MEMORIA PRÁCTICA:
Identificación de pacientes de ELA a través de la voz mediante redes neuronales

Aplicaciones de la Biometría de la Voz

Curso 2021/22

GRUPO 15:

Víctor Morcuende Castell

Guillermo Nájera Lavid

Antonio Ruiz García

ÍNDICE DE CONTENIDOS

1. Introducción

En esta memoria vamos a explicar de manera detallada los procesos que se han llevado a cabo para aplicar la detección de la enfermedad neurodegenerativa conocida como Esclerosis Lateral Amiotrófica mediante el uso aplicado de redes neuronales a ficheros de audio que contienen locuciones de pacientes y personas sanas emitiendo las vocales “a” e “i”.

Para esto, se nos han proporcionado datos de entrenamiento consistentes en 128 ficheros, de los cuales 62 son pacientes de ELA y 66 son personas aparentemente sanas. Estos ficheros, están divididos, a su vez, en dos ficheros para cada sujeto, uno pronunciando la vocal “a” y otro pronunciando la vocal “i”, tratando de mantener unos tonos y niveles de volumen lo más constantes y homogéneos posible para así evitar cualquier tipo de desviación o interferencia en el experimento.

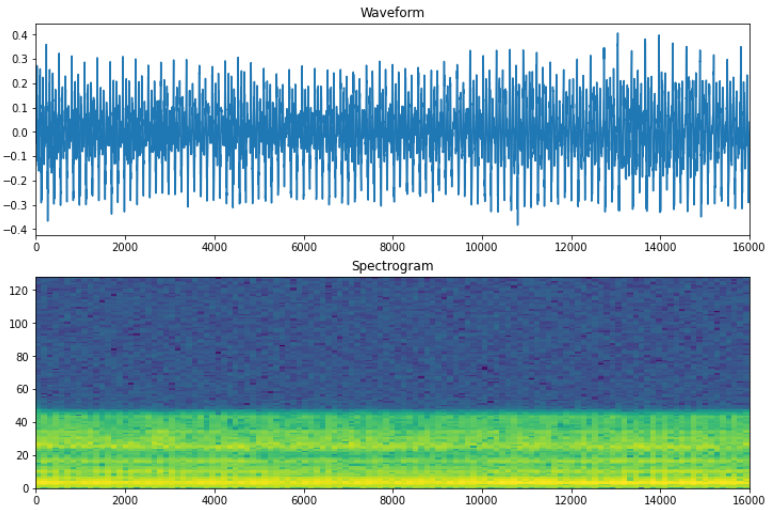
Así mismo, cabe destacar que en el subconjunto de pacientes se incluyen datos de 17 hombres, de entre 40 y 69 años de edad, y 14 mujeres de entre 39 y 70 años, y en el subconjunto de sujetos sanos se incluyen datos de 13 hombres con edades comprendidas entre los 34 y los 80 años de edad y 20 mujeres con edades comprendidas entre 37 y 68 años. Estos datos se han obtenido de manera homogénea para todos los pacientes, con una tasa de muestreo de 44100 Hercios, mediante el uso de el micrófono mono de unos auriculares corrientes conectados a un Smartphone, generando un fichero sin compresión de 16 bits de profundidad de tipo PCM. Por último, la duración media de las grabaciones de los ficheros de los pacientes es de 3,7 segundos, mientras que para los sujetos sanos es de 4,1 segundos.

2. Desarrollo

Con relación a la distribución de los archivos de audio usados en entrenamiento, validación y test, desarrollamos un algoritmo que asignaba un 70% de las muestras para entrenamiento, un 15% de los datos para validación y por último un 15% de las muestras para test. En cada asignación, se repartía de forma equitativa pero aleatoria a los pacientes y a los sujetos sanos, de tal manera que se pudiera encontrar una cantidad considerable de cada uno de ellos en cada grupo. Además, también debimos asegurarnos de que los audios de la vocal "a" e "i" de un determinado sujeto estuvieran en un solo conjunto de datos. Esto se debe a que si no realizábamos esta tarea los resultados del clasificador podrían quedar sesgados, ya que el clasificador podría reconocer a un mismo paciente en distinto grupos.

En cuanto a la realización del experimento en cuestión, primero implementamos una serie de funciones auxiliares que se encargarían de decodificar los archivos de audio, calcular el espectrograma de cada uno de ellos, y asociar a cada uno de los espectrogramas la etiqueta correspondiente para poder añadirlo a los datasets de entrenamiento, validación y test. Para la decodificación de los archivos de audio, se ha hecho uso de la función incluida en TensorFlow "decode_wav", incluyendo el valor 44100*3 (ya que al tener una tasa de frecuencia de 44100 hercios, para poder obtener 3 segundos de muestras debemos multiplicarla por 3) como parámetro "desired_samples". Se tomó la decisión de incluir este parámetro para poder obtener el mismo número de muestras para todos los ficheros de audio, ya que, al no tener todos la misma longitud, de no haber normalizado el tamaño de los archivos de onda, el modelo habría fallado al compilar. De esta forma aseguramos la obtención de un mínimo de 3 segundos para cada paciente, longitud más que suficiente para el correcto entrenamiento del modelo.
A continuación, para obtener los espectrogramas a partir de las muestras de audio, se ha hecho uso del método stft de TensorFlow (short time fourier transform), especificando un frame_length de 255 y un frame_step de 128. La razón por la que hemos elegido estos valores es debida a que son los que arrojaban mejores resultados, probablemente a causa del tamaño elegido para las muestras de audio.

Seguidamente, podemos observar una onda de audio junto a su correspondiente espectrograma, ambos representados en un intervalo de tiempo igual a 1 segundo de duración.

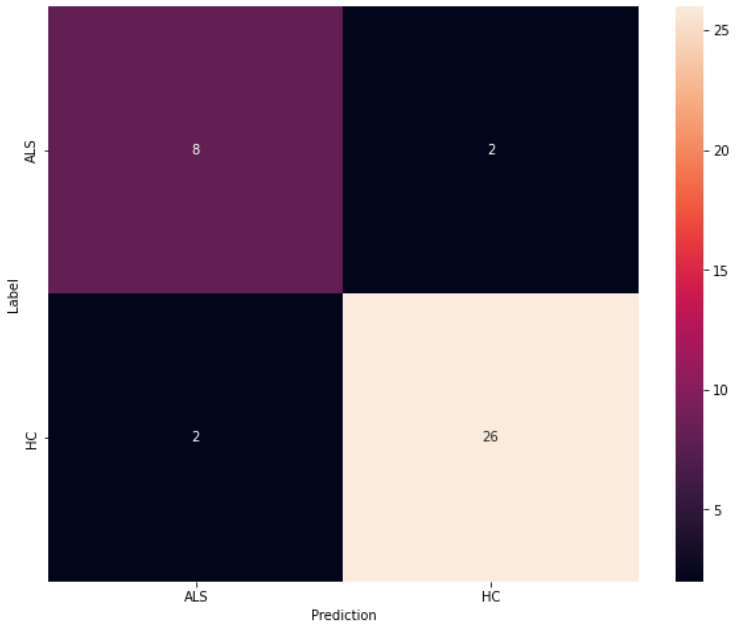


3. Implementación

Con respecto a nuestra implementación, en un principio probamos a usar la red suministrada por el ejemplo simple_audio de TensorFlow. Sin embargo, esta arquitectura no proporcionaba unos resultados satisfactorios, ya que el modelo no era capaz de etiquetar correctamente ninguna muestra, ni siquiera el dataset de Train. Por esta razón nos decidimos a emplear un modelo más potente, que hiciera uso de más capas. De esta forma fuimos capaces de incrementar la tasa de acierto para el dataset de Train hasta acercarnos al 100% de aciertos.

No obstante, al haber potenciado de tal manera el modelo para que aprendiera a ajustarse a los datos de Train, este empezó a dar preocupantes muestras de overfitting. Tras descubrir esta nueva situación, nos dispusimos a emplear todos los recursos a nuestra disposición para tratar de mitigarla. En primer lugar, intentamos incrementar el valor de la capa Dropout, sin éxito alguno, ya que esto solo provocaba un incremento del error de Train y muy poca o nula mejoría en la precisión de Validación y Test. Esta resultó ser una situación tremendamente complicada y tediosa para nosotros, ya que nos fue complicado llegar a averiguar la razón por la cual nuestro modelo no mejoraba. Sin embargo, estudiando el funcionamiento de la regularización, decidimos implementar más capas Dense, incluyendo un kernel de regularización L1 y L2 y un tamaño de 32 unidades, y gracias a ello logramos mejorar los resultados de precisión en Validación considerablemente. No obstante, los resultados de Test parecían seguir estancados, lo que nos llevó a implementar en cada capa convolucional de la red neural un kernel de regularización L2, lo que contribuyó a alcanzar una tímida mejora de los resultados de Test.

Sin embargo, esto no fue suficiente, y tras estudiar detenidamente nuestras opciones, decidimos suplir el problema de la baja disponibilidad de datos utilizando grabaciones de las vocales “a” e “i” de la base de datos Saarbruecken Voice Database. Gracias a esta base de datos, conseguimos añadir un total de 4 pacientes enfermos y 162 sujetos sanos, comprendiendo estos edades y sexos variados. Asimismo, pese a que esta acción supone un desequilibrio hacia los sujetos sanos en el experimento, lo que en un principio debería sesgar al clasificador, a la hora de clasificar a cada paciente la red neuronal realizó un mejor trabajo, no solo clasificando a sujetos sanos sino también a pacientes enfermos. Estos resultados se pueden observar en la siguiente figura, sacada de una de las iteraciones realizadas durante el análisis de los resultados:



4. Resultados

En la tabla que se ilustra a continuación se pueden observar los resultados de los experimentos realizados con 40 iteraciones y patience = 15 para tanto los espectrogramas clásicos como para los que usan bandas mel y MFCC.

	Espectrograma	Espectrograma con bandas mel y MFCC
Datos originales	Precisión media: 51.805	Precisión media: 60.417
	Precisión media ALS: 29.25	Precisión media ALS: 41
	Precisión media HC: 80	Precisión media HC: 84.688
	Batch óptimo: 4	Batch óptimo: 32
Datos ampliados	Precisión media: 76.447	Precisión media: 85.088
	Precisión media ALS: 73.25	Precisión media ALS: 78.0
	Precisión media HC: 77.589	Precisión media HC: 87.619
	Batch óptimo: 4	Batch óptimo: 32

Después de realizar los experimentos oportunos y tras analizar los resultados que estos han ofrecido, hemos podido observar como el hecho de utilizar bandas mel y las características MFCC mejora considerablemente el resultado, no solo con datos ampliados, sino también con los datos originales. Además, ha quedado claro como el hecho de ampliar los datos de entrenamiento mejora sustancialmente el rendimiento de la red neuronal, que, aunque haya quedado sensiblemente sesgado, proporciona tasas de acierto sustancialmente mayores incluso para pacientes enfermos.

5. Conclusión

Por último, tras realizar este experimento podemos concluir que es posible construir un clasificador que identifique a pacientes enfermos de ELA. No obstante, aunque la red neuronal creada aporta una información que pueda resultar útil para un diagnóstico rápido, no se puede clasificar estos resultados de forma determinante ya que, teniendo en cuenta los resultados obtenidos con la red neuronal que usa espectrogramas con bandas mel y MFCC y datos ampliados, existe un 22% de falsos negativos en pacientes enfermos y un 12,381% de falsos positivos en sujetos sanos.

A continuación, se muestran los diferentes experimentos realizados, con diferentes parámetros de patience y batch para bandas mel y espectrogramas clásicos, utilizando tanto una CPU como una GPU para realizar los entrenamientos. Además, todos los experimentos han sido realizados con un epoch de 50.

`Mel spectrogram – 10 iterations:`
Average accuracy: 78.421
Average accuracy ALS: 77
Average accuracy HC: 78.9

`Normal spectrogram – 10 iterations:`
Average accuracy: 75
Average accuracy ALS: 45
Average accuracy HC: 85.7

`Mel spectrogram – 40 iterations:`
Average accuracy: 82.30263157894736
Average accuracy ALS: 68.75
Average accuracy HC: 87.14285714285714

`Spectrogram GPU CUDA Cores – 30 iterations, BATCH: 2, PATIENCE:5`
Average accuracy: 75.35087719298245
Average accuracy ALS: 32.333333333333336
Average accuracy HC: 90.71428571428571

`Spectrogram GPU CUDA Cores – 30 iterations, BATCH: 4, PATIENCE:5`
Average accuracy: 78.24561403508771
Average accuracy ALS: 40.33333333333333
Average accuracy HC: 91.78571428571428

`Spectrogram GPU CUDA Cores – 40 iterations, BATCH: 2, PATIENCE:5`
Average accuracy: 74.80263157894738
Average accuracy ALS: 26.0
Average accuracy HC: 92.23214285714285

`Spectrogram CPU – 30 iterations, BATCH: 32, PATIENCE:5`
Average accuracy: 78.859649122807
Average accuracy ALS: 34.66666666666667
Average accuracy HC: 94.64285714285714

Spectrogram GPU CUDA Cores – 40 iterations, BATCH: 2, PATIENCE:10
Average accuracy: 78.42105263157895
Average accuracy ALS: 59.25
Average accuracy HC: 85.26785714285714

Spectrogram GPU CUDA Cores – 40 iterations, BATCH: 4, PATIENCE:10
Average accuracy: 77.56578947368421
Average accuracy ALS: 57.25
Average accuracy HC: 84.82142857142857

Spectrogram GPU CUDA Cores – 40 iterations, BATCH: 2, PATIENCE:15
Average accuracy: 76.90789473684211
Average accuracy ALS: 60.75
Average accuracy HC: 82.67857142857142

Mel Spectrogram CPU – 40 iterations, BATCH: 32, PATIENCE:15
Average accuracy: 85.0877192982456
Average accuracy ALS: 78.0
Average accuracy HC: 87.61904761904763

Spectrogram GPU CUDA Cores – 40 iterations, BATCH: 4, PATIENCE:15
Average accuracy: 0.7644736842105263
Average accuracy ALS: 73.25
Average accuracy HC: 77.58928571428572

Mel Spectrogram GPU CUDA Cores – 40 iterations, BATCH: 4, PATIENCE:15
Average accuracy: 0.8230263157894736
Average accuracy ALS: 76.0
Average accuracy HC: 84.55357142857143

Mel Spectrogram GPU CUDA Cores – 40 iterations, BATCH: 2, PATIENCE:15
Average accuracy: 0.8243421052631579
Average accuracy ALS: 73.5
Average accuracy HC: 85.625

Mel Spectrogram GPU CUDA Cores – 40 iterations, BATCH: 16, PATIENCE:15
Average accuracy: 0.8164473684210527
Average accuracy ALS: 77.75
Average accuracy HC: 83.03571428571429

Mel Spectrogram GPU CUDA Cores – 40 iterations, BATCH: 32, PATIENCE:15
Average accuracy: 0.8480263157894736
Average accuracy ALS: 76.5
Average accuracy HC: 87.76785714285714

Mel Spectrogram GPU CUDA Cores – 40 iterations, BATCH: 32, PATIENCE:10
Average accuracy: 0.8328947368421051
Average accuracy ALS: 74.25
Average accuracy HC: 86.51785714285715