Universidad Nacional Autónoma de México

Programa de Maestría y Doctorado en Música

Facultad de Música

Instituto de Ciencias Aplicadas y Tecnología

Instituto de Investigación Antropológica

Sistema Interactivo para el violín.
Detección de las técnicas de arco *détaché, martelé* y *spiccato.*

Tesis para optar el grado de Maestro en Música

(Tecnología Musical)

Presenta:

Beatriz Adriana Botello Castillo

Tutor:

Dr. Cristian Manuel Bañuelos Hinojosa
Programa de Maestría y Doctorado en Música

Ciudad de México, Enero, 2020

Se agradece el apoyo recibido para la realización de esta tesis por parte del proyecto PAPIME PE405018 "Desarrollo e implementación didáctica de recursos informáticos y de cómputo para apoyar la enseñanza de la Música".

AGRADECIMIENTOS

A mi esposo, Alberto Hernández, por apoyarme a lo largo de este trayecto e impulsarme a culminarlo. Gracias por siempre lograr que dé más de mí y confiar en mi capacidad. Eres mi inspiración.

A mi tutor, el Dr. Cristian Bañuelos, por nunca abandonarme y enseñarme todo lo necesario para lograr esta tesis.

Al Dr. Felipe Orduña quien siempre estuvo con Cristian y conmigo brindándonos su tiempo, apoyo y dedicación.

A Andrea Zamora, mi lectora fiel, quien estuvo a mi lado en los momentos de estrés y me apoyó en la elaboración de este texto.

A Amado Álvarez por poner su granito de arena en este proyecto.

A Enrique Palma quien siempre estuvo dándome consejos y apoyándome para llegar a la meta.

A Jasmin Ocampo quien siempre estuvo apoyándome y guiándome en este proceso.

ÍNDICE GENERAL

ÍNDICE DE FIGURAS

LISTA DE CÓDIGOS

1

INTRODUCCIÓN

La presente tesis tiene como objetivo principal abordar el proceso mediante el cual se desarrolló una metodología de clasificación de sonidos del golpe de arco de violín por medio de un herramienta computacional responsiva que se adapta a dispositivos móviles y de escritorio. Esta permite el reconocimiento de tres golpes de arco; la metodología para su creación implicó la realización de varios experimentos para obtener una clasificación por medio de archivos de audio, estudiar la señal, obtener descriptores de ella, segmentarla y para clasificarla utilizar métodos de aprendizaje automático.

En función de la delimitación y viabilidad del proyecto, se propone un sistema interactivo que puede ayudar a violinistas de nivel intermedio[1] al reconocimiento de tres tipos de golpes de arco: *dé-*

1 En efectos de este trabajo de investigación, nivel intermedio serán los alumnos que se encuentren estudiando el propedéutico en la FaM o en su defecto que tengan conocimiento de la técnica del instrumento, que puedan ejecutar diferentes

taché, martelé y spiccato. Esto se logró creando una webapp con aprendizaje automático que busca ser una herramienta que pueda brindar apoyo a los violinistas para reconocer, mejorar y optimizar las técnicas de arco antes mencionadas.

Es importante mencionar que el desarrollo de una aplicación para el estudio de un instrumento musical presenta todo un reto tanto en la elaboración como en la aceptación. Consecuentemente, realizar un software para el violín condujo a las siguientes preguntas de investigación: ¿cuál es la precisión de detección de un golpe de arco en un sistema interactivo y cómo contribuye a mejorar la relación cuerpo-sonido del intérprete?

Las respuestas a dichas interrogantes se ven reflejadas a lo largo del trabajo pero es importante recalcar desde ahora que el diseño de la webapp mantuvo como eje central estas intenciones.

Por otro lado, surgieron otras preguntas que mantienen relación directa con el desarrollo de un sistema interactivo y de este trabajo de investigación: ¿a qué rango de edades va dirigido el sistema interactivo?, ¿qué tipo de repertorio es el adecuado?, ¿cuál sería la interfaz ideal?, ¿qué lenguaje de programación es el adecuado?

Todas estas interrogantes se van contestando una a una a lo largo de los capítulos, pero se concluyó que más que un rango de edad, este sistema interactivo va dirigido a violinistas con un nivel

---

tipos de golpe de arco y que cuente con un repertorio de música de concierto intermedio.

intermedio, ya que cuentan con los conocimientos suficientes de la técnica en el violín, así como de los golpes de arco a los cuáles está enfocada esta investigación (*détaché, martelé y spiccato*).

La selección del repertorio adecuado para trabajar en la presente tesis fue un tema complejo. Esto debido a que ciertos métodos de estudio requerían de una licencia de uso como el Kayser y col. 1915 y Kreutzer 1915; por ello, la solución más viable fue escribir mis propios ejercicios de estudio.

Adentrándose en esta investigación se encontraron diversas aplicaciones de cómputo como *Cortosia, Music Wrench, Music jacket, An interface for real-time classification of articuations produced by violin bowing, Effort-based analysis of bowing movements: evidence of anticipation effects.*. Las dos primeras son comerciales, esto quiere decir que se encuentran en *Google Play* o *Apple Store* para poder ser descargadas en un dispositivo móvil y poder usarlas en cualquier momento. Las demás aplicaciones han sido desarrolladas en laboratorios para investigaciones del funcionamiento del instrumento.

Considerando los resultado obtenidos de la investigación sobre aplicaciones de cómputo enfocadas al violín, se propuso el siguiente aporte: si existen aplicaciones que apoyan a la afinación con la detección de la frecuencia, entonces también podrían existir trabajos específicos para el arco.

En la búsqueda por la resolución de ese pensamiento se encontró el estímulo tecnológico para esta tesis, siendo éste el trabajo de Young 2008, el cual habla sobre la clasificación de diferentes golpes de arco (*accented detaché, detaché lancé, louré, martelé, staccato, spiccato*) utilizando un sistema de medición llamado *Hyperbow* desarrollado por Young en 2003.

Tomando como base el trabajo de Young 2008 se propuso una metodología de desarrollo. Con ello se busca que los resultados obtenidos sean de ayuda para los violinistas, para que tengan una herramienta que ayude en el reconocimiento de los golpes de arco tratados en este trabajo.

Se comenzó por escoger tres tipos de golpe de arco y el lenguaje de programación con el que se trabajaría (*Python*). Después, se escribieron estudios para cada tipo de golpe, teniendo los estudios se realizaron grabaciones a tres violinistas con la intención de obtener las muestras de audio que posteriormente sirvieron para el desarrollo y la evaluación de la aplicación.

A dichas muestras se les aplicaron diferentes descriptores de audio de la biblioteca *librosa* (McFee y col. 2015) para obtener datos que pudieran ayudar a clasificar los golpes de arco y poder entrenar un sistema con aprendizaje automático.

Para el aprendizaje automático se experimentó con modelos computacionales de agrupamientos. Se usó el aprendizaje automático de-

bido a que es una forma sencilla de obtener un clasificador, ya que al sistema se le proporcionan varias muestras similares, las cuáles se van almacenando y aprendiendo mediante algoritmos y sistemas estadísticos.

El resultado final permite el reconocimiento de los patrones aprendidos, es decir, si toco un pasaje con algún golpe de arco, el sistema es capaz de detectar si fue interpretado en *détaché, martelé y spiccato*.

El capítulo 2 aborda de manera general los antecedentes de esta tesis. En principio se presenta el marco teórico trabajado en esta investigación, dando paso a la explicación del análisis de las señales de audio, mencionando los procedimientos que se utilizaron en este trabajo.

Asimismo, se explican las técnicas de aprendizaje automático, se enumeran y describen diversas técnicas compatibles con el análisis necesario para este trabajo y se dan ejemplos de algunos trabajos de música, en específico del violín, que utilizaron estas técnicas. También se analizan algunas plataformas de implementación con distintos lenguajes de programación como *JavaScript* para un sistema web y *Python* para una aplicación de escritorio.

Finalmente, se presenta un estado de la cuestión donde se ejemplifican algunas aplicaciones que abordan un tema similar al del presente objeto de estudio.

En el capítulo 3 se describe la metodología para el desarrollo del sistema interactivo. También se explica cómo se entrenó el sistema, para lo cual se necesitaron grabaciones de partituras que se escribieron específicamente para este trabajo de investigación. A su vez, se presentan los fragmentos de código que se desarrollaron y con los que se experimentó, describiendo por qué se desecharon o por qué se considera que tienen el funcionamiento necesario para el desarrollo de la webapp.

En el mismo capítulo se exponen también los códigos de entrenamiento automático explicando las características del sistema elegido en este trabajo. Por último, se presentan los resultados técnicos que se obtienen al momento de ser utilizado el sistema por un violinista.

En el capítulo 4 se describe la presentación en términos musicales de los resultados que se obtienen del capítulo anterior, de modo que los violinistas puedan aproximarse a la información que produce la webapp de una manera más convencional, es decir, a través de una interfaz de usuario.

2

ANTECEDENTES

En este capítulo se aborda el estado de la investigación, el por qué del desarrollo de un sistema interactivo para el violín, los aportes, las diferencias y las mejoras en los trabajos académicos y en el mercado de software, los métodos usados en la aplicación, así como la selección del lenguaje de programación y las implicaciones tecnológicas.

2.1 MARCO TEÓRICO

Hace aproximadamente veinte años comenzó la investigación de la técnica del arco orientada a la tecnología en los trabajos de Young 2002; Young 2008; Young y Serafin 2007, creando sus propios sistemas de medición para el arco como *Hyperbow*, en el cual se utilizan sistemas de sensores conectados al arco y un software

para medir las grabaciones. Este campo no ha sido desarrollado lo suficiente debido a la complejidad de la interacción entre la cuerda y el golpe de arco.

N. H. Rasamimanana, Fléty y Bevilacqua 2005 nos dicen que algunos trabajos tecnológicos de reconocimiento confiable de las técnicas de golpe de arco en conjunto de la técnica de la mano derecha (posición, fuerza) aportan beneficios para los músicos de cuerda frotada y que estos trabajos pueden ser aplicados en la pedagogía musical o en el diseño de interfaces musicales novedosas.

El elemento principal que se trabajó en este sistema interactivo es el timbre. El timbre es la cualidad del sonido que permite diferenciar una fuente sonora de otra y depende de la cantidad de armónicos que contenga, de la intensidad y sobre todo la evolución en el tiempo de cada uno (Gutiérrez 1985). En los instrumentos de cuerda frotada, el timbre se genera con la mano derecha al pasar el arco por las cuerdas, esta acción debe mantener una aceleración uniforme para obtener un sonido uniforme.

De acuerdo a la pedagogía tradicional académica del violín el paso del arco por las cuerdas se enseña tocando cuerdas sueltas pasando el arco completo por la cuerda a diferentes velocidades, sumándole cambios de cuerda para lograr que las vibraciones sean lo más uniformes posibles (Galamian 1998). Este tipo de ejercicios suelen ser los primeros de cada método de estudio, pero es algo

que se debe estudiar a diario durante toda la vida de violinista a modo de calentamiento antes de empezar con las escalas, de esa manera se puede lograr un sonido uniforme (Crickboom 1994).

Para lograr lo anterior, se necesita una buena coordinación psicomotriz. Primero se debe lograr una relajación en los músculos del brazo derecho para poder usar su peso natural y transmitirlo al arco para ejercer la presión sobre las cuerdas de forma uniforme y constante para generar sonido (Galamian 1998).

Después de controlar esto, se debe aprender que la presión ejercida sobre el arco no siempre será la misma durante toda una pieza, ya que existen diferentes tipos de golpes de arcos que ocupan diferente presión o modo de pasar el arco sobre las cuerdas, también las dinámicas entran en este juego con los niveles de intensidad de la fuerza ejercida por el arco sobre la cuerda.

En la música denominada como "clásica"[1] también se utilizan diferentes técnicas de ejecución aunque se refiera a un mismo golpe de arco.

La velocidad del arco, la fuerza de presión ejercida sobre las cuerdas, la distancia respecto al puente y la inclinación en las cuerdas son parámetros que se ocupan para generar el timbre que se desea (Schoonderwaldt 2009).

1 En este trabajo se denomina música "clásica" a la música académica no referida al periodo clásico exclusivamente. Este tipo de música es mal denominada así de forma genera.l

Como parte del estudio de los tipos de sonido generados en el violín, en esta tesis se analizarán tres tipos de golpe de arco básicos: *détaché, spiccato y martelé*. Esto con el fin de obtener un sonido limpio, es decir, sin ruido al momento de la ejecución. A continuación se describirán estos golpes de arco:

Spiccato. De acuerdo con (Ferreira y Ray 2006), este golpe de arco parte desde fuera de la cuerda, tocándola brevemente volviendo a la posición inicial (fuera de la cuerda). Puede ser tocado en cualquier parte del arco. El movimiento del arco debe ser semi circular a la cuerda para tener equilibrados los componentes horizontal y vertical (Figura 1).

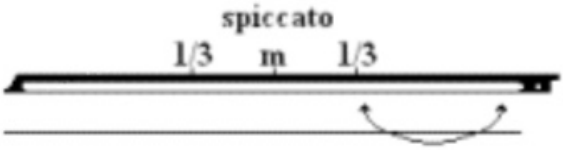


Figura 1.: Movimiento del Arco del *Spiccato*. (Ferreira y Ray 2006)

Cuando el movimiento es lento se ejecuta en la parte inferior del arco, conforme va aumentando la velocidad se va recorriendo el arco a la mitad. La notación gráfica del *spiccato* es un punto debajo o encima de la nota (Figura 2).

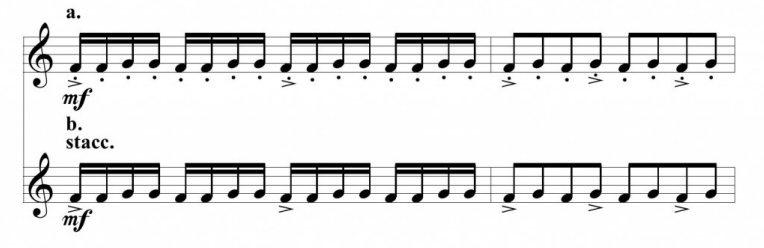


Figura 2.: Notación del *Spiccato*. (Ferreira y Ray 2006)

Détaché. Es el golpe de arco más común del repertorio orquestal, cada nota corresponde a un nuevo movimiento del arco en dirección contraria a la nota que precede. Es más común que se ejecute en la mitad superior aunque puede ser tocado en cualquier parte del arco. Puede dar origen a otros golpes de arco según el aumento de la presión, forma de ataque y cantidad de arco que se ocupa para ejecutarlo. Mientras más rápida sea la velocidad del arco sobre la cuerda menor será la cantidad de arco que se utilizará (Ferreira y Ray 2006). Dourado 2008 dice que contrariamente al *legato*, en el *détaché* cada nota es ejecutada en una arcada por el movimiento del antebrazo. Existen diferentes tipos de *détaché*: *grand-détaché*, *détaché* acentuado, *détaché* con todo el arco (Figura 3) En este trabajo de investigación sólo se analizará *détaché* con todo el arco.

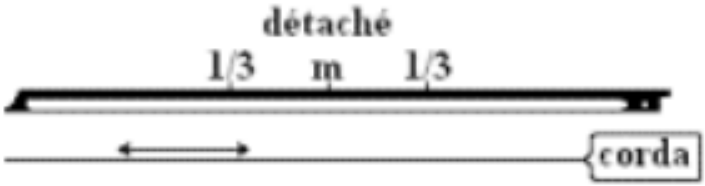


Figura 3.: Movimiento del Arco *Détache*. (Ferreira y Ray 2006)

Martelé. En este golpe de arco cada nota es seguida por un gran acento inicial y con pausas entre ellas. Este acento es el resultado de la pronación del antebrazo realizada antes del movimiento horizontal del antebrazo produciendo un ataque de tipo *sforzando*. Galamian 1998 dice que el *martelé* puede ser ejecutado con cualquier cantidad de arco y en cualquier región, su uso más común se da en la punta (Figura 4). Para lograr este golpe de arco la muñeca debe ser posicionada un poco más abajo de lo normal, y el antebrazo ligeramente pronado, lo que permite que la base de los dedos (articulación metacarpo-falange) quede más baja.

La notación gráfica del *martelé* es una cuña encima o debajo de la nota (Figura 5).

En la práctica orquestal los golpes de arco deben ser hechos de manera uniforme entre todos los músicos para lograr una conjun-

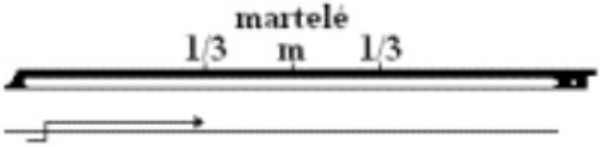


Figura 4.: Movimiento del Arco del *Martelé*. (Ferreira y Ray 2006)

ción en los pasajes de las obras interpretadas, por eso las arcadas[2] deben ser ejecutadas de manera segura. Ningún tipo de arcada es la verdad absoluta, estas cambian dependiendo de la agrupación, o del director según el sentido que quiere darle a la obra. A continuación se mostrarán ejemplos de algunos pasajes orquestales, los cuales contienen los golpes de arcos estudiados en este trabajo, para ejemplificar el uso de cada uno de ellos.

El primer ejemplo es un pasaje de la Ópera de Los Maestros Cantores de Nuremberg de Wagner (Figura 6). Este pasaje comunmente se toca en *détaché* usando toda la longitud del arco en los cuartos o negras considerando que se tocan dobles cuerdas. Debe ser ejecutado un poco acentuado por su carácter vigoroso y su di-

2 Se denomina arcada al paso del arco sobre la cuerda para producir sonido en los instrumentos de cuerda frotada

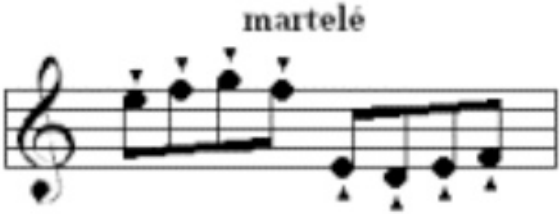


Figura 5.: Notación de *Martelé* . (Ferreira y Ray 2006)

námica en *forte*. En el tercer compás se indica una reposición de arco hacia abajo la cual debe tener la misma fuerza e intensidad sonora que las notas anteriores. Para mantener un sonido en *forte* es recomendable tocar cerca del puente (Ferreira y Ray 2006).

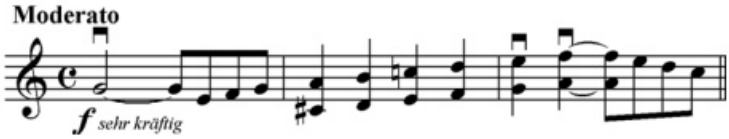


Figura 6.: Wagner, Preludio de la Ópera de Los Maestros Cantores de Nuremberg, parte de violín 1, compases del 1 al 4. (Ferreira y Ray 2006)

En el siguiente ejemplo se demostrará el *martelé* con un pasaje de la Quinta Sinfonía de Beethoven, (Figura 7). Cada nota acentuada necesita un impulso inicial, con una velocidad rápida y de-

creciendo la velocidad inmediatamente después del ataque. Para poder ser ejecutado en fortísimo como lo indica el pasaje, se debe tocar cerca del puente (Ferreira y Ray 2006).

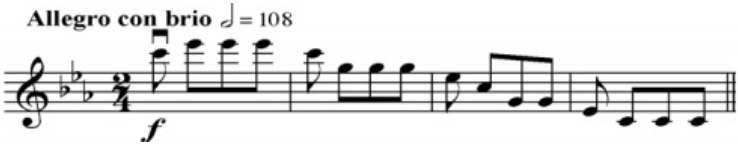


Figura 7.: Beethoven, Sinfonía 5, 1er movimiento, parte de violín 1, compases del 44 al 47. (Ferreira y Ray 2006)

Finalmente se ejemplificará el *spiccato* con la *Suite Masques et Bergamasques* de Fauré (8). En este pasaje en la mayor parte de las figuras rítmicas conocidas como dieciseisavos se usa el *spiccato* (Ferreira y Ray 2006).

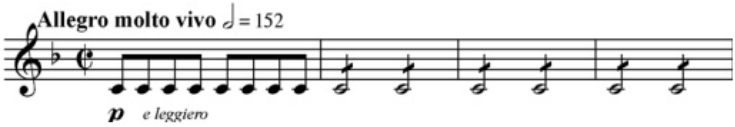


Figura 8.: Fauré, *Suite Masques et Bergamasques*, Overtura, parte de violín II, compases del 1 al 4. (Ferreira y Ray 2006)

2.1.1 *Análisis de señales de audio*

Para analizar los golpes de arco, utilizamos técnicas de análisis de señales de audio para crear el sistema interactivo desarrollado en este trabajo mediante diversas técnicas de codificación. A diferencia de otros trabajos que se mencionarán más adelante, en este trabajo se hicieron grabaciones de estudios escritos por mí para cada tipo de golpe de arco, estas grabaciones fueron analizadas mediante señales de audio y descriptores, obteniendo gráficas y archivos de audio para lograr los objetivos de este sistema.

Pedro 1990, define al timbre como la combinación de tres parámetros: espectro, envolvente dinámica y formante. El espectro de frecuencia es la distribución de la energía en función de los parciales (armónicos o inarmónicos) de un sonido complejo. La envolvente dinámica es la variación de la amplitud en el dominio del tiempo. La formante es el pico de intensidad o concentración energética en una determinada frecuencia en el espectro de un sonido.

Para la física, el timbre es la cualidad que da al sonido los armónicos que acompañan a la frecuencia fundamental generando variaciones en la onda sinusoidal fundamental. El teorema de Fourier demuestra que cualquier forma de onda periódica puede descomponerse en una serie de ondas (armónicos) que tiene una fre-

cuencia que es múltiplo de la frecuencia de la onda original (frecuencia fundamental), es decir, los armónicos son múltiplos de la frecuencia fundamental (Roederer 2008).

El timbre se determina por la cantidad e intensidad de armónicos. Los armónicos varían según la fuente, el tipo de instrumento, el diseño del propio instrumento y la forma de ejecutarlo. A veces, como en el caso del oboe, los armónicos pueden tener una amplitud igual o superior a la forma de onda fundamental (Roederer 2008).

Descriptores del timbre

A continuación se enuncian los descriptores de timbre. Estos son los caracterízadores que permitieron obtener los datos deseados y lograr el proceso de aprendizaje automático para completar el software interactivo.

Los descriptores de timbre contienen las características de percepción del sonido de un instrumento. Dentro de los descriptores de timbre podemos distinguir dos tipos: los temporales y espectrales. A su vez los primeros se dividen *long-attack time* (LAT) y centroide temporal (TC). Mientras que los segundos, los espectrales de timbre, se dividen en: Centroide armónico espectral (HSC), desviación del armónico espectral (HSD), propagación del armó-

nico de espectral (HSS), variación del armónico espectral (HSV) y centroide espectral (SC), Manjunath, Salembier y Sikora 2002.

A continuación se mencionarán los descriptores utilizados en este trabajo:

Zero crossing rate

En el ámbito de las señales de tiempo discreto, se dice que ocurre un cruce por cero si las muestras sucesivas tienen diferentes signos algebraicos. La velocidad a la que se producen los cruces por cero es una medida simple del contenido de frecuencia de una señal, es el número de veces en un intervalo de tiempo dado donde la amplitud de las señales pasa por un valor de cero.

En las señales de voz que son de banda ancha la interpretación de la velocidad de cruce por cero es mucho menos preciso (Figura 9). Sin embargo, se pueden obtener estimaciones aproximadas de las propiedades espectrales utilizando una representación basada en la tasa promedio de cruce por cero a corto plazo (Shete, SB Patil y Patil 2014). Existen trabajos como el de Gouyon, Pachet, Delerue y col. 2000 en el que se usa esta técnica para la clasificación de sonidos percutidos.

Mel spectogram

La escala mel, propuesta por Stevens, Volkman y Newmann en 1937, es una escala musical perceptual de tonos vistos como intervalos equiespaciados. El punto de referencia entre esta escala y la

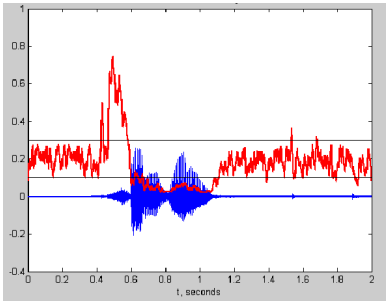


Figura 9.: *Zero Crossing Rate*. (Shete, SB Patil y Patil 2014)

frecuencia normal se define equiparando un tono de 1000 Hz, 40 dB por encima del umbral de audición del oyente, con un tono de 1000 mels.

Por encima de 500 Hz, los intervalos de frecuencia espaciados exponencialmente son percibidos como si estuvieran espaciados linealmente. $m = 1127{,}01048 \log_e(1 + f/700)$, (Bedoya-Jaramillo y col. 2012).

Villa-Canas y col. 2012 dice que el espaciado entre frecuencias de mel se asemeja a las frecuencias que se reciben en los nervios de la cóclea (Figura10). Así que agrupar un espectro en frecuencias de mel sirve para usar información espectral de la misma manera que la audición humana. Esto es útil para el reconocimiento de voz. Sakashita y Aono 2018 en su trabajo usa *Mel Spectrogram* para clasificar mediante redes neuronales diferentes tipos de audios.

Mel-frequency cepstral coefficients (MFCC)

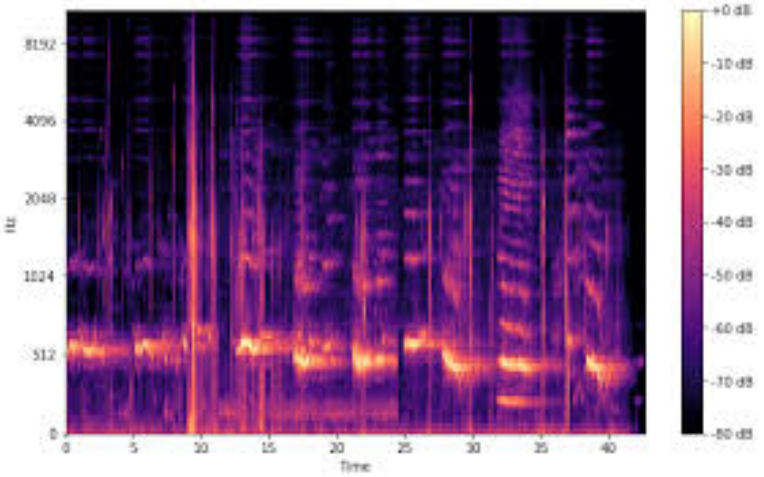


Figura 10.: *Mel Spectogram*

En procesamiento de audio *Mel-frequency cespstral* es una representación del espectro de potencia de un sonido en un lapso corto, basado en una transformada de coseno lineal de un espectro de potencia de registro en una escala de frecuencia no lineal de mel (Xu y col. 2004). *Mel-Frequency Cepstral Coefficients (MFCC)* son coeficientes que unidos forman un MFC, estos se derivan de un tipo de representación cepstral del audio (Figura 11).

La diferencia entre el *cepstrum* y el *cepstrum mel-frequency* es que en la MFC, las bandas de frecuencia están igualmente espaciadas en la escala de mel, como resultado su aproximación es más a la respuesta del sistema auditivo humano mientras que en

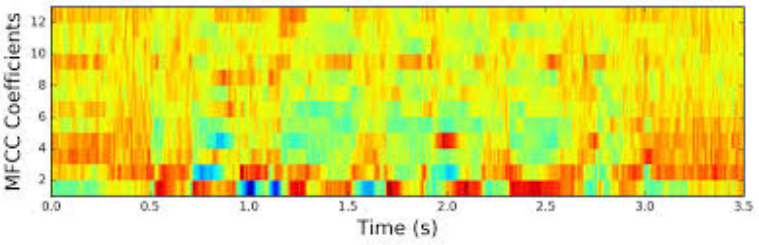


Figura 11.: *Mel-Frequency Cesptral Coefficients*

el *cepstrum* las bandas de frecuencia son espaciadas linealmente. Esta distorsión lineal de frecuencia puede permitir una mejor representación del sonido en la compresión de audio (Zheng, Zhang y Song 2001). También es utilizado para el reconocimiento de voz, pero existen trabajos como el de Logan y col. 2000 que muestran que el *Mel-Frequency Cepstral Coefficients (MFCC)* también es utilizado para modelar música.

Spectral centroid

Se asocia comúnmente con la medida del brillo de un sonido. Esta medida se obtiene al evaluar el centro de gravedad utilizando la frecuencia y la magnitud de la transformada de Fourier. El centroide individual de un cuadro espectral se define como la frecuencia promedio ponderada por las amplitudes, dividida por la suma de las amplitudes (Peeters 2004).

donde x(n) representa el valor de frecuencia ponderado, o la magnitud, del número n y f(n) es la frecuencia central.

$$\text{Centroid} = \frac{\sum_{n=0}^{N-1} f(n)x(n)}{\sum_{n=0}^{N-1} x(n)}$$

Este descriptor ayuda a obtener el color del timbre o frecuencia analizada. Schubert, Wolfe y Tarnopolsky 2004 lo utiliza para generar texturas en distintos instrumentos musicales.

Técnicas espectrales basadas en muestras

Las técnicas espectrales de muestras que fueron ocupadas en este trabajo son las grabaciones hechas por violinistas para entrenar el sistema. Tienen la ventaja de ofrecer una buena calidad de sonido inherente a las grabaciones, pero son más flexibles para permitir un mayor control de las muestras. Existe una gran variedad de estas técnicas dependiendo del uso de los modelos y transformaciones espectrales que se requieran.

En su trabajo, Bonada y Serra 2007 proponen muestrear fragmentos largos y musicalmente significativos en lugar de notas individuales para la síntesis de la voz cantada, denominada muestreo de desempeño. Esta misma técnica se puede aplicar para cualquier tipo de muestra, ya que teniendo modelos más grandes se puede tener una mejor síntesis y obtención de datos que pueden ser transformados en el dominio espectral.

A continuación se enumeran las técnicas espectrales usadas en el presente trabajo:

Feature extraction (Extracción de descriptores)

Se usa *machine learning* (aprendizaje automático)[3] para el reconocimiento de patrones y procesamiento de imágenes y datos en general. La extracción de descriptores comienza con un conjunto inicial de datos medidos, el resultado es la construcción de valores derivados (descriptores) con la intención de ser informativos y no redundantes. Esto facilita el posterior aprendizaje y generalización de los pasos.

La extracción de descriptores es una reducción de dimensión, donde un conjunto inicial de variables se acorta a grupos más manejables para su procesamiento. Los descriptores seleccionados contienen la información relevante de los datos de entrada para poder entrenar un sistema. Para la detección de cada golpe de arco en esta investigación, se utilizó la extracción de descriptores de la biblioteca *librosa* (McFee y col. 2015), programada en *Python*. A continuación se describirán las extracciones de descriptores usadas en este trabajo.

3 Es el estudio científico de algoritmos y modelos estadísticos que los sistemas informáticos utilizan para realizar de manera efectiva una tarea específica sin utilizar instrucciones explícitas, se basan en patrones e inferencias Mitchell 1997.

Spectral bandwith

El ancho de banda espectral se define como el ancho de la banda en la mitad del pico máximo (o ancho completo en la mitad máxima). El ancho de banda espectral del instrumento siempre será más estrecho que el ancho de corte espectral (figura 12) (Keppy y Allen 2016). Es usado generalmente en el análisis de imágenes.

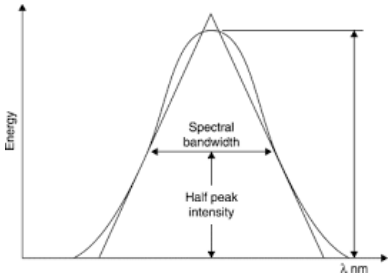


Figura 12.: *Spectral bandwith*. (Keppy y Allen 2016)

Tempograma

Un tempograma es la representación del tiempo que dura una señal de audio, distribuida de tal manera que indique la variación del tiempo dado un retraso en específico.

La construcción de un tempograma se puede dividir en dos partes: la primera que es la etapa de *onset detection* que se caracteriza por una serie de eventos musicales que constituyen el contenido rítmico básico del audio; y la segunda que es la estimación del tiem-

po mediante la autocorrelación o la transformada de Fourier de la función de detección de inicio (Grosche, Müller y Kurth 2010).

Tian y col. 2015 usan el tempograma para describir el contenido de la música en señales de tiempo. Utilizan una colección de características novedosas basadas en tempogramas que están inspiradas en hipótesis musicológicas sobre la relación entre la estructura de la música y sus componentes rítmicos prominentes en diferentes niveles métricos.

Onset detection

Puede funcionar en el dominio de tiempo, de frecuencia o de fase. Los resultados que arroja son: incrementos en la energía espectral, cambios en la distribución de energía espectral (flujo espectral) o fase, cambios en el tono detectado, patrones espectrales reconocibles por técnicas de aprendizaje automático y detección de aumentos en la amplitud en el dominio del tiempo. Si se tiene una muestra pequeña se puede tener una detección más confiable (Bello y col. 2005).

En su trabajo, Klapuri 1999 diseñó un sistema que puede detectar los inicios perceptivos de los sonidos en las señales acústicas. Se encontró que era un sistema robusto para diferentes tipos de señales. Propuso un método que puede determinar el comienzo de los sonidos que exhiben imperfecciones de inicio, es decir, la

envolvente de amplitud la cual no aumenta. El rendimiento del sistema se validó aplicándolo a la detección de inicios en señales musicales que van desde rock hasta grabaciones de big band y música clásica.

2.1.2 *Técnicas de aprendizaje automático*

En esta sección se presentarán las técnicas de aprendizaje automático supervisado que se usaron para entrenar el sistema interactivo planteado en la tesis.

Las técnicas de aprendizaje automático son modelos de inteligencia artificial utilizados para entrenar un sistema. La generación de modelos que usan algoritmos inductivos está sujeta a los modelos predictivos. Tales modelos proporcionan una alta precisión debido a la inducción de un conjunto grande de reglas altamente complejas que generalmente son ininteligibles (Widmer 2000).

A continuación se describirán las técnicas de aprendizaje automático con las que se experimentó en el presente trabajo:

Machine learning

Es el estudio científico de algoritmos y modelos estadísticos que los sistemas informáticos utilizan para realizar de manera efectiva

una tarea específica sin utilizar instrucciones explícitas. Se basan en patrones e inferencias (Mitchell 1997).

El aprendizaje automático cuenta con diferentes tipos de algoritmos que pueden ser de tipo supervisado, no supervisado, semi-supervisado, aprendizaje por refuerzo, transducción y aprendizaje por multitarea.

Estos algoritmos tienen varios métodos de clasificación que ayudan a que sea más eficiente el aprendizaje de las máquinas: árboles de decisiones, reglas de asociación, algoritmos genéticos, redes neuronales artificiales, máquinas de vectores de soportes, algoritmos de agrupamiento, redes bayesianas.

En trabajos de audio se puede enunciar a Gómez y Herrera 2004, en el cual hace una estimación de grabaciones polifónicas con una comparación entre el aprendizaje automático contra la cognición musical.

Redes neuronales convolucionales (CNN)

Una red neuronal convolucional es un tipo de red neuronal artificial donde las neuronas corresponden a campos receptivos de una manera muy similar a las neuronas en la corteza visual primaria (V1) de un cerebro biológico, (Fukushima y Miyake 1982).

Este tipo de red es una variación de un perceptrón multicapa[4], sin embargo, debido a que su aplicación es realizada en matrices bidimensionales, son muy efectivas para tareas de visión artificial, como en la clasificación y segmentación de imágenes, entre otras aplicaciones (Isasi Viñuela y Galván León 2004).

Según Ciresan y col. 2011 estas redes pasan por 3 procesos:

- Capas convolucionales: el operador convolución filtra la características más dominantes de cada información que entra.
- Capas de reducción de muestreo: estas neuronas permiten que haya cierta tolerancia a cambios pequeños, permitiendo que la información sea procesada de manera correcta.
- Capas de clasificación: como su nombre lo indica, en esta última fase se hace la clasificación de las características creando etiquetas según sea el objetivo de entrenamiento del sistema.

Piczak 2015 hace un trabajo completo de clasificación del sonido del ambiente mediante redes neuronales.

4 El perceptrón multicapa es una red neuronal artificial (RNA) formada por múltiples capas, de tal manera que tiene capacidad para resolver problemas que no son linealmente separables, lo cual es la principal limitación del perceptrón (también llamado perceptrón simple). El perceptrón multicapa puede estar totalmente o localmente conectado, cada neurona de la capa i es entrada de una serie de neuronas de la capa $i+1$, Isasi Viñuela y Galván León 2004.

K-means

Es un método de agrupamiento que tiene como objetivo la partición de un conjunto de n observaciones en k grupos en el que cada observación pertenece al grupo cuyo valor medio es más cercano (Figura 13), (Kanungo y col. 2002).

Este algoritmo se conforma en 3 pasos.

- Inicialización: se escogen los grupos k y se escogen k centroides.
- Asignación de objetos a los centroides: cada objeto se asigna al centroide más cercano.
- Actualización de centroides: actualiza la posición de cada centroide de cada grupo, siendo el nuevo centroide el promedio de la posición de los objetos de cada grupo.

Algunos ejemplos con *K-means* aplicados en investigaciones musicales son: *Musical instrument extraction through timbre classification* (Park 2013), *Machine recognition of timbre using steady-state tone of acoustic musical instruments* (Fujinaga 1998).

Support vector machines (SVM)

Es una técnica usada para la clasificación y regresión de datos. Los vectores de entrenamiento se mapean en un espacio dimensio-

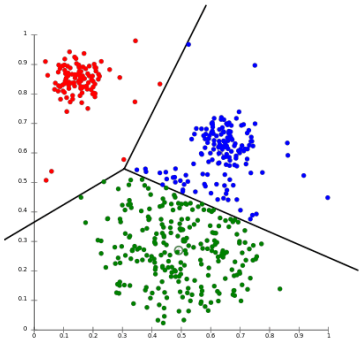


Figura 13.: *K-means*. (Kanungo y col. 2002)

nal superior (incluso infinito) mediante una función del núcleo. Un SVM encuentra un hiperplano lineal o un conjunto de hiperplanos que separan las diferentes clases de datos de entrenamiento en el espacio dimensional superior, logrando una buena separación con el hiperplano que tiene la mayor distancia a los puntos de datos de entrenamiento más cercanos de cualquier clase (el llamado margen funcional), ya que cuanto mayor sea el margen, menor será el error de generalización del clasificador (Figura 14), (Pérez 2009).

A continuación se mencionarán dos ejemplos de investigación musical que incluyen en su metodología *SVM*: *Vivi, the virtual violinist* (Percival, Bailey y Tzanetakis 2011), *Musical instrument extraction through timbre classification* (Park 2013).

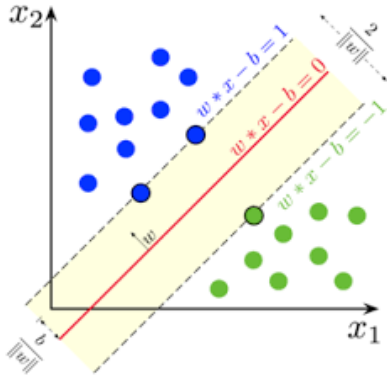


Figura 14.: *Support Vetor Machine.* (Pérez 2009)

Transfer learning

En este método se entrena un modelo para alguna tarea y dominio específico, por lo tanto, se proporcionan datos para dicha tarea aprovechando los conocimientos de modelos entrenados previamente para poder preparar modelos nuevos o con menos datos. Torrey y Shavlik 2010, explica que existen diferentes estrategias y técnicas que se pueden aplicar según el dominio, la tarea en cuestión y la disponibilidad de los datos, estás son: *Inductive transfer learning, Unsupervised transfer learning, Transductive transfer learning, Instance transfer, Feature-representation transfer, Parameter transfer, relational-knowledge transfer*. A su vez existen 3 tipos de *Deep transfer learning*: *Domain adaptation, Domain confusion, Multitask learning, One-shot learning, Zero-shot learning*.

Transfer Learning se utiliza cuando se tiene pocos elementos para entrenar el sistema, es por eso que se escogió este método para entrenar el sistema interactivo mencionado en este trabajo.

A continuación se mencionarán algunas aplicaciones de este método: *transfer learning for NLP* es usando en *word2vec y fastText* los cuáles se pueden usar para el análisis de sentimiento y clasificación de documentos. *Transfer learning for audio/speech*, su base son datos de audio, se utiliza con *automatic speech recognition (ASR)* para reconocimiento de voz en diferentes idiomas. *Transfer learning for computer vision* se usa en diferentes tareas de visión artificial como la identificación de objetos y el reconocimiento.

Austin 2009 formuló una teoría que se centró en la cognición humana, en lugar de la capacidad y las características de la tecnología. Al medir el efecto de las diferencias cognitivas individuales y las manipulaciones del diseño de la pantalla en el rendimiento, la investigación evalúa el impacto de las combinaciones multimedia en la productividad de las pruebas de transferencia de estudiantes universitarios.

2.1.3 *Plataformas de implementación*

La creciente disponibilidad de contenido de audio, a través de una amplia distribución de canales, ha generado la necesidad de

sistemas que puedan analizar automáticamente este contenido. Dependiendo de los tipos individuales de canales de distribución, los tipos de clases de audio (voz, música, etc.), la existencia de otros medios y los requisitos específicos del empleo, ha surgido una amplia gama de aplicaciones diferentes durante los últimos años: recuperación de información musical, detección de eventos de audio, análisis del habla, reconocimiento de la emoción en la voz, análisis multimodal, etc.

A continuación presentamos dos de las plataformas que se utilizan en este trabajo, por un lado, una aplicación de escritorio desarrollada en Python y por otro, un sistema web.

Python

Es un lenguaje de programación orientado a objetos, funcional, multiplatarforma y de código abierto, cualidades que nos sirven para lograr uno de los objetivos deseados, que es diseñar una aplicación pensada para músicos exclusivamente. Su biblioteca estándar nos da pie a usar diferentes instrucciones para lograr lo que se desea (Zelle 2004).

Existen varias bibliotecas para procesamiento de audio en Python, entre ellas *librosa* (McFee y col. 2015), que tiene el beneficio de proporcionar una amplia gama de funcionalidades de análisis de audio a través de un fácil uso y programación integral del di-

seño. *Librosa* se puede usar para extraer características de audio, entrenar y aplicar clasificadores de audio, segmentar una transmisión de audio utilizando metodologías supervisadas o no supervisadas y visualizar relaciones de contenido. En comparación con Matlab u otros programas similares, *Python* es software libre, otra gran ventaja es que existe un número de bibliotecas que proporcionan funcionalidades relacionadas con la programación científica (Giannakopoulos 2015).

2.1.4 *Archivo de JavaScript object notation (JSON)*

Es un formato estándar abierto basado en texto que se utiliza para serializar y transmitir datos estructurados entre un servidor y una aplicación web. Se trata de un subconjunto de la notación de objetos de JavaScript. Una de las ventajas de *Json* sobre xml como formato de intercambio de datos es que resulta más sencillo escribir un analizador sintáctico (parser) para él .*Json* se emplea en entornos donde el tamaño del flujo de datos entre cliente y servidor es de vital importancia cuando la fuente de datos es explícitamente de fiar y donde no es importante el hecho de no disponer de procesamiento xslt para manipular los datos en el cliente,(Mora-Castillo 2016).

Plataforma web

La plataforma web es programada en JavaScript. Se creó así con el propósito de construir un repositorio de recursos en línea y proporcionar a sus usuarios una manera sencilla para publicar, compartir y acceder a sus propios recursos y de otros usuarios. El repositorio se genera e incrementa mediante la aportación de recursos elaborados o referenciados por los usuarios. La plataforma cuenta con un acceso basado en roles.

Cada rol determina las funciones habilitadas para un usuario y por lo tanto las acciones que éste puede realizar. Los usuarios son identificados como registrados. Cada operación ejecutada por un usuario registrado, es identificada con su nombre y perfil.

La plataforma Web cuenta con un conjunto de funcionalidades que proporcionan la interacción con los usuarios que acceden a ella. Un diseño homogéneo y una interfaz sencilla, amigable e intuitiva, permiten proporcionar un conjunto de funciones básicas e indispensables (Madueño 2003).

2.2 LAS APPS EN LA EDUCACIÓN MUSICAL

De acuerdo con (Salas y Sandoval 2007), se puede decir que la educación en línea es la pionera para implementar softwares y

herramientas interactivas en la educación. Con el desarrollo del correo electrónico, boletines electrónicos y los grupos de discusión se empieza a abrir camino en este tipo de educación, pero fue con el desarrollo del internet y de las tecnologías de la información cuando obtuvo su auge. En la actualidad se siguen teniendo diferentes opiniones acerca de la educación a distancia ya sea de aceptación o rechazo, optimismo o pesimismo, esto porque se rompen los esquemas tradicionales del proceso de enseñanza-aprendizaje.

Hablando en específico de la educación musical, Brandao, Wiggins y Pain 1999 nos dicen que existen numerosos intentos de usar computadoras en la educación musical, dividiendo las aplicaciones computacionales en la música en varias categorías: enseñanza de los fundamentos de la música; enseñanza de habilidades de interpretación musical; análisis de música; enseñanza de habilidades de composición musical.

Existen varias escuelas de música que ofrecen clases en línea como: *Berklee College of Music*[5] la cual ofrece cursos sobre producción musical, guitarra, escritura de canciones, bajo, teclado, orquestación, teoría de la música, historia de la música. *Julliard School of Music*[6] tiene un curso en línea llamado *Julliard eLearning* enfocado a alumnos de primaria, secundaria, adultos y profesores con acceso a internet. Este fue el primer curso en línea para la

5 https://welcome.online.berklee.edu/learn-music-online.html
6 https://www.juilliard.edu/

educación musical en dicha escuela. El Conservatorio Virtual pertenece al Instituto para la Formación, Investigación y Desarrollo de la Música y otras Artes[7], su objetivo es es superar las carencias de la enseñanza musical tradicional en todos los niveles.

En México existe la educación en línea en el ámbito musical en escuelas que optan por tener esta práctica ya sea en posgrados para la investigación o en la enseñanza tradicional, como la Facultad de Música o la Universidad Tito Puente.

2.3 PROGRAMAS EXISTENTES PARA EL VIOLÍN

En esta sección se describirán los trabajos académicos, apps y softwares enfocados a la música o al violín que sirvieron de referencia para este trabajo:

- *Cortosia (KORG).*

  Es una aplicación realizada por la Universidad Pompeu Fabra en Barcelona, España, bajo la dirección del profesor Xavier Sierra, en conjunto con KORG, usando la tecnología *"Artistry"(Automatic Rating Tecnology for Musical Instrument Players).*

7 http://www.ifidma.com/

El objetivo de esta aplicación es desarrollar un buen sonido, para lograrlo, tiene un sonido preprogramado con los parámetros aceptados. Los instrumentos para los que fue realizada la aplicación son: flauta, clarinete y trompeta; actualmente se le adicionaron violín, violoncello, saxofón, trombón y tuba; esto dependiendo de la versión de la aplicación con la que se cuenta (H.-M. Lin y C.-M. Lin 2015).

Para tener un buen sonido analiza cinco parámetros: estabilidad de la nota (frecuencia), estabilidad de la dinámica, estabilidad en el timbre, riqueza del timbre y ataque. Teniendo completos los cinco rangos se tiene un 100 por ciento de calidad en el sonido, ya que la pantalla está dividida en los cinco parámetros para dar el puntaje (Figura 15). El sonido se puede analizar en tiempo real o en grabación, dichas grabaciones se pueden guardar para contar con avances y es recomendable para alumnos intermedios o básicos. Aunado a esto la aplicación cuenta con un metrónomo. Las especificaciones con las que cuenta la aplicación son las siguientes:

- Cuenta con las 12 notas del mismo temperamento.
- Rango de altura: A1(55[Hz])- G#7(3322[Hz]).
- Pitch de referencia: A4 (430-450 [Hz]).
- Rango de transposición: C, F, B bemol, E bemol.

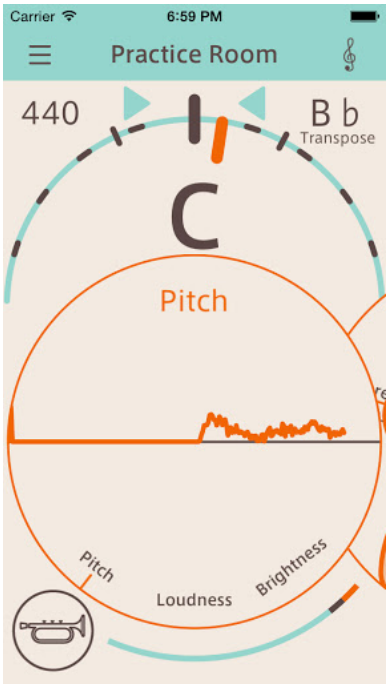


Figura 15.: *Cortosia*.(App Cortosia)

- Precisión: +/-0.1 cent.

A partir de esta aplicación se empezó a tomar el camino del timbre para la investigación. Esta aplicación fue creada para instrumentos de aliento, las últimas versiones ya funcionan para cello y violín pero sólo con notas largas. Uno de los aportes del sistema desarrollado en este trabajo a diferencia de Cortosia, es que al ser para violín, se estudia el timbre no

solamente al paso del arco en nota sostenida sino también se estudiaron golpes de arco *spiccato, détaché, martelé*.

- *Music Wrech*.

  Es una aplicación diseñada por Alden Doyle, ingeniero y violinista, la cual transcribe lo que toca el violín en tiempo real, incluido el *vibrato* y dobles cuerdas. Puede ser usada solamente en sistemas operativos iOS. La pantalla es interactiva y contiene colores, para que se rellene la nota cuando haya un error en la interpretación. También cuenta con un metrónomo que se puede usar mientras se está tocando. La sesión puede ser grabada para revisarla después (Figura 16).

  Tiene un menú con varias opciones:

  - Afinación: es para cuerdas sueltas. Marca la cuerda con color rojo o azul y debajo de esta muestra una gráfica del comportamiento del sonido. Puede ser ajustado para quintas perfectas o temperamento igual.
  - Práctica: El modo de práctica ayuda a la afinación de las notas desde la cuerda G (sol) hasta las dos octavas arriba de la cuerda E (mi). Se muestran las cuatro cuerdas y a partir de ellas se ilumina la ubicación de las notas de forma ascendente, teniendo solamente el metrónomo puesto.

- Práctica Avanzada: muestra dos ventanas, una con las cuerdas del violín y en la otra se muestra un pentagrama, en ambas se va pintando el lugar donde va la nota, permite tocar dobles cuerdas y muestra la identificación de un intervalo justo. Se puede configurar para que tenga temperamento igual, pitagórico o justo.
- Canciones y Escalas: pone en la pantalla una pieza para ser tocada con *score following*. Contiene escalas de 1 o 3 octavas, y canciones que se van actualizando. Marca una nota en rojo cuando está baja de afinación y azul cuando se encuentra alta de afinación, la nota verde es la que se está tocando actualmente.

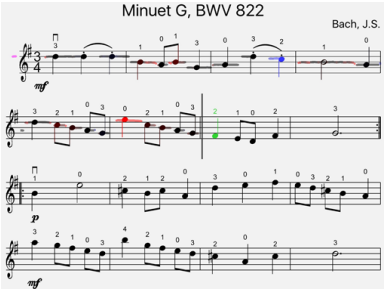


Figura 16.: *Music Wrench. (App Music Wrench)*

Esta app contiene *score following* y *pitch detection* dando una retroalimetación en vivo. La diferencia existente con el trabajo presentado en esta tesis es que *Music Wrench* no ana-

liza el timbre, ni golpes de arco. También es importante la compatibilidad del software, ya que ésta sólo funciona en iOS, por eso el sistema interactivo que se desarrolló en este trabajo fue pensado para tener compatibilidad en todos los dispositivos móviles, al ser software libre, las personas que tengan interés en ella podrán enriquecerla de la manera que crean adecuada.

- *Music Jacket*.

  Es un proyecto desarrollado por Rose Jhonson, Jante van del Linden e Yvonnne Rogers en la Universidad de Milton Keynes, UK (Van Der Linden y col. 2011). Este proyecto se desarrolló como una ayuda para enseñar el violín. Se enfoca en el estudio de los brazos, la inclinación del arco sobre las cuerdas mediante el uso de retroalimentación vibrotáctil en tiempo real para guiar las manos. La inclinación es una habilidad compleja y se ha demostrado que lleva más de 700 horas de práctica para lograr un dominio básico de las habilidades motrices involucradas. El objetivo es ver que los principiantes aprendan más rápido y se mantengan motivados.

  MusicJacket se compone de dos partes: captura de movimiento y retroalimentación. La posición de las extremidades

de la parte superior del cuerpo se mide utilizando *Animazoo* una chaqueta de captura de movimiento portátil que se comunica de forma inalámbrica con la computadora (Figura 17).

La retroalimentación vibrotáctil es entregada por pequeños vibradores similares a los encontrados en los teléfonos móviles. Estos están conectados a un Arduino que se conecta con la computadora a través de USB. La desviación de la posición de cada mano con respecto a una posición ideal es calculada. La posición ideal para la mano izquierda se obtiene sosteniendo el violín bajo la instrucción de un maestro.

Para introducir la posición de la mano derecha, un asistente sostiene el arco en la cuerda y el alumno pasa su mano por el arco. La línea que se forma se dibuja en el marco de referencia del violín, de modo que si el violín se mueve, la trayectoria ideal para el arco se moverá con ella (Van Der Linden y col. 2011).

Music Jacket es totalmente diferente al sistema interactivo desarrollado en esta investigación ya que lo propuesto en este trabajo analiza el timbre que surge de la mano derecha analizando tres golpes de arco sin necesidad de sensores.

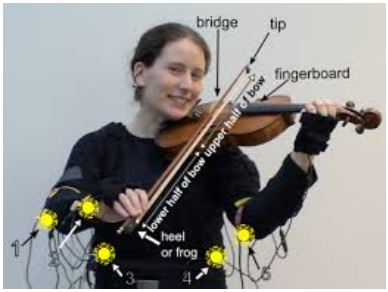


Figura 17.: *MusicJacket*. (Van Der Linden y col. 2011)

- *An Interface for Real-time Classification of Articulations Produced by Violin Bowing*

  El proyecto CyberViolin de Peiper, Warden y Garnett 2003 busca captar aspectos de la interpretación del violín en músicos profesionales. El software para la clasificación de los golpes de arco de violín utiliza un sistema de seguimiento de movimiento electromagnético con dos sensores. Un sensor está conectado a la parte posterior de un violín acústico y el otro está conectado al talón del arco. Este sistema pretende detectar *détaché, martelé, staccato, spiccato y legato*. La diferencia entre estos distintos golpes de arco es el resultado de variaciones sutiles en las articulaciones, que son discretos y medibles.

  CyberViolin consta de los siguientes componentes: caracterización del golpe de arco, inducción del árbol de decisión y

clasificación del golpe de arco. La inducción del árbol de decisión permite que las reglas de clasificación se generen fácilmente y se comparen con las expectativas de los expertos. La identificación del golpe de arco implica una representación de cada golpe en términos de un conjunto de datos que traduce la actividad física en varias clases para facilitar el reconocimiento de la computadora. El siguiente paso es la creación de una estructura de árbol que permita a la computadora reconocer los diversos flujos de datos que constituyen el desempeño exitoso de cada golpe. El paso final es la selección de los datos a través del árbol, un proceso que resulta en la identificación en tiempo real por parte de la computadora de las articulaciones. El sistema se puede utilizar en dos modos: entrenamiento y clasificación.

La diferencia de Cyberviolin con el desarrollado en esta investigación es que está programado en C++ y necesita sensores para poder ser medido.

- *Effort-based analysis of bowing movements: evidence of anticipation effects*.

  Esta investigación hecha por N. Rasamimanana y Bevilacqua 2008 se basa en dos golpes de arco *détaché y martelé*, para la medición de la velocidad y el esfuerzo de los vio-

linistas al tocar en un sonido continuo. Utilizaron escalas y algunos estudios mixtos que contienen estos dos tipos de arco.

El sistema de medición se compone de un sistema de captura de movimiento óptico *Vicon System* 460 para medir el movimiento del arco. Se colocaron seis cámaras M2 alrededor del instrumentista, proporcionando una resolución espacial inferior a 1 mm a una velocidad de fotogramas de 500Hz, en un volumen de aproximadamente 1 metro cúbico.

Se colocaron seis sensores en el instrumento, cuatro en la tapa superior del violín, uno en la cabeza y uno en el cordal para indicar la posición de las cuerdas. Tres sensores fueron colocados en el arco. Un acelerómetro ADXL202 de 3 ejes adicional se fijó al talón del arco. Los datos del acelerómetro se digitalizaron a 500Hz y se transmitieron a una computadora portátil para su grabación. Para garantizar la sincronización posterior a la grabación entre los datos de captura de movimiento y los datos del acelerómetro, la pista de sonido fue grabada simultáneamente por cada sistema de detección.

El acoplamiento de los datos de captura de movimiento y los datos del acelerómetro garantiza una medición precisa del espacio y el tiempo de la posición, velocidad y acelera-

ción de los violinistas (Figura 18) (N. Rasamimanana y Bevilacqua 2008). Los marcadores y sensores colocados en el arco agregaron menos de dos gramos, principalmente en el talón (el arco tenía 62 g y el talón tenía 17 g).

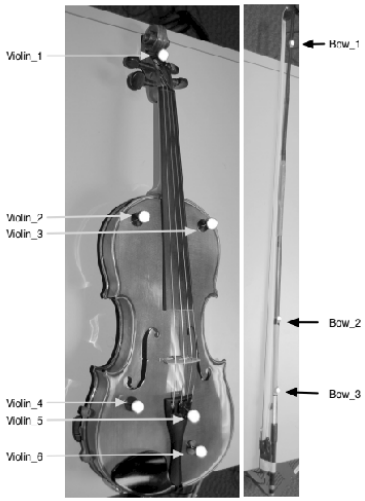


Figura 18.: *Effort-based analysis of bowing movements: evidence of anticipation effects*. (N. Rasamimanana y Bevilacqua 2008)

Este es todo un sistema complejo, lleno de sensores y un circuito el cual se ocupa para medir la fuerza, distancia, y posición. Sólo se analizan dos golpes de arco obteniendo como resultado final gráficas para la velocidad. Todo esto es una gran diferencia con el sistema propuesto en este trabajo

de investigación, ya que este detecta tres tipos de golpes de arco para poder dar una retroalimentación de dichos golpes en la ejecución y el producto final no son gráficas de velocidad porque nos interesa la precisión del golpe de arco mas no la velocidad de él.

- *Classification of common violin bowing techniques using gesture data from a playable measurement system.*

  Este artículo realizado por Young 2008, presenta los resultados de un estudio acerca de las técnicas del golpe de arco del violín más comunes, utilizando un sistema de medición basado en los diseños anteriores de *Hyperbow* (Young 2003).

  Este sistema incluye los sensores de posición (fuerza de arco hacia abajo y arriba), inercial (aceleración 3D y velocidad angular 3D) y sensores de posición instalados en un arco de violín y un violín eléctrico de fibra de carbono, que permite el registro del golpe de arco del violinista en condiciones normales.

  Se grabaron ocho violinistas con seis diferentes técnicas de arco (*accented detaché, detaché lancé, louré, martelé, staccato, spiccato*) de un extracto musical de una obra clásica con tiempo, arcadas y notas específicas, con estos datos se llevó a cabo la tarea de clasificarlos mediante la técnica del

arco utilizando la descomposición del valor singular (SVD) para calcular los componentes principales y utilizar un clasificador *k-nearest-neighbor* (k-nn) para los componentes principales como entradas. En este trabajo, las características se extraen de los datos de posición producidos por un sistema de seguimiento de movimiento electromagnético. Un árbol de decisiones toma estas características como entradas para clasificar hasta seis técnicas de golpes de arco diferentes en tiempo real.

La configuración de los componentes de este sistema son (Figura 19): el sistema de medición de golpes de arco del violín instalado en un arco *CodaBowR ConservatoryTM* y en un violín eléctrico Yamaha SV-200; auriculares (a través de los cuales los participantes escucharon todos los pasajes pregrabados); interfaz de audio M-Audio Fast Track USB; y una computadora Apple MacBook con una Intel de 2 GHz, procesador Core Duo (OS X) que ejecuta PureData (Pd) versión 0.40.0-test08.

El audio se grabó en un archivo por medio de un *patch* de Pd, para codificar los datos como audio multicanal y sincronizar todos los datos juntos. Cada archivo se registró con un número de prueba, un número de repetición y un sello de fecha y hora (Young 2003).

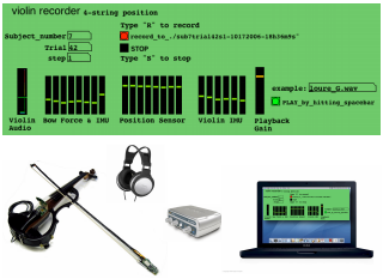


Figura 19.: Sistema de medición. Young 2008
Young 2008

Este es de los primeros trabajos de Young en el que se trata la medición del golpe de arco del violín. Es importante en este trabajo de investigación ya que utiliza el clasificador K-nn con el cual también se experimentó en el sistema interactivo aquí propuesto. La diferencia de Young es que utiliza sensores en el arco.

- *Gesture analysis of violin bow strokes*.

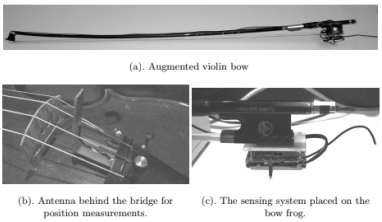
  N. H. Rasamimanana, Fléty y Bevilacqua 2005 desarrollaron un violín aumentado, es decir, un instrumento acústico con capacidades adicionales de captura de gestos para controlar los procesos electrónicos. Analizan tres golpes de arco diferentes (*detaché, spiccato, martelé*), utilizando este violín aumentado. Se consideraron diferentes características basadas en la velocidad y la aceleración. Realizaron un aná-

lisis discriminante lineal para estimar un número mínimo de características pertinentes necesarias para modelar esos golpes de arco. Dentro de sus resultados se tiene que las aceleraciones máximas y mínimas de un golpe de arco fueron eficientes para parametrizar los diferentes tipos de golpes de arco, así como las diferencias en la dinámica.

Las tasas de reconocimiento se estimaron utilizando un método knn (*k-nearest-neighbor*) con varios conjuntos de entrenamiento. El prototipo que crearon mide dos tipos de datos utilizando una tecnología similar a la descrita por Young 2002: posición del arco y aceleraciones del arco. El primero es un sistema de detección para medir la posición de contacto de las cuerdas y del arco en dos direcciones: entre la punta y el talón, entre el puente y el diapasón.

El segundo es la aceleración, la cual se mide con dos dispositivos analógicos ADXL202 colocados en el talón arco. Los dos acelerómetros se fijan al talón del arco de tal manera que la aceleración se mide en tres dimensiones: dirección de arqueamiento, dirección de la cuerda y dirección vertical (Figura 20).

Estos dos últimos sistemas mencionados tienen el principio básico del sistema desarrollado en esta investigación que es la detección de golpes de arco, pero ambos enfocados en la



(a). Augmented violin bow

(b). Antenna behind the bridge for position measurements.

(c). The sensing system placed on the bow frog.

Figura 20.: Prototipo del violín aumentado. N. H. Rasamimanana, Fléty y Bevilacqua 2005

pura medición de la velocidad de ellos sin darle importancia a la ejecución del golpe de arco.

La diferencia con el trabajo realizado en esta investigación es que en éste se propone tener una retroalimentación para los violinistas, a partir de la medición de cada golpe de arco para que ellos sepan reconocer la arcada que están ejecutando.

Esto no deja de lado que Young 2008 con sus múltiples investigaciones que ha hecho sobre el arco del violín sea una vertiente en el trabajo expuesto en esta tesis, con lo cual se pretende tomar la base dichas investigaciones para que los datos generados en este trabajo se puedan traducir para los músicos para tener una mejor comprensión y conciencia de su mano derecha en la ejecución del instrumento.

Estos sistemas mencionados anteriormente cuentan con unos circuitos complejos con sensores para lograr la medición, en este trabajo se utilizan grabaciones para entrenar un sistema que clasifique los golpes de arco. Por medio del audio de entrada se puede obtener una respuesta sobre el estado del golpe de arco sin necesidad de que el violinista tenga todo un circuito al momento de tocar y sea de fácil manejo.

Otro punto importante es que *Hyperbow* usa violín eléctrico y aunque también se basa en grabaciones, es muy distinta la información tímbrica que da un violín eléctrico a un acústico, ya que el violín acústico es rico en armónicos y el timbre también es un propiedad del sonido que nos interesa resguardar.

De los sistemas y apps antes mencionados para el violín a lo largo de esta sección se puede concluir que los sistemas de retroalimentación asistida por medio electrónico apoyan de manera favorable el aprendizaje de la música.

3

DESARROLLO DEL SISTEMA

En este capítulo se describe la metodología que se utilizó para entrenar el sistema interactivo, desde la preparación de datos hasta la forma de clasificarlos.

3.1 PREPARACIÓN DE DATOS

Lo primero que se realizó fue la obtención de datos para entrenar el sistema y asegurar que fuera funcional para los músicos. Ademas, en comparación con los proyectos existentes, el presente trabajo no requiere de la conexión con un circuito, facilitando el uso del sistema interactivo.

Se realizaron grabaciones de los tres tipos de golpes de arco *détaché, martelé, spiccato* ejecutadas por diferentes violinistas con la finalidad de entrenar un sistema que pudiera clasificarlos. De tal

forma, cuando un usuario toque con alguna técnica de arco con las que fue entrenado el sistema, este pueda darle una retroalimentación acerca de su ejecución.

3.2 DISEÑO DE LAS GRABACIONES

En principio se escribieron estudios para los tres tipos de golpes de arco que se analizan en la investigación, *Martelé* (Do Mayor), *Spiccato* (Sol Mayor) y *Detaché* (Re Mayor). Se escogieron estas tonalidades porque son utilizadas para el estudio de la posición de los dedos de la mano izquierda sobre el diapasón. Estos estudios están escritos solamente para la primera posición del violín. Cada ejercicio cuenta con un compás de silencio seguido de un compás con notas, esto con el fin de darle tiempo al programa para procesar el golpe de arco en el entrenamiento.

En la Fig. 21 se muestra la imagen del ejercicio escrito para *Spiccato*[1].

Se realizaron cuarenta grabaciones de los ejercicios: diez de cada golpe de arco y diez grabaciones extras mal ejecutadas. Todas las grabaciones fueron guardadas en formato *.wav* debido a que es un formato de simple manejo en *Python* y que contiene direc-

1 Los demás ejercicios se pueden consultar en el apéndice.

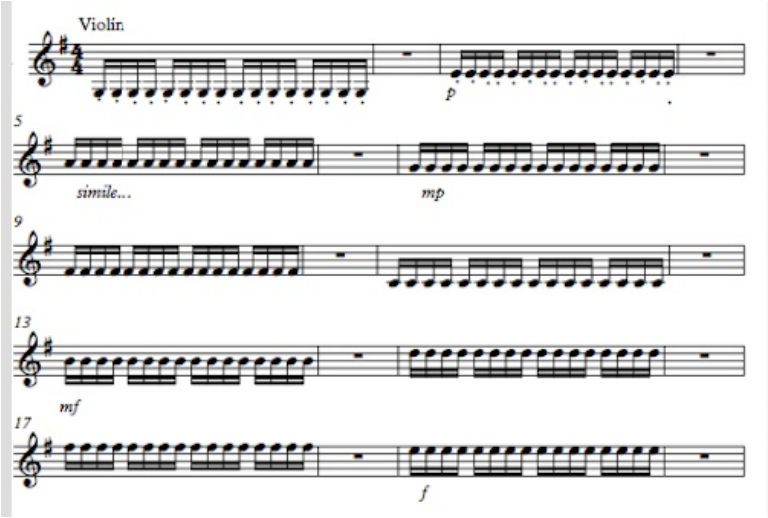


Figura 21.: Ejercicio *Spiccato*

tamente los datos de las muestras de sonido para realizar análisis numérico, científico y gráfico.

El siguiente paso fue seleccionar a los violinistas que realizarían las grabaciones. Los criterios de selección contemplaron que los músicos tuvieran diversos perfiles y diferentes campos de acción. En otras palabras, era importante considerar músicos activos en la escena de la música clásica occidental con formación académica propia de un conservatorio o escuela de educación superior, pero también, músicos activos en la escena no académica cuyos géneros de interpretación abarcan desde pop, rock, jazz, etc.

Con base en dichos criterios, los músicos seleccionados fueron:

Amado Álvarez Luz, concertino de la Orquesta Filarmónica de la Marina, es egresado de la Facultad de Música de la UNAM y a su vez es ejecutante de grupos de música pop, rock y otros estilos.

Mariano Batista Viveros, violinista de la Orquesta Filarmónica de la UNAM (OFUNAM) y violín principal de las sección de segundos de la Orquesta Filarmónica de la Marina, es egresado del Conservatorio Nacional de Música y su carrera como intérprete se ha desarrollado principalmente en el ámbito orquestal.

Beatriz Adriana Botello Castillo[2], violín primero de la Orquesta Filarmónica de la Marina, cursó sus estudios musicales en la Facultad de Música de la UNAM. Aunque su carrera como músico se desarrolla principalmente en la escena orquestal, ha incursionado en la interpretación de la música popular.

3.3 EXPERIMENTACIÓN DE LOS DESCRIPTORES

Para entrenar un sistema se necesitan realizar varios experimentos con el fin de crear patrones similares que serán guardados en el sistema y posteriormente identificados. Así, el sistema será capaz de dar la retroalimentación que el violinista necesita.

2 Considerando mi perfil como ingeniera, como músico y como autora del proyecto resultó oportuno incluirme dentro del grupo de violinistas seleccionados.

Hablando particularmente de los descriptores de audio que se enunciaron en el capítulo anterior, se experimentó con algunos algoritmos[3] con al finalidad de decidir cuál entrenaría el sistema. Como resultado de estas pruebas se obtuvieron gráficas y cifras que permiten corroborar la funcionalidad del sistema.

Todos los ejemplos mostrados a continuación serán sobre el golpe de arco *spiccato*. Este golpe de arco resulta un tanto complejo para el entrenamiento del sistema ya que cada golpe tiene una duración muy corta y una mayor velocidad en comparación con los otros dos.

Como primer paso se analizó cada audio completo, es decir, la grabación del estudio íntegro (duración de 1:17 min aproximadamente). El análisis se realizó con algunos descriptores de audio programados en *Python* con la biblioteca *Librosa*.

El primer análisis se realizó con el algoritmo llamado *Pitch Transcription*; el cual sirve para analizar el *pitch* de un archivo de audio.

Como resultado de este ejercicio se obtuvo un espectrograma, (fig 22), que sirve para comparar imágenes y entrenar el sistema con este descriptor.

En este espectrograma se puede ver toda la multiplicidad de frecuencias que componen a la frecuencia fundamental de las notas

3 Este código se podrá consultar en el anexo

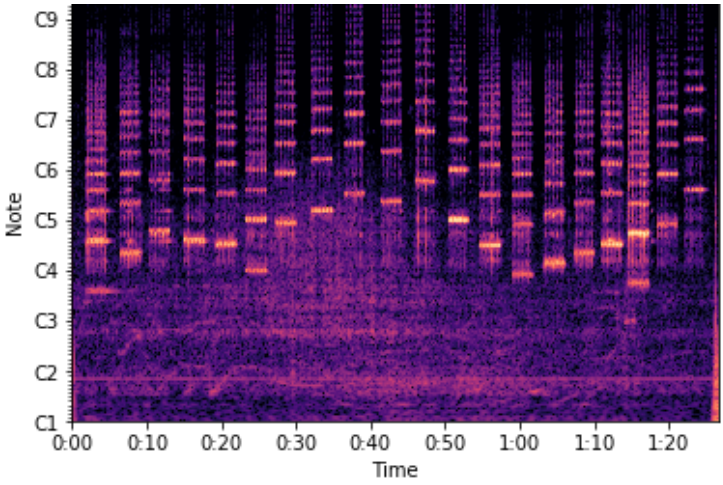


Figura 22.: Espectrograma *Spiccato*

ejecutadas. Estas múltiples frecuencias son los denominados armónicos y los espacios entre cada frecuencias son las pequeñas pausas entre cada golpe de arco.

El espectrograma también se utilizó para programar un *onset env* y obtener la gráfica del *onset detection* del audio (Figura 23).

Los *onsets* sirven para obtener más descriptores como el *onset time* que detecta cada nota tocada en un tiempo específico. En la Figura 24 está representado en cada línea roja. Se puede notar que la distancia entre las líneas es mínima, esto significa que el silencio es breve entre cada nota ya que el spiccato en este ejercicio está escrito en dieciseisavos.

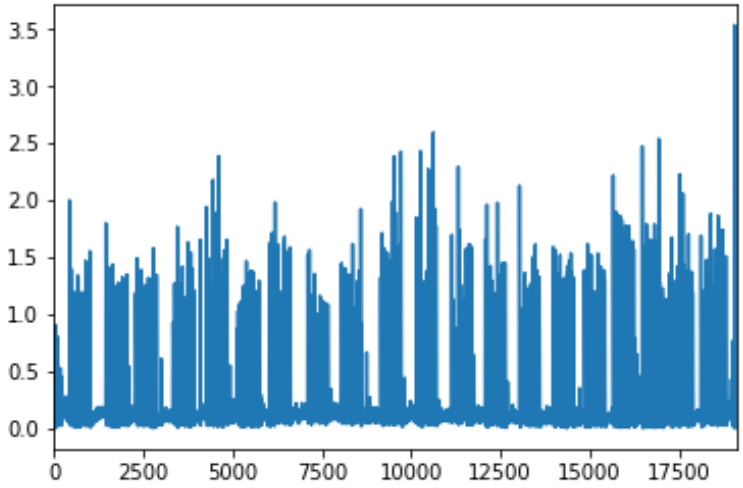


Figura 23.: *Onset del Spiccato*

Para terminar con este algoritmo de *pitch tanscription* se graficaron los *onsets* en un espectrograma (Figura 25). La diferencia del *Onset Time* con el *Onset Spectrogram*, es que el primero detecta el tiempo en que fue tocada cada nota, al contrario del segundo que marca la frecuencia que se tocó en cierto lapso.

Como segundo experimento se realizó un algoritmo denominado *Tempo Estimation*, cuyo objetivo era analizar el tiempo. Igual que con el código anterior, a lo largo de los siguientes párrafos se explicará cómo fue programado y la utilización del código en la investigación.

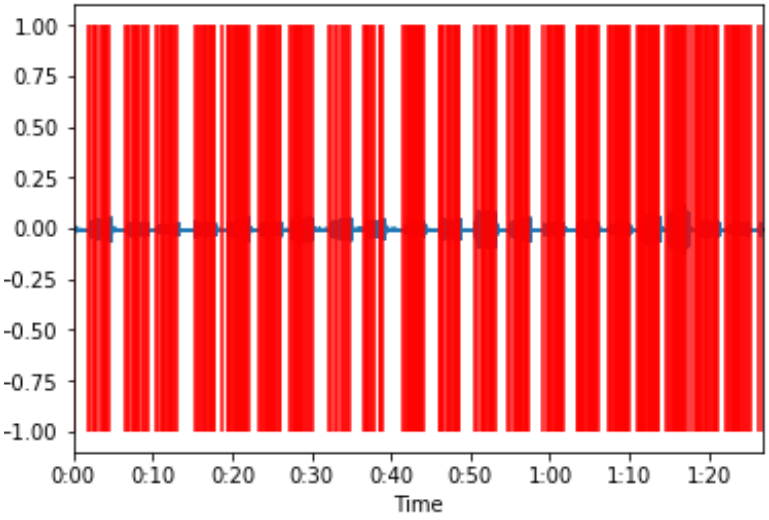


Figura 24.: *Onset time* del *Spiccato*

Se escogió *Tempo Estimation* porque el objeto de estudio es el golpe de arco, su duración y su ejecución. A diferencia del algoritmo de *Pitch Transcription*, éste contiene un estudio de los descriptores en el tiempo.

Como resultado del análisis de este algoritmo, se obtuvieron los *onsets* contenidos en él. La gráfica e información obtenida es de relevancia porque a partir de ésta derivan los descriptores que conformarán la integridad del algoritmo.

Como se mostró en el capítulo anterior, los *onsets* detectan el momento en que comienza un sonido. Con esa información se gra-

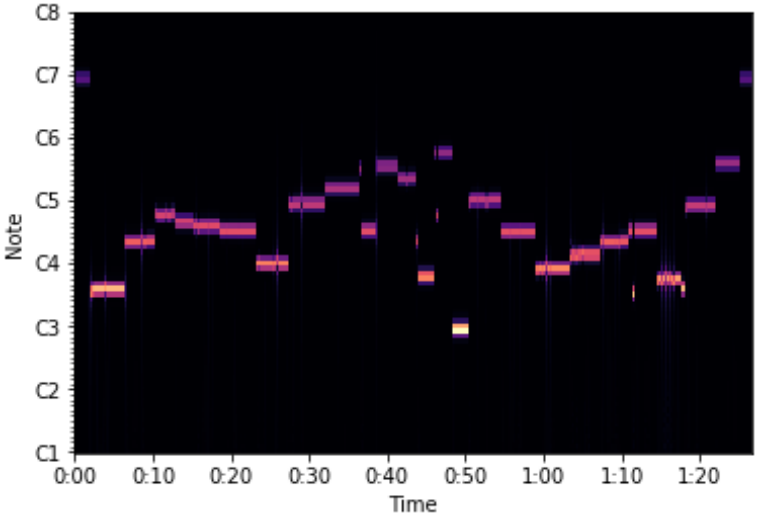


Figura 25.: *Onset Spectogram del Spiccato*

ficaron los picos de energía en cierto número de muestra (Figura 26).

Después, se obtuvo la autocorrelación (Figura 27) que permite graficar la estimación de un retardo (Figura 28) o de un BPM[4] (Figura 29).

Con los *onsets* también se graficó un tempograma; ya que su objetivo es el tiempo, y no la frecuencia, puede funcionar mejor que el espectograma para entrenar el sistema por medio de igualación de imágenes (Figura 30).

4 Beats per minute (pulsos por minuto)

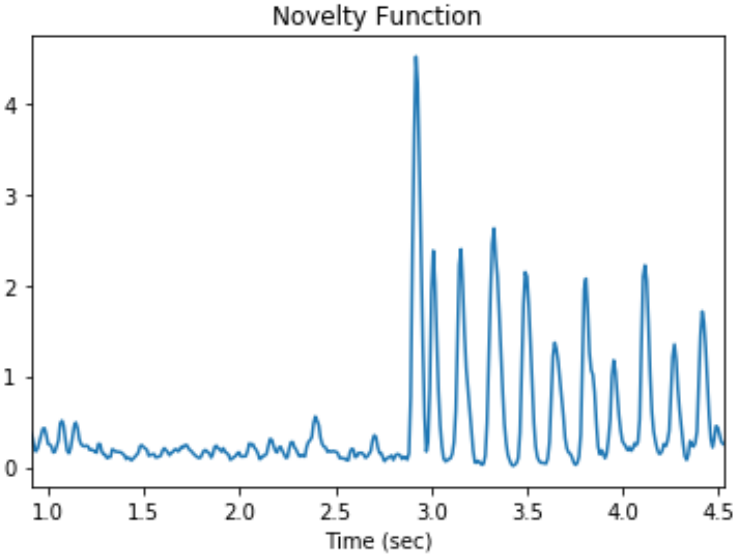


Figura 26.: Picos de energía en las muestras

Como siguiente etapa se obtuvo el tiempo en un beat para poder seccionar el audio en cada golpe de arco y detectarlo, lo cual permite que este procedimiento resulte más fino y a su vez tenga un mejor funcionamiento en comparación con el algoritmo de *Pitch Detection*. Se obtuvo información precisa, ya que cada beat signfica cada golpe de arco ejecutado.

También se programó un *zero crossing rate* para poder encontrar el periodo de cada señal obteniendo los cruces en cero, esto con el fin de detectar cada golpe de arco o en su defecto, cada compás de silencio separado de la señal de audio (Figura 31).

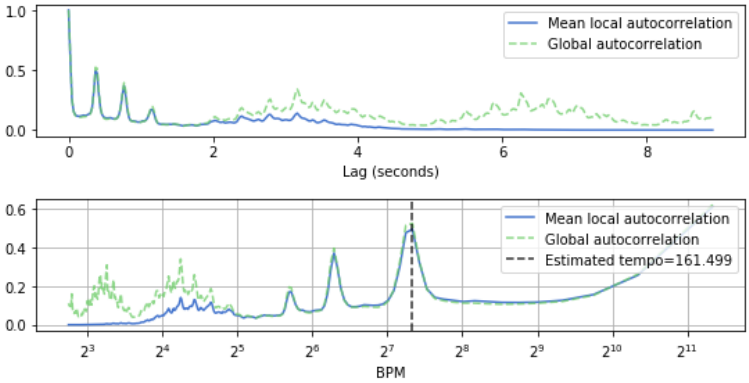


Figura 27.: Autocorrelación

Como parte del proceso también se obtuvieron las propiedades de la escala de mel (de la cual se habló en el capítulo anterior). En la Figura 32 se ejemplifica una de ellas, *Mel Power Spectogram*. Esta gráfica es un espectrograma de una grabación de Martelé. La gráfica está en el rango de frecuencia [Hz] y tiempo [s] (el tiempo que dura la grabación), en ella se observa con precisión las notas y sus armónicos que se producen en cada tiempo.

En el mismo proceso se graficaron dos *Mel Power Spectogram*, uno armónico y otro percusivo. Para el presente estudio resulta mas útil el espectograma percusivo porque divide cada golpe de arco mostrándolo en el tiempo, en cambio el espectro armónico muestra las frecuencias fundamentales tocadas con sus armónicos correspondientes (Figura 33).

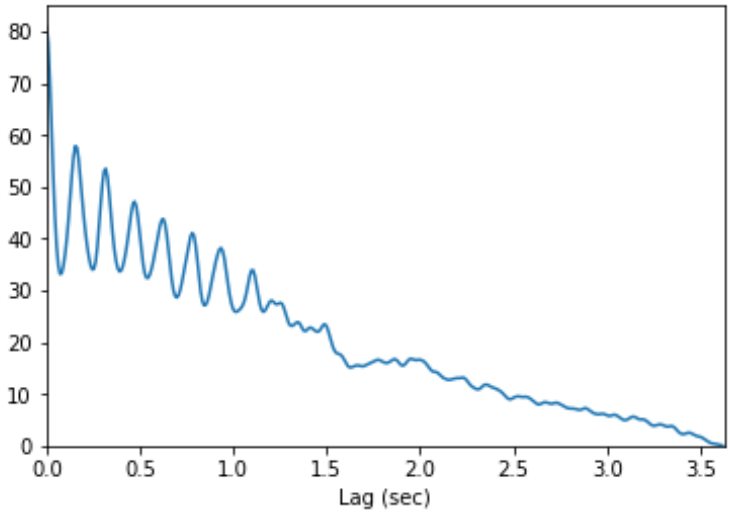


Figura 28.: Tiempo de Retardo

En el procesamiento de audio, el Cepstrum de Frecuencia de Mel (MFC) es una representación del espectro de potencia de un sonido en una escala de frecuencia de mel no lineal. Dentro de este proceso, se consideró la obtención de los MFCC[5] como parte fundamental para el entrenamiento. (Figura 34).

Para continuar, paralelamente con el entrenamiento del sistema, cada grabación realizada se seccionó por compás para crear subarchivos que se convirtieron en metadatos para entrenar el sistema.

5 Los Coeficientes Cepstrales de Frecuencia de Mel (MFCC) son coeficientes que colectivamente forman un MFC Logan y col. 2000

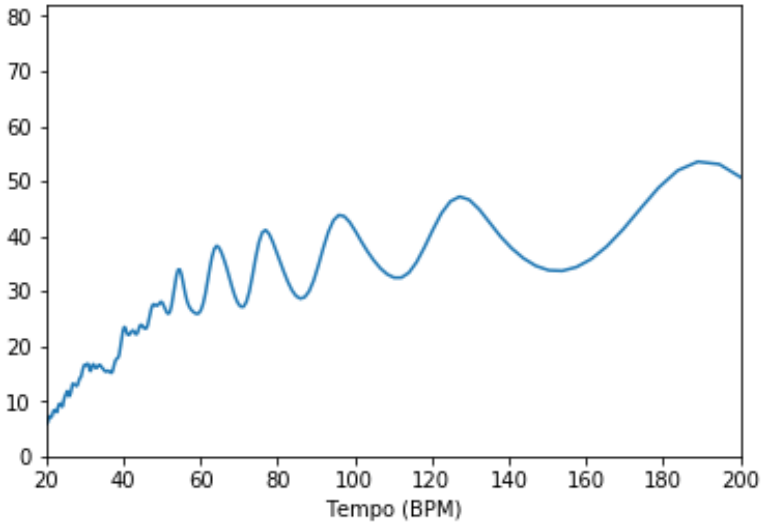


Figura 29.: Tiempo en BPM

La segmentación de archivos se realizó con Audacity[6]. Con esta segmentación se obtuvieron 979 archivos en total con una duración de 1 a 4 [s] aproximadamente. La cantidad de archivos resulta de relevancia tomando en cuenta que a mayor cantidad de archivos el sistema entrenado es más robusto.

Dentro del mismo proceso se volvieron a realizar los mismos experimentos para cada microarchivo. Con la finalidad de reafirmar que la hipótesis planteada hasta el momento era la correcta,

6 Audacity es una aplicación informática multiplataforma libre, que se puede usar para grabación y edición de audio, distribuido bajo la licencia GPLv2+.

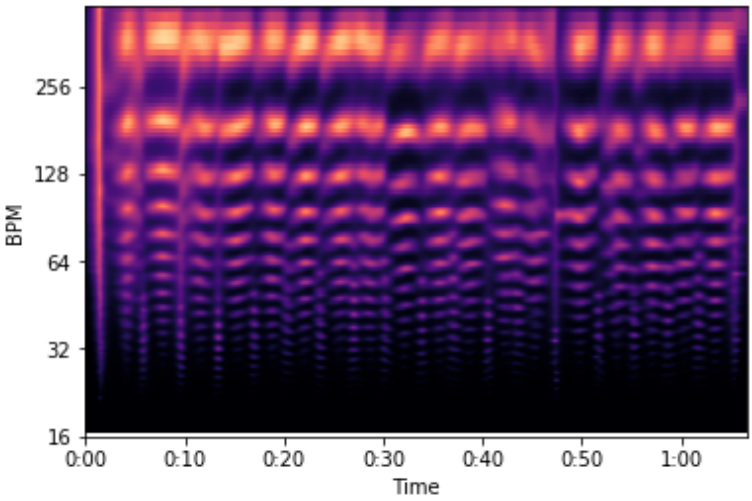


Figura 30.: Tempograma

es decir, que el algoritmo de *Tempo Estimation* es el adecuado para entrenar el sistema.

Los resultados del análisis fueron los siguientes:

Se comenzó a programar *Zero Crossing Rate*, de manera sobresaliente el resultado no fue el esperado; los cruces por cero no son tan definidos como en el archivo completo (Figura 35).

Posteriormente, se obtuvo el *Mel Power Spectrogram*. Una diferencia de este descriptor con el audio completo experimentado es que la frecuencia y sus armónicos, como demuestra la gráfica, resultan más precisos (Figura 36).

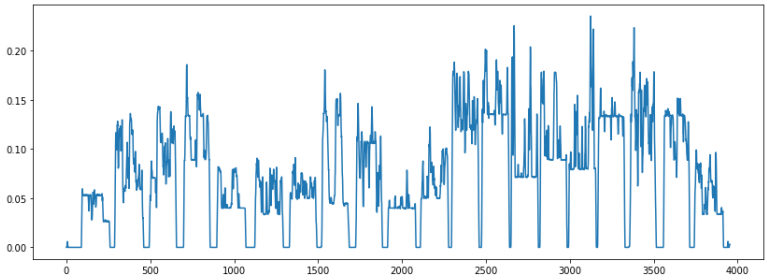


Figura 31.: *Zero Crossing Rate*

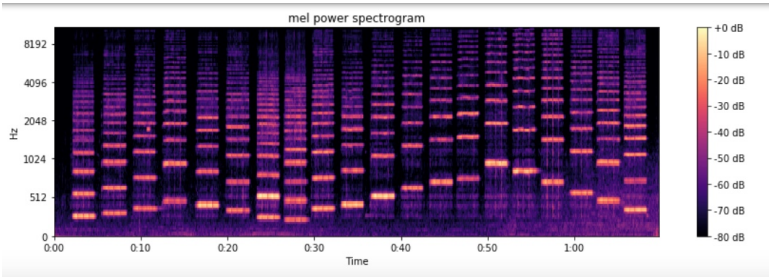


Figura 32.: *Mel Power Spectogram*

Aunado a lo anterior se graficaron los *Mel Power Spectrogram Harmonic y Percusiv*; en dichos diagramas puede notarse una mayor presencia de ruido y no se capta la señal de una manera tan nítida como se nota en el audio completo (Figura 38).

Después se graficó el MFCC con todos sus delta, la información, a pesar de la diferencia de frecuencias, es parecida a la obtenida en el audio completo como se muestra en las gráficas anteriores.

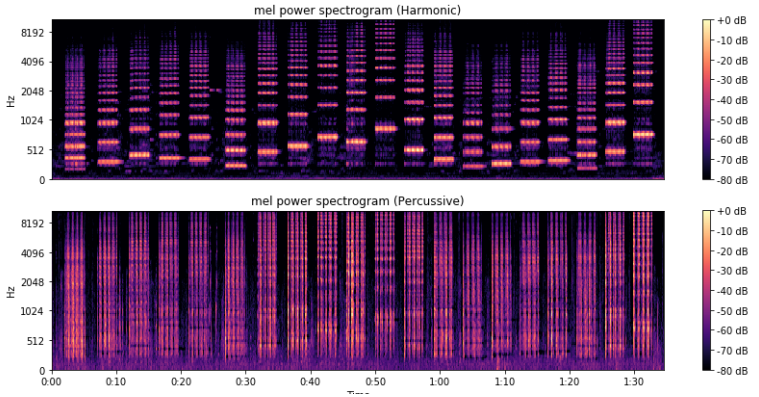


Figura 33.: *Mel Power Spectogram Harmonic and Percussive*

Por último se graficó el *Beat Tracking* donde se puede notar que la información arrojada es más precisa, ya que al mostrar la frecuencia y los armónicos también se muestra en que momento comienza cada grupo de 4 *spiccatos* y aparecen marcados con una línea blanca (Figura 39).

En función de corroborar las premisas principales, se realizó el algoritmo de *Tempo Estimation* medido en BPM.

Primero se generan los *onsets* para obtener el tempograma. El cuál es muy diferente al tempograma presentado con el audio completo, ya que en este no existe la separación entre cada silencio sino que el sonido se grafica de manera continua (Fig 40).

Con los *onsets* se graficó el *beat time*, el resultado de ésta es más preciso en un audio completo. Al hacerlo con un audio de menores

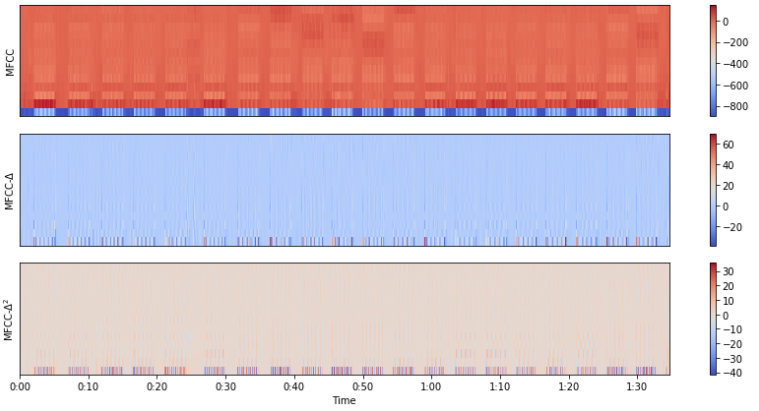


Figura 34.: *MFCC*

dimensiones se puede notar que los resultados no son del todo útiles para esta investigación porque no detecta cada dieciseisavo sino cada cuarto. Para lograr que se detecte cada nota, es decir, cada dieciseisavo, se debe manipular la velocidad manualmente. Por lo tanto, esto no resultó favorable para entrenar el sistema ya que no lo hace de manera automática (Figura 41).

Continuando con la comprobación de la hipótesis, se programó *pitch transcription* de *librosa*. En comparación con los resultados de los audios completos, con los audios seccionados se obtuvo información más precisa y estos descriptores resultan más pertinentes para entrenar el sistema.

En referencia del pitch, se extrajeron los *onsets*, uno para cada golpe de arco y su cruce en cero (Figura 42).

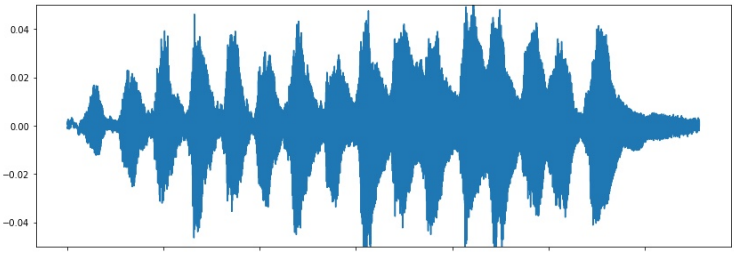


Figura 35.: *Zero Crossing Rate*

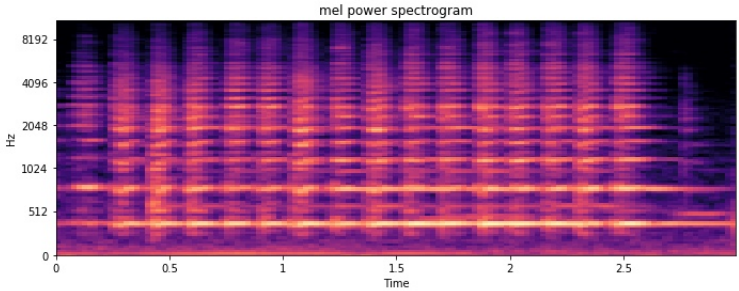


Figura 36.: *Mel Power Spectrogram*

Como consecuencia, se obtuvieron los *onset time*, los cuales se generan de forma automática. De manera contraria, en el algoritmo de *tempo estimation* es necesario manipular la velocidad del *onset time* dentro del código. En la figura (43) se muestra como se marca claramente los 16 golpes de *spiccato* que se encuentran en ese audio.

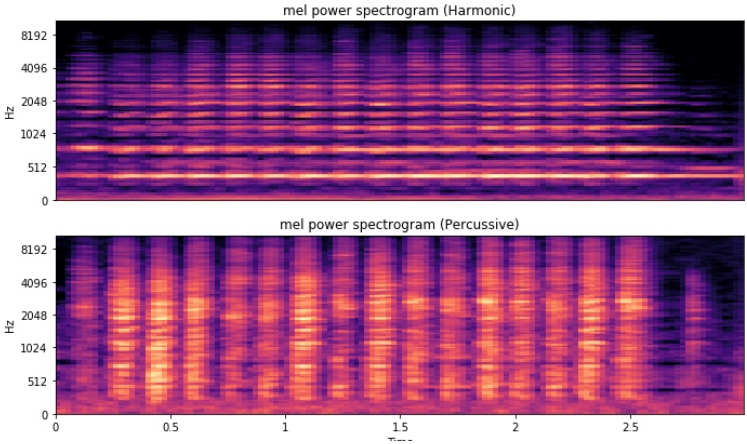


Figura 37.: *Mel Power Spectrogram Harmonic y Percusiv*

Con los experimentos realizados con archivos completos y microarchivos se pudo concluir que los resultados arrojados entre uno y otro resultan muy diferentes. Los datos obtenidos con los microarchivos fueron más precisos.

3.4 CLASIFICACIÓN DE DATOS

Continuando con la organización del código, para clasificar los datos se realizaron listas de los audios para extraer sus vectores y guardar las definiciones de los descriptores obtenidas en los códigos.

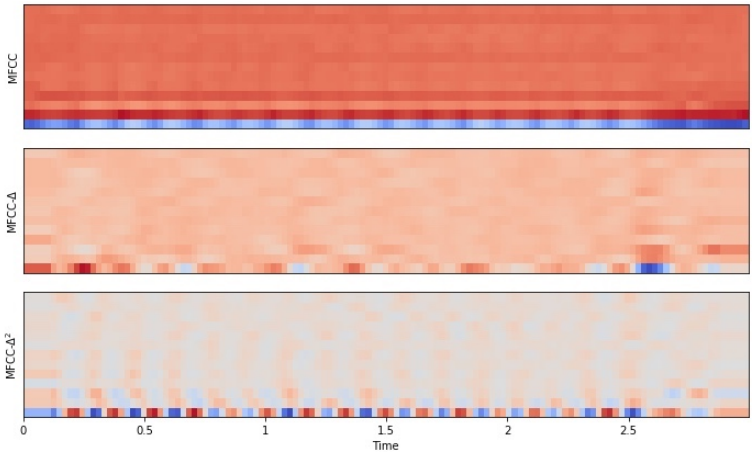


Figura 38.: *MFCC*

Se realizó un algoritmo titulado *Descriptors.py*[7] en el cual se incluyeron todos los descriptores hasta ahora presentados. Este código es el creador de los *json* que son los formatos de texto donde se guarda la información de los descriptores de cada audio.

El algoritmo denominado *File Manager.py* fue diseñado como un diccionario. Aquí se manejan los archivos de audio para obtener su longitud. En este algoritmo se corrobora si existen los archivos *json* para los audios, si es así estos se guardan en el *path* en forma de arreglo para poder clasificar esa información.

Se creó un algoritmo titulado *main*, cuya función es llamar a los algoritmos de *Descriptors.py y File Manager.py* como bibliote-

7 Consultar en el anexo

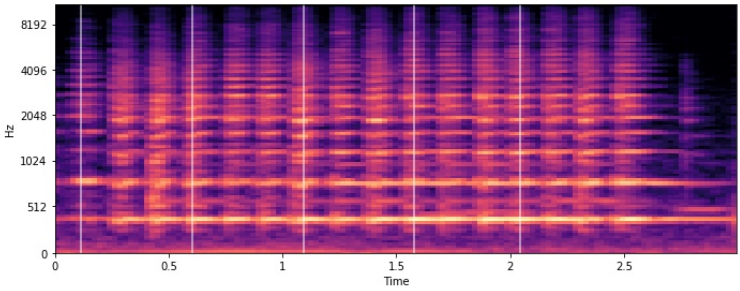


Figura 39.: *Beat Tracking*

cas. Este código regresa la lista de audio con *paths*, contiene una versión para que cada que se experimentara en este código se cambie el número de versión y así tener un control de los ejercicios realizados. También muestra los porcentajes de avance y si ya se cuenta con los archivos *jsons* en algún audio, arroja la leyenda *file already exists*, así se puede saber si el código está trabajando de la manera adecuada.

Para clasificar los datos se experimentó con códigos sobre Aprendizaje Automático. Para entrenar estos sistemas se escogieron tres métodos, *Support Vector Machine, K-Nearest Neighbors* y Redes Neuronales Convolucionales.

Las grabaciones permiten que se logre un aprendizaje de cada uno de los golpes de arco y así poder interpretarlas como un modelo temporal (Choi, Fazekas y Sandler 2016). Se requiere que los

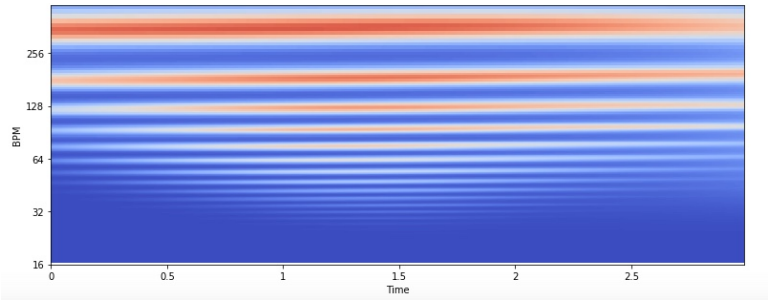


Figura 40.: *Tempograma*

descriptores aprendidos y el modelo temporal se optimicen conjuntamente, lo que podría implicar una ventaja en comparación con los métodos anteriores.

Es importante considerar que un elemento presente en los tres métodos utilizados es la matriz de confusión. Una matriz de confusión es una herramienta de inteligencia artificial que evalúa el desempeño de un algoritmo utilizado en el aprendizaje supervisado, a partir de un conteo de aciertos y errores de cada una de las clases en la clasificación (Van Son y col. 1994). La figura 44 se ejemplifica una matriz de confusión de dos clases:

donde:

- *a* es el número de predicciones correctas de que un caso es negativo.

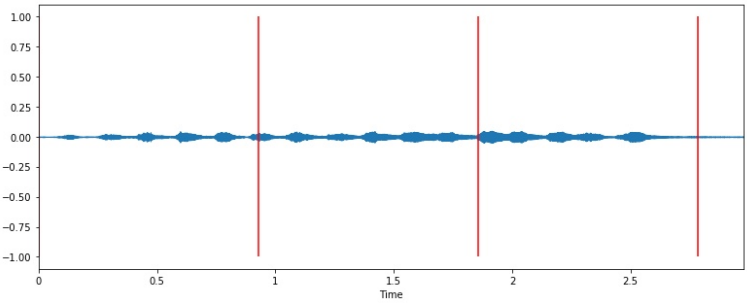


Figura 41.: *Beat Time*

- *b* es el número de predicciones incorrectas de que un caso es positivo, o sea la predicción es positiva cuando realmente el valor tendría que ser negativo. A estos casos también se les denomina errores de tipo I.

- *c* es el número de predicciones incorrectas de que un caso es negativo, o sea la predicción es negativa cuando realmente el valor tendría que ser positivo. A estos casos también se les denomina errores de tipo II.

- *d* es el número de predicciones correctas de que un caso es positivo.

Cada columna de la matriz representa el número de predicciones de cada clase, mientras que cada fila representa a las instancias en la clase real (Corso y Alfaro 2009). Por ejemplo, existen 900

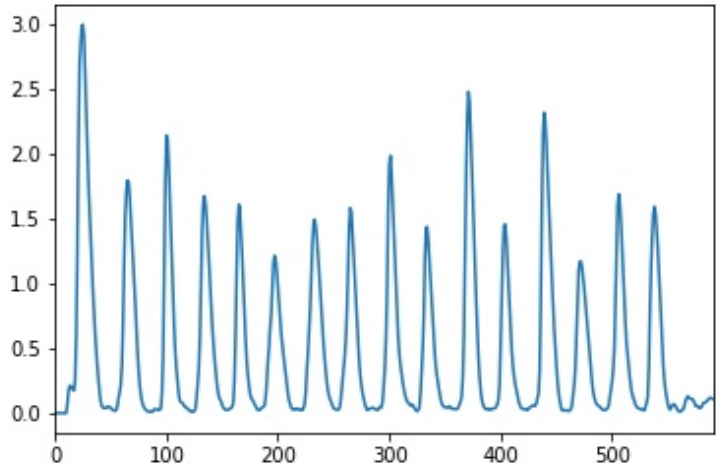


Figura 42.: *Onset*

muestras de la clase uno y solamente 100 muestras de la clase dos, con estos datos el clasificador puede tener una predisposición hacia la clase uno, si clasifica todas las clases como clase uno tendrá un 99 por ciento de precisión, lo cual no significa que sea un buen clasificador porque tuvo un error del 100 por ciento en la clasificación de las muestras de la clase dos.

A partir de lo enunciado es necesario abordar el tema de las redes neuronales convolucionales. Este método fue seleccionado para la aplicación porque al usar MFCC como entrada, puede explotar eficientemente las invariantes de tiempo y de frecuencia en

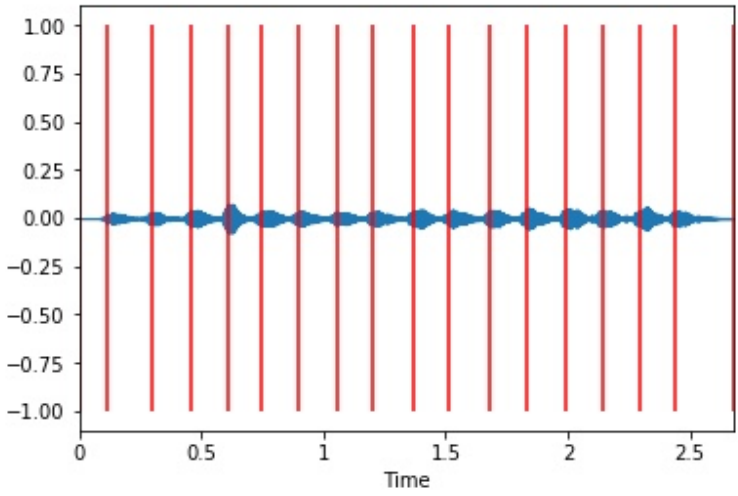


Figura 43.: *Onset Time*

los datos, debido a que comparte una cantidad reducida de parámetros, (Han y col. 2017).

El algoritmo titulado MLP[8] es el clasificador perteneciente a la red neuronal convolucional. Es de clase 4 y contiene 5 sub capas. El resultado que arrojó se muestra en la siguiente tabal (45):

Este método no es estable ya que al contener un estado aleatorio, al ser probado genera diferentes resultados en cada ocasión. En la primera tabla se puede observar que las cuatro clases tiene una precisión aproximada al 100 por ciento, mientras que en la segunda

8 Consultar en el anexo

		Predicción	
		Positivos	**Negativos**
Observación	**Positivos**	Verdaderos Positivos (VP)	Falsos Negativos (FN)
	Negativos	Falsos Positivos (FP)	Verdaderos Negativos (VN)

Figura 44.: Matriz de confusión

tabla, generada con los mismos datos, la clase cuatro alcanza un 100 por ciento de precisión y las dos primeras clases no tienen precisión alguna. Como consecuencia, este método resulta poco útil para el sistema, razón por la cual fue necesario implementar otro método.

Se utilizó *Support Vectors Machine (SVM)* por ser un algoritmo de aprendizaje supervisado que sirve para problemas de clasificación y regresión. Teniendo un conjunto de muestras se puede etiquetar las clases y entrenar un SVM para construir un modelo que prediga la clase de una nueva muestra.

Un SVM es un modelo que representa a los puntos de muestra en el espacio, separando las clases a dos espacios lo más amplios posibles mediante un hiperplano de separación definido como el

	precision	recall	f1-score	support
0	0.95	0.99	0.97	141
1	0.95	0.95	0.95	133
2	1.00	0.99	0.99	208
3	0.92	0.87	0.89	52
accuracy			0.97	534
macro avg	0.96	0.95	0.95	534
weighted avg	0.97	0.97	0.97	534

Figura 45.: Tabla MLP

	precision	recall	f1-score	support
0	0.00	0.00	0.00	97
1	0.00	0.00	0.00	94
2	0.40	1.00	0.57	150
3	1.00	0.15	0.26	41
avg / total	0.26	0.41	0.25	382

Figura 46.: Tabla MLP

vector entre los dos puntos, de las dos clases más cercanos al que se llama vector soporte, (Scholkopf y Smola 2001). Cuando las nuevas muestras se ponen en correspondencia con dicho modelo, en función de los espacios a los que pertenezcan, pueden ser clasificadas a una o la otra clase.

El sistema se entrenó con una pequeña cantidad de muestras obteniendo el siguiente resultado (47):

	precision	recall	f1-score	support
0	0.73	0.82	0.77	62
1	0.70	0.81	0.75	53
2	0.97	0.95	0.96	95
3	1.00	0.30	0.46	20
accuracy			0.83	230
macro avg	0.85	0.72	0.74	230
weighted avg	0.85	0.83	0.82	230

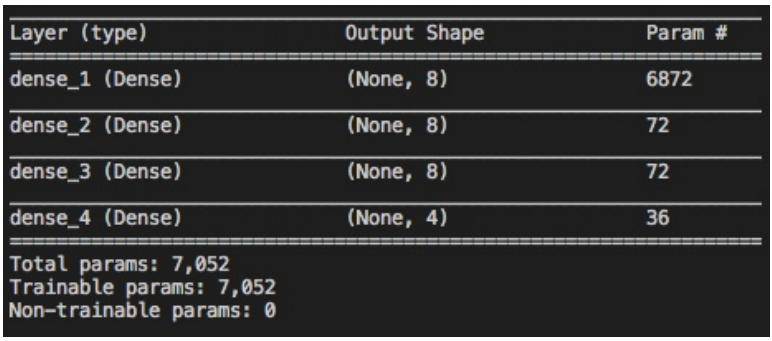
Figura 47.: Tabla SVC

A diferencia de la red neuronal, el resultado fue consistentemente el mismo aún cuando se ejecutó el algoritmo en varias ocasiones. En este la precisión de las dos primeras clases es de un 70 por ciento, mientras que la cuarta capa tiene una precisión del 100 por ciento y la tercera se aproxima a éste porcentaje.

K-Nearest Neighbors es un método de clasificación supervisada que sirve para estimar la función de densidad $F = (x/C_j)$ en cada clase C_j. En el reconocimiento de patrones, el algoritmo k-nn es usado como método de clasificación de objetos basado en un entrenamiento mediante ejemplos cercanos en el espacio de los elementos. Es un tipo de aprendizaje impreciso, donde la función se aproxima sólo localmente y todo el cómputo es diferido a la clasificación, (Zouhal y Denoeux 1998).

De acuerdo con la descripción del algoritmo (que puede consultarse en el anexo), el resultado que arroja es una precisión del 65

por ciento con 611 muestras entrenadas y 153 muestras probadas. La tabla de resultados que genera el algoritmo es el cálculo de la densidad de cada clase, (48).



Layer (type)	Output Shape	Param #
dense_1 (Dense)	(None, 8)	6872
dense_2 (Dense)	(None, 8)	72
dense_3 (Dense)	(None, 8)	72
dense_4 (Dense)	(None, 4)	36

Total params: 7,052
Trainable params: 7,052
Non-trainable params: 0

Figura 48.: Tabla K-nn

3.5 IMPLEMENTACIÓN DEL SISTEMA EN UNA WEBAPP

Para implementar el sistema en una página web se utilizó la plataforma heroku. A través de ésta se pueden revisarse los tres métodos de entrenamientos del sistema. Cada método permite reconocer los tres golpes de arco con diferente precisión. Cabe mencionar que sólo reconocerá *el détaché, el spiccato y el martelé*, en caso de ejecutar uno distinto, el sistema buscará, por aproximación, el más similar a los existentes dando esto como resultado. Para su implementación se ocupó la librería *meyda.js*(Rawlinson, Segal y Fiala

2015). Esta librería proporciona una base de investigación para un mayor desarrollo de la extracción de características en tiempo real de un audio web. El código de implementación puede ser consultado en el anexo.

4

INTERFAZ DE USUARIO

La interfaz de usuario es uno de los elementos importantes del sistema desarrollado en esta investigación, ya que es la interacción directa que tendrá el usuario con la aplicación. Por esta razón se buscó que el diseño de este sistema fuera lo más simple posible. Así se espera que el manejo del sistema resulte sencillo para el usuario, considerado también que las complicaciones de uso y manejo de un sistema muchas veces hacen al usuario desistir del uso de las aplicaciones.

A lo largo de este capitulo se describirán las 3 posibles formas de interacción con el sistema.

4.1 FUNCIONAMIENTO DE LA INTERFAZ

Como se mencionó en el párrafo anterior, se describen tres formas posibles de interacción con el sistema:

- Interactiva. En ésta, el usuario toca algún pasaje y el sistema detectará qué golpe de arco es el que está ejecutando.
- Línea de comando con archivos separados. Esta interacción se enfoca a los programadores más que a los ejecutantes del instrumento, ya que mediante grabaciones previamente cargadas, el sistema las revisa y detecta que golpe de arco es el que se ejecuta en las grabaciones.
- Etiquetado automático. Se puede introducir algún audio de alguna pieza musical y el programa etiquetará en que momentos de la música se encuentran los tres tipos de golpes de arco con los que se entrenó este sistema (*détaché, martelé y spiccato*).

A continuación se irá presentando por pantallas la interfaz propuesta para el sistema:

- Inicio del Sistema Interactivo. En la siguiente imagen se muestra la pantalla de inicio del sistema interactivo: en la esquina superior izquierda se encuentra el botón de menú

donde se pueden encontrar las otras dos formas de interactuar (En línea de comando con archivos separados y archivos grandes con etiqueta automática). En esta pantalla el usuario tocará algún tipo de golpe de arco, el sistema reconocerá si es alguno de los tres con los que fue entrenado (*détaché, spiccato, martelé*). En el ejemplo se muestra que el golpe de arco que detectó la webapp fue *spiccato* (figura 49).

Figura 49.: Sistema Interactivo

- Línea de comandos con archivos separados. En el botón de cargar archivos se subirán los archivos de audio que serán analizados por el sistema. Después de cierto tiempo en que el sistema termina el análisis, muestra el resultado en la pantalla con el nombre del golpe de arco que contienen los archivos de audio (figura 50).

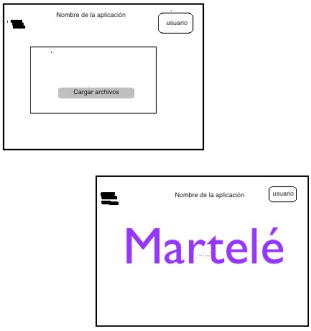


Figura 50.: Linea de comandos con archivos separados

- Etiquetado Automático. Al desplegar el menú y elegir esta opción, se pedirá subir el audio a analizar. Al terminar el programa la comparación muestra qué tipos de golpe de arco se encuentran en la grabación y en qué tiempos (figura 51).

Con lo expuesto anteriormente se puede notar que el sistema interactivo cuenta con herramientas básicas de apoyo en el estudio

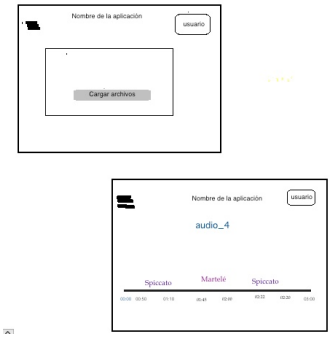


Figura 51.: Etiquetado automático

del violín como afinador y metrónomo. Pasando a la característica particular de este sistema que son las técnicas de arco *détaché, spiccato, martelé* se utilizaron gráficos sencillos, para la ejecución de la técnica de arco, mediante un semáforo se califica la interpretación del golpe y para detección muestra la señal de entrada y abajo de esta te dice qué tipo de técnica fue la que se ejecutó. Este sistema entrenado por *Python* se puede transferir a web por medio de Tensorflow.js (Abadi y col. 2016).

5

CONCLUSIONES

Antes que nada debe asentarse que al término de este trabajo el software presentado es funcional. De manera sobresaliente, no solo se puede corroborar la funcionalidad del software sino que además el punto clave y el aporte más significativo es la creación de un nuevo algoritmo. Este nuevo algoritmo no requiere un circuito conectado al violín para obtener la información deseada y con ello se convierte en una herramienta práctica para el intérprete y de fácil uso, ya que sólo necesita su instrumento, debido a que el sistema fue entrenado sólo por medio de grabaciones.

El proceso de investigación concluye con esta etapa, es decir, con la funcionalidad del sistema; sin embargo, tras dos años de trabajo, programación, investigación, contacto con músicos, etc. estoy convencida de que la implementación de esta aplicación en la práctica cotidiana de los músicos puede traducirse en una herra-

mienta que permita mejorar eficaz y significativamente el estudio de los golpes de arco trabajados a lo largo de la tesis.

Mi inquietud por desarrollar una herramienta para el instrumento que interpreto y que tuviera una aplicación novedosa en el ámbito de la tecnología musical pero también en el campo de la interpretación musical delimitó mi investigación al estudio de los golpes de arco.

Como consecuencia después de esta tesis he podido llegar a las siguientes conclusiones:

La experimentación de los algoritmos de los descriptores de audio fue un tanto larga, primero se realizó con los audios completos, es decir, de duración de un minuto aproximadamente que es lo que duran los ejercicios realizados para este sistema. Al hacerlo de esta manera los algoritmos de *tempo estimation* se creían los más viables para realizar la clasificación, pero al hacer la segmentación de los audios por compás y así obtener casi mil audios, se hicieron los mismos experimentos para cada audio y se concluyó que el mejor descriptor era el de *pitch detection*, ya que detecta cada golpe de arco ejecutado. Con este descriptor se realizaron las pruebas para los clasificadores; la primera idea viable que se tenía por la investigación es que se ocuparía *Transfer Learning* ya que esta técnica no utiliza tantos ejemplos para entrenar el sistema porque tiene una base de datos que contiene ejemplos parecidos a los que se ocu-

pan, después se concluyó que lo mejor era usar una red neuronal utilizando TensorFlow.

Utilizar el aprendizaje automático para entrenar el sistema fue una opción acertada, porque se obtuvo una aplicación con un uso más sencillo ya que no se ocupan sensores para la detección de los movimiento del arco. Al realizar las grabaciones, hacer segmentación de los audios, usar algoritmos para los descriptores de audio y clasificarlos por redes neuronales hicieron que fuera compatible para transferirla a una webapp y los violinistas pudieran utilizarla con facilidad.

Cabe mencionar que también se utilizó la librería *Essentia*[1] para obtener de manera más sencilla los archivos *json* que se ocuparon para etiquetar los audios y clasificar los datos para entrenar el sistema. Una de las conclusiones aledañas a esta investigación fue descubrir la viabilidad y la sencillez del uso de esta librería. Consecuentemente puedo asentar que este descubrimiento será de ayuda en futuras investigaciones; no se continuó experimentando con ella ya que se tenía un gran avance de trabajo con la biblioteca *librosa*.

1 *Essentia* es un extractor de funciones de línea de comandos configurable que calcula un gran conjunto de descriptores espectrales, de tiempo, de ritmo, tonales y de alto nivel. El uso de este extractor es una forma sencilla de obtener muchos descriptores de música sin ninguna programación. Contiene aproximadamente 100 descriptores de nivel bajo, ritmo y tonal.

Este sistema clasifica solamente los tres golpes de arco que se abordaron en esta investigación. Si se llega a ejecutar otro golpe de arco diferente, el sistema tratará de clasificarlo con los datos que se ocuparon para entrenarlo, así que el resultado será el golpe de arco que más se asemeje. Por tanto, se requiere una gran cantidad de ejemplos para entrenar el sistema y que este sea robusto.

Como suele suceder en todo trabajo de investigación, pese a que por ahora se ha llegado a un punto final, la creación de este sistema abre las puertas a trabajos y desarrollos futuros. Entre otras cosas y de manera sobresaliente la forma en la que esta desarrollada la aplicación permite la implementación de nuevas herramientas, por ejemplo, ahora se puede añadir no sólo los tres golpes de arco trabajados hasta el momento sino todos los golpes de arco existentes.

Aunado a esto también puede implementarse la medición del la fuerza del dedo índice de la mano derecha sobre el arco del violín, hecho que contribuye a mejorar la noción del cuerpo del intérprete sobre el instrumento y la búsqueda de un sonido satisfactorio para el ejecutante.

Entre otras cosas la aplicación puede enriquecerse con la añadidura de otros métodos tecnológicos como *score following* y *pitch detection*. Estos métodos abrirían el panorama de la interfaz de usuario para el violinista y sería una manera gráfica y sencilla para mejorar las condiciones de estudio del ejecutante.

Por último, el impacto de esta aplicación en el mundo de la interpretación musical puede engrandecerse implementándola a todos los instrumentos de cuerda frotada. La forma en la que está diseñada puede permitir dicha posibilidad y con ello el aporte sería significativo no sólo para los violinistas sino para que una mayor cantidad de músicos de cuerda frotada tengan una mejor aproximación en la relación de cuerpo-sonido.

A

ANEXO, CÓDIGOS

A.1 ENTRENAMIENTO DEL SISTEMA

Code Listing A.1: File Manager

```
import os
import json
import librosa
import numpy as np
import os.path

def getFileList(directory, extension='.wav'):
    # Muestra la lista de los audios
    listFiles = []
    # directory = '/content/drive/My Drive/audios/'
    for filename in os.listdir(directory):
        if filename.endswith(extension):
            path = os.path.join(directory, filename)
            listFiles.append([path, filename])
            continue
        else:
            continue

    return listFiles
```

```
def loadFileListLibrosa(listFiles, version):
    # Realizar el diccionario para todos los audios
    file_vectors_list = []
    print('Librosa loading '+str(len(listFiles))+' files')
    for i, file_name in enumerate(listFiles):
        print('\n'+file_name[0])
        if not checkIfJsonExists(file_name[0]+'.json', version):
            print('Librosa Loading Files:'+str(100*i/len(listFiles)))
            x, sr = librosa.load(file_name[0])
            file_obj = {}
            file_obj['x'] = x
            file_obj['sr'] = sr
            file_obj['path'] = file_name[0]
            file_obj['name'] = file_name[1]
            file_vectors_list.append(file_obj)
        else:
            print('file exists')
    return file_vectors_list


def save_file(data, filename="test.txt"):
    file = open(filename, 'w')
    file.write(data)
    file.close()


def printLengthKeyList(desc_obj):
    for k in desc_obj:
        print(k)
        if k != 'sr':
            print(len(desc_obj[k]))
            print(type(desc_obj[k]))


def checkIfJsonExists(filePath, version):
    # try to open file
    if os.path.exists(filePath):
        # read file
        with open(filePath, 'r') as myfile:
            data = myfile.read()
            obj = json.loads(data)
            if obj['version'] == version:
```

```
                return True
            else:
                return False
    else:
        return False


def writeObjectDisc(file_path, fileObject):
    # select keys to save
    ommit = ['x', 'onset_env', 'tempogram', 'bins_per_octave']
    objExport = {}
    for key in fileObject:
        # if array convert tkf:o regular list
        # print('KEY', type(fileObject))
        if not(key in ommit):
            if isinstance(fileObject[key], np.ndarray):
                objExport[key] = fileObject[key].tolist()
            else:
                objExport[key] = fileObject[key]
    # print(objExport)

    # save to path
    js_file = json.dumps(objExport)
    save_file(js_file, file_path)
    return objExport
```

Code Listing A.2: Descriptors.

```
  : import numpy as np
import librosa


def cqt(x, sr):
    bpoct = 37
    nbins = 300
    cqt = librosa.cqt(x, sr=sr, n_bins = nbins, bins_per_octave = bpoct)
    log_cqt = librosa.amplitude_to_db(np.abs(cqt))
    lis = log_cqt.flatten().tolist()
    return {"name": "cqt",
            "stats": {"bins_per_octave":bpoct,"number_bins":nbins},
            "shape":log_cqt.shape,
```

```
            "length":len(lis),
            "data":lis}


def onset_env(x, sr):
        # no uses el mismo nombre de la funci'on como nombres de variables
    hop_length = 100
    onset_env_val = librosa.onset.onset_strength(
            x, sr=sr, hop_length=hop_length)
    lis = onset_env_val.flatten().tolist()
    return {"name": "onset_env",
            "stats": {"hop_length":hop_length},
            "shape":onset_env_val.shape,
            "length":len(lis),
            "data":lis}
    return onset_env_val


def onset_samples(x, sr):
    onset_samples_val = librosa.onset.onset_detect(x, sr=sr,
            units='samples',
            hop_length=100,
            backtrack=False,
            pre_max=20,
            post_max=20,
            pre_avg=100,
            post_avg=100,
            delta=0.2,
            wait=0)
    lis = onset_samples_val.flatten().tolist()
    return {"name": "onset_samples",
            "stats": {
            "units":'samples',
            "hop_length":100,
            "backtrack":"False",
            "pre_max":20,
            "post_max":20,
            "pre_avg":100,
            "post_avg":100,
            "delta":0.2,
            "wait":0},
            "shape":onset_samples_val.shape,
            "length":len(lis),
            "data":lis}
```

```
    return onset_samples_val


def estimate_pitch(x, sr, fmin=50.0, fmax=2000.0):
    # Compute autocorrelation of input segment.
    r = librosa.autocorrelate(x)
    # Define lower and upper limits for the autocorrelation argmax.
    i_min = sr/fmax
    i_max = sr/fmin
    r[:int(i_min)] = 0
    r[int(i_max):] = 0
    # Find the location of the maximum autocorrelation.
    i = r.argmax()
    f0 = float(sr)/i
    return {"name": "estimate_pitch",
            "stats": {"fmin":50.0,"fmax":2000 },
            "shape": 1,
            "length":1,
            "data":[f0]}

def mfcc(x, sr):
    # Let's make and display a mel-scaled power
    # (energy-squared) spectrogram
    nmels = 128
    S = librosa.feature.melspectrogram(x, sr=sr, n_mels = nmels)
    # Convert to log scale (dB).
    # We'll use the peak power (max) as reference.
    log_S = librosa.power_to_db(S, ref=np.max)
    # extract the top 13 Mel-frequency cepstral coefficients (MFCCs)
    mfcc_val = librosa.feature.mfcc(S=log_S, n_mfcc=13)
    lis = mfcc_val.flatten().tolist()
    return {"name": "mfcc",
            "stats": {
                "n_mels":nmels},
            "shape":mfcc_val.shape,
            "length":len(lis),
            "data":lis}


def tempo(x, sr):
    return librosa.beat.tempo(x, sr=sr)
```

```
def beat_times(x, sr):
    T = len(x)/float(sr)
    # here you are calling tempo, but you can't access it
    # you need to pass it as a reference or calculate it again
    temp = tempo(x, sr)
    seconds_per_beat = 120.0/temp
    beat_times_val = np.arange(0, T, seconds_per_beat)
    return beat_times_val


# Let's pad on the first and second deltas while we're at it
    # delta_mfcc  = librosa.feature.delta(mfcc)
    # delta2_mfcc = librosa.feature.delta(mfcc, order=2)


def createTagFromName(fileName):
    # if nombre del archivo tiene detache
    # regresa detahce
    # if normbre incluye spicctao.
    if 'martele' in fileName.lower():
        return 'martele'
    elif 'detache' in fileName.lower():
        return 'detache'
    elif 'portato' in fileName.lower():
        return 'portato'
    else:
        return 'spiccato'


def createDescriptorObject(file_obj, version):
    # creamos una variable temporal x con la lista  a numpy para
    # hacer los calculos
    # Agregar las dimensiones de cada descriptor
    x = file_obj['x']
    sr = file_obj['sr']
    file_obj['version'] = version
    file_obj['cqt'] = cqt(x, sr)
    file_obj['onset_env'] = onset_env(x, sr)
    file_obj['onset_samples'] = onset_samples(x, sr)
    file_obj['estimate_pitch'] =estimate_pitch(x, sr, fmin=50.0, fmax=2000.0)
    file_obj['mfcc'] = mfcc(x, sr)

    return file_obj
```

Code Listing A.3: Data Explorer.

```
import fileManager as files
import json
import sys
# Cargar todos los json


path = sys.argv[1]


f = files.getFileList(path, 'json')
detacheLen = []
spiccatoLen = []
marteleLen = []
portatoLen = []

for e in f:
    # print(e)
    with open(e[0]) as json_file:
        data = json.load(json_file)
        if 'spiccato' in data['name'].lower():
            spiccatoLen.append(data['mfcc']['shape'][1])
        if 'detache' in data['name'].lower():
            detacheLen.append(data['mfcc']['shape'][1])
        if 'martele' in data['name'].lower():
            marteleLen.append(data['mfcc']['shape'][1])
        if 'portato' in data['name'].lower():
            portatoLen.append(data['mfcc']['shape'][1])

print(min(spiccatoLen))
print(min(detacheLen))
print(min(marteleLen))
print(min(portatoLen))
```

Code Listing A.4: Main.

```
import fileManager as files
import descriptors as desc
import pprint
```

```
import sys

print(sys.argv)

# path = 'audio'
path = sys.argv[1]

# Returns list of audio paths and names
listFile = files.getFileList(path)

for f in listFile:
    print(f)


version = '1.41'
# Load files to librosa
# maybe only load the objects we will need
fileObjectList = files.loadFileListLibrosa(listFile, version)
print('\n\n')
# its better to batch load in case I need to interrupt the program


for idx, f in enumerate(fileObjectList):
    print('\n'+str(100*idx/len(fileObjectList))+'%\n\n')
    # files.printLengthKeyList(f)
    # check if object with current version exist
    if not files.checkIfJsonExists(f['path']+'.json', version):
        print('creating object', f['path'])
        desc_obj = desc.createDescriptorObject(f, version)
        files.writeObjectDisc(
                'audio/analysisFiles/'+desc_obj['name']+'.json', desc_obj)
    else:
        print('\n\nfile already exists')
```

Code Listing A.5: Prepare Data.

```
import fileManager as files
import json
import sys
# Cargar todos los json


path = sys.argv[1]
```

```
f = files.getFileList(path, 'json')
detacheLen = []
spiccatoLen = []
marteleLen = []
portatoLen = []
X = []
Y = []

for e in f:
    print(e[1])
    with open(e[0]) as json_file:
        data = json.load(json_file)
        X.append(data['mfcc']['data'])
        if 'spiccato' in data['name'].lower():
            Y.append(0)
        if 'detache' in data['name'].lower():
            Y.append(1)
        if 'martele' in data['name'].lower():
            Y.append(2)
        if 'portato' in data['name'].lower():
            Y.append(3)

# Normalize X to smallest length
lengths = [len(elem) for elem in X]
minLen = min(lengths)
Xnorm = [elem[:minLen] for elem in X]


writeData ={ "descriptors":Xnorm, "classes":Y }

with open('MAIN/data.json', 'w') as outfile:
    json.dump(writeData, outfile)
```

Code Listing A.6: Keras NN.

```
from __future__ import print_function

import keras
from keras.datasets import mnist
from keras.models import Sequential
```

```
from keras.layers import Dense, Dropout
from keras.optimizers import RMSprop
from sklearn.model_selection import train_test_split
import numpy as np
import json


# Read data

with open('MAIN/data.json') as json_file:
    data = json.load(json_file)
    # print(data)
    dataX = np.array(data['descriptors'] )
    dataY = np.array(data['classes'])

x_train, x_test, y_train, y_test = train_test_split(
    dataX, dataY, test_size=0.2, shuffle=False)
print(type(x_train))
print(x_train.shape)
print(y_train.shape)
print(x_test.shape)
print(y_test.shape)



batch_size = 100
num_classes = 4
epochs = 30
input_size= 858

print()
# # the data, split between train and test sets
# (x_train, y_train), (x_test, y_test) = mnist.load_data()
# print(type(x_train))
# print(x_train.shape)
# print(y_train.shape)
# print(x_test.shape)
# print(y_test.shape)

# x_train = x_train.reshape(60000, 784)
# x_test = x_test.reshape(10000, 784)
x_train = x_train.astype('float32')
x_test = x_test.astype('float32')
# x_train /= 255
```

```
# x_test /= 255
print(x_train.shape[0], 'train␣samples')
print(x_test.shape[0], 'test␣samples')

# convert class vectors to binary class matrices
y_train = keras.utils.to_categorical(y_train, num_classes)
y_test = keras.utils.to_categorical(y_test, num_classes)

model = Sequential()
model.add(Dense(8, activation='relu', input_shape=(input_size,)))
model.add(Dense(8, activation='relu'))
model.add(Dense(8, activation='relu'))
# model.add(Dense(8, activation='relu'))
# model.add(Dropout(0.2))
model.add(Dense(num_classes, activation='softmax'))

model.summary()

model.compile(loss='categorical_crossentropy',
              optimizer=RMSprop(),
              metrics=['accuracy'])

history = model.fit(x_train, y_train,
                    batch_size=batch_size,
                    epochs=epochs,
                    verbose=1,
                    validation_data=(x_test, y_test))
score = model.evaluate(x_test, y_test, verbose=0)
print('Test␣loss:', score[0])
print('Test␣accuracy:', score[1])

print('\nSaving␣model')
model.save('model.h5')
```

Code Listing A.7: Load kerasNN

```
    from __future__ import absolute_import, division, print_function, unicode_literals
from tensorflow import keras
from tensorflow.keras import layers
import tensorflow as tf
import numpy as np
import json
```

```
with open('MAIN/data.json') as json_file:
    data = json.load(json_file)
    dataX = np.array(data['descriptors'] )
    dataY = np.array(data['classes'])

model = keras.models.load_model('model.h5')

# Here goes the input data
elem = np.array([dataX[0],])

prediction = model.predict(elem)[0]
predictionClass = list(prediction).index(max(prediction))

print(predictionClass)
```

Code Listing A.8: MLP (Multi Layer Perceptron).

```
        from sklearn.model_selection import train_test_split
from sklearn.neural_network import MLPClassifier
from sklearn.metrics import classification_report,confusion_matrix
import json
import numpy as np

with open('MAIN/data.json') as json_file:
    data = json.load(json_file)
    # print(data)
    dataX = data['descriptors']
    dataY = data['classes']

# print(dataY)
X = np.array(dataX)
y = np.array(dataY)

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.50, random_state=40)
print(X_train.shape); print(X_test.shape)

clf = MLPClassifier(hidden_layer_sizes=(6,6,6,6,6), activation='relu', solver='adam', max_iter=500)
# clf = MLPClassifier(solver='lbfgs', alpha=1e-5, hidden_layer_sizes=(5, 2), random_state=1)
clf.fit(X_train, y_train)

predict_train = clf.predict(X_train)
predict_test = clf.predict(X_test)

print(confusion_matrix(y_train,predict_train))
```

```
print(classification_report(y_train, predict_train))
```

Code Listing A.9: SVC de Support Vector Machine.

```
import matplotlib.pyplot as plt
from sklearn import datasets, svm, metrics
from sklearn.model_selection import train_test_split
import json
import sys
import numpy as np

with open('MAIN/data.json') as json_file:
    data = json.load(json_file)
    # print(data)
    dataX = data['descriptors']
    dataY = data['classes']


n_samples = len(dataX)
classifier = svm.SVC(gamma='scale')
# Split data into train and test subsets
X_train, X_test, y_train, y_test = train_test_split(
    dataX, dataY, test_size=0.3, shuffle=False)

# We learn the digits on the first half of the digits
classifier.fit(X_train, y_train)

# Now predict the value of the digit on the second half:
predicted = classifier.predict(X_test)

print("Classification report for classifier %s:\n%s\n"
      %(classifier, metrics.classification_report(y_test, predicted)))
disp = metrics.plot_confusion_matrix(classifier, X_test, y_test)
disp.figure_.suptitle("Confusion Matrix")
print("Confusion matrix:\n%s" % disp.confusion_matrix)

print('predict one')
num = 4
elem = np.array([X_test[num],])
predicted = classifier.predict(elem)
# plt.show()
print(y_test[num])
print(predicted)
```

A.2 IMPLEMENTACIÓN DE LA WEBAPP

Code Listing A.10: Routes.

```
from flask import request, render_template


def configure_routes(app):
    @app.route('/')
    def get_main():
        return render_template('index.html')

    @app.route('/svn', methods=['POST'])
    def post_svn():
        print('Enter post svn')
        # print(request.files['soundBlob'])
        # print(request.json)
        content = request.get_json()
        print(content)
        return {"value": "calculo svn"}, 200

    @app.route('/mlp')
    def get_mlp():
        return {"value": "calculo mlp"}, 200

    @app.route('/kerasnn')
    def get_kerasnn():
        return {"value": "calculo kerasnn"}, 200
```

Code Listing A.11: App.

```
from flask import Flask
from routes import configure_routes
from dotenv import load_dotenv
from pathlib import Path  # python3 only
from flask_cors import CORS
import os
env_path = Path('.') / 'devLocal.env'
load_dotenv(dotenv_path=env_path)
```

```
PORT = os.getenv('PORT')
print('Port')
print(PORT)


app = Flask(__name__)
app.config['SEND_FILE_MAX_AGE_DEFAULT'] = 0
app.config['CORS_HEADERS'] = 'Content-Type'
CORS(app)
configure_routes(app)

if __name__ == '__main__':
    app.run(port=PORT, threaded=True)
```

Code Listing A.12: Analysis.

```
/* globals Meyda */
console.log('MAIN')


let input;
let analyzer;

// let global_isRec = false;
// let global_threshold = 0.3;

function setup() {
    console.log('enter start')

    createCanvas(710, 200);
    background(255);
    input = new p5.AudioIn();
    input.start();
    userStartAudio().then(function() {
        console.log('Audio start');
        mainAudio()
    });
}

function mainAudio() {

    const audioContext = new AudioContext();
```

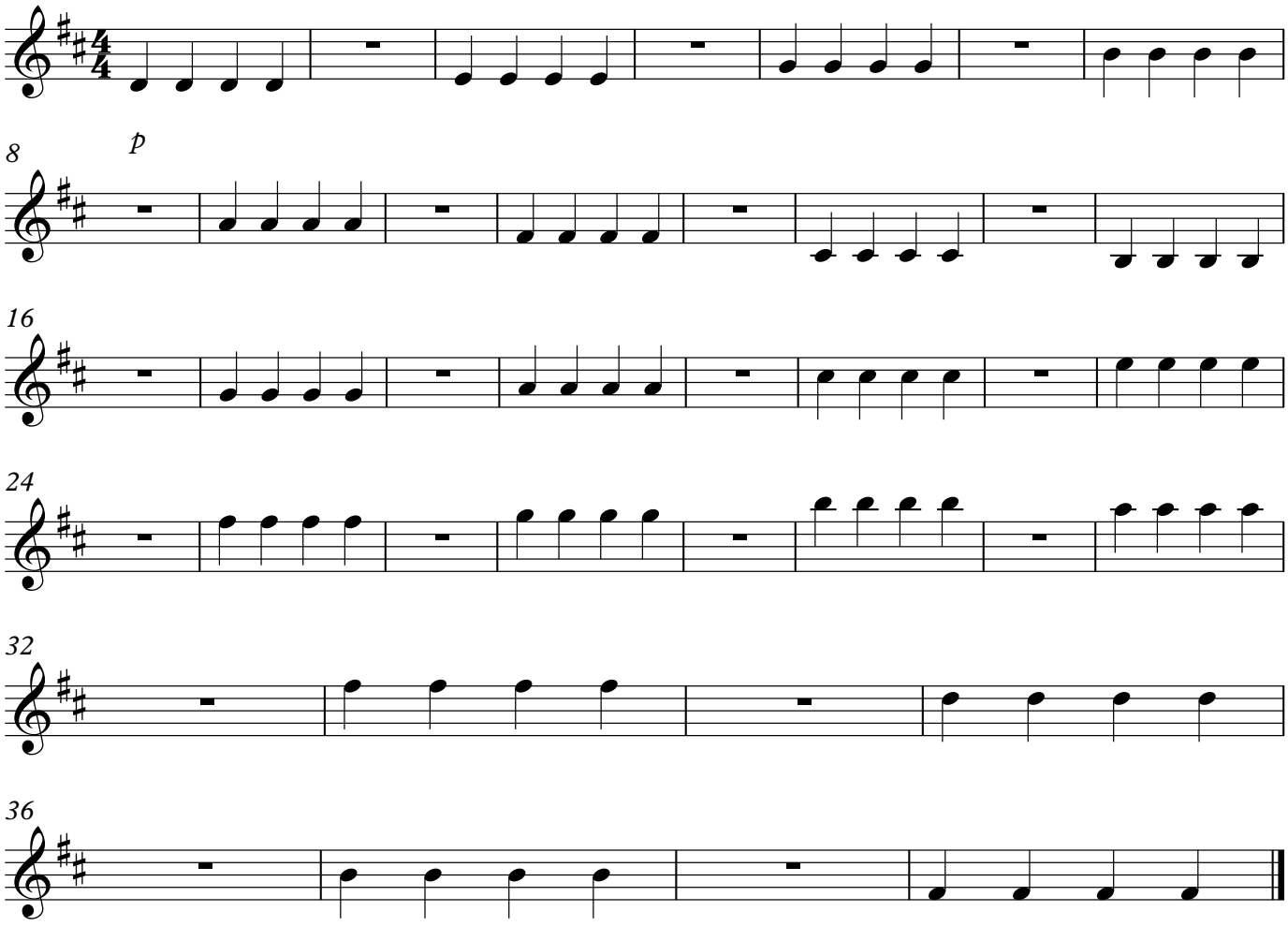
```
const htmlAudioElement = document.getElementById("audio");
const source = audioContext.createMediaElementSource(htmlAudioElement);
source.connect(audioContext.destination);
const levelRangeElement = document.getElementById("levelRange");

let global_recording = false
let global_threshold = 0.05
let mfccArray = []


if (typeof Meyda === "undefined") {
    console.log("Meyda could not be found! Have you included it?");
}
else {
    const analyzer = Meyda.createMeydaAnalyzer({
        "audioContext": audioContext,
        "source": source,
        "bufferSize": 512,
        "featureExtractors": ["mfcc"],
        "callback": features => {
            if(global_recording  ){
                console.log('START REC')
                if (mfccArray.length<66) {// To create the 858 vector needed
                    mfccArray.push(features.mfcc)
                }else {
                    console.log('Sending mfcc')
                    console.log('STOP REC')
                    sendRequest(mfccArray)
                    console.log(mfccArray.length)
                    global_recording= false
                    mfccArray = []
                }
            }
        }
    });

    analyzer.start();
    const rms = Meyda.createMeydaAnalyzer({
        "audioContext": audioContext,
        "source": source,
        "bufferSize": 512,
        "featureExtractors": ["rms"],
        "callback": features => {
            // console.log(features.rms);
```

```
                global_rms = features.rms
                if(features.rms > global_threshold) global_recording=true
            }
        });
        rms.start();
    }

}

function sendRequest(mfccData) {
    // flatten array
    console.log(mfccData)
    const message = {data:mfccData}
    let options = {
        method: 'POST',
        mode: 'cors',
        headers: {
            'Content-Type': 'application/json'
        },
        body:JSON.stringify(message)
    };
    fetch('/svn',options)
        .then((response) => {
            return response.json();
        })
        .then((myJson) => {
            console.log(myJson);
        });
}
```

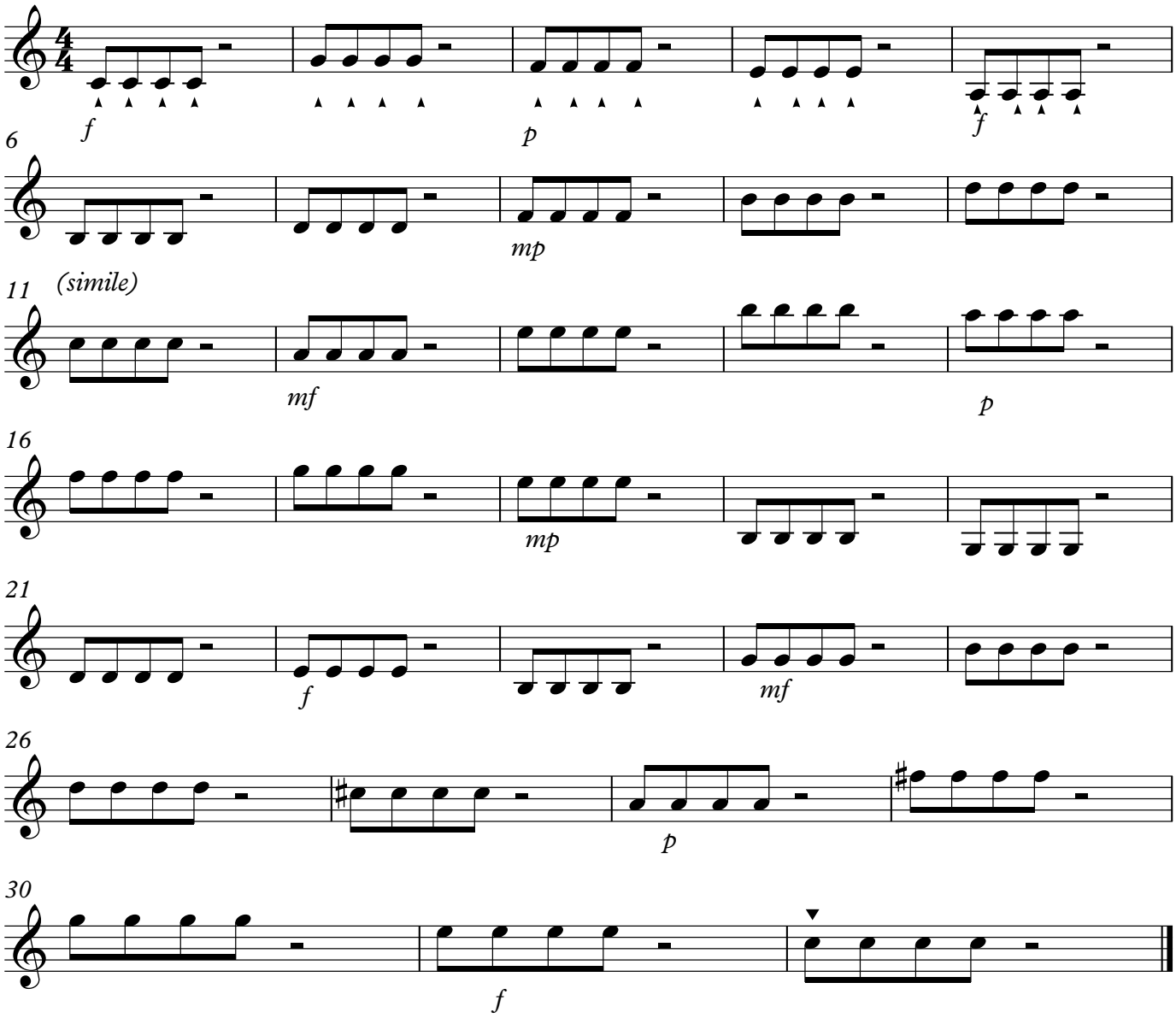
B

ANEXO, PARTITURAS

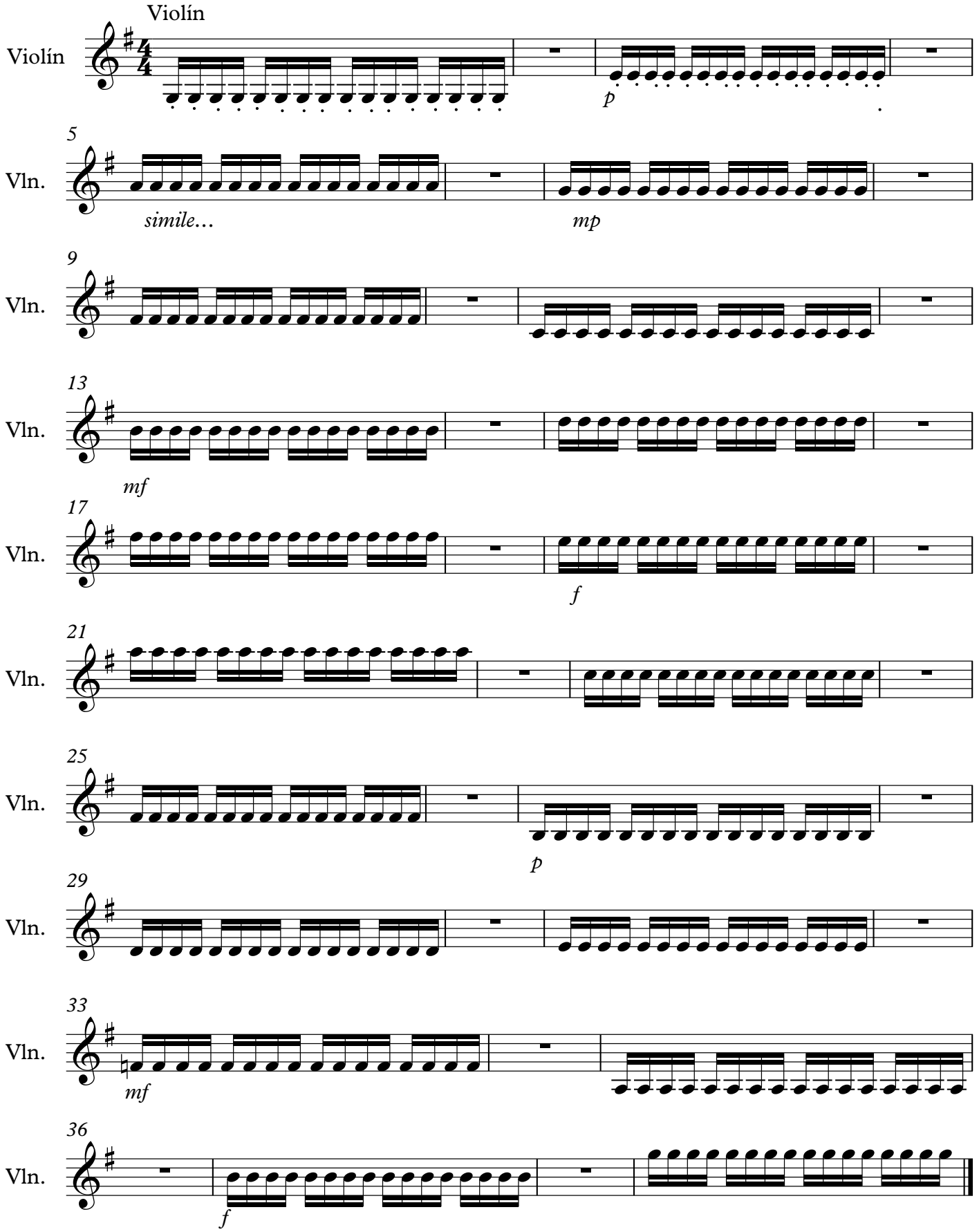
Détaché



Martelé



Spiccato


Violín
Violín
p
5
Vln.
simile…
mp
9
Vln.
13
Vln.
mf
17
Vln.
f
21
Vln.
25
Vln.
p
29
Vln.
33
Vln.
mf
36
Vln.
f

BIBLIOGRAFÍA

Abadi, Martín y col. (2016). «Tensorflow: A system for large-scale machine learning». En: *12th {USENIX} Symposium on Operating Systems Design and Implementation ({OSDI} 16)*, págs. 265-283.

Austin, Katherine A (2009). «Multimedia learning: Cognitive individual differences and display design techniques predict transfer learning with multimedia learning modules». En: *Computers & Education* 53.4, págs. 1339-1354.

Bedoya-Jaramillo, S y col. (2012). «Automatic emotion detection in speech using mel frequency cesptral coefficients». En: *2012 XVII Symposium of Image, Signal Processing, and Artificial Vision (STSIVA)*. IEEE, págs. 62-65.

Bello, Juan Pablo y col. (2005). «A tutorial on onset detection in music signals». En: *IEEE Transactions on speech and audio processing* 13.5, págs. 1035-1047.

Bonada, Jordi y Xavier Serra (2007). «Synthesis of the singing voice by performance sampling and spectral models». En: *IEEE signal processing magazine* 24.2, págs. 67-79.

Brandao, Márcio, Geraint Wiggins y Helen Pain (1999). «Computers in music education». En: *Proceedings of the AISB'99 Symposium on Musical Creativity*. Division of Informatics,University of Edinburgh, págs. 82-88.

Choi, Keunwoo, George Fazekas y Mark Sandler (2016). «Automatic tagging using deep convolutional neural networks». En: *arXiv preprint arXiv:1606.00298*.

Ciresan, Dan C y col. (2011). «Flexible, high performance convolutional neural networks for image classification». En: *IJCAI Proceedings-International Joint Conference on Artificial Intelligence*. Vol. 22. 1. Barcelona, Spain, pág. 1237.

Corso, Cynthia Lorena y Sofía Lorena Alfaro (2009). «Alternativa de herramienta libre para la implementación de aprendizaje automático». En:

Crickboom, M (1994). «El violín: Teórico y práctico». En: *Crickboom–Schott-Frères El violin, teórico y práctico, II–M. Crickboom–Schott-Frères El violin, teórico y práctico, III–M. Crickboom–Schott-Frères El violín, teórico y práctico, IV–M. Crickboom–Schott-Frères Chants et morceaux, II–M. Crickboom–Schott-Frères*.

Dourado, Henrique Autran (2008). *O arco dos instrumentos de cordas*. Irmãos Vitale.

Ferreira, Eliseu y Sonia Ray (2006). «Planejamento de Arco na Prática Orquestral: considerações e aplicações em grupos semi-profissionais». En: *Anais do* 16, págs. 658-664.

Fujinaga, Ichiro (1998). «Machine recognition of timbre using steady-state tone of acoustic musical instruments.» En: *ICMC*.

Fukushima, Kunihiko y Sei Miyake (1982). «Neocognitron: A self-organizing neural network model for a mechanism of visual pattern recognition». En: *Competition and cooperation in neural nets*. Springer, págs. 267-285.

Galamian, Ivan (1998). *Interpretación y enseñanza del violín*. Pirámide.

Giannakopoulos, Theodoros (2015). «pyaudioanalysis: An open-source python library for audio signal analysis». En: *PloS one* 10.12, e0144610.

Gómez, Emilia y Perfecto Herrera (2004). «Estimating The Tonality Of Polyphonic Audio Files: Cognitive Versus Machine Learning Modelling Strategies.» En: *ISMIR*. Citeseer.

Gouyon, Fabien, François Pachet, Olivier Delerue y col. (2000). «On the use of zero-crossing rate for an application of classification of percussive sounds». En: *Proceedings of the COST G-6 conference on Digital Audio Effects (DAFX-00), Verona, Italy*. Audiovisual Institute, Pompeu Fabra University, Barcelona / Sony Computer Science Laboratory, Paris, pág. 26.

Grosche, Peter, Meinard Müller y Frank Kurth (2010). «Cyclic tempogram—A mid-level tempo representation for musicsignals». En: *Acoustics Speech and Signal Processing (ICASSP), 2010 IEEE International Conference on*. IEEE, págs. 5522-5525.

Gutiérrez, Mariano Pérez (1985). *Diccionario de la música y los músicos*. Vol. 87. Ediciones AKAL.

Han, Yoonchang y col. (2017). «Deep convolutional neural networks for predominant instrument recognition in polyphonic music». En: *IEEE/ACM Transactions on Audio, Speech and Language Processing (TASLP)* 25.1, págs. 208-221.

Isasi Viñuela, Pedro e IM Galván León (2004). «Redes de neuronas artificiales». En: *Un Enfoque Práctico, Editorial Pearson Educación SA Madrid España*.

Kanungo, Tapas y col. (2002). «An efficient k-means clustering algorithm: Analysis and implementation». En: *IEEE Transactions on Pattern Analysis & Machine Intelligence* 7, págs. 881-892.

Kayser, Heinrich y col. (1915). «Elementary and progressive Studies for Violin». En:

Keppy, Nicole K y Michael Allen (2016). «Understanding spectral bandwidth and resolution in the regulated laboratory». En: *Thermo Fisher Scientific Technical Note* 51721.

Klapuri, Anssi (1999). «Sound onset detection by applying psychoacoustic knowledge». En: *1999 IEEE International Con-*

ference on Acoustics, Speech, and Signal Processing. Proceedings. ICASSP99 (Cat. No. 99CH36258). Vol. 6. IEEE, págs. 3089-3092.

Kreutzer, Rodolphe (1915). *Forty-two studies for violin*. Carl Fischer.

Lin, Hsin-Ming y Chin-Ming Lin (2015). «Harmonic intonation trainer: an open implementation in pure data.» En: *NIME*. International Conference on New Interfaces for Musical Expression, Baton Rouge, LA, USA, págs. 38-39.

Logan, Beth y col. (2000). «Mel Frequency Cepstral Coefficients for Music Modeling.» En: *ISMIR*. Vol. 270. One Cambridge Center, págs. 1-11.

Madueño, L (2003). «Desarrollo de software educativo bajo plataforma Web». En: *Congreso Internacional EDUTEC*. Universidad del Zulia Venezuela.

Manjunath, Bangalore S, Philippe Salembier y Thomas Sikora (2002). *Introduction to MPEG-7: multimedia content description interface*. John Wiley & Sons.

McFee, Brian y col. (2015). «librosa: Audio and music signal analysis in python». En: *Proceedings of the 14th python in science conference*, págs. 18-25.

Mitchell, TM (1997). «Machine Learning, McGraw-Hill Higher Education». En: *New York*.

Mora-Castillo, Juan Antonio (2016). «Serialización/deserialización de objetos y transmisión de datos con JSON: una revisión de la literatura». En: *Revista Tecnología en Marcha* 29.1, ág-118.

Park, Sang Hyun (2013). «Musical Instrument Extraction through Timbre Classification». En: *NVIDIA Corporation Santa Clara, CA* 95050.

Pedro, Dionisio de (1990). *Teoría completa de la música: en dos volúmenes*. Real Musical.

Peeters, G (2004). «A large set of audio features for sound description (Technical report)». En: *Paris: IRCAM*.

Peiper, Chad, David Warden y Guy Garnett (2003). «An interface for real-time classification of articulations produced by violin bowing». En: *Proceedings of the 2003 conference on New interfaces for musical expression*. National University of Singapore, págs. 192-196.

Percival, Graham, Nicholas Bailey y George Tzanetakis (2011). «Physical modeling meets machine learning: Teaching bow control to a virtual violinist». En: *Sound and Music Conference*.

Pérez, Alfonso (2009). «Enhancing spectral synthesis techniques with performance gestures using the violin as a case study». En: *Department of Information and Communication Technologies*.

Piczak, Karol J (2015). «Environmental sound classification with convolutional neural networks». En: *2015 IEEE 25th International Workshop on Machine Learning for Signal Processing (MLSP)*. IEEE, págs. 1-6.

Rasamimanana, Nicolas H, Emmanuel Fléty y Frédéric Bevilacqua (2005). «Gesture analysis of violin bow strokes». En: *Gesture in human-computer interaction and simulation*. Springer, págs. 145-155.

Rasamimanana, Nicolas y Frédéric Bevilacqua (2008). «Effort-based analysis of bowing movements: evidence of anticipation effects». En: *Journal of New Music Research* 37.4, págs. 339-351.

Rawlinson, Hugh, Nevo Segal y Jakub Fiala (2015). «Meyda: an audio feature extraction library for the web audio api». En: *The 1st Web Audio Conference (WAC). Paris, Fr*.

Roederer, Juan G (2008). *The physics and psychophysics of music: an introduction*. Springer Science & Business Media.

Sakashita, Yuma y Masaki Aono (2018). «Acoustic scene classification by ensemble of spectrograms based on adaptive temporal divisions». En: *Detection and Classification of Acoustic Scenes and Events (DCASE) Challenge*.

Salas, Lewis McAnally y Javier Organista Sandoval (2007). «La educación en línea y la capacidad de innovación y cambio de las instituciones de educación». En: *Apertura* 7.7, págs. 82-94.

Scholkopf, Bernhard y Alexander J Smola (2001). *Learning with kernels: support vector machines, regularization, optimization, and beyond*. MIT press.

Schoonderwaldt, Erwin (2009). «The violinist's sound palette: spectral centroid, pitch flattening and anomalous low frequencies». En: *Acta Acustica united with Acustica* 95.5, págs. 901-914.

Schubert, Emery, Joe Wolfe y Alex Tarnopolsky (2004). «Spectral centroid and timbre in complex, multiple instrumental textures». En: *Proceedings of the international conference on music perception and cognition, North Western University, Illinois*. UNSW, Sydney, School of Physics, Faculty of Science, Australia, págs. 112-116.

Shete, DS, SB Patil y S Patil (2014). «Zero crossing rate and Energy of the Speech Signal of Devanagari Script». En: *IOSR-JVSP* 4.1, págs. 1-5.

Tian, Mi y col. (2015). «On the use of the tempogram to describe audio content and its application to music structural segmentation». En: *2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, págs. 419-423.

Torrey, Lisa y Jude Shavlik (2010). «Transfer learning». En: *Handbook of Research on Machine Learning Applications and Trends: Algorithms, Methods, and Techniques*. IGI Global, págs. 242-264.

Van Der Linden, Janet y col. (2011). «Musicjacket—combining motion capture and vibrotactile feedback to teach violin bowing». En: *IEEE Transactions on Instrumentation and Measurement* 60.1, págs. 104-113.

Van Son, RJJH y col. (1994). «A method to quantify the error distribution in confusion matrices». En: *Proceedings of the Institute of Phonetic Sciences of the University of Amsterdam*. Vol. 18, págs. 41-63.

Villa-Canas, T y col. (2012). «Automatic detection of laryngeal pathologies using cepstral analysis in Mel and Bark scales». En: *Image, Signal Processing, and Artificial Vision (STSIVA), 2012 XVII Symposium of*. IEEE, págs. 116-121.

Widmer, Gerhard (2000). «Learning about musical expression via machine learning: A status report». En: *17th National Conference on Artificial Intelligence*. The Journal of the Acoustical Society of America.

Xu, Min y col. (2004). «HMM-based audio keyword generation». En: *Pacific-Rim Conference on Multimedia*. Springer, págs. 566-574.

Young, Diana (2008). «Classification of Common Violin Bowing Techniques Using Gesture Data from a Playable Measurement System.» En: *NIME*. Citeseer, págs. 44-48.

— (2002). «The hyperbow controller: Real-time dynamics measurement of violin performance». En: *Proceedings of the 2002*

conference on New interfaces for musical expression. National University of Singapore, págs. 1-6.

Young, Diana (2003). «Wireless sensor system for measurement of violin bowing parameters». En: *Stockholm Music Acoustics Conference*. Citeseer, págs. 111-114.

Young, Diana y Stefania Serafin (2007). «Investigating the Performance of a Violin Physical Model: Recent Real Player studies.» En: *ICMC*. Citeseer.

Zelle, John M (2004). *Python programming: an introduction to computer science*. Franklin, Beedle & Associates, Inc.

Zheng, Fang, Guoliang Zhang y Zhanjiang Song (2001). «Comparison of different implementations of MFCC». En: *Journal of Computer science and Technology* 16.6, págs. 582-589.

Zouhal, Lalla Meriem y Thierry Denoeux (1998). «An evidence-theoretic k-NN rule with parameter optimization». En: *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)* 28.2, págs. 263-271.