UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

Programa de Maestría y Doctorado en Música

Facultad de Música
Instituto de Ciencias Aplicadas y Tecnología
Instituto de Investigaciones Antropológicas

ASCIML

UN PROTOTIPO DE APRENDIZAJE AUTOMÁTICO INTERACTIVO PARA LA SÍNTESIS DE SEÑALES

TESIS
QUE, PARA OPTAR POR EL GRADO DE

MAESTRO EN MÚSICA (TECNOLOGÍA MUSICAL)

PRESENTA

GERARDO ANDRÉS MEZA GÓMEZ

TUTOR

DR. JORGE RODRIGO SIGAL SEFCHOVICH (ESCUELA NACIONAL DE ESTUDIOS SUPERIORES,UNAM)

CIUDAD DE MÉXICO. AGOSTO 2023

Declaro conocer el Código de Ética de la Universidad Nacional Autónoma de México, plasmado en la Legislación Universitaria. Con base en las definiciones de integridad y honestidad ahí especificadas, aseguro mediante mi firma al calce que el presente trabajo es original y enteramente de mi autoría. Todas las citas de obras elaboradas por otros autores, o sus referencias, aparecen aquí debida y adecuadamente señaladas, así como acreditadas mediante las convenciones editoriales correspondientes.

A mis padres, Mariana y Mei

Agradecimientos

Agradezco de manera sincera a todas las personas que han hecho posible la realización de este trabajo. En primer lugar, quiero agradecer a mi director de tesis, el Dr. Rodrigo Sigal, por su orientación, paciencia y apoyo en todo momento. Su conocimiento y experiencia fueron fundamentales para llevar a cabo este proyecto.

Asimismo, me gustaría agradecer a Jorge David e Irán Román por constantemente motivarme y orientarme, dentro y fuera de clase, a pesar de la pandemia y la distancia.

También quiero agradecer a mis compañeros y amigos del seminario de investigación, Hegel, Miguel, Mariana y Enrique, por las pláticas, discusiones, cartas y más que nada, por su tiempo y comentarios contantes sobre este trabajo.

A mi sínodo, el Mtro. Luis Pastor, el Dr. Francisco Colasanto, el Dr. Edmar Soria y el Dr. Cristian Bañuelos por su tiempo y comentarios invaluables a este trabajo.

Además, quiero agradecer a la UNAM, ENES y el CMMAS, por brindarme las oportunidades, espacios y recursos necesarios para realizar este trabajo de investigación.

Por último, quiero agradecer a mi compañera de vida Mariana Sepúlveda, con quién compartí cada minuto, emoción y logros de este proceso.

Índice general

Índice de figuras

Índice de tablas

Introducción

El presente trabajo de investigación surge de una búsqueda artística personal en la que tecnología e usuarios operan al mismo nivel, es decir, en la que los dispositivos, tecnologías y usuarios humanos se involucran de la misma manera en la creación musical. Esta exploración comenzó con la creación de un par de obras electrónicas y mixtas que exploran los procesos de memoria «algorítmica». De manera específica, esta idea se trabajó en las piezas *LCSE* [1] y Transversable [2]. Con el tiempo esta búsqueda artística evolucionó hacia la implementación de algoritmos de inteligencia artificial (IA) para la generación de materiales sonoros. Este proceso fue largo y altamente exigente desde el punto de vista técnico, tardando meses en lograr escuchar algún tipo de resultado sonoro. A pesar de haber comenzado desde la experiencia personal, opiniones similares han surgido en los distintos espacios académicos y artísticos en los que se incrusta esta investigación. Por lo cual, este trabajo de maestría busca tratar de reducir la brecha entre los usuarios y los modelos de aprendizaje automático en la música, con la finalidad de acelerar la manipulación y el proceso de generación de materiales sonoros con aprendizaje automático.

Desde 2003, han surgido proyectos de investigación que buscan acortar esta curva de aprendizaje entre los usuarios y los modelos de aprendizaje automático de manera interactiva [3, 4, 5]. En particular, en disciplinas creativas como la música, estos proyectos han dado resultados muy interesantes [6, 7, 8]. En este mismo sentido, la presente investigación busca vincular el aprendizaje automático interactivo con un modelo generativo para la síntesis y generación de materiales sonoros, con el objetivo de crear el prototipo *ASCIML* (Assistant

for Sound Creation with Interactive Machine Learning). El aporte de esta investigación radica en la creación de una herramienta que permite a los usuarios dirigir el entrenamiento de algoritmos generativos con conjuntos de datos personalizados, evaluar el proceso y, posteriormente, generar nuevas fuentes sonoras a partir de la interpolación entre sonidos.

Una de las preguntas centrales que guían esta investigación es si es posible generar modelos que brinden resultados sonoros interesantes desde una perspectiva creativa de manera rápida y si existen estrategias que puedan ser implementadas para agilizar este proceso en el aprendizaje automático interactivo. Otro aspecto central de este trabajo es encontrar estrategias que permitan generar conjuntos de datos de manera sencilla, que en este caso implica la generación de señales. Por último, se busca explorar el aspecto generativo del sistema y cómo el músico busca crear nuevos sonidos.

Para evaluar lo anterior, se realizó una revisión del estado del arte del aprendizaje automático interactivo y su aplicación en casos musicales, así como una revisión similar de los modelos generativos aplicados a la generación de señales. Además, se programó un prototipo en *Google Colab* que conjunta lo encontrado en la revisión del estado del arte. Dentro de este prototipo, se articulan distintas tecnologías, como algoritmos para el diseño de conjuntos de datos personalizados, un autocodificador de variación y funciones para la generación de materiales sonoros desde la representación latente de los modelos, '*upsampling*' y funciones para la interacción con el usuario, tales como la interfaz gráfica de usuario (GUI), visualizaciones y audio. Como último paso, se realizaron tres laboratorios acompañados de una encuesta y se interpretaron los datos obtenidos de los ejercicios anteriores.

El documento se encuentra organizado en tres capítulos, en el primer capítulo se plantea un panorama del estado del arte del aprendizaje de máquina interactivo, partiendo de su planteamiento en el 2003 [3] hasta las últimas implementaciones en el campo musical. Asimismo, se discuten distintos modelos generativos empleados para el procesamiento de señal y cómo articular este proceso en modelos que aprendan directamente formas de ondas. En el segundo capítulo se discuten los aspectos teóricos y de implementación de las

distintas partes de la herramienta desde el modelo, la generación interactiva de bases de datos, la generación de nuevos materiales a través de la exploración del espacio latente y por último, la estructura y las partes gráficas e interactivas de la herramienta *ASCIML*. Finalmente, se habla del primer prototipo del sistema, haciendo énfasis en la parte gráfica y controles que permiten al usuario interactuar con el sistema.

En el tercer capítulo se habla de los laboratorios realizados con la herramienta con miembros de la comunidad UNAM y los resultados obtenidos de dicho ejercicio. Finalmente, en el apartado de conclusiones se discute el futuro de la herramienta.

Para concluir esta introducción es necesario advertir al lector que a lo largo del documento se encontraran varios anglicismos cuando se habla de la parte técnica de la herramienta. Se optó por esta opción por la gran aceptación de los términos y utilización en distintos blogs, artículos, videos y libros. Asimismo, se le informa al lector que todos los materiales audiovisuales del proyecto se encuentran disponibles en `https://asciml.github.io/ASCIML/public/tesis.html`.

Capítulo 1

Aprendizaje de máquina interactivo y modelos generativos

El interés principal de esta investigación es la creación de una herramienta que permita a los compositores/creadores sonoros realizar síntesis con redes neuronales de manera interactiva sin la necesidad de escribir código o conocer a profundidad los conceptos de aprendizaje automático. Para esto, se creó un primer prototipo que vincula la síntesis de sonido, los algoritmos generativos y el aprendizaje de máquina interactivo. El primer paso contempló una revisión de la literatura y avances recientes. A partir de esta exploración se realizó la programación de la herramienta y finalmente una serie de ejercicios con músicos en formación.

En este primer capítulo se parte de establecer qué es interacción y bajo qué interpretación se inserta en esta investigación, seguido de una breve descripción de cómo operan los algoritmos de aprendizaje automático comúnmente y el problema que esto puede llegar a implicar para la creación musical. Posteriormente se realiza una revisión cronológica del aprendizaje automático interactivo, desde su inserción hasta implementaciones actuales en la música, con la finalidad de identificar las características de este paradigma de manera general y cómo se articula particularmente en la tecnología musical. Tras esta revisión se

discuten los retos del aprendizaje de máquina interactivo (IML, por sus siglas en inglés) al trabajar con señales de audio y la propuesta teórica articulada en la presente investigación. Finalmente, se realiza una revisión de proyectos de investigación y modelos actuales generativos que realizan síntesis neuronal. Este capítulo establece las bases teóricas sobre las cuales se construye el prototipo *ASCIML*.

1.1. Aprendizaje de máquina interactivo

El término interacción puede ser definido como la «acción que se ejerce recíprocamente entre dos o más objetos, personas, agentes, fuerzas, funciones, etc»[9]. Por otro lado, el término interactividad, como objeto de estudio de las ciencias de la comunicación, puede ser definido como:

> El grado en que una tecnología de comunicación puede crear un entorno mediado en el que los participantes pueden comunicarse (uno a uno, uno a muchos y muchos a muchos), tanto sincrónica como asincrónicamente, y participar en intercambios recíprocos de mensajes [...]. Con respecto a los usuarios humanos, también se refiere a su capacidad para percibir la experiencia como una simulación de comunicación interpersonal y aumentar su conciencia de telepresencia [10].

En el contexto de la música, la interacción puede referirse a la forma en que los músicos o los oyentes interactúan entre sí durante una presentación o una experiencia musical. Esto puede incluir la interacción entre músicos en una banda u orquesta, la interacción entre un músico y su audiencia, o la interacción entre los oyentes y el ambiente musical en general.

En el contexto de la tecnología musical, la interacción puede referirse a la forma en que los usuarios se relacionan con aplicaciones, juegos o dispositivos musicales, como instrumentos electrónicos o aplicaciones de producción musical [11, 12, 13, 14]. Esto incluye la capacidad de los usuarios para crear y modificar la música a través de la interacción con la tecnología.

La presente investigación se enmarca en la aproximación tecno-musical que surge entre modelos de aprendizaje automático y el usuario/artista, así como en la definición planteada por [10].

El paradigma usual al trabajar con algoritmos de aprendizaje automático (CML por sus siglas en inglés), consiste en trabajar con un conjunto de ejemplos, escoger previamente las características y permitir que el modelo se optimice a través de periodos largos de entrenamiento. Bajo esta idea, el programador asume que el entrenamiento del modelo será necesario una sola vez, por lo cual se sacrifica tiempo, entrenando por días, semanas o meses. Una vez entrenado, este tipo de algoritmos pueden ser utilizados en tiempo real para realizar rápidamente la predicción o tarea [3, 5]. Este tipo de modelos suelen ser evaluados por su poder de generalización a partir de circunstancias particulares, por ejemplo, en el caso de los clasificadores quiere decir qué tan bien efectúa su tarea con nuevos datos a partir de los datos de entrenamiento. Para que un algoritmo pueda realizar esto, es necesario un análisis detallado de combinaciones de características por parte del usuario y periodos largos (semanas o meses) de procesamiento.

Particularmente esto representa un problema en las herramientas desarrolladas para la música, ya que estas contienen modelos entrenados para resolver una cantidad específica de tareas y los modelos previamente entrenados no puede ser reentrenados por los usuarios para realizar tareas fuera del dominio de los conjuntos de datos o características extraídas, lo cual lleva a los usuarios más entusiastas de estas tecnologías a emplear varias herramientas para realizar tareas muy similares.

En el 2003, Fails y Olsen proponen el IML con su sistema *Crayons* [3], en el cual el usuario puede corregir y enseñar al clasificador a escoger las características correctas, integrando la extracción de las mismas al proceso de aprendizaje del modelo. Esto se logra alimentando al modelo con una lista grande de posibles características y dejar al clasificador filtrarlas. Los autores proponen realizar una dupla entre un humano y un modelo capaz de entrenar rápidamente con la finalidad de optimizarse de manera iterativa a través de cientos o miles de entrenamientos. Para lograr esta optimización entre el humano y la máquina en el IML,

el modelo debe ser capaz de aprender de manera rápida, lo cual implica la utilización de un modelo compacto para poder explorar y seleccionar múltiples características.

Después de *Crayons*, una de las primeras implementaciones realizadas del IML fue el *Wekinator* de Rebecca Fiebrinck [6] (figura 1.1), el cual permite a usuarios crear y modificar mapeos de controladores con parámetros sonoros [1]. La arquitectura del *Wekinator* integra dos módulos en distintos lenguajes de programación comunicándose por protocolo OSC. Por un lado, el lenguaje Chuck se encarga de la extracción de características y la síntesis, mientras Java maneja los algoritmos de aprendizaje automático y la interfaz gráfica de usuario (GUI). El *Wekinator* permite realizar la creación de los datos de entrenamiento en tiempo real por el usuario, modificar el proceso de aprendizaje, el conjunto de entrenamiento y el re-entrenamiento en segundos, todo integrado en un solo GUI.

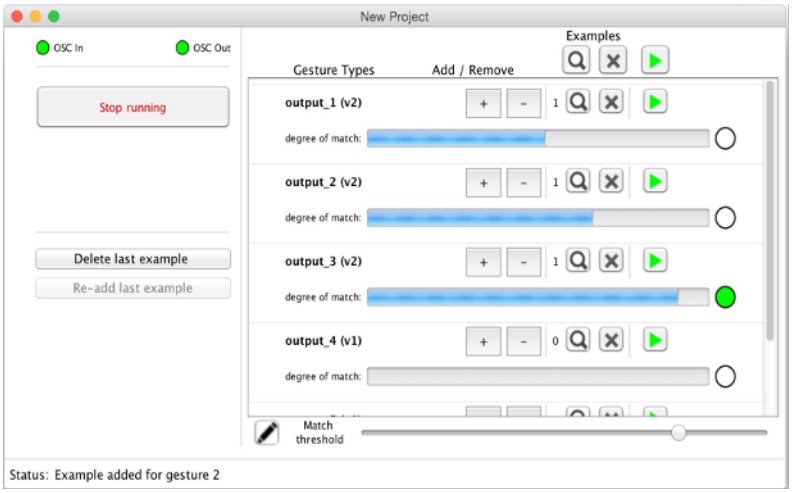


Figura 1.1: Interface principal de *Wekinator*.
Fuente: Tomado de [15].

Por su parte, en el 2011, Saleema Amershi presentó el clasificador *CueFlik* introduciendo el active learning al IML [4], esto permite ilustrar al usuario el entendimiento que tiene el modelo a través del proceso de optimización iterativo. Para esto, la interfaz gráfica debe ser capaz de ilustrar efectivamente la versión actual de un concepto aprendido y guiar a los usuarios para seleccionar ejemplos de entrenamiento que optimicen el proceso de

[1] Aparte de utilizado para el mapeo y la creación de instrumentos musicales, el Wekinator puede ser utilizado para crear sistemas de visión y escucha de máquina.

aprendizaje.

En el 2014, Amershi y sus colaboradores en su artículo *Power to the people: The role of humans in interactive machine learning* reflexionan sobre la relación que existe entre los usuarios de aprendizaje automático y los desarrolladores de estos algoritmos. En él mencionan que a pesar de que los usuarios sean expertos en las áreas en que se quiere emplear estos algoritmos, cuando son tomados en cuenta para el desarrollo de estas herramientas, su papel se ve siempre limitado en proveer datos, contestar preguntas que los desarrolladores tengan sobre el campo de conocimiento en el que se va a insertar el modelo y dar retroalimentación sobre los resultados del modelo [5]. En este sentido, el paradigma que ofrece IML «puede facilitar la democratización del aprendizaje automático aplicado, capacitando a los usuarios finales para crear sistemas basados en aprendizaje automático para sus propias necesidades y propósitos»[5][2].

Tras realizar la revisión de varios casos en otras disciplinas, incluyendo algunos de los ya mencionados, los autores concluyen que el grado de interactividad necesaria que debe proveer una interfaz que emplea IML debe ser probada con usuarios potenciales, ya que esto permite descubrir nuevas formas de interacción, encontrar distintos escenarios y destapar obstáculos que los usuarios encuentren en estos.

Del lado del desarrollo web, en el 2020 Louis McCallum y Mick Grierson crearon dos librerías de *JavaScript* que permiten realizar las tareas comunes de IML y sintetizar audio [7]. Esto con la finalidad de auxiliar a músicos en la creación de instrumentos musicales que empleen aprendizaje automático en la red. Los autores subrayan que el flujo de trabajo de esta herramienta consiste en escoger un dispositivo de entrada, los parámetros a los que serán mapeados, ejemplos de entrenamiento, uso del mismo e interacción.

En el 2021, la arquitectura de la herramienta *GIMLeT* [16] implementó el paradigma de IML en la musica en conjunto con un asistente de aprendizaje por refuerzo para la co-exploracion artística. En particular, esta arquitectura implementa lo anterior para es-

[2] «can facilitate the democratization of applied machine learning, empowering end users to create machine-learning-based systems for their own needs and purposes»(Esta y todas las traducciones del documento fueron realizadas por el autor)

tablecer y explorar relaciones entre el movimiento de un ejecutante capturado por un dispositivo y su asignacion con parámetros de un sintetizador concatenativo en Max. Para lograr esta tarea, se entrena una red neuronal que implementa un sistema de regresión lineal con el conjunto de datos obtenidos para obtener los valores del sintetizador. Una vez entrenado el modelo, el intérprete o usuario explora los mapeos resultantes. En caso de ser un resultado grato, el usuario puede guardar el mapeo o dar retroalimentación al agente por medio de una interfaz gráfica, lo cual lleva a un nuevo ciclo de exploración y generación de datos, entrenamiento y resultados.

Otros proyectos y organizaciones que han surgido en torno a este paradigma son el *German Research Center for Artificial Intelligence*[3], dónde se conducen investigaciones centradas en aprendizaje de máquina interactivo en distintas áreas del conocimiento, y el *teachable machines* de *Google*[4] , una herramienta web que permite crear modelos sin necesidad de código (figura 1.2), permitiendo al usuario entrenar a una computadora para que reconozca imágenes, sonidos y poses permitiendo la exportación del modelo para su uso en otros sitios o aplicaciones.

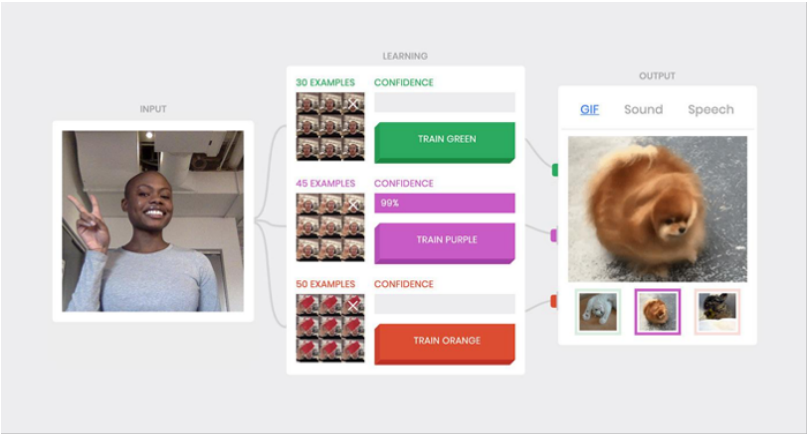


Figura 1.2: Interfaz principal de *teachable machines* de *Google*
Fuente: Tomado de [17].

[3] https://www.dfki.de/en/web/research/research-departments/interactive-machine-learning

[4] https://experiments.withgoogle.com/teachable-machine

A manera de resumen, el aprendizaje interactivo se caracteriza por: a) permitir a usuarios expertos en una determinada disciplina generar modelos de aprendizaje automático a través de la interacción con una interfaz gráfica que facilite la creación de datos de entrenamiento en tiempo real, b) modificar tanto el conjunto de datos como el proceso de aprendizaje y c) un modelo capaz de entrenar rápidamente y acoplarse con el usuario de manera iterativa para lograr optimizarse.

De manera particular en las implementaciones musicales antes mencionadas encontramos el uso del IML en la creación de mapeos complejos entre dispositivos de control con sintetizadores. Este proceso se lleva a cabo a través del entrenamiento de modelos con datos simbólicos (ej. valores MIDI). En estos modelos podemos encontrar cuatro etapas fundamentales: la etapa de diseño de conjunto de datos, donde el usuario provee todo tipo de ejemplos sonoros, gestuales etc; la etapa de entrenamiento, donde un modelo de inteligencia artificial procesa el conjunto de datos anterior para realizar una tarea determinada: clasificación, regresión, etc; la tercera etapa contempla la evaluación del modelo donde el músico o artista es capaz de evaluar los resultados obtenidos. Estas evaluaciones comúnmente son llevadas a cabo a partir de la escucha y experimentación con los resultados. Finalmente, viene la etapa de interpretación y creación donde el intérprete utiliza el dispositivo con fines creativos (figura 1.3)[5].

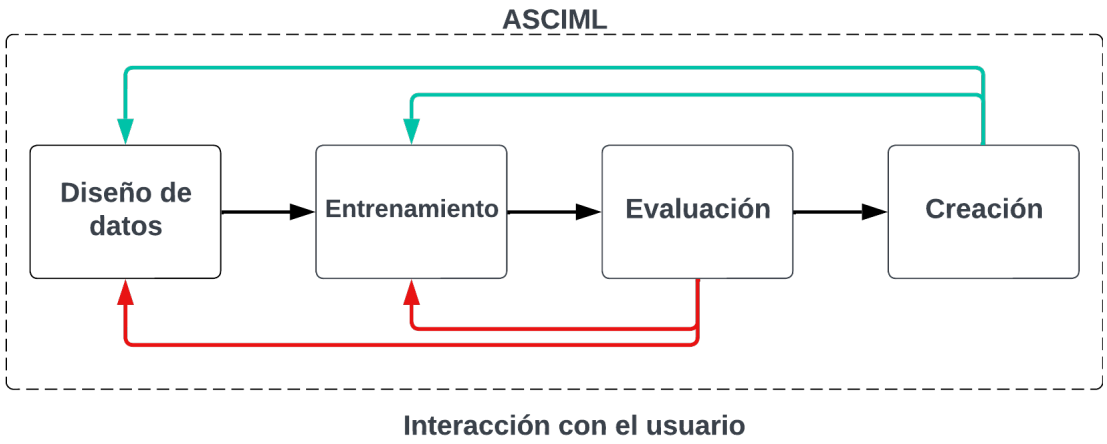


Figura 1.3: *Pipeline* de IML.

[5]Todas las imágenes del documento que no especifican la fuente fueron creadas por el autor.

1.2. Modelos generativos

El proceso de aprendizaje de un modelo generativo es diferente a los modelos discriminativos. Los modelos deterministas buscan identificar características de un conjunto de datos para hacer predicciones precisas, los modelos generativos aprenden patrones de datos y generan nueva información [18]. Por ejemplo, un modelo generativo entrenado con imágenes de animales puede generar imágenes de animales que podrían parecer reales, pero que en realidad no existen en el mundo real. Mientras un modelo determinista podría ser entrenado para identificar perros y gatos (figura 1.4).

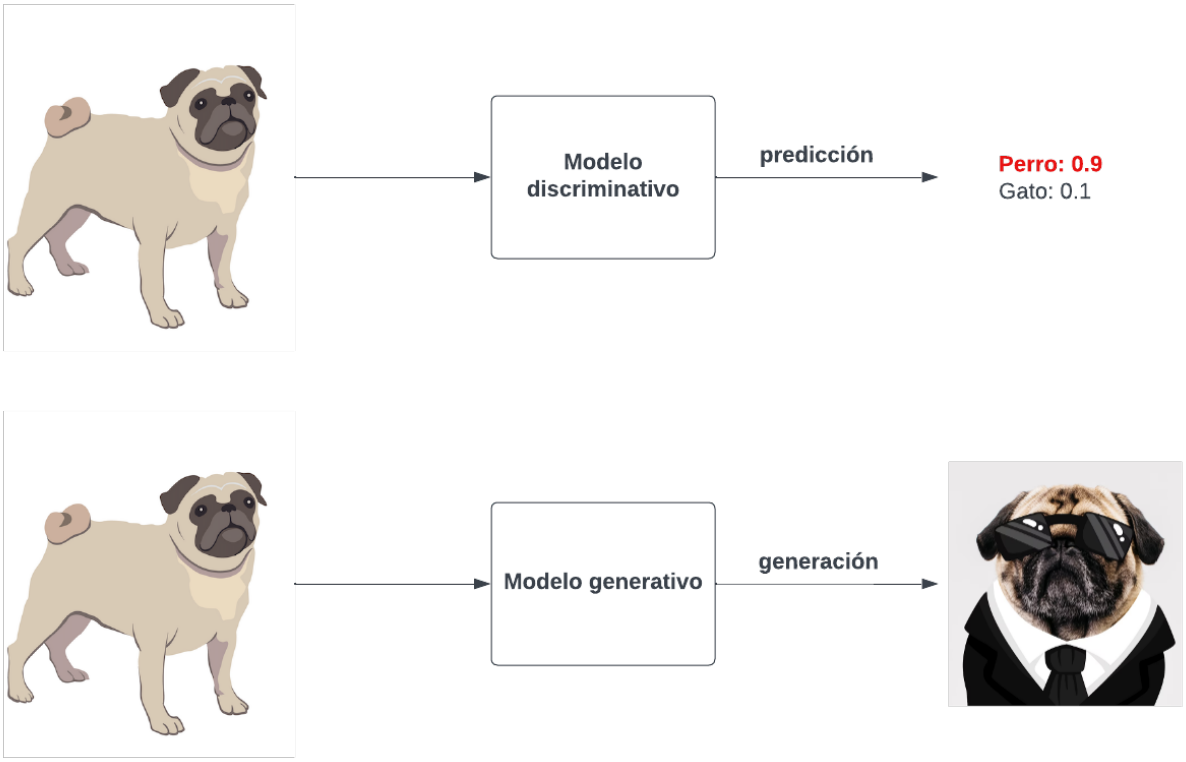


Figura 1.4: Generación vs discriminación.

De manera general, los modelos generativos aprenden distribuciones de probabilidad de datos y utilizan esa información para generar nuevos datos al tomar muestras aleatorias. La calidad de generación depende de la cantidad, la calidad de los datos de entrenamiento y la complejidad del modelo utilizado.

En los últimos años, dos arquitecturas se han erigido como los principales pilares para desarrollar modelos generativos: los autocodificadores de variación (VAEs, por sus siglas en inglés)[19] y las redes generativas antagónicas (GANs)[20]. Los VAEs se encargan de comprimir la entrada en una representación compacta, llamada representación latente, espacio latente o código para luego generar nuevos datos a partir de ella (figura 1.5).

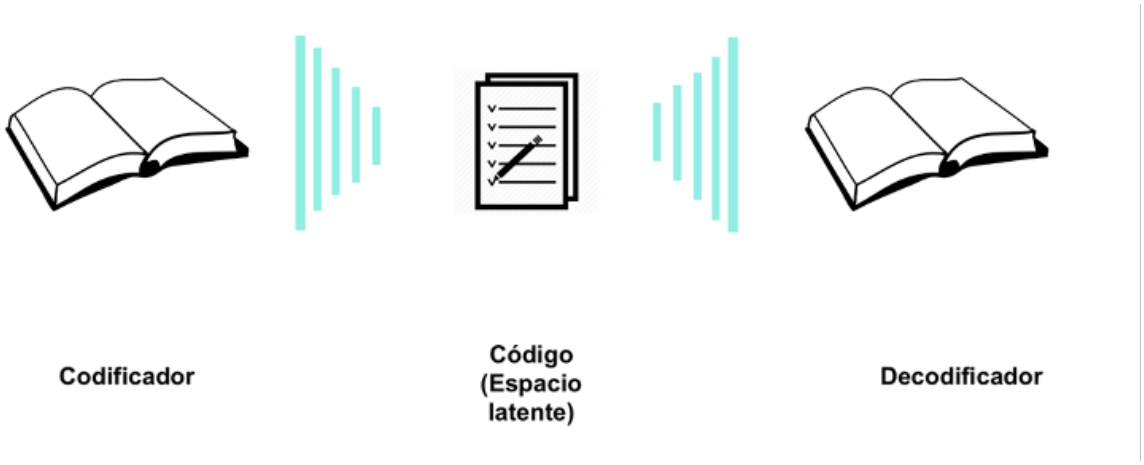


Figura 1.5: Autocodificador de variación.

En comparación, las GANs consisten en dos redes neuronales, el generador y el discriminador, que compiten entre sí para generar datos realistas, mientras la primera se optimiza para burlar al clasificador, la segunda se optimiza para realizar mejores predicciones (figura 1.6).

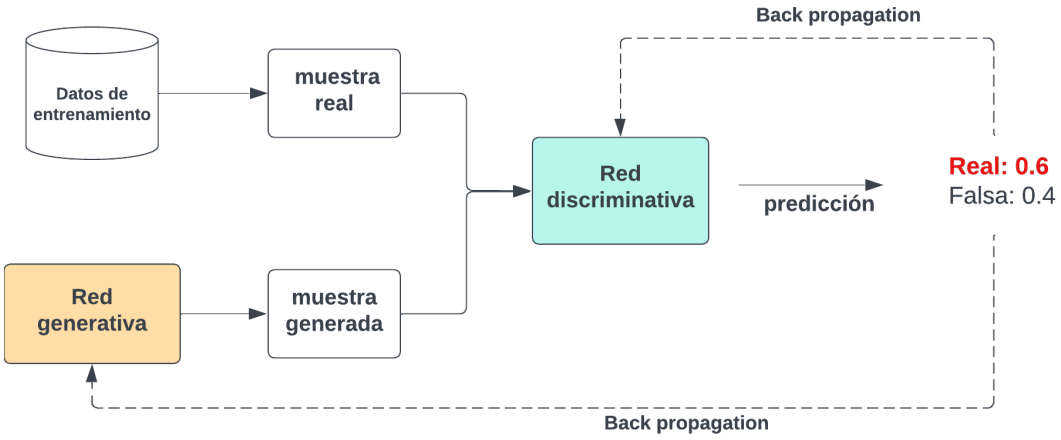


Figura 1.6: Arquitectura de un GAN.

Es importante decir que existen múltiples modelos generativos que se insertan en el área de la música y el sonido [21, 22, 23, 24, 25]. El siguiente apartado se concentra en revisar algunos modelos de aprendizaje profundo que trabajan específicamente con la manipulación de la señal y que influyeron directamente el desarrollo de *ASCIML*.

1.2.1. Modelos generativos de señales

En el artículo *WaveNet: A Generative Model for Raw Audio* [26], se presenta el *WaveNet*, un modelo generativo de audio basado en la arquitectura *Pixel CNN* [27]. Este modelo entrena directamente sobre la forma de onda de la señal de audio y predice sus muestras basándose en las muestras previas generadas[6] (figura 1.7). Los autores del artículo realizaron una serie de experimentos para evaluar el alcance del modelo, incluyendo la generación de palabras con tonos de voz, síntesis de voz (TTS), y generación musical.

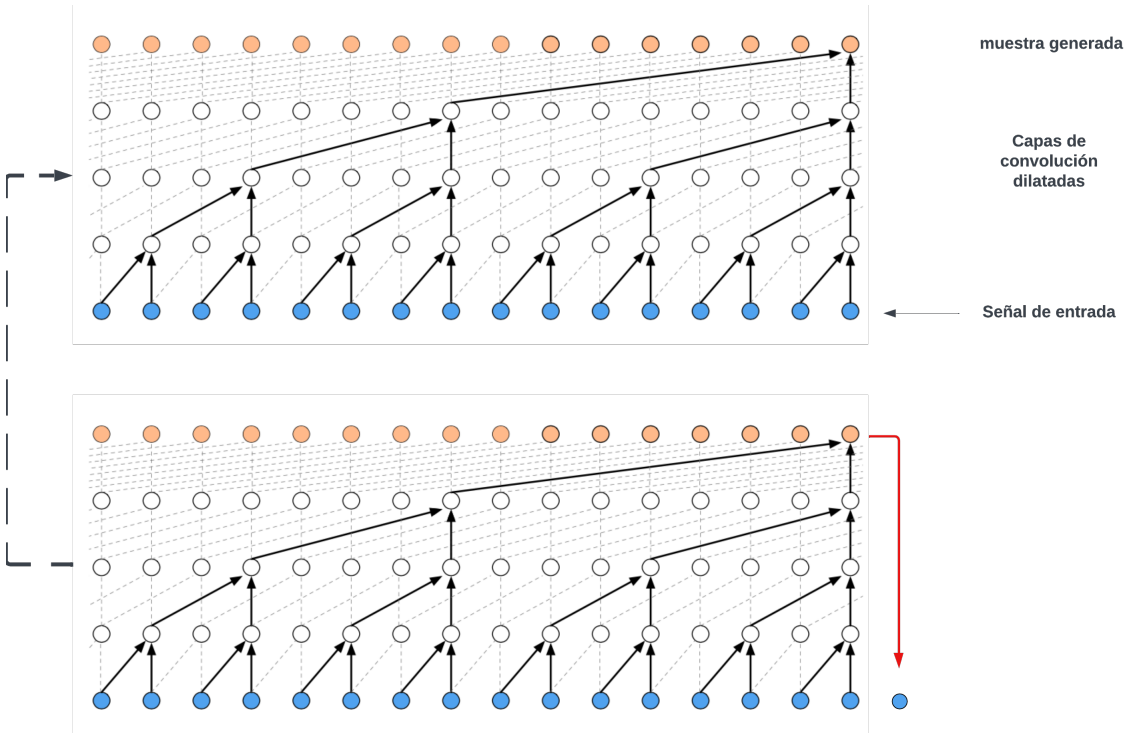


Figura 1.7: Arquitectura del *WaveNet*.
Fuente: Adaptado de [26].

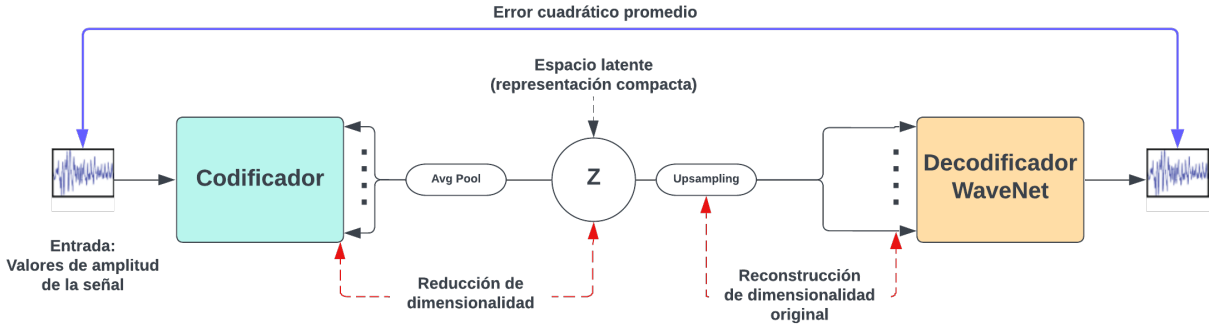
[6]Para mayor información consultar `https://www.deepmind.com/blog/wavenet-a-generative-model-for-raw-audio`

Para realizar experimentos de generación de palabras se entrenó el modelo para generar palabras «falsas» [7] con una entonación realista. Para esto se utilizó un conjunto de datos de 44 horas con 109 hablantes. El modelo fue capaz de aprender la entonación realista, cualidades acústicas del espacio, la calidad de grabación y sonidos asociados a la emisión del habla. Algunas de las observaciones de este experimento es que solo existía coherencia entre fonemas cercanos (2 o 3 fonemas). Este estudio fue ampliado al condicionar el modelo a características lingüísticas y a la fundamental (f_0) de los sonidos con voces femeninas en dos distintos idiomas, mediante la comparación con otros algoritmos y el uso de la métrica *Mean Opinion Score* (MOS), se midió la naturalidad de la voz sintetizada.

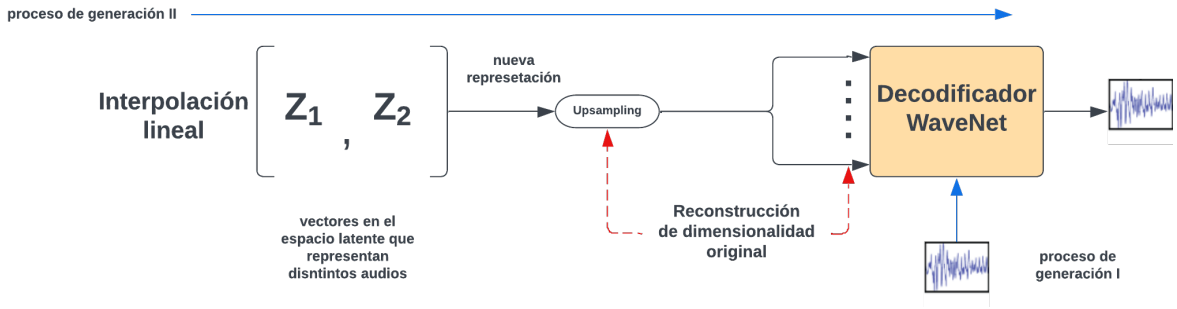
Para el estudio de generación musical, los autores realizaron un análisis aural de los resultados del modelo. Debido a los resultados del primer estudio, los autores ampliaron el tamaño de señal que el modelo puede crear para generar pasajes más coherentes musicalmente. Sin embargo, el modelo no fue capaz de generar grandes pasajes musicales y los resultados que arrojaba la red llegaban a variar de género, instrumentación, volumen y calidad sonora. A pesar de esto, los autores consideran que los pasajes producidos eran placenteros y armónicos.

En el 2017 surge el *NSynth* [28], un modelo basado en el *WaveNet*. Aunque está basado en esta arquitectura, el *NSynth* cuenta un codificador y un decodificador. Durante el entrenamiento, el codificador toma la forma de onda como entrada para producción de una representación con menos valores de la misma (espacio latente). Estas representaciones son alimentadas al decodificador *WaveNet*, para posteriormente calcular el error cuadrático promedio entre la entrada y la salida (figura 1.8a). Durante la inferencia, el decodificador genera autorregresivamente, de manera similar al WaveNet, una única muestra de salida a la vez de la forma de onda. Para generar nuevos audios se puede inferir de forma determinista a partir de los audios de entrenamiento o extraerse de otros puntos en el espacio latente, por ejemplo, al realizar por interpolación entre dos muestras (figura 2.12e).

[7] Por palabras «falsas» se hace referencia a fonemas sin tener sentido gramatical necesariamente.



(a) Arquitectura del *NSynth*.



(b) Proceso de generación de audios.

Figura 1.8: *NSynth*.

Para evaluar la reconstrucción de nota, la interpolación instrumental y la interpolación de altura de este modelo, se utilizó el conjunto de datos con el mismo nombre que consistía en 306,043 notas musicales con distintas alturas, timbres y envolventes.

Al igual que el *WaveNet*, en la reconstrucción de la señal del *NSynth* existe una distorsión ocasionada por la compresión de la ley de mu, que resulta muy pronunciada en las frecuencias graves. Al realizar interpolaciones lineales en el espacio latente, los autores encontraron que los modelos generativos fusionan aspectos tímbricos creando nuevas estructuras armónicas (figura 1.9).

El *RAVE* [29] es un autocodificador de variación de audio en tiempo real creado en el 2021 para el aprendizaje de señales de alta calidad. Este modelo es el más compacto de los ya mencionados y tiene la facultad de generar señales de audio de 48 kHz. Los autores con *RAVE* proponen un procedimiento de entrenamiento de dos etapas. En la primera

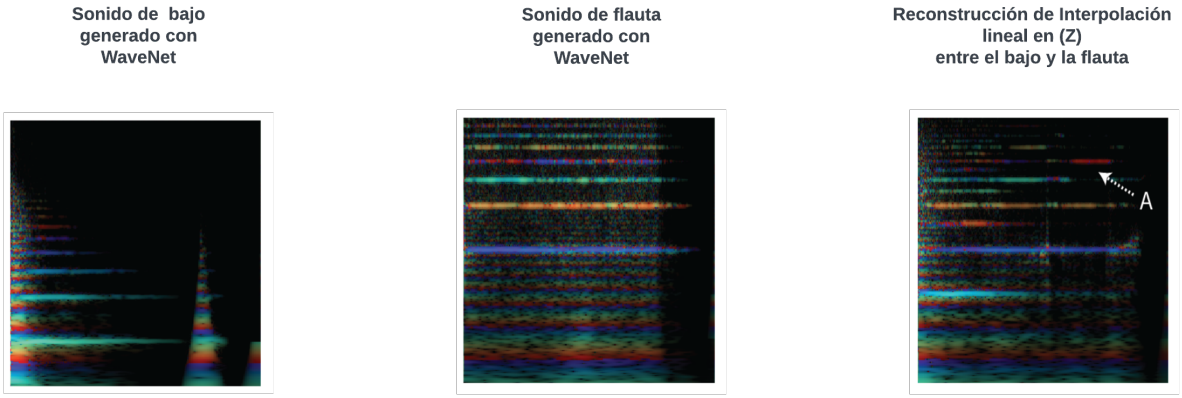


Figura 1.9: Ejemplos de generación por interpolación de *NSynth*.
Nota: El modelo va más allá de la suma de armónicos en la (A) se señala una modificación expresiva sobre el armónico. Fuente: Adaptado de [28]

etapa se realiza la descomposición multibanda de las señales [8] y con estas se entrena el codificador y decodificador utilizando la distancia espectral multibanda para ajustar los hiperparámetros de la red. En la segunda etapa, se utiliza un GAN para comparar las señales que genera el modelo respecto a las originales y poder generar señales de mayor calidad (figura 1.10). Esto le permite al modelo realizar una síntesis 20 veces más rápida en una CPU de computadora portátil estándar.

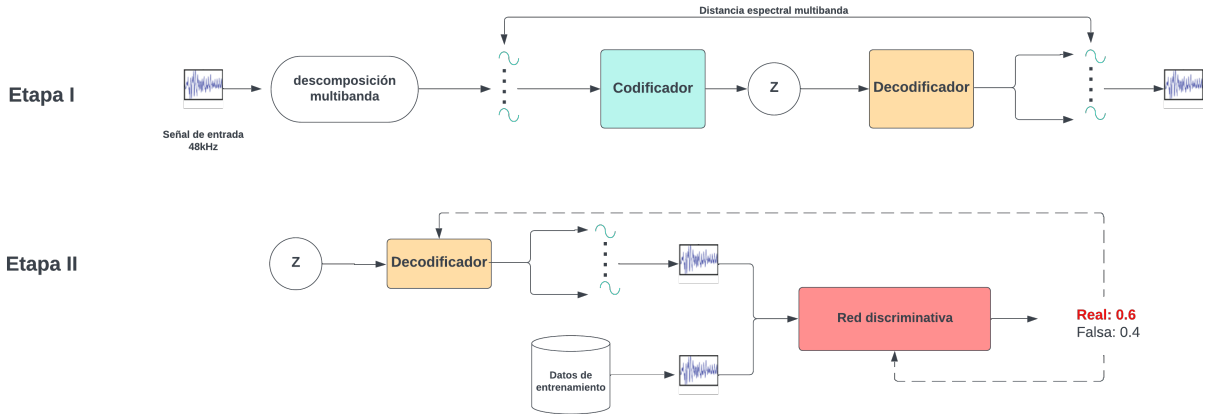


Figura 1.10: *RAVE*.

[8] El uso principal de una descomposición multibanda es representar una señal de audio con una frecuencia de muestreo determinada (p. ej., 48 kHz) como una combinación de varias subseñales submuestreadas (p. ej., 3 kHz), donde cada subseñal cubre un rango particular de frecuencias. Esta descomposición es útil para fines de compresión, como la codificación/descodificación de mp3, ya que permite el uso de diferentes tasas de bits en áreas específicas del espectro [29].

A manera de resumen, en este capítulo se realizó una revisión del IML y sus componentes al ser aplicados en la música. Asimismo, se presentó de manera general cómo funcionan los modelos generativos, cuáles son los más comunes y algunos modelos de generación de señal que influenciaron el desarrollo de este trabajo. A través de los próximos capítulos, la presente investigación buscará tomar el paradigma de IML y adaptarlo a la creación de sonido con modelos generativos. Recordemos que las señales de audio necesitan miles de valores de amplitud para ser representadas a diferencia de las imágenes y datos simbólicos (como el MIDI). Por lo cual las incógnitas centrales que guían esta investigación son: ¿hay la posibilidad de generar modelos que brinden resultados sonoros interesantes desde una perspectiva creativa de manera rápida?, ¿existen estrategias que puedan ser implementadas para agilizar este proceso en el IML en este paradigma generativo? Otro aspecto central de este trabajo es encontrar qué estrategias se pueden seguir para la generación de conjuntos de datos de manera ágil y práctica, que en este caso implica generar señales. Finalmente, se busca explorar el aspecto generativo del sistema y cómo el músico interactúa con el mismo para buscar la creación de nuevos sonidos.

Capítulo 2

ASCIML

En este capítulo se describen las partes que conforman la herramienta *ASCIML*. En un principio se realiza una descripción general de los autocodificadores de variación, la función de error que se utiliza y en qué consiste su proceso de optimización. En seguida, se detalla el modelo desarrollado dentro de esta investigación, el cual utiliza las librerías de *Keras* y *TensorFlow* en su programación para ser entrenado con señales de duración de 1 segundo a 16 kHz y utilizar los valores de amplitud de las distintas señales. Después, se aborda la problemática que envuelve la generación de conjuntos de datos dentro del ML para su utilización en la creación y cómo el IML es una opción viable para el desarrollo de herramientas como *ASCIML* por su adaptabilidad. A continuación, se describen las fórmulas y arquitecturas de programación utilizadas y desarrolladas para generar interactivamente conjuntos de datos, realizar la exploración de código y procesos de *upsampling* necesarios para articular el prototipo. Finalmente, se describe la arquitectura general del *ASCIML* y como interactúan las partes anteriores para su funcionamiento. Este prototipo se encuentra disponible para su consulta y utilización en `https://colab.research.google.com/github/mezaga/ACIML-prototype/blob/main/ASCIML_prototype_lab.ipynb`.

2.1. Modelo

La presente investigación busca acoplar el IML con un modelo generativo, en este caso particular, un autocodificador de variación (VAE). Este tipo de red neuronal se compone de tres partes: un codificador, el código y un decodificador (figura 2.1) y a diferencia de un autocodificador, este tipo de modelo se regulariza a través del entrenamiento para evitar el sobre ajuste y garantizar que el espacio latente tenga las propiedades adecuadas que permitan el proceso generativo [30].

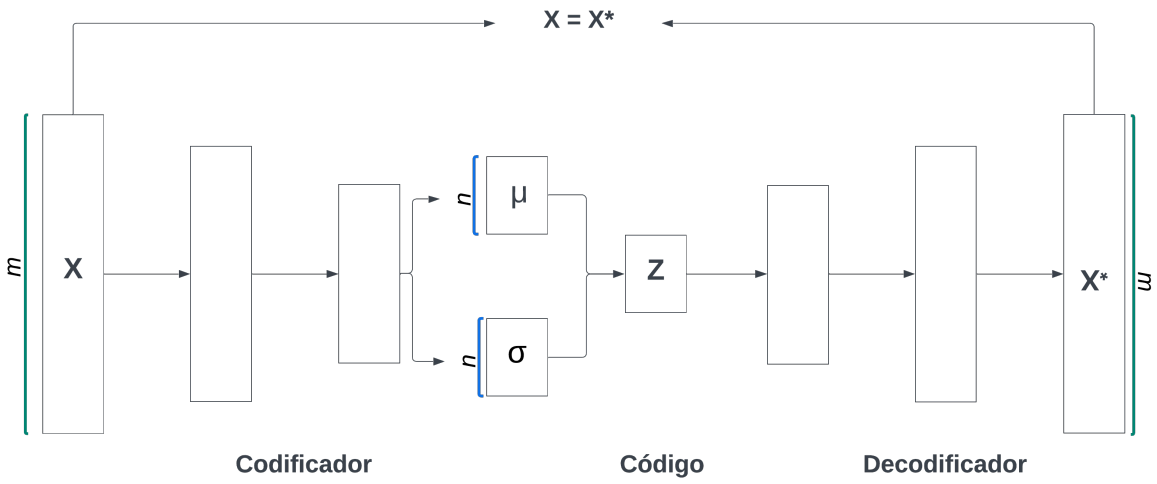


Figura 2.1: Autocodificador de variación y sus partes.

El proceso de entrenamiento de estas arquitecturas se lleva a cabo a través de las épocas de entrenamiento y consiste en minimizar el error de reconstrucción entre los datos decodificados (X^*) y los datos iniciales (X)(figura 2.1), [30] [31]. Durante este proceso, la parte del codificador se encarga de crear dos vectores (σ, μ) de tamaño $n(m \gg n)$, donde n es similar al tamaño de la entrada al código y m es la dimensión de la entrada al codificador (figura 2.1). En seguida, en el código se toman muestras aleatorias de estos dos vectores $z = \sigma \cdot \epsilon + \mu$, donde ϵ representa un tensor con distribución gaussiana. Esto se alimenta al decodificador para producir X^*.

Al llegar a la última capa del decodificador, es necesario calcular el error respecto a la entrada (X) y el resultado se utiliza para corregir la función de la parte del codificador. Esto se repite a través de las épocas de entrenamiento preestablecidas. Como recién se mencionó, tanto μ como σ del VAE se entrenan combinando dos funciones de pérdida: error de reconstrucción y la *divergencia Kullback-Leibler* [30](fórmula 2.1). Por un lado, la primera se encarga de comparar la salida del decodificador con la entrada al modelo y calcular el error en función de la diferencia entre ambos, obligando al decodificador a producir una salida similar a la entrada original ($X = X^*$). Por el otro lado, la divergencia de *Kullback-Leibler* funciona como término de regularización evitando que el modelo codifique los datos muy separados en el código, fomentando en la medida de lo posible que las distribuciones devueltas se «superpongan», satisfaciendo así las condiciones esperadas de continuidad y completitud [30](figura 2.2).

$$loss = ||x - x||^2 + KL[N(\mu, \sigma)] \tag{2.1}$$

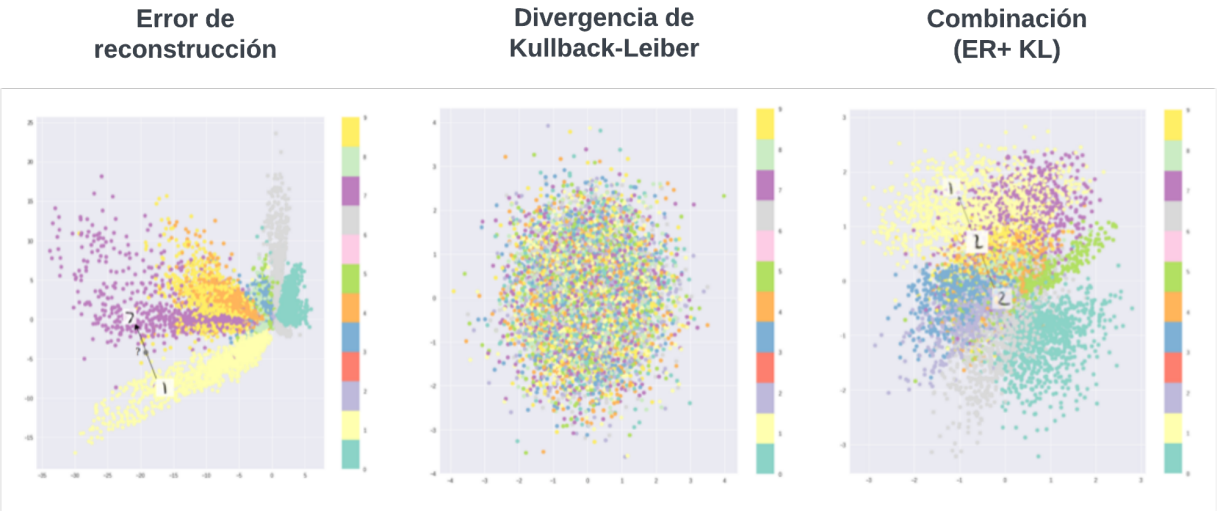


Figura 2.2: Visualización del código generado por un VAE entrenado con un conjunto de datos de números escritos a mano (MNIST dataset url: `https://www.kaggle.com/datasets/hojjatk/mnist-dataset`) utilizando error de reconstrucción(izquierda), divergencia de *Kullback-Leibler* (centro) y la combinación de ambas (derecha).

2.1.1. VAE de *ASCIML*

Una vez establecido lo anterior, la arquitectura propuesta en esta investigación utiliza señales de audio con una duración de un segundo y una frecuencia de muestreo de 16 kHz/ 24 bits como entrada. Como resultado, el vector X_n tiene una longitud de 16,000 valores, donde cada valor representa una muestra de la señal de audio. Este vector se somete al proceso de codificación, que reduce su dimensionalidad a 128 valores en el código (Z). Posteriormente, se reconstruye el vector original y se devuelve a su dimensionalidad inicial (figura 2.3).

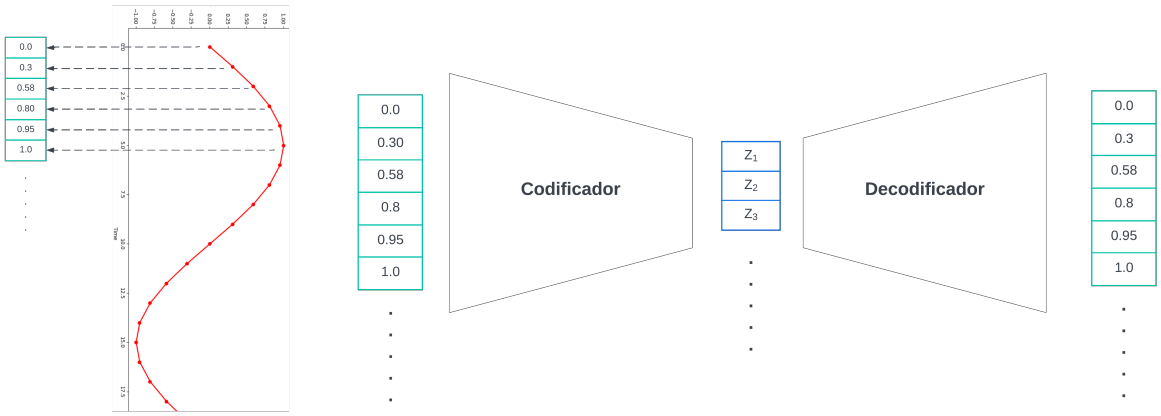


Figura 2.3: Visualización de X_n y el proceso de codificación.

De manera más específica, el codificador utilizado en el modelo consta de tres capas de convolución. Cada capa tiene un tamaño de kernel fijo de 66, 1 y utiliza la función de activación *ReLU (Rectified Linear Unit)* [1]. Las cantidades de filtros utilizadas son 64, 128 y 256, respectivamente. Se aplican *strides* de 4. Estos valores fueron utilizados con la intensión de ir reduciendo la dimensionalidad de la señal en un 25 % de su dimensionalidad anterior al salir de cada capa, mientras el número de filtros va en aumento.

Al pasar una señal de audio de dimensiones 16000, 1, 1 a través del codificador, las capas de convolución realizan transformaciones y extraen las características más relevantes de la señal. Es importante destacar que en este codificador no se utilizan capas de *Pooling*. En

[1]La función de activación ReLU es de las más utilizadas con capas de convolución y su valor es igual a 0 cuando el valor es negativo

cambio, las capas de convolución con tamaños de kernel y pasos mencionados anteriormente se encargan de reducir la dimensión de la señal. En este caso, la primera capa de convolución con 64 filtros y un tamaño de kernel de 66, 1 transforma la señal en una representación de tamaño 4000, 1, 64. A continuación, la segunda capa de convolución con 128 filtros mapea la señal a una representación de tamaño 1000, 1, 128. Posteriormente, la tercera capa de convolución con 256 filtros reduce aún más la dimensión de la señal a 250, 1, 256.

Después de las capas de convolución, se aplica una capa Flatten para aplanar la salida y se utilizan dos capas densas con 256 y 128 unidades, respectivamente. (figura 2.4) Estas capas densas reducen aún más la dimensionalidad de la señal. En resumen, la señal original con 16000 valores se transforma y se representa con solo 128 valores a través de las capas de convolución y las capas densas del codificador.

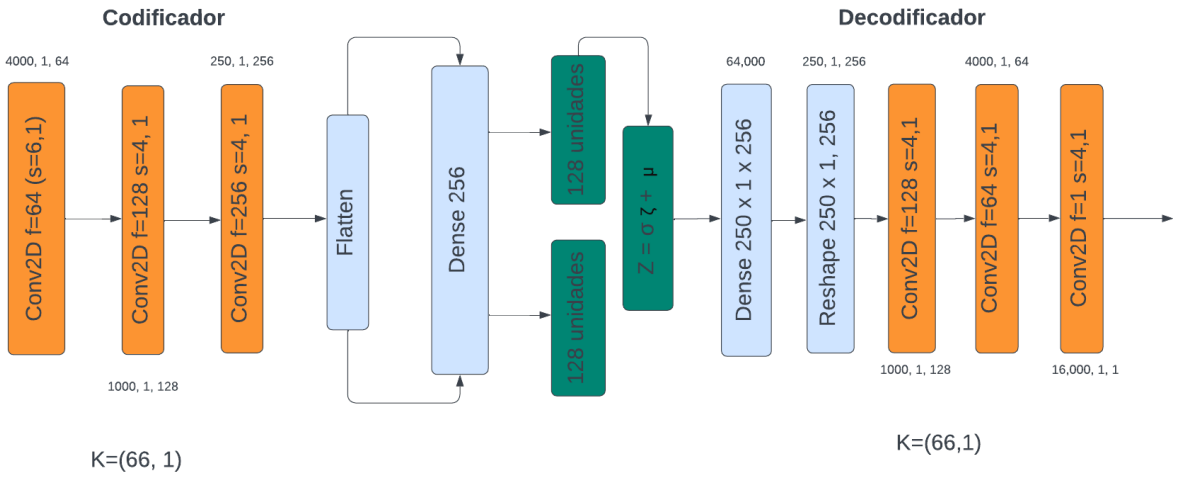


Figura 2.4: Autocodificador de variación de *ASCIML*.

El decodificador del modelo se compone de capas simétricas a las del codificador para que la señal reconstruida tenga las mismas dimensiones que la señal original de entrada, tal y como se muestra en la figura 2.1.

Una vez que el modelo ha sido entrenado, el usuario puede generar nuevos sonidos utilizando el código y el decodificador sin necesidad de entrenar el modelo nuevamente. Para hacerlo, puede samplear desde el espacio latente o realizar interpolación lineal entre vectores de Z (figura 2.5).

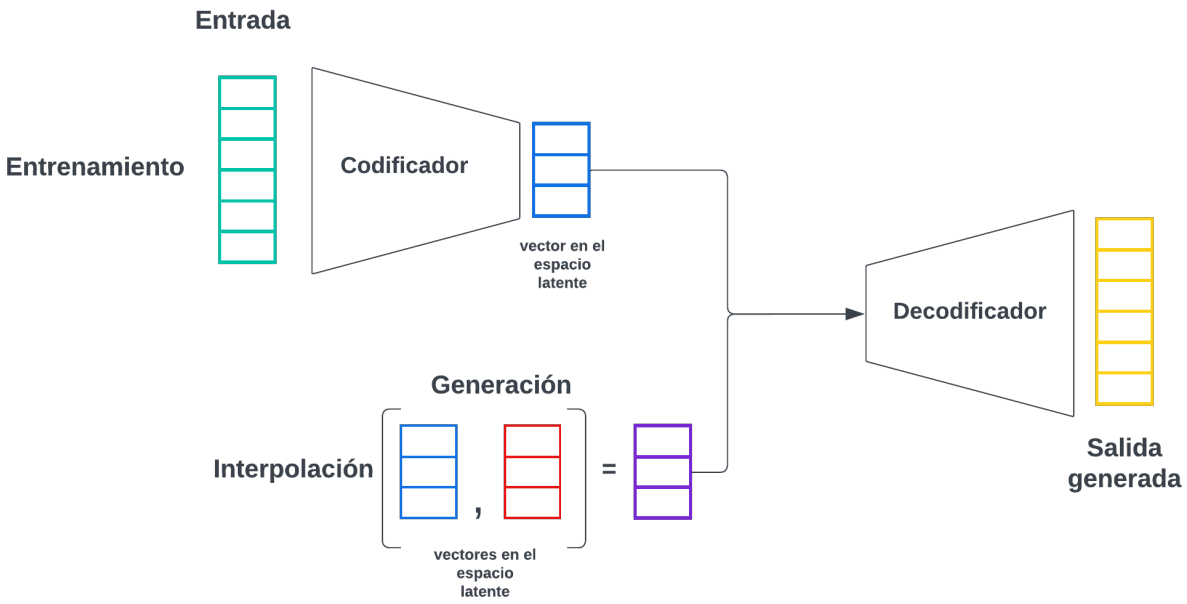


Figura 2.5: Entrenamiento y generación por interpolación en *ASCIML*.

2.2. Generación dinámica de conjunto de datos

Para articular el IML es necesario que el usuario sea capaz de crear datos dinámicamente. En los ejemplos mencionados en el apartado Sección 1.1 la generación de datos de entrenamiento se lleva a cabo de manera interactiva valiéndose de dispositivos de control como sensores, controladores, etc. (figura 2.6).

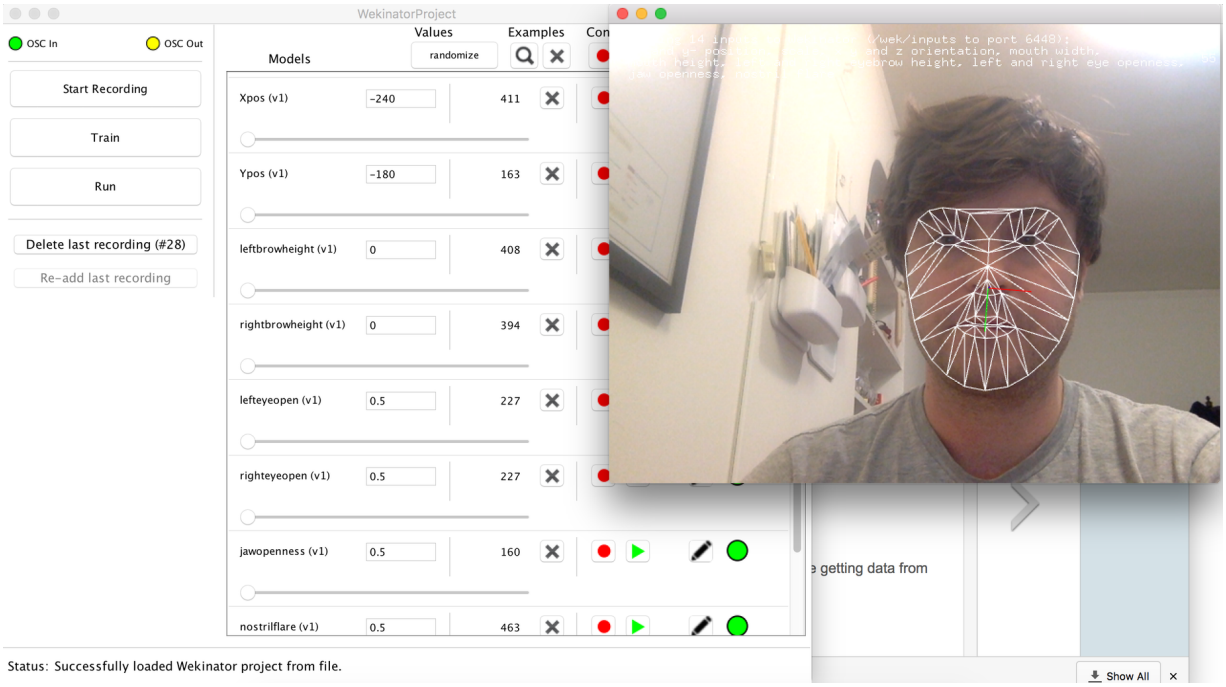


Figura 2.6: Recopilación de datos para *Wekinator* con *Face OSC*.
Nota: En la imagen se observa en entrenamiento del *Wekinator* con los datos de posición y movimiento de distintas partes de la cara. Fuente: Tomado de [32].

En contraste, dentro del paradigma convencional del aprendizaje automático en música, es común crear o utilizar conjuntos de datos y repositorios anotados y comentados, como *MagnaTagATune* [33], *AudioSet* [34], *mirdata* [35] y *FSD* [36] entre otros ([37]). Sin embargo, al consultar la documentación y sus artículos correspondientes queda claro que las intenciones de estos proyectos son para estudiar tareas específicas en el ámbito del procesamiento de señales de audio y la recuperación de información musical, y no necesariamente consideran una aplicación para la generación de audio o la composición. Diseñar conjuntos de datos que aborden estas problemáticas como una tarea integral puede llegar a ser una actividad exhaustiva, ya que existen múltiples estilos y búsquedas estéticas entre

los usuarios.

Además, pedirle a un usuario que genere grandes cantidades de materiales musicales a "mano" para solamente entrenar un modelo, requiere un doble esfuerzo innecesario y lo aleja del ejercicio creativo. Por lo tanto, el IML y su enfoque, dinámico, modular e interactivo ofrecen una alternativa viable para crear conjuntos de datos *ad hoc* a cada usuario y búsqueda. Tomando en cuenta lo anterior, en *ASCIML* se optó por implementar tres distintas formas de generación: la síntesis, la grabación directa y la carga de archivos. Estas permiten diseñar conjuntos de datos de manera flexible y rápida adaptable a un rango amplio de búsquedas sonoras[2].

2.2.1. Síntesis

Al comienzo del proyecto de investigación se generó una clase de *SuperCollider* que implementaba distintas técnicas de síntesis. En esta etapa temprana del proyecto se buscaba establecer conexiones entre *SuperCollider* y *Python* como las investigaciones que influenciaron el desarrollo de este proyecto. Sin embargo, tras unas pruebas preliminares surgió la necesidad de unificar el proceso de generación dentro de una misma plataforma para volver todo más accesible y operable. Dado que *SuperCollider* solo se utilizaba para generar audios y las demás operaciones ocurrían en *Python*, se descartó esta plataforma y se realizaron funciones para sintetizar sonido desde cero. Para lograr lo anterior se utilizó la librería de *NumPy* y *SciPy* para realizar la clase *SynthGen* en *Python* y utilizarla dentro del prototipo [3]. *ASCIML* implementa actualmente cuatro tipos distintos de síntesis de audio: amplitud modulada, frecuencia modulada [38], síntesis aditiva y sustractiva.

Antes de continuar con las partes de la clase, es necesario describir la función *envelope* que implementa el prototipo. Esta función consiste en concatenar arreglos con una distribución lineal para representar los distintos segmentos de la envolvente (ataque, decaimiento, sostenimiento y relajación). Similar que la clase *SynthGen*, la función envelope toma la

[2]Los resultados del uso de estas técnicas se comparten en el siguiente capítulo.

[3]Ejemplos en la literatura proponen formas más eficientes para trabajar *DSP* (e.g Wavetables). Sin embargo, por la duración de los sonidos que se piensa generar (1 segundo) y el poder de cómputo de NumPy (operaciones vectoriales) estas implementaciones no se necesitaron para el prototipo.

frecuencia de muestreo multiplicándola por las duraciones de los distintos segmentos y realiza una división redondeada (//) (código 1) para obtener el valor de cada uno de los puntos por los cuales será multiplicada la señal.

```
def envelope(atk,dcy,rel, durs, sr):
 env = np.concatenate((np.linspace(0,atk,int(durs[0]* sr//1)), #atk
                       np.linspace(atk, dcy, int(durs[1]* sr//1)), #dcy
                       np.linspace(dcy, dcy, int(durs[2]* sr//1)), #sus
                       np.linspace(dcy,0, int(durs[3]* sr//1))),axis = None) #rel
 if env.shape[0] < sr:
   env = np.concatenate((env,np.zeros(sr - env.shape[0])))
 return env

```

Código 1. Función *envelope*.

SynthGen es una clase que toma como argumentos la frecuencia de muestreo y una envolvente para generar la señal.

```
synth = SynthGen(sr16k, z)
```

Código 2. *SynthGen*.

Esta clase utiliza un oscilador sinusoidal como unidad generativa partiendo de la siguiente fórmula:

$$x(t) = Asin(2pift + \theta) \tag{2.2}$$

Donde A representa la amplitud, f el número de oscilaciones o ciclos por segundo. $2pi * f$ la frecuencia angular y la θ. Los tipos de síntesis están disponibles en la clase a manera de métodos: *am*, *fm*, *additive*, *substractive*. El método *am* de la clase toma el oscilador antes mencionado para realizar modulación de amplitud siguiendo la fórmula:

$$S(t) = [A_c + A_m \cos(2\pi f_m t)]cos(2\pi f_c t)] \tag{2.3}$$

Donde m representa la señal moduladora y A_m la portadora. En ambos osciladores f representa la frecuencia. Por otro lado, el método fm utiliza la siguiente fórmula:

$$y(t) = A_c \cos(\frac{2\pi f_c t_i}{f_s} + K \sin(\frac{2\pi f_m t_i}{f_s})) \tag{2.4}$$

Donde:

- Ac es la amplitud de la señal moduladora.
- f la frecuencia de la portadora en Hz.
- f_m la frecuencia de la moduladora en Hz.
- $K = \frac{A_m f'}{f_m}$.
- t_i número de muestra.
- f_s la frecuencia de muestreo.

Asimismo, el método *additive* emplea el código 3 para generar múltiplos de una frecuencia fundamental y normalizar la amplitud resultante, la suma de estos osciladores se multiplica por la envolvente (código 3, línea 21).

```
def additive(self, freq = 100, harms = 3, dur = 1):
    i = 1
    t = np.linspace(0, dur, math.ceil(self.sr * dur))
    sig = np.zeros(t.size)
    for harm in range(harms):
      if i == 0:
        x = np.multiply(sinusoid(freq * harm, dur),0.5)
        sig = sig + x
        i+=1
      else:
        x = np.multiply(sinusoid(freq * harm, dur), np.random.rand(1)[0])
        sig = sig + x
        i+=1

    if np.max(sig)>abs(np.min(sig)):
       sig = sig / np.max(sig)
    else:
       sig = sig / -np.min(sig)
    return sig * self.env
```

Código 3. Función *additive* de *SynthGen*.

El último método disponible de la clase es *substractive*. Esta función utiliza un filtro pasa-banda (código 4, línea 3-13) que filtra ruido blanco (código 4, línea 17) donde se puede especificar el límite inferior y superior de frecuencia en el filtro. Si bien era posible implementar otro tipo de osciladores, la prioridad de la investigación no es el desarrollo *DSP* convencional, por lo tanto, se decidió priorizar implementaciones de algoritmos sencillos para entender su pertinencia en el IML.

2.2.2. Códigos para la grabación y subida de archivos

Para la subida y grabación de archivos se decidió utilizar soluciones ya desarrolladas previamente. En la primera se utilizó el módulo *files* de la librería *google.colab*, la cual se integra sin mayor dificultad al prototipo, ya que es nativa de *Colaboratory* y *ASCIML* fue programado en la misma plataforma [4]. Igualmente, para grabar audio directo en el prototipo implementaron distintas funciones disponibles en la red; sin embargo, algunas

[4]Para mayor información respecto a *Colaboratory*: sección: 2.3.

```
def sustractive(self, lowcut= 800, highcut = 1000):

    def butter_bandpass(lowcut, highcut, fs, order=5):
        nyq = 0.5 * fs
        low = lowcut / nyq
        high = highcut / nyq
        sos = butter(order, [low, high], analog=False, btype='band', output='sos')
        return sos

    def butter_bandpass_filter(data, lowcut, highcut, fs, order=5):
            sos = butter_bandpass(lowcut, highcut, fs, order=order)
            y = sosfilt(sos, data)
            return y

    num_samples = self.sr
    sig = np.random.rand(num_samples) * 2 - 1
    sig = butter_bandpass_filter(sig, lowcut, highcut, num_samples)

    return sig * self.env

```

Código 4. Clase *substractive* de *SynthGen*.

no funcionaban o presentaban errores al abrir el cuaderno en equipos distintos. Al final se utilizó la implementación propuesta por Sadaf Shafí [5] que combina *HTML*, *JavaScript* y *Python*.

En esta sección en particular es necesario mencionar que ambas funciones forman parte de una estructura más grande dónde se realizan otro tipo de procesamientos a las señales que involucran directamente la interfaz y su interacción con el usuario. Esto se discute a mayor profundidad en el apartado Subsección 2.3.2.

[5] En el siguiente enlace se puede consultar el bloque de código diseñado por Sadaf utilizado en *ASCIML*: `https://stackoverflow.com/questions/55701631/how-can-we-use-microphone-in-google-colab`

2.2.3. Exploración del código y *upsampling*

Para la generación de nuevos materiales es necesario encontrar métodos para explorar el código (Z) o espacio latente del modelo (véase Sección 2.1). En investigaciones previas [28, 22, 39] los autores han utilizado la interpolación lineal como método exploratorio del espacio latente en modelos generativos. Autores como Tatar et al. [39] denominan este proceso como "*«Latent Timbre Synthesis»*" porque los materiales resultantes de esta técnica suelen compartir cualidades tímbricas [28]. Por lo anterior, esta técnica fue tomada como base para el desarrollo de la herramienta.

La interpolación lineal a través del código (Z) se realiza utilizado la representación latente de dos audios. A estos vectores se les puede calcular el promedio utilizando la siguiente fórmula:

$$nuevosonido = \frac{Z_n + Z_m}{2} \tag{2.5}$$

Al calcular puntos intermedios equidistantes entre los vectores es posible generar sonidos con un mayor porcentaje a uno que a otro. Para realizar lo anterior se utilizó el siguiente código:

```
def latent_inter(audio_a, audio_b, step = 100, porcentaje = 50, audio_list = Xw):
  if step == 1:
    listas = np.linspace(audio_a, audio_b, step)
    res = latent_reconstruction(listas[porcentaje-1:])
  else:
    listas = np.linspace(audio_a, audio_b, step)
    res = latent_reconstruction(listas[porcentaje-1:])
  return res
```

Código 5. Generación de puntos equidistantes en el espacio latente.

Ya que la creación y producción musical trabajan con frecuencias de muestreo entre 44.1 kHz - 96 kHz y el modelo entrena con señales a 16 kHz, resulta necesario ejecutar un proceso de *upsampling* de las señales producidas en *ASCIML*. Para realizar esta tarea se

implementó el código 6 que realiza interpolación lineal entre las muestras existentes de la señal para generar señales con la frecuencia de muestreo deseada.

```
def upsampling_by_interpolation(signal, input_fs, output_fs):
    scale = output_fs / input_fs
    n = round(len(signal) * scale)
    upsampled_signal = np.interp(
        np.linspace(0.0, 1.0, n, endpoint=False),
        np.linspace(0.0, 1.0, len(signal), endpoint=False),
        signal,
    )
    return upsampled_signal

```

Código 6. *Upsampling* de señales.

Hasta este punto se ha presentado el diseño y la implementación de las principales funciones del prototipo *ASCIML*, las cuales están enfocadas en la generación y manipulación de sonidos a través de técnicas de síntesis y grabación. Estas funciones permiten crear conjuntos de datos de manera flexible y adaptable a una amplia variedad de búsquedas sonoras, lo que hace que el prototipo sea una herramienta útil para la creación musical y la exploración sonora. A continuación, se abordará la integración de estas funciones en un entorno de aprendizaje automático interactivo para la generación de sonidos.

2.3. Prototipo

ASCIML[6] fue diseñado como un *Jupyter notebook*[7] operando en *Colaboratory* de Google. Esta elección se hizo con el propósito de organizar el ejercicio en pasos, aprovechando las celdas de código y texto disponibles en Jupyter, así como para aprovechar los recursos computacionales proporcionados por Google a través de Colaboratory. Una de las ventajas de utilizar Colaboratory es la posibilidad de ejecutar el código en entornos sin GPU, ya que Google provee acceso a potentes recursos de cómputo en la nube. Esto permite a los usuarios utilizar *ASCIML* incluso en computadoras que no disponen de una unidad de procesamiento gráfico. Además, Colaboratory proporciona una interfaz de usuario familiar y una integración fluida con otras herramientas y servicios de Google, lo que facilita el flujo de trabajo y la colaboración en proyectos de investigación y desarrollo de aprendizaje automático. [8]

Este *notebook* consta de tres secciones: Diseño de datos, Entrenamiento y Generación de audios (figura 2.7). En la primera sección, el usuario puede generar conjuntos de datos personalizados y etiquetarlos. Estos conjuntos pueden ser creados utilizando el micrófono de la computadora, cuatro tipos de síntesis o cargando archivos de audio. Durante la creación del conjunto de datos también se extraen descriptores de audio que permiten describir distintos parámetros de los sonidos (altura, amplitud y timbre). La conjunción de las etiquetas y descriptores permite al usuario visualizar tablas, histogramas y diagramas de dispersión (figura 2.7).

[6] `https://colab.research.google.com/github/mezaga/ACIML-prototype/blob/main/ASCIML_prototype_lab.ipynb`.

[7] Jupyter Notebook es una aplicación web de código abierto que permite crear y compartir documentos interactivos que contienen código en vivo, visualizaciones, texto explicativo y otros elementos. Fue desarrollado originalmente como parte del proyecto IPython.

[8] *Colaboratory*, más comúnmente conocido como «Google Colab» o simplemente «Colab», es un proyecto de investigación para crear prototipos de modelos de aprendizaje automático en potentes opciones de hardware como GPU y TPU. Proporciona un entorno de Jupyter notebook sin servidor para el desarrollo interactivo [40].

La segunda sección consta del entrenamiento interactivo donde un modelo nuevo o pre-entrenado puede ser entrenado con las especificaciones dadas por el usuario. En la sección de evaluación el usuario puede supervisar el proceso de aprendizaje del modelo mediante herramientas audiovisuales.

En la última sección el usuario puede visualizar la organización interna del modelo y escoger etiquetas de distintos audios para crear de manera interactiva nuevos audios interpolando entre las distintas representaciones internas del modelo. El resto del capítulo se dedicará a describir a mayor profundidad estas secciones y sus partes.

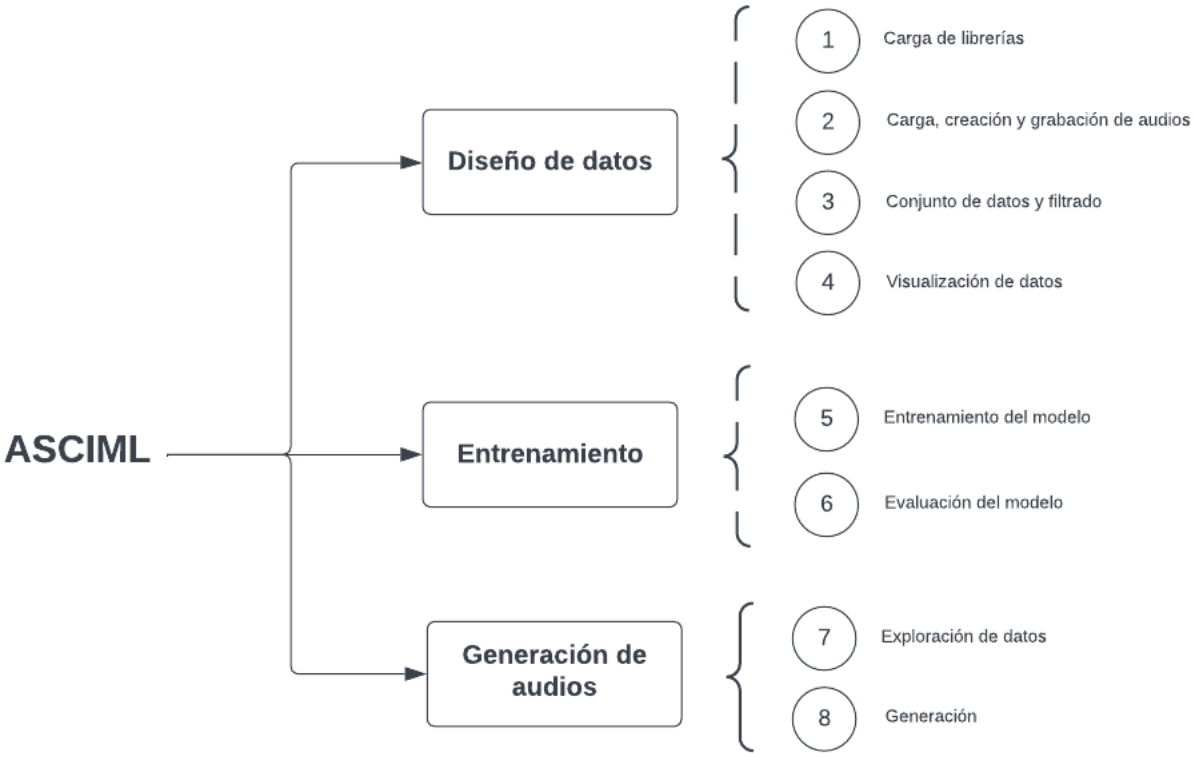


Figura 2.7: Partes de *ASCIML*.

2.3.1. Diseño de datos (Carga, exploración y visualización)

Esta primera sección de *ASCIML* comprende de la celda 1 a la celda 4 del *notebook*. La primera celda inicializa el servidor, descarga y descomprime los modelos encapsulados en un archivo *.rar* guardados en una locación remota. Una vez finalizado lo anterior, se cargan automáticamente todas las librerías de *Python* necesarias para la correcta operación de *ASCIML*. Aparte de *NumPy* [41], *pandas* [7] y *scikit-learn* [42] para el manejo de datos, *ASCIML* utiliza *SciPy* [43] y *librosa* [41] para operaciones y análisis de señal, *Keras* para generar el autocodificador de variación y *Ipython* [44] y *ipywidgets* para la parte gráfica interactiva con el usuario. En la celda 2, el usuario tiene la posibilidad de generar su base de datos con tres distintas estrategias: grabación, carga de audios y síntesis. A continuación, se describe cada una de estas formas de diseño actualmente disponibles en el prototipo.

2.3.2. Grabación y subida de archivos

Al escoger alguna de estas dos opciones, el usuario tiene la capacidad de grabar con el micrófono de la computadora y/o subir la cantidad deseada de archivos de audio. Estos se almacenan en la memoria temporal del *Colab*. En seguida, todos los archivos son resampleados a 16 kHz y todos los archivos con duración mayor a un segundo son cortados en ventanas de un segundo (Anexo 3). Cada una de las ventanas son multiplicadas por una envolvente Hamming y se guarda la señal en un diccionario junto a la extracción de sus características (transformada rápida de Fourier (FFT), f_0,centroide espectral, *root-mean-square*) (figura 2.8) (Anexo 3).

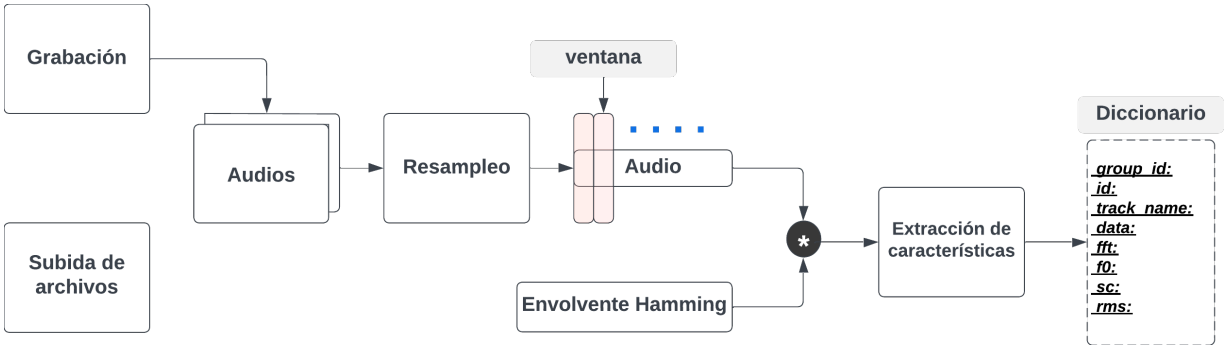


Figura 2.8: Diagrama del proceso de grabación y subida de archivos en *ASCIML*.

2.3.3. Síntesis

En esta función del prototipo el usuario asigna el identificador del grupo (e.g., "sonidos de campanas", "percusiones") y la cantidad de archivos que desea generar. Posteriormente, el usuario puede manipular los parámetros del sintetizador deseado (*fm*, *am*, sustractiva, aditiva) el cual es multiplicado por una envolvente. Los parámetros de la envolvente que pueden ser manipulados son el porcentaje de duración de cada segmento y la amplitud como rango. Al audio resultante se le extraen las características y es almacenado de igual manera en el diccionario (figura 2.9).

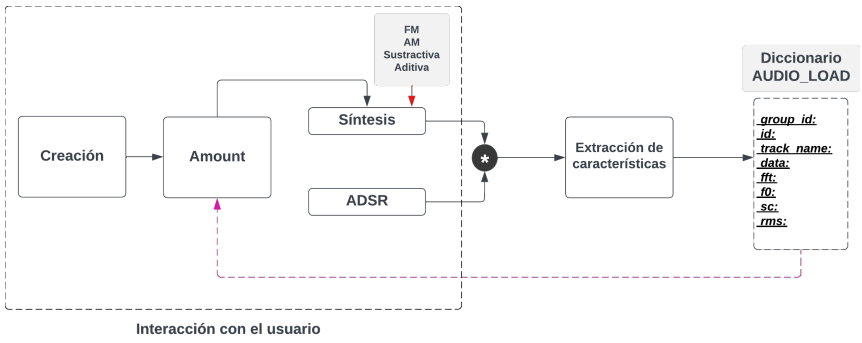


Figura 2.9: Diagrama del proceso de síntesis en *ASCIML*.

Como es posible observar a través de las tres estrategias generativas, los datos son almacenados con la misma información en el diccionario, esto permite usar la librería *pandas* [9] para generar una sola base de datos con la siguiente información:

- **ID de grupo:** Independientemente de la forma de carga, el usuario asigna un nombre de grupo y este se almacena en esta columna.
- **ID de pista:** El identificador de grupo se expande con una terminación numérica para lograr identificar cada elemento del grupo ($Ex.audio_1, audio_2$).
- **Trackname:** Esta información hace referencia al nombre original del archivo. Cuando los audios son sintetizados es similar al dado en el ID de pista.

[9] `https://pandas.pydata.org/`

- **Data:** En esta columna se almacenan los valores originales de amplitud de cada una de las señales.
- **FFT:** Tras realizar el cambio de base al dominio de la frecuencia con una ventana de 2048 muestras y solapamiento de 512, los datos resultantes del análisis espectral de cada audio se guardan en esta columna.
- **F_0**: Tras analizar con el algoritmo de *pyin* de *librosa* [10], se promedia la fundamental a través de cada uno de los audios.
- **SC:** El promedio del centroide espectral de cada archivo.
- **RMS:** La información de la envolvente individual por ventana.

De manera particular, las características extraídas del audio permiten obtener información respecto a la frecuencia fundamental (f_0), la envolvente de amplitud (RMS) y el timbre (FFT) lo cual será útil para realizar visualizaciones de los conjuntos.

En seguida en la penúltima celda de esta sección (celda 3) el usuario tiene la posibilidad de filtrar la tabla generada con *pandas* de manera interactiva al operar unos sliders creados con *ipywidgets* que reducen la cantidad de puntos de datos generados en caso de ser necesario. Esto se logra a partir de crear subconjuntos de los audios por ID (establecidas previamente por el usuario) y acceder por índice a cada una de estas tablas. Al finalizar el filtrado, los subconjuntos se concatenan como una nueva tabla (figura 2.10).

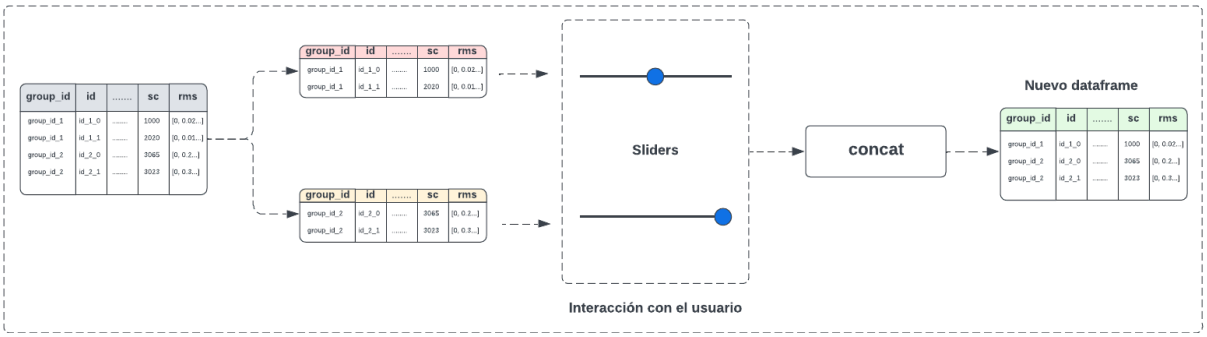


Figura 2.10: Manipulación de tablas para filtrado por IDs.

[10] `https://librosa.org/doc/main/generated/librosa.pyin.html`

En la última subsección (celda 4) el usuario tiene la posibilidad de visualizar su conjunto de datos final. Las visualizaciones disponibles permiten al usuario entender las características sonoras de las distintas clases. Para esto, *ASCIML* proporciona visualizaciones como la dispersión de f_0 de los archivos por ID, el histograma de los archivos por ID, la dispersión espectral tomando en cuenta el promedio de frecuencias, el *envelopegram* [45] que consiste en la envolvente promedio de los archivos por ID y finalmente un análisis de dos componentes principales del FFT (PCA)[11](figura 2.11, 2.12). Esto permite al compositor entender posibles relaciones de los datos tanto estadísticas como del material tímbrico. De ser necesario, el usuario puede volver a filtrar el conjunto de datos hasta que visualice las relaciones entre los datos que satisfaga su práctica compositiva.

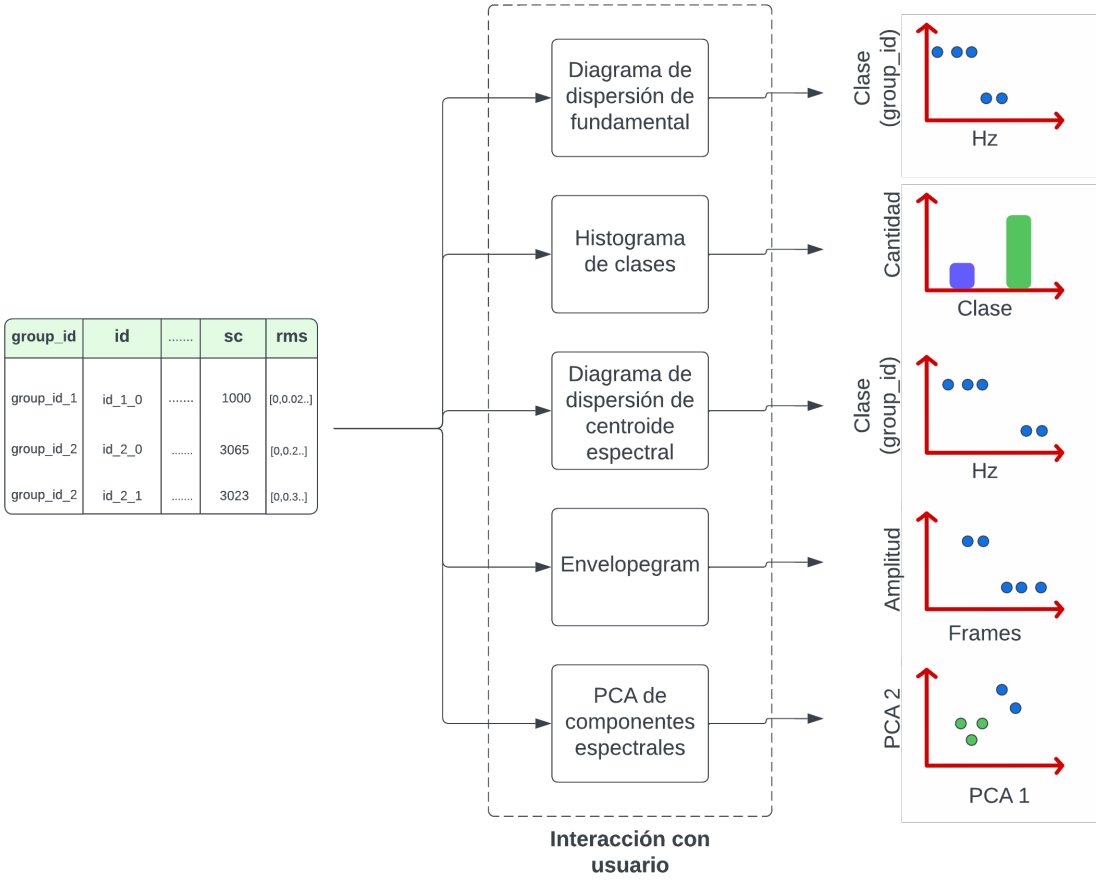
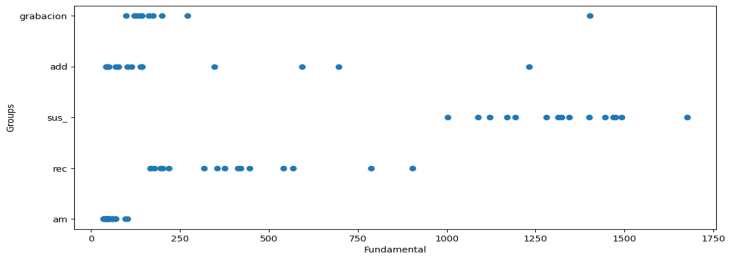
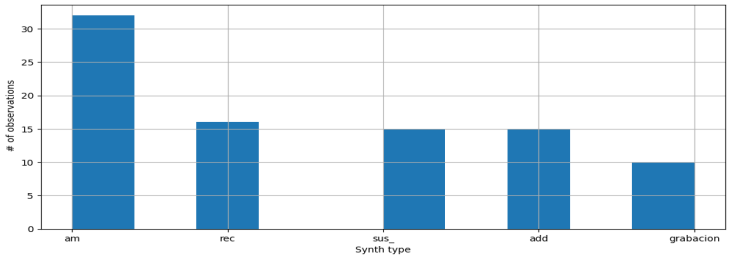


Figura 2.11: Asignación de características extraídas y visualizaciones disponibles en *ASCIML*.

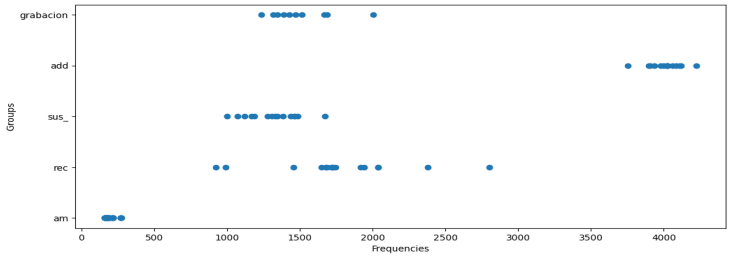
[11]https://towardsdatascience.com/principal-component-analysis-pca-explained-visually-with-zero-math-1cbf392b9e7d



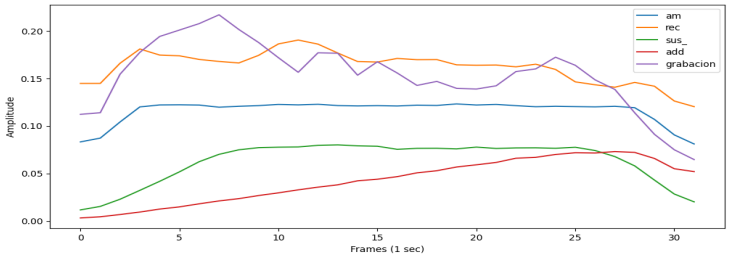
(a) Dispersión de fundamental.



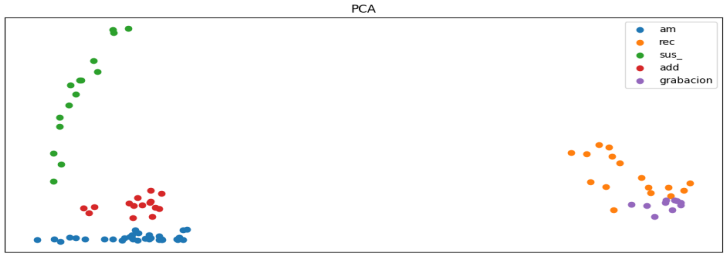
(b) Histograma de IDs.



(c) Visualización de centroide espectral de los archivos.



(d) Visualización de la envolvente promedio de los grupos de sonidos.

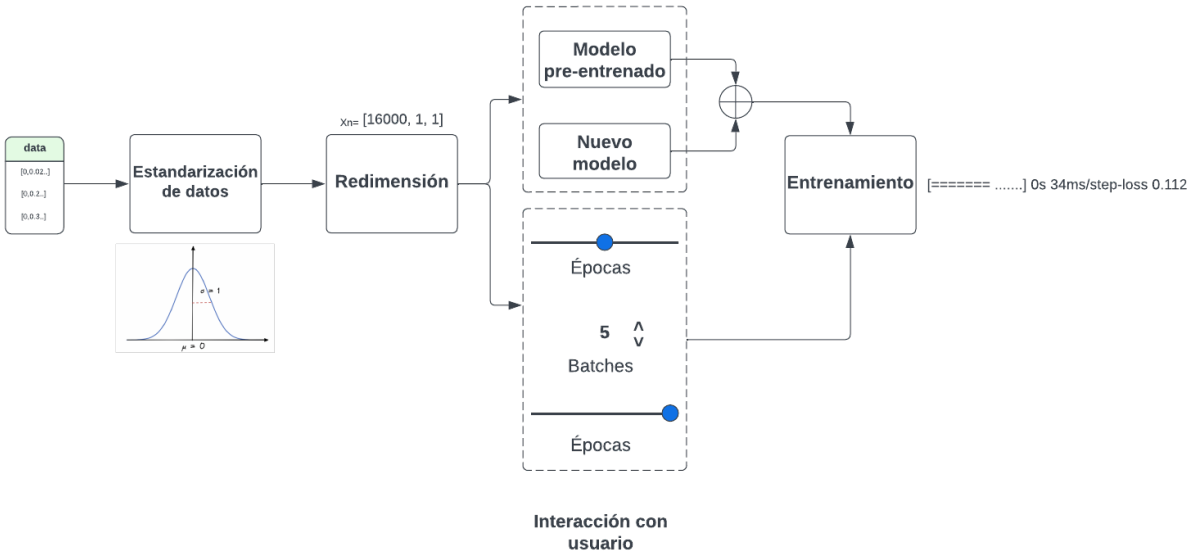


(e) Visualización de 2 componentes principales de la FFT de los archivos.

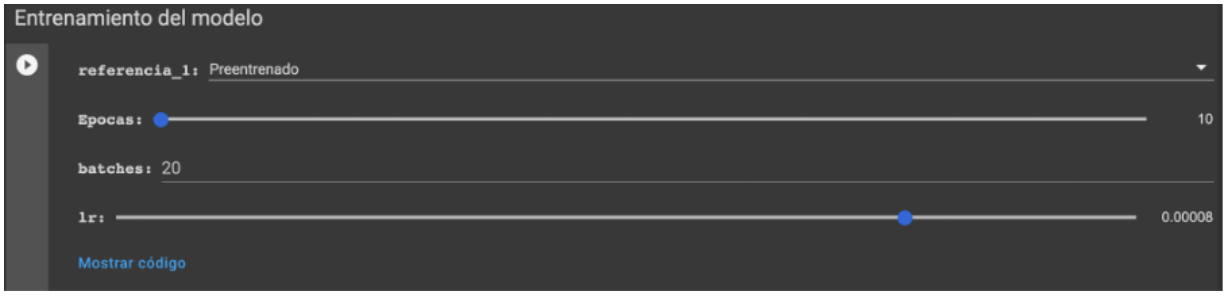
Figura 2.12: Visualizaciones disponibles en *ASCIML*.

2.3.4. Entrenamiento

En esta celda los datos son extraídos del dataframe filtrado y son estandarizados N(0,1) (figura 2.13). Ya que el modelo utiliza capas de convolución de dos dimensiones es necesario redimensionar los datos para que sean compatibles, agregando dos dimensiones vacías extra (figura 2.13). En la parte interactiva de la celda el usuario puede escoger entre un modelo nuevo o un modelo pre-entrenado[12] y modificar el número de épocas de entrenamiento, el número de *batches* y la tasa de aprendizaje mediante *sliders*.



(a)



(b)

Figura 2.13: Estandarización de datos, módulo de interacción y entrenamiento.

[12]Este modelo tiene la misma arquitectura descrita anteriormente en Subsección 2.1.1. Sin embargo, fue entrenado por 1600 épocas con 8720 sonidos sintetizados con la clase *SynthGen*.

2.3.5. Evaluación

Para la evaluación del modelo escogido por el usuario, se implementaron tres maneras distintas de evaluación. Una enmarcada dentro de la práctica común de ML como la visualización de la función de pérdida[13] y otras más orientadas a la auralidad y la visualización de elementos más comunes a los músicos, como la escucha y la visualización de la forma de onda. Estas formas de evaluación toman una muestra aleatoria de los datos de entrenamiento y la pasan por ambas partes del modelo reconstruyendo las señales y su visualización en contraposición con los archivos originales (figura 2.14).

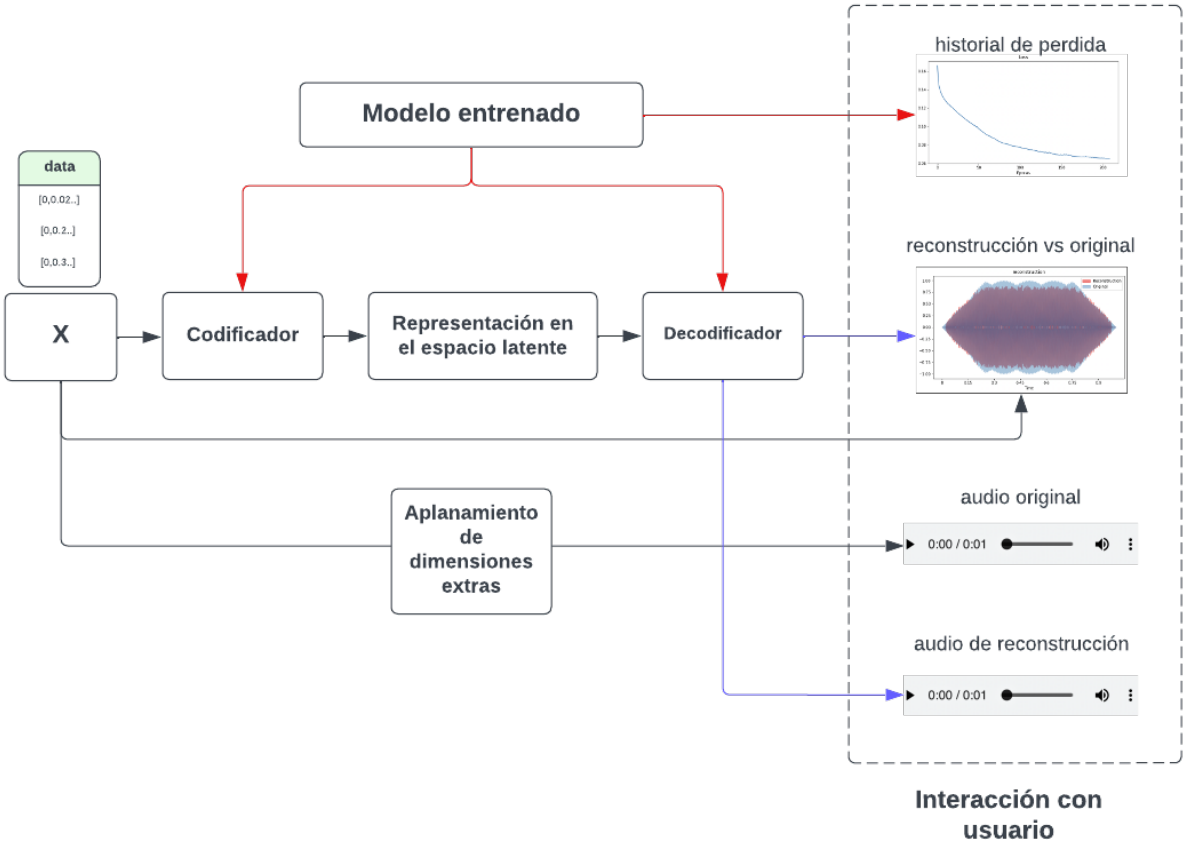


Figura 2.14: Procesamiento de datos y módulo de evaluación de *ASCIML*.

[13]La visualización de la función de pérdida en el aprendizaje automático es una técnica utilizada para representar gráficamente el rendimiento del modelo durante el proceso de entrenamiento. La función de pérdida es una medida que cuantifica qué tan bien se está desempeñando el modelo en función de las diferencias entre las predicciones del modelo y los valores reales del conjunto de datos de entrenamiento.

2.3.6. Generación

En esta celda el usuario nuevamente tiene acceso a la visualización de los datos pero, a diferencia de la celda 4, ahora se visualiza la representación interna (código). En un principio, el usuario evalúa la celda y puede visualizar una reducción a dos componentes principales de la representación interna del modelo de los archivos de entrenamiento. En la siguiente pestaña de la misma celda el usuario tiene acceso nuevamente a una tabla similar a la de la celda 3 (figura 2.15). El uso de ambas pestañas en conjunto o separadas buscan alimentar el imaginario de los compositores para tratar de buscar relaciones entre dos audios ya sea de complemento o contraste con la finalidad de generar nuevos materiales a través de la interpolación.

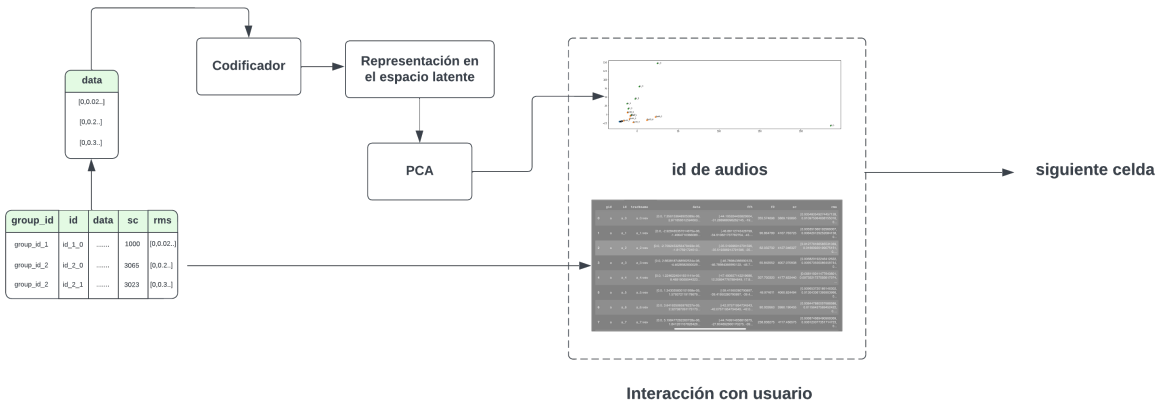
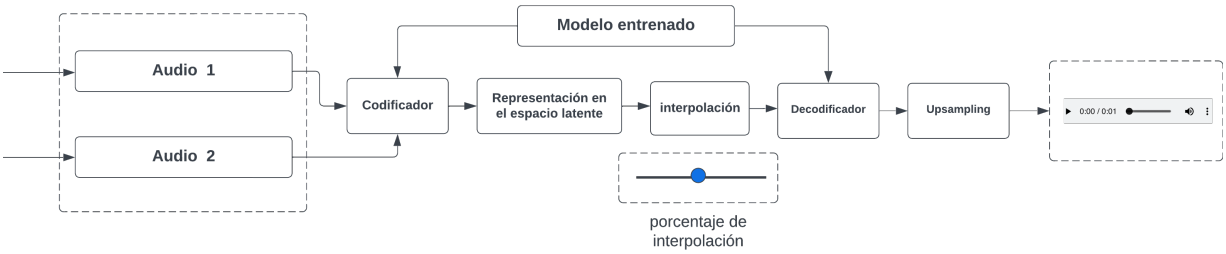
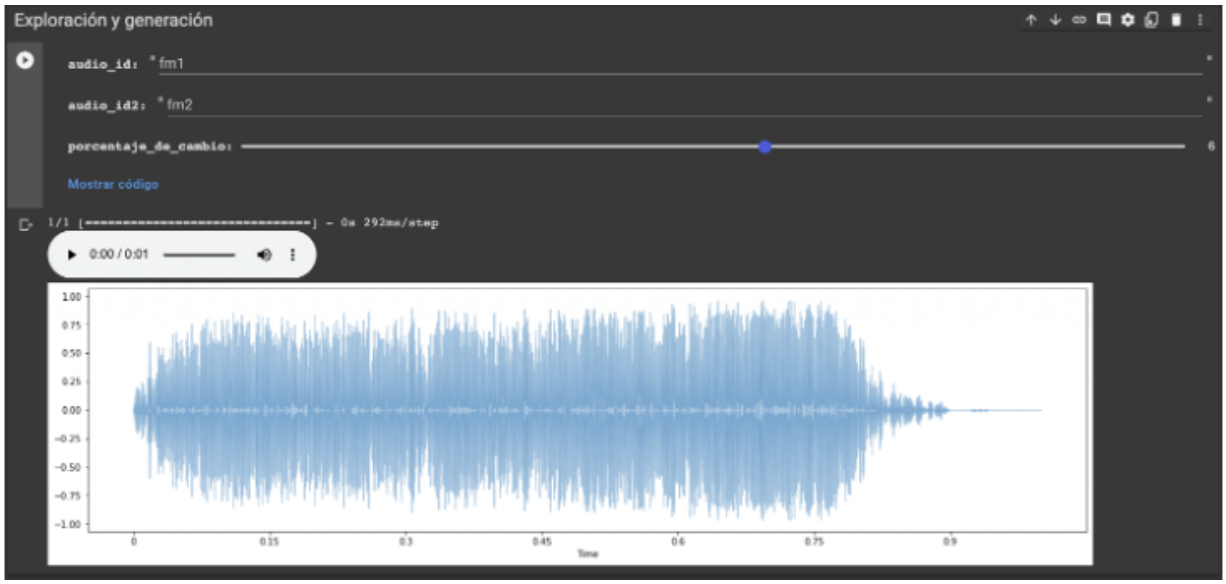


Figura 2.15: Primera sección del módulo generativo (tabla y PCA a 2 dimensiones).

Una vez seleccionados los audios e introducidas sus etiquetas, se pasan por el codificador para obtener su representación interna (figura 2.16) y así utilizar la función de interpolación discutida en el Subsección 2.2.3 El usuario controla el porcentaje de interpolación operando un *slider*, el nuevo vector generado es alimentado al decodificador para generar el nuevo sonido y posteriormente ser pasado de 16 khz a 44.1 khz. Este nuevo audio o audios pueden ser escuchados por el usuario y en caso de ser musicalmente interesantes pueden ser descargados para su utilización o manipulación en otras plataformas.



(a)



(b) Módulo generativo de *ASCIML* (#2).

Figura 2.16: Segunda sección del módulo generativo (diagrama e interfaz).

A través de este capítulo se mostraron las distintas funciones y códigos utilizados para la programación de las distintas partes del prototipo. Asimismo, se mostró la conjugación de estas tecnologías, sus puentes y los apartados de interacción con el usuario. Todo lo anterior forma parte de *ASCIML* y le permite realizar sus distintas tareas. Por un lado, para el proceso de diseño de conjuntos, el prototipo combina tres distintas formas de generar audios (carga de archivos, síntesis y grabación), lo cual posibilita generar conjuntos de datos de manera dinámica y flexible. Además, para el proceso de entrenamiento *ASCIML* cuenta con un autocodificador de variación que se entrena directamente con las señales. Durante el proceso de entrenamiento, el modelo aprende a recrear estos audios con la finalidad de organizar una representación más compacta de los mismos. A través de los distintos elementos audiovisuales del prototipo, el usuario puede monitorear el proceso mencionado anteriormente. Finalmente, *ASCIML* cuenta con un apartado generativo que

usa la interpolación lineal para que el usuario pueda generar nuevos archivos, los cuales pasan por un proceso de *upsampling* para poder ser utilizados en otras plataformas como DAWs. En el siguiente capítulo se mostrará la utilización de la herramienta por músicos con distintos perfiles y se mostrarán los resultados de esta interacción, partiendo de la pertinencia de las herramientas dispuestas y su aplicación en ejercicios creativos.

Capítulo 3

Laboratorios de prueba y retroalimentación práctica

En este capítulo se hablará sobre el diseño de los laboratorios, el instrumento de medición, el modelo generativo utilizado y los participantes. Se compartirán los resultados obtenidos tanto en los laboratorios como en las composiciones realizadas por los participantes. Por último, se revisitarán los resultados y se dará una interpretación conjunta de los datos obtenidos en ambas actividades.

En primer lugar, se diseñaron tres laboratorios con músicos de la comunidad estudiantil de la UNAM. El objetivo principal de estos laboratorios era evaluar *ASCIML* como herramienta de síntesis sonora y su relevancia para los usuarios potenciales, así como recopilar información para futuras versiones.

Para investigar lo anterior, se diseñó una encuesta que evaluaba las etapas de *ASCIML* (diseño, entrenamiento y generación) y la eficiencia de las herramientas implementadas. La encuesta utilizó una escala con tres niveles de eficiencia o relevancia de la información y también incluyó preguntas específicas para cada etapa, como los criterios musicales de diseño, el modelo preferido y los criterios de generación. Además, se recopiló información sobre el tiempo que les llevó a los participantes entrenar los distintos modelos y realizar

cada etapa de *ASCIML* (Anexo 5).

Una vez que se consideró la pertinencia de los laboratorios, se planteó una dinámica para entender el funcionamiento de *ASCIML*, sus posibles fallas y utilizaciones reales fuera del diseño experimental anterior y su uso en un entorno menos controlado más cercano a su uso real. Se pidió a un grupo de participantes de los laboratorios que realizaran una miniatura operando el prototipo de manera independiente a lo largo de una semana.

Durante el invierno del 2022, el prototipo de *ASCIML* fue utilizado por un grupo de 36 músicos de licenciatura pertenecientes a la carrera de Música y Tecnología Artística de la Escuela Nacional de Estudios Superiores (MyTA, ENES, UNAM) y la licenciatura en música de la Facultad de Música (FAM) de la UNAM.

(a) Segundo laboratorio (MyTA) (b) Tercer laboratorio (FAM)

Figura 3.1: Laboratorios con *ASCIML*

En total se realizaron tres laboratorios con los participantes, el primer laboratorio se llevó a cabo con estudiantes de primer semestre de MyTA, el segundo con alumnos de tercer semestre en la misma carrera y el último aproximadamente un mes después con los estudiantes pertenecientes a la FAM. Cada uno de estos laboratorios consistió en una breve introducción a los conceptos clave del ML (conjuntos de datos, entrenamiento y evaluación de modelos). Respecto a este punto es importante mencionar que solo un participante del total que participaron en el ejercicio tenía experiencia previa con el aprendizaje automático y muchos de los temas y conceptos mencionados eran nuevos.

En seguida cada participante disponía de dos horas para generar un conjunto de datos con un mínimo de 60 archivos de audio. Particularmente todos los conjuntos debían ser creados utilizando las tres técnicas disponibles en el prototipo (síntesis, carga de audios y grabación). Seguido de esto los participantes debían entrenar un nuevo modelo y evaluar el proceso de aprendizaje, una vez estuvieran satisfechos con los resultados debían generar nuevos sonidos. Una vez finalizada esta primer parte, se pidió a los participantes que entrenaran un modelo pre-entrenado con el mismo conjunto y repitieran los últimos pasos (evaluación y generación).

Una vez terminada la actividad los participantes contestaron una encuesta para recopilar sugerencias y aplicarlas a futuras mejoras. En esta se evaluaron las preferencias de los músicos en la creación de conjuntos de datos personalizados, la efectividad de diferentes visualizaciones de datos para comprender el conjunto de datos, el impacto de un modelo pre-entrenado, las estrategias utilizadas por los músicos para generar nuevos sonidos y los factores que influyen en estas decisiones, y la experiencia general de los músicos en el uso de IML para la síntesis de sonido.

En el siguiente apartado se presentan los resultados de los laboratorios y la revisión de las miniaturas realizadas con el prototipo, lo que permitirá obtener una visión más clara y completa de los hallazgos obtenidos a través de esta actividad.

3.1. Resultados

3.1.1. Enfoques creativos al diseño de conjunto de datos y utilización de *ASCIML*

Al revisar los resultados de la eficiencia de las tres formas disponibles para crear datos (carga de archivos, síntesis y grabación), el método más popular fue sintetizar audio con 10 participantes calificándolo como «muy eficiente» y 20 como «eficiente». La grabación de audio le siguió con 7 participantes calificándolo como «muy eficiente», 23 como «eficiente» y 6 como «ineficiente». Algo interesante a notar es que a pesar de que la carga

de archivos fue el método con más participantes calificándolo como «muy eficiente» (13 participantes) fue también el que tuvo más votos de poca eficiencia (8 participantes), con 5 como «eficiente» y 8 como "poco eficiente" (Figura 3.2).

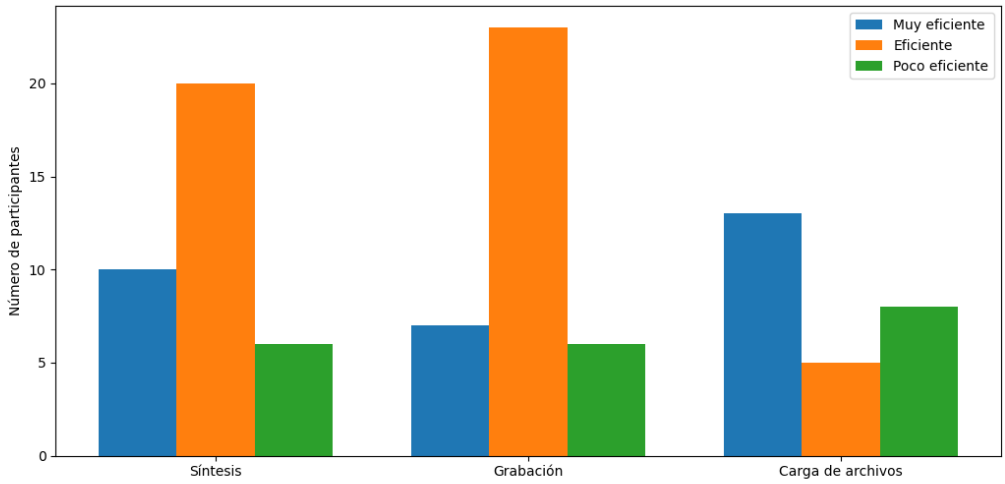


Figura 3.2: Tabla de métodos de diseño

En general, los resultados obtenidos nos indican que la mayoría de los participantes encontraron competentes los métodos disponibles en *ASCIML* para el diseño de conjuntos de datos, con una ligera preferencia por la síntesis y la grabación.

En este mismo apartado del instrumento de medición, se les preguntó a los participantes qué tipo de estrategia musical habían tenido en cuenta al diseñar sus conjuntos de datos. Dentro de las respuestas posibles se encontraban criterios como: interválico, tímbrico, evolución temporal, experimental u otro. Del total de participantes resultó que el 44.4 % tenía una aproximación tímbrica, el 36.1 % una más experimental y un 19.4 % de evolución temporal.

Al revisitar las formas de generación teniendo en cuenta las estrategias seguidas por los participantes se encontró que:

- En lo que respecta a la aproximación tímbrica un 75 % favoreció la grabación seguido de la síntesis (68.7 %) y la carga de archivos (50 %) como método eficiente.
- Un 61.5 % de los participantes que tomaron en cuenta un criterio experimental, consideró que la grabación era un método eficiente para generar sus datos. También un 54 % coincidió en que las otras dos formas eran eficientes para el diseño de conjuntos de datos.
- El 71.5 % de los estudiantes con estrategia de evolución temporal consideró muy eficaz la síntesis como táctica de diseño.

Ya que *ASCIML* cuenta con seis formas de visualizar los conjuntos de datos (tablas, PCA, F_0, histograma, centroide espectral y envelopegram) a través del instrumento de medición, se buscó entender que tan significativa resultaba para los usuarios la información mostrada.

Al revisar las respuestas partiendo de las líneas de diseño y sus opiniones respecto a las visualizaciones, podemos encontrar que los estudiantes con estrategia tímbrica encontraron muy significativa la información del *envelopegram* seguido por las tablas, el PCA y F_0 como significativas (tabla 3.1).

	Muy significativa	Significativa	Poco significativa
Tablas	6 (46.15 %)	5 (38.46 %)	2 (15.38 %)
PCA	4 (30.77 %)	6 (46.15 %)	3 (23.08 %)
F_0	7 (53.85 %)	4 (30.77 %)	2 (15.38 %)
Histograma	7 (53.85 %)	4 (30.77 %)	2 (15.38 %)
Centroide espectral	7 (53.85 %)	3 (23.08 %)	3 (23.08 %)
Envelopegram	8 (61.54 %)	3 (23.08 %)	2 (15.38 %)

Cuadro 3.1: Tiempos estimados para reconstruir la señal con nuevos modelos y modelos pre-entrenados por grupo de aproximación tímbrica.

A diferencia de la aproximación tímbrica, un 53.8 % del subconjunto experimental consideró como muy significativas tres herramientas: F_0, los histogramas y el centroide espectral (tabla 3.2).

	Muy significativa	Significativa	Poco significativa
Tablas	5 (31.25 %)	10 (62.5 %)	1 (6.25 %)
PCA	4 (25 %)	10 (62.5 %)	2 (12.5 %)
F_0	4 (25 %)	10 (62.5 %)	2 (12.5 %)
Histograma	7 (43.75 %)	8 (50 %)	1 (6.25 %)
Centroide espectral	6 (37.5 %)	9 (56.25 %)	1 (6.25 %)
Envelopegram	13 (81.25 %)	3 (18.75 %)	0 (0 %)

Cuadro 3.2: Tiempos estimados para reconstruir la señal con nuevos modelos y modelos pre-entrenados por grupo de aproximación experimental.

Finalmente, el subconjunto de evolución temporal valoró como muy significativa la información del *envelopegram*. Asimismo, los integrantes de este subconjunto encontraron significativa la información que el PCA y el F_0 brindan (tabla 3.3).

	Muy significativa	Significativa	Poco significativa
Tablas	2 (28.57 %)	3 (42.86 %)	2 (28.57 %)
PCA	2 (28.57 %)	5 (71.43 %)	0 (0 %)
F_0	1 (14.29 %)	5 (71.43 %)	1 (14.29 %)
Histograma	1 (14.29 %)	3 (42.86 %)	3 (42.86 %)
Centroide espectral	3 (42.86 %)	3 (42.86 %)	1 (14.29 %)
Envelopegram	5 (71.43 %)	2 (28.57 %)	0 (0 %)

Cuadro 3.3: Tiempos estimados para reconstruir la señal con nuevos modelos y modelos pre-entrenados por grupo de aproximación de envolución temporal.

En las primeras dos iteraciones del ejercicio, se pudo observar que los participantes destinaron una gran parte del tiempo del laboratorio a la búsqueda de muestras de sonido en repositorios en línea o en sus bibliotecas personales. Esta percepción se cotejó con las respuestas de todos los estudiantes respecto al tiempo invertido a esta parte del proceso. Los resultados demuestran que el 39 % dedicó más de 30 minutos, de los cuales el 22.2 % tenía una aproximación experimental (tabla 3.4). En cuanto a las aproximaciones tímbricas y de evolución temporal, el mayor número de participantes dedicó entre 20-30 minutos y entre 10-20 minutos, respectivamente (tabla 3.4).

Intervalo de tiempo	Experimental	Tímbrica	Evolución temporal	Total
Menos de 10 min.	-	1 (6.3 %)	-	1 (2.8 %)
10-20 min.	4 (30.8 %)	4 (25.0 %)	3 (42.9 %)	11 (30.6 %)
20-30 min.	1 (7.7 %)	7 (43.8 %)	2 (28.6 %)	10 (27.8 %)
Más de 30 min.	8 (61.5 %)	4 (25.0 %)	2 (28.6 %)	14 (38.9 %)
Total	**13 (36.1 %)**	**16 (44.4 %)**	**7 (19.4 %)**	**36**

Cuadro 3.4: Intervalo de tiempo dedicado a la etapa de búsqueda de muestras tomando en cuenta las estrategias de diseño

Cabe mencionar que al observar esto en las primeras dos intervenciones se sugirió a los participantes pertenecientes a la FAM preparar un conjunto de archivos previamente al laboratorio. Sin embargo, al revisar los datos observamos que los participantes de la FAM dedicaron también bastante tiempo al diseño de conjunto de datos (tabla 3.4).

Intervalo de tiempo	FAM	ENES	Total
Menos de 10 min.	-	1 (3 %)	1 (3 %)
10-20 min.	2 (6 %)	9 (25 %)	11 (31 %)
20-30 min.	3 (8 %)	7 (19 %)	10 (28 %)
Más de 30 min.	4 (11 %)	10 (28 %)	14 (39 %)
Total	**9 (25 %)**	**27 (75 %)**	**36 (100 %)**

Cuadro 3.5: Tiempo dedicado a la etapa de búsqueda de muestras

Con estos datos se encontró que los usuarios invierten bastante tiempo en el diseño de sus conjuntos y esto no necesariamente se debe al acto de recolección, si no, es más posible que esto se deba a las distintas estrategias de diseño. También se encontró que no existe una sola aproximación a la generación de conjuntos de datos. Finalmente, las formas de visualización disponibles en *ASCIML* han demostrado ser eficientes para auxiliar a los usuarios a diseñar sus conjuntos de datos.

3.1.2. Modelo Nuevo vs Modelo pre-entrenado

Para evaluar la pertinencia de la inclusión de un modelo pre-entrenado en esta articulación del IML se preguntó a los participantes: 1) cuál modelo funcionaba mejor para aprender

de sus datos y 2) aproximadamente en cuánto tiempo comenzaban a escuchar o ver que los modelos aprendían a reconstruir las señales de entrenamiento.

Respecto a la experiencia de los usuarios con los modelos y sus conjuntos de datos personalizados, se encontró que un 47.2 % de los participantes prefirieron un nuevo modelo para reconstruir las señales, un 38.9 % prefirió un modelo pre-entrenado y un 13.9 % opinó que ambos modelos realizaban el trabajo de buena manera (tabla 3.6).

Tras revisar más a fondo esta información se encontró que del total de participantes que prefirieron el modelo nuevo, un 77.7 % eran participantes de la FAM, los cuales centraron la mayor parte del ejercicio en los audios pre-seleccionados[1]. Es relevante mencionar que los sonidos escogidos por los estudiantes de la FAM pueden ser descritos como complejos y con una alta cantidad de sonidos inarmónicos (ex. sonidos de guitarras eléctricas, bicicletas, etc.). Esto cobra importancia porque el modelo pre-entrenado no contenía ejemplos similares a los mencionados durante su entrenamiento, lo que da sentido a la preferencia de estos usuarios por utilizar un modelo nuevo.

Modelo	**ENES**	**FAM**	**Total (%)**
Nuevo	10 (27.8 %)	7 (19.4 %)	17 (47.2 %)
Pre-entrenado	12 (33.3 %)	2 (5.6 %)	14 (38.9 %)
Ambos	5 (13.9 %)	0 (0 %)	5 (13.9 %)
Total (%)	**27 (75 %)**	**9 (25 %)**	**36 (100 %)**

Cuadro 3.6: Preferencia de tipo de modelo

Al comparar las estrategias de diseño con la utilización de ambos modelos se encontró que en la aproximación experimental los participantes tenían una opinión muy similar de ambos modelos (tabla 3.7). Por otro lado, en la exploración tímbrica seis usuarios preferían un nuevo modelo y cinco un modelo pre-entrenado. Finalmente, en la búsqueda relacionada a la evolución temporal se encontró la mayor diferencia, con cinco usuarios favoreciendo el modelo nuevo y dos el pre-entrenado.

[1] `https://drive.google.com/drive/folders/1zHNvALSIPEkgaZ4eKErT6dRzNe2Q4Zhz?usp=sharing`

Aproximación	**Nuevo modelo**	**Pre-entrenado**	**Ambos**	**Total**
Experimental	6	5	2	13
Tímbrica	6	7	3	16
Evolución tímbrica	5	2	-	7
Total	**17**	**14**	**5**	**36**

Cuadro 3.7: Preferencias de modelo según aproximación y tipo de modelo

Respecto al marco temporal de esta sección de *ASCIML* los participantes disponían de intervalos similares a los de diseño de conjuntos (<10 min, 10-20 min, 20-30 min, >30 min) para evaluar cuándo comenzaban a percibir que los modelos reconstruían la señal (aprendiendo). Los integrantes de la actividad notaron que tanto un modelo nuevo como un modelo pre-entrenado eran capaces de reconstruir las señales dentro de los primeros 10 minutos de entrenamiento. Estos resultados son particularmente interesantes al tomar en cuenta nuevamente los criterios de diseño y observar que los resultados eran similares a través de las distintas categorías (tabla **??**).

	Experimental		**Evolución temporal**		**Tímbre**	
Intervalo	Modelo nuevo	Modelo pre-entrenado	modelo nuevo	modelo pre-entrenado	modelo nuevo	modelo pre-entrenado
<10 min	**5 (3.8 %)**	**7 (19.4 %)**	**4 (11.1 %)**	**4 (11.1 %)**	**8 (22.2 %)**	**10 (27.7 %)**
10-20 min	3 (8.3 %)	3 (8.3 %)	3 (8.3 %)	3 (8.3 %)	4 (11.1 %)	4 (11.1 %)
20-30 min	2 (5.5 %)	1 (2.7 %)	-	-	4 (11.1 %)	1 (2.7 %)
>30 min	3 (8.3 %)	2 (5.5 %)	-	-	-	1 (2.7 %)

Cuadro 3.8: Intervalo temporal según la aproximación y tipo de modelo utilizado.

Finalmente, casi la mitad de los participantes consideró que el modelo pre-entrenado dio mejores resultados en la reconstrucción de sus datos en el tiempo dado de la actividad. Los datos obtenidos demuestran que *ASCIML* logra adaptarse rápidamente a distintas búsquedas compositivas, ya sea con la utilización de un modelo nuevo o pre-entrenado. Esto puede verse a mayor profundidad con las miniaturas realizadas.

3.2. Evaluación del modelo

A diferencia de *ASCIML*, las investigaciones que influencian este trabajo, mencionadas en el primer capítulo, no cuentan necesariamente con un módulo de evaluación previamente a la interpretación o utilización. Por esto, para esta investigación resulta importante evaluar la pertinencia de este apartado y sus propuestas (función de perdida, contraste auditivo y superposición de formas de onda) (figura 3.3). Para ello se preguntó a los participantes si las herramientas audiovisuales de evaluación de *ASCIML* aportaban suficiente información para la toma de decisiones a la hora de guiar el entrenamiento del modelo (aumentar los datos de entrenamiento, entrenar por más épocas el modelo, aumentar la tasa de aprendizaje y los batches).

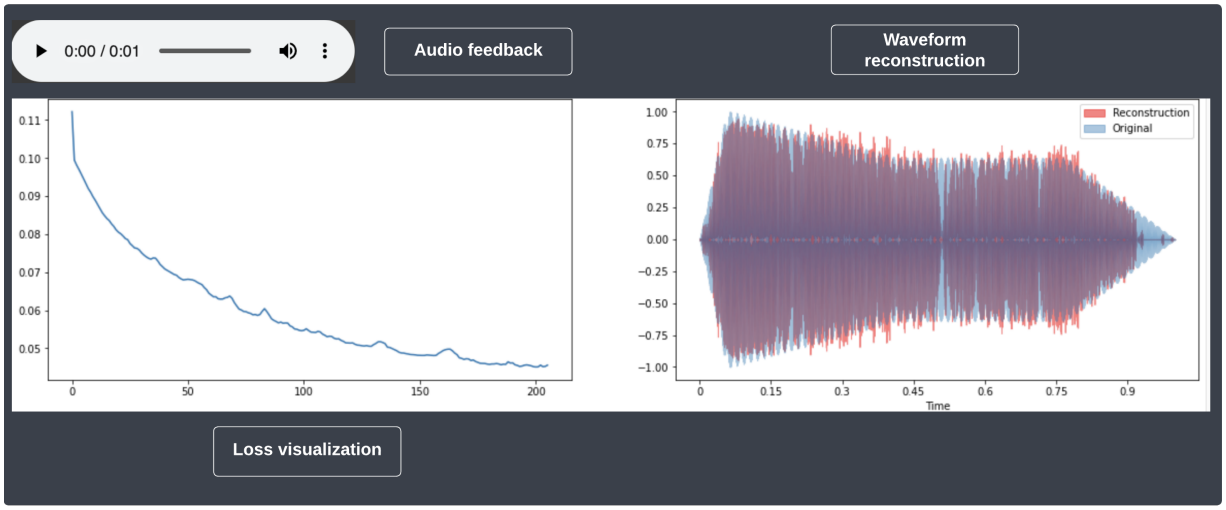


Figura 3.3: Celda de evaluación de *ASCIML*

De la totalidad de los participantes en los laboratorios, 86.1 % de ellos coincidió que las herramientas audiovisuales con las que cuenta *ASCIML* brindan la suficiente información para tomar estas decisiones. El resto de los participantes (13.9 %) a pesar de no estar satisfechos con las herramientas indicaron que de las opciones disponibles la mejor era la auditiva. De manera particular, 55.6 % de la totalidad de los músicos indicaron que el audio era la opción que más ayudaba a la tarea, seguida por la forma de onda con un 33.3 % y por último la gráfica de pérdida (11.1 %)(figura 3.4).

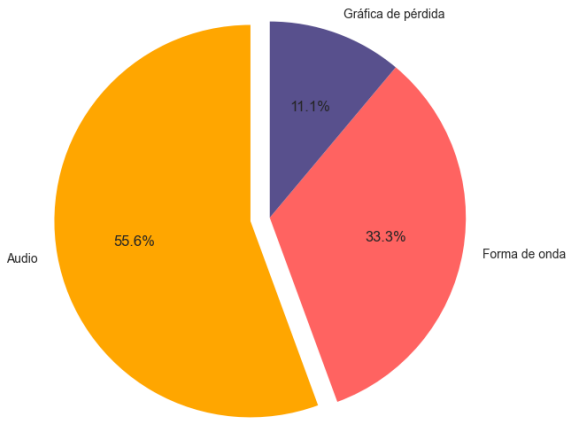


Figura 3.4: Preferencia de los músicos

Asimismo, los subconjuntos de participantes con una búsqueda experimental y tímbrica prefirieron la reconstrucción auditiva para evaluar el proceso de aprendizaje (tabla 3.9) A diferencia de estos, el subconjunto de evolución temporal prefirió la visualización de la forma de onda. Después de analizar estos datos surgió la curiosidad de investigar la preferencia del o la participante con experiencia en ML, interesantemente, se encontró que optó por la función de pérdida.

Estrategia	**Forma de onda**	**gráfica de perdida**	**Audio**
Experimental	2 (5.55 %)	3 (8.33 %)	8 (22.2 %)
Evolución temporal	4 (11.1 %)	-	3 (8.33 %)
Timbre	6 (16.6 %)	1 (2.7 %)	9 (25 %)
Total (%)	**12 (33.3 %)**	**4 (11.1 %)**	**20 (55.5 %)**

Cuadro 3.9: Preferencia de herramienta para monitoreo de entrenamiento del modelo

3.3. Generación y criterios compositivos

Similar a los criterios de diseño, los laboratorios buscaban entender los criterios de generación de los distintos compositores de nuestra comunidad. Para esto se preguntó a los músicos qué tipo de aproximación tuvieron a la hora de explorar el código resultante del entrenamiento del modelo. Las opciones disponibles en la encuesta eran: experimental, tímbrica, de evolución temporal, interválica u otras más propias de la *latent space synthesis* como la exploración visual del espacio latente.

A diferencia de las estrategias de diseño, en este punto las respuestas fueron un poco más diversas. De los criterios ya mencionados el más utilizado fue nuevamente el experimental (50 %) seguido por del tímbrico (33.3 %). Continuando con un empate entre la evolución temporal y la navegación del espacio latente (5.6 %), siendo el criterio interválico y evasión de errores los menos utilizados (2.8 %).

		Criterios de generación					
		Experimental	**Tímbrico**	**Evolución Temporal**	**Navegación del espacio latente**	**Evitar errores**	**Interválico**
Criterios de diseño	**Experimental**	**10 (27.7 %)**	1 (2.7 %)	1 (2.7 %)	1 (2.7 %)	-	-
	Tímbrico	5 (13.8 %)	**9 (25 %)**	-	-	1 (2.7 %)	1 (2.7 %)
	Evolución Temporal	3 (8.3 %)	2 (5.5 %)	**1 (2.7 %)**	1 (2.7 %)	-	-
Total de participantes por criterio de generación		**18 (50 %)**	**12 (33.3 %)**	**2 (5.6 %)**	**2 (5.6 %)**	**1 (2.7 %)**	**1 (2.7 %)**

Cuadro 3.10: Contraste entre técnicas de diseño y los criterios de generación

En este punto, resulta interesante resaltar que gran parte de los participantes pertenecientes a los distintos subconjuntos de criterios de diseño mantuvieron la misma estrategia para la el proceso de generación (interpolación). El único cambio observable se presentó en el subconjunto de evolución temporal, donde un 85 % del subgrupo cambió de opinión a otras estrategias para la generación. También es importante destacar que solo dos participantes decidieron explorar el espacio latente de manera visual y solo una persona buscó no cometer errores. Asimismo, a diferencia de los criterios de diseño, en este punto sí hubo el interés de buscar relaciones interválicas entre los materiales.

Respecto al marco temporal de la generación, se encontró que el 30.5 % de los participantes encontraron resultados musicalmente interesantes después de 30 minutos de interactuar con esta sección, un 27.7 % reportó encontrarlos entre 20-30 minutos, el 22 % entre 10-20

minutos y un 19.4 % en menos de 10 minutos. Al contrastar la estrategia de generación con los márgenes temporales, la información resulta insuficiente para comprender los factores o tendencias por los cuales a algunos usuarios les toma diferentes intervalos de tiempo encontrar resultados musicales.

Finalmente, para dar cierre a la sección de los laboratorios y dar paso a la revisión y análisis de miniaturas musicales, al comparar los tiempos invertidos por los usuarios de las distintas tareas de *ASCIML*, encontramos que un 38.8 % de los participantes dedicaron más de 30 minutos al diseño de conjuntos de datos, un 47 % reportó invertir <10 minutos en el entrenamiento de un modelo nuevo mientras que un 58.3 % invirtió menos de 10 minutos en el entrenamiento de un modelo pre-entrenado y por último, un 30 % tardó más de 30 minutos en la etapa de generación (tabla 3.11).

	Diseño de datos	**Entrenamiento (M.N)**	**Entrenamiento (M.P)**	**Generación**
Menos de 10 min	1 (3 %)	17 (47 %)	21 (58 %)	7 (19.4 %)
10-20 min	11 (31 %)	10 (28 %)	8 (22 %)	8 (22.2 %)
20-30 min	10 (28 %)	6 (17 %)	10 (28 %)	10 (28 %)
Más de 30 min	14 (39 %)	3 (8 %)	11 (31 %)	11 (31 %)
Total	**36**	**36**	**36**	**36**

Cuadro 3.11: Tiempo invertido por tarea en minutos y porcentaje de participantes.

Al revisar la proporción temporal que los participantes dedicaron a cada una de las tres partes (generación de base de datos, entrenamiento y generación) encontramos que los participantes invirtieron la mayor cantidad de tiempo en las etapas más creativas del proceso como generación de datos y la creación de nuevos sonidos.

A continuación, se discutirán una serie de miniaturas realizadas por músicos de la FAM, con la finalidad de entrever los resultados y retos de este tipo de herramientas aplicadas en escenarios que involucran *ASCIML* ya no como un prototipo, sino como una herramienta para la composición y el diseño sonoro.

3.4. Miniaturas

Una vez realizada la actividad, y contestada la encuesta con los participantes del último taller, se les pidió componer miniaturas electrónicas sin ninguna imposición estilística de una duración mínima de 20 segundos, acompañadas de una breve descripción de su enfoque composicional y uso de *ASCIML*[2]. Estas piezas debían ser desarrolladas en el transcurso de una semana, durante este tiempo los participantes trabajaron de manera individual con el prototipo. El día de la presentación de las piezas se realizó una plática por videollamada con los compositores para compartir las experiencias y hallazgos al trabajar de manera individual con el dispositivo. El propósito de esta actividad fue evaluar el funcionamiento del prototipo fuera de un ambiente controlado como los laboratorios y entender a mayor profundidad cómo los músicos realmente integraban esta herramienta a su praxis artística. En esta sección se resumen la intersección de los resultados de los escritos, la plática por videollamada y el análisis de las piezas.

3.4.1. Miniatura #1 por Andrés Limón Benítez

En este caso, el compositor expresó que el proceso para utilizar y conectar el prototipo fue bastante fácil de recordar. Asimismo, decidió trabajar con un conjunto de 10 audios pre-grabados con características muy contrastantes entre ellos como percusión vocal e instrumental y sonidos genéricos como cortinas y libros. Al terminar el proceso de entrenamiento, Andrés encontró que los sonidos generados eran una combinación interesante de ruido y altura.

3.4.2. Miniatura #2 por Kevin Requena

«Para esta pieza utilicé grabaciones de una pequeña tuerca de metal rebotando sobre una plancha metálica, una grabación del canto de un búho y otra del canto de un quetzal (aproximadamente 20 audios)»(Fragmento de entrevista). En la presentación de la mi-

[2]Las miniaturas se encuentran disponibles en el siguiente repositorio: `https://asciml.github.io/ASCIML/public/tesis.html`

niatura, Kevin compartió que utilizó modelos nuevos para entrenar y también algunos pre-entrenados. Sin embargo, por la naturaleza de sus materiales, el modelo nuevo le permitía tener resultados con menor ruido de reconstrucción. También nos compartió que composicionalmente buscaba crear híbridos entre los sonidos contrastantes metálicos y los cantos de las aves. Los elementos de estas 'clases' o grupos eran introducidos a *ASCIML* en distintas cantidades e iteraciones y para esto la visualización de las envolventes le permitía monitorear su proceso compositivo, combinando y compensando la cantidad de audios introducidos al modelo. Los resultados obtenidos con *ASCIML* fueron editados en *Audacity*. Respecto a la búsqueda estética de la pieza, Kevin nos comenta que:

> Al ser una pieza breve, pensé en construir un pequeño crescendo que inicia con sonidos breves y agudos y finaliza en un último sonido que contiene las frecuencias más graves que aparecen en toda la pieza (Kevin Requena, fragmento de entrevista)

3.4.3. Miniatura #3 Martín Alonso

Al igual que Kevin, Martín trabajó con un conjunto de audios pequeño, entre los elementos de este, se encontraban sonidos percutivos y texturales como: un palo de lluvia, canicas girando en una taza de café vacía, el timbre (campana) de una bicicleta, un globo inflándose y una trompeta.

> Yo agrupé en 5 intentos (entrenamientos) con un mismo sonido evolucionándose y luego otros 5 o 6 intentos agregando otros 2 o 3 sonidos (desde cero), así entonces conseguí algunas variantes del sonido original que pude ordenar a modo de fichas o bloques que acertadamente tenían cierta continuidad (fragmento de entrevista).

En este fragmento de la conversación por videollamada, podemos observar un alto interés en el desarrollo de los materiales a través de la variación y combinación. Asimismo, Martín nos comentó que no logró etiquetar sus fragmentos de audio al cargarlos al prototipo y, por lo tanto, las visualizaciones no fueron de mucha ayuda durante su proceso. Esto condicionó su uso de la herramienta a una exploración aural de los materiales y sus distribuciones en el código.

En estas dos miniaturas (Miniatura #2, #3) podemos observar una aproximación de mezcla entre audios de naturaleza compleja y contrastantes. Muchos de estos sonidos eran de duración mayor a un segundo, lo cual permitía tener múltiples ejemplos por clase y que el modelo aprendiera a realizar reconstrucciones utilizables, pero con poco «campo de exploración» en el espacio latente. Asimismo, los compositores priorizaron entrenamientos cortos y desde cero.

3.4.4. Miniatura #4 Carlos Flores

Carlos trabajó una pieza electrónica tonal con elementos rítmicos, melódicos y de acompañamiento. Para la sección rítmica decidió trabajar con librerías para generar *beats* (*kicks*, *snares*, *hi-hats*), ya que quería tener la posibilidad de trabajar con estos mismos sonidos, pero modificados de una manera distinta a las estrategias convencionales de producción musical. En un principio, Carlos comentó que trabajó todos los audios en conjunto para entrenar el modelo y luego cuando quería cosas más específicas entrenó el modelo por clases (>2000 épocas de entrenamiento). Al igual que Andrés, Carlos reportó que algunas muestras contenían ruido después del entrenamiento, pero que esto complementaba su búsqueda estética. Los instrumentos melódicos (guitarras, piano y sintetizadores) los trabajó como un proceso distinto (otro cuaderno); sin embargo, tuvo que entrenar el modelo sin GPU, ya que agotó su uso semanal de los recursos gratuitos de *Google*, lo cual volvió muy tardado el ejercicio y los resultados no se acercaron a los obtenidos con las percusiones. Además, reportó que los sonidos melódicos generados por *ASCIML* no estaban afinados y tuvo que trabajar con el *piano roll* y un afinador en su DAW.

Otros tres participantes que desarrollaron la actividad de la miniatura reportaron algunos problemas al evaluar el modelo, al generar las etiquetas de sus conjuntos de datos y al generar los sonidos. Al revisar estos problemas presentados durante la plática por videollamada, se encontró que los participantes respetaban poco el flujo de trabajo lineal establecido y esto originaba problemas para cargar las librerías, los datos y evaluación de las funciones.

Los resultados obtenidos a través de estas miniaturas demuestran que *ASCIML* puede llegar a ser una herramienta valiosa para la creación de sonidos para la composición, ya que logra enfocar el manejo del ML hacia el ejercicio creativo (diseño y generación).

3.5. IML y experiencia de uso

Durante esta primera etapa de investigación, encontramos que tanto los modelos nuevos como los pre-entrenados son útiles para exploraciones sonoras diversas. En su publicación[8], los autores discuten los casos [46, 47] de arte generativo en los que se re-entrenan modelos con conjuntos de datos personalizados para generar sus obras. Sin embargo, los autores también señalan que esta estrategia no es necesariamente adecuada en todos los casos. En nuestro caso particular, observamos esto en los resultados, ya que la predilección de un modelo sobre otro (nuevo vs pre-entrenado) está directamente relacionado con la facilidad de aprendizaje de los materiales y su presencia o no el en modelo pre-entrenado.

Basándonos en los hallazgos de los laboratorios y en las reflexiones de [8], parece importante continuar con el desarrollo del prototipo incluyendo ambos modelos con la finalidad de que los músicos puedan realizar un rango amplio de exploraciones musicales. Esto permitirá a los usuarios adaptar el proceso de entrenamiento a sus necesidades y preferencias individuales, manteniendo lo más amplio posible el alcance de la herramienta.

Es importante mencionar que estos hallazgos también justifican la necesidad de contar con un módulo de evaluación para este tipo de herramientas. Los resultados obtenidos en los laboratorios muestran que estas evaluaciones pueden incluir elementos audiovisuales siem-

pre y cuando la información sea presentada de manera clara y comprensible, involucrando elementos aurales o familiares al usuario. Por otro lado, aunque las herramientas de visualización de datos (histograma, PCA, F_0, etc.) no integran por completo estos elementos, se considera que se verían beneficiada de las observaciones anteriores.

También se encontró que para la mayoría de usuarios fue fácil utilizar el prototipo, mientras otra parte de los usuarios encontraban problemas al operar de manera secuencial el cuaderno, asignar etiquetas y generar nuevos materiales. Estos resultados sugieren que este tipo de herramientas a pesar de hacer más accesible el ML a los músicos, aún contiene una curva de aprendizaje que no necesariamente está relacionada con los lenguajes de programación. Esto puede deberse a la necesidad de un diseño de experiencia de usuario más detallada, la necesidad de documentación, o la poca familiaridad con el paradigma de aprendizaje automático de los usuarios. Independientemente de la causa, resulta importante resaltar que durante los laboratorios, todos los participantes lograron guiar el proceso de aprendizaje de ambos modelos y tanto en los laboratorios como en las miniaturas, los participantes lograron aplicar de manera creativa los elementos creados con *ASCIML*.

3.6. Intersección entre lo creativo y *ASCIML*

Los resultados del laboratorio muestran que de las tres opciones de métodos de diseño, la subida de archivos obtuvo el mayor número de votos como «muy eficiente» y simultáneamente como la más «poco eficiente» de las tres (Figura 3.2). En este contexto, se encontró también que los usuarios favorecen de una manera más consistente la síntesis y la grabación de manera muy similar. Esto último se reafirma al tomar las estrategias de diseño (espectral, de envolvente, experimental) en consideración y encontrar resultados similares. Lo anterior sugiere que la carga de archivos fue votada como «muy eficiente» por tantos participantes por su practicidad y por lo tanto, su inclusión en herramientas similares a *ASCIML* es necesaria. Sin embargo, deben existir opciones que permitan a los usuarios realizar estrategias más controladas como la síntesis y/o directas como la grabación. En este sentido, se puede creer que el diseño ágil y fácil es importante a la hora de crear

bases de datos, al menos en usuarios poco experimentados en el manejo de la tecnología musical. Es crucial no confundir la búsqueda de practicidad con la relevancia de la sección. El diseño de los audios resulta ser un proceso fundamental para los músicos y no puede ser pasado por alto. En una revisión preliminar [45] esto se interpretó como un problema de recolección de los archivos. Sin embargo, los resultados actuales muestran que no se trata solo de recolectar cualquier archivo de audio, sino de elegir cuidadosamente los materiales que se utilizarán para la creación musical. Este proceso es minucioso y está estrechamente relacionado con el acto de composición. Por lo tanto, es necesario que cualquier herramienta como *ASCIML* tenga en cuenta la importancia de esta etapa y brinde opciones para que los usuarios puedan seleccionar y diseñar sus propios materiales de audio de manera efectiva.

Como compositor considero que esto puede deberse a la formación musical que poseen los participantes y su manejo de las escalas temporales y unidades de estructura musical [48, 49]. Esto es particularmente observable con el nivel de control que los compositores ejercen sobre la herramienta al tratar de trabajar con grupos pequeños de archivos de manera iterativa y siempre desde cero. Algo similar es mencionado en [8] respecto al trabajo creativo con pequeños conjuntos de datos. En lo que concierne a este proyecto de investigación, considero que estos resultados deben seguir explorándose, ahora abordando relaciones a largo plazo e incluyendo artistas más experimentados. Lo anterior permitirá tener un mayor entendimiento de la búsqueda de control en creadores con mayor experiencia y si esta es similar a los resultados obtenidos en esta etapa. De ser el caso o no, ver si la herramienta y las posibilidades del manejo de grandes conjuntos de datos inciden en sus procesos y resultados musicales. Dicho esto, una solución a corto plazo que no interfiere con lo anterior sería la integración de aumentación de datos automática.

Finalmente, al hablar de la generación de nuevos materiales con *ASCIML* los resultados nos muestran que los compositores lo utilizan para crear variaciones de materiales ya seleccionados para ser parte de una pieza. Esto refuerza la idea dicha previamente de que los conjuntos de datos son materiales potenciales para una obra y no solamente una excusa o punto de partida generativa.

En este capítulo se presentó la parte experimental en la cual se utilizó *ASCIML*. Esta constó de tres laboratorios llevados a cabo con integrantes de la comunidad musical de UNAM. La pertinencia de las propuestas del prototipo, su relación con el usuario y las distintas estrategias de los mismos, fueron estudiadas a través de una encuesta y consecutivamente un ejercicio de composición acompañado de un escrito y una entrevista libre por videollamada. Los resultados de las experiencias anteriores ilustran que *ASCIML* puede acoplarse a distintas propuestas estéticas y estrategias de composición. Asimismo, que los compositores prefieren trabajar con archivos de audio de características contrastantes y en pequeñas cantidades a la hora de componer. Finalmente, a pesar de las restricciones de reconstrucción del modelo (temporal y de reconstrucción con ruido) los músicos son capaces de adaptarse y utilizar los resultados obtenidos de manera creativa. En el siguiente y último apartado, se compartirán las conclusiones de la presente investigación y vistas a futuro del proyecto.

Conclusiones

Esta investigación comenzó su desarrolló en 2021 con la creación de una herramienta que permitiera a los compositores/creadores sonoros realizar síntesis con redes neuronales de manera interactiva sin la necesidad de escribir código. Para lograr esto se buscaba interconectar el aprendizaje interactivo automático con modelos generativos de inteligencia artificial. Esto fue llevado a cabo utilizando *Python* y *Google Colab* para crear un prototipo que pudiera presentar la parte interactiva con el usuario y al mismo tiempo entrenar modelos de inteligencia artificial con señales de audio. Uno de los primeros obstáculos que buscó resolver esta investigación era la generación automática de bases de datos de manera interactiva. Para esto, los tres métodos propuestos lograron en su conjunción facilitar el ejercicio a distintos usuarios. Este paso fue crucial para realizar entrenamientos que fueran significativos para los usuarios.

Para realizar la síntesis de señal se utilizó una arquitectura propia de un modelo generativo (VAE), que entrena directamente con valores de amplitud de las señales. Esto presenta la ventaja de que el modelo sea capaz de reconstruir y generar formas de onda similares al conjunto de entrenamiento. Esta arquitectura, a diferencia de otras similares como las GANS, presenta la gran ventaja de ser muy compacta, con una cantidad reducida de hiperparámetros y contar solo con un paso de entrenamiento. Este modelo ha resultado exitoso al ser utilizado desde cero o preentrenado, al ser capaz de recrear distintos tipos de sonidos con pocos entrenamientos. Sin embargo, una de las limitaciones actuales de este modelo es la duración corta y fija de señales que puede reconstruir.

Una vez creado este prototipo se diseñaron dos dinámicas para entender el potencial de *ASCIML*. La primera consistió en tres laboratorios con jóvenes músicos de la comunidad UNAM. En esta actividad los participantes diseñaron conjuntos de datos de manera interactiva, dirigieron el proceso de aprendizaje de dos modelos y generaron nuevos audios. Al final del ejercicio, los usuarios respondieron una encuesta en la cual se recopilaba información respecto a la eficacia y utilidad de las tecnologías implementadas, la intersección del pensamiento composicional con *ASCIML* y la inversión temporal en el flujo de trabajo del IML. Un pequeño subconjunto de los participantes realizaron miniaturas de manera individual, lo que permitió probar el prototipo en situaciones más realistas y comprobar su eficacia para el ejercicio creativo.

Las actividades anteriores nos permiten entender que *ASCIML* y herramientas similares requieren una curva de aprendizaje que es independiente de la utilización de código o no. Asimismo, futuras versiones del prototipo deben incluir modelos sin entrenamiento y pre-entrenados para que la herramienta pueda satisfacer distintas búsquedas compositivas. Adicionalmente, deben incluir retroalimentación audiovisual inspirada en herramientas ya conocidas, como los plug-ins en los DAW. Los resultados de esta investigación también arrojan que los compositores diseñan pequeños conjuntos de datos con el fin de tener mayor control de los materiales y la herramienta, ya que de cierta manera ya están componiendo al momento de utilizar *ASCIML*.

Los resultados anteriores nos permiten entender cómo los compositores de nuestra comunidad utilizarán este tipo de herramientas y cómo puede ir mejorando en un futuro cercano. También revelan que estas son pertinentes en la disciplina y pueden insertarse de distintas formas con las debidas adecuaciones, no solamente en el mapeo de superficies de control.

A pesar de los avances logrados en el desarrollo del prototipo de *ASCIML*, todavía hay desafíos pendientes que deben ser superados antes de llevarlo a un nivel de producción utilizable por los artistas. La aplicación de la ventana de reconstrucción del modelo y la mejora de la experiencia del usuario son aspectos clave que deben ser abordados para garantizar el éxito de esta herramienta. Se debe trabajar en reemplazar el modelo actual

por arquitecturas más avanzadas y complejas para lograr audios de alta calidad sin ruido de reconstrucción. Además, se debe continuar trabajando en el diseño de la interfaz de usuario, la documentación y el soporte al usuario para facilitar su uso. Establecer puentes de comunicación entre aplicaciones o tecnologías diferentes también será crucial para lograr la migración de *ASCIML* a otros lenguajes de programación. En definitiva, se debe seguir trabajando para mejorar el prototipo de *ASCIML* con el objetivo de proporcionar una herramienta útil y efectiva para los artistas que buscan explorar nuevas ideas sonoras.

Bibliografía

[1] G. Meza, *Entrelazados: el posthumanismo en obras electroacústicas.* Universidad Nacional Autónoma de México, 2017.

[2] ——, "Transversable," Soundcloud, consultado: 21 de junio del 2023. [Online]. Available: https://soundcloud.com/gerardomezacompositor/transversable

[3] J. A. Fails and D. R. Olsen Jr, "Interactive machine learning," in *Proceedings of the 8th international conference on Intelligent user interfaces*, 2003, pp. 39–45.

[4] S. Amershi, "Designing for effective end-user interaction with machine learning," in *Proceedings of the 24th annual ACM symposium adjunct on User interface software and technology*, 2011, pp. 47–50.

[5] S. Amershi, M. Cakmak, W. B. Knox, and T. Kulesza, "Power to the people: The role of humans in interactive machine learning," *Ai Magazine*, vol. 35, no. 4, pp. 105–120, 2014.

[6] R. Fiebrink and P. R. Cook, "The wekinator: a system for real-time, interactive machine learning in music," in *Proceedings of The Eleventh International Society for Music Information Retrieval Conference (ISMIR 2010)(Utrecht)*, vol. 3. Citeseer, 2010, pp. 2–1.

[7] L. McCallum and M. S. Grierson, "Supporting interactive machine learning approaches to building musical instruments in the browser," in *Proceedings of the International Conference on New Interfaces for Musical Expression.* Birmingham City University Birmingham, UK, 2020, pp. 271–272.

[8] G. Vigliensoni, P. Perry, R. Fiebrink *et al.*, "A small-data mindset for generative ai creative work," 2022.

[9] R. A. E. Asale, "Interacción: Diccionario de la lengua española," 2022. [Online]. Available: https://dle.rae.es/interacci%C3%B3n?m=form

[10] S. Kiousis, "Interactivity: a concept explication," *New media & society*, vol. 4, no. 3, pp. 355–383, 2002.

[11] N. Collins and J. d'Escriván, *The Cambridge companion to electronic music*. Cambridge University Press, 2022.

[12] M. Pichlmair and F. Kayali, "Levels of sound: On the principles of interactivity in music video games." in *DiGRA Conference*, 2007.

[13] R. Rowe, *Interactive music systems: machine listening and composing*. MIT press, 1992.

[14] ——, *Machine musicianship*. MIT press, 2004.

[15] R. Fiebrink, "Wekinator," Página web, consultado: 10 de abril del 2023. [Online]. Available: http://www.wekinator.org/downloads/

[16] F. G. Visi and A. Tanaka, "Towards assisted interactive machine learning: exploring gesture-sound mappings using reinforcement learning," in *Proceedings of the International Conference on Live Interfaces (ICLI)*, 2020.

[17] Google, "Teachable machine," Página web, consultado: 4 de febrero del 2023. [Online]. Available: https://teachablemachine.withgoogle.com/

[18] S. Causevic, "Generative vs discriminative probabilistic graphical models," Aug 2021. [Online]. Available: https://towardsdatascience.com/generative-vs-2528de43a836

[19] D. P. Kingma and M. Welling, "Auto-encoding variational bayes," *arXiv preprint arXiv:1312.6114*, 2013.

[20] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial networks," *Communications of the ACM*, vol. 63, no. 11, pp. 139–144, 2020.

[21] P. Dhariwal, H. Jun, C. Payne, J. W. Kim, A. Radford, and I. Sutskever, "Jukebox: A generative model for music," *arXiv preprint arXiv:2005.00341*, 2020.

[22] P. Esling, N. Masuda, A. Bardet, R. Despres, and A. Chemla-Romeu-Santos, "Flow synthesizer: Universal audio synthesizer control with normalizing flows," *Applied Sciences*, vol. 10, no. 1, p. 302, 2019.

[23] S. Kim, S.-g. Lee, J. Song, J. Kim, and S. Yoon, "Flowavenet: A generative flow for raw audio," *arXiv preprint arXiv:1811.02155*, 2018.

[24] A. Huang and R. Wu, "Deep learning for music," *arXiv preprint arXiv:1606.04930*, 2016.

[25] S. Vasquez and M. Lewis, "Melnet: A generative model for audio in the frequency domain," *arXiv preprint arXiv:1906.01083*, 2019.

[26] A. v. d. Oord, S. Dieleman, H. Zen, K. Simonyan, O. Vinyals, A. Graves, N. Kalchbrenner, A. Senior, and K. Kavukcuoglu, "Wavenet: A generative model for raw audio," *arXiv preprint arXiv:1609.03499*, 2016.

[27] A. Van den Oord, N. Kalchbrenner, L. Espeholt, O. Vinyals, A. Graves *et al.*, "Conditional image generation with pixelcnn decoders," *Advances in neural information processing systems*, vol. 29, 2016.

[28] J. Engel, C. Resnick, A. Roberts, S. Dieleman, M. Norouzi, D. Eck, and K. Simonyan, "Neural audio synthesis of musical notes with wavenet autoencoders," in *International Conference on Machine Learning*. PMLR, 2017, pp. 1068–1077.

[29] A. Caillon and P. Esling, "Rave: A variational autoencoder for fast and high-quality neural audio synthesis," *arXiv preprint arXiv:2111.05011*, 2021.

[30] J. Rocca, "Understanding variational autoencoders (vaes)–building, step by step, the reasoning that leads to vaes (2019)."

[31] S. Min, B. Lee, and S. Yoon, "Deep learning in bioinformatics," *Briefings in bioinformatics*, vol. 18, no. 5, pp. 851–869, 2017.

[32] Jamesdec, "Using machine learning and your face to control scratch," Página web, consultado: 15 de enero del 2023. [Online]. Available: https://www.instructables.com/Using-Machine-Learning-and-YOUR-FACE-to-Control-Sc/

[33] E. Law, K. West, M. I. Mandel, M. Bay, and J. S. Downie, "Evaluation of algorithms using games: The case of music tagging." in *ISMIR*. Citeseer, 2009, pp. 387–392.

[34] J. F. Gemmeke, D. P. Ellis, D. Freedman, A. Jansen, W. Lawrence, R. C. Moore, M. Plakal, and M. Ritter, "Audio set: An ontology and human-labeled dataset for audio events," in *2017 IEEE international conference on acoustics, speech and signal processing (ICASSP)*. IEEE, 2017, pp. 776–780.

[35] R. M. Bittner, M. Fuentes, D. Rubinstein, A. Jansson, K. Choi, and T. Kell, "mirdata: Software for reproducible usage of datasets." in *ISMIR*, 2019, pp. 99–106.

[36] E. Fonseca, J. Pons Puig, X. Favory, F. Font Corbera, D. Bogdanov, A. Ferraro, S. Oramas, A. Porter, and X. Serra, "Freesound datasets: a platform for the creation of open audio datasets," in *Hu X, Cunningham SJ, Turnbull D, Duan Z, editors. Proceedings of the 18th ISMIR Conference; 2017 oct 23-27; Suzhou, China.[Canada]: International Society for Music Information Retrieval; 2017. p. 486-93.* International Society for Music Information Retrieval (ISMIR), 2017.

[37] H. Pedroza, G. Meza, and I. R. Roman, "Egfxset: Electric guitar tones processed through real effects of distortion, modulation, delay and reverb," in *Ismir 2022 Hybrid Conference*, 2022.

[38] J. M. Chowning, "The synthesis of complex audio spectra by means of frequency modulation," *Computer Music Journal*, pp. 46–54, 1977.

[39] K. Tatar, D. Bisig, and P. Pasquier, "Latent timbre synthesis: Audio-based variational auto-encoders for music composition and sound design applications," *Neural Computing and Applications*, vol. 33, pp. 67–84, 2021.

[40] E. Bisong and E. Bisong, "Google colaboratory," *Building machine learning and deep learning models on google cloud platform: a comprehensive guide for beginners*, pp. 59–64, 2019.

[41] C. R. Harris, K. J. Millman, S. J. Van Der Walt, R. Gommers, P. Virtanen, D. Cournapeau, E. Wieser, J. Taylor, S. Berg, N. J. Smith *et al.*, "Array programming with numpy," *Nature*, vol. 585, no. 7825, pp. 357–362, 2020.

[42] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg *et al.*, "Scikit-learn: Machine learning in python," *the Journal of machine Learning research*, vol. 12, pp. 2825–2830, 2011.

[43] P. Virtanen, R. Gommers, T. E. Oliphant, M. Haberland, T. Reddy, D. Cournapeau, E. Burovski, P. Peterson, W. Weckesser, J. Bright *et al.*, "Scipy 1.0: fundamental algorithms for scientific computing in python," *Nature methods*, vol. 17, no. 3, pp. 261–272, 2020.

[44] F. Pérez and B. E. Granger, "Ipython: a system for interactive scientific computing," *Computing in science & engineering*, vol. 9, no. 3, pp. 21–29, 2007.

[45] G. Meza, "Exploring the potential of interactive machine learning for sound generation: A preliminary study with sound artists," in *NIME 2023*, 2023.

[46] "Okachihuali, by hexorcismos," 2021. [Online]. Available: https://hexorcismos.bandcamp.com/album/--2

[47] D. J. Johnston., "Rerites," 2019. [Online]. Available: https://www.anteism.com/shop/rerites-david-jhave-johnston

[48] C. Roads, *Microsound*. The MIT Press, 2004.

[49] J. Tenney, *From Scratch: Writings in Music Theory.* University of Illinois Press, 2014.

Apéndice

```
##encoder
latent_dim = 128
original_dim = (16000, 1, 1)
encoder_inputs = keras.Input(original_dim)
x = layers.Conv2D(64, (66,1), activation="relu", padding="same",strides=(4,1))(encoder_inputs)
x = layers.Conv2D(128,(66,1), activation="relu", padding="same",strides=(4,1))(x)
x = layers.Conv2D(256,(66,1), activation="relu", padding="same",strides=(4,1))(x)
latent = layers.Conv2D(512,(66,1), activation="relu", padding="same",strides=(3,1))(x)
x = layers.Flatten()(x)
x = layers.Dense(256, activation="relu")(x)
z_mean = layers.Dense(units=latent_dim, name='Z-Mean')(x)
z_log_sigma = layers.Dense(units=latent_dim, name='Z-Log-Sigma')(x)
z_regular = tf.keras.layers.Concatenate(activity_regularizer= KLDivergenceRegularizer(
optimizer.iterations, warm_up_iters,latent_dim))([z_mean,z_log_sigma])
z = Sampling()([z_mean, z_log_sigma])
#z = Sampling()(z_regular)
vanilla_encoder = keras.Model(encoder_inputs, [z_mean, z_log_sigma, z], name="encoder_vae")
#vae_encoder.summary()

```

Anexo 1. Codificador

```
###decoder
latent_inputs = keras.Input(shape=(latent_dim,))
x = layers.Dense(250 * 1 * 256, activation="relu")(latent_inputs)
x = layers.Reshape((250, 1, 256))(x)
#x = layers.Conv2DTranspose(512,(4,1), activation="relu",padding= "same",strides= (3,1))(x)
x = layers.Conv2DTranspose(128,(66,1), activation="relu",padding= "same",strides= (4,1))(x)
x = layers.Conv2DTranspose(64,(66,1), activation="relu",padding= "same",strides= (4,1))(x)
#x = layers.Conv2DTranspose(64,(16,1), activation="relu",padding= "same",strides=(4,1))(x)
decoder_outputs = layers.Conv2DTranspose(1, (66, 1), activation="tanh", padding="same", strides = (4,1))(x)
vanilla_decoder = keras.Model(latent_inputs, decoder_outputs, name="decoder_vae")
#vae_decoder.summary()

```

Anexo 2. Decodificador

```
if dropdown == 'Subir archivos':
  button = widgets.Button(description= 'SUBIR')
  text = widgets.Text(value='Synth',
                      placeholder='Type something',
                      description='group_id:',
                      disabled=False)
  output = widgets.Output()
  rec_gui2 = HBox(children = [button, text])

  def win_analysis_audio(b):
    # Display the message within the output widget.
    with output:
      sr16k = 16000
      secs = 1
      uploaded = files.upload()
      for idx, track in enumerate(uploaded.keys()):
        print(track)
        if track.endswith("wav") or track.endswith("mp3"):
          SR = librosa.get_samplerate(track)
          y, fs = librosa.load(track, SR)
          data = librosa.resample(y, orig_sr= SR, target_sr= sr16k)
          cuente = 0
          for idx_2, window in enumerate(range(0, data.shape[0], sr16k)):

              sx = data[window: (window + sr16k)]

              if sx.shape != sr16k:
                sx = zero_padding(sx,sr16k)

              sx = sx * envelope(1,1,1,[0.05,0.1,0.8,0.05],sr16k)
              ##features (tenemos que arreglar el fft apara que sean valores de amp)
              fft = librosa.amplitude_to_db(np.abs(np.fft.fft(sx)))
              f0 = np.mean(librosa.yin(sx, 30, 2093, sr= sr16k, frame_length= 2048, win_length= 1024))
              sc = np.mean(librosa.feature.spectral_centroid(sx, sr =sr16k))
              rms = librosa.feature.rms(sx)[0]
              etiquetas.append(text.value)
              AUDIO_LOAD.append({'gid': text.value,
                                 'id': text.value + str(cuente),
                                 'trackname': text.value + str(cuente)+'.wav',
                                 'data': sx,
                                 'fft': fft,
                                 'f0': f0,
                                 'sc': sc,
                                 'rms': rms})
              cuente += 1

  button.on_click(win_analysis_audio)
  display(rec_gui2, output)

```

Anexo 3. Carga de archivos y funcion de ventaneo.

```
fft = librosa.amplitude_to_db(np.abs(np.fft.fft(sx)))
              f0 = np.mean(librosa.yin(sx, 30, 2093, sr= sr16k, frame_length= 2048, win_length= 1024))
              sc = np.mean(librosa.feature.spectral_centroid(sx, sr =sr16k))
              rms = librosa.feature.rms(sx)[0]

```

Anexo 4. Función de extracción de características

ASCIML Lab

¡Hola! Muchas gracias por tomarte el tiempo de participar en este estudio. Las preguntas que vas a contestar el día de hoy están basadas en tu experiencia personal con el prototipo ASCIML. Tus respuestas nos ayudarán a entender como se puede hacer más accesible el entrenamiento y desarrollo de modelo de inteligencia artificial a músicos para crear.

1. Fecha de laboratorio: DD/MM/AAAA

2. Escuela de procedencia:

☐ ENES
☐ FAM

3. ¿Tienes alguna experiencia previa entrenando modelos de inteligencia artificial?

☐ Si
☐ No

Sección 2 /4 Base de datos

4. ¿Qué tan eficiente consideras las formas de generar el conjunto de datos? (Eficiente considerado como bien presentado, útil y claro)

	Muy eficiente	**Eficiente**	**Poco eficiente**
Subir archivos			
Sintetizar audios			
Grabar con el micrófono			

5. ¿Qué tipo de criterio utilizaste para generar tu base de datos?

(x)	Interválico
	Tímbrico
	Evolución temporal (envolventes)
	Ninguno en particular

6. Respecto al criterio de creación de la base da datos, ¿Qué tan significativa consideras la información que provee cada tipo de visualización?

	Muy significativa	**Significativa**	**Poco significativa**
Tablas			
PCA			
F0			
Histograma			
Centroide espectral			
Envelopegram			

7. ¿Existe algún otro tipo de visualización que te hubiera ayudado en esta tarea?

8. ¿Cuánto tiempo consideras que tomó generar la base de datos?

	menos de 10 minutos
	10 min- 20 min
	20- 30 min
	más de 30 min

Sección 3/4 Entrenamiento del modelo

9. ¿Consideras la retroalimentación audiovisual de la herramienta brinda la suficiente información respecto al aprendizaje del modelo?

10. ¿Cuál apoya más a la tarea?

	Formas de onda
	Gráfica lineal
	Audio

11. Utilizando un modelo nuevo ¿En cuánto tiempo consideras que comenzó a reconstruir la señal? Reconstruir la señal considerado como que puedas identificar su parecido a la señal original.

- [ ] menos de 10 minutos
- [ ] 10 min - 20min
- [ ] 20 - 30 min
- [ ] más de 30 min

12. Utilizando un modelo pre-entrenado, ¿En cuánto tiempo consideras que comenzó a reconstruir la señal?

- [ ] menos de 10 minutos
- [ ] 10 min - 20min
- [ ] 20 - 30 min
- [ ] más de 30 min

13. ¿Cuál de los dos modelos consideras que funcionó mejor en aprender de la base de datos?

- [ ] Nuevo modelo
- [ ] Pre-entrenado
- [ ] Ambos

Sección 4/4 Generación

14. ¿Qué tipo de criterio utilizaste para interpolar entre sonidos?

- [ ] Ninguno en particular
- [ ] Interválico
- [ ] Tímbrico
- [ ] Evolución temporal (envolventes)
- [ ] Navegación del espacio latente

15. Tomando en cuenta los resultados obtenidos, ¿en cuánto tiempo consideras que comenzaste a obtener resultados sonoros interesantes? Reconstruir la señal considerado como que puedas identificar su parecido a la señal original.

- [ ] menos de 10 minutos
- [ ] 10 min- 20 min
- [ ] 20- 30 min
- [ ] más de 30 min

Anexo 5: Encuesta de los laboratorios.