastante bajo despues de todas estas epocas.

En la tabla 5.24 se observa que los valores alcanzados por cada arquitectura fueron de 64.26 %, 49.81 % y 10.94 % respectivamente.

| Arquitectura | Valor inicial | Valor final | Valor mínimo | Valor máximo |
|--------------|---------------|-------------|--------------|--------------|
| 1 | 2.2510 | 64.2632 | 2.2510 | 64.2632 |
| 2 | 3.1008 | 49.8122 | 3.1008 | 49.8122 |
| 3 | 0.0021 | 10.9422 | 0.0021 | 10.9422 |

Tabla 5.24: Comparativa de arquitecturas en la fase de validación para guitarra.

En la figura 5.20 se muestra el comportamiento que tuvieron las arquitecturas para bajo, siendo de nuevo la mejor la arquitectura 1.

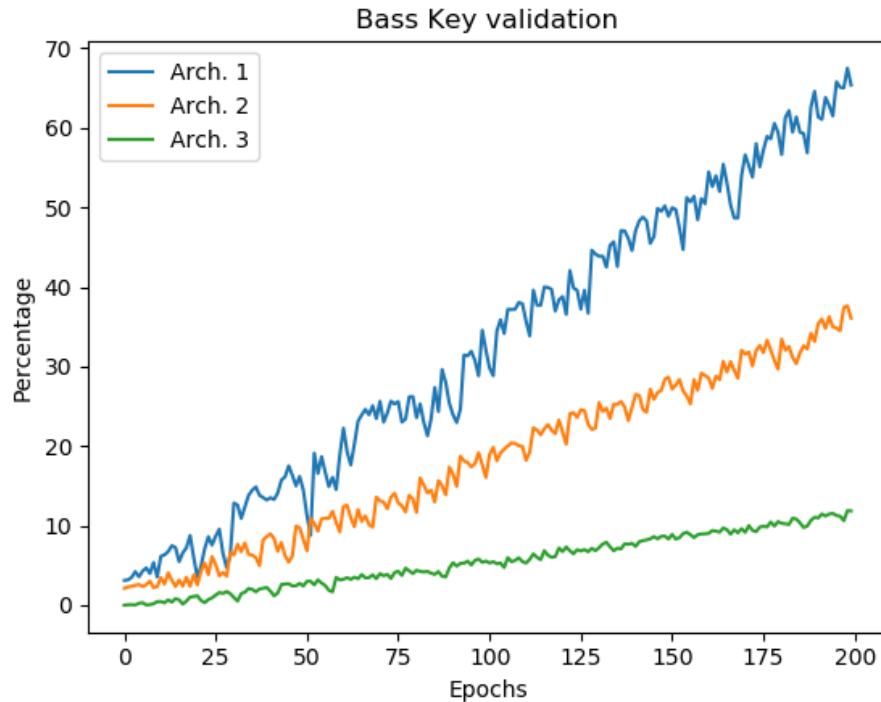


Figura 5.20: Gráfica comparativa de arquitecturas para bajo

En la tabla 5.25 se observa que los valores alcanzados por cada arquitectura fueron de 65.38 %, 36.07 % y 11.85 % respectivamente.

| Arquitectura | Valor inicial | Valor final | Valor mínimo | Valor máximo |
|--------------|---------------|-------------|--------------|--------------|
| 1 | 3.1204 | 65.3834 | 3.1204 | 67.5041 |
| 2 | 2.1270 | 36.0719 | 2.1270 | 37.6297 |
| 3 | 0.0022 | 11.8519 | 0.0022 | 11.8865 |

Tabla 5.25: Comparativa de arquitecturas en la fase de validación para bajo.

Capítulo 6

Conclusiones

Las diferentes arquitecturas presentadas en este documento fueron capaces de aprender y generar musica a partir de secuencias de notas de entrada. A pesar de que algunas arquitecturas no tuvieron el desempeño esperado, los resultados finales fueron bastante alentadores.

Estas redes fueron capaces de generar tanto melodías como armonías musicales en guitarra y melodías básicas de bajo.

Las secuencias de salida generadas por las redes poseen los 2 elementos básicos de la musica: sonidos y silencios. Sin embargo lo interesante de estas salidas es ver como están interactuando estos elementos para finalmente crear una nueva pieza musical.

Al parecer las secuencias de guitarra fueron mas sencillas de aprender que las de bajo, esto es un dato bastante curioso, puesto que en bajo no se tenían tantos cambios de tonos como en la guitarra.

Despues de todas las pruebas realizadas a las arquitecturas, la que tuvo el mejor desempeño fue sin duda la arquitectura 1, la cual cuenta con capaz intermedias de Dropout asi como tambien un optimizador RMSProp. Sin embargo la arquitectura 2 tambien presento resultados bastante alentadores.

Es interesante analizar que aunque se use un optimizador con momento como el Nadam, este no genera un resultado sobresaliente en redes LSTM, al parecer entre mas simple sea el optimizador mejores resultados se obtienen.

Las capas de Dropout fueron vitales para el correcto aprendizaje de estas redes, ya que sin ellas, el error iba aumentando en lugar de disminuir al transcurrir las épocas. Este comportamiento lo vemos mas claramente en la arquitectura 3.

Dentro de la búsqueda por encontrar un algoritmo computacional que nos permitiera realizar una validación de la salida, se encontró con el algoritmo de Krumhansl-Schmuckler, que a pesar, de ser bastante sencillo, nos genera muy buenos valores de

validación, lo que nos permitio dar certeza de que las piezas generadas por estas redes, estaban cumpliendo con lo establecido.

A pesar de que las redes se entrenaron por separado para guitarra y bajo, la salida que se obtiene al conjuntar la salida de ambos instrumentos se escucha bastante bien.

El tiempo de procesamiento es algo a tomar en cuenta cuando se trabajan con muchos datos y este tipo de redes, ya que computacionalmente hablando el procesamiento de estas redes es bastante costoso, y entre mas capas se esten usando en las redes, mas tiempo tarda por cada epoca de entrenamiento.

En este trabajo únicamente se realizaron 200 epocas de cada instrumento por cada una de las arquitecturas, ya que el tiempo promedio de procesamiento era de alrededor de 3 semanas por instrumento, lo que en conjunto de todas las pruebas realizadas se tomo cerca de 4 meses.

6.1. Trabajos futuros

Dentro de los trabajos futuros de esta investigación seria bueno experimentar con otro tipo de arquitecturas usando capas LSTM, para ver si se logran mejores resultados a los obtenidos en este trabajo.

La infinidad de posibilidades de nuevas arquitecturas solamente se limita a la imaginación del ser humano, por lo tanto no dudo que en un futuro se creen arquitecturas mas complejas para el procesamiento de secuencias. Posiblemente estas arquitecturas sean capaces de aprender y obtener buenos resultados en menor cantidad de epocas.

Conforme el tiempo pasa, el hardware también sufre modificaciones haciendo a los equipos mas potentes para procesar grandes cantidades de datos, es posible que en un futuro no muy lejano se puedan procesar estas redes en cuestión de horas, en lugar de dias o semanas, eso también nos abriría la posibilidad de introducir una mayor cantidad de datos o de inclusive usar arquitecturas mas robustas, sin importarnos tanto el tiempo de procesamiento.

Seria bueno en un trabajo futuro analizar estas mismas arquitecturas usando un algoritmo de validación de armonías musicales, para ver todo el campo armónico y dar una mayor certeza de que se estas redes están generando música de calidad. Este trabajo únicamente se centro en entrenar a las redes para que fueran capaces de aprender tonalidades y generar música a partir de ellas.

Es posible extrapolar este proyecto a otros instrumentos de cuerda o de viento,

tales como violin, saxofon, trompeta, entre otros. El metodo de aprendizaje seria muy similar, lo unico que se tendria que hacer es que en la etapa de preprocesamiento, separar los Tracks de estos instrumentos.

La batería y los instrumentos de percusión se manejan un poco diferente a los demas instrumentos, por lo que seria interesante ver si una red es capaz de realizar ritmos bases, los cuales puedan ser integrados a nuevas canciones, así como también ver si se generan nuevos ritmos diferentes a los que la red aprendió.

Es posible entrenar estas redes con otros estilos musicales, lo unico que se necesita es una base de datos bastante grande de canciones de estos generos, pero basicamente todas las demas etapas no sufririan cambios.

Bibliografía

- [1] C. Raffel, “Learning-based methods for comparing sequences, with applications to audio-to-midi alignment and matching,” Ph.D. dissertation, Columbia University, 2016.
- [2] M. H. Hassoun, *Fundamentals of Artificial Neural Networks*, 1st ed. Cambridge, MA, USA: MIT Press, 1995.
- [3] K. Gurney, *An Introduction to Neural Networks*. Bristol, PA, USA: Taylor & Francis, Inc., 1997.
- [4] I. Nunes, D. Hernane, R. Andrade, L. Bartocci, and S. dos Reis, *Artificial Neural Networks: A Practical Course*, Springer, Ed., 2017.
- [5] N. Buduma, *Fundamentals of Deep Learning: Designing Next-Generation Machine Intelligence Algorithms*, O. Reilly, Ed., 2017.
- [6] L. Deng and D. Yu, *Deep Learning: Methods and Applications*. Hanover, MA, USA: Now Publishers Inc., 2014.
- [7] J. Briot, G. Hadjeres, and F. Pachet, “Deep learning techniques for music generation - A survey,” *CoRR*, vol. abs/1709.01620, 2017. [Online]. Available: <http://arxiv.org/abs/1709.01620>
- [8] E. Herrera, *Teoría musical y armonía moderna*. Bosch, 1991. [Online]. Available: <https://books.google.com.mx/books?id=K5Qmp0GjJIEC>
- [9] E. Taylor and A. B. of the Royal Schools of Music (Great Britain), *The AB Guide to Music Theory*, ser. The AB Guide to Music Theory. Associated Board of the Royal Schools of Music, 1989, no. parte 1. [Online]. Available: <https://books.google.es/books?id=h7hZSAAACAAJ>

- [10] H. Lee, Y. Largman, P. Pham, and A. Y. Ng, “Unsupervised feature learning for audio classification using convolutional deep belief networks,” in *Proceedings of the 22Nd International Conference on Neural Information Processing Systems*, ser. NIPS’09. USA: Curran Associates Inc., 2009, pp. 1096–1104. [Online]. Available: <http://dl.acm.org/citation.cfm?id=2984093.2984217>
- [11] E. J. Humphrey, J. P. Bello, and Y. Lecun, “Feature learning and deep architectures: New directions for music informatics,” *J. Intell. Inf. Syst.*, vol. 41, no. 3, pp. 461–481, Dec. 2013. [Online]. Available: <http://dx.doi.org/10.1007/s10844-013-0248-5>
- [12] X. Wang and Y. Wang, “Improving content-based and hybrid music recommendation using deep learning,” in *Proceedings of the 22Nd ACM International Conference on Multimedia*, ser. MM ’14. New York, NY, USA: ACM, 2014, pp. 627–636. [Online]. Available: <http://doi.acm.org/10.1145/2647868.2654940>
- [13] J. Pons, O. Nieto, M. Prockup, E. M. Schmidt, A. F. Ehmann, and X. Serra, “End-to-end learning for music audio tagging at scale,” *CoRR*, vol. abs/1711.02520, 2017. [Online]. Available: <http://arxiv.org/abs/1711.02520>
- [14] A. Huang and R. Wu, “Deep learning for music,” *CoRR*, vol. abs/1606.04930, 2016. [Online]. Available: <http://arxiv.org/abs/1606.04930>