Brandon Alain Cruz Ruiz
198573

Braulio Piña Amaros
153058

Diego Villegas Juárez
197902

Resumen Técnico

Este este resumen presentamos una descripción general de las principales referencias usadas en el modelo de Automatic Speech Recognition o ARS, esta documentación utiliza arquitecturas basadas en Transformer en el reconocimiento de voz. Los tres siguientes artículos muestran cómo son aplicadas estas arquitecturas con distintas modificaciones para lograr mejores tiempos de entrenamiento o mejores resultados en los puntajes BLEU (bilingual evaluation understudy).

Attention is all you need

Las redes neuronales recurrentes, las redes neuronales de memoria a largo y corto plazo así como las redes neuronales recurrentes cerradas son ampliamente utilizadas y consideradas como el estado del arte en modelado de secuencias y problemas de traducción. A pesar de que se han hecho grandes esfuerzos para mejorar la eficiencia computacional mediante factorización y cómputo condicional, aún está presente los límites del cómputo secuencial. En este paper se propone el *Transformer* que es un modelo que depende totalmente de los mecanismos de atención. Este *Transformer* permite más paralelización y en 12 horas de entrenamiento en 8 GPUs puede lograr buenos resultados.

Se hace mención a los modelos *ConvS2S* y *ByteNet* en los cuales el número de operaciones necesarias para relacionar dos señales de entrada o salida, depende de la distancia entre los puntos, creciendo lineal y exponencialmente respectivamente. Lo cual dificulta el proceso de aprendizaje. Por otro lado, con *Transformer* este número de operaciones es constante con el costo de obtener una solución reducida. También, *Transformer* depende totalmente de los mecanismos de auto-atención para representar entradas y salidas sin usar convoluciones o RNN alineadas secuencialmente.

El modelo Transformer utiliza la estructura de *encoder-decoder* utilizando capas apiladas y totalmente conectadas para el *encoder* y el *decoder*.

Encoder: El encoder está conformado por 6 capas idénticas, cada capa tiene dos subcapas: la primera es el mecanismo de auto-atención con multihead (lo que permite correr el modelo de atención múltiples veces en paralelo) es una red *feed-forward* totalmente conectada. Se utilizan conecciones residuales[1] en ambas subcapas seguidas de una capa más de normalización.

Decoder: El decoder también tiene 6 capas idénticas. Además de las dos subcapas del encoder, se inserta una tercera subcapa que se alimenta de la salida del encoder para utilizar mecanismos de auto-atención con multihead. También hay conexiones residuales en cada subcapa y se modificó la primera subcapa para asegurar que las predicciones de la posición *i* dependan únicamente de las salidas conocidas de las posiciones menores que *i.*

Para las funciones de atención, se mencionan los siguientes puntos importantes en donde se explican los componentes principales.

[1] They are used to allow gradients to flow through a network directly, without passing through non-linear activation functions. Non-linear activation functions, by nature of being non-linear, cause the gradients to explode or vanish (depending on the weights). [1]

Scaled Dot-Product Attention: Los autores lo llaman “Scaled Dot-Product Attention”. Las entradas consisten de queries y llaves de dimensión d_k y valores de dimensión d_v. Se calcula el producto punto de los queries con todas las llaves, se divide entre $\sqrt{d_k}$ y se aplica una función softmax para obtener los pesos de los valores. Para valores grandes de d_k se sospecha que los producto puntos pueden crecer mucho así que la función de softmax puede llegar a regiones donde los gradientes sean demasiado pequeños, es por eso que se escala el producto punto con $\frac{1}{\sqrt{d_k}}$.

Multi-head Attention: Se menciona que se utilizaron proyecciones lineales de los queries, de las llaves y los valores *h* veces. Con estas versiones proyectadas, se hace una función de atención paralela, las salidas se concatenan y son proyectadas de nuevo. Esto permite que el modelo atienda conjuntamente información de diferentes representaciones en diferentes posiciones.

Position-wise Feed-forward Networks: Esta capa, que se encuentra en el encoder y decoder, consiste de dos transformaciones lineales con una función de activación ReLU entre ellas. Estas transformaciones usan diferentes parámetros de capa en capa.
Embeddings and Softmax: Debido a que no se usan convoluciones o recurrencia, los autores explican que deben de tener información relativa o absoluta de los tokens en la secuencia, para ello utilizan “*positional encodings*” a las entradas posicionados en la parte inferior del stack del encoder y decoder. Se utilizaron las funciones de seno y coseno con diferentes frecuencias, esto con la teoría de que el modelo podría aprender fácilmente a asistir por posiciones relativas.

¿Por qué utilizar este tipo de modelo de atención?, principalmente por tres razones:

- Complejidad computacional por capa
- Por la cantidad de cómputo que puede ser paralelizado
- La longitud del camino entre dependencias de largo alcance en la red

Principalmente el tercer punto es uno de los principales factores que afectan la capacidad de aprender. Entre más corto sea este camino, es más fácil aprender dependencias de largo alcance. Se menciona que las capas de auto-atención son más rápidas que las capas recurrentes cuando la secuencia de longitud *n* es más pequeña que la dimensión de representación *d,* que suele ser el caso en modelos que usan el estado del arte. Uno de los beneficios es que estos modelos podrían arrojar modelos más interpretables.

Para sus pruebas de su modelo se utilizaron diferentes datasets, para hacer la traducción del Inglés al Alemán y del Inglés al Francés. Se utilizaron 8 GPUs NVIDIA P100 con un modelo base que tomó 12 horas y un modelo más amplio que tomó 3.5 días entrenar. Se utilizó el optimizador Adam, y las tasas de aprendizaje se fueron cambiando a través del entrenamiento, incrementando la tasa de aprendizaje linealmente por los primeros 4000 pasos de entrenamiento y posteriormente se redujo proporcionalmente a la inversa de la raíz cuadrada del paso actual. Sobre la regularización se implementó dropout y suavizamiento de etiquetas.

Con los resultados obtenidos por los autores se demuestra que reduciendo el tamaño de d_k afecta considerablemente la calidad del modelo, el dropout, por otra parte ayuda mucho para evitar sobreajuste y obtener buenos resultados. Los autores concluyen que el *Transformer* puede ser entrenado significativamente más rápido que arquitecturas basadas en capas recurrentes o convolucionales.

Brandon Alain Cruz Ruiz 198573
Braulio Piña Amaros 153058
Diego Villegas Juárez 197902

Very deep self-attention networks for end-to-end speech recognition

Los modelos de auto-atención son atractivos debido a la habilidad de establecer una conexión directa entre cada elemento en la secuencia. Estos modelos son escalables, pues la longitud de la secuencia no presenta factores limitantes como en el caso de las CNN con el tamaño del kernel o con el problema del desvanecimiento del gradiente en LSTM.

Los componentes del modelo consta de un encoder que consume una secuencia de origen y genera una representación de alto nivel, y un decoder, que genera una secuencia objetivo. Tanto el encoder como el encoder son redes neuronales que requieren componentes neuronales que sean capaces de aprender la relación entre los pasos de tiempo en la secuencia de entrada y salida.

Como se muestra en el paper “Attention is all you need”, para este caso también se hace uso del *multi-head attention* con las mismas proyecciones para los queries *Q,* las llaves *K* y los valores *V,* esto se procesan en *n* cabezas paralelas. Los resultados de las *n* cabezas son concatenadas. También se utilizan capas que tienen subcapas de auto atención y con redes neuronales *feed-forward*. Para el caso de reconocimiento de voz, el *positional encoding* ofrece una mayor ventaja en comparación con *positional embeddings*, debido a que las señales de voz pueden ser más grandes con una varianza mayor comparado con el texto.

Para el encoder, las entradas son enviadas a una capa de auto-atención seguida por una red neuronal *feed-forward* con una capa oculta con función de activación ReLU. Además, se incluyen conexiones residuales que establecen atajos entre las representaciones de bajo nivel y las capas más altas. Estas conexiones residuales pueden incrementar la magnitud de los valores de la neurona que son después normalizados. Para el decoder, es el mismo que en el documento “Attention is all you need”. Haciendo énfasis en que se debe mantener una máscara para que cada estado sólo pueda tener acceso a los estados pasados.

En varios modelos profundos sufren de problemas de sobreajuste y hacen que la optimización sea difícil. Es por eso que los autores proponen aplicar capas residuales estocásticas, que son similares a utilizar *dropout*, en donde la idea es que algunas capas se omitan durante el entrenamiento. Esta capa estocástica fundamentalmente aplica una máscara *M* a la función *F* (esta función *F* puede ser una función de auto-atención, una capa *feed-forward* o incluso un decoder-encoder). Esta máscara M puede tomar solo dos valores: 1 ó 0, generados por una función Bernoulli. Cuando es 1, la función F se activa, por otro lado, cuando M = 0, la función es omitida. Estas conexiones estocásticas permiten crear más configuraciones de subredes durante el entrenamiento, mientras que en el proceso de inferencia, toda la red está presente.

Finalmente, como las capas son seleccionadas durante el entrenamiento con probabilidad 1 - p_l y siempre están presentes durante la inferencia, la salida de las capas se escalan por $\frac{1}{1-p_l}$. Por consiguiente, cada conexión estocástica tiene la siguiente forma durante el **entrenamiento** (la escala se quita durante las pruebas):

$$p_l = \frac{l}{L}(1-p)$$

$$R(x) = \text{LayerNorm}(M * F(x) * \frac{1}{1-p_l} + x)$$

Para los experimentos realizados por los autores se basaron mucho en las características de las pruebas realizadas en el paper “Attention is all you need”, se utilizó Adam, con tasa de aprendizaje que se adapta, pero en vez de ser 4000 pasos, se utilizaron 8000 pasos para la tasa de aprendizaje. Se aplicó *dropout* = 0.2, con *character dropout con p = 0.1* y suavizamiento de etiquetas con $\epsilon = 0.1$.

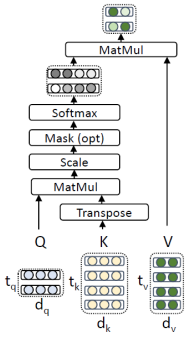
Principalmente el enfoque de las pruebas fue el uso de redes estocásticas, observando que en pocas capas, estas redes no muestran mucha mejora, en comparación con 24 o 48 capas, demostrando la habilidad del modelo de generalizar mejor. Los autores demuestran en sus resultados que el rendimiento de una red más profunda con menor número de parámetros es mejor que una configuración más amplia pero menos profunda. Por otra parte se observa que el encoder requiere una red más profunda que el decoder. Concluyendo que el *Transformer* puede ser efectivo para reconocimiento automático de voz teniendo en cuenta que es necesario configurar modelos estocásticos muy profundos.

Speech-Transformer: A no-recurrence sequence-to-sequence model for speech recognition

Los modelos secuencia-a-secuencia han sido muy exitosos últimamente, se han visto beneficiados por la construcción de encoder más profundos, el esquema efectivo de suavizamiento de etiquetas y la descomposición de secuencias latentes, a pesar de todo esto, sufren de un gran problema: el tiempo de entrenamiento sigue siendo lento.

El modelo de Speech-Transformer está basado en la arquitectura de encoder-decoder esta transformación toma una secuencia $(x_1, ..., x_T)$ a una representación de una oculata $h = (h_1, ..., h_L)$ data esta h el decoder genera una secuencia output de $(y_1, ..., y_T)$ caracteres. En cada paso, el decoder toma los caracteres emitidos previamente como entradas adicionales al emitir el siguiente carácter.

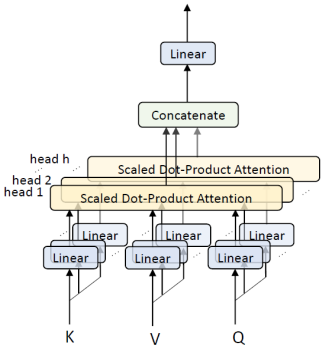
Esto difiere de un modelo recurrente seq2seq, primeramente por un producto punto escalado o Scaled Dot-Product Attention, esto se usa para poder calcular representaciones de los inputs, tiene tres entradas: queries, keys y values. El resultado o output de una query es calculado como una suma ponderada de los values, donde cada peso de un value es calculado por una función diseñada de la query con la key. esto se puede ver gráficamente manera:



$$\text{Attention}(Q, K, V) = \text{softmax}\frac{QK^T}{\sqrt{d_k}}V$$

La segunda diferencia está dada por un Multi-head attention core, esté core del modelo está aplicado para aprovechar las diferentes representaciones de forma conjunta, se calculan $h$veces es producto punto escalado, después estos productos punto son concatenados independientemente para

actualizar otra proyección lineal, obteniendo así la la proyección o outputs final de este d-modelo dimensional. esta relación se puede ver de la siguiente forma:



Multi-head (Q, K, V) =Concat$(head_1, ..., head_h)W^o$

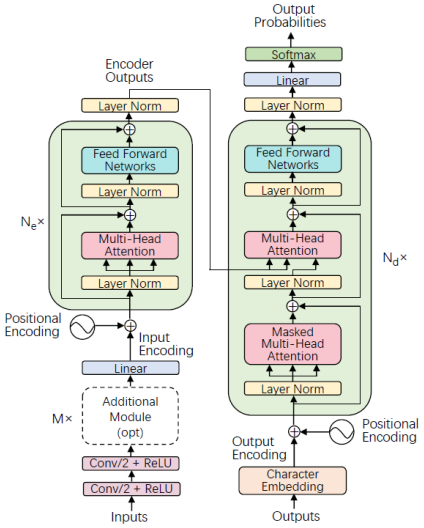
donde

$head_i$ = Attention(QW_i^Q, KW_i^K, VW_i^V)

Por último es Position-wise Feed-Forward Network, este último core consiste en dos transformaciones lineales con una activación ReLu entre ellas, data por la siguiente relación

FFN(x)=max$(0, xW_1 + b_1)W_2 + b_2$

Para este modelo definimos distintos cores para su aplicación, este modelo se centra en transformar la secuencia de características del habla a la secuencia de caracteres correspondiente, esta se transformacion se puede representar en espectrogramas bidimensional de tiempo y frecuencia, esta representación gráfica se puede ver en el siguiente diagrama.



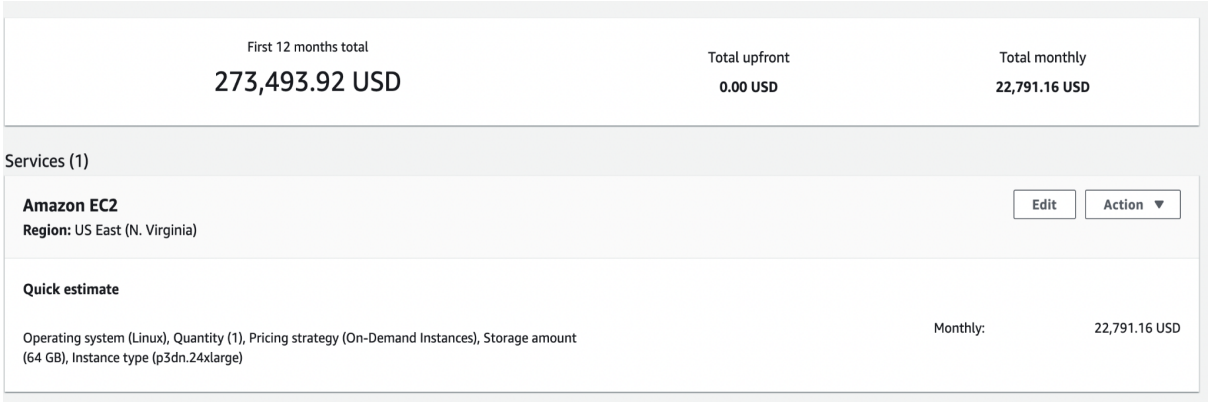
En primer lugar, se utiliza una incrustación a nivel de carácter para convertir la secuencia de caracteres a un output, después la suma se coloca en una pila de N_d bloques para obtener la decodificación final, para finalmente, estas salidas se transforman a las probabilidades de clases de salida mediante una proyección lineal y una posterior función softmax.

Este modelo se aplicó con un conjunto de prueba de del Wall Street Journal (WSJ) con características acústicas de un tamaño de ventana de 25 ms. este conjunto obtuvo como resultado disminución en los costos de entrenamiento. Se mitigaron los elevados costos de procesamiento de información de los modelos seq2seq recurrentes, en el conjunto de datos de WSJ, el modelo converge con costos considerablemente pequeños y logró un rendimiento competitivo, que muestra la eficiencia y eficacia del Speech-Transformer. Estos resultados se pueden ver mas a detalle en el

Costos de las arquitecturas y tiempo de procesamiento

Precios AWS

De acuerdo a los tiempos que se dan en el paper “Attention is all you need”, se menciona que se utilizaron 8 GPU y el modelo se entrenó en 3.5 días. Investigando un poco en AWS, hay distintas instancias de EC2 que podríamos utilizar.



- Instancias Amazon EC2 P3 tiene hasta 8 GPU V100 de NVIDIA Tesla.
- Instancias G3 de Amazon EC2 G3 tiene hasta 4 GPU NVIDIA Tesla M60.
- Instancias G4 de Amazon EC2 tienen hasta 4 GPU NVIDIA T4.
- Instancias Amazon EC2 P4 tiene hasta 8 GPU NVIDIA Tesla A100 Tesla.

El paper menciona los GPUs P100. En la lista anterior no hay GPUs NVIDIA Tesla P100, pero para propósitos de está práctica podríamos utilizas las instancias EC2 P3 y las instancias EC2 P4 haciendo un estimado de 12 horas y los 3.5 días que se mencionan en el documento.

Debido a que los modelos pueden entrenarse en días, no es necesario tener una instancia reservada por todo un año, lo cual nos deja la opción de utilizar “On demand instances” lo cual es más caro, pero como solo lo utilizaremos por poco tiempo es la mejor opción.

Instancia P3

Para esta estimación se utilizó la instancia p3dn.24xlarge con las siguientes características.

Los precios calculados son los siguientes:

Como se puede ver se estima un costo de casi **23 mil dólares al mes**, pero si lo reducimos a los 3.5 días se estarían pagando aproximadamente **3400 dólares** y si entrenamos al modelo en las 12 horas tendríamos un costo de casi **480 dólares.**

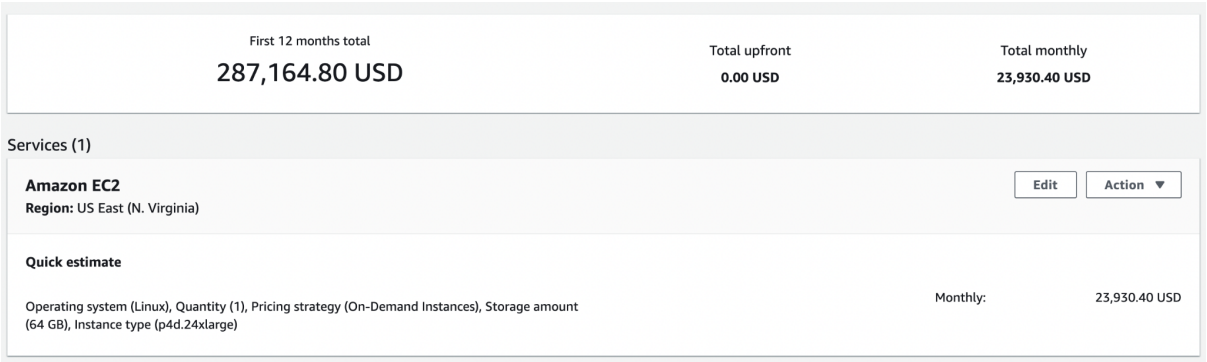
Instancia P4

Para esta estimación se utilizó la única instancia P4 disponible en EC2.

Instancia	GPU	CPU virtuales	Mem(GiB)	Ancho de banda de la red	GPUDirect RDMA	GPU de pares	Almacenamiento	Ancho de banda de EBS
p4d.24xlarge	8	96	1152	400 Gbps ENA y EFA	Sí	600 GB/s NVSwitch	8 x 1 TB NVMe SSD	19 Gbps

Con estas especificaciones obtenemos la siguiente estimación:

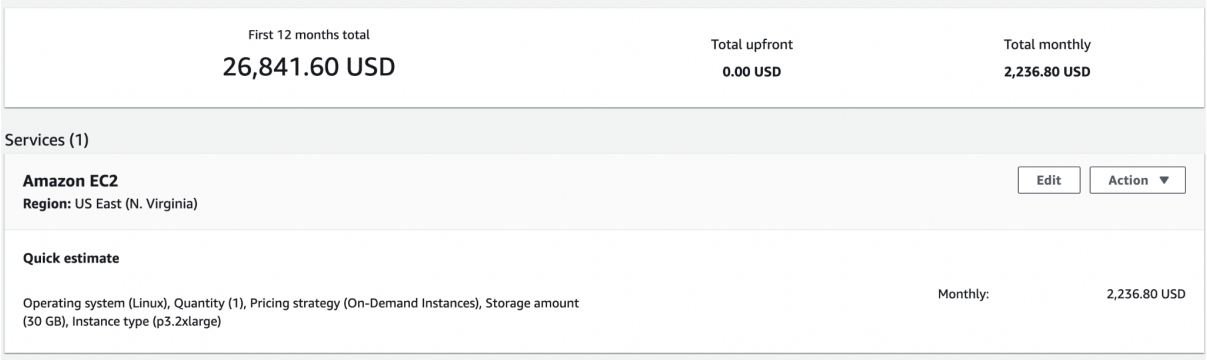
Instancia	GPU	CPU virtual	Memoria (GiB)	Memoria de GPU (GiB)	GPU PUNTO A PUNTO	Almacenamiento (GB)	Ancho de banda dedicado a EBS	Rendimiento de redes
p3.2xlarge	1	8	61	16	-	Solo EBS	1,5 Gbps	Hasta 10 Gigabits
p3.8xlarge	4	32	244	64	NVLink	Solo EBS	7 Gbps	10 gigabits
p3.16xlarge	8	64	488	128	NVLink	Solo EBS	14 Gbps	25 gigabits
p3dn.24xlarge	8	96	768	256	NVLink	2 x 900 SSD NVMe	19 Gbps	100 gigabits



Los precios en comparación con la estimación anterior no difieren mucho, solo hay una diferencia de 1000 dólares mensuales **(24 mil dólares al mes)**. Para los 3.5 días estaríamos pagando **3500 dólares** y por las 12 horas **500 dólares**.

Speech-Transformer

En el paper titulado “SPEECH-TRANSFORMER: A NO-RECURRENCE SEQUENCE-TO-SEQUENCE MODEL FOR SPEECH RECOGNITION” se menciona que únicamente es necesario 1 GPU con 1.2 días, así que este sería el costo para entrenar este modelo.



Para este caso se utilizaría una instancia **EC2 p3.2xlarge** que únicamente cuenta con 1 GPU, para este caso nos costaría por 1.2 días aproximadamente **112 dólares** que es un precio significativamente menor que las dos estimaciones mencionadas anteriormente.

Consideraciones

Como nota general, haciendo un poco más de investigación los modelos de NVIDIA A100 (Junio 2020) y V100 (Junio 2017) son relativamente más nuevos que los GPU P100 (Junio 2016) y presentan mejores características varios aspectos que podrían mejorar considerablemente el poder de cómputo, por lo que seguramente los precios se reducirían un poco porque el tiempo de entrenamiento debería ser menor.

En esta liga se puede hacer la comparación de las tarjetas gráficas, en este caso se pone la liga de la comparación entre V100 y P100:

https://technical.city/es/video/Tesla-P100-PCIe-16-GB-vs-Tesla-V100-PCIe

Para las comparación de P100 y A100 utilizamos este comparador:

https://technical.city/es/video/A100-PCIe-vs-Tesla-P100-PCIe-16-GB

Conclusiones

Como seguramente se sabe, estos papers están fuertemente relacionados, hay muchas especificaciones que comparten principalmente con el paper “attention is all you need”, sin embargo, se hacen modificaciones particulares que benefician aún más este tipo de modelos. Un ejemplo muy claro es el tiempo de entrenamiento que toman estos modelos e incluso en el número de GPUs necesarios para tener un modelo base. Con la evolución del procesamiento del cómputo, con las nuevas capacidades de las distintas tarjetas gráficas y con el aprovechamiento de los procesadores (paralelización), muy probablemente este tiempo se reducirá aún más. Sin duda, las técnicas y

conceptos empleados ayudaron a entender con mayor facilidad como es que funciona un Transformer, la estructura general en el encoder y decoder y las responsabilidades, entradas y salidas de cada una de las capas y subcapas de dichos componentes. Gracias a esto, pudimos llevar a cabo las modificaciones necesarias para entrenar nuestro modelo con diferentes *datasets* en otro idioma, en este caso en español.

Limitantes

Las principales limitantes que encontramos fueron:

- El entrenamiento con voces masculinas y femeninas arrojaba una pérdida nula, lo que nos hace pensar que quizá es necesario aumentar el tamaño de muestra de ambos tonos de voces, o quiźa se tenga que hacer algún tipo de adaptación para tonos diferentes.
- Los caracteres especiales del español no son admitidos en la predicción, en especial los acentos, lo que resulta ser una característica muy limitativa por la función que cumplen las tildes en el Español escrito, y que a veces son lo que diferencia a una palabra de otra en su escritura y connotación.

Decidimos meterle una cosa extra a nuestro proyecto, queremos ver como se ven las ondas de sonido en nuestro dataset, comparando las de mujeres con hombres.

- Quizá podamos meterle una capa de procesamiento a nuestros datos para homologar las voces de mujeres y hombres y entrenar un modelo con las voces homologadas.

La liga al proyecto es la siguiente:

https://github.com/BraulioPi/Bumblebee-deep_learning/blob/main/ASR_parcero.ipyn