PERITAJE INFORMÁTICO SOBRE IDENTIFICACIÓN AUTOMÁTICA DE HABLANTES: Pre-procesamiento de prueba forense

Juan Manuel Miguez
Luis Enrique Angulo

Resumen

El pre-procesamiento de registros de voz, permite obtener un conjunto de datos que será procesado posteriormente por un modelo predictivo de aprendizaje supervisado.
El objetivo es realizar predicciones con un alto nivel de confianza, para generar pruebas forenses donde el informe no resulte impugnado por motivos técnicos. En ese sentido, los atributos de la voz que componen el set de datos que se utilizará en el modelo, no deben ser distorsionados.
La identificación forense de hablantes por medio de métodos biométricos automáticos, implica que el método de predicción sea reproducible, utilizando muestras de voz sin alterar su cadena de custodia.
Los laboratorios forenses en la materia, pueden utilizar esta tecnología como soporte a los dictámenes elaborados en causas judiciales, y emplear métodos combinados para evitar la impugnación.

Palabras Clave: *Google-Colab, Anaconda, Jupyter notebook, Pytube, sox y pysox, ffmpeg, pydub, xlrd, IPython.display, librosa, pandas, keras, Matplotlib y Numpy, essentia.standard.Monoloader, método predict, performance, Deep Fake, EPOCH, loss, accuracy, GPU TPU, VALID_SPLIT.*

Introducción

El presente análisis es una extensión al documento "PERITAJE INFORMÁTICO SOBRE IDENTIFICACIÓN AUTOMÁTICA DE HABLANTES"[41]
Dentro de la clasificación de los sistemas que permiten la identificación forense de hablantes, se encuentran múltiples enfoques como: análisis espectrográfico-auditivo, enfoque perceptual, fonético-acústico, métodos automáticos, semiautomáticos y métodos combinados.
El presente trabajo se enfoca en brindar una ayuda para los métodos semiautomáticos y automáticos, que trabajan directamente con archivos de audio digitalizados, cuya conversión y normalización componen el conjunto de datos, siendo así el pre-procesado de los registros de voz una parte fundamental para crear éstos conjuntos de datos que a la postre serán usados como evidencia en casos dónde se requiere validar si la voz de los registros pertenece al presunto implicado o no.
Usualmente uno de los puntos críticos del problema de identificación forense de hablantes en procesos judiciales suele ser la impugnación de los resultados por controversias técnicas con respecto a los modelos de aprendizaje automático empleados, puesto que pueden llegar a carecer de un concepto válido unificado con respecto a la base científica de los mismos, con lo cual el dictamen expedido por un perito puede llegar a ser impugnado.

El desarrollo de un método paso a paso para el pre-procesamiento de archivos de audio correspondientes a grabaciones de personas de interés en arbitramentos judiciales, contribuye a sentar las bases para formalizar los procesos y mejores prácticas que conlleven a una estandarización que permitirá tanto a los peritos informáticos como a especialistas fonéticos forenses reducir los tiempos de ejecución de pruebas y entrega de dictámenes con altos grados de certeza y prevenir impugnaciones.

Antecedentes

El análisis de registros de audio que son usados en estrados judiciales con el fin de reconocimiento biométrico de voz, ha tenido muchas controversias a lo largo de los años; sin embargo, la tendencia puede empezar a cambiar gracias a los procesos de aprendizaje automático que se basan en modelos científicos. Empero, se sostiene la resolución del juez para determinar la identidad del presunto sospechoso teniendo en cuenta las comparaciones hechas por los peritos informáticos o forenses fonéticos.

Justificación

Para aprovechar al máximo la capacidad de procesamiento al mínimo costo, el set de datos debe estar normalizado para llegar con mayor rapidez a un modelo entrenado en óptimas condiciones, de manera que la predicción se resuelva con un alto nivel de certeza.

Librerías utilizadas

1. ***Lib 1: pytube***

Pytube[38][39] es una librería ligera, abierta y gratuita desarrollada en Python, que permite descargar videos de Youtube. Esta librería incluye la utilidad de línea de comando, permitiendo descargar videos desde la terminal de comandos, además, no tiene dependencias de terceros. A continuación se presenta algunas de sus características:

- Soporta streams progresivos y tipo DASH.
- Puede descargar una playlist completa.
- Fácil registro de callbacks como on_download_progress y on_download_complete.
- Interfaz de línea de comando incluida.
- Soporta caption track.
- Salida de caption tracks en formato .srt.
- Capacidad de captura de thumbnail URL.
- Código fuente extensamente documentado.
- Sin dependencias de terceros.

2. ***Lib 2: sox y pysox***

SoX[27][28][32] es una utilidad de línea de comando multiplataforma (Windows, Linux, MacOS X, etc.) usada para procesamiento de audio, puede convertir varios formatos de archivos de audio en otros formatos, convertir tasas de muestreo. También puede aplicar varios efectos de sonido a estos archivos, adicionalmente puede reproducir y grabar archivos de audio en muchas de las plataformas. Pysox[40] es la librería construida en Python alrededor de las funcionalidades propias de SoX[33][34].

3. ***Lib 3: ffmpeg***

ffmpeg[26][31] es un convertidor muy rápido de audio y video, incluso desde una fuente en vivo de audio/video. Puede convertir entre archivos con tasas de muestreo arbitrarias y cambiar el tamaño del video con una alta calidad. Puede ser usado para convertir WAV a MP3 y viceversa.

4. ***Lib 4: pydub***

Pydub[1] es una librería de Python que trabaja solo con archivos .wav. Al hacer uso de la librería se puede reproducir, dividir, integrar o editar los archivos de audio .wav.
Las siguientes son las funcionalidades que se pueden ejecutar en la pydub:

- Reproducir archivos de audio.
- Obtener información del archivo como longitud de canales.
- Incrementar/disminuir el volumen de un archivo .wav determinado.
- Integrar dos o más archivos de audio.
- Exportar un archivo de audio.
- Dividir un archivo de audio.

5. ***Lib 5: xlrd***

La librería xlrd es usada para extraer datos de una hoja de cálculo, solo trabaja con archivos de extensión .xls. Es usado para leer, escribir o modificar datos, permite navegar entre varias páginas de la hoja de cálculo y extraer datos dependiendo de ciertos criterios definidos.

6. ***Lib 6: IPython.display***

Es una API pública usada para desplegar herramientas en IPython[22], dentro de sus clases se encuentra una para audio, con la cual se crea un objeto que ante una entrada o una función retornará como resultado controles de audio que se mostrarán en el frontend[6].

7. ***Lib 7: librosa***

Es un paquete de Python para el análisis de música y audio. Provee los bloques funcionales que permiten crear sistemas de recuperación de información musical. Dentro de sus capacidades está la de cargar audio desde el disco, computar varias representaciones de espectrogramas y una variedad de herramientas usadas comúnmente para el análisis musical.

8. ***Lib 8: pandas***

Es una librería de Python especializada en el manejo y análisis de estructuras de datos. Sus principales características son:

- Definir nuevas estructuras de datos basadas en arreglos de la librería NumPy pero con nuevas funcionalidades.
- Permite leer y escribir de varias fuentes de datos como archivos csv, Excel, Bases de datos SQL.
- Acceso a datos mediante índices o nombres para filas y columnas.
- Métodos para reordenar, dividir y combinar conjuntos de datos.
- Trabaja con datos de series de tiempos.

9. ***Lib 9: keras***

Es una API diseñada para la implementación y manejo de redes neuronales de forma sencilla, desarrollada en Python y es de código abierto.
Keras maneja redes neuronales estándar, convolucionales y redes neuronales recurrentes.

10. ***Lib 10: Matplotlib y Numpy***

Matplotlib permite graficar los datos en figuras, las cuales se puede tener uno o muchos ejes, una área dónde los puntos pueden ser especificados en términos de coordenadas x-y. Dentro de las entradas esperadas para la ejecución de la función de ploteo se encuentran arreglos provenientes de la librería de NumPy[11].

La combinación de estás dos librerías[10][12] proveerá la capacidad de plotear un archivo de audio de extensión .wav en Python, pero se debe asegurar que el archivo sea monocanal.

11. ***Lib 11: essentia.standard.Monoloader***

Esta es una librería de código abierto para obtener información de archivos de música para análisis de audio. dentro de las funcionalidades con las que cuenta essentia[7][8][9] están:

- Cargar archivos de audio.
- Ejecutar operaciones numéricas como FFT.
- Graficar resultados.
- Exportar los resultados del análisis a un archivo.

Particularmente, Monoloader permite obtener un nuevo archivo de audio mezclado y con un remuestreo a una tasa de muestreo específica.

Repositorio para desarrollo

El desarrollo persiste en el archivo en el repositorio público, cuya URL de acceso es: https://github.com/jmiguez/reconocimiento_de_hablantes.git , archivos: "*preprocesamientoV0020102.ipynb*" y "*procesamiento_V0020204.ipynb*"

Desarrollo de los fragmentos de código:

- Instalar dependencias que se usarán en el entorno Google-Colab
- Importar las librerías
- Inicialización y seteos

En este caso, hay filas comentadas que fueron utilizadas para ejecutar el código en un entorno local sobre Anaconda[13][14][15] y Jupyter notebook[16].
A la variable V_VIDEO_LINK se le asigna la ruta del video de Youtube.
A la variable V_LABELS_DESDE_EXCEL, se le asigna el nombre del archivo con los nombres de los hablantes, y el tiempo en segundos de inicio y fin donde se encuentre hablando dicha persona en el video de Youtube.

	A	B	C
1	segs_ini	segs_fin	Nombre_Senador
2	1857	2091	MariaBelenTapia
3	2125	2619	LuciaCorpacci
4	2633	2718	LuisPetcoffNaidenoff
5	2723	2763	PabloBlanco
6	2773	2911	CarolinaLosada
7	2924	3055	AnabelFernandezSagasti

- Función cargarAudio()

Desde la URL de Youtube, descarga únicamente el audio en formato MP4.

- Función convertirMP4aWAV

Convierte el formato de audio MP4 a un archivo con formato WAV[30].

- Función identificarArchivoAudioSinPreProcesar

Devuelve la ruta donde se encuentra el archivo WAV sin pre-procesar.

- Función eliminarArchivoMP4

Permite eliminar el archivo original[3] en formato MP4, para liberar espacio en el disco en la nube (unidad-drive del usuario que está ejecutando la notebook)

- secuencia de procesamiento del primer bloque principal (llamada a funciones)

- Bloque de lectura de archivo Excel con los datos de los hablantes

Usando la librería Pandas, se logró parametrizar el nombre de cada hablante, de manera que se considere como el nombre de cada etiqueta.
Además, para cada hablante, también se parametriza el tiempo desde y tiempo hasta del audio de cada hablante.

- Función normalizarAudioCrudo

Normaliza el audio a una potencia de -3dB[29]. Esta operación no modifica los atributos del habla de la persona, sino que nivela la amplitud de la señal de audio para que sonidos suaves sean más fuertes, y sonidos muy fuertes sean suavizados.
Toma la muestra de audio completa, y recorta sólo la porción de tiempo correspondiente al hablante que se esté procesando[35].
Recorta la señal[23][24][25] cuya amplitud no supere el 0.5% y su duración supere los 0.1 segundos, de manera que elimina silencios[4][5].
Informa cuáles son los cambios que se realizaron a la señal de audio, y la guarda en disco en un archivo con el nombre de la etiqueta extraído del archivo Excel del bloque anterior.

- dividirArchivoNormalizadoEnChuncks1seg

La función se encarga de dividir el archivo normalizado del paso anterior, en muestras de 1 segundo[2]. Este es esencial para poder utilizar transformadas rápidas de Fourier. Si no existe una carpeta con el nombre de la etiqueta, la crea.
Finalmente, mueve todos los archivos normalizados de 1 segundo, a la carpeta correspondiente a su etiqueta.
El resultado es un directorio path/nombre_de_hablante, que contiene múltiples archivos de 1 segundo normalizados, de dicho hablante.

- Secuencia de procesamiento del segundo bloque principal (llamada a funciones)

Este método automático permite obtener la muestra normalizada de los hablantes, para que el procesamiento de las mismas devuelva predicciones confiables.

Equipo de procesamiento Local

Procesador: AMD® A4-4000 apu with radeon(tm) hd graphics × 2

Capacidad de disco: 980,2 GB

Memoria: 10GiB

Nombre y Tipo de SO: Ubuntu 22.04 LTS 64 bits

Gráficos: AMD® Aruba

Software

python3

pip + pip3

Anaconda Navigator 2.2.0 con Jupyter Notebook

Equipo de procesamiento en la nube

Google Colab - El desarrollo quedó en: https://colab.research.google.com/drive/1SLaFajUzAviDuXbAkllIdPHm-vg8_TF#scrollTo=d506314b

Espacio de almacenamiento por defecto= 15GiB

1. Preparación de datos

El conjunto de datos de prueba[36][37] se descargó desde:

i. Diputados. Proyecto de Ley con el Fondo Monetario Internacional / sesión extraordinaria del 08-Mar-2022: https://youtu.be/O7M6qsPd0rI

ii. Senadores - SESIÓN ESPECIAL 10-Ago-22: https://youtu.be/8Jpwgw-kwQA

iii. Cristina Fernández de Kirchner - Derecho de Defensa. Causa vialidad. 23-Ago-2022: https://youtu.be/aNtbi1tSiWo

iv. Cristina Fernández de Kirchner - En el año 2000: https://youtu.be/2t8h8E8-lkg

v. Fátima Florez imitando a Cristina - 05-Dic-2020: https://youtu.be/I6t8goOfbXI

vi. Val Kilmer y su nueva voz generada con Inteligencia Artificial - 2022: https://youtu.be/OSMue60Gg6s

vii. Val Kilmer y su voz original en la película TOP GUN 1 (usada para predicción)

https://youtu.be/zq8rHTcveyA

viii, Morgan Freeman y su voz generada con Inteligencia Artificial mediante Deep Fake en 2020:

https://youtu.be/F4G6GNFz0O8

ix. Morgan Freeman, voz original:

https://youtu.be/TyyzL-AHPCA

x. Donald Trump, voz original:

https://youtu.be/gP-JkWgLPNU

xi. Donald Trump, y su voz generada con Inteligencia Artificial mediante Deep Fake en 2020:

https://youtu.be/aPp5lcqgISk

xii. Muestras de ruido de fondo, con 2 carpetas y un total de 6 archivos. Estos archivos duran más de 1 segundo (y originalmente no se muestrearon a 16000 Hz, por eso se volverán a muestrear a 16000 Hz). Durante este proyecto, se usarán los 6 archivos de ruido, para crear 354 muestras de

ruido de 1 segundo de duración que se usarán para el entrenamiento.

Las 2 categorías se ordenan en 2 carpetas. Audio: con todas las carpetas de muestra de voz por hablante y Ruido: con todas las muestras de ruido.

2. Preparación de ruido

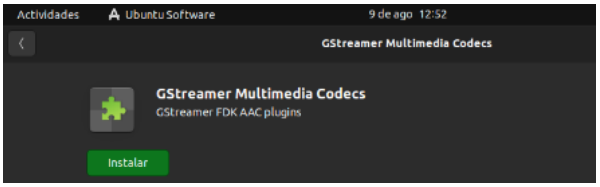
Se realizó la carga de todas las muestras de ruido muestreadas a 16000 hertz.

Se dividieron las muestras de ruido en fragmentos de 1 segundo de duración cada una, a 16000 hertz.

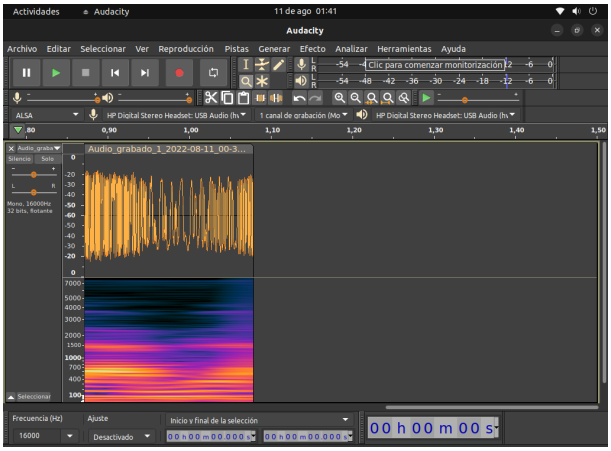
Problemas encontrados

No se puede reproducir audio con formato mpeg4 descargado de YouTube

Instalar codec plugin “GStreamer Multimedia Codecs” para reproducir archivos de audio mpeg4 descargados desde youtube.



Luego, instalar Audacity como herramienta de reproducción de archivos de audio[21].



Error con permisos de ejecución, durante la creación del entorno local con Anaconda.

El archivo *nbserver-20665-open.html* no tenía permitido su acceso. El navegador mostraba la leyenda “Acceso denegado al archivo”.

Desde una consola, se generó el archivo de configuración “jupyter_notebook_config.py” en el directorio “~/.jupyter” con el siguiente comando:
$ jupyter notebook --generate-config
y luego se editó el mismo para configurar el parámetro “c.NotebookApp.use_redirect_file” al valor “False”
Finalmente, se otorgaron permisos de ejecución sobre los siguientes directorios: “~/.local/share/jupyter” “~/.jupyter” al usuario que ejecuta Anaconda.

Error de dimensionado de pantalla al abrir Anaconda
“Can't detect system scaling factor settings for primary monitor.” Es un problema de configuración[18].
Se editó el archivo “anaconda-navigator.ini”[19], que está en el directorio “.anaconda/navigator”, y se configuró la propiedad “enable_high_dpi_scaling” al valor “False”[20].

Mensajes de advertencia al abrir Anaconda.

Para resolver los mensajes sobre Links que no funcionan, se debe instalar la última versión de gcc desde un terminal, con los siguientes comandos:

$ conda install libgcc

Buscar los links que no funcionan en los mensajes de pantalla cuando se desplegó Anaconda:

$ sudo find / -wholename "*conda*/**/libstdc++.so*"

Eliminar los links que no funcionan del directorio específico:

$ rm /home/jmmiguez/anaconda3/lib/libstdc++*

Cambiar en navegador por defecto de Firefox a Chrome

Editar desde línea de comandos el archivo: “~/.jupyter/jupyter_notebook_config.py” para quitar el comentario de la entrada “c.NotebookApp.browser = ‘’”

Luego, buscar desde dónde se ejecuta Google Chrome en el Sistema Operativo local: en este caso: ‘/usr/bin/google-chrome’ y reemplazar en la línea que se está editando: “c.NotebookApp.browser = ‘/usr/bin/google-chrome’”[17]

Escasos recursos de memoria y de disco en Google Colab.

Una de las librerías de ploteo, comenzó a consumir más memoria de lo esperado al graficar las señales de audio en función del tiempo, y también al graficar el espectrograma. Como resultado, la sesión se quedó sin memoria y el proceso terminó abruptamente, causando la pérdida de información procesada hasta el momento.

Durante la conversión de archivos MP4 a WAV, el espacio en disco se quedó sin espacio. Esto causó una interrupción abrupta, impidiendo continuar con el procesamiento.

Debido a esto, se descartó usar Google Colab hasta resolver ambos problemas. Mientras tanto, se usó Anaconda, con Jupyter Notebook, en un entorno local.

Luego, se eliminaron las librerías de ploteo del desarrollo para ganar performance y continuar utilizando Google Colab.

Límite de ejecuciones en Google Colab

Con el fin de ejecutar más de una notebook al mismo tiempo con procesamiento GPU o TPU, se utilizaron 2 terminales de igual Hardware. Sin embargo, para limitar el uso de recursos, Google Colab no permite usar más de un tipo de procesador al mismo tiempo para la misma cuenta (1 TPU, 1 GPU) y además, luego de la décima prueba, Google Colab no permitió volver a tomar otra sesión con procesamiento por TPU ni GPU. Esto provocó demoras aproximadas de 4 horas para volver a poner en marcha las pruebas.

Para aprovechar el tiempo, se utilizó el procesamiento genérico para descargar archivos de audio por Deep Fake para terminar las pruebas.

Método predict, entrenando la muestra de un hablante.

Se procesaron los archivos de un sólo hablante para entrenamiento y validación. Luego, se ingresó el audio de un hablante diferente y el modelo predijo que el hablante era compatible con certeza absoluta, al hablante que se usó para entrenar el modelo (resultado equivocado)

```
history = model.fit(
    train_ds,
    epochs=EPOCHS,
    validation_data=valid_ds,
    callbacks=[earlystopping_cb, mdlcheckpoint_cb],
)
```

```
Epoch 1/3
7/7 [==============================] - 151s 21s/step - loss: 0.7611 - accuracy: 0.8513 - val_loss: 0.0000e+00 - v
l_accuracy: 1.0000
Epoch 2/3
7/7 [==============================] - 175s 24s/step - loss: 0.0000e+00 - accuracy: 1.0000 - val_loss: 0.0000e+00
- val_accuracy: 1.0000
Epoch 3/3
7/7 [==============================] - 167s 23s/step - loss: 0.0000e+00 - accuracy: 1.0000 - val_loss: 0.0000e+00
- val_accuracy: 1.0000
```

```
print(model.evaluate(valid_ds))
```

```
3/3 [==============================] - 5s 1s/step - loss: 0.0000e+00 - accuracy: 1.0000
[0.0, 1.0]
```

fig.1 Entrenamiento y evaluación, con 1 categoría.

```
1/1 [==============================] - 0s 76ms/step
CristinaFernandezDeKirchner
1) 1.0 109 CristinaFernandezDeKirchner
2) predicción(Voz en Vivo) = CristinaFernandezDeKirchner ... y_pred=[109] index=132
3) registro (Voz en Vivo) = /home/jmmiguez/proyectoAudio/audio//audio/VozViva/VozViva.wav - [109]
```


fig 2 Predicción del hablante distinto al entrenado.

Para descartar malas prácticas que conducen a este tipo de resultados (como sobre-entrenamiento), se volvió a ejecutar cada conjunto de prueba con diferentes valores de VALID_SPLIT (desde 0,1 hasta 0,9 con intervalos incrementales de +0,1). Además, se modificó la cantidad de archivos de muestra, entre 50, 100, 200 y 400 archivos. Finalmente, todas las pruebas devolvieron el mismo resultado.

El método model.predict devuelve un array[categorías] de probabilidades de cercanía a cada categoría. Si el modelo entrena con 1 categoría, entonces la certeza será total con cualquier dato nuevo que se intente predecir. Si el modelo entrena con más de 1 categoría, entonces la probabilidad puede tender a 1 (uno) en el caso de certeza, para manifestar compatibilidad, o a 0 (cero) en caso contrario.

Voz de un imitador vs. Voz real

Para aportar resultados confiables, durante la prueba se tomaron archivos de audio de un hablante (Cristima Fernández de Kirchner - voz actual y voz de hace 20 años), y también se entrenó al modelo con archivos de audio de su imitador (Fátima Florez)

Al realizar la predicción, el modelo pudo diferenciar con predicciones del orden del 98,5% si la voz era compatible con el imitador o con la voz real.

Rendimiento

Uno de los objetivos alcanzados, fue la optimización del tiempo de entrenamiento y validación. En primera instancia, tomando el conjunto de 140 hablantes, con procesamiento local y Hardware / Software mencionados, para VALID_SPLIT=0,1, el tiempo de cada EPOCH fue 10702 segundos, equivalente a 2.97 horas. Esto quiere decir, que con este equipo de procesamiento, 3 EPOCHS tardaron 30024 segundos, equivalente a 8.34 horas. Luego, accuracy fue muy bajo <10%, dando por descartado este tipo de prueba.

En iguales condiciones, pero en Google Colab, con GPU, cada EPOCH tardó 5:36 horas, y se interrumpió el procesamiento.

Iguales condiciones, para TPU, el tiempo de cada EPOCH fue de 0,38 horas. Sin embargo, accuracy para 3 EPOCHS no fue aceptable <10% fue inaceptable.

El tiempo de procesamiento debe finalizar antes de las 12 horas, ya que es el tiempo límite fijado en Colab para mantener la sesión activa. Fuera de ese tiempo, el entrenamiento se debe repetir. Por eso, a 30 minutos por EPOCH, 10 EPOCHS equivalen a 300 minutos (o 50 horas, que no son aceptables)

Usando VALID_SPLIT = 0.8 y EPOCHS = 10 con TPU, el procesamiento de 1 EPOCH demoró hasta 11 minutos. 11 minutos x 10 EPOCHS = 110 minutos = 1,8 horas. Sin embargo, VALID_SPLIT = 0,8 implica menor cantidad de archivos de entrenamiento, pero mayor cantidad de archivos para validación. En este otro caso extremo, la validación demoró más de 1 hora y el proceso se finalizó intencionalmente por no ser viable.

Menor cantidad de hablantes

Con 45 hablantes (en lugar de 140), VALID_SPLIT = 0.1 y EPOCHS = 5, se estimó estimó 1 EPOCH = 18:50 minutos y 5 EPOCHS ~ 1,57 horas, siguiendo estos resultados:

en 1 EPOCH se alcanzó ~ 0.48 accuracy

en 2 EPOCHS se alcanzó ~ 0.54 accuracy

en 3 EPOCHS se alcanzó ~ 0.56 accuracy

en 4 EPOCHS se alcanzó ~ 0.64 accuracy - 19 minutos promedio por EPOCH

en 5 EPOCHS se alcanzó ~ 0.65 accuracy

En consecuencia, ciertas voces que se usaron en el entrenamiento, fueron reconocidas casi con certeza. Las voces que no se usaron para entrenar, no fueron reconocidas. Algunas voces que fueron entrenadas, no se reconocieron con certeza (Errores tipo I y II)

Luego de realizar varios ajustes a modo de prueba y error, empíricamente se identificó que 5 hablantes con 500 archivos en promedio por hablante, son necesarios para conseguir que el modelo consiga realizar predicciones aceptables.
Con VALID_SPLIT = 0.1 y EPOCHS = 50, promedio 217 segundos o 3,6 minutos por EPOCH y Accuracy ~ 0.95. Todas las voces que se usaron en el entrenamiento, fueron reconocidas casi con certeza.

Las voces que no se usaron para entrenar, no fueron reconocidas.

En una nueva prueba, TPU, VALID_SPLIT = 0.1, EPOCHS = 50 ⇒ alcancé modelo estable en 16 EPOCHS, en 3448 segs (57.5 minutos) con loss: 0.1525 - accuracy: 0.9494.

Durante las pruebas de performance y optimización, se utilizaron conjuntos de datos con menos de 50 archivos por hablante para validar la efectividad frente a Deep Fake. Para los casos de los hablantes Val Kilmer y Morgan Freeman, esta cantidad de archivos provocó un sobreentrenamiento del modelo, devolviendo predicciones absurdas.

Optimizando VALID_SPLIT de 0,1 a 0,9

NOTEBOOK	VALID_ SPLIT	EPOCHS	USED EPOCHS	ELAPSED _TIME	MEAN_ EPOCH_ TIME	accuracy	loss	Backend
procesamiento_V0020101.ipynb	0,1	50	16	3448	215,5	0,9494	0,1525	TPU
procesamiento_V0020103.ipynb	0,2	50	16	1825	114,1	0,9376	0,1984	GPU
procesamiento_V0020110.ipynb	0,25	50	27	5502	203,7	0,9394	0,1714	-
procesamiento_V0020104.ipynb	0,3	50	17	967	56,8	0,9356	0,1806	GPU
procesamiento_V0020111.ipynb	0,35	50	17	3225	189,7	0,9269	0,2204	TPU
procesamiento_V0020105.ipynb	0,4	50	22	3760	62,7	0,9309	0,2062	TPU
procesamiento_V0020102.ipynb	0,5	50	37	4532	122,5	0,9483	0,1667	TPU
procesamiento_V0020106.ipynb	0,6	50	36	2463	68,4	0,9307	0,2040	GPU
procesamiento_V0020107.ipynb	0,7	50	33	4162	126,1	0,9232	0,2493	-
procesamiento_V0020108.ipynb	0,8	50	20	1838	91,9	0,8288	0,4750	-
procesamiento_V0020109.ipynb	0,9	50	35	3637	103,9	0,8769	0,4366	-

Tras haber ejecutado diferentes pruebas de carga, se pudo observar que se consiguen óptimos resultados con GPU ; VALID_SPLIT=0,2 ; 5 hablantes con 500 archivos de audio en formato WAV, de 1 segundo en promedio, con muestreo a 16000Hz (1 canal, mono) -3dB de potencia, nivelado y sin silencios.

⇒ El modelo quedó estable en 16 EPOCHS, en 1825 segs (30.41 minutos) con loss: 0.1984 - accuracy: 0.94

Deep Fake

La voz generada mediante Inteligencia Artificial (Deep Fake) se utilizó para entrenar el modelo y predecir si el mismo es vulnerable a este tipo de engaño.

El conjunto de datos para el hablante Donald Trump, alcanzó la cantidad de archivos de voz de la persona real esperada. Loss: 0.089 - accuracy: 0.9674. Sin embargo, el modelo predijo que la voz artificial era compatible con alta certeza con la voz real de Donald Trump.

Conclusiones

Alcanzamos un accuracy elevado más rápido que antes de tener implementado el pre-procesamiento. Es decir, empíricamente se observó que bajo las condiciones antes mencionadas, accuracy superó 90% luego del 3er. EPOCH, siendo que ese umbral antes se lograba luego de la 10ma. EPOCH sin pre-procesamiento.

El modelo tiene algunas limitaciones, que no se salvan en etapa de preprocesamiento:

- No es posible entrenar con la voz de un sólo hablante y predecir compatibilidad con otros hablantes diferentes, ya que es un modelo supervisado. Si hay 5 hablantes y ninguno de ellos corresponde a la voz a predecir, entonces la predicción será del orden inferior a 98%.
- Ante Deep Fake, con un modelo bien entrenado, el modelo falla. Sin embargo, detecta bien a los imitadores (personas físicas)

En el resto de los casos, los resultados de las predicciones fueron acertadas y reproducibles. Es decir, se pudo repetir el experimento en distintos entornos y alcanzar el mismo resultado exitoso independientemente de la plataforma y sesión.

Futuras líneas de investigación

A futuro se pretende resolver errores por Deep Fake.

Mejorar el modelo, utilizándolo como medio de soporte en peritajes informáticos.

Referencias

1. Documentos electrónicos

[1] Pydub - (2022) http://pydub.com/

[2] How to splice an audio file into 1 sec splices in python? - (2021) https://stackoverflow.com/questions/36799902/how-to-splice-an-audio-file-wav-format-into-1-sec-splices-in-python?rq=1

[3] os.path - Common pathname manipulations - (2022) https://docs.python.org/3/library/os.path.html

[4] Audio Processing and Remove Silence using Python - (2020) https://ngbala6.medium.com/audio-processing-and-remove-silence-using-python-a7fe1552007a

[5] Remove silents using VAD - (2020) https://malaya-speech.readthedocs.io/en/latest/remove-silent-vad.html

[6] pyAudioAnalysis_An_Open-Source_Python_Library_for - (2015) https://www.researchgate.net/publication/286637817_pyAudioAnalysis_An_Open-Source_Python_Library_for_Audio_Signal_Analysis

[7] Using Audio in IPython - (2022) https://notebook.community/imsparsh/stanford-mir/notebooks/ipython_audio https://notebook.community/imsparsh/stanford-mir/notebooks/ipython_audio#Playing-Audio

[8] Essentia.standard Python tutorial - (2022) https://notebook.community/ChristianFrisson/essentia/src/examples/tutorial/essentia_tutorial_standard

[9] Essentia Python tutorial - (2022) https://github.com/MTG/essentia/blob/master/src/examples/python/essentia_python_tutorial.ipynb https://notebook.community/ChristianFrisson/essentia/src/examples/tutorial/essentia_tutorial_standard

[10] Basic Sound Processing in Python | SciPy 2015 | Allen Downey (2015) - https://www.youtube.com/watch?v=0ALKGR0I5MA https://nbviewer.org/github/AllenDowney/ThinkDSP/blob/master/code/scipy2015_demo.ipynb https://tinyurl.com/scipy15dsp

[11] Python Program - Plot A Wave Audio File | Matplotlib & NumPy Tutorial - (2020) https://www.youtube.com/watch?v=4rzpMA6CUPg

[12] ThinkDSP package installation in Anaconda - (2017) https://www.youtube.com/watch?v=feR3L8gcYZc

[13] Instalar Anaconda en Ubuntu - (2021) https://ourcodeworld.co/articulos/leer/1609/como-instalar-anaconda-en-ubuntu-2004

[14] Anaconda Distribution - (2022) https://www.anaconda.com/products/distribution

[15] Ancaconda Documentation. Verifying your installation - (2022) https://docs.anaconda.com/anaconda/install/verify-install/#nav-problems

[16] How to set Jupyter Notebook to open on browser automatically - (2021) https://stackoverflow.com/questions/55756151/how-to-set-jupyter-notebook-to-open-on-browser-automatically

[17] Anaconda Jupyter doesn't open in a browser - (2018) https://stackoverflow.com/questions/53982363/anaconda-jupyter-doesnt-open-in-browser

[18] Issue with anaconda-navigator: WARNING linux_scaling.get_scaling_factor_using_dbus:27 #12133 - (2020) https://github.com/ContinuumIO/anaconda-issues/issues/12133

[19] Zoom and Anaconda Navigator have weird scaling - (2020) https://askubuntu.com/questions/1252036/zoom-and-anaconda-navigator-have-weird-scaling

[20] libGL error Anaconda Navigator Launch Ubuntu 22.04 #12889 - (2022) https://github.com/ContinuumIO/anaconda-issues/issues/12889

[21] Mic y cámara - (2020) https://www.youtube.com/watch?v=aLPnmeMtrUY

[22] Using Audio in IPython - (2022) https://notebook.community/imsparsh/stanford-mir/notebooks/ipython_audio#Playing-Audio

[23] Audio Processing and Remove Silence using Python - (2020) https://ngbala6.medium.com/audio-processing-and-remove-silence-using-python-a7fe1552007a

[24] A New Silence Removal and Endpoint Detection Algorithm for Speech and Speaker Recognition Applications - (2003) https://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.608.8649&rep=rep1&type=pdf

[25] The SoX of Silence - (2009) https://digitalcardboard.com/blog/2009/08/25/the-sox-of-silence/

[26] sox_and_ffmpeg.ipynb - (2022) https://colab.research.google.com/github/stevetjoa/musicinformationretrieval.com/blob/gh-pages/sox_and_ffmpeg.ipynb#scrollTo=3TkyJuXBmD_M

[27] SoX - Sound eXchange | HomePage - (2015) http://sox.sourceforge.net/

[28] Piping SoX in Python - subprocess alternative? - (2022) https://pypi.org/project/sox/

[29] AUDIO DATA AUGMENTATION - (2022) https://pytorch.org/audio/main/tutorials/audio_data_augmentation_tutorial.html

[30] Building Machine Learning Systems with Python: Explore machine learning and … - (2018) https://books.google.com.ar/books?id=owZnDwAAQBAJ&pg=PA268&lpg=PA268&dq=run+sox+command+from+notebook&source=bl&ots=T40JBzc17T&sig=ACfU3U1znnzvQixxOI4Oj-NT8jidjTe27g&hl=es&sa=X&ved=2ahUKEwjajcOQ_rX5AhXQkZUCHa06C_wQ6AF6BAg9EAM#v=onepage&q=run%20sox%20command%20from%20notebook&f=false

[31] Audio Handling Basics: Process Audio Files In Command-Line or Python - (2020) https://hackernoon.com/audio-handling-basics-how-to-process-audio-files-using-python-cli-jo283u3y

[32] PYSOX: LEVERAGING THE AUDIO SIGNAL PROCESSING POWER OF SOX IN PYTHON - (2016) https://wp.nyu.edu/ismir2016/wp-content/uploads/sites/2294/2016/08/bittner-pysox.pdf

[33] Source code for sox.transform - (2016) https://pysox.readthedocs.io/en/latest/_modules/sox/transform.html

[34] Pysox: Python Wrapper Around Sox - (2022) https://morioh.com/p/8ac4dd1bbbcc

[35] Denoising Data with FFT [Python] (buen ejemplo, filtro ruido) - (2020) https://www.youtube.com/watch?v=s2K1JfNR7Sc https://www.youtube.com/watch?v=spUNpyF58BY

[36] Compartimos nuestro Common Voice: Mozilla lanza el segundo mayor conjunto de datos de voz público - (2017) https://blog.mozilla.org/press-es/2017/11/29/commonvoicemozilladatosvoz/

[37] Datasets: Common Voices - (2022) https://commonvoice.mozilla.org/es/datasets

[38] Extract audio from YouTube video using Python - (2022) https://blog.balasundar.com/extract-audio-from-youtube-video-using-python https://github.com/SBalaSundar/youtube_audio_extractor

[39] PyTube3 - Documentación - (2022) https://pypi.org/project/pytube3/ https://github.com/YouTubeDownload/YouTubeDownload

[40] pysox Documentation - (2021) https://buildmedia.readthedocs.org/media/pdf/pysox/latest/pysox.pdf

[41] Reconocimiento_forense_de_voz - (2022) https://github.com/jmiguez/reconocimiento_de_hablantes.git