Proyecto: Clasificación de audio

31508811-1 Martiñón Luna Jonathan José,
41800471-9 Tapia López José de Jesús

***Resumen*—Resulta de gran interés el conteo de personas en un determinado ambiente, dígase restaurante, transporte, etc. Tener un conocimiento del número de personas a las que interesa un lugar, explotar los horarios de mayor afluencia o simplemente conocer los sectores (Masculino o femenino) que atraemos, permite llevar una mejor administración en cuanto a publicidad, trabajadores, etc. El presente proyecto pretende usar las herramientas, técnicas y modelos de Aprendizaje Profundo para clasificar el número de hablantes (con 1, 2 o 3 como posibles cantidades) a partir de señales de audio con una duración de un segundo y una frecuencia de muestreo de 16000 hz, comparando 2 métodos diferentes:**

- **Redes neuronales convolucionales 1D.**
- **Redes neuronales convolucionales y 2D.**

I. Objetivos

- **Sensar**: A través de un audio, poder identificar la cantidad de personas que se encuentren en ciertas áreas.
- **Simplificar**: Dejar de lado el uso de sensores o contadores manuales y dejar el trabajo a un simple micrófono. Queremos resolver este problema usando herramientas de Aprendizaje Profundo.

II. Introducción

Existen Situaciones a lo largo de la vida cotidiana, en las que resulta de vital importancia llevar un manejo de la población que se encuentra en dicho lugar. Podemos orientar el ejemplo a la situación actualmente vivida; la pandemia. Es bien sabido que hay lugares en los que la cantidad de personas es limitada, por consecuente, al haber una disminución en el público, existe también una disminución en los ingresos, lo que es a su vez una limitante al momento de contratar personal. Trabajadores limitados (reducidos) significa un mayor trabajo para aquellos que han conseguido mantenerse. Si tomamos en cuenta lo anterior mencionado, podremos intuir que el conteo de personas (a pesar de ser necesario) sería una tarea compleja o simplemente no prioritaria para algunos.

Para nosotros, conocer la cantidad de personas tiene varias ventajas:

- Conocimiento de horarios con mayor y menor cantidad de personas.
- Reconocimiento sobre la población que se alcanza.
- Información sobre el punto en que se alcanza el límite de afluencia (Hablando de la pandemia).
- Al conocer sectores alcanzados, se puede saber qué acciones implementar para *capturar* el sector que se pierde.

Claro está que también se deben tener en cuenta algunos aspectos relacionados con el proyecto, es decir, podemos pensar en la situación más común: *El uso de los datos*. Y es que si bien es cierto, el proyecto tiene como objetivo principal el realizar un conteo de personas visto como clasificación, pero puede resultar inquietante la situación de tener un micrófono que está escuchando las conversaciones de las personas (desde cierto punto generando una invasión a la privacidad).

No obstante, creemos que esta tarea puede brindar facilidades de uso, ya que podría resultar menos complicado y más eficiente trabajar con un micrófono y un modelo; en lugar de instalar sensores o hacer el conteo manualmente.

El presente proyecto busca clasificar el número de hablantes (con 1, 2 o 3 como posibles cantidades) a partir de señales de audio con una duración de un segundo y una frecuencia de muestreo de 16000 hz, comparando 2 métodos diferentes:

- Redes neuronales convolucionales 1D.
- Redes neuronales convolucionales 2D.

Entonces, vamos a resolver este problema como si fuera de clasificación (el número de hablantes solo puede tomar 1 de 3 posibles valores; no son datos 'continuos').

II-A. Ambiente

El presente proyecto fue desarrollado a través de *Google Colab*, mediante el lenguaje de programación: *Python*.

II-B. Métricas

II-B1. Decibeles: Unidad logarítmica que compara dos magnitudes de un mismo fenómeno. No tienen unidades. Se utiliza para comparar voltajes, potencias, presiones sonoras, etc. [1]

Existen diversas formas de calcularlo [2], por ejemplo:

- Para watts
$$db = 10\ log_{10}\left(\frac{P_1}{P_2}\right) \tag{1}$$
- Para Voltajes.
$$db = 20\ log_{10}\left(\frac{V_1}{V_2}\right) \tag{2}$$
- Para intensidad.
$$db = 20\ log_{10}\left(\frac{I_1}{I_2}\right) \tag{3}$$

II-B2. Espectrogramas: Sirve para detectar anomalías intermitentes. Es una gráfica que representa una señal a través del tiempo. [3]

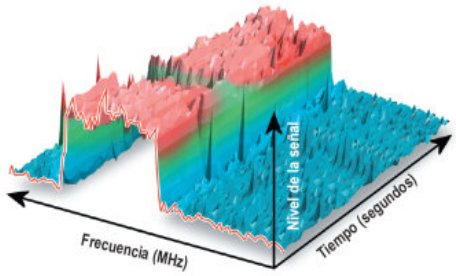

Figura 1. Imagen recuperada de: https://www.promax.es/esp/noticias/158/Espectrograma/

A continuación (Figura 2), podemos apreciar un histograma generado a partir de nuestro conjunto de validación con 3 hablantes.

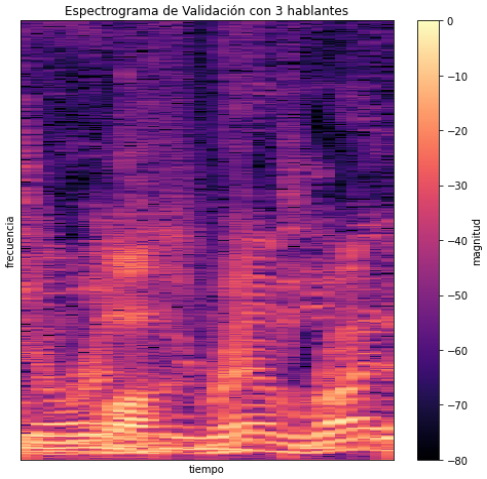

Figura 2. Ejemplo de espectrograma

II-C. Datos

II-C1. Generación de los datos: Lo primero que realizamos fue la extracción de datos. Estos fueron recopilados a partir de la siguiente página web: https://www.openslr.org/12. Este corpus es el de *LibriSpeech ASR* y contiene aproximadamente 1000 horas de discursos leídos en inglés de 16 kHz, preparado por Vassil Panayotov con la asistencia de Daniel Povey. Los datos se derivan de audiolibros leídos del proyecto *LibriVox* y se han segmentado y alineado cuidadosamente.

A continuación ejemplificamos los atributos de un audio en el corpus:

- **WaveForm** (tensor): [[0.0003 ... 0.0016]]
- **Sample Rate** (int): 16000
- **Utterance** (string): ”He Hoped...”
- **Speaker ID** (int): 1089
- **Chapter ID** (int): 134686
- **Utterance ID** (int): 0

Además, estos también cuentan con metadatos, y por ello también mostramos en seguida un ejemplo de la información contenida:

- **ID** (int): 14
- **Sex** (bool): F
- **Subset** (string): ”train-clean-360”
- **Minutes** (float): 25.03
- **Name** (string): Kristin LeMoine

Aquí apreciamos que estos audios tienen una duración variada. Igualmente, podemos notar que solamente son de un solo hablante. Por ello, debido a que queremos generar audios con más de un hablante, empezamos a acotar nuestra tarea limitando audios con hasta tres hablantes. Esto quiere decir que nuestros datos los generamos al unir aleatoriamente audios para así tenerlos con 1, 2 o 3 hablantes; recabando información del sexo de el, la, los o las que participan en estos y de tal forma que todos tuvieran tanto un *sample rate* (frecuencia de muestreo; cuántas veces por segundo se muestrea el audio) de 16,000 hz como una duración de un segundo, el cual fue también seleccionado al azar del audio original. Así, generamos datos de entrenamiento, validación y prueba; en los cuales se generaron 2 conjuntos de datos:

1. Metadatos: Contienen la información referente al audio:

WaveForm	Speakers	Speakers Sex	F	M
[[0.0001,...,0.003]]	3	MMF	1	2

Tabla I
REGISTRO EJEMPLO DE METADATOS

Donde:

- **Waveform** (array): Forma de onda del audio mezclado.
- **Speakers** (int): Cantidad de hablantes en el audio: 1, 2 o 3.
- **Speakers Sex** (string): Distribución en Masculino y Femenino de los hablantes
- **F** (int): Cantidad de hablantes femeninos en el audio.
- **M** (int): Cantidad de hablantes masculinos en el audio.

2. Audios: Contienen únicamente el audio de los n hablantes ($n \in \{1, 2, 3\}$) en formato *.wav*. Van nombrados con el siguiente formato: Audio_**Conjunto**k.wav; donde **Conjunto** puede tomar el valor de **Train**, **Validation** o **Test**; $k \in \{0, 1, \ldots, 99999\}$ para los datos de entrenamiento y $k \in \{0, 1, \ldots, 9999\}$ para los datos de validación y prueba. Por ejemplo: Audio_**Train**12.wav. Para conocer los metadatos del audio ejemplificado, basta con seleccionar el conjunto perteneciente a entrenamiento con el índice 12 (El número de audio va de 0 a $n - 1$ para que coincidan directamente audio y metadatos).

Esto también nos dice que generamos 100,000 datos de entrenamiento, 10,0000 datos de validación y 10,000 datos de prueba; con un peso de 3.5 GB, 364 MB y 364 MB respectivamente; dando un total de 120,000 audios.

Por lo que concierne a la columna *Speakers Sex* debemos mencionar que podemos obtener las siguientes combinaciones (M: Masculino; F: Femenino):

- M
- F
- MM
- FF
- MF
- MMF
- FFM
- MMM
- FFF

Esto nos dice que realmente no importa el orden de los sexos al seleccionar los audios.

II-C2. Análisis exploratorio de los datos: Para efectos prácticos presentaremos la distribución del número de hablantes en los audios generados, así como la proporción del sexo por cada conjunto de datos.

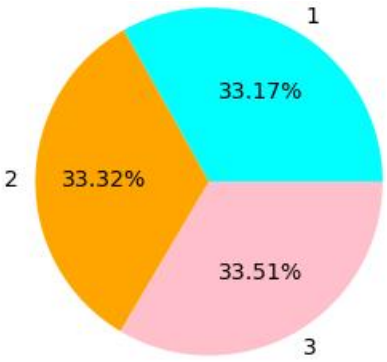

Figura 3. Distribución del número de hablantes: **Entrenamiento**

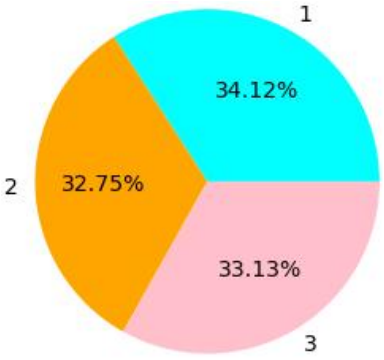

Figura 4. Distribución del número de hablantes: **Validación**

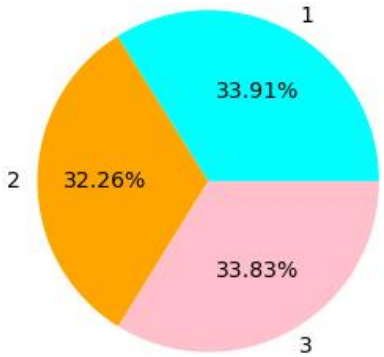

Figura 5. Distribución del número de hablantes: **Prueba**

- **Entrenamiento**:

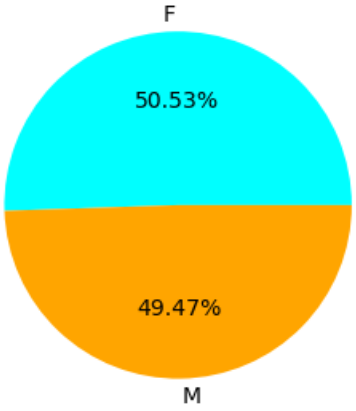

Figura 6. Distribución con 1 hablante, para los datos de Entrenamiento.

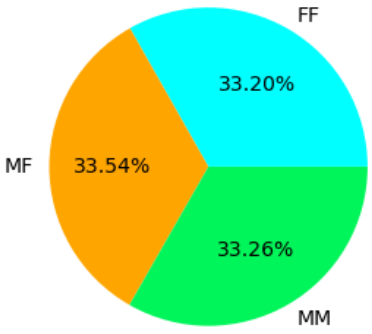

Figura 7. Distribución con 2 hablantes, para los datos de Entrenamiento.

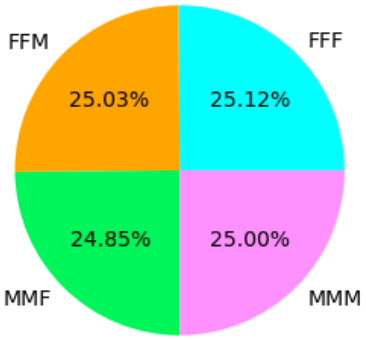

Figura 8. Distribución con 3 hablantes, para los datos de Entrenamiento.

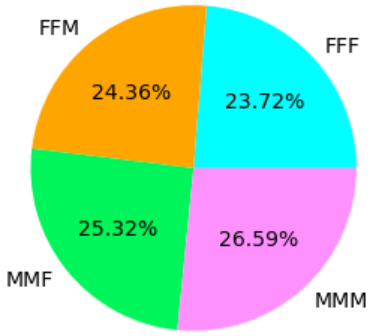

Figura 11. Distribución con 3 hablantes, para los datos de Validación.

- **Validación**:

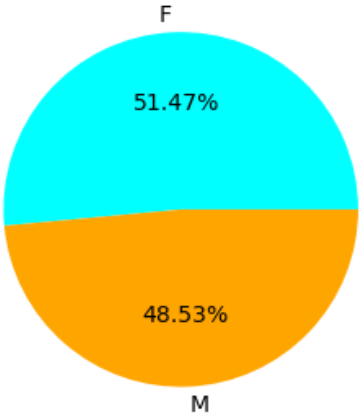

Figura 9. Distribución con 1 hablante, para los datos de Validación.

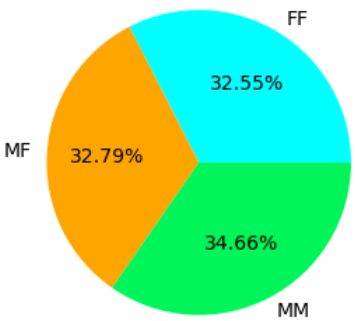

Figura 10. Distribución con 2 hablantes, para los datos de Validación.

- **Prueba**:

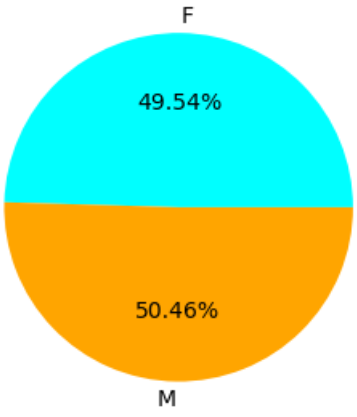

Figura 12. Distribución con 1 hablante, para los datos de Prueba.

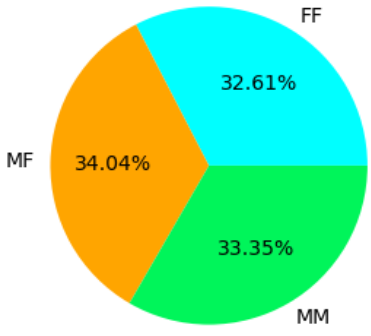

Figura 13. Distribución con 2 hablantes, para los datos de Prueba.

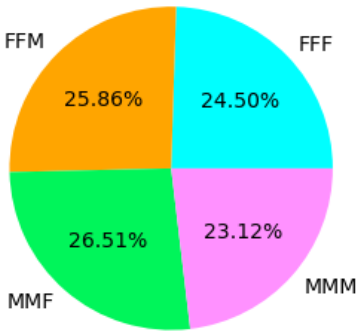

Figura 14. Distribución con 3 hablantes, para los datos de Prueba.

Tal como es posible apreciar de las figuras anteriores (3-14) los datos han sido balanceados lo más posible, para evitar sesgos durante el desarrollo del proyecto.

Es importante mencionar que para aquellos audios dónde existe más de un hablante, se promediaron los valores entre los n audios seleccionados. Digamos:

$$audio_{nuevo} = \frac{audio_1 + ... + audio_n}{n}, \quad n \in \{1, 2, 3\} \quad (4)$$

III. Desarrollo

El presente proyecto compara el rendimiento entre arquitecturas usando redes neuronales convolucionales 1D (usando la forma de onda) y redes nueronales convolucionales 2D (usando espectrogramas); para ambos casos modelamos la tarea como clasificación. Entonces, de nuestro conjunto de datos correspondiente a los Metadatos, lo primero que hicimos fue utilizar el atributo **Speakers** (que indicaba el número de hablantes del audio; 1, 2 o 3) para obtener un nuevo atributo nombrado **Speakers_Class**, de tal forma que las categorías empezaran desde cero. Así, en los próximos modelos que vamos a presentar, tenemos las siguientes clases:

- 0: Audios con un hablante.
- 1: Audios con dos hablantes.
- 2: Audios con tres hablantes.

Esto quiere decir que dicha variable es la que usamos como respuesta, mientras que la variable explicativa es la respectiva a las formas de audio en formato *.wav* (ya sea como audio o visto como imagen, dependiendo el caso).

En todos las arquitecturas (intentos) que llevamos a cabo, el ciclo de entrenamiento fue el mismo y lo describiremos en seguida.

Creamos el cargador de datos de tal forma que trabajáramos con *DataLoaders*. Es importante mencionar que todos los experimentos (en los conjuntos de entrenamiento, validación y prueba) los hicimos con un tamaño de lote de 32.

Luego, entrenamos cada época; por cada lote: calculamos los logits y la pérdida (entropía cruzada al ser una tarea de clasificación) entre los logits y los valores reales, vaciamos los gradientes, retropropagamos y actualizamos parámetros. Luego, ponemos el modelo en modo de evaluación (para entrenamiento y validación); validación de la época con todos los lotes: calculamos los logits por medio del modelo, calculamos los puntajes (softmax), calculamos las clases acorde a los argumentos máximos de las probabilidades y obtenemos la pérdida y exactitud. Como optimizador usamos *Adam* con una tasa de aprendizaje de 0.0001. El número de épocas lo establecimos a 10.

Por lo tanto, este proceso es el mismo para todas las arquitecturas, es decir, lo único diferente en cada experimentación son las arquitecturas como tal. Nuestros análisis se van a basar en comparaciones relacionadas a tiempo de ejecución, eficacia, número de parámetros en cada arquitectura, pérdida y exactitud, etc. Cabe destacar que en todos los casos se utilizó la GPU gratuita que provee *colab*.

III-A. Redes Neuronales Convoluciones 1D

Recordemos que en la tarea de clasificación de audio, en particular en redes neuronales, existen dos aproximaciones de cómo resolver esto, en la que una de ellas tiene que ver con plantear un modelo de red neuronal para aprender de esa forma de onda; esto se conoce como ***redes neuronales para audio en crudo***, y en esta sección se hará dicha aproximación.

En [4] proponen de manera general redes neuronales convolucionales basada en convoluciones 1D; lo que estudian en el artículo es cómo explorar diferentes parámetros de la arquitectura en la red neuronal: tamaño de filtros, número de filtros, número de capas; para poder hacer uso de redes neuronales profundas.

Para construir redes muy profundas, utilizan un campo receptivo de 3, muy pequeño para todas las capas convolucionales 1D excepto la primera. Esto reduce el número de parámetros en cada capa y controla los tamaños del modelo y el costo de cálculo a medida que se profundiza. Después de las dos primeras capas, la reducción de la resolución se complementa con la duplicación del número de mapas de características.

El funcionamiento de los modelos se presenta en la siguiente tabla e imagen:

M3 (0.2M)	M5 (0.5M)	M11 (1.8M)	M18 (3.7M)	M34-res (4M)
Input: 32000x1 time-domain waveform				
[80/4, 256]	[80/4, 128]	[80/4, 64]	[80/4, 64]	[80/4, 48]
Maxpool: 4x1 (output: 2000 × n)				
[3, 256]	[3, 128]	[3, 64] × 2	[3, 64] × 4	$\begin{bmatrix} 3, 48 \\ 3, 48 \end{bmatrix} \times 3$
Maxpool: 4x1 (output: 500×n)				
	[3, 256]	[3, 128] × 2	[3, 128] × 4	$\begin{bmatrix} 3, 96 \\ 3, 96 \end{bmatrix} \times 4$
	Maxpool: 4x1 (output: 125 × n)			
	[3, 512]	[3, 256] × 3	[3, 256] × 4	$\begin{bmatrix} 3, 192 \\ 3, 192 \end{bmatrix} \times 6$
	Maxpool: 4x1 (output: 32 × n)			
		[3, 512] × 2	[3, 512] × 4	$\begin{bmatrix} 3, 384 \\ 3, 384 \end{bmatrix} \times 3$
Global average pooling (output: 1 × n)				
Softmax				

Figura 15. Arquitecturas de la red totalmente convolucional propuesta para entradas con forma de onda. La Mk, con $k = \{3, 5, 11, 18, 34-res\}$ denota k capas de peso

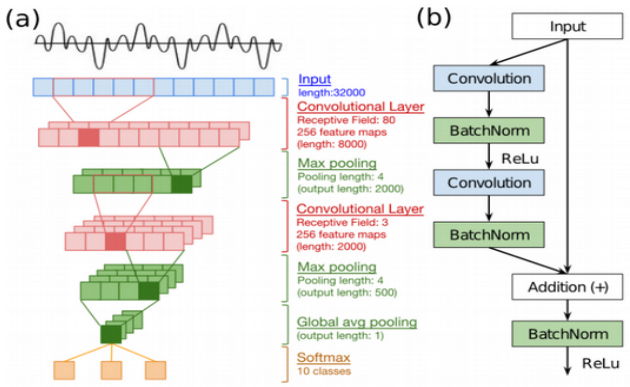

Figura 16. (a) La arquitectura del modelo de M3 (Tabla anterior). El audio de entrada está representado por un único mapa de características (o canal). En cada capa convolucional, un mapa de características codifica el nivel de actividad del núcleo convolucional asociado. (b) bloque residual (resblock) utilizado en M34-res. Un resblock consta de dos capas convolucionales.

Lo que tiene es su secuencia de tiempo (bloques azules en (b) de la figura anterior). Ellos plantean de forma general un bloque convolucional que va a estar formado por una capa de convolución, seguido por una capa de normalización, después una función de activación y después una capa de submuestreo para reducir la dimensión temporal.

La primera capa de convolución de todas las arquitecturas tiene un campo receptivo muy amplio; es una convolución de 1D con un tamaño del filtro de 80. En el artículo también comentan que si se usara exclusivamente un campo receptivo pequeño para todas las capas convolucionales, el modelo necesitaría muchas capas para poder extraer características de alto nivel, lo que podría ser costoso desde el punto de vista computacional. Esto también está relacionado con tratar de emular con algunas técnicas de procesamiento de audio de un filtro de paso de banda; y se refiere a que se tiene un filtro de dicho tamaño pues lo que buscan es acotar las frecuencias que nosotros estamos interesados de procesar; pues el sistema de audición tiene ciertas frecuencias que puede escuchar; y entonces eliminan o atenuan todas las frecuencias que estén del campo receptivo en temporalidad de la primera convolución. Esta es la única que tiene una amplitud de 80 y posteriormente usan convoluciones de 3.

Tampoco está de más mencionar que la normalización por lote sirve como punto de control para evitar que se nos disparen las activaciones (lo que alivia el problema de desvanecimiento del gradiente), la ReLU es la activación en sí (se usa esta para un menor costo de cálculo) y después tendríamos la capa de submuestreo para ir reduciendo poco a poco la dimensión. Finalmente, el tensor va a ir reduciendo en temporalidad y va a ir aumentando el número de canales, simplemente promediando la temporalidad, quitando las dimensiones que nos estorban y termina haciendo la clasificación.

Ahora bien, para los fines de este proyecto, vamos a echar a andar las arquitecturas M3, M5, M11 y M18 para resolver nuestro problema. Primero, recordemos la representación que estamos usando de los datos; un lote tiene 32 ejemplos, 1 ejemplo tiene un canal representado por un vector de 16,000 muestras que son las amplitudes de la forma de onda. Esto es lo que vamos a usar como representación de audio para poder procesarla.

Por ello, nosotros tomamos una convolución en 1D que va a tener cierto tamaño o campo receptivo y así ventanear esa convolución a lo largo del audio en la temporalidad; que es lo equivalente a las imágenes, pero en las convoluciones con imágenes lo que se hace es ventanear en ancho y en alto. Es decir, en nuestro caso vamos a ventanear simplemente una dimensión que sería la temporalidad; y esta última, por como generamos nuestros datos, está acotada a 16000 en todos los ejemplos.

Posteriormente, vamos a aplicar esos bloques convolucionales (convolución, normalización, una función de activación y un submuestreo para ir reduciendo la temporalidad). La intención es irnos haciendo más profundos en canales pero ir reduciendo la temporalidad.

Es importante mencionar las adaptaciones de dichas arquitecturas con nuestro proyecto; nosotros consideramos los saltos (*stride*) en la primera capa de 16 (en el artículo lo proponen de 4), el número de canales producidos por la primera convolución nosotros lo consideramos de 32 en los 4 modelos (en el artículo lo proponen de 128 para la M3 y M5, y de 64 en la M11 y M18), mientras que a partir del segundo bloque se indica que el *kernel size* sea de 3 y este lo dejamos de 80 en la primera capa convolucional (estas dos últimas cosas sí están tal cual como en el artículo).

III-B. Redes Neuronales Convolucionales 2D

En la sección anterior se ha desarrollado la idea sobre las formas de audio, lo que requería ser tratado como un problema cuya solución sería analizada a través de redes convolucionales con 1D. En esta sección, se ha optado por llevar a cabo un análisis de desempeño, generando redes

convolucionales 2D.

Dicho lo anterior, procedemos inicialmente a presentar un ejemplo de forma de onda para 1D.

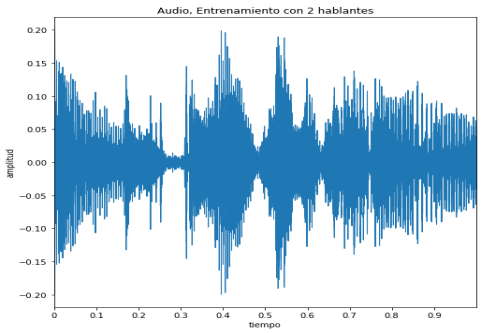

Figura 17. Forma de onda extraída del conjunto de entrenamiento, con 2 hablantes.

Ahora bien, tal cual se ha mencionado, procederemos a calcular y mostrar el espectrograma presentado en la Figura 18.

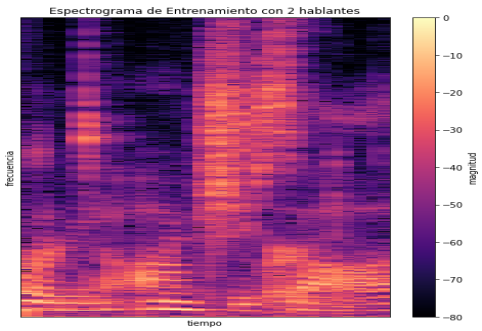

Figura 18. Espectrograma para forma de onda presentado en la figura 17 (con 2 hablantes)

Como es bien sabido, aprovecharemos la librería de *Pytorch* para el armado de redes. Para ser más específico, en el desarrollo, a diferencia del modelo anteriormente mencionado, comenzamos por extraer los audios en lotes de 32 cada uno. Una vez extraídos, recuperamos las características que resultan de nuestro interés, es decir, nuestros valores en X y en Y. En este punto todo resulta semejante, la diferencia radica precisamente en que a partir del audio extraído se realiza una conversión a espectrograma y posteriormente una conversión de amplitud a decibeles.

Resulta de suma importancia recalcar que se desarrollaron 3 diferentes arquitecturas, que podemos resumir en:

- 4 Bloques convolucionales.
- 2 Bloques convolucionales.
- 1 Bloque convolucional.

Realmente las arquitecturas estuvieron basadas en ejercicios realizados a través del curso: *Introducción al Aprendizaje Profundo* de la *Licenciatura en Ciencia de datos*; *Semestre 2021-II*.

Inicialmente, en un *simulacro de ejecución* se optó por entrenar con 4 bloques, todo aparentó salir sin problemas y se procedió a repetir el experimento, desgraciadamente, dicha repetición tuvo algún fallo (Desconocido), puesto que al transcurrir 30 minutos, el modelo no lograba terminar de entrenar ni una sola época o simplemente consumía toda la memoria RAM proporcionada por *colab*.

Lo anterior impulsó la idea de desarrollar una arquitectura que trabajara la mitad, y de esa forma procurar no consumir todos los recursos presentes en la plataforma y obtener resultados interpretables. Para efectos prácticos, mostramos la arquitectura en la Figura 19.

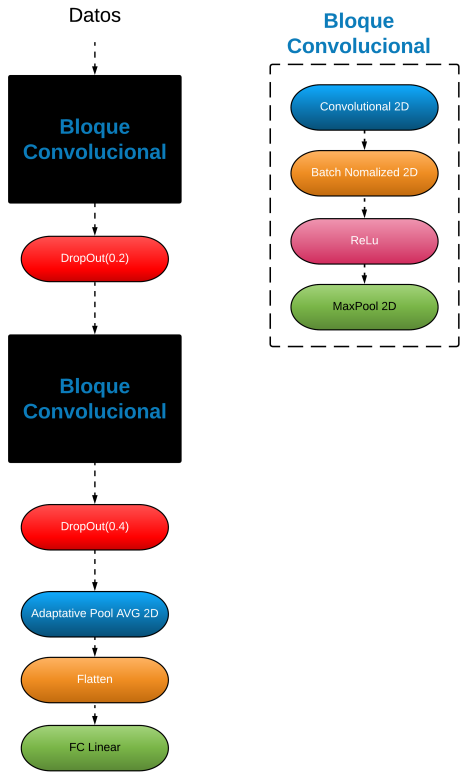

Figura 19. Arquitectura 1. 2 bloques convolucionales, para el análisis de espectrogramas. Se añadieron 2 DropOuts.

Tal cual podemos apreciar, se añadieron 2 DropOuts, con el objetivo de evitar la *memorización* al *desactivar* algunas neuronas. La primera sección posee un DropOut con una probabilidad de 0.2, mientras que para el segundo bloque, tenemos una probabilidad de 0.4

Los resultados de la red fueron bastante satisfactorios (Figura 30). No presentó problemas en *colab*. Pensando en la posibilidad de replicar el ejercicio tal cual fue presentado en el curso, se reintentó el diseño de la arquitectura con 4 bloques convolucionales, al cual, también se le dropouts (3 en este

caso). En la figura 20 Podremos apreciar de mejor forma, la arquitectura presente en este método.

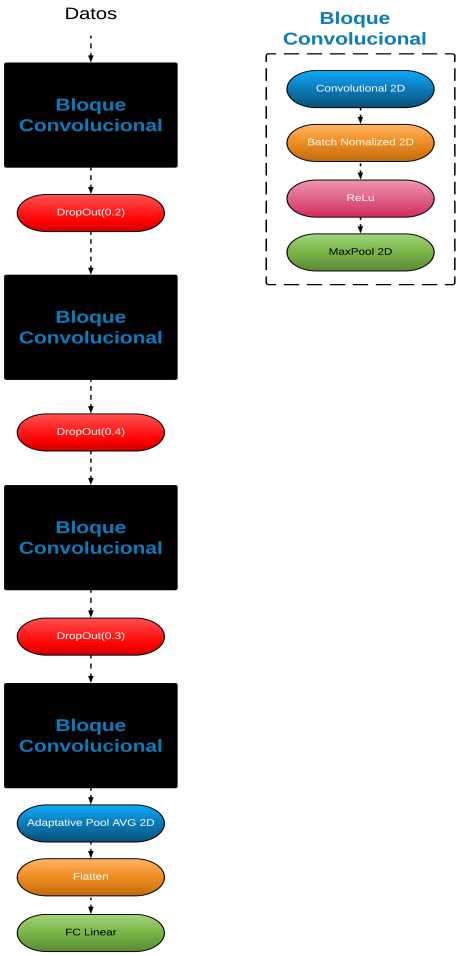

Figura 20. Arquitectura 2. 4 bloques convolucionales, para el análisis de espectrogramas. Se añadieron 3 DropOuts.

Curiosamente, al llevar a cabo el análisis de la presente red, uno pensaría que obtendría un mayor y mejor desempeño. Sin embargo, los resultados arrojados (Figura IV-B2) demostraron un desempeño menor que en la arquitectura evaluada con 2 bloques convolucionales.

Basado en los 2 resultados anteriores y que (en apariencia) a menor bloque convolucional, mayor desempeño, decidimos optar por una arquitectura bastante sencilla (1 solo bloque convolucional) y observar los resultados. Para evitar obviedades, nuevamente colocamos la arquitectura.

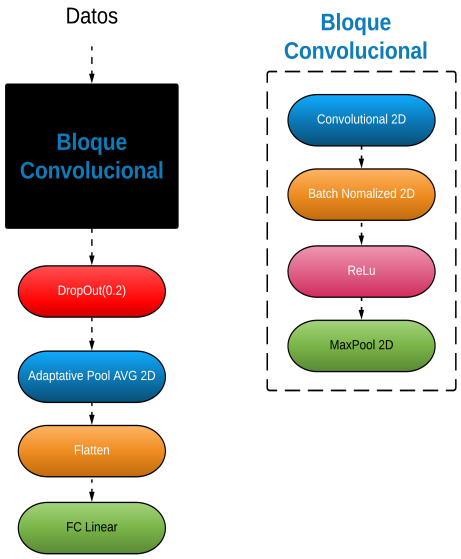

Figura 21. Arquitectura 3. 1 bloque convolucional, para el análisis de espectrogramas. Se añadió 1 solo DropOut.

Tal cual era de suponerse, no obtuvo resultados mejores (Figura 34), sin embargo, tampoco estuvo muy distante en comparación con la arquitectura 2 (Figura 20).

IV. Resultados

IV-A. Redes Neuronales Convolucionales 1D

A continuación presentamos las gráficas relacionadas a la pérdida y exactitud durante cada época en las 4 arquitecturas.

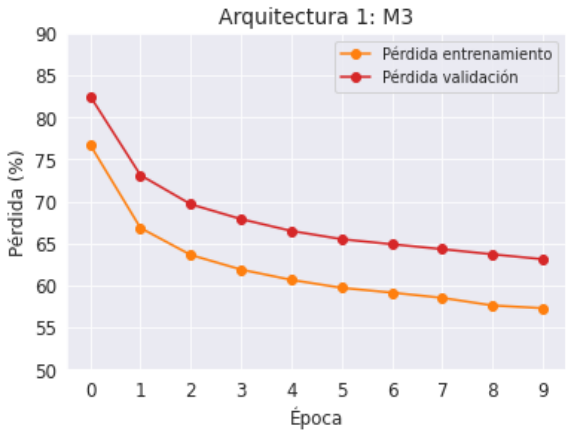

Figura 22. Pérdida en la última época: Entrenamiento: 57.29 %, Validación: 63.12 %

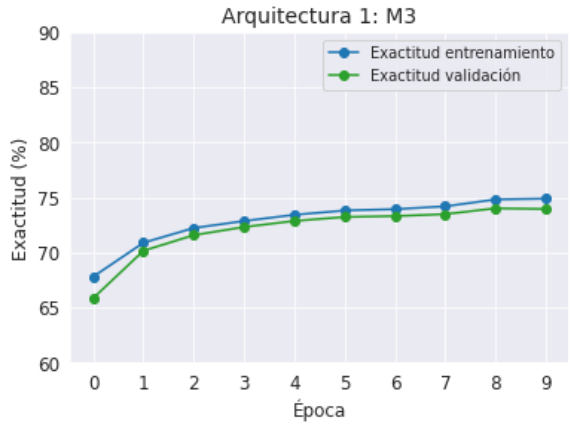

Figura 23. Exactitud en la última época: Entrenamiento: 74.89 %, Validación: 73.95 %

IV-A1. M3:

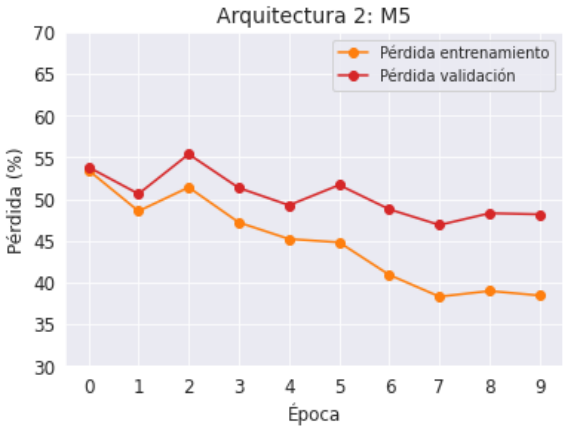

Figura 24. Pérdida en la última época: Entrenamiento: 38.44 %, Validación: 48.17 %

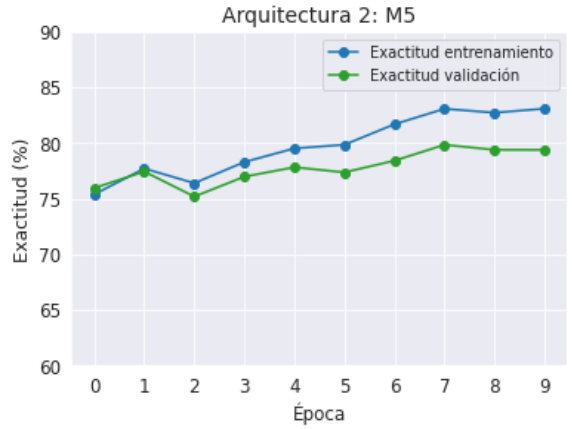

Figura 25. Exactitud en la última época: Entrenamiento: 83.08 %, Validación: 79.37 %

IV-A2. M5:

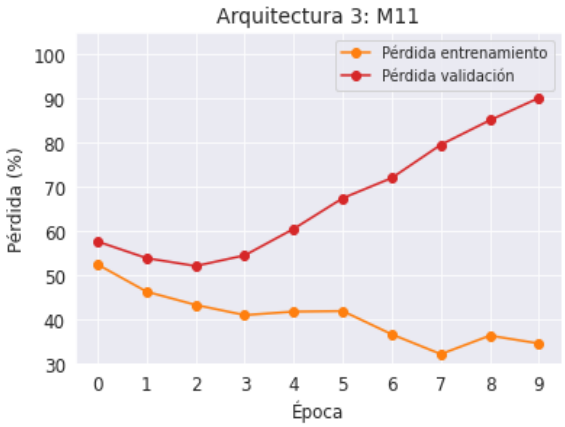

Figura 26. Pérdida en la última época: Entrenamiento: 34.62 %, Validación: 90.17 %

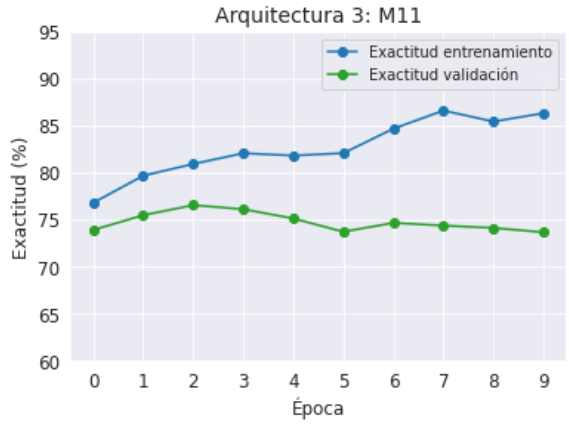

Figura 27. Exactitud en la última época: Entrenamiento: 86.32 %, Validación: 73.67 %

IV-A3. M11:

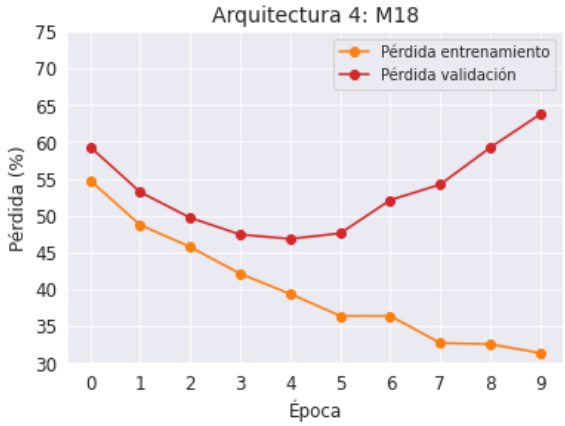

Figura 28. Pérdida en la última época: Entrenamiento: 31.34 %, Validación: 63.75 %

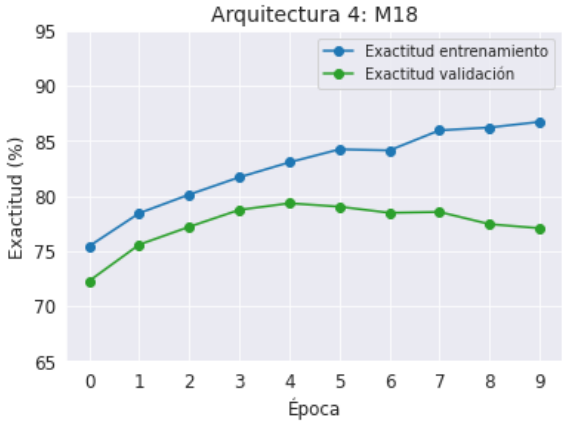

Figura 29. Exactitud en la última época: Entrenamiento: 86.72 %, Validación: 77.07 %

IV-A4. M18: De las gráficas mostradas anteriormente podemos apreciar que con M3 tenemos un menor sobreajuste, seguida de M5; sin embargo, con M5 tenemos mayor exactitud y menor pérdida en los conjuntos de entrenamiento y validación. Además, con estas arquitecturas parece ser que si aumentáramos un poco más la cantidad de épocas, seguirían mejorando los modelos; aunque realmente en ambos ya se empieza a apreciar la convergencia, por lo que quizás la mejora no sería tan significativa. En estas, la exactitud si va mejorando conforme pasan las épocas tanto en entrenamiento como en validación; y algo análogo ocurre con la pérdida, ya que igualmente va disminuyendo en ambos conjuntos.

Por otro lado, con la arquitectura M11 es con la que tenemos un mayor sobreajuste, aunque la M18 no se queda atrás. En uno y en otro la exactitud en los datos de entrenamiento es buena, y de hecho, es mejor que con la M3 o la M5, pero vemos que en los de validación se aleja; a pesar de que la exactitud en los datos de entrenamiento en ambas arquitecturas va mejorando conforme pasan las épocas, en los datos de validación en la época 3 y en la 5 es donde empiezan a decaer (poco, pero aún así no son muy buenas señales) en la arquitectura M11 y en la M18 respectivamente. Si bien es cierto que la pérdida va disminuyendo en el conjunto de entrenamiento, aquí también en los datos de validación a partir de la época 3 y de la época 5 vuelve a subir en la M11 y en la M18 respectivamente.

De igual manera, presentamos una tabla que contiene información relacionada al número de parámetros (***# Paráms.***) y tiempo de ejecución aproximado (***Tiempo***) en las 4 arquitecturas (con las adaptaciones mencionadas anteriormente):

Tabla II

Arquitectura	# Paráms.	Tiempo
M3	5,923	24 minutos
M5	37,507	36 minutos
M11	449,059	50 minutos
M18	924,643	80 minutos

Con respecto a esta última tabla, podemos notar que en la arquitectura M11 y en la M18 la cantidad de parámetros es muy grande comparándola con la de las otras dos. Peor aún, el tiempo de ejecución es demasiado en estas y como lo comentamos anteriormente, no vale la pena debido a que el sobreajuste que presentan es más notable.

Dicho todo esto, si quisiéramos resolver nuestra problemática usando convolucionales 1D, nos decidiríamos seleccionando a la arquitectura M5 pues:

- En la última época, es con la que tenemos la mayor exactitud en los datos de validación (79.37 %) y una menor pérdida (48.17 %) en estos mismos.
- Su número de parámetros es relativamente pequeño; su tiempo de ejecución es relativamente adecuado.
- Consideramos que realmente no se sobreajusta.

Por lo tanto, a continuación presentamos algunas predicciones hechas a 32 audios de los datos de prueba usando la arquitectura M5:

Donde la letra **V** corresponde a la etiqueta verdadera, mientras que la **P** es la relacionada a la etiqueta predicha. Así, de estas últimas dos imágenes podemos notar que acertó en 25 de 32 ($\frac{25}{32} = 78.125\,\%$ de exactitud en las predicciones), lo cual creemos que no está tan mal para este primer intento.

IV-B. Redes Neuronales Convolucionales 2D

A continuación presentaremos 2 gráficas resultado de cada una de las arquitecturas anteriormente presentadas. Cada gráfica corresponde a los valores obtenidos en *accuracy* y *Loss* para entrenamiento y validación.

IV-B1. : Arquitectura 1

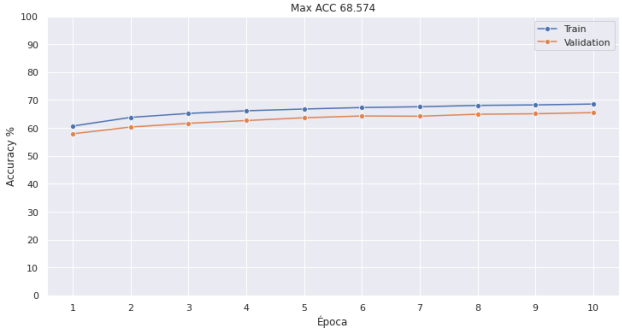

Figura 30. Resultados obtenidos (Hablando de Accuracy) para la arquitectura 1.

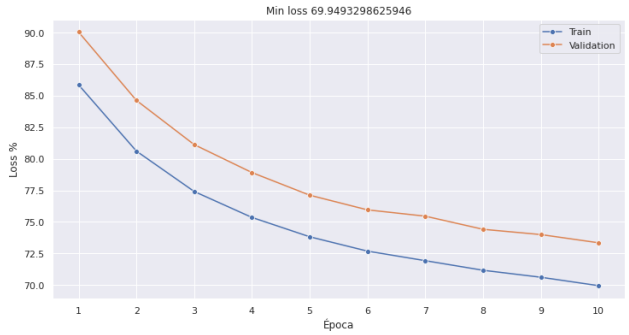

Figura 31. Resultados obtenidos (Hablando de pérdida) para la arquitectura 1.

Tal como se puede apreciar en las gráficas, hemos tenido un accuracy máximo en Entrenamiento es de 68.574, mientras que, hablando de la pérdida, nuestro valor mínimo nuevamente está presente en los datos de entrenamiento (69.949).

Como podemos apreciar de ambas líneas, parecen correr paralelamente, por lo que no hay presencia de *overfitting*.

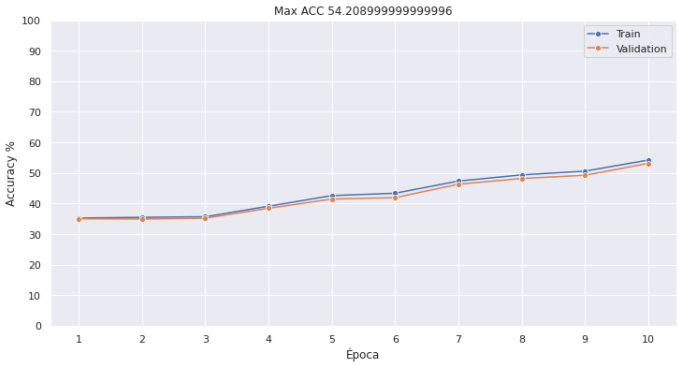

Figura 32. Resultados obtenidos (Hablando de Accuracy) para la arquitectura 2.

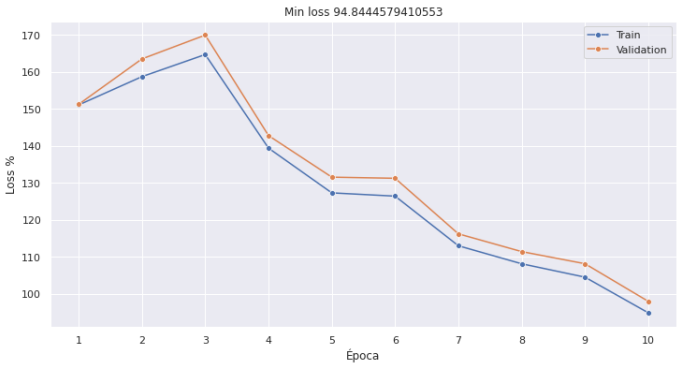

Figura 33. Resultados obtenidos (Hablando de pérdida) para la arquitectura 2.

IV-B2. Arquitectura 2: Para el caso de la segunda arquitectura, podremos estar seguros que a pesar de tener un desempeño no tan agradable como en el caso de nuestra primer arquitectura, no contamos con un *overfitting*. Podemos observar un accuracy máximo de 54.20 en entrenamiento y una pérdida mínima de 94.84.

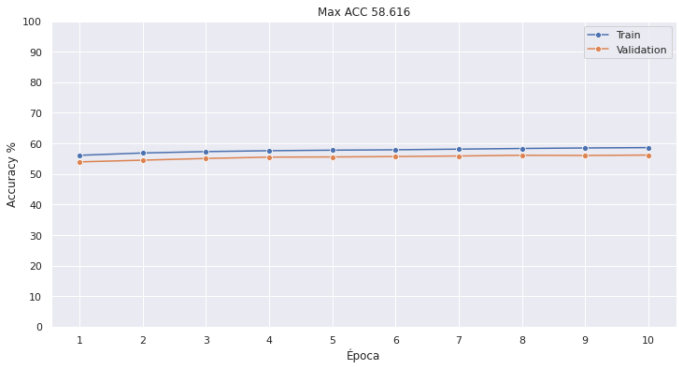

Figura 34. Resultados obtenidos (Hablando de Accuracy) para la arquitectura 3.

IV-B3. Arquitectura 3: Finalmente, podemos hablar acerca de la arquitectura con un solo bloque convolucional, el cual aparentemente no tuvo mayor incremento o aprendizaje (hablando de accuracy), pues realmente nunca sale del rango 50 - 60. Podemos observar un máximo en entrenamiento de 58.616, mientras que para la pérdida el valor mínimo fue de 88.92.

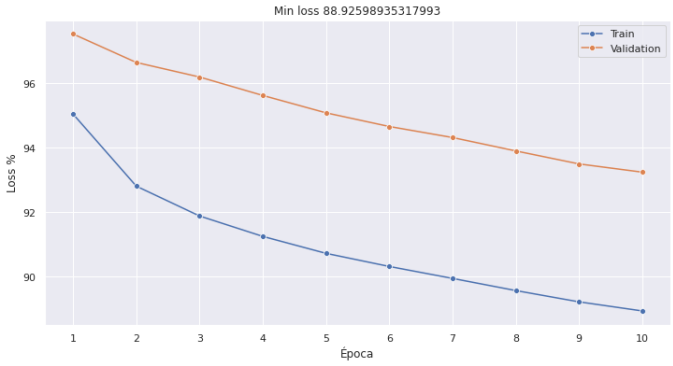

Figura 35. Resultados obtenidos (Hablando de pérdida) para la arquitectura 3.

IV-B4. Resumen: En resumen, podremos tener los siguientes datos por cada arquitectura:

Arquitectura	Accuracy	Pérdida	Tiempo	Parámetros
1	68.574	69.949	28.58	9,795
2	54.20	94.84	...	65,571
3	58.616	88.92	...	483

Tabla III
COMPARACIÓN DE ARQUITECTURAS PARA ESPECTROGRAMAS

Como hemos podido notar, tomando en cuenta que el primer modelo tuvo la menor pérdida y mayor accuracy, decidimos seleccionarlo como el mejor (Hablando de la sección de espectrogramas).

A continuación, presentamos una serie de predicciones a través de dicho modelo.

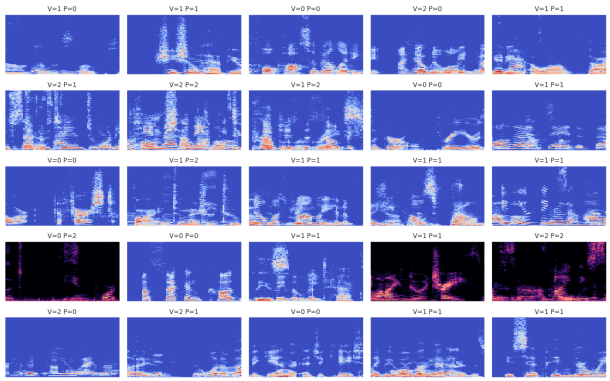

Figura 36. Presentación de desempeño de modelo. Se llevó a cabo la selección de un lote en el conjunto de prueba, mediante el cual, se pudo a preciar, a grandes rasgos, el comportamiento del modelo seleccionado.

Al igual que en las predicciones de la sección de convolucionales 1D, en este caso **V** corresponde al valor verdadero de la clase a la que pertenece, mientras que **P** indica la clase en la que el modelo predice pertenencia.

V. CONCLUSIÓN

Esta tarea nos resultó complicada desde el momento en que no contábamos con los datos (audios) que nos permitieran atacarla. Afortunadamente, conseguimos generarlos por medio de otros conjuntos de datos que se encontraban en internet y logramos que estuvieran balanceados, lo cual ayudó bastante.

Para efectos prácticos, presentamos la comparación de rendimiento obtenido a través de las 10 épocas en *Accuracy* y *Loss*, para los que consideramos, resultaron en los mejores modelos para cada sección (1D y 2D).

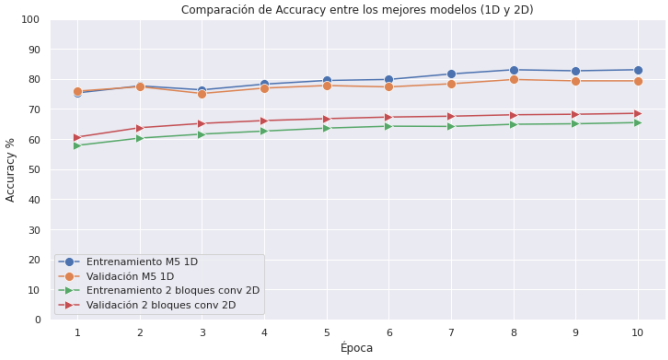

Figura 37. Accuracy. Comparación entre el rendimiento obtenido para la arquitectura M5 (para el problema de formas de audio) y la arquitectura diseñada con 2 bloques convolucionales (para el problema de espectogramas).

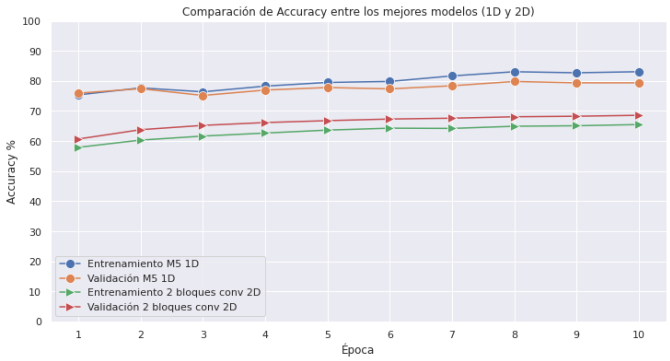

Figura 38. Loss. Comparación entre el rendimiento obtenido para la arquitectura M5 (para el problema de formas de audio) y la arquitectura diseñada con 2 bloques convolucionales (para el problema de espectogramas).

De los análisis y gráficas pasadas, parece ser que esta problemática se resuelve de una mejor manera utilizando redes neuronales convolucionales 1D, específicamente con la arquitectura M5 propuesta en [4] con las pocas adaptaciones que mencionamos. Es importante decir que decidimos no usar la M34-Res debido a que sus parámetros eran demasiados y creíamos que presentaría dificultades similares a la M11 y M18.

Por lo tanto, podemos concluir que la resolución parece ser adecuada, pero en un futuro no descartamos atacarla basándonos, por ejemplo, en redes recurrentes. Asimismo, próximamente pretendemos darle más complejidad a esta tarea tratando de predecir el sexo de el, la, los o las participantes en el audio.

REFERENCIAS

[1] MORAT, J. (2017). *Los Decibeles ¿qué son y para qué sirven?*. Junio 3, 2021, de audiomidilab Sitio web: https://audiomidilab.com/los-decibeles-que-son-y-para-que-sirven/.

[2] VASQUEZ, E. (2021). *¿Qué son los decibeles y cómo se miden?*. Junio 3, 2021, de altavocesprofesionales Sitio web: https://altavocesprofesionales.com/que-son-los-decibeles-y-como-se-miden/.

[3] PROMAX. (2008). *Espectrograma: Monitorizar fácilmente el espectro de la señal*. Junio 3, 2021, de promax Sitio web: https://www.promax.es/esp/noticias/158/Espectrograma/.

[4] DAI, WE. ET AL. (2016). *Very Deep Convolutional Neural Networks For Raw Waveforms*. CoRR, abs/1610.00087, 5. Sitio web: http://arxiv.org/abs/1610.00087.

Jesús Tapia (1999 - ...) Actualmente cursando su último semestre de la licenciatura en Ciencia de Datos en la Universidad Nacional Autónoma de México (UNAM). Anteriormente cursó la carrera de Actuaría en la Facultad de Ciencias (UNAM), aunque no la terminó. Decidió optar por la ciencia de datos (Como cambio de carrera) y ese ha sido su camino hasta ahora.

Jonathan Martiñón (1998 - ...) Actualmente cursando su último semestre de la licenciatura en Ciencia de Datos en la Universidad Nacional Autónoma de México (UNAM). Anteriormente cursó la carrera de Ingeniería en Computación, impartida en la Facultad de Estudios Superiores Aragón, aunque al final optó por dedicarse a la ciencia de datos, realizando cambio de carrera en su segundo año de ingeniería.