

Pontificia Universidad Javeriana Cali  
Facultad de Ingeniería.  
Ingeniería de Sistemas y Computación.  
Proyecto de Grado.

Detección de pulsaciones por minuto en tiempo real a partir de un  
Sistema de Inteligencia Artificial para el género musical Salsa

Danny Julian Murcia Gomez

Director: Dr. Gerardo Sarria

Fecha de Entrega: 01/02/2022





Santiago de Cali, Fecha de Entrega: 01/02/2022.

Señores

**Pontificia Universidad Javeriana Cali.**

Dr. Gerardo Sarria

Director Carrera de Ingeniería de Sistemas y Computación.  
Cali.

Cordial Saludo.

Por medio de la presente me permito informarle que el estudiante de Ingeniería de Sistemas y Computación Danny Julian Murcia Gomez (cod: 8853508) trabaja bajo mi dirección en el proyecto de grado titulado “Detección de pulsaciones por minuto en tiempo real a partir de un Sistema de Inteligencia Artificial para el género musical Salsa”.

Atentamente,

---

Dr. Gerardo Sarria

Santiago de Cali, Fecha de Entrega: 01/02/2022.

Señores

**Pontificia Universidad Javeriana Cali.**

Dr. Gerardo Sarria

Director Carrera de Ingeniería de Sistemas y Computación.  
Cali.

Cordial Saludo.

Me permito presentar a su consideración el anteproyecto de grado titulado “Detección de pulsaciones por minuto en tiempo real a partir de un Sistema de Inteligencia Artificial para el género musical Salsa” con el fin de cumplir con los requisitos exigidos por la Universidad para llevar a cabo el proyecto de grado y posteriormente optar al título de Ingeniero de Sistemas y Computación.

Al firmar aquí, doy fe que entiendo y conozco las directrices para la presentación de trabajos de grado de la Facultad de Ingeniería aprobadas el 26 de Noviembre de 2009, donde se establecen los plazos y normas para el desarrollo del anteproyecto y del trabajo de grado.

Atentamente,

---

Danny Julian Murcia Gomez  
Código: 8853508

# Resumen

La música es la combinación de un sin fin de posibilidades de ritmos que escuchamos todo el tiempo, donde constantemente nos ofrecen diferentes tipos de herramientas para experimentar, innovar y disfrutar de uno de los placeres más grandes que podemos encontrar. Por eso urge investigar modelos clave para la interpretación de los diferentes modelos de música que existen y sus diferentes. La Salsa es un género muy popular en América Central y del Sur, conocida por su variedad de instrumentos de percusión, por su piano y su clave de son [1]. Su ritmo es uno de los factores que depende de la clave de son, la cual puede ser 2-3 o 3-2 [2]. Este género musical es uno de los más interesantes para estudiar, en cuanto a saber a qué ritmo se refiere, en particular la determinación del beat, es decir, la unidad básica de ritmo de una canción y uno de los menos analizados en el campo científico.

Actualmente ya existen muchos patrones que detectan a partir de una canción el beat que maneja la canción, permitiendo controlar la canción. En el caso del género musical Salsa, no se han realizado muchos estudios en el análisis de obtener el beat en tiempo reproducción de la canción. Es requerido un Sistema de inteligencia Artificial que sea capaz de analizar una canción y retorne el beat en tiempo de reproducción de la canción.

**Palabras Clave:** Machine Learning, Música, Inteligencia Artificial, Detección de Pulsaciones por minuto.



# Índice general

<b>1. Descripción del Problema</b>	<b>11</b>
1.1. Planteamiento del Problema . . . . .	11
1.1.1. Formulación . . . . .	12
1.1.2. Sistematización . . . . .	12
1.2. Objetivos . . . . .	12
1.2.1. Objetivo General . . . . .	12
1.2.2. Objetivos Específicos . . . . .	12
1.3. Justificación . . . . .	13
1.4. Marco de Referencia . . . . .	14
1.4.1. Áreas Temáticas . . . . .	14
1.4.2. Marco Teórico . . . . .	14
1.4.3. Trabajos Relacionados . . . . .	17
<b>2. Desarrollo del Proyecto</b>	<b>19</b>
2.1. Conjunto de Datos . . . . .	19
2.1.1. Términos . . . . .	21
2.1.2. Funciones . . . . .	22
2.1.3. Procedimientos y Transformación de datos . . . . .	24
2.1.4. Explicación del uso del Rango de Hz . . . . .	26
2.2. Red Neuronal . . . . .	30
2.2.1. Entradas . . . . .	30
2.2.2. Salidas . . . . .	30
2.2.3. Arquitectura . . . . .	30
2.2.4. Modelos . . . . .	33
2.2.5. Entrenamiento . . . . .	40
2.2.6. Resultados . . . . .	42
2.3. Detección de BPM . . . . .	45
2.4. Interfaz de Usuario . . . . .	45
2.4.1. Encuesta . . . . .	45
2.4.2. Predicción de BPM . . . . .	49
<b>3. Conclusiones y trabajos futuros</b>	<b>55</b>
<b>Bibliografía</b>	<b>57</b>



# Introducción

En la actualidad plataformas como Spotify, iMusic, Deezer y otras registran en promedio más de 30 millones de canciones<sup>[3]</sup> y más de 100 millones de usuarios usando este servicio<sup>[2]</sup>. Al haber tal cantidad de datos en canciones, se busca hallar modelos que faciliten la interpretación de ondas de sonido a datos, para facilitar nuevos controles sobre la música, como son los modelos de estilo, análisis de armonía, similitud tímbrica, detección del beat, entre otros. El género musical salsa, es uno de los géneros musicales que presenta poco análisis en el campo científico por su gran variedad de ritmos caribeños que lo hacen uno de los géneros con mayor cantidad de combinaciones de géneros musicales.

La salsa se origina en New York en la década de 1960<sup>[4]</sup>, demostrando ser única por su singularidad de “clave de son”; consiste en 4 ritmos principales que se reparten en 2 compases al mismo tiempo, junto con un grupo de instrumentos incluyendo el tambor mambo, la trompeta y la guaracha<sup>[1]</sup>.

El BPM (Beat per Minute) es una unidad empleada para medir el ritmo en música; usado por científicos, productores y mezcladores de música que lo aprovechan para el análisis, reproducción o mezcla de la música<sup>[5]</sup>. Pero al género musical salsa al tener un ritmo tan complejo y variado, produce en los modelos de detección de pulso actuales, incertidumbre, siendo incapaces de detectar en tiempo real a que BPM (Beat per Minute) va el genero musical Salsa.

Por eso con respecto a la salsa urge la necesidad de realizar análisis computacionales que permitan entender mejor las características de la canción del género en tiempo real para el uso investigativo y comercial. También nos permite estudiar el comportamiento de la salsa como proceso investigativo que servirá en el desarrollo de nuevos productos con respecto a la Salsa.



## CAPÍTULO 1

# Descripción del Problema

---

### 1.1. Planteamiento del Problema

La mayoría de modelos que se utilizan en el área de la música para detectar el pulso son entrenados con música de ritmo sencillo, por ejemplo: pop, rock, electrónica, clásica, entre otras. A diferencia de estos ritmos la salsa cuenta con un ritmo complejo y difícil[1]; causando que la mayoría de algoritmos actuales presente incertidumbre frente a canciones complejas, en este caso la Salsa.

Estos modelos de aprendizaje no contemplan para su entrenamiento el uso de la salsa, dejando unas predicciones muy erradas con respecto a este género a la hora de cualquier predicción que se quiera realizar. Por eso la necesidad de investigar a partir de una base de datos conformada solo por salsa, investigar todos los diferentes modelos que ya existen, pero aplicados a la salsa únicamente.

El mercado cada vez crece más; hay más canciones y más usuarios, haciendo que se requieran nuevas comodidades e instrumentos para mejorar la experiencia de la persona a la hora de interactuar con la música (escucharla, mezclarla, investigarla, etc). Realizar estudios a la música crea nuevas oportunidades de interactuar con la música, permitiendo conocerla, analizarla y disfrutarla.

Este proyecto hace parte de uno más grande llamado “Deep salsa” que involucra investigadores de la universidad Javeriana de Cali, Universidad Icesi y la Universidad de Aizo, en Japón. Se continúa el proyecto de caracterización computacional de la salsa introduciendo técnicas de inteligencia artificial.

### **1.1.1. Formulación**

Se realizan diferentes tipos de aprendizaje automático para la detección del beat de una canción, pero muestra resultados únicamente con canciones simples. Se requiere una inteligencia artificial capaz de detectar el beat para canciones únicamente de salsa, pero ¿Cómo construir y entrenar un sistema de inteligencia artificial capaz de detectar el beat de canciones de salsa en tiempo de reproducción?

### **1.1.2. Sistematización**

- ¿Cómo determinar la validez de los datos?
- ¿Cómo aplicar métodos de inteligencia artificial en el análisis de las canciones?
- ¿Cómo determinar si el modelo cumplió su finalidad?
- ¿Cómo interactuar con el producto final?

## **1.2. Objetivos**

### **1.2.1. Objetivo General**

Desarrollar un prototipo de inteligencia artificial que sea capaz de detectar el beat en tiempo real de un conjunto arbitrario de únicamente canciones del género musical salsa.

### **1.2.2. Objetivos Específicos**

- Analizar del conjunto de datos de la base de datos que se usará para el entrenamiento de la inteligencia artificial, su consistencia, errores, valores duplicados y valores nulos.
- Estudiar las ventajas y desventajas de las técnicas de aprendizaje a usar para el entrenamiento de la inteligencia artificial.
- Analizar y estudiar técnicas que se usarán para el entrenamiento del sistema de Inteligencia Artificial.
- Evaluar el desempeño del entrenamiento del sistema de inteligencia artificial.
- Crear una interfaz visual limpia e intuitiva para el uso del sistema de inteligencia artificial.

### 1.3. Justificación

La música es uno de los mayores entretenimientos de las personas en la actualidad, usada para relajarse, divertirse, recordar, etc. Definida como un arte de combinar diferentes sonidos donde produzcan deleite, conmoviendo la sensibilidad, ya sea alegre o tristemente [6].

Actualmente la música se encuentra en su mayor etapa de globalización, llegando a todos los rincones gracias al acceso a internet. La industria cada vez se preocupa más por dar nuevas funciones y experiencias a los usuarios que mejoren el uso de su aplicación con el fin de retener a la persona de que use su servicio a partir de una suscripción.

La Salsa viene siendo objeto de estudio con los demás géneros musicales, esto con el fin de conocer e investigar sus características; aun así, la variedad de estudios que se realizan a la música no lo incluye mucho. Genera preocupación la escasez de estudio con respecto al género musical Salsa e incentiva a aportar nuevas investigaciones para el desarrollo de nuevas aplicaciones y conocimientos a cerca de la ella.

Este proyecto viene siendo parte de la comunidad de investigación de la universidad Javeriana de Cali, la universidad ICESI y la universidad Aizu llamado “Deep Salsa”. Busca incentivar a estudiantes a realizar investigaciones que giren alrededor de la salsa. Al haber en el campo de investigación tantas modelos de detección de beat y que sean ineficientes con el género musical salsa dio causa a un punto de partida donde debía ser analizada la detección de beat exclusivamente para el género musical salsa [1].

## 1.4. Marco de Referencia

### 1.4.1. Áreas Temáticas

- Computing Methodologies - Artificial Intelligence - Distributed artificial Intelligence.
- Computing methodologies - Machine learning - Learning paradigms - Supervised learning.
- Applied Computing - Arts and Humanities - sound and music computing.

### 1.4.2. Marco Teórico

El uso de detección del beat y el seguimiento del “downbeat” sigue siendo al día de hoy uno de los campos más complejos en la investigación de la música-audio. La detección del beat viene siendo hallada de varias maneras siendo resumido en un problema de estimación periódica y un problema de localización al principio del periodo dentro de la señal. De igual forma el “Downbeat” es principalmente una noción perceptiva que viene del proceso de construcción musical. [7]

Dentro de la investigación acerca de la detección del beat el “tempo” juega un papel importante, es el encargado de estimar el patrón básico de un beat en un pedazo de música o a la velocidad que debe ejecutarse la canción[8]. El tempo puede ser analizado en 3 diferentes escalas de tiempo:

- Tatum: que se refiere al nivel de pulso atómico.
- Tactus: que corresponde al tempo de una pieza (o a 4 tatum).
- Nivel de compás musical (o a 4 tactum).

Pero este proceso presenta una dificultad donde recae en la decisión del periodo de la estimación del tatum que es ocasionado por el pulso atómico, el cual no se presenta todo el tiempo, sino, ocasionalmente[9]. El lograr determinar el tactum y tatum como un patrón dentro de una canción nos da como resultado el compás o tactus que es el beat de la canción (BPM)[10]. Estos cálculos se hacen con el propósito de indicar la duración y velocidad de los diferentes símbolos musicales con exactitud. Por ejemplo si el tempo global de una canción es de 60, quiere indicar que en 1 minuto hay 60 golpes o donde una negra será equivalente a 1 golpe por segundo.

Usando las anteriores escalas de tiempo musical que corresponde al tempo de la canción y el nivel de medida musical, como los parámetros de busca del modelo de aprendizaje, se puede hacer entrenamientos de Inteligencias artificiales que serán capaces de hacer acercamientos de la detección del beat de acuerdo a una representación de inicio discreto extraído de la señal de audio o función de inicio de valor continuo. Los siguientes modelos se basan en estos acercamientos: [7]

- Goto propone un método para detectar cambios de acordes analizando el espectro de frecuencia cortado en tiempos de tiempo provisionales, esto supone una primera teoría de lo que sería la percepción de la música como símbolos musicales. Usan multi-agentes donde cada agente

hace una suposición del periodo de tiempo y fase de tiempo; un gerente al final determina el mejor agente. [10]

- Dixon funciona de manera similar a Goto [10], usa múltiples agentes que simultáneamente considera varias hipótesis diferentes con respecto a la frecuencia y la ubicación de los ritmos musicales, lo que resulta en un seguimiento preciso del ritmo y una recuperación rápida de los errores. [11]
- Modelos como Scheirer usan pequeños filtros de “paso de banda” y bancos de filtros de peine en paralelo, para analizar el tempo y extraer el ritmo de las señales musicales de complejidad polifónica arbitraria y que contienen timbres arbitrarios. [12]
- Jehan [13] parecido a Scheirer[12] usan filtros de peines resonantes donde cada estado provisoria directamente la fase, que viene siendo la información del beat.
- Otros modelos como Klapuri extiende estos métodos donde para el análisis inicial de frecuencia de tiempo, se propone una nueva técnica que mide el grado de acento musical en función del tiempo en cuatro rangos de frecuencia diferentes. Seguido a eso usa los filtro de peine que realizan extracción por función para estimar los períodos y fases de los tres pulsos. Una vez obtenido esto, las características se procesan mediante el uso de agentes que determinan una estimación conjunta del pulso y el tactus. Usando estados como parámetros de entrada de un modelo oculto de Markov el cual tiene una evolución de la fase de seguimiento. [9]

La Salsa tiene una amplia variedad de instrumentos, por ello, cuando se obtiene varios instrumentos es recomendable para la detección del beat el uso de: múltiples agentes para comparar múltiples hipótesis de beats, para que la Inteligencia Artificial sea capaz de tomar una decisión dependiendo del contexto a partir de los patrones del tambor; los procesos se deben realizar de acuerdo a la confiabilidad de los eventos e hipótesis detectados (es imposible manejar señales complejas sin errores); y los parámetros de análisis de frecuencia se deben de ajustar dinámicamente mediante la interacción entre el procesamiento de bajo nivel y alto nivel. [14]

La salsa contiene ritmos bastante complejos por eso para su análisis de entrenamiento es recomendable la construcción de 2 capas, una capa inferior de la extracción de beats perceptivos que encuentra el nivel en el que el pulso se divide uniformemente con el tiempo; después un modelo de agrupación de nivel superior se debe encargar de procesar y seleccionar los beats para formar un orden jerárquico de las señales dependiendo de la detección de reconocimiento de patrones de estructuras de acento (posibles beats en el tiempo) y patrones de beats instrumentales. [12]

Para el análisis de todas las características de la detección del beat se debe de hacer uso de una base datos para el entrenamiento y prueba de la Inteligencia artificial, por lo cual el conjunto de datos de “salón de baile” (ballroom) es importante. Gracias a su gran variedad de canciones se combina con que cada una de ellas maneja diferentes tonalidades, instrumentos y tempo. Su uso en redes neuronales profundas ayuda a que pueda usar mayor cantidad de datos para su entrenamiento. [15]

Estudios realizados por la universidad Icesi lograron crear una base de datos con los beats debidamente anotados de varias canciones de Salsa [1], esto ayuda a la extensión de la base de datos de entrenamiento y de prueba que usará la Inteligencia artificial. Con el uso de estos recursos es posible generar exclusivamente para el género musical Salsa un detector de beat.

El aprendizaje que usará la Inteligencia artificial estará sujeta a los siguiente modelos:

- Aprendizaje supervisado: Su entrada es un conjunto de vectores de características, donde cada uno corresponde a los atributos que pertenecen a un objeto y se espera entregar un modelo etiquetado. El conjunto de datos se debe preparar previamente etiquetando el resultado que se espera del algoritmo y se harán comparaciones entre el resultado esperado y el obtenido para mejorar el modelo. [16]
- Aprendizaje no supervisado: Su entrada es un conjunto de vectores de características, pero no posee información del resultado esperado. Este tipo de aprendizaje permite la recolección de información de patrones que no son fácilmente visibles en el conjunto de datos. [16]
- Aprendizaje semi-supervisado: Su entrada es un conjunto de vectores con y sin etiquetas. Generalmente la cantidad de datos sin etiquetas es mucho mayor a las datos etiquetados y se incluyen datos no etiquetados con el propósito de aportar mayor información al problema a solucionar. [17]
- Aprendizaje reforzado: Este aprendizaje, el algoritmo existe en un ambiente y es capaz de percibir sus estados como un vector de características. La máquina puede ejecutar instrucciones cuyo impacto es medido en recompensas. Su uso en general es para aprender políticas, que se resume en la entrega de una acción que se ejecuta en un estado dado. Esta acción debe de ser óptima, es decir, buscar siempre maximizar la recompensa obtenida. [17]

Cada modelo de Inteligencia artificial se compone de un perceptrón multicapa, esto quiere decir que está conformado por varias capas. En general se componen de 3 capas generales que son:

- Capa de entrada: Neuronas de entrada.
- Capa Oculta o intermedia: Neuronas que tiene conexiones anteriores y posteriores.
- Capa de salida: Neuronas que tienen conexiones anteriores y no tienen conexiones posteriores, son las neuronas de salida.

La evaluación del entrenamiento estará sujeta a dos de las técnicas más comunes, holdout y cross-validation.

- Holdout: Su uso tiene como propósito que sea imparcial el desempeño. Usualmente el conjunto es dividido en 3 partes, siendo el conjunto de entrenamiento, pruebas y validación. El conjunto de validación se usa para validar la correctitud del modelo construido a partir del conjunto de entrenamiento y el modelo resultante del conjunto de validación se usa para mirar el desempeño que tendrá el modelo. [18]

- Cross-validation: Su uso tiene como propósito dividir el conjunto en 2 partes, donde uno es para las pruebas/validaciones y otra para el entrenamiento. Su técnica más usada trata de usar un valor K en el que subdivide el conjunto de datos, después uno de estos subconjuntos es utilizado para la validación y el resto se usa para el entrenamiento de la inteligencia artificial. Esto se repite K veces y el error estimado es hallada como el promedio de los errores de cada prueba. [18]

#### 1.4.3. Trabajos Relacionados

- Se ha demostrado con redes neuronales convolucionales que es posible lograr más de un 99 % de precisión al hallar el beat de una canción de salsa si se escucha por completo. Lo realiza con un modelo secuencial de 6 capas. Solo detecta el beat una vez analizado toda la canción, es incapaz de hacerlo en tiempo real. [1]
- BeatRoot es un sistema interactivo de seguimiento de ritmo y anotación de métricas que se ha utilizado durante varios años en estudios de sincronización de rendimiento. Utiliza una arquitectura de múltiples agentes que simultáneamente considera varias hipótesis diferentes con respecto a la frecuencia y la ubicación de los ritmos musicales, lo que resulta en un seguimiento preciso del ritmo, una recuperación rápida de los errores y una degradación elegante en los casos en que el ritmo sólo está débilmente implicado por los datos. [11]
- Un sistema de detección de beat en tiempo real creado por Mastaka Goto y Yoichi Muraoka que detecta una estructura jerárquica de ritmo en señales de audio musicales sin sonidos de una batería. Se propone un método que permite al sistema rastrear beats a diferentes niveles rítmicos y seleccionar la mejor de varias hipótesis sobre las posiciones de los beats. [10]
- Extracción automática de tempo y ritmo de interpretaciones expresivas. Un programa de computador capaz de estimar el tempo y los tiempos de ritmos musicales en música expresada. Los datos son procesados fuera de línea para detectar los eventos rítmicos más destacados y se analiza el tiempo de estos eventos para generar hipótesis del tempo en varios niveles métricos. Con base en estas hipótesis de tempo, una búsqueda de hipótesis múltiples encuentra la secuencia de tiempos de latido que mejor se adapta a los eventos rítmicos. [19]
- Método propuesto por Geoffroy Peeters, tiene como objetivo detectar el tempo de la música con y sin percusión. Primero se propone una función de energía de inicio basada en el flujo de energía espectral reasignado. Se utiliza una combinación de la Transformación discreta de Fourier y la Función de correlación automática de frecuencias para estimar las periodicidades dominantes en cada momento. Luego se utiliza un algoritmo de Viterbi para detectar el tempo más probable y las rutas de subdivisión de medidor / tiempo a lo largo del tiempo. [20]
- Enfoques basados en datos para la estimación de tempo y clave de grabaciones musicales por Hendrik Schreiber proponen una red neuronal convolucional usando aprendizaje supervisado para hallar el tempo y BPM. Usan diferentes librerías y diferentes tipos de datos, evalúan su impacto y su rendimiento. Dan una amplia gama de diferentes tipos de transformación

y análisis de los datos. Su objetivo es analizar lo que es la percepción del Tempo para las inteligencias artificiales. [21]

- Aprendizaje profundo para el procesamiento de señales de audio. Artículo dedicado a enseñar las diferentes técnicas de aprendizaje profundo para el procesamiento de señales de audio. Desde el análisis de los datos, las diferentes transformaciones, el procesamiento y creación de modelos, hasta las diferentes formas de clasificación que podemos encontrar. [22]
- Seguimiento de ritmo específico de estilo con redes neuronales profundas por Julius Richter, el propone un método computacional para extraer las posiciones de los tiempos de las señales del audio. Usando redes convolucionales temporales que capturan la estructura secuencial de la entrada de audio. Necesita menos parámetros de aprendizaje aumentando la eficiencia computacional. [23]

## CAPÍTULO 2

# Desarrollo del Proyecto

---

El proyecto se separa en 5 partes:

1. Conjunto de datos. [2.1](#)
2. Investigación de los datos. [2.1.4](#)
3. Modelos de redes neuronales convolucionales. [2.2](#)
4. Entrenamiento y resultados. [2.2.6](#)
5. Interfaz. [2.4](#)

Queremos encontrar en el género musical salsa patrones en las canciones para entrenar una red neuronal que sea capaz de clasificar el BPM de una canción entre un rango de 60 BPM hasta 220 BPM y crear una vista amigable para un usuario final que pueda hacer uso de la Inteligencia artificial de una manera rápida, fácil y sencilla.

Este proyecto seguirá el siguiente orden: se comenzará explicando los datos desde su obtención hasta las herramientas usadas para la extracción y transformación de ello; después se explicará el análisis e investigación que se hizo respecto a los datos y porque fueron escogidos como los datos de entrada de la red neuronal; explicado ambos temas procedemos con el análisis de los modelos de redes neuronales usados en el proyecto, explicando sus entradas, salidas, los términos, las librerías usadas y cada arquitectura usada en los modelos. Seguimos con el entrenamiento de cada modelo de red neuronal, explicando sus resultados y su expectativa.

Finalmente explicamos cómo se creó la interfaz gráfica para la red neuronal, toda su construcción, las librerías usadas y el paso a paso de un usuario final.

## 2.1. Conjunto de Datos

El conjunto de datos se recolectó de manera manual, de una base de datos de colección privada de 1119 canciones de salsa y de la base de datos Extended-BIGROOMDATASET que contiene 47 canciones de salsa, para un total de 185730 minutos de canciones únicamente de Salsa (Imagen: [2.5](#))

La distribución de los datos fueron los siguiente:

- Conjunto de datos de entrenamiento. 2.1
- Conjunto de datos de validación. 2.2
- Conjunto de datos de prueba. 2.3
- Conjunto de datos de Extended-BIGROOM. 2.4

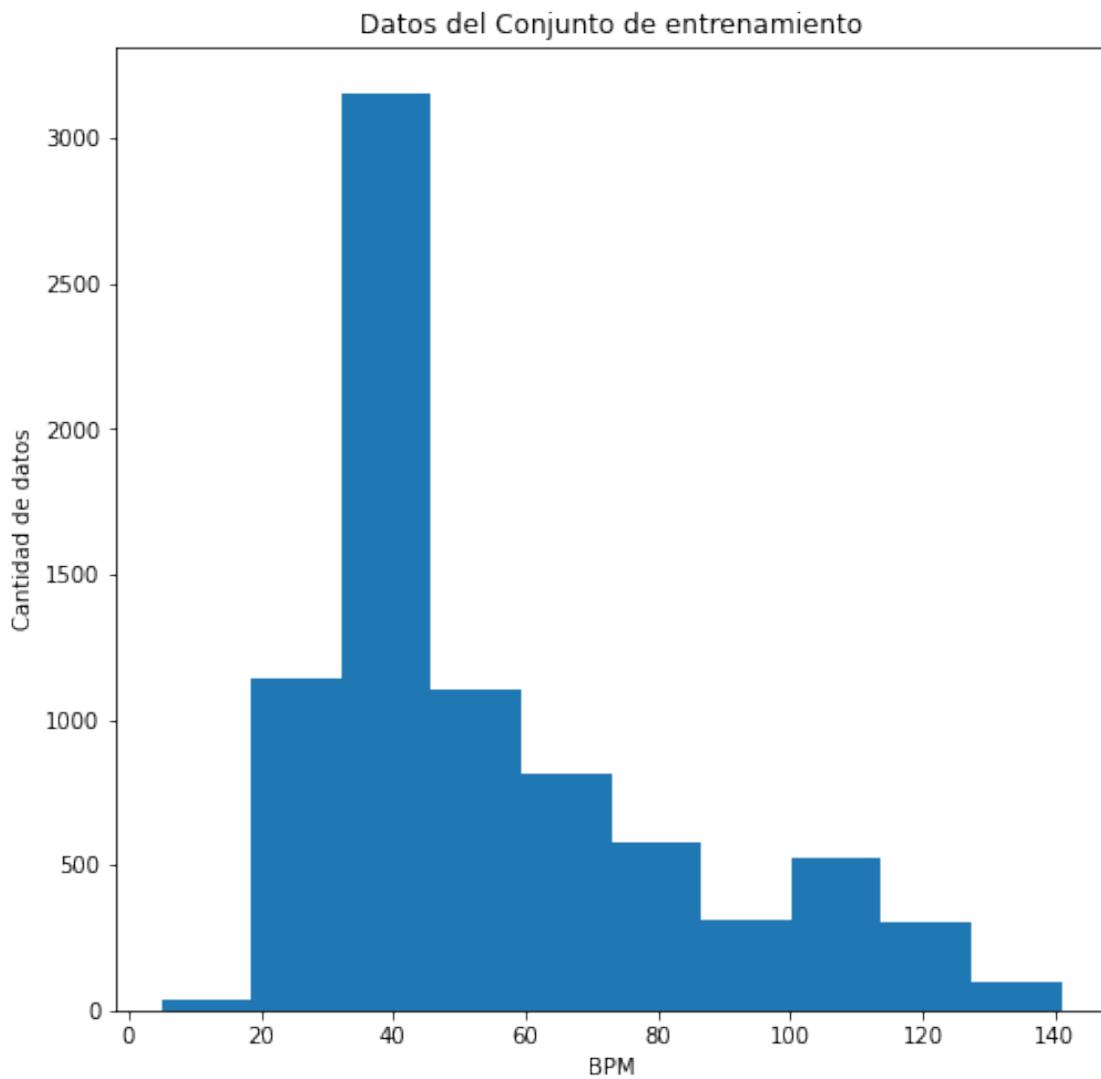


Figura 2.1: Conjunto de datos de entrenamiento.

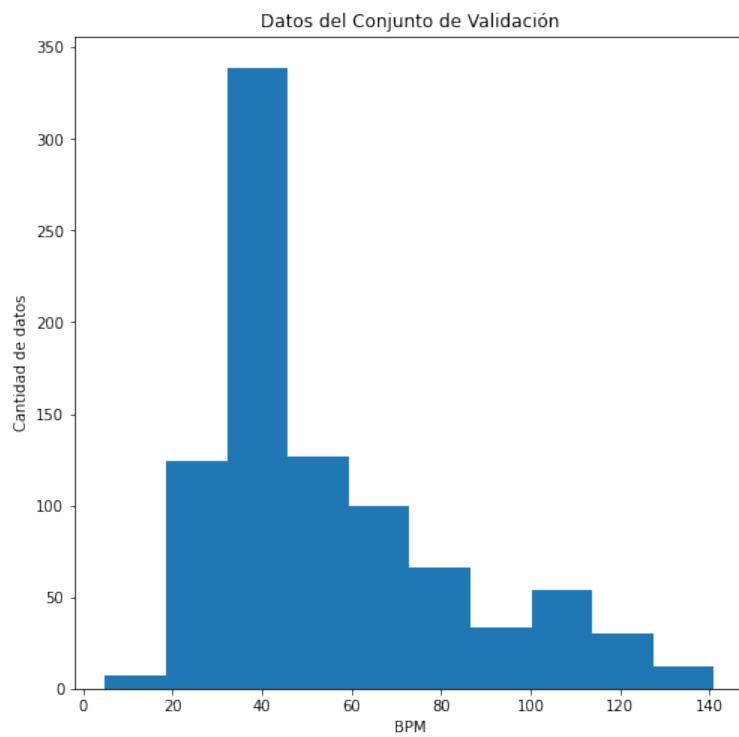


Figura 2.2: Conjunto de datos de validación.

Los datos antes de ser parte de la entrada de la red neuronal tiene que pasar por una serie de transformaciones.

Para las transformaciones de estos datos usamos Python 3 como lenguaje de programación con ayuda de las siguientes librerías:

- Librossa: Librería especializada para procesamiento del audio. [24]
- Numpy: Librería especializada en el manejo de datos. [25]
- Matplotlib: Librería especializada en generar gráficos de datos. [26]

La librería Librossa cuenta con diferentes funciones y términos, se explicará cada una de las usadas en el proyecto.

### 2.1.1. Términos

- Decibelio (dB): Es una unidad de medida que se usa para expresar la relación entre 2 valores de presión sonora y potencia eléctrica. Esta unidad no es lineal, sino logarítmica, adimensional y matemáticamente escalar. [27]

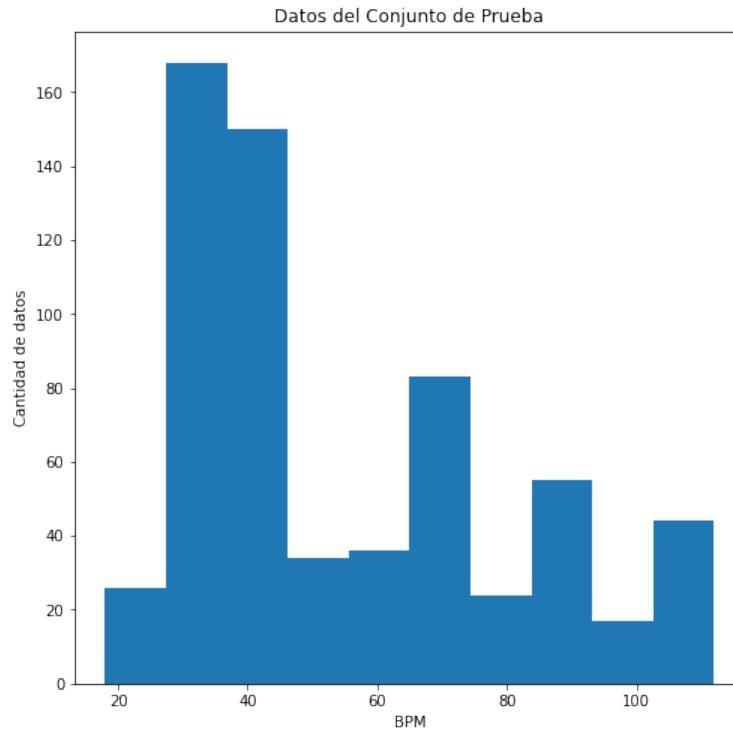


Figura 2.3: Conjunto de datos de prueba.

- Hercio (Hz): Es una unidad de medida de frecuencia. Se entiende por ella como el suceso que se repite por segundo. [28]
- Bin: Es el resultado de la transformación de frecuencias continuas usando la STFT. [24]
- Escala Mel: En el contexto matemático, la escala de Mel es “un resultado de alguna transformación no lineal de la escala de frecuencia”. Esta escala intenta simular una percepción humana de los tonos; de ahí su nombre mel que deriva de la palabra melodía. [29]

### 2.1.2. Funciones

- STFT (Short-time Fourier Transform): Las señales de música son muy cambiantes en el tiempo y estas señales se capturan como ondas de presión de sonido. La transformada de fourier nos ayuda con la transformación de una función de tiempo-dominio a una representación en el dominio de la frecuencia. Esto quiere decir que convertimos una señal del dominio del tiempo en una señal al dominio de la frecuencia. La transformada de fourier en corto tiempo se calcula dado una función de ventana (“Window function”) con saltos específicos para darnos los valores del dominio de frecuencia por cada “salto”.

Esta función se diferencia de la transformado de fourier normal porque “STFT proporciona la

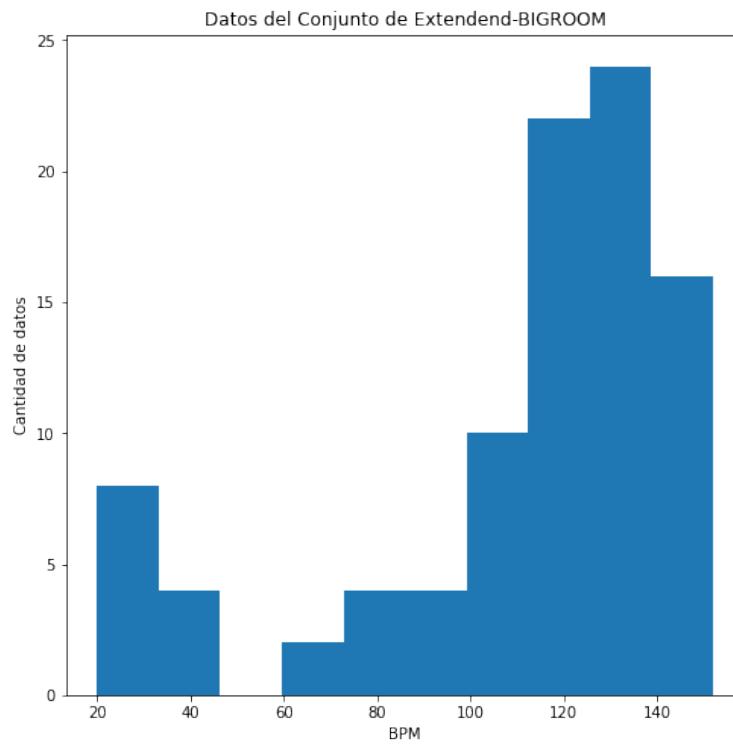


Figura 2.4: Conjunto de datos de Extended-BIGROOM.

información de frecuencia localizada en el tiempo para situaciones en las que los componentes de frecuencia de una señal varían con el tiempo, mientras que la transformada de Fourier estándar proporciona la información de frecuencia promediada durante todo el intervalo de tiempo de la señal” [30]. [24]

- Amplitud\_to\_Db: Función que convierte un espectrograma de amplitud a un espectrograma de decibeles. [24]
- Melspectrogram: Función encargada de dividir la escala de Hercios en escala de bins y cada bin es transformado en la escala de Mel. Genera a partir de una STFT y una escala mel; una matriz que es una percepción de lo que escuchamos.

Esto nos genera señales no periódicas que necesitan ser representadas en un espectro de señales en el tiempo, para eso generaremos un espectrograma que simula la percepción de lo que escuchamos de un sonido. [29] [31] [24]

- Power\_to\_Db: Función que convierte un espectrograma de potencia en unidades de decibelios (dB). [24]

```

EXTENDED = [
    {"index": 0, "nombre": "108416.mp3", "duracion": 30, "tiempo": 0, "bpm": 90},
    {"index": 1, "nombre": "108737.mp3", "duracion": 30, "tiempo": 0, "bpm": 90},
    {"index": 2, "nombre": "108736.mp3", "duracion": 30, "tiempo": 0, "bpm": 100},
    {"index": 3, "nombre": "108738.mp3", "duracion": 30, "tiempo": 0, "bpm": 98},
    {"index": 4, "nombre": "108838.mp3", "duracion": 30, "tiempo": 0, "bpm": 192},
    {"index": 5, "nombre": "111931.mp3", "duracion": 30, "tiempo": 0, "bpm": 184},
    {"index": 6, "nombre": "111933.mp3", "duracion": 30, "tiempo": 0, "bpm": 200},
    {"index": 7, "nombre": "111932.mp3", "duracion": 30, "tiempo": 0, "bpm": 188},
    {"index": 8, "nombre": "112535.mp3", "duracion": 30, "tiempo": 0, "bpm": 80},
    {"index": 9, "nombre": "112536.mp3", "duracion": 30, "tiempo": 0, "bpm": 83},
    {"index": 10, "nombre": "115501.mp3", "duracion": 30, "tiempo": 0, "bpm": 212},
    {"index": 11, "nombre": "116601.mp3", "duracion": 30, "tiempo": 0, "bpm": 202},
    {"index": 12, "nombre": "116603.mp3", "duracion": 30, "tiempo": 0, "bpm": 192},
    {"index": 13, "nombre": "116605.mp3", "duracion": 30, "tiempo": 0, "bpm": 180},
    {"index": 14, "nombre": "116607.mp3", "duracion": 30, "tiempo": 0, "bpm": 168},
    {"index": 15, "nombre": "116609.mp3", "duracion": 30, "tiempo": 0, "bpm": 168},
    {"index": 16, "nombre": "116611.mp3", "duracion": 30, "tiempo": 0, "bpm": 156},
    {"index": 17, "nombre": "116613.mp3", "duracion": 30, "tiempo": 0, "bpm": 144},
    {"index": 18, "nombre": "116602.mp3", "duracion": 30, "tiempo": 0, "bpm": 200},
    {"index": 19, "nombre": "116604.mp3", "duracion": 30, "tiempo": 0, "bpm": 184},
    {"index": 20, "nombre": "116606.mp3", "duracion": 30, "tiempo": 0, "bpm": 176},
    {"index": 21, "nombre": "116608.mp3", "duracion": 30, "tiempo": 0, "bpm": 168},
    {"index": 22, "nombre": "116610.mp3", "duracion": 30, "tiempo": 0, "bpm": 168},
    {"index": 23, "nombre": "116612.mp3", "duracion": 30, "tiempo": 0, "bpm": 148},
    {"index": 24, "nombre": "116614.mp3", "duracion": 30, "tiempo": 0, "bpm": 132},
    {"index": 25, "nombre": "118601.mp3", "duracion": 30, "tiempo": 0, "bpm": 140},
    {"index": 26, "nombre": "118603.mp3", "duracion": 30, "tiempo": 0, "bpm": 180},
]

```

Figura 2.5: Ejemplo de formato de datos inicial

### 2.1.3. Procedimientos y Transformación de datos

Primero se transforman los datos de un mp3 a una matriz de series de tiempo de audio donde cada uno de las canciones es remuestreado a una frecuencia de muestreo de 11kHz; usamos esa matriz para generar una transformada de Fourier corta duración que usa una función de ventana estandarizada del mismo tamaño de los frames de la transformada de Fourier; devolviendo una matriz de valores complejos de coeficientes de transformada de Fourier a corto plazo (stft) (Imagen: 2.6); después transformamos esos datos a una escala en Mels (melspectrogram) y finalmente transformamos los datos de energía a una escala de decibelios con la ayuda de la función power\_to\_db (Imagen: 2.7); todo esto con la ayuda de la librería Librossa. Esto con el propósito de obtener una simulación gráfica de lo que escucha nuestro oído al escuchar la canción. [32] [33]

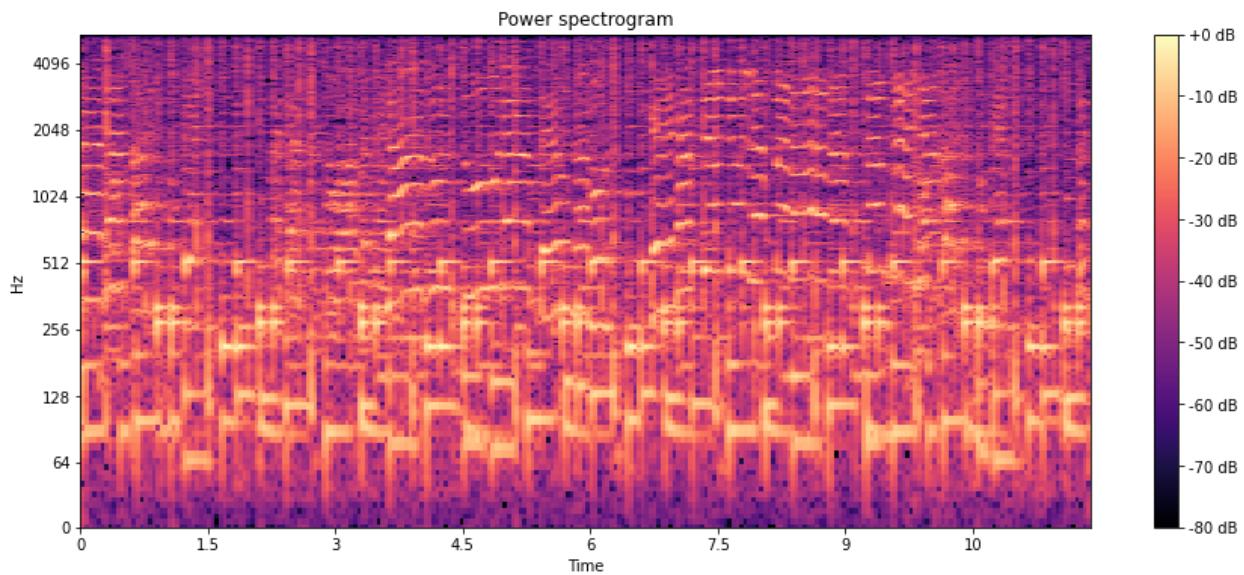


Figura 2.6: Imagen de un Power Spectrogram de un pedazo de la canción Horas Lindas de Adolescentes Orquesta.

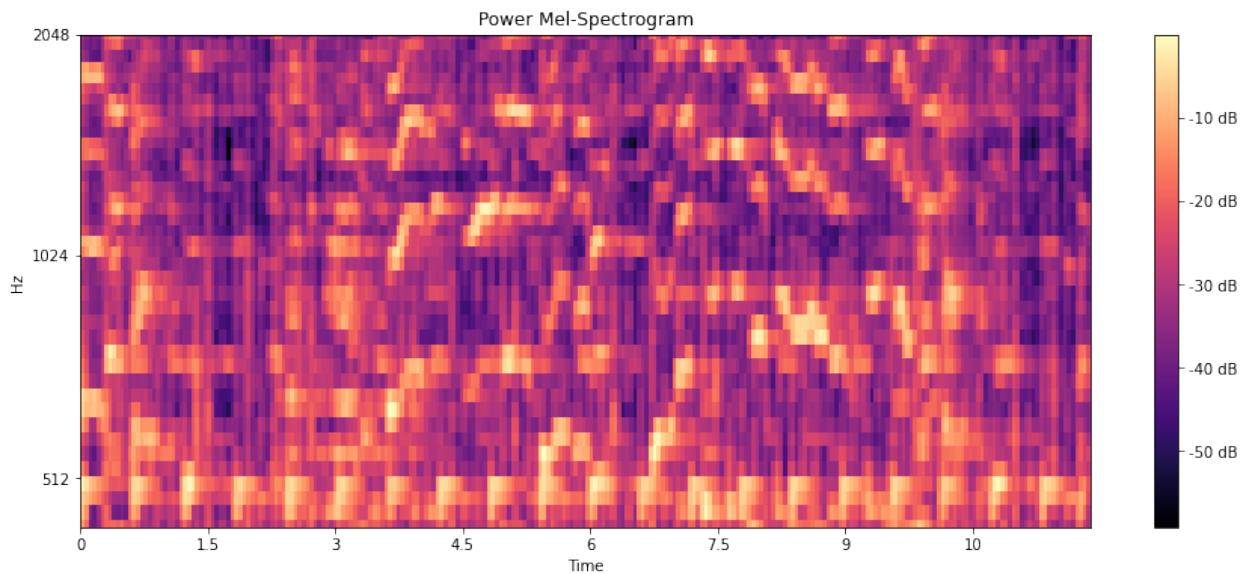


Figura 2.7: Imagen de un Power Mel-Spectrogram de un pedazo de la canción Horas Lindas de Adolescentes Orquesta. Entrada de la red neuronal.

Los datos tendrán 256 frames de tiempo por una escala de 40 Mels. Una vez con estos datos en una escala de Mels se procede a aislar la frecuencia desde un rango aproximado de 400 Hz a 2048 Hz (Imagen: 2.7); esto lo hacemos con el propósito de captar un instrumento en específico, esto se explicará más adelante en detalle. Una vez se tiene la frecuencia aislada, se procede a generar una foto de 256 frames de tiempo por una escala de 40 Mels por un canal de color.

Finalmente al canal de color se normaliza y se estandariza para que los valores sean de 0 a 1, este resultado va a ser la entrada de nuestra red neuronal.

#### 2.1.4. Explicación del uso del Rango de Hz

Para un rango de frecuencias bajas se pueden obtener algunos instrumentos de percusión y bajos, para una frecuencia alta se pueden obtener la voz del artista, una guitarra o un instrumento de tonalidad alta. La salsa se caracteriza por ser una combinación variada de muchos instrumentos, donde usualmente hay más instrumentos de percusión que otra cosa. Nuestro objetivo es buscar patrones en los datos que pueda aprender la red neuronal, pero en la salsa únicamente los instrumentos de percusión son los que posiblemente tenga un patrón.

Dentro de los instrumentos de percusión que se usan en la Salsa podemos encontrar diferentes instrumentos que nos pueden ayudar a obtener patrones, como por ejemplo la conga, maracas, timbales, bongo, cencerro, etc. Al ser de percusión sus tonos varían muy poco y dentro de ellas destaca un instrumento que llamó mi atención el cual es el Cencerro.

El cencerro es una pequeña campana que es hecha a base de cobre o acero; su tono varía dependiendo del tamaño de la campana y donde se toque. Se caracteriza por ser un idiófono (Su tono es único), por lo tanto su sonido siempre va a ser indeterminado (no es constante en un rango de Hz). En determinados momentos este instrumento en la salsa puede llevar el ritmo y velocidad; y al ser un instrumento con sonido único, es fácilmente diferenciable de los demás.

Globalmente se puede obtener 4 tonos diferentes independiente del tamaño las cuales son:

- Sonido Agudo Abierto
- Sonido Agudo Cerrado
- Sonido Grave Abierto
- Sonido Grave Cerrado

Se realizó pequeños experimentos donde se grabó el sonido de un cencerro tocando el instrumento en diferentes partes, aquí los resultados:

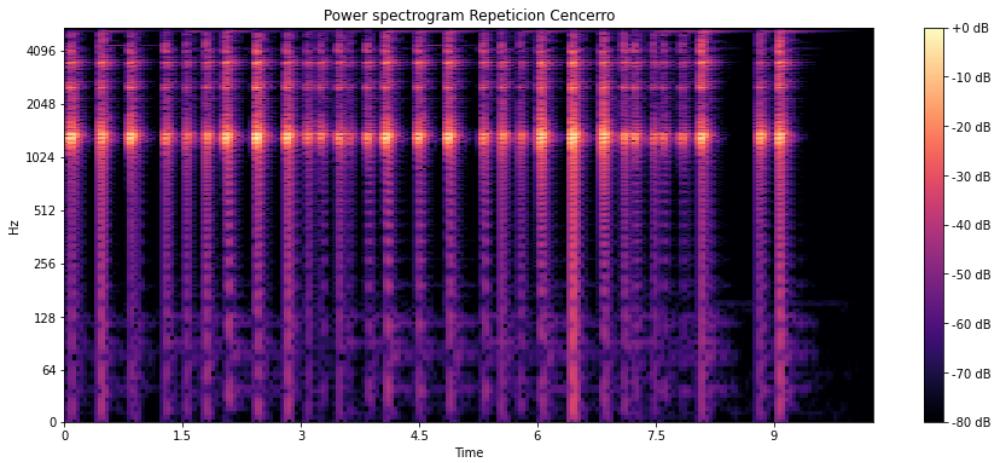


Figura 2.8: Imagen de un Power Spectrogram de un Cencerro tocado repetidamente. Autor: Freddythehat de Wikipedia. Fecha: 19 de febrero del 2007. Propósito únicamente investigativo.

En la imagen 2.8 podemos observar que se repite el mismo sonido del cencerro al golpearlo en una misma posición, dando como resultado un tono y esta se destaca en una frecuencia entre 1024 Hz y 2048 Hz.

También se grabó únicamente el sonido del cencerro simulando en un tono alto o un toque en lo alto (Imagen: 2.9), otro en un tono medio o un toque en el medio (Imagen: 2.10) y otro en un tono bajo o un toque en lo bajo del cencerro (Imagen: 2.11).

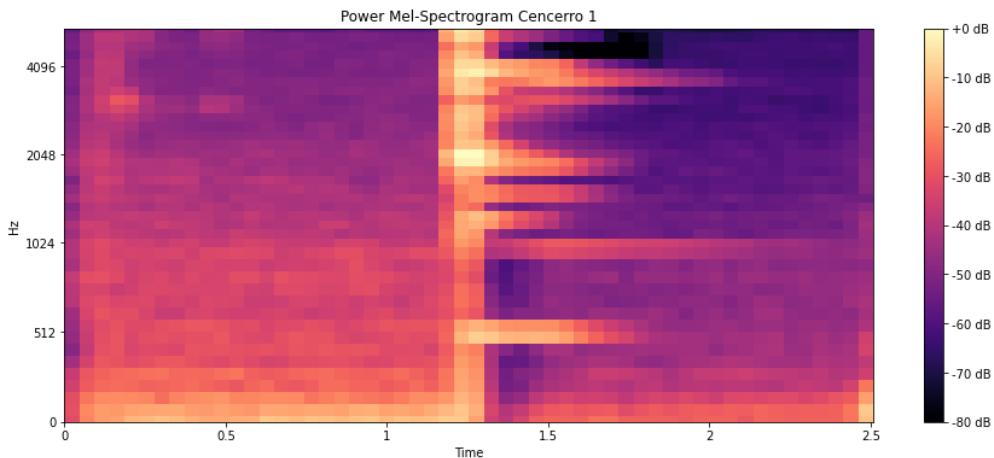


Figura 2.9: Imagen de un Power Mel-Spectrogram de un Cencerro tocado únicamente en la parte superior.

En la imagen 2.9 podemos observar que el mismo instrumento al tocarse una vez puede tener activaciones en diferentes frecuencias como por ejemplo la de 400 Hz a 1024 Hz, también de 1024 Hz a 2048 Hz o la de 3000 Hz a 4096 Hz.

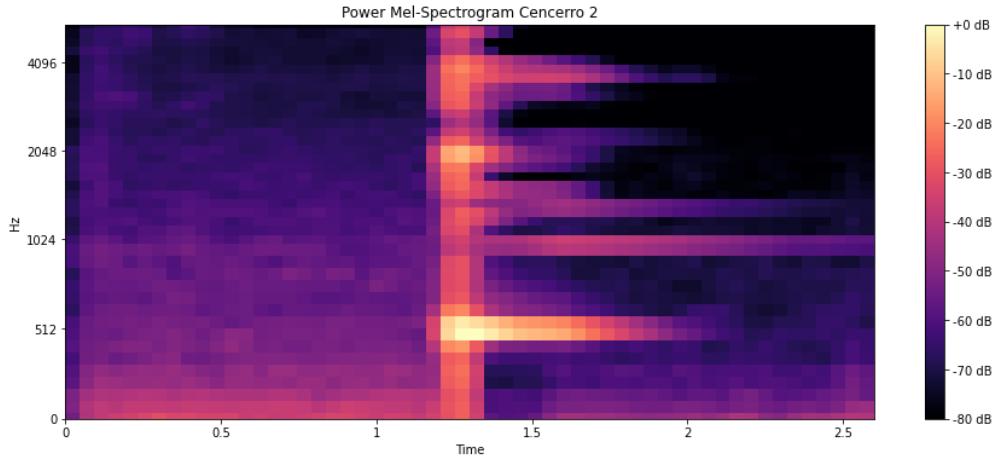


Figura 2.10: Imagen de un Power Mel-Spectrogram de un Cencerro tocado únicamente en la parte del medio.

En la imagen 2.10 podemos observar como influye el tocar en el medio del instrumento, el tono es más fuerte en una frecuencia y no como en la imagen 2.9 que tiene varias activaciones. Aquí explícitamente podemos observar como entre 400 Hz y aproximadamente 700 Hz está la activación o percepción del sonido del cencerro.

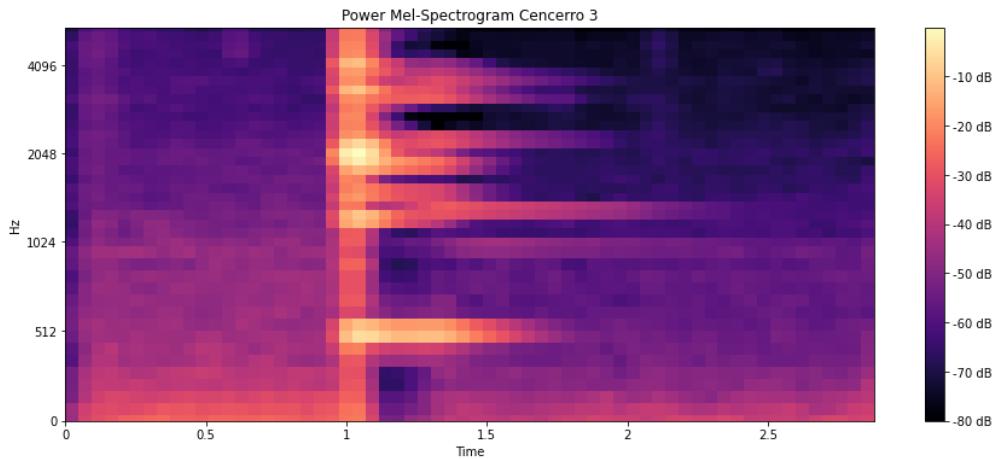


Figura 2.11: Imagen de un Power Mel-Spectrogram de un Cencerro tocado únicamente en la parte baja.

En la imagen 2.11 podemos observar cuando es tocado en la parte baja del cencerro observando

cómo tiene varias activaciones con haberlo tocado una vez, al igual que con la imagen 2.9, podemos observar que entre 400 Hz a 700 Hz hay una activación o de 1600 Hz a 2100 Hz hay otra activación.

Podemos concluir que el cencerro es un instrumento que más o menos entre 400 Hz hasta los 2048 Hz se pueden encontrar al menos una activación. Si se escoge un rango mayor a 2048 Hz se vuelve muy complicado captar la activación, gracias a que en este género los diferentes instrumentos y voces en la canción pueden interferir mandando datos innecesarios.

También a mayor rango de Hz es más difícil encontrar patrones como lo podemos observar en la imagen 2.6. Son casi constantes y no demuestran que esos datos puedan hacer la diferencia a la hora de encontrar un patrón en los datos. En esta imagen 2.6 se escogió un rango de tiempo donde exactamente se escucha el cencerro, podemos observar que sus activaciones están entre el rango 400 Hz hasta 700 Hz.

Por lo tanto se seleccionó este rango de frecuencia.

## 2.2. Red Neuronal

### 2.2.1. Entradas

Las entradas consisten en fotos que se caracterizan por tener 1 canal de color con 256 columnas con 40 filas. Son 256 columnas que hacen referencia al tiempo de la canción, dónde 256 columnas de tiempo es equivalente a un aproximado de 11.85 segundos. Son 40 filas que hacen referencia a las unidades de frecuencia. Tiene un canal de color, el cual esta en un rango de 0 a 1. Las entradas hacen referencia a la imagen 2.7.

### 2.2.2. Salidas

Tenemos un total de 160 salidas que nos representan los BPM desde 60 BPM siendo la posición 0 en el arreglo de salida hasta 220 BPM que representa la posición 160 del arreglo de salida.

Las redes neuronales que se crearon nos predicen una salida en 11.85 segundos de un Tempo local, para poder hallar el BPM que sería el Tempo global, se predice varias veces la canción en diferentes tiempos y los mejores resultados son los escogidos para estimar el BPM de la canción.

### 2.2.3. Arquitectura

- Tensorflow-2.2.0: Es una librería de código abierto que ayuda en el desarrollo y entrenamiento de modelos de aprendizaje automático. [34]

La librería cuenta con modelos que usan diferentes funciones para el análisis de los datos y se explicaran cada una de las diferentes capas usadas en este proyecto y también sus características.

#### 2.2.3.1. Capas

- BatchNormalization: Capa encargada de normalizar los datos en un rango de 0 a 1. Esto con el propósito de ayudar a la red neuronal para que los datos sean más fáciles de procesar. [34]
- Dropout: Capa encargada de desactivar neuronas aleatorias. La función sirve para evitar el “sobreentrenamiento”, porque las neuronas tienden a aprender a veces patrones muy específicos, esta función ayuda a evitar eso. [34]
- Flatten: Capa encargada de transformar múltiples dimensiones a una sola dimensión. Esto sirve para conectar las capas ocultas a las capas de salida o también para la reducción de dimensiones en caso de que se tenga entradas multidimensionales. [34]
- AveragePooling2D: Capa encargada de hallar el promedio para cada pedazo de características y reduciendo su entrada. Esta función trabaja sobre 3 dimensiones, al darle como entrada el área que va a analizar (por ejemplo 2x2) por cada pedazo de entrada, promedia y reduce la

cantidad de datos. Podríamos decir que comprime las características por promedio dado un área. [34]

- MaxPooling2D: Capa encargada de hallar el máximo por cada pedazo de características y restando su entrada. Esta función trabaja sobre 3 dimensiones y al igual que “AveragePooling” con una entrada de área (por ejemplo 2x2) va a maximizar los valores de las características y los reduce. Podríamos decir que comprime las características por máximos dado un área. [34]
- Concatenate: Capa que une una lista de entradas a partir de un eje. [34]
- Convolutional2D: Es una capa de operación que necesita de una cantidad de filtros o canales, una matriz de pesos sobre una entrada de 3 dimensiones para crear más características a partir de la entrada. Funciona a partir de la cantidad de filtros o canales, donde se aplica una matriz de pesos sobre cada canal, al final devuelve la cantidad de canales por la matriz de pesos. Un ejemplo sería una imagen con 1 solo color y le aplico esta capa con 16 filtros y una matriz de 2x2, lo que me da como salida es la imagen con 16 “colores” o canales a la cual se les aplicó una matriz 2x2 de pesos aleatorios. [35]
- Dense: Es una capa que recibe información de las neuronas anteriores y devuelve la cantidad de neuronas que específico en la capa de salida. [34]

### 2.2.3.2. Funciones de Activación

- Relu (Rectified linear unit): Esta función se activa si la entrada está por encima de 0; la salida es una relación lineal con la variable de entrada. [36] [37]

$$f_{(x)} \begin{cases} 0 & \text{si } x \leq 0 \\ x & \text{si } x > 0 \end{cases}$$

- Elu: Esta función se activa tanto para entradas negativas como positivas. Si el dato de entrada es menor o igual a 0 se aplica una función alpha, donde su entrada es el cálculo de su elevado a la euler menos uno. Si su entrada es mayor que 0, sencillamente devuelve el mismo valor. [37]

$$f_{(x)} \begin{cases} \alpha(e^x - 1) & \text{si } x \leq 0 \\ x & \text{si } x > 0 \end{cases}$$

- Softmax: Esta función devuelve “la distribución de probabilidad sobre clases de salida mutuamente excluyentes” [36]. Se usa para la capa de salida porque permite la clasificación de los datos.

### 2.2.3.3. Función de Perdida

- Sparse Categorical Crossentropy:

Es una función de pérdida para el uso de clasificación de múltiples clases. Esta función cumple con cuantificar la diferencia entre múltiples distribuciones de probabilidad. El proyecto cuenta con una clasificación amplia de datos que van desde 0 hasta 160, por lo tanto esta función de pérdida se adapta a nuestras características. [38] [39] [34]

#### 2.2.3.4. Función de Optimización

- Adam:

“Este algoritmo de optimización usa gradientes de primer orden de funciones objetivo estocásticas, basados en estimaciones adaptativas de momentos de orden inferior” [40]. El algoritmo es muy eficiente y excelente para problemas con grandes datos o grandes clasificaciones. Como el proyecto cuenta con una gran cantidad de datos de entrada y una gran clasificación que va desde 0 a 160, esta función de optimización es ideal para nuestra red neuronal. [34]

### 2.2.4. Modelos

#### 2.2.4.1. Modelo 1

La red neuronal cuenta con un Batch Normalization, una capa convolucional de 2 dimensiones con 16 filtros con un kernel de 5 una función de activación “ELU” y con un relleno igual (“same”), así repetimos 2 veces más el Batch Normalization y la capa convolucional de 2 dimensiones. Estas capas se usan con el propósito de ampliar las características de la foto de entrada; estas capas trabajan y buscan activaciones o valores altos en los datos.

Para la capa intermedia tenemos un Average Pooling de 2 dimensiones con un pool de 5, un Batch Normalization y 6 capas convolucionales de 2 dimensiones en paralelo donde cada una usa 24 filtros pero su kernel estará fragmentado en 6 partes diferentes: (1 x 32), (1 x 64), (1 x 96), (1 x 128), (1 x 192) y (1 x 256); usando una función de activación “ELU” y un relleno igual (“same”). Esas 6 capas en paralelo las concatenamos por su AXIS 1 y aplicamos una capa convolucional de 2 dimensiones de 36 filtros con un kernel de 1 y un stride de 1, usando una función de activación “ELU”. Volvemos a aplicar un Average Pooling de 2 dimensiones pero su tamaño de pool ahora es de 2; después volvemos a repetir el proceso desde el Batch Normalization hasta la capa convolucional de 2 dimensiones de 36 filtros. Estas capas se usan con el propósito de analizar seis regiones diferentes en el tiempo, donde cada región es de aproximadamente 32 frames para poder ver cada sección con más características. [41] [42]

Para la capa de salida convertimos los datos de 3 dimensiones a 1 dimensión con la función Flatten, aplicamos un Batch Normalization, un DropOut de 0.5, una capa densa de 64 unidades con una activación “RELU”, otro Batch Normalization, otra capa densa de 64 unidades con una activación “RELU”, otro Batch Normalization y finalizamos con la capa de densa de salida de 160 unidades y una función de activación “SOFTMAX”. Estas capas se usan para comprimir todos las características y sacar el resultado que deseamos; se le aplica un dropout con el propósito de que la red neuronal no aprenda más de lo debido.

Este modelo cuenta con 5'094,578 parámetros entrenables (Imagen 2.12 y 2.13).

Esta red neuronal se reduce de la siguiente red neuronal [21].

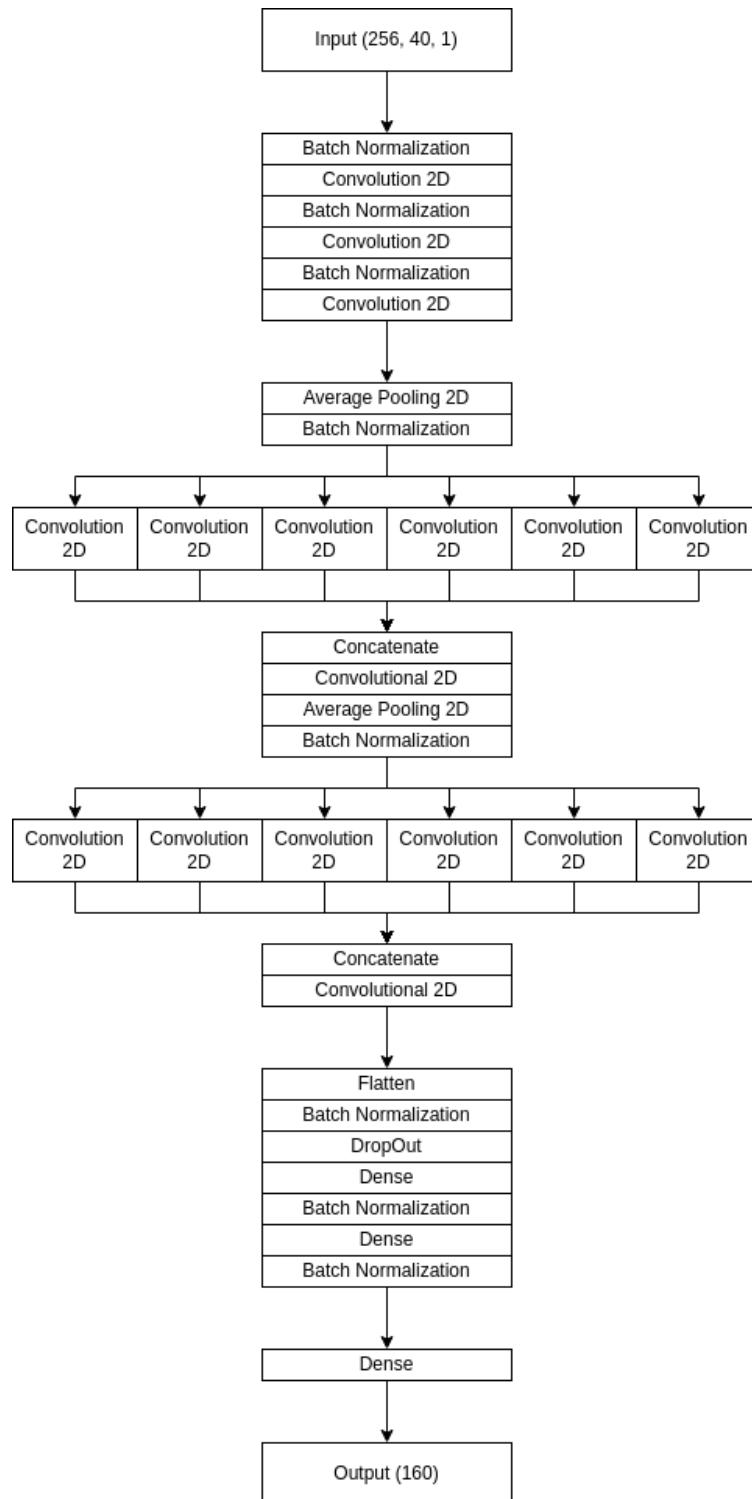


Figura 2.12: Imagen del modelo de la red neuronal 1

Model: "model_4"			
Layer (type)	Output Shape	Param #	Connected to
input_5 (InputLayer)	[None, 256, 40, 1] 0		
batch_normalization_30 (BatchNo)	(None, 256, 40, 1) 4	input_5[0][0]	
conv2d_54 (Conv2D)	(None, 252, 36, 16) 416	batch_normalization_30[0][0]	
batch_normalization_31 (BatchNo)	(None, 252, 36, 16) 64	conv2d_54[0][0]	
conv2d_55 (Conv2D)	(None, 248, 32, 16) 6416	batch_normalization_31[0][0]	
batch_normalization_32 (BatchNo)	(None, 248, 32, 16) 64	conv2d_55[0][0]	
conv2d_56 (Conv2D)	(None, 244, 28, 16) 6416	batch_normalization_32[0][0]	
average_pooling2d_6 (AveragePoo)	(None, 48, 5, 16) 0	conv2d_56[0][0]	
batch_normalization_33 (BatchNo)	(None, 48, 5, 16) 64	average_pooling2d_6[0][0]	
conv2d_57 (Conv2D)	(None, 48, 5, 24) 12312	batch_normalization_33[0][0]	
conv2d_58 (Conv2D)	(None, 48, 5, 24) 24600	batch_normalization_33[0][0]	
conv2d_59 (Conv2D)	(None, 48, 5, 24) 36888	batch_normalization_33[0][0]	
conv2d_60 (Conv2D)	(None, 48, 5, 24) 49176	batch_normalization_33[0][0]	
conv2d_61 (Conv2D)	(None, 48, 5, 24) 73752	batch_normalization_33[0][0]	
conv2d_62 (Conv2D)	(None, 48, 5, 24) 98328	batch_normalization_33[0][0]	
concatenate_6 (Concatenate)	(None, 288, 5, 24) 0	conv2d_57[0][0] conv2d_58[0][0] conv2d_59[0][0] conv2d_60[0][0] conv2d_61[0][0] conv2d_62[0][0]	
conv2d_63 (Conv2D)	(None, 288, 5, 36) 900	concatenate_6[0][0]	
average_pooling2d_7 (AveragePoo)	(None, 144, 2, 36) 0	conv2d_63[0][0]	
batch_normalization_34 (BatchNo)	(None, 144, 2, 36) 144	average_pooling2d_7[0][0]	
conv2d_64 (Conv2D)	(None, 144, 2, 24) 27672	batch_normalization_34[0][0]	
conv2d_65 (Conv2D)	(None, 144, 2, 24) 55320	batch_normalization_34[0][0]	
conv2d_66 (Conv2D)	(None, 144, 2, 24) 82968	batch_normalization_34[0][0]	
conv2d_67 (Conv2D)	(None, 144, 2, 24) 110616	batch_normalization_34[0][0]	
conv2d_68 (Conv2D)	(None, 144, 2, 24) 165912	batch_normalization_34[0][0]	
conv2d_69 (Conv2D)	(None, 144, 2, 24) 221208	batch_normalization_34[0][0]	
concatenate_7 (Concatenate)	(None, 864, 2, 24) 0	conv2d_64[0][0] conv2d_65[0][0] conv2d_66[0][0] conv2d_67[0][0] conv2d_68[0][0] conv2d_69[0][0]	
conv2d_70 (Conv2D)	(None, 864, 2, 36) 900	concatenate_7[0][0]	
flatten_4 (Flatten)	(None, 62208) 0	conv2d_70[0][0]	
batch_normalization_35 (BatchNo)	(None, 62208) 248832	flatten_4[0][0]	
dropout_4 (Dropout)	(None, 62208) 0	batch_normalization_35[0][0]	
dense_12 (Dense)	(None, 64) 3981376	dropout_4[0][0]	
batch_normalization_36 (BatchNo)	(None, 64) 256	dense_12[0][0]	
dense_13 (Dense)	(None, 64) 4160	batch_normalization_36[0][0]	
batch_normalization_37 (BatchNo)	(None, 64) 256	dense_13[0][0]	
dense_14 (Dense)	(None, 160) 10400	batch_normalization_37[0][0]	

Total params: 5,219,420  
Trainable params: 5,094,578  
Non-trainable params: 124,842

Figura 2.13: Resumen del modelo de la red neuronal 1.

### 2.2.4.2. Modelo 2

La segunda red neuronal cuenta con un Batch Normalization, una capa convolucional de 2 dimensiones de 32 filtros con un kernel de (3 x 40) usando una función de activación “RELU”, después aplicamos otro Batch Normalization y aplicamos un Max Pooling de 2 dimensiones de un pool de tamaño (1 x 1); volvemos a usar otra capa convolucional de 2 dimensiones de 32 filtros con un kernel de (3, 1) usando una función de activación “ELU”, aplicamos otro batch Normalization y finalizamos con un Max Pooling de 2 dimensiones con un pool de tamaño (2, 1).

Esta se hace con el propósito de analizar rápidamente la imagen y aumentar la cantidad de características para un mayor análisis, sobre el tiempo y la frecuencia.

Una vez tenemos una red neuronal con suficientes características, convertimos los datos de 3 dimensiones a 1 dimensión con la función Flatten. Usamos una capa densa de 256 unidades usando una función de activación “ELU”, un DropOut de 0.5 para evitar que la IA aprenda más de lo debido y finalizamos con una capa densa de 160 unidades usando una función de activación “SOFTMAX”.

Este modelo cuenta con 1'080,752 parámetros entrenables (Imagen: [2.14](#) y [2.15](#)).

Esta red neuronal es sencilla y simple; se basa en el modelo de la siguiente red neuronal [33].

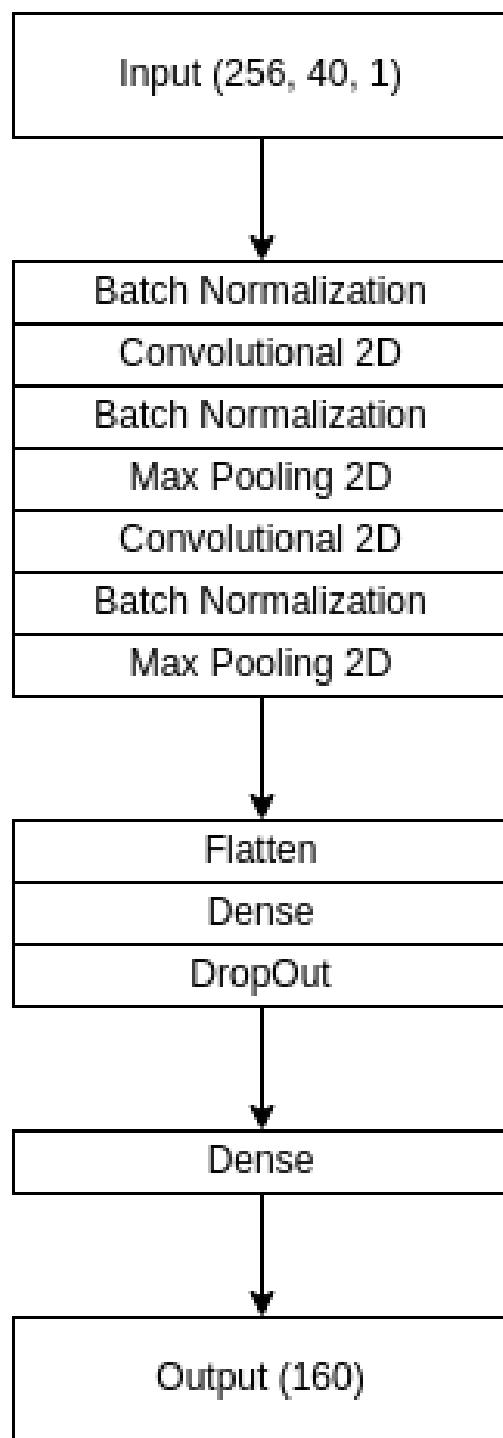


Figura 2.14: Imagen del modelo de la red neuronal 2.

```
Model: "model_4"
```

Layer (type)	Output Shape	Param #
input_9 (InputLayer)	[ (None, 256, 40, 1) ]	0
batch_normalization_28 (BatchNormalization)	(None, 256, 40, 1)	160
conv2d_45 (Conv2D)	(None, 254, 1, 32)	3872
batch_normalization_29 (BatchNormalization)	(None, 254, 1, 32)	128
max_pooling2d_8 (MaxPooling2D)	(None, 254, 1, 32)	0
conv2d_46 (Conv2D)	(None, 252, 1, 32)	3104
batch_normalization_30 (BatchNormalization)	(None, 252, 1, 32)	128
max_pooling2d_9 (MaxPooling2D)	(None, 126, 1, 32)	0
flatten_6 (Flatten)	(None, 4032)	0
dense_14 (Dense)	(None, 256)	1032448
dropout_6 (Dropout)	(None, 256)	0
dense_15 (Dense)	(None, 160)	41120
<hr/>		
Total params: 1,080,960		
Trainable params: 1,080,752		
Non-trainable params: 208		

---

Figura 2.15: Resumen del modelo de la red neuronal 2.

### 2.2.4.3. Modelo 3

La tercera red neuronal es un modelo secuencial reducido del modelo 2 que contiene 2 convoluciones de 2 dimensiones, la primera cuenta con 32 filtros y la segunda 64 filtros, ambas tienen un kernel de (3 x 3) y una activación “RELU” y 2 Max Pooling con un pool de (2 x 2). Esto con el propósito de aumentar las características de la imagen de entrada.

Después aplicamos un DropOut de 0.25 para evitar que aprenda mas de lo debido, procedemos con convertir los datos de 3 dimensiones a una 1 dimensión con la función Flatten y terminamos con 2 capas densas de 256 unidades con activación RELU". Terminamos con una capa densa de 160 unidades que va a ser nuestra salida usando de activación la función SOFTMAX (Imagen: 2.16).

Este modelo cuenta con 8'186,656 parámetros entrenables (Imagen: 2.16 y 2.17).

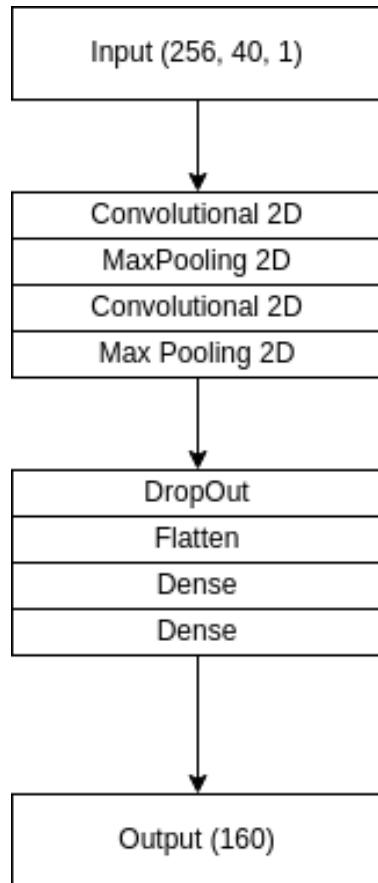


Figura 2.16: Imagen del modelo de la red neuronal 3

```

Model: "sequential_1"

Layer (type)          Output Shape       Param #
=====
conv2d_5 (Conv2D)      (None, 254, 38, 32)   320
max_pooling2d_4 (MaxPooling2D) (None, 127, 19, 32)   0
conv2d_6 (Conv2D)      (None, 125, 17, 64)    18496
max_pooling2d_5 (MaxPooling2D) (None, 62, 8, 64)    0
dropout_2 (Dropout)    (None, 62, 8, 64)    0
flatten_2 (Flatten)    (None, 31744)        0
dense_4 (Dense)        (None, 256)          8126720
dense_5 (Dense)        (None, 160)          41120
=====
Total params: 8,186,656
Trainable params: 8,186,656
Non-trainable params: 0

```

Figura 2.17: Resumen del modelo de la red neuronal 3.

### 2.2.5. Entrenamiento

Para los entrenamientos se usó un computador con las siguientes características:

- Intel Core I7 - 9 generación de 8 núcleos, de 3.6Ghz hasta 4.9Ghz
- Ram: 16 Gb de 4200 Mhz
- GPU: GTX 1060 de 6gb

#### 2.2.5.1. Modelo 1

Se usó aproximadamente el 90 % de los datos para entrenar en la red y aproximadamente un 10 % para validación; los datos son randomizados. Se usa para su función de precisión “Sparse Categorical Crossentropy” que nos permite obtener un resultado con muchos tipos de clasificación. Para evitar el sobreentrenamiento se determinó que después de 20 generaciones donde cada una tiene 20 imágenes por ciclo y usando como optimizador una función “Adam”, es donde se obtuvo el mejor resultado.

**2.2.5.2. Modelo 2**

Se usó aproximadamente el 90 % de los datos para entrenar en la red y aproximadamente un 10 % para validación; los datos son randomizados. Se usa para su función de precisión “Sparse Categorical Crossentropy” que nos permite obtener un resultado con muchos tipos de clasificación. Para evitar el sobreentrenamiento se determinó que después de 30 generaciones donde cada una tiene 20 imágenes por ciclo y usando como optimizador una función “Adam”, es donde se obtuvo el mejor resultado.

**2.2.5.3. Modelo 3**

Se usó aproximadamente el 90 % de los datos para entrenar en la red y aproximadamente un 10 % para validación; los datos son randomizados. Se usa para su función de precisión “Sparse Categorical Crossentropy” que nos permite obtener un resultado con muchos tipos de clasificación. Para evitar el sobreentrenamiento se determinó que después de 20 generaciones donde cada una tiene 10 imágenes por ciclo y usando como optimizador una función “Adam”, es donde se obtuvo el mejor resultado.

### 2.2.6. Resultados

Los resultados fueron los siguientes:

#### 2.2.6.1. Modelo 1

- Datos de Entrenamiento:

Con un total de 8073 datos y un tiempo estimado de más o menos 2 horas se obtuvo una pérdida aproximada de 0.6535 y una predicción del 88.49 %.

- Datos de Validación:

Con un total de 892 datos y un tiempo compartido con los datos de entrenamiento se obtuvo una pérdida aproximada de 2.3211 y una predicción aproximada del 51.57 %.

Los datos de validación es escoger un dato de cada una de las canciones de los datos de entrenamiento y quitarlo de ese conjunto, para tener una validación por cada canción.

- Datos de Prueba:

Con un total de 637 datos y un tiempo de 8 segundos para su cálculo, se obtuvo una pérdida aproximada de 1.0039 y una predicción aproximada del 84.93 %.

- Datos de Extended-BIGROOM:

Con un total de 94 datos y un tiempo de un 1 segundo para su cálculo, se obtuvo una pérdida aproximada de 8.4479 y una predicción aproximada del 9.57 %.

El modelo del cual se basó este nuevo modelo propuesto, tuvo una probabilidad máxima del 97.4 % y una probabilidad mínima de 50.2 %; con un promedio de aproximado de 92 % [21]. De acuerdo a estos datos se estima que los resultados estuvieron entre ese rango que fue lo logrado.

No se logró un buen resultado en los datos del Extended-BIGROOM, esto a causa de los pocos datos que hay, aun así, el resultado obtenido es mucho mejor que el modelo 2 y el modelo 3 para este conjunto de datos.

Se puede concluir que en ese rango de frecuencias entre 400 Hz y 2048 Hz el modelo pudo encontrar un patrón dentro de los datos.

### 2.2.6.2. Modelo 2

- Datos de Entrenamiento:

Con un total de 8073 datos y un tiempo estimado de más o menos 1 hora se obtuvo una pérdida aproximada de 0.0222 y una predicción del 99.63 %.

- Datos de Validación:

Con un total de 892 datos y un tiempo compartido con los datos de entrenamiento se obtuvo una pérdida aproximada de 0.5619 y una predicción aproximada del 92.83 %.

Los datos de validación es escoger un dato de cada una de las canciones de los datos de entrenamiento y quitarlo de ese conjunto, para tener una validación por cada canción.

- Datos de Prueba:

Con un total de 637 datos y un tiempo de 1 segundo para su cálculo, se obtuvo una pérdida aproximada de 6.5750 y una predicción aproximada del 12.87 %.

- Datos de Extended-BIGROOM:

Con un total de 94 datos y un tiempo de un 1 segundo para su cálculo, se obtuvo una pérdida aproximada de 9.3137 y una predicción aproximada del 1.06 %.

Este modelo tuvo muy buen desempeño con las canciones que entreno, en el caso de los datos de entreno obtuvo una media por encima de 90 %, prediciendo casi todo los datos.

El modelo aprendió únicamente fotos iguales y no con patrones en el tiempo, causando que únicamente pueda reconocer fotos con patrones muy parecidos a los de su entrenamiento. Aunque se probó con diferentes generaciones con menor porcentaje de entrenamiento, al evaluar en los datos de prueba este aun así no era capaz de superar el 10 %.

El desempeño para el conjunto de datos de Extended-BIGROOM es extremadamente pobre, podemos asumir que no encontró ningún patrón dentro de los datos.

Podemos concluir que este modelo no sirve para reconocer imágenes debido a que no se hace un análisis particionado de la imagen en el tiempo.

### 2.2.6.3. Modelo 3

- Datos de Entrenamiento:

Con un total de 8073 datos y un tiempo estimado de más o menos 2 horas se obtuvo una pérdida aproximada de 0.0529 y una predicción del 99.24 %.

- Datos de Validación:

Con un total de 892 datos y un tiempo compartido con los datos de entrenamiento se obtuvo una pérdida aproximada de 0.6198 y una predicción aproximada del 92.60 %.

Los datos de validación es escoger un dato de cada una de las canciones de los datos de entrenamiento y quitarlo de ese conjunto, para tener una validación por cada canción.

- Datos de Prueba:

Con un total de 637 datos y un tiempo de 1 segundo para su cálculo, se obtuvo una pérdida aproximada de 8.6703 y una predicción aproximada del 7.85 %.

- Datos de Extended-BIGROOM:

Con un total de 94 datos y un tiempo de un 1 segundo para su cálculo, se obtuvo una pérdida aproximada de 22.0336 y una predicción aproximada del 2.13 %.

El modelo se redujo y se trató de aumentar la cantidad de parámetros entrenables del modelo 2 para que la red pudiera captar más características.

Obtuvo con los datos de entrenamiento un porcentaje similar al modelo 2, mayor de 90 %. Aun así el modelo volvió a sufrir los mismos errores a la hora de validar con los datos de prueba. Su pobre predicción nos deja concluir que el modelo no sirve al aumentar la cantidad de parámetros. Se debe a que no hay análisis particionado sobre el tiempo, por lo tanto el modelo analiza toda la imagen e intenta buscar patrones sobre toda la imagen y no particionado sobre partes de ella, causando que solo aprenda sobre la imagen y no patrones sobre el tiempo.

Igual que el modelo 2 para el desempeño del conjunto de datos de Extended-BIGROOM fue muy bajo, solo demostrando ser por 1 % más eficiente que el modelo 2. Podemos concluir que no encontró ningún patrón dentro de los datos.

## 2.3. Detección de BPM

Los resultados obtenidos por las redes neuronales no detectan un BPM global de la canción, si no que detectan un tempo local de la canción, por lo que es necesario pasar varias veces la canción en diferentes puntos de reproducción para así obtener un estimado de BPM global.

La canción a la cual se quiera detectar el BPM se parte en 6 partes diferentes y se hace un promedio de sus resultados; los resultados al ser un arreglo de 160 datos donde cada posición se refiere a un BPM (si está en la posición 0 se refiere a que tiene un BPM de 60) después de promediar los resultados se obtiene un BPM global.

Cómo se obtienen varios resultados de la red, se puede obtener más de una predicción, por lo tanto no podemos descartar todas las posibilidades o al menos las que tiene una probabilidad cercana a la mayor de ellas. Se dejan las 10 mejores probabilidades.

## 2.4. Interfaz de Usuario

### 2.4.1. Encuesta

Se creó una encuesta con el propósito de que las personas introduzcan manualmente el BPM, indicando donde esta cada tempo de la canción.

Para el Back-end se usó Python3 con la librería Flask [43] para crear un servicio tipo HTTP. También se usó MongoDB para guardar los datos obtenidos de la encuesta. Para el Front-end se usó Angular8 [44] con la ayuda de la librería “wavesurfer.js” [45] que nos ayuda con la visualización de la canción y parametrización de los datos de la encuesta.



Figura 2.18: Imagen de la encuesta explicando la introducción.

Primero se da una breve información de lo que tiene que hacer el usuario.



Figura 2.19: Imagen de la encuesta.

Una vez el usuario de clic en el botón de “Empezar” se desplegará las instrucciones de como debe de hacer la encuesta. Puede arrastrar el puntero de la canción, como también reproducirla; donde esté parado el puntero el usuario le puede dar en el botón de marcar para identificar que ahí hay un tempo.

Al seleccionar dónde está el ritmo, la persona puede terminar la encuesta y subirla. La información después es almacenada en un Mongo, donde a partir de las marcas que hizo se hace un cálculo estimado del BPM de la canción. (Imagen: 2.20)



Figura 2.20: Imagen de la encuesta final.

A continuación se explica el proceso de construcción. Para su implementación se crean las siguientes historias de usuarios:

Nombre	Encuesta-001
Tiempo Estimación	22-HORAS
Riesgo	Alto
<b>Descripción:</b>	
<ul style="list-style-type: none"> <li>● Se requiere de una aplicación que sea capaz de realizar una encuesta en línea.</li> <li>● La encuesta tendrá los siguientes parámetros: <ul style="list-style-type: none"> <li>○ La encuesta tendrá 100 canciones.</li> <li>○ Se debe de mostrar en la encuesta la canción en forma de ondas en el dominio del tiempo.</li> <li>○ La canción solo podrá durar 30 segundos.</li> <li>○ La persona podrá marcar el beat en la onda de la canción.</li> <li>○ Se podrá volver a hacer la encuesta si la persona lo quiera.</li> </ul> </li> </ul>	
<b>Validación:</b>	
<ul style="list-style-type: none"> <li>● El usuario tendrá una introducción de lo que va a realizar a continuación en la encuesta.</li> <li>● El usuario podrá ver, reproducir y pausar una canción aleatoria únicamente 30 segundos.</li> <li>● El usuario podrá marcar sobre la onda de sonido de la canción el beat en donde cree que se encuentre; podrá hacerlo varias veces.</li> <li>● El usuario tendrá que terminar la encuesta.</li> <li>● El usuario podrá volver a hacer la encuesta si quiere.</li> </ul>	

Figura 2.21: Requerimiento General 1.

Nombre	Encuesta-001.1
Tiempo Estimación	2-HORAS
Riesgo	Alto
<b>Descripción:</b>	
<ul style="list-style-type: none"> <li>● Reunir 100 canciones.</li> </ul>	
<b>Validación:</b>	
<ul style="list-style-type: none"> <li>● Recolectar de una base de datos privada un conjunto de 100 canciones aleatorias.</li> </ul>	

Figura 2.22: Requerimiento Específico 1.

Nombre	Encuesta-001.2
Tiempo Estimación	8-HORAS
Riesgo	Alto
<b>Descripción:</b>	
<ul style="list-style-type: none"> <li>● Construir back-end que tenga los siguientes entradas:           <ul style="list-style-type: none"> <li>○ <u>infoCancion</u></li> <li>○ <u>guardarEncuesta</u></li> </ul> </li> </ul>	
<b>Validación:</b>	
<ul style="list-style-type: none"> <li>● La infraestructura back-end tendrá los siguientes back-end:           <ul style="list-style-type: none"> <li>○ <u>infoCancion</u>:               <ul style="list-style-type: none"> <li>▪ <u>get</u>:                   <ul style="list-style-type: none"> <li>● Obtener información de la canción de la cual se hará la encuesta.</li> </ul> </li> <li>▪ <u>post</u>:                   <ul style="list-style-type: none"> <li>● Obtener la canción en tipo blob.</li> </ul> </li> </ul> </li> <li>○ <u>guardarEncuesta</u>:               <ul style="list-style-type: none"> <li>▪ <u>post</u>:                   <ul style="list-style-type: none"> <li>● Guardar un arreglo de la posición de los beats que marcó el usuario en la encuesta en una base de datos Mongo.</li> </ul> </li> </ul> </li> </ul> </li> </ul>	

Figura 2.23: Requerimiento Específico 2.

Nombre	Encuesta-001.3
Tiempo Estimación	12-HORAS
Riesgo	Alto
<b>Descripción:</b>	
<ul style="list-style-type: none"> <li>● Construir un front-end que sea capaz de:           <ul style="list-style-type: none"> <li>○ Visualizar la canción en una forma de onda en el dominio del tiempo.</li> <li>○ Poder marcar en la onda en el dominio del tiempo el beat donde se estima que esta; poder hacerlo varias veces.</li> <li>○ Reproducir y pausar la canción.</li> <li>○ Subir los datos de la encuesta.</li> <li>○ Volver a hacer la encuesta.</li> <li>○ Explicar lo que tiene que hacer el usuario</li> </ul> </li> </ul>	
<b>Validación:</b>	
<p>La vista se desplegará de la siguiente forma.</p> <ol style="list-style-type: none"> <li>1. Se crea una vista informando informando de que lo debe de hacer el usuario.</li> <li>2. Se crea una vista donde desplegará lo siguiente:       <ol style="list-style-type: none"> <li>a. Con la ayuda de la librera <u>waveserver.js</u> desplegar la canción en forma de onda en el dominio del tiempo.</li> <li>b. Agregar el botón reproducir.</li> <li>c. Agregar el botón pausar</li> <li>d. Agregar el botón marcar, que marcará una línea en la forma de onda de la canción indicando que ahí hay un beat.</li> <li>e. Agregar botón terminar encuesta.</li> </ol> </li> <li>3. Crear una vista que pregunte al usuario si desea volver a hacer la encuesta.</li> </ol>	

Figura 2.24: Requerimiento Específico 3.

Este desarrollo tiene el siguiente diagrama de flujo para el usuario:

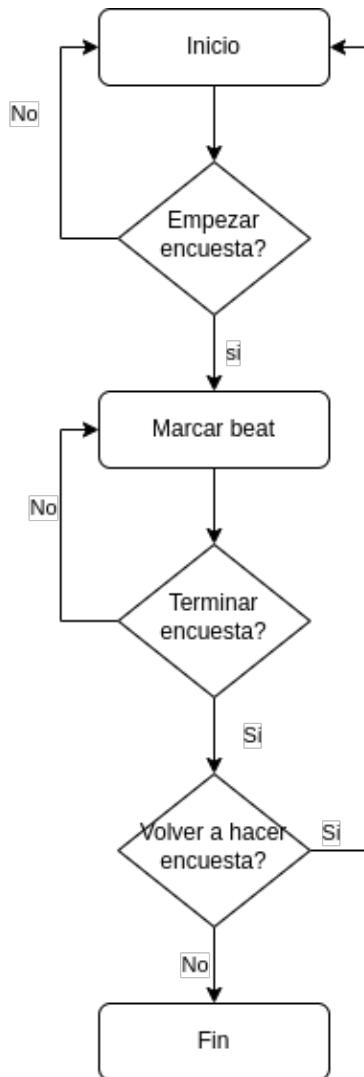


Figura 2.25: Diagrama de flujo de la vista Encuesta.

#### 2.4.2. Predicción de BPM

Se creó un Back-end en Python usando la librería Flask [43] para crear un servicio tipo HTTP. Este se encarga de procesar la canción, cargarla en el modelo y devolver los resultados.

Para el Front-end se usa Angular8 [44] con la librería “echarts” [46] para mostrar gráficamente los resultados obtenidos de esa canción.

Se crea una interfaz de usuario amigable y sencilla donde el usuario selecciona la canción que quiere hallar el BPM (Imagen: 2.26).

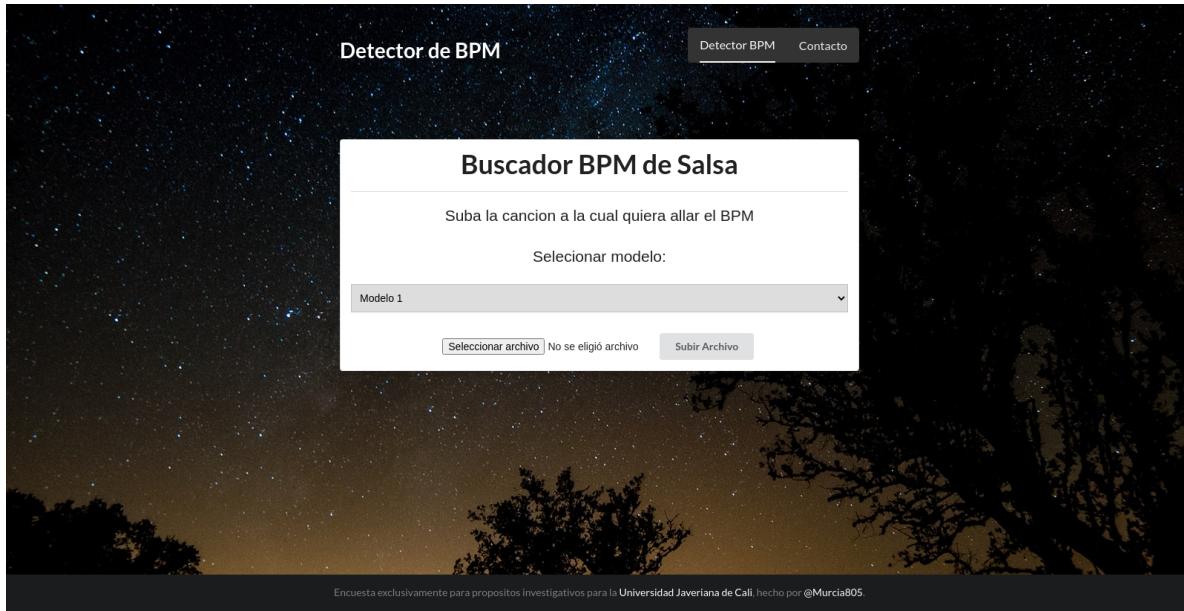


Figura 2.26: Imagen de la interfaz de Predicción.

Una vez se sube la canción al servidor, este se le aplicará una transformación igual a la de entrada de los datos de la red neuronal, pero se cogerán diferentes partes de la canción, para un total de 6 pedazos diferentes de la canción, esto con el propósito de hallar un BPM global, se suman sus probabilidades y se promedia en porcentaje. Una vez hecho este cálculo los datos son mostrados en un gráfico.

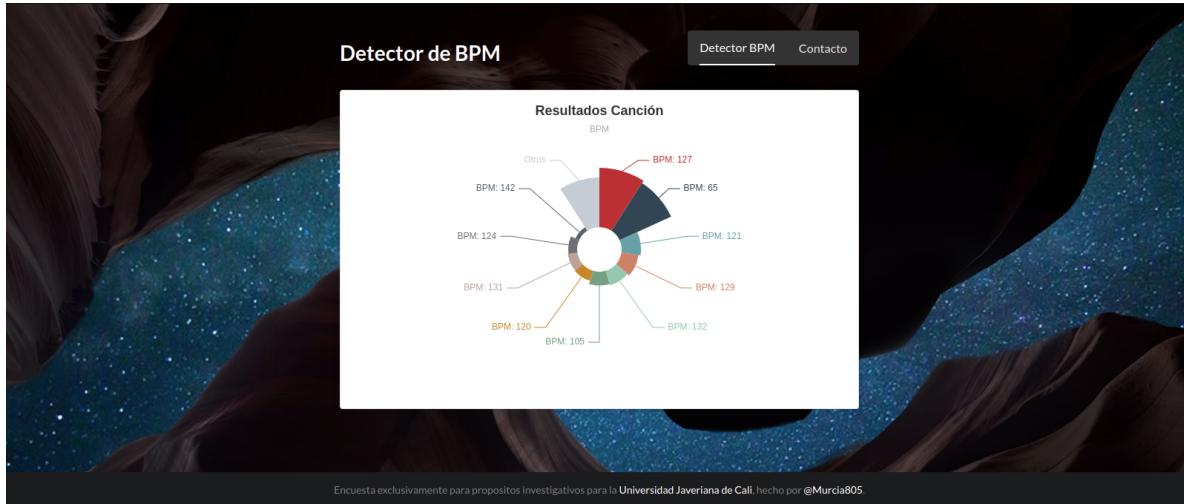


Figura 2.27: Imagen de la interfaz de Predicción resultado final.

Así finalmente los datos son impresos de manera gráfica y sencilla, al ubicarse sobre cada pedazo se muestra la probabilidad obtenida por cada una de los valores BPM mostrados.

A continuación se explica el proceso de construcción. Para su implementación se crean las siguientes historias de usuarios:

Nombre	Prediccion-001
Tiempo Estimación	30-HORAS
Riesgo	Alto
Descripción:	<ul style="list-style-type: none"> <li>● Se requiere de una aplicación que sea capaz de mostrar las predicciones de la inteligencia artificial de una canción de salsa.</li> <li>● La interfaz contará con los siguientes opciones: <ul style="list-style-type: none"> <li>○ Mostrar la canción subida en ondas en el dominio del tiempo.</li> <li>○ Poder pausar o reproducir la canción.</li> <li>○ Cuando el usuario presiona el botón de reproducir, se debe de empezar el cálculo de la inteligencia artificial.</li> <li>○ Se debe de poder ver las predicciones finales con sus porcentajes de predicción.</li> <li>○ Se debe escoger el BPM de las predicciones y mostrar en la canción dónde está cada beat.</li> <li>○ Se debe de volver a hacer la predicción si el usuario lo requiere.</li> </ul> </li> </ul>
Validación:	<ul style="list-style-type: none"> <li>● El usuario podrá escoger entre los 3 modelos de predicción entrenados.</li> <li>● El usuario podrá subir su canción de salsa.</li> <li>● El usuario podrá ver su canción en ondas en el dominio del tiempo.</li> <li>● El usuario podrá reproducir y pausar la canción.</li> <li>● El usuario podrá ver la información de la predicción de la Inteligencia artificial de manera práctica.</li> <li>● El usuario podrá seleccionar la predicción que quiera y podrá ver el beat en la canción en ondas en el dominio del tiempo.</li> <li>● El usuario podrá ver el tiempo que se demora a la hora de predecir los resultados.</li> <li>● El usuario podrá volver a hacer la predicción si lo requiere.</li> </ul>

Figura 2.28: Requerimiento General 1.

Nombre	Prediccion-001.1
Tiempo Estimación	6-HORAS
Riesgo	Alto
Descripción:	<ul style="list-style-type: none"> <li>● Construir back-end que tenga los siguientes entradas: <ul style="list-style-type: none"> <li>○ predicción.</li> </ul> </li> </ul>
Validación:	<ul style="list-style-type: none"> <li>● La infraestructura back-end tendrá los siguientes back-end: <ul style="list-style-type: none"> <li>○ predicción: <ul style="list-style-type: none"> <li>▪ post: <ul style="list-style-type: none"> <li>● A partir del modelo seleccionado y la canción se predecirá el BPM de la canción.</li> </ul> </li> </ul> </li> </ul> </li> </ul>

Figura 2.29: Requerimiento Específico 1.

Nombre	Prediccion-001.2
Tiempo Estimación	24-HORAS
Riesgo	Alto
<b>Descripción:</b>	
<ul style="list-style-type: none"> <li>● Construir un front-end que sea capaz de:           <ul style="list-style-type: none"> <li>○ Seleccionar el modelo de inteligencia artificial.</li> <li>○ Subir la canción de salsa.</li> <li>○ Reproducir o pausar la canción.</li> <li>○ Mostrar los resultados en un diagrama de pastel.</li> <li>○ Poder seleccionar el BPM predecido y mostrarlo en la canción.</li> <li>○ Mostrar el tiempo de ejecución del proceso de predicción.</li> <li>○ Volver a hacer la predicción.</li> </ul> </li> </ul>	
<b>Validación:</b>	
<p>La vista se desplegará de la siguiente forma.</p> <ol style="list-style-type: none"> <li>1. Se crea una vista en donde:       <ol style="list-style-type: none"> <li>a. Selecciona el modelo de Inteligencia Artificial.</li> <li>b. Sube la canción de salsa.</li> </ol> </li> <li>2. Se crea una vista donde se desplegará:       <ol style="list-style-type: none"> <li>a. La canción como onda en el dominio del tiempo.</li> <li>b. Botón de reproducir.</li> <li>c. Botón de pausar.</li> <li>d. Botón de mostrar predicciones.</li> <li>e. Mostrar tiempo de predicción de la Inteligencia artificial.</li> <li>f. Mostrar el BPM de la canción.</li> <li>g. Mostrar las líneas de los beats en la canción.</li> <li>h. Mostrar Botón de "Mostrar predicciones"</li> <li>i. Botón de "regresar" para volver a hacer una predicción.</li> </ol> </li> <li>3. Se crea una vista para mostrar las predicciones de la Inteligencia artificial:       <ol style="list-style-type: none"> <li>a. Una gráfica en forma de pastel.</li> <li>b. Mostrar la estadística de cada predicción.</li> <li>c. Si se selecciona un BPM devolverse a la vista (2).</li> </ol> </li> </ol>	

Figura 2.30: Requerimiento Específico 2.

Este desarrollo tiene el siguiente diagrama de flujo para el usuario:

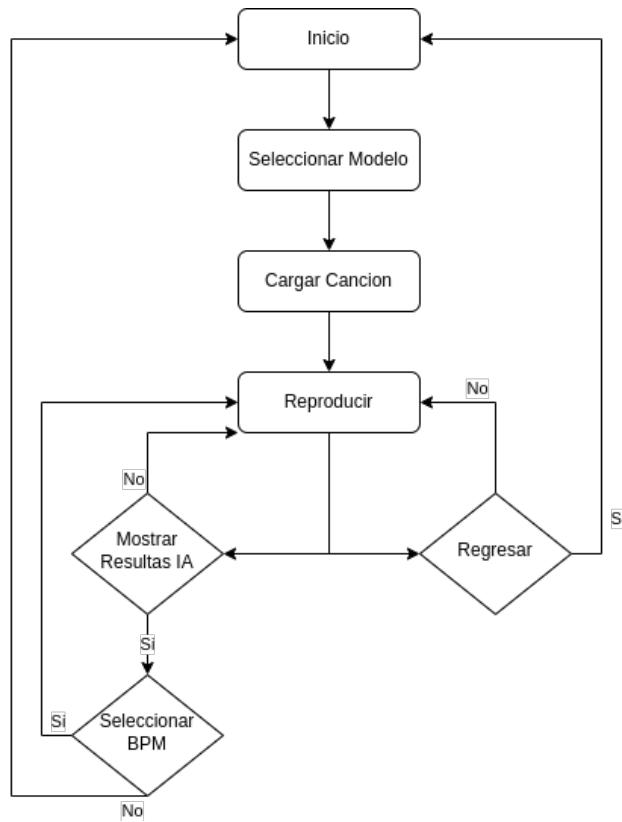


Figura 2.31: Diagrama de flujo de la vista Predicción.



## CAPÍTULO 3

# Conclusiones y trabajos futuros

---

Esto es un acercamiento para la detección de BPM en la música salsa usando redes neuronales convolucionales. Con la ayuda de las diferentes librerías, los procesos de extracción de datos y análisis de ello son cada vez más fáciles para la comunidad investigativa, ayudando a entender poco a poco como la música y la tecnología pueden estar de la mano.

No solo eso, también podemos ver el alcance que puede aprender las Inteligencias artificiales y de que no solo se reduce a aprender pequeñas datos, queda demostrado que no tiene un límite de aprendizaje, lo único que necesita son datos, un buen procesador o GPU y tiempo. En este caso, podríamos decir que aprendió a entender la música salsa, o al menos un abrebotas.

La música es un campo infinito, nunca para de crear nuevos sonidos, nuevos tonos, nuevos ritmos. Cada día se obtienen nuevas canciones, con diferentes patrones y melodías, creciendo de una manera inesperada y sin darnos cuenta todas estas canciones tienen datos interesantes detrás de ellos; deberíamos sorprendernos con la cantidad de variables y campos inexplorados de la música. En esta investigación se tocó un tema en específico de un género en específico; el BPM y la salsa.

La salsa es una viva muestra de la cultura de nuestra región, incluso en el país se usa para el entretenimiento de las personas y aun así su campo investigativo es muy pobre. ¿Quién no ha escuchado salsa? Les aseguro que todos y cada uno de los Colombianos hemos escuchado este género musical.

Su campo necesita de apoyo, pueden servir para el uso de herramientas como mezcladores de música que usan los DJ o para crear transiciones en una aplicación de reproducción de música; sus aplicaciones son infinitas.

Los modelos propuesto en esta investigación no son los únicos, hay infinidad de modelos; se requiere de más personas que continúen con el análisis de estos. Mi objetivo es dar un abrebotas a la gente que quiera apoyar y seguir innovando en este campo.

En un futuro si es posible separar cada instrumento de una canción, es posible que la detección del BPM sea más sencilla; con el cencerro puede ser posible encontrar más fácil el BPM de este género musical salsa, ya que este instrumento a veces está sincronizado literalmente al BPM de la canción.



# Bibliografía

- [1] J. H. Paz Bolaños, D. Gomez, J. Gustavo Diaz, and C. A. Arce Lopera, “MODELAMIENTO DE LA PERCEPCIÓN DEL PULSO EN LA MÚSICA SALSA,” Ph.D. dissertation, ICESI, 2019.
- [2] F. Richter, “Chart: Spotify Keeps Apple Music at Arm’s Length | Statista,” 2020. [Online]. Available: <https://www.statista.com/chart/8399 spotify-apple-music-paid-subscribers/>
- [3] Goodwater, “Understanding Spotify: Making Music Through Innovation,” p. www.goodwatercap.com, 2018. [Online]. Available: [www.goodwatercap.comhttps://www.goodwatercap.com/thesis/understanding-spotify](http://www.goodwatercap.comhttps://www.goodwatercap.com/thesis/understanding-spotify)
- [4] Televisa, “Salsa: la música latina que nació en Nueva York – Noticieros Televisa,” 2017. [Online]. Available: <https://noticieros.television.com/especiales/salsa-musica-latina-que-nacio-nueva-york/>
- [5] R. de Cande, *Nuevo Diccionario de la Música*, 202.
- [6] RAE, “músico, música | Definición | Diccionario de la lengua española | RAE - ASALE.” [Online]. Available: <https://dle.rae.es/m{ú}sico>
- [7] G. Peeters and H. Papadopoulos, “Simultaneous Beat and Downbeat-Tracking Using a Probabilistic Framework: Theory and Large-Scale Evolution,” vol. 19, no. 6, pp. 1–17, 2011.
- [8] S. Stanley and J. Tyrrell, *The Grove Dictionary of Music and Musicians*, 2001.
- [9] A. P. Klapuri, A. J. Eronen, and J. T. Astola, “Analysis of the meter of acoustic musical signals,” *IEEE Transactions on Audio, Speech and Language Processing*, vol. 14, no. 1, pp. 342–355, 2006.
- [10] M. Goto and Y. Muraoka, “Real-time Rhythm Tracking for Drumless Audio Signals - Chord Change Detection for Musical Decisions -,” *Change*, pp. 135–144, 1996.
- [11] S. Dixon, “Evaluation of Audio Beat Tracking System BeatRoot,” *Preprint for Journal of New Music Research*, 2007. [Online]. Available: <http://www.eecs.qmul.ac.uk/{~}simond/pub/2007/jnmr07.pdf{%}0Ahttp://www.tandfonline.com/doi/pdf/10.1080/09298210701653252>
- [12] E. D. Scheirer, “Tempo and beat analysis of acoustic musical signals,” *The Journal of the Acoustical Society of America*, vol. 103, no. 1, pp. 588–601, 1998.
- [13] T. Jehan, “Creating music by listening,” *Media Arts and Sciences*, vol. PhD, 2005.
- [14] M. Goto and Y. Muraoka, “A Real-time Beat Tracking System for Audio Signals,” *Workshop on Computational Auditory Scene Analysis Music*, pp. 68–75, 1995.

- [15] U. Marchand and G. Peeters, “The Extended Ballroom Dataset,” *ISMIR 2016 Late-Breaking Session*, pp. 1–3, 2016.
- [16] R. Sathya and A. Abraham, “Comparison of Supervised and Unsupervised Learning Algorithms for Pattern Classification,” *International Journal of Advanced Research in Artificial Intelligence*, vol. 2, no. 2, pp. 34–38, 2013.
- [17] A. Burkov, *The Hundred-Page Machine Learning Book*. Andriy Burkov, 2019.
- [18] O. Pentakalos, “Introduction to machine learning,” in *CMG IMPACT 2019*, apr 2019, pp. 1–10. [Online]. Available: <https://heartbeat.fritz.ai/introduction-to-machine-learning-model-evaluation-fa859e1b2d7f>
- [19] S. Dixon, “Automatic extraction of tempo and beat from expressive performances,” *International Journal of Phytoremediation*, vol. 21, no. 1, pp. 39–58, 2001.
- [20] G. Peeters, “Time variable tempo detection and beat marking,” *International Computer Music Conference, ICMC 2005*, no. 1, 2005.
- [21] H. Schreiber, “Data-Driven Approaches for Tempo and Key Estimation of Music Recordings,” no. January, 2020.
- [22] H. Purwins, B. Li, T. Virtanen, J. Schlüter, S. Y. Chang, and T. Sainath, “Deep Learning for Audio Signal Processing,” *IEEE Journal on Selected Topics in Signal Processing*, vol. 13, no. 2, pp. 206–219, 2019.
- [23] J. Richter, “Style-Specific Beat Tracking with Deep Neural Networks Improving Polyphonic Piano Learning,” no. June, 2019.
- [24] B. McFee, C. Raffel, D. Liang, D. P. W. Ellis, M. McVicar, E. Battenberg, and O. Nieto, “librosa: Audio and music signal analysis in python,” in *Proceedings of the 14th python in science conference*, vol. 8, 2015.
- [25] C. R. Harris, K. J. Millman, S. J. van der Walt, R. Gommers, P. Virtanen, D. Cournapeau, E. Wieser, J. Taylor, S. Berg, N. J. Smith, R. Kern, M. Picus, S. Hoyer, M. H. van Kerkwijk, M. Brett, A. Haldane, J. F. del Río, M. Wiebe, P. Peterson, P. Gérard-Marchant, K. Sheppard, T. Reddy, W. Weckesser, H. Abbasi, C. Gohlke, and T. E. Oliphant, “Array programming with NumPy,” *Nature*, vol. 585, no. 7825, pp. 357–362, sep 2020.
- [26] J. D. Hunter, “Matplotlib: A 2D graphics environment,” *Computing in science & engineering*, vol. 9, no. 3, pp. 90–95, 2007.
- [27] “dB: What is a decibel?” [Online]. Available: <https://www.animations.physics.unsw.edu.au/jw/dB.htm>
- [28] RAE, “Definición hercio.” [Online]. Available: <https://dle.rae.es/hercio>

- [29] D. Gartzman, “Getting to Know the Mel Spectrogram,” 2019. [Online]. Available: <https://towardsdatascience.com/getting-to-know-the-mel-spectrogram-31bca3e2d9d0>
- [30] A. N. Akansu and R. A. Haddad, “Time-Frequency Representations,” *Multiresolution Signal Decomposition*, pp. 331–390, 2001.
- [31] L. Roberts, “Understanding the Mel Spectrogram,” mar 2020. [Online]. Available: <https://medium.com/analytics-vidhya/understanding-the-mel-spectrogram-fca2afa2ce53>
- [32] D. Cazzani, “Audio processing in TensorFlow. An implementation of the Short Time Fourier Transform,” jan 2017. [Online]. Available: <https://towardsdatascience.com/audio-processing-in-tensorflow-208f1a4103aa>
- [33] D. Schwerfeger, “How to Easily Process Audio on Your GPU with TensorFlow,” mar 2020. [Online]. Available: <https://towardsdatascience.com/how-to-easily-process-audio-on-your-gpu-with-tensorflow-2d9d91360f06>
- [34] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard, and Others, “Tensorflow: A system for large-scale machine learning,” in *12th Symposium on Operating Systems Design and Implementation*, 2016, pp. 265–283.
- [35] I. Shafkat, “Intuitively Understanding Convolutions for Deep Learning .” [Online]. Available: <https://towardsdatascience.com/intuitively-understanding-convolutions-for-deep-learning-1f6f42faee1>
- [36] J. Torres Viñals, *Deep Learning Introducción Prácticas Con Keras*, 2018. [Online]. Available: <https://torres.ai/deep-learning-inteligencia-artificial-keras/>
- [37] D. A. Clevert, T. Unterthiner, and S. Hochreiter, “Fast and Accurate Deep Network Learning by Exponential Linear Units (ELUs),” *4th International Conference on Learning Representations, ICLR 2016 - Conference Track Proceedings*, nov 2015. [Online]. Available: <https://arxiv.org/abs/1511.07289v5>
- [38] J. Brownlee, “How to Choose Loss Functions When Training Deep Learning Neural Networks,” jan 2019. [Online]. Available: <https://machinelearningmastery.com/how-to-choose-loss-functions-when-training-deep-learning-neural-networks/>
- [39] A. H. Jackson, *Machine learning*, 1988, vol. 5, no. 2.
- [40] D. P. Kingma and J. L. Ba, “Adam: A method for stochastic optimization,” in *3rd International Conference on Learning Representations, ICLR 2015 - Conference Track Proceedings*, 2015.
- [41] X. Pons, Jordi; Serra, “DESIGNING EFFICIENT ARCHITECTURES FOR MODELING TEMPORAL FEATURES WITH CONVOLUTIONAL NEURAL NETWORKS Jordi Pons and Xavier Serra Music Technology Group , Universitat Pompeu Fabra , Barcelona,” *IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP) 2017*, pp. 2472–2476, 2017.

- [42] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, “Going deeper with convolutions,” in *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015, pp. 1–9.
- [43] M. Grinberg, *Flask web development: developing web applications with python.* .O'Reilly Media, Inc.", 2018.
- [44] N. Jain, A. Bhansali, and D. Mehta, “AngularJS: A modern MVC framework in JavaScript,” *Journal of Global Research in Computer Science*, vol. 5, no. 12, pp. 17–23, 2014.
- [45] V. Saiz, “wavesurfer.js,” 2014. [Online]. Available: <https://wavesjs.github.io/>
- [46] D. Li, H. Mei, Y. Shen, S. Su, W. Zhang, J. Wang, M. Zu, and W. Chen, “ECharts: A declarative framework for rapid construction of web-based visualization,” *Visual Informatics*, vol. 2, no. 2, pp. 136–146, jun 2018.