UNIVERSIDAD AUTÓNOMA DE MADRID
ESCUELA POLITÉCNICA SUPERIOR

Grado en Ingeniería de Servicios y Tecnologías de la Telecomunicación

TRABAJO FIN DE GRADO

Segmentador de audio basado en técnicas de Machine Learning

Autor: Juan Ignacio Álvarez Trejos
Tutor: Alicia Lozano Díez
Ponente: Joaquín González Rodríguez

julio 2020

Juan Ignacio Álvarez Trejos
Segmentador de audio basado en técnicas de Machine Learning

Juan Ignacio Álvarez Trejos
C\ Francisco Tomás y Valiente Nº 11

IMPRESO EN ESPAÑA – PRINTED IN SPAIN

Agradecimientos

Me gustaría dar las gracias a todas las personas que me han ayudado a lo largo de todos estos años de universidad, ya que sin ellas no habría sido posible la realización de este Trabajo de Fin de Grado.

En primer lugar, quiero agradecer a mis padres y abuelos por darme el privilegio de permitirme embarcarme en esta aventura que ha supuesto toda la carrera, así como por todos aquellos momentos algo más difíciles en los que el apoyo resulta imprescindible.

También quiero dar las gracias a mis amigos y amigas, por las innumerables tardes de debates, risas y discusiones. Gracias por esa mezcla heterogénea e inseparable de conocimientos y opiniones que formamos.

Gracias a mi tutora Alicia, por introducirme en el mundo del *Machine Learning* aplicado al análisis de audio, por las innumerables dudas resultas y por la paciencia empleada conmigo. También quiero dar las gracias a todo el grupo AUDIAS por permitirme este año de aprendizaje, ya que ha supuesto un importante escalón de adquisición de conocimientos en mi vida, haciéndome asimismo encontrar fascinante el mundo de la investigación.

Por último, no podía faltar dar las gracias a esa persona que ha estado conmigo desde segundo de carrera, ayudando siempre que se ha podido. Gracias por enseñarme a desaprender, a cuestionar todos esos hábitos y valores presuntamente incuestionables. Gracias Marina por hacerlo todo más fácil.

Resumen

El análisis de la señal de audio es un campo complejo donde las técnicas de Machine Learning (Aprendizaje Automático) han demostrado su eficacia, ya que, mediante estos modelos aplicados sobre diferentes características de la señal se puede reconocer diferentes tipos de patrones que nos permiten segmentar la señal de audio según la tarea de interés. Este Trabajo de Fin de Grado tiene como objetivo explorar las diferentes técnicas de *Machine Learning* y *Deep Learning* empleadas en diferentes fases de segmentado del audio.

En primer lugar, se ha realizado un sistema de Detección de Actividad de Voz (*Voice Activity Detection*, VAD), donde interesa detectar si un segmento del audio contiene voz o no. Para ello, por una parte, se ha realizado una aproximación basada en una red neuronal *feed forrward* (*Deep Neural Network*, DNN), otra basada en una red neuronal recurrente (*Recurrent Neural Network*, RNN), concretamente una *Long Short-Term Memory* (LSTM) y por último una red neuronal convolucional (*Convolutional Neural Network*, CNN). Se ha usado como características acústicas los MFCC (*Mel-frequency Cepstral Coefficients*) para los dos primeros casos y Melgramas para la CNN. Por otro lado, se ha empleado el método *Random Forest* para la misma tarea, usando esta vez la energía por trama como característica de entrada.

En segundo lugar, se ha abordado la tarea de Clasificación de Género (Hombre/Mujer) por trama de audio. Para ello se ha usado un modelo de mezclas gaussiano (*Gaussian Mixture Model*, GMM). Como característica de audio para entrenar el modelo se ha empleado la frecuencia fundamental de la señal.

Como última fase de segmentado se ha probado con un diarizador de locutor (*speaker diarization*), el cual consiste en un modelo de *clustering* entrenado *i-vectors*.

Todos los sistemas se han implementado mediante Python. En el caso de las redes neuronales se ha usado la librería Keras, la cual funciona sobre TensorFlow. Para los algoritmos de *Machine Learning* se ha empleado la librería Sci-kit learn. Para la visualización de los datos mediante gráficas se ha hecho uso de Matlab.

Los datos tanto de entrenamiento como de validación y test empleados en el sistema VAD forman parte de la base de datos OpenSAT (*Open Speech Analytic Technologies*) desarrollada por el Instituto Nacional de Estándares y Tecnología (*National Institute of Standards and Technologies*, NIST). Concretamente se ha empleado el conjunto de audios de conversaciones telefónicas de IARPA Babel (*Intelligence Advanced Research Projects Activity*). Para el clasificador de género se ha empleado la base de datos SRE10 (*Speaker Recognition Evaluation 2010*), concedida por el NIST también.

Las métricas empleadas para comparar el rendimiento y resultados entre los diferentes sistemas son el *accuracy* y la detección de la función de coste (*Detection Cost Function*, DCF).

Para la tarea VAD se ha obtenido una leve mejora respecto los demás sistemas en el caso de las LSTM como cabía esperar por ser más adecuada para modelar señales temporales. Para el clasificador de género, se obtienen buenos resultados gracias a la distribución de los datos de la frecuencia fundamental y el uso de dos GMM para modelar estas distribuciones.

Palabras clave

Machine Learning, *Deep Learning*, Detección de Actividad de Voz, DNN, LSTM, CNN, *Random Forest*, GMM, *Accuracy*, DCF

Abstract

Audio analysis conforms an important subdomain within the growing Machine Learning, since, through these models applied to different characteristics of the signal, different types of patterns can be recognized allowing us to segment the audio signal according to the task of interest. This Final Degree Project aims to explore the different techniques of Machine Learning and Deep Learning used in different stages of audio segmentation.

First, a Voice Activity Detection system (VAD) has been implemented, where the task is to detect if a segment of the audio contains voice or not. For this, on the one hand, it has been done an approach based on a deep neural network (DNN), another based on a recurrent neural network (RNN), specifically a Long Short-Term Memory (LSTM) and finally a convolutional neural network (CNN). MFCC (Mel-ferquency Cepstral Coefficients) for the first two cases and Melgrams for CNN are used as acoustic characteristics. On the other hand, the Random Forest method has been used for the same task, this time using energy per frame as the input characteristic.

Second, a Gender Classification system (Male/Female) has been developed. For this, the Gaussian mixture model (GMM) has been used. The audio characteristic used to train the model has been the pitch of the signal.

As a last segmentation stage, a speaker diarization system has been tested, which consists of a clustering model previously trained with i-vectors. All systems have been implemented using Python. In the case of neural networks, the Keras library has been used, which works on TensorFlow. The Sci-kit learn library has been used for the Machine Learning algorithms. Matlab was used to visualize the data through graphs.

The training, validation and test data used in the VAD system are part of the OpenSAT database (Open Speech Analytic Technologies) developed by the National Institute of Standards and Technology (NIST). Specifically, the IARPA Babel set of telephone conversations audios (Intelligence Advanced Research Projects Activity) has been used. The SRE10 database (Speaker Recognition Evaluation 2010), granted by NIST, was used for the gender classifier.

The metrics employed to compare performance and results between different systems are accuracy and cost function detection (DCF).

For the VAD task, a slight improvement has been obtained with respect to the other systems in the case of LSTMs, as could be expected to be more suitable for modeling temporary signals. For the gender classifier, quite satisfactory results are achieved due to the pitch distribution and the capability of the trained GMM for model this data.

Keywords

Machine Learning, *Deep Learning*, Voice Activity Detection, DNN, LSTM, CNN, *Random Forest*, GMM, *Accuracy*, DCF

Índice

Listas

Lista de ecuaciones

Lista de figuras

Lista de tablas

1

Introducción

1.1. Motivación

El segmentado del audio nos permite reconocer diferentes eventos acústicos (voz, música y ruido). En este Trabajo de Fin de Grado se emplearán técnicas de Machine Learning o Deep Learning para así poder elaborar modelos que se adecúen a cada una de las tareas analizadas, orientadas a eventos de voz. Nos centraremos en el segmentado de la voz en diferentes etapas, tratándose de un módulo de especial importancia, ya que este tipo de sistemas suelen ir seguidos de sistemas más complejos como eliminación de ruido o de música donde interesa tener la señal lo más limpia posible para evitar incrementar el error de estos sistemas posteriores.

La primera parte del Segmentador de Audio se centrará en la Detección de Actividad de Voz (*Voice Activity Detection*, VAD), esto es, detectar de forma automática si un segmento de audio se trata de Voz, o por el contrario, No Voz (silencio, ruido, etc.). Este módulo se suele tratar del primero a la hora de procesar la señal de voz, para después, poder elaborar sistemas de procesamiento de voz, transcripción de voz a texto. Por ello, es clave realizar un segmentado correcto y clasificación con un error mínimo posible. Para ello, se han diseñado distintos vectores de características para representar la señal de voz basados en la información de expertos. Una de las características acústicas más empleadas actualmente se tratan de los Coeficientes Cepstrales de Frecuencias Mel (*Mel Frequency Cepstral Coefficients*, MFCC), aunque también se emplearán la energía por trama de la señal de audio, melgramas (espectogramas en escala Mel) y por último la señal de audio RAW sin procesar.

La mayoría de sistemas en el estado del arte en la actualidad están basados en redes neuronales profundas (*Deep Neural Network*, DNNs), las cuales siendo entrenadas con vectores de características como pueden ser los MFCCs, son capaces de detectar los segmentos de voz y no voz tras haber realizado un suavizado de la salida, puesto que las decisiones a veces cambian demasiado al no tener en cuenta el contexto temporal tanto como en las RNN.

Puesto que la voz se trata de una señal temporal, las redes neuronales recurrentes (*Recurrent Neural Network*, RNNs), ideales para este tipo de señales. En este trabajo se hará uso de un tipo de RNNs en particular, las *Long Short Term Memory* (LSTM), gracias a las cuales se tiene en cuenta el

contexto temporal para modelar la información en el instante actual.

Se emplearán las Redes Neuronales Convolucionales (*Convolutional Neural Network*, CNNs). Este tipo de Redes son ideales para clasificar imágenes, y puesto que los melgramas se pueden interpretar como tal, resulta interesante implementar y analizar tales CNNs para comparar resultados con las otras redes más empleadas en esta tarea, también se emplearán redes convolucionales en una dimensión empleando como característica la señal de audio RAW.

Por último, se empleará el algoritmo Random Forest con la energía por trama de cara al sistema final fusión.

El bloque siguiente al VAD se trata de un Clasificador de Género sencillo entrenado con la característica frecuencial fundamental de la voz (denominada *Pitch*) como característica de entrada. Gracias a este sistema, la salida final nos indicará el género correspondiente en los segmentos que contengan voz.

Como última parte de este TFG se integrarán ambos módulos con un diarizador de locutor entrenado con la misma base de datos que el clasificador de género pero esta vez con otro tipo de características derivadas de los MFCCs, los *I-vectors* [1], los cuales representan en un vector de longitud fija un segmento de dimensión variable. Este sistema nos indicará el número de locutores que hay en un mismo audio, así como los segmentos en los que habla cada uno de los locutores. Este sistema es el último del conjunto, por lo que a su entrada recibirá ya únicamente segmentos con voz.

1.2. Objetivos

El objetivo principal de este Trabajo de Fin de Grado es desarrollar un Segmentador de Audio compuesto por tres subsistemas.

El primer subsistema se trata de un sistema de Detección de Actividad de Voz (VAD). Para ello se analizarán diferentes arquitecturas de NN frente a las distintas características empleadas. Para comprobar qué arquitectura y qué representación de la señal ofrece mejores resultados en la detección de voz, se realizarán varios experimentos en cada tipo de Red Neuronal. Como se ha mencionado en la motivación, se realizará sobre tres arquitecturas: *Deep Neural Networks* (DNNs), *Long Short-Term Memory Neural Networks* (LSTMs) y *Convolutional Neural Networks* (CNNs). Como añadido se realizará un experimento con Random Forest dado que muestra resultados comparables con las Redes Neuronales.

El segundo subsistema se compone de un Clasificador de Género elaborado con una GMM (*Gaussian Mixture Model*) entrenada con el *Pitch* de la señal de voz como característica fundamental.

Por último, se hará una pequeña prueba añadiendo un diarizador de locutor entrenado con *i-vectors*, el cual emplea un método de *clustering* para agrupar las partes del audio donde habla el mismo locutor,

y por tanto indicar los cambios de locutor en el fichero de audio.

1.3. Organización de la memoria

El contenido principal de este trabajo de fin de grado, se divide en las siguientes secciones:

- *Introducción*: Se presenta la motivación del Trabajo de Fin de Grado, los objetivos que este presenta y este mismo apartado, *Organización de la memoria*.
- *Estado del Arte*: Se expone la situación actual y fundamentos de la Detección de Actividad de Voz y Clasificación de Género, así como de las Redes Neuronales profundas, recurrentes y convolucionales. También se expone los fundamentos de Random Forest y GMM.
- *Entorno Experimental*: Equipos empleados, así como los datos, técnicas y métodos empleados para el desarrollo de los distintos sistemas.
- *Diseño y Desarrollo*: En esta parte se expone en qué consiste el diseño de las diferentes Redes Neuronales y algoritmos empleados.
- *Pruebas y Resultados*: Se muestran variantes empleadas en los diferentes experimentos, así como su posterior evaluación con las correspondientes métricas empleadas.
- *Conclusiones y trabajo futuro*: Con los resultados obtenidos, se elabora una conclusión y las posibles futuras líneas de trabajo donde se podría mejorar los resultados.
- *Referencias*

2

Estado del arte

2.1. Sistemas de Detección de Actividad de Voz

2.1.1. Introducción

La tarea de Detección de Actividad de Voz (*Voice Activity Detection*, VAD) es la encargada de determinar si los fragmentos de audio contienen voz o no [2].

En un proceso de comunicación en el que se incluye un sistema VAD integrado se pueden aportar muchas mejoras. Como puede ser, el incremento de la capacidad del canal, la reducción de interferencias de canales adyacentes o el ahorro de potencia consumida en un dispositivo haciendo que sólo trabaje en el momento en el que detecta voz, suponiendo que es esta la señal que se pretende procesar. Además, el ruido incluido en los audios o ciertas interferencias no estacionarias también pueden ser detectados por el sistema VAD, de forma que se puede suavizar su impacto en la señal.

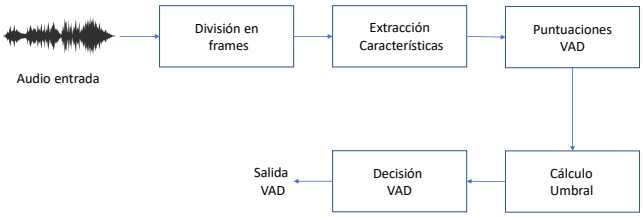

Figura 2.1: Diagrama de bloques de Detector de Actividad de Voz

En la figura 2.1, se puede ver cómo la señal de entrada se trata de la señal de audio. A continuación se procede a dividir esta en *frames* del mismo tamaño. Estos *frames* suelen tener una duración de decenas de milisegundos, ya que es en este orden donde la señal de voz se muestra menos estacionaria y donde se puede analizar mejor sus propiedades. El paso siguiente es la extracción de características,

en este trabajo de Fin de Grado se extraerán *MFCCs*, la energía por trama y *Melgramas*. El bloque de *Puntuaciones VAD* hace referencia a la puntuación obtenida respecto de las etiquetas reales. Teniendo en cuenta estas puntuaciones, se procede a escoger el umbral adecuado para nuestro conjunto de datos tal y como se ha mencionado anteriormente. Con el umbral calculado, ya se puede recalcular la decisión final de nuestro sistema VAD.

2.1.2. Extracción de Características

El audio puede estar sujeto a numerosas condiciones ambientales, haciendo que el ruido forme parte importante del audio a segmentar. Por lo que estas características deben ser representativas de la voz frente a otros eventos (silencio, ruido, música), y robustas a distintos niveles de SNR (*Signal To Noise Ratio*). Si bien la energía por segmento de audio para clasificar puede ser una buena medida, no lo es tanto en diferentes condiciones ambientales, ya que en presencia de fuertes ruidos se clasificarán mal esos segmentos debido a la alta energía de este, puesto que un sistema VAD basado en energía funciona segmentando el audio en función de la energía por trama y de un umbral escogido.

Así como la energía, encontramos otras características tradicionales como tasa de cruces por cero (*Short-Time Zero Crossing Rate*, STZCR) o magnitud media a corto plazo (*Short-Time Average Magnitude*, STAM). Sin embargo, se ha visto que ante bajos niveles de SNR, su tasa de acierto es muy baja.

Aún así, dependiendo de la aplicación posterior, puede interesar unas características u otras, como se puede ver en [3].

Cabe resaltar que las características mencionadas anteriormente hacen referencia a características extraídas del dominio temporal de la señal acústica, es decir, de la forma de onda. En este trabajo de Fin de Grado se emplearán características correspondientes al dominio frecuencial (MFCCs y Melgramas) y al dominio temporal (energía y señal de audio RAW).

Coeficientes Cepstrales en las Frecuencias de Mel

Una de las características que más se emplean en la actualidad en tareas como VAD, reconocimiento de locutor o detección de música, son los Coeficientes Cepstrales en las Frecuencias de Mel (*Mel Frequency Cepstral Coefficients*, MFCCs).

Las características MFCC resultan muy efectivas a la hora de discriminar voz respecto de otros sonidos, ya que, a diferencia de otras transformadas, se hace uso de la escala Mel [4]. La escala Mel está basada en el sistema perceptual auditivo humano, logrando así equiespaciar las frecuencias según el promedio de oyentes. De esta forma, en el dominio frecuencial del sonido se comprimirá los valores altos espectrales, siendo al contrario en las bajas frecuencias.

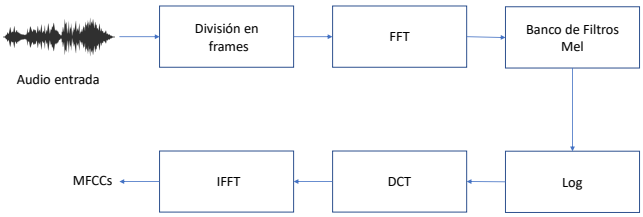

Figura 2.2: Diagrama de bloques del proceso de extracción de MFCC's

En la figura 2.2 se muestra el diagrama de bloques correspondiente al proceso de extracción de MFCC, donde el paso al dominio espectral del segundo bloque se hace de la manera tradicional empleando la Transformada de Fourier en su versión discreta (*Fast Fourier Transform*, FFT).

Melgramas

Otra de las características de la señal de audio usadas en este Trabajo de Fin de Grado son los Melgramas. Estos son una representación de la evolución temporal del espectro de nuestra señal a tratar. Para ello, resulta muy conveniente usar la Transformada de Fourier de tiempo corto (*Short-Time Fourier Transform*, STFT).

$$STFT[x(t)] = \sum_{n=-\infty}^{\infty} x[n]w[n-m]e^{-jwn} \tag{2.1}$$

Donde, $x[n]$ es la señal y $w[n]$ la ventana, ambas en tiempo discreto.

Mediante esta transformada, se realiza un enventanado con solapamiento de la señal temporal, aplicando a cada tramo enventanado la FFT. Como resultado, obtenemos una señal bidimensional en la cual el eje horizontal se corresponde con el tiempo, mientras que en el vertical tenemos una representación de la FFT. A esta representación de la señal se le denomina espectrograma.

Si por último esta matriz es ponderada en el eje frecuencial por la escala Mel, obtendremos lo que se denominan Melgramas [5].

2.1.3. Modelos tradicionales de VAD

Dentro de los modelos acústicos tradicionales que se emplean como VAD, podemos destacar los Modelos de Mezclas Gaussianas (GMM, *Gaussian Mixture Models*). Como se puede ver en [6], resulta

interesante evaluar la distribución de la energía de la base de datos a clasificar, ya que la señal de ruido normalmente es más estacionaria que la señal de voz, por lo que la varianza de los segmentos de *no-voz* es menor que en los segmentos donde sí esté presente la señal del locutor.

Se pueden emplear las GMM para modelar un espacio de características más grande, como es el caso de los MFCCs. Sin embargo, este algoritmo fácilmente sobreentrena, esto es, se ajusta demasiado a unos datos de entrenamiento, por lo que en el conjunto de evaluación no generalizaría.

Otro modelo tradicional para la tarea VAD se trata de los vectores de soporte máquina (*Support Vector Machine*, SVM), en el cual se busca un hiperplano que separe los datos de entrenamiento con el mayor margen posible. Por contra, cabe decir por una parte que es muy fácil sobreentrenar sobre los datos de entrenamiento, y por otra parte, conforme aumenta el número de datos de entrenamiento, también lo hace el coste computacional de forma exponencial.

Por último, encontramos los árboles de decisión (*Decision Trees*, DT), los cuales siguen un enfoque de división binaria recursiva supervisada (o no supervisada) para el conjunto de entrenamiento.

En [7] se realiza una comparación de estos tres modelos, concluyendo que el mejor clasificador es el *Random Forest*. En este algoritmo se emplean varios DTs para establecer varias decisiones sobre diferentes partes del conjunto de entrenamiento, evitando de esta manera el sobreentrenamiento.

2.2. Redes Neuronales

El término red neuronal tiene sus orígenes en sucesivos intentos de encontrar una representación matemática de cómo se procesa la información en los sistemas biológicos [8] [9]. Es por ello por lo que se busca en las Redes Neuronales modelos eficientes para el reconocimiento de patrones estadístico.

Las redes neuronales se pueden ver como un procesador de información, de distribución altamente paralela y constituido por muchas unidades sencillas de procesamiento llamadas neuronas. Se caracterizan principalmente por:

- Tener inclinación a adquirir el conocimiento a través de los datos, el cual es almacenado en el peso relativo de las conexiones interneuronales.
- Tienen una alta plasticidad y gran adaptabilidad.
- Tener un comportamiento altamente no-lineal, lo que les permite procesar información procedente de otros fenómenos no-lineales.

Debido a estas características y muchas otras, las redes neuronales se han convertido en una gran ayuda en el procesamiento de datos experimentales de comportamiento complejo.

2.2.1. Conceptos básicos de las redes neuronales

A continuación se explicará en las siguientes secciones los conceptos básicos que conforman las Redes Neuronales actualmente, desde la neurona hasta las diferentes topologías y metodos de aprendizaje.

Neurona

Las neuronas biológicas transmiten información (estímulo) entre ellas través de la sinapsis. Análogamente, las redes neuronales artificiales (*Artificial Neural Network*, ANN), tratan de modelar los estímulos de una neurona (entrada) entre las conexiones (sinapsis) de las neuronas, adaptando los pesos a las entradas. Cada neurona recibe información de la capa previa (exceptuando las situadas en la primera capa), la procesa y transmite información a la siguiente capa de neuronas. Cada entrada a la neurona se pondera dándole un valor o peso, posteriormente se añaden los resultados de todas las entradas y este cómputo pasa por la neurona a través de una función de activación.

El Perceptrón es un tipo de ANN de los más utilizados en éste ámbito por su capacidad de resolver problemas no linealmente separables. El Perceptrón simple tiene varias entradas x_i para cada neurona, donde se calcula una suma ponderada de ellas junto el vector de pesos w_i, dando una salida en función de la bias b, la entrada y el peso correspondiente, tal y como se puede ver en la siguiente ecuación:

$$y(x) = \begin{cases} 1 \quad if \quad \sum_{i=1}^{n} w_i x_i + b > 0 \\ \\ 0 \quad if \quad \sum_{i=1}^{n} w_i x_i + b \leq 0 \end{cases} \tag{2.2}$$

La función de activación del perceptrón es una función escalón, simple e intuitiva. Produce una salida discreta, haciendo que los pesos sean dificilmente optimizables, al contrario de lo que ocurre en funciones de activación diferenciables.

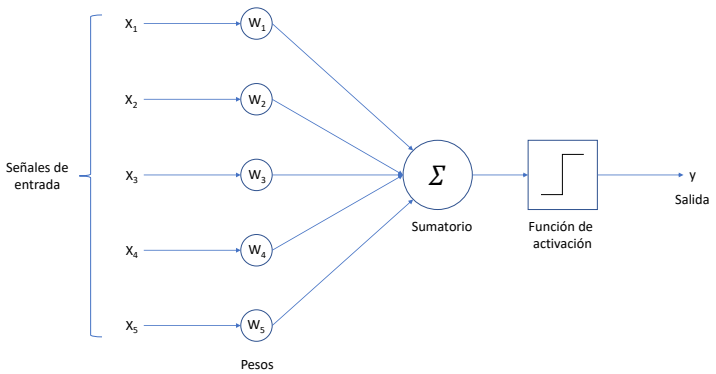

Figura 2.3: Perceptrón Simple

Función de Activación

Al igual que una neurona biológica puede estar activa (excitada) o inactiva (no excitada), las neuronas artificiales también tienen distintos estados de activación. El caso del Perceptrón descrito en el apartado anterior tiene dos estados al igual que las neuronas biológicas, sin embargo dependiendo de la función de activación este valor puede estar dentro de un conjunto acotado en las neuronas artificiales. Las funciones de activación empleadas en este Trabajo de Fin de Grado son las siguientes:

- **Función sigmoidal**: Es una función de activación muy popular que viene dada por la función sigmoidal. La salida está acotada entre $[0, 1]$ y está definida entre $[-\infty, +\infty]$.

$$S(z) = \frac{1}{1 + e^{-z}} \tag{2.3}$$

Donde z es el valor de entrada global a la neurona, por lo que y sería:

$$y(x) = \frac{1}{1 + e^{-z(x)}} = \frac{1}{1 + e^{-\sum_{i=1}^{n} w_i x_i + b}} \tag{2.4}$$

Esta función es una versión suavizada de la función escalón, como ventaja encontramos que esta función sí que es diferenciable además de ser continua, por lo que nos permite realizar cambios en los pesos.

- **Función ReLU**: Esta función es de las más empleadas hoy en día en *Deep Learning*. Aunque no es diferenciable en $z = 0$, esta función resulta muy fácil de optimizar ya que es muy parecida a la función lineal de activación. El rango de entrada y salida en este caso está entre $[0, +\infty]$, esta viene definida por:

$$R(z) = max(0, z) \tag{2.5}$$

- **Función Softmax**: Esta función de activación conocida como función exponencial normalizada es una generalización de la Función Sigmoidal ya descrita, se emplea para normalizar la probabilidad en la última capa de las redes neuronales, su expresión viene dada por:

$$S(y_i) = \frac{e^{y_i}}{\sum_j e^{y_j}} \tag{2.6}$$

En la figura 2.4 se muestra la representación de las funciones sigmoidal y ReLU para los mismos valores de z.

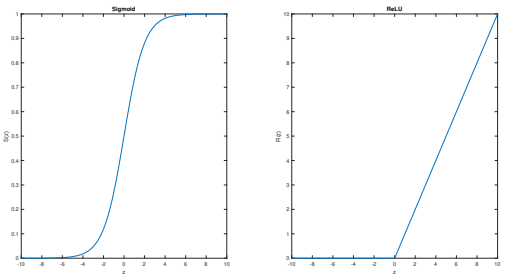

Figura 2.4: Funciones Sigmoid y ReLU

Métodos de aprendizaje

El proceso de aprendizaje de las redes neuronales consiste en la optimización del valor de los pesos asociados de tal forma que se puedan ajustar los parámetros en base a una función de coste o técnica de minimización del error. Éstos parámetros se suelen actualizar por cada *batch* del conjunto de entrenamiento.

Además, dependiendo de si se presentan o no las etiquetas reales para minimizar la función objetivo, encontramos diferentes tipos de aprendizaje (supervisado, no supervisado, semisupervisado). En este TFG, las redes sólo se han entrenado de forma supervisada, ya que los datos empleados están etiquedados (etiquetas de género y etiquetas de voz).

Para encontrar el mínimo de la función de coste, se emplea el método de descenso por gradiente (*Gradient Descent*), el cual se trata de un método general para minimizar cualquier función. Para evitar caer en mínimos locales en el descenso por gradiente, se pueden emplear diversas técnicas, en nuestro caso se ha escogido una tasa de aprendizaje variable.

2.2.2. Redes Neuronales Profundas

Las Redes Neuronales Profundas (Deep Neural Networks, DNNs) son un tipo de ANNs que se basan en el aprendizaje de las representaciones de los datos de forma abstracta, adecuándose así a la tarea en cuestión. Como se puede ver en la figura 2.5, son redes neuronales con varias capas, en el caso de la figura hay dos capas ocultas, una de entrada y una de salida. Además, todas las neuronas están conectadas con la totalidad de las de la siguiente capa, creando así un modelo *Fully-connected*. Por otra parte, los datos confluyen desde la entrada hasta la salida sin realizar ningún tipo de bucle, a este tipo de propagación se le denomina *feed-forward*.

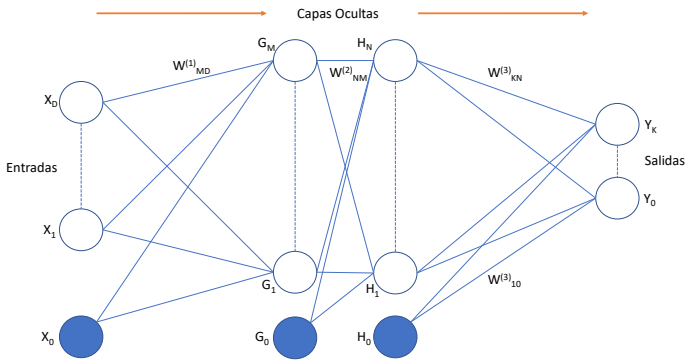

Figura 2.5: Estructura Red Neuronal Profunda

Cuantas más capas se añaden más complejo es el modelo creado y mayor cantidad de parámetros

tendrá la red, por lo que hay que establecer una relación en cada tarea entre el número de datos que se tiene y la complejidad de esta para estimar el número de parámetros necesario.

2.2.3. Redes Neuronales Recurrentes

Las Redes Neuronales Recurrentes fueron diseñadas para que estas tuvieran persistencia temporal conforme se introducen datos, permitiendo así que la salida en un instante temporal dependa de momentos temporales anteriores, otorgando a la red de memoria.

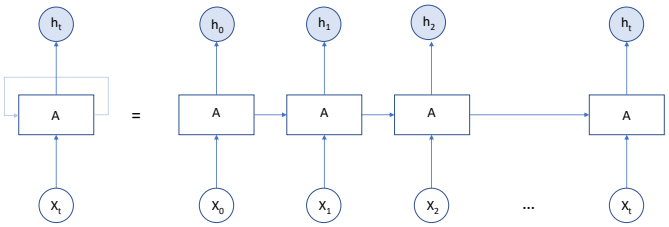

Figura 2.6: Estructura Red Neuronal Recurrente

En la figura 2.6 se muestra un ejemplo de cómo, para una RNN A y un conjunto de muestras x_t, según avanza el tiempo, la información se almacena y se utiliza para la salida en momentos posteriores,

Por esta arquitectura tan característica este tipo de redes están fuertemente relacionadas con cualquier tipo de señal temporal y han sido usadas para una amplia variedad de problemas de este tipo, como modelado del lenguaje, traducción, etc. Sin embargo, estas redes tienen un problema con las dependencias temporales a largo plazo [10], las LSTM (*Long Short Term Memory Networks*) surgen para resolver este problema. Las LSTM son un caso concreto de RNNs, introducidas en [11].

En la figura 2.7 se muestra la diferencia entre ambos tipos de redes, mientras que la RNN está compuesta por una capa de una tangente, la LSTM presenta cuatro puertas o unidades que interaccionan entre sí.

Resumidamente el funcionamiento de este bloque es el siguiente:

- La línea superior horizontal se denomina C_t e indica el estado de la celda. En esta solo se producen operaciones lineales.
- Los bloques grises se corresponden con transformaciones no lineales de la red neuronal, tres sigmoidales (σ) y una tangencial (tanh).
- Por último, cabe mencionar que las capas sigmoidales cumplen la función de deshechar muestras o no y las capas tanh acotan los valores en el rango $[-1, 1]$.

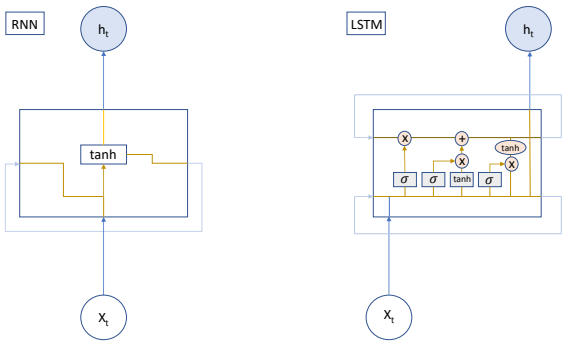

Figura 2.7: Diferencias entre RNN y LSTM

2.2.4. Redes Neuronales Convolucionales

Las redes neuronales convolucionales (CNN) son uno de los algoritmos más populares hoy en día en el mundo del *Deep Learning*. Son especialmente útiles para localizar patrones en imágenes con el objetivo de reconocer objetos, caras o incluso sonidos, aplicación que se le ha querido dar en este TFG. Estas redes aprenden directamente a partir de los datos de las imágenes.

Tal es su uso que actualmente existen numerosas redes convolucionales preentrenadas con un gran número de imágenes (LeNet, VGG, GoogLeNet, ResNet...), facilitando su posterior reentrenamiento, técnica que se conoce como *Transfer Learning* [12].

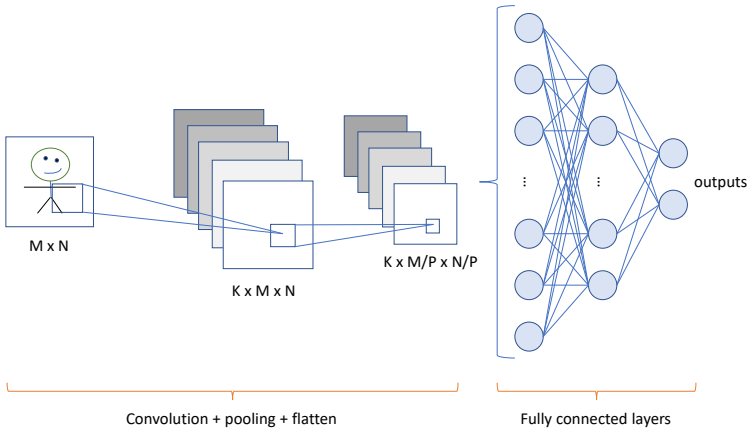

Figura 2.8: Estructura Red Neuronal Convolucional

Como se muestra en la figura 2.8, la red está compuesta por tres tipos de capas principalmente:

- La primera capa de una CNN siempre es una capa convolucional, la cual aplica una convolución bidimensional por bloques a la imagen entera. El tamaño del bloque o *Kernel* es una variable a definir. Esta operación extrae características de la imagen preservando la relación entre los píxeles de la imagen. Es común también aplicar

una función de activación (reLU normalmente) como las anteriormente descritas después de esta operación.

- Seguidamente de la capa convolucional suele usarse una capa *Pooling*. La función de esta es reducir la resolución de las imágenes, resumiendo la información de estas por bloques nuevamente. Hay diferentes tipos de *pooling*, como la media, la suma total o el máximo entre los píxeles del bloque.
- A continuación, se pueden poner más bloques correspondientes a las dos capas descritas en los puntos anteriores para extraer características en diferentes resoluciones y partes de la imagen. Por último, la matriz de características resultante se convierte en un vector unidimensional y se pasa a capas *Fully connected*, las cuales ya se ha explicado como funcionan en apartados anteriores.

2.2.5. Detección de Actividad de Voz mediante Redes Neuronales

Se han encontrado numerosos casos de estudio en los que se emplean redes neuronales para detectar la actividad de voz, partiendo del sistema baseline que hace uso de una DNN y MFCCs [13]. Un caso interesante es [14], en este experimento se emplea como característica de entrada el audio sin procesar RAW. Se emplea posteriormente un bloque convolucional en el eje temporal y otro en el eje frecuencial. Los resultados nos muestran que estos dos bloques al final acaban aprendiendo a extraer del audio una representación de éste muy similar a los melgramas ya descritos.

Por otra parte, en la literatura más reciente nos encontramos con estos dos casos [15] y [16], los cuales se tratan de sistemas presentados a las evaluaciones OpenSat 2019, una evaluación tecnológica organizada por el NIST (*National Institute of Standards and Technology*), por lo que se han empleado las mismas bases de datos que las de este TFG para la elaboración del sistema. Cabe destacar que estos sistemas se componen de subsistemas que, una vez entrenados, se fusionan sus resultados obteniendo la predicción deseada (al igual que se hace en este TFG con los mejores sistemas).

2.3. Random Forest

Random Forest (Bosques Aleatorios) [17] se trata de un algoritmo de Machine Learning de aprendizaje supervisado, el cual está conformado por bloques denominados Decision Trees (Árboles de decisión, DT).

La únidad básica del Random Forest son los Decision Trees, por ello en el siguiente apartado se explicará el funcionamiento de estos. La idea principal de este algoritmo parte del término *Bagging* (Bootstrap Aggregating) [18], método que reduce la varianza de las predicciones.

2.3.1. Conceptos básicos de los árboles de decisión

Los árboles de decisión, a su vez, son modelos de predicción que pueden ser entrenados tanto para tareas de clasificación como de regresión. Dependiendo de la tarea, el funcionamiento del algoritmo

es diferente. En este TFG se emplearán únicamente como clasificadores.

La terminología relacionada con los DTs es la siguiente:

- Nodo Raíz, el cual representa a todo el subconjunto D_i mencionado previamente.
- División del subconjunto en dos o más subnodos.
- Nodo de decisión, cuando un subnodo se divide en más subnodos.
- Hoja o nodo terminal (*Leaf*) son denominados aquellos nodos que no se dividen.
- Un SubTree (*Sub-árbol* constituye una sección entera del árbol.

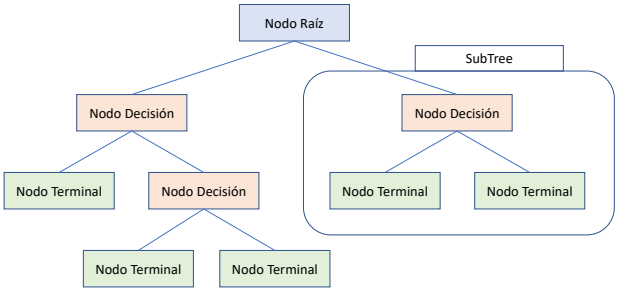

Figura 2.9: Esquema Decision Tree

Para dividir el subconjunto entre los subnodos, el DT puede emplear diversos algoritmos, dependiendo de la tarea. Para el caso de clasificación el algoritmo más empleado es el ID3. Este algoritmo comienza con el subconjunto D_i, en cada iteración opera sobre las partes no usadas de D_i y calcula su Entropía (H) de su atributo. Después selecciona el atributo con menor entropía y divide el conjunto según este.

2.3.2. Detección de Actividad de Voz mediante Random Forest

Se han encontrado recientes estudios en los que se emplea Random Forest para tareas relacionadas con audio, como por ejemplo para clasificar distintos tipos de ruido [19], en el que empleando como características los MFCCs ya descritos, consiguen implementar un sistema de eliminación de ruido para implantes de clóquea adaptativos en el que mejoran los sistemas previos un 10 % gracias al uso de este algoritmo.

Otra aplicación interesante de este es la detección de señales de ecolocalización de murciélagos gracias a la combinación de un sistema VAD basado en GMMs y un Random Forest [20], empleando como señales de entrada los espectrogramas correspondientes a estas señales.

Como último ejemplo cabe mencionar el uso de este algoritmo para la detención de eventos acústicos [21], en este artículo se estudia cómo los RF son capaces de clasificar estos eventos gracias al

método de *bagging* implícito en este.

2.4. Sistemas de Clasificación de Género

2.4.1. Introducción

La clasificación de género del locutor ha ganado poco a poco popularidad debido a todas las aplicaciones que puede tener en diversos dominios, como puede ser reconocimiento de locutor o identificación de emociones por ejemplo.

En este tipo de sistemas el audio se fragmenta en tramas (*frames*) del mismo tamaño, las cuales posteriormente serán clasificadas por el sistema. En nuestro caso nos ceñiremos a una tarea binaria, en la que se tendrá que decidir si estos frames son *male* o *female*. Para esto, se debe encontrar un umbral de decisión óptimo que minimice el error total y maximice la distancia entre nodos o grupos.

Además, gracias al sistema VAD previamente explicado, si se descartan aquellas tramas que correspondan a no-voz aseguramos que a la entrada del clasificador de género no entran segmentos que pudieran inducir a error por no pertenecer a ninguna clase y sólo ser ruido o sonidos indistinguibles.

2.4.2. Extracción de características

La información de género es invariante al tiempo, invariante al tipo de fonemas e independiente del locutor para un género dado [22]. Los autores para esta tarea han empleado características como los MFCCs, LPC (*Linear Prediction Cepstrun Coefficients*), LPCC (*Linear Prediction Coefficients*), *Pitch* o los vectores de formantes.

Pitch

En este TFG se ha optado por emplear como característica el *Pitch*, debido a que se trata de una característica muy discriminativa para esta tarea y se puede modelar como una distribución Gaussiana, *male* y *female*, lo que facilitaría emplear un Modelo de Mezcla Gaussiano. Para ello se han encontrado numerosas técnicas de estimación del Pitch aplicando diferentes operaciones sobre el audio, como el cálculo de la autocorrelación de la señal, cepstrum o *Average Magnitude Difference Function* (AMDF) y más en [23].

En la figura 2.10 se refleja el proceso de extracción del Pitch realizado.

Para el caso que nos concierne, se ha extraído el Pitch mediante el *toolkit* Kaldi, el cual está basado en encontrar los valores que maximizan la función de correlación cruzada normalizada (*Normalized Cross Correlation Function*, NCCF) [24].

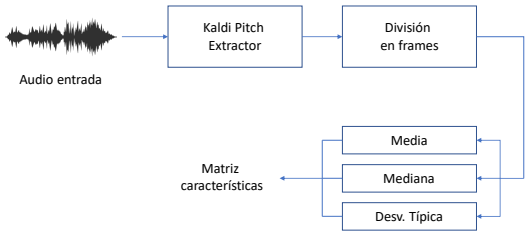

Figura 2.10: Extracción Pitch

2.4.3. Modelos tradicionales

Dentro de los modelos tradicionales de Machine Learning para la Clasificación de Género encontramos el empleo de SVMs (*Support Vector Machines*) entrenados con características como la frecuencia fundamental (F0) y los formantes de la voz [25], así como el uso también del algoritmo Naive-Bayes con el Pitch como entrada [26]. Como último modelo tradicional se plantea [27], en el cual se obtiene un $98\,\%$ de Accuracy para ambas clases sin ruido de fondo. En este sistema hacen uso de GMMs (*Gaussian Mixture Models*) empleando como características el Pitch y LPC. Asimismo, comparan el algoritmo con otros conocidos y mencionados previamente como HMM (*Hidden Markov Model*), SVMs y ANNs, siendo este el que mejor resultados ofrece.

2.5. Modelos de Mezcla Gaussiana (GMM)

Los modelos de mezcla gaussiana (*Gaussian Mixture Models*, GMMs) son un algoritmo de aprendizaje no supervisado ampliamente utilizado. La solución de este es similiar al algoritmo tradicional de *clustering K-means*, pero más robusto, ya que *K-means* emplea la distancia euclídea para encontrar los diferentes clusters posibles (ideal para distribuciones circulares), sin embargo si los datos no son lineales o siguen una distribución elipsoidal el algoritmo no es capaz de modelar correctamente los clusters.

En las GMMs se asume que estos subconjuntos (K) de X siguen una distribución normal, siendo la tarea principal encontrar los parámetros de las K distribuciones gaussianas para poder visualizar los datos o realizar predicciones.

2.5.1. Conceptos básicos de las GMMs

El principal objetivo de las GMM es maximizar el valor *likelihood* para los datos X, o el *log-likelihood*. Asumiendo una combinación de K gaussianas, se puede escribir $p(X)$ como la probabilidad marginal, sumada sobre todos los *clusters* para cada dato.

$$p(X) = \prod_{i=1}^{N} p(x_i) \tag{2.7}$$

De donde

$$p(x_i) = \sum_{k=1}^{K} p(x_i|c_k)p(c_k) \tag{2.8}$$

En la ecuación 2.9 se muestra el valor correspondiente al *likelihood*, por lo que el logaritmo de este será:

$$L = log(p(X)) = \sum_{i=1}^{N} log(\sum_{k=1}^{K} p(x_i|c_k)p(c_k)) \tag{2.9}$$

Una posible solución para esta ecuación es el uso del *Expectation Maximization (EM) algorithm*, un algoritmo iterativo usado para encontrar el máximo *likelihood* estimado (MLE) para modelos donde los parámetros no se pueden encontrar directamente, como GMM, Mezclas de distribuciones de Bernoulli o regresión lineal Bayesiana.

No se va a profundizar en el desarrollo matemático del algoritmo EM, simplemente cabe decir que este está conformado por dos pasos:

- *Expectation Step*: Se calcula en este paso la probabilidad de que los puntos x_i pertenezcan al cluster c.
- *Maximization Step*: Se calcula un nuevo parámetro mc, que determina la fracción de puntos pertenecientes a cada cluster. Se actualizan los parámetros.

2.5.2. Ejemplos de aplicación de GMMs

Se han encontrado varios estudios en los que se emplea el algoritmo GMM para tareas de voz. Por ejemplo para la tarea VAD basada en el reconocimiento de fonemas a partir de MFCC [28]. Otro caso interesante es [29], donde después de pasar el audio por un sistema VAD, se entrena un sistema basado en GMM en dos etapas, en la primera identificando el género del locutor y en la segunda estimando la edad de este. Por último, mencionar este estudio [27] en el que se hace uso de GMM para clasificar género nuevamente, con el uso del Pitch y de los LPC como características de entrada, obteniendo una muy alta tasa de acierto en el conjunto de test.

3

Entorno Experimental

3.1. Herramientas

En esta sección se explicarán las herramientas empleadas para poder llevar a cabo el Trabajo de Fin de Grado. Por una parte se explicarán las tecnologías que conciernen al desarrollo de los experimentos realizados y por otra, las bases de datos requeridas para cada tarea.

Lenguajes de Programación

El lenguaje principal que se ha empleado para desarrollar todos los *scripts* necesarios ha sido Python, debido a su rápida curva de aprendizaje y a que es el lenguaje más empleado actualmente en todas las tareas que giran en torno a Machine Learning, esto debido al gran número de librerías que tiene para este propósito. Además se trata de un lenguaje con licencia de código abierto y multiplataforma, aunque en el caso que nos ocupa se ha usado sobre Ubuntu.

Por otra parte, también se ha empleado GNU Bash, el cual se trata de un lenguaje de comandos y *shell* de Unix, predeterminado para la terminal de Ubuntu. Se ha empleado con el motivo de acelerar el lanzamiento de experimentos y ejecutar diversas acciones en cascada, automatizando así procesos.

Librerías

Como mencionaba en el apartado anterior, Python destaca por las librerías que tiene de código abierto, las cuales están en continuo desarrollo.

Por una parte, el desarrollo de todos los modelos de redes neuronales se hizo con la librería Keras, la cual puede funcionar sobre el *framework* Tensorflow. Esta librería también destaca por ser amigable con el usuario, además de ser modular y extensible, de tal forma que es común su uso en el inicio en *Deep Learning* con Redes Neuronales. sKeras contiene varias implementaciones de los bloques que conforman las redes neuronales, como pueden ser optimizadores, distintos tipos de capas, funciones de activación o funciones de coste.

Por otro lado, para la extracción de características como los melgramas o los MFCCs, se han

empleado las librerías LibROSA y Python Speech Features respectivamente. Estas permiten extraer diferentes tipos de características con los parámetros deseados, como puede ser el tamaño de la ventana deslizante, la frecuencia de muestreo, el número de filtros del banco de filtros, etc. También se ha empleado el *toolkit* Kaldi, diseñado específicamente para la investigación en reconocimiento del habla (*Speech Recognition*), todo su código es de libre acceso y se encuentra disponible en Github. Este software se ha empleado para extraer el Pitch para la tarea de clasificación de género.

Por último, también se ha empleado el *toolkit* Scikit-learn, el cual es una librería diseñada para realizar experimentos de Machine Learning en Python, de código abierto y de uso muy simple. Esta ha sido empleada para probar diferentes algoritmos (Random Forest, SVMs y GMMs) y para evaluar los sistemas, ya que ofrece una función rápida para calcular las matrices de confusión, entre otras métricas comunes.

Matlab

MATLAB (MATrix LABoratory), es un sistema de cómputo numérico que ofrece un entorno de desarrollo integrado (IDE) disponible en diversas plataformas, en nuestro caso se ha empleado en Windows. Se ha usado MATLAB principalmente para la elaboración de todas las gráficas de entrenamiento de las redes neuronales, así como para visualizar los datos. Además, en los inicios de desarrollo del TFG sirvió como puente para la adaptación final a Python.

GPUs

Las unidades de procesamiento gráfico o GPU son coprocesadores dedicados al procesamiento de gráficos u operaciones en coma flotante, de tal forma que aligeran la carga de trabajo del procesador principal (CPU) del ordenador. Estas unidades son de especial interés en el ámbito de Deep Learning, ya que en este campo se requieren operaciones de gran carga computacional entre matrices (Tensores). Estas unidades de procesamiento constan de un alto ancho de banda de memoria y de unos registros mucho más grandes y rápidos en el nivel L1 de memoria, el cual es más facilmente programable, optimizando las multiplicaciones y convoluciones matriciales.

Bases de Datos

Para la tarea VAD, los datos se han obtenido de la base de datos NIST OpenSAT (Open Speech Analytic Technologies). OpenSAT es una evaluación orientada a tecnologías del habla organizada por el Instituto Nacional de Estándares y Tecnología (National Institute of Standards and Technology, NIST). Esta evaluación consta de tres tareas principales: VAD, Búsqueda de palabra clave (Key Word Search, KWS) y reconocimiento automático del habla (Automatic Speech Recognition, ASR). Esta base de datos se compone de diferentes conjuntos aunque en este Trabajo de Fin de Grado se ha empleado únicamente la colección IARPA BABEL. Este conjunto se compone de grabaciones en lenguajes como

el Pashto (Pastún, hablado en Agfanistán, Pakistán e Irán) de los que se consta de pocas horas de grabación. Las grabaciones son telefónicas en su mayoría (Conversational Telephone Speech, CTS).

Por otra parte, para la tarea de Clasificación de Género se ha empleado la base de datos SRE10 (Speaker Recognition Evaluation 2010), la cual también está concedida por el Instituto Nacional de Estándares y Tecnología (NIST) y fue en 2010 una evaluación de reconocimiento del habla. Estas grabaciones se encuentran en inglés en este caso.

3.2. Medidas de Evaluación

En este apartado se procederá a describrir las medidas empleadas para evaluar el rendimiento de los métodos y algoritmos usados para las tareas de clasificación descritas en los anteriores puntos.

Matriz de confusión

En primer lugar, se busca evaluar el porcentaje de error respecto las etiquetas obtenidas por los diferentes métodos y las etiquetas originales. Por lo que las probabilidades de error se medirán en base a segmentos correctos o incorrectos respecto de la clase real.

En la figura 3.1 se muestra en primer lugar un ejemplo de etiquetado real, y la relación entre este y dos posibles predicciones para cada uno de los sistemas empleados. Para el caso de VAD, aquellas partes del segmento correspondiente a Voz que son clasificadas como No-Voz se corresponden con Falsos Negativos (*false negative*, FN). Por otro lado, las partes que se tratan de No-Voz y son clasificadas como Voz, son Falsos Positivos (*false positive*, FP). Por el contrario, aquellos fragmentos bien clasificados, se denominan para los segmentos de Voz, verdaderos positivos (*true positive*, TP) y para los segmentos de No-Voz, verdaderos negativos (*true negatives*, TN).

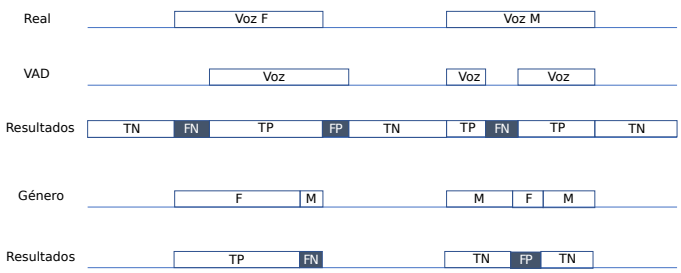

Figura 3.1: Comparación predicción y target para los dos tipos de sistemas empleados

Lo mismo ocurre en el caso del clasificador de género, asignando *Positive* a la clase *Female* y

Negative a *Male*.

Si se calculan los porcentajes de falsos negativos y falsos positivos sobre la duración total del audio obtenemos dos medidas muy útiles para poder minimizar esta tasa de error posteriormente, falso rechazo(P_{miss}) y falsa aceptación(P_{FA}).

$$P_{miss} = \frac{T(FN)}{T(Voz)} \tag{3.1}$$

$$P_{FA} = \frac{T(FP)}{T(NoVoz)} \tag{3.2}$$

Estas dos medidas han sido empleadas para encontrar el umbral óptimo para así minimizar el error al máximo en la medida de lo posible con los sistemas que se tienen.

Para poder visualizar los resultados de un sistema es muy común emplear lo que se denomina matriz de confusión, representada en la figura 3.2

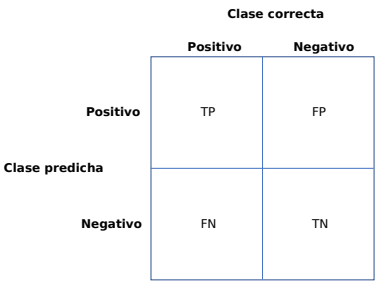

Figura 3.2: Representación matriz de confusión

En ésta queda reflejado lo bien que clasifica un sistema cada una de las clases y qué tasa de error tiene, pudiendo ver en dónde falla más y dónde acierta más. Es una matriz con los valores descritos más arriba: verdaderos positivos (TP), verdaderos negativos (FP), falsos positivos (FP) y falsos negativos (FN) como se muestra en 3.2.

DCF

Otra medida común para evaluar los sistemas empleados se trata del DCF (*Detection Cost Function*, del inglés Detección de la función de coste). Esta medida se ha empleado para comparar los resultados obtenidos con, por ejemplo, el sistema baseline del que se partía [13]. Se trata de ponderar las dos medidas descritas anteriormente, dependiendo del caso, penalizando unos fallos más que otros. Se ha querido penalizar por igual ambas medidas para obtener predicciones lo más simétricas

posibles.

$$DCF = 0{,}5 * P_{miss} + 0{,}5 * P_{FA} \tag{3.3}$$

Accuracy

Como última medida de evaluación se ha empleado el *accuracy* o precisión del sistema de predicción por frame. Esta medida se ha usado para evaluar nuestros sistemas. Para calcular el accuracy sobre un conjunto de predicciones basta con emplear la siguiente ecuación:

$$Accuracy = \frac{1}{N}\sum_{i=1}^{N} Acc_i \tag{3.4}$$

de donde

$$Acc_i = \begin{cases} 1 & if \quad Y_{predi} == Y_i \\ 0 & if \quad Y_{predi} = Y_i \end{cases} \tag{3.5}$$

Donde N es la longitud del audio en cuestión, por lo que el resultado será el porcentaje de acierto.

4

Diseño y Desarrollo

4.1. Introducción

Tras haber expuesto en el *Estado del Arte* el estudio previo que se ha realizado al desarrollo del proyecto para adquirir los conocimientos necesarios, en este apartado se procederá a explicar cómo se han tenido que preparar y extraer las características para cada uno de los métodos y sistemas empleados, así como el porqué de la estructura de cada red neuronal y configuración de cada algoritmo.

Este Trabajo de Fin de Grado se puede dividir en dos sistemas diferenciados, por una parte el Sistema de Detección de Voz, y por otra, el Sistema de Clasificación de Género. Para cada uno de estos sistemas se han empleado diferentes técnicas de Aprendizaje Máquina (*Machine Learning*), como las ya descritas GMMs o Redes Neuronales de tres clases diferentes.

4.2. Red DNN

El sistema *baseline* sobre el que se empezó a desarrollar este Trabajo de Fin de Grado [13] se trata de una Red Neuronal Profunda (*Deep Neural Network*, DNN), que, como ya se ha explicado en apartados anteriores, se trata de una red *Feed-Forward*, esto es, las conexiones van desde la capa de entrada hasta la capa de salida. Esta Red Neuronal se empleará para la tarea de VAD, empleando como vectores de características los *MFCCs* descritos en el *Estado del Arte*. También se han probado otras características como los LPCC (*Linear Prediction Cepstral Coefficients*), pero se han descartado al no mostrar resultados relevantes [30].

4.2.1. Preparación Previa

Para la extracción de los MFCCs sobre la base de datos se ha empleado una librería de *Python* (*Python Speech Features*). El proceso de extracción usual de características sobre una señal temporal se hace con una ventana de tamaño 25 ms, con un paso de 10 ms, ya que en segmentos de pocos milisegundos la señal presenta un comportamiento estacionario. De esta manera, al calcular la

característica en un instante, se tiene en cuenta unas pocas muestras previas y posteriores,

Todos los audios se encuentran grabados a una frecuencia de muestreo de 8KHz, es decir, 8000 muestras por segundo debido a que se trata de llamadas telefónicas. Por lo que, por ejemplo, si tenemos un audio de 10 minutos de duración, esto supone que tendrá 4800000 muestras. Al hacer el proceso de extracción de MFCCs, la matriz resultante será de $(60000, 20)$. La segunda dimensión hace referencia al tamaño de nuestro vector de características. En nuestro caso se ha empleado una dimensión de 20. El conjunto de la base de datos consta de 1028 audios. De aquí se han extraído los sets de entrenamiento (949 audios, 93 %) y validación (79 audios, 7 %), para ello se ha comprobado que estén igualmente equilibrados, es decir, que cada clase (*Voz* y *No Voz*) esté igualmente representada en cada conjunto. Como conjunto de test se empleará el subconjunto */dev* de IARPABABEL, el cual estaba originalmente orientado a ser empleado como conjunto de test en las evaluaciones del NIST OPENSAT de 2019. Este set consta de 128 audios.

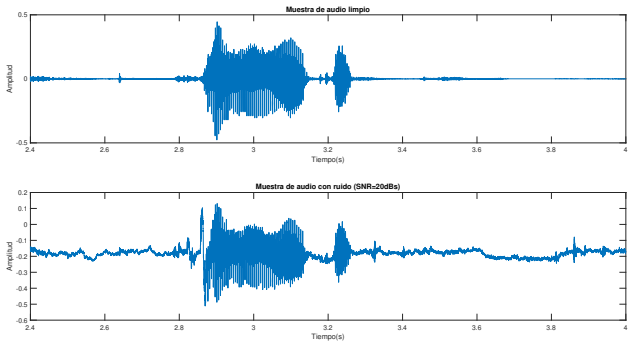

Figura 4.1: Representación de audio limpio y con ruido añadido

Además, a la base de datos se le ha incrementado la cantidad de datos mediante *Data Augmentation*. Esto es, sumarle a la señal original ruido de fondo o voces de otra base de datos, para de esta manera, ganar robustez en el aprendizaje y predicciones de la Red Neuronal ya que es muy frecuente que el frame a clasificar esté en presencia de ruido. En la figura 4.1 se muestra un ejemplo de un archivo al que se le ha aplicado este *data augmentation*. Se ha empleado el *corpus* MUSAN [31], el cual consta de tres conjuntos de datos: música, voz y ruido. La única restricción que se ha impuesto para este proceso es que a cada audio en limpio se le sume un ruido aleatorio de MUSAN con una SNR aleatoria entre 30 y 10 dBs.

Por último, una vez que se tiene los MFCCs en limpio y con el Data Augmentation realizado, se puede llevar a cabo el entrenamiento. De cada audio se cogen 50 segundos. Después, cada vector de MFCCs se ha preprocesado para añadir contexto temporal de 15 frames en cada extremo.

4.2.2. Diseño de la Red Neuronal

Para el diseño de la DNN hay que tener en cuenta que el número de parámetros empleado y el tamaño del conjunto de entrenamiento de la base de datos tiene una relación muy importante. Para encontrar este balance, hay que saber que si se empleasen capas de pocas unidades o pocas capas, se podría estar generando un modelo demasiado simple para nuestro problema, por lo que la red no generalizaría de forma adecuada. Por el otro lado, si empleamos un número demasiado alto de parámetros, puede llevar a que se produzca *overfitting*.

Basándonos en el Estado del Arte actual [32], se ha optado por emplear 3 capas ocultas, con 500 unidades cada una. A la entrada de la red introducimos un vector de MFCCs de dimensión 620 (29 tramas de contexto + trama actual, $(30, 20)$). Además, la red consta de una capa de salida con dos neuronas, una para cada clase. Para las siguientes dos capas se hará uso de la función ReLu. El tamaño de batch escogido es de 512, por lo que en cada pasada hacia delante de la red (*forward*), estaremos introduciendo una matriz de tamaño $(620, 512)$. Por último, se ha empleado como optimizador Adam y como función de pérdida *categorical crossentropy*.

4.3. Red LSTM

Otro tipo de redes neuronales empleadas son las Redes Neuronales Recurrentes (*Recurrent Neural Networks*, RNNs). En concreto se ha hecho uso de un tipo específico de RNNs denominadas LSTM (*Long Short Term Memory*), las cuales, como ya se ha explicado en la sección 2, son ideales para reconocimiento de patrones temporales. Esto es debido a que, como toda RNN, son capaces de modelar señales temporales teniendo en cuenta el contexto temporal. En la sección previa se mencionaba que en la DNN se forzaba a que las tramas introducidas en la red tuvieran presente este contexto, sin embargo, en las LSTMs este tipo de información viene implícito.

4.3.1. Preparación Previa

Para las LSTM se han empleado las mismas características que las descritas en el apartado 4.2, con la diferencia de que este tipo de redes esperan a la entrada una señal tridimensional, es decir, se espera una secuencia de secuencias. En el caso que nos ocupa emplearemos también segmentos de 50 segundos por audio, que dividimos en fragmentos de 3 segundos, resultando en un tamaño final de $(16, 300, 20)$.

4.3.2. Diseño de la Red Neuronal

En cuanto al diseño de la Red, se ha empleado un modelo con 3 capas ocultas de 500 unidades, una capa de entrada que recibe como parámetro la matriz descrita previamente y una capa de salida, la cual consta de dos unidades, de nuevo una por cada clase.

Además cabe destacar que en la última capa se añade el *wrapper Time Distributed*, el cual se encarga de aplicar a cada paso temporal de la LSTM la misma operación *Dense (fully-connected)*. Esto es necesario ya que se emplea como parámetro $return\_sequences = True$, lo que implica que la salida de la LSTM para cada paso temporal se pasa hasta la última capa.

4.4. Red CNN

El último tipo de redes neuronales exploradas en este Trabajo de Fin de Grado son las Redes Neuronales Convolucionales, como ya se ha explicado en el capítulo 2, estas redes son de gran interés en reconocimiento de patrones de imágenes y han demostrado gran capacidad de aprendizaje gracias a su intrínseca función de poder extraer información en zonas localizadas de la entrada, teniendo en cuenta la información adyacente.

Este tipo de redes serán empleadas para la tarea de VAD, pero, a diferencia de los dos casos descritos previamente, no se hará uso de MFCCs como vectores de características. Se realizarán dos aproximaciones, una con Melgramas, donde la red aprenderá a reconocer la forma espectral de la voz humana en escala Mel, y, por otra parte, se ha querido experimentar con la forma de onda como entrada a las CNNs.

4.4.1. Preparación Previa

CNN con Melgramas

Por una parte, la extracción de los Melgramas se ha hecho con la librería *LibRosa* de Python. Para ello se han empleado una ventana de la FFT de 2048 muestras. El resultado de la extracción resulta ser una matriz de $(128, N)$, donde la primera dimensión corresponde al eje frecuencial y la segunda al eje temporal. El problema aquí es que las CNNs son capaces de distinguir dentro de una imagen si está o no el patrón buscado, pero no localizan y devuelven la posición del patrón. Por este motivo hay que hacerle a la matriz un *reshape* como en el caso de las LSTM. El resultado será una matriz de tamaño $(128, RES, N/RES)$. El parámetro RES hace referencia a la resolución temporal que implica ese cambio de tamaño de la matriz. La elección de la resolución se ha establecido midiendo cuál es el mínimo tiempo de duración de un segmento de VOZ y no VOZ dentro de la base de datos, ya que el objetivo es que cada matriz o parte del melgrama de tamaño $(128, RES)$ únicamente contenga una

clase, evitando así errores de predicción en los bordes. Por esto, se ha escogido un valor de RES que estuviera entre 0.03 segundos y 0.2 segundos. El valor final de la resolución temporal elegida es de 0.05 segundos, correspondiente a 5 muestras del eje temporal del melgrama.

CNN con RAW-AUDIO

En el caso de emplear la forma de onda como entrada para la CNN, el parámetro que hay que ajustar vuelve a ser nuevamente la resolución temporal que queremos que tengan las predicciones. El valor empleado para este caso es de 80 muestras sobre el audio, lo que supone una resolución temporal de 1 milisegundo.

4.4.2. Diseño de la Red Neuronal

A continuación se describirán las diferentes arquitecturas de CNNs empleadas para cada una de las características a introducir en la red. Se consta de una topología empleada para los melgramas y otras dos distintas para la forma de onda como entrada.

CNN-1

Esta red recibe como entrada los melgramas con el tamaño descrito anteriormente. Se han escogido tres capas convolucionales (2D) como capas ocultas con 128 unidades cada una y con función de activación ReLU. Después de cada capa se añade una operación de MaxPooling2D únicamente en el eje frecuencial. Posteriormente se añade una capa densa (*Fully Connected*) con 512 neuronas, dando paso a la última capa con dos elementos, uno por cada clase como en casos anteriores. Como optimizador se ha empleado de nuevo Adam y como función de pérdida *categorical cross-entropy*.

CNN-2

Este caso es muy similar al anterior, teniendo en cuenta que la señal consta únicamente de una dimensión, ya que le estamos introduciendo la señal RAW de audio. Por lo que, si en la CNN previa empleábamos filtros convolucionales y de *MaxPooling* de 2 dimensiones, en este caso serán de una dimensión. Además, el tamaño del Kernel empleado también es distinto, siendo en este caso para la primera capa de 64.

CNN-LSTM

Como se introduce en [5], resulta interesante combinar la capacidad de reconocimiento de patrones visuales de los filtros convolucionales con la capacidad de las neuronas LSTM de retener información y así poder modelar señales temporales de forma más robusta. Es por esto por lo que se propone

cambiar el modelo descrito anteriormente añadiéndole después del último *Max pooling* dos capas LSTM con 128 unidades cada una. Respecto del resto de parámetros, la red se estructura de la misma manera que en CNN-2.

4.5. Random Forest

El algoritmo *Random Forest* sigue siendo muy empleado en tareas de clasificación binaria debido a que previene el *overfitting* entrenando sobre partes aleatorias del conjunto de entrenamiento. Para este caso se entrenará el algoritmo con la energía por trama, para ello se ha empleado el coeficiente 0 de los MFCCs de los experimentos anteriores para este sistema.

4.5.1. Preparación Previa y desarrollo

Hay dos principales razones por las que no se ha evaluado este algoritmo con el vector completo de MFCC's. Por una parte, resulta muy costoso emplear un vector tan grande de características para este tipo de modelo, ya que el tiempo de cómputo sería excesivo. Además, de cara a generar un último modelo Fusión VAD, se buscará en qué zonas cada uno de los sistemas empleados fallan más, para así, ponderando cada sistema de la forma óptima, se logrará una combinación de predicciones empleando sistemas con diferentes caracerísticas cada uno. Se han empleado un total de 100 componentes o DTs para el algoritmo, cada uno de ellos con una profundidad máxima de cuatro nodos.

4.6. Modelos de Mezcla Gaussiana

Por último se ha empleado el Modelo de Mezcla Gaussiana (*Gaussian Mixture Model*, GMM) tanto en Clasificación de Género como en Detección de Actividad de Voz. Estos modelos, como ya se explicó en el *Estado del Arte*, son modelos probabilísticos que asumen que los datos son generados de una combinación finita de distribuciones Gaussianas. Para implementar los sistemas basados en GMMs se ha empleado el *toolkit* Scikit-learn.

4.6.1. Preparación Previa y desarrollo

Para la clasificación de género se ha empleado como característica la frecuencia fundamental (*pitch*). Para ello se han entrenado de nuevo dos GMMs, esta vez de 2 y 3 componentes. Como estimadores se han empleado la media, mediana y desviación típica del *pitch* por trama.

5

Pruebas y Resultados

A continuación se procederá a mostrar los resultados de los experimentos más relevantes, para ello se empezará por la tarea VAD y sus correspondientes experimentos, finalizando esta sección con un sistema Fusión que englobará los mejores experimentos de cada una de las metodologías empleadas.

Después se expondrán los experimentos realizados para la tarea de clasificación de género.

5.1. Sistemas VAD

En la tabla 5.1 se muestra un resumen de los experimentos realizados para esta tarea con cada una de las metodologías ya descritas. Se puede ver como las redes neuronales son las que mejores resultados dan, aunque a costa de un mayor tiempo de entrenamiento y empleando un número bastante alto de parámetros. El número de parámetros reflejado en la tabla es $p = log(parametros)$.

Exp.	Red/Algoritmo	Data Augmentation	Features	p	Acc. Test	DCF
1	DNN	No	MFCC's	5.91	89.3	0.106
2	DNN	Sí	MFCC's	5.91	84.3	0.156
3	LSTM	No	MFCC's	6.70	90.3	0.090
4	LSTM	Sí	MFCC's	6.70	**90.8**	**0.085**
5	CNN-1	No	Melgramas	7.88	90.4	0.095
6	CNN-1	No	Melgramas	7.88	87.2	0.126
7	CNN-2	No	RAW-Audio	7.03	90.5	0.097
8	CNN-LSTM	No	RAW-Audio	5.88	90.4	0.095
9	RF	No	Energía	-	89.9	0.100

Tabla 5.1: Experimentos realizados

Un factor a tener en cuenta cuando se está trabajando con problemas de clasificación y evaluando su rendimiento con *accuracy*, es que la proporción de datos de cada clase debe estar balanceada. Se ha extraído sobre el conjunto de entrenamiento total un 7 % de los datos para formar el conjunto de validación, de tal forma que ambos conjuntos estén balanceados tal y como está el conjunto de test,

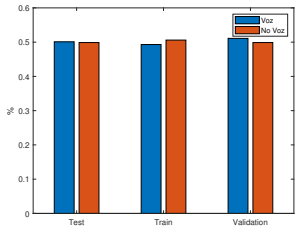

Figura 5.1: Distribución conjuntos

esta distribución se puede ver en la figura 5.1.

Conjunto	Nº Muestras	Porcentaje
Train	52932981	93
Validation	4533799	7
Test	7250980	100

Tabla 5.2: Cantidad de muestras por conjunto

Por último, cabe mencionar que en los experimentos realizados con *Data Augmentation* los conjuntos siguen estando balanceados, ya que no se modifica el número de segmentos por conjunto. De hecho sólo se aplica esta transformación sobre el set de entrenamiento, ya que el conjunto de validación nos sirve para evaluar el rendimiento de nuestro sistema sobre unos datos reales no vistos por el sistema y para parar el entrenamiento cuando se empieza a sobreentrenar.

5.1.1. Sistemas basados en DNN

El diseño de la red DNN y preparación previa de los datos se ha introducido en el apartado 4.2, además en la tabla 5.1 se reflejan los experimentos con mejores resultados obtenidos para las DNN, correspondientes a los experimentos 1 y 2.

Para ambos experimentos se ha empleado el modelo que se refleja en la tabla 1 del apéndice, modelo que consta de tres capas con 500 unidades cada una. Si se hubiera empleado una capa menos, el número de parámetros total se reduciría, formando un modelo más simple que entrenase más rápido. Sin embargo este parece no tener capacidad suficiente para aprender la tarea, como se ha podido ver en los experimentos realizados.

Como se puede ver en la figura 1 del apéndice, la DNN con los datos limpios empieza a sobreentrenar a partir de la época 15, que es cuando el porcentaje de acierto en validación se estanca, mientras el porcentaje de acierto en el conjunto de entrenamiento sigue subiendo. Por esto, se cogerán los pesos de esta época para evaluar en el conjunto de test.

Por otra parte, en la figura 2 se muestra el *accuracy* correspondiente al experimento 2 para las

35 épocas, esta vez empleando los datos con ruido añadido. El rendimiento de esta red es bastante inferior al del Experimento 1, ya que el porcentaje de acierto en validación permanece bastante por debajo del de entrenamiento, sobreentrenando desde las primeras épocas. Se puede concluir que en nuestros experimentos con Data augmentation, no hemos conseguido que las DNNs consigan un mejor resultado en validación.

Experimento	Data Augmentation	Pmiss	Pfa	DCF
1	No	0.113	0.099	**0.106**
2	Sí	0.106	0.206	0.156

Tabla 5.3: Resultados Test DNN

Para establecer el umbral óptimo se ha buscado minimizar la función de coste DCF. Además, puesto que la predicción de las DNNs resulta ser muy ruidosa, se ha aplicado un suavizado de esta con una ventana de 100 frames. Este valor también se ha optimizado para minimizar el error. Se muestra en la tabla 5 del apéndice los valores correspondientes a las dos matrices de confusión para cada uno de los experimentos realizados. En la tabla 5.3 se pueden observar los resultados en términos de DCF.

Gracias a las matrices de confusión podemos ver que el experimento 1 logra una predicción bastante equilibrada para ambas clases (TP 44.4 %, TN 44.9 %), mientras que el experimento 2 muestra una tendencia a clasificar mal los segmentos de No-Voz (TP 44.7 %, TN 39.6 %), aumentando así la tasa de falsos positivos.

5.1.2. Sistemas basados en LSTM

Como se puede ver en la tabla 5.1, los experimentos correspondientes a los sistemas basados en LSTMs son los 3 y 4. Se muestra cómo en el conjunto de test este tipo de redes son más precisas que las DNN empleadas previamente, obteniendo mejores resultados en *accuracy* y DCF.

En la tabla 2 del apéndice se puede observar un resumen del modelo empleado para la LSTM, compuesto por tres capas con 500 unidades cada una. Por otra parte, los resultados en cuanto a porcentaje de fallos y falsa aceptación se encuentran en la tabla 5.4. El experimento 4 es el que mejores resultados nos da según las métricas empleadas (Acc = 90.8 %, DCF = 0.085), por lo que se puede concluir que el *Data Augmentation* sí que refleja una leve mejora en los resultados.

En futuros experimentos se podría probar con otros tipos de ruido añadido, incluyendo voces y/ó música además de ampliar el rango de variación de SNR.

Viendo las curvas de entrenamiento representadas en la figura 6 del apéndice resulta interesante cómo en el caso del experimento 4, en las primeras épocas el *Accuracy* de validación está por encima del de entrenamiento, este comportamiento es debido al Data Augmentation. Se puede deducir que el ruido añadido hace que en épocas tempranas la red haya aprendido más características de las que

pueda haber en el batch análogo del conjunto limpio. Esta opción puede resultar muy útil en casos en los que no se cuente con bases de datos muy grandes.

Experimento	Data Augmentation	Pmiss	Pfa	DCF
3	No	0.081	0.099	0.090
4	Sí	0.094	0.077	**0.085**

Tabla 5.4: Resultados LSTM

Por último, en la tabla 5 del apéndice se encuentran los valores correspondientes a las matrices de confusión para los dos experimentos. Para obtener estos resultados esta vez no ha sido necesario optimizar el umbral (0.5), y sólo en el Experimento 4 ha habido que emplear un suavizado de las predicciones debido al ruido añadido previamente. Los resultados de ambos experimentos son muy semejantes, la aplicación de cada una de estas puede ser muy distinta dependiendo de la cantidad de datos que se tenga en la base de datos.

5.1.3. Sistemas basados en CNN

En la tabla 3 del apéndice se reflejan los modelos CNN-1 y CNN-2, compuestos por tres capas convolucionales seguidas de dos capas *fully connected*, donde lo único que cambia es la dimensión de entrada de cada capa debido a que las imágenes introducidas tienen diferente tamaño.

A continuación, en la figura 9 del apéndice se muestran las curvas de aprendizaje para los experimentos 5 y 6. Se puede ver cómo desde una primera instancia la red es capaz de distinguir entre los segmentos que contienen voz y no-voz gracias a los melgramas. En el caso de los experimentos 7 y 8, en vez de emplear los melgramas como características, se ha querido probar sobre la señal RAW de audio sin ningún tipo de procesamiento, a excepción de la normalización estándar de los batches a la entrada de la red. La señal de audio es una señal que consta de dos dimensiones, la del amplitud del audio y el tiempo, por lo que las capas convolucionales deben aplicar únicamente en una dimensión, en la del tiempo.

En la tabla 4 del apéndice se encuentra el modelo empleado para la CNN-LSTM, compuesto por tres capas convolucionales de 128 unidades cada una, dos capas LSTM de 128 unidades cada una y una capa *fully connected* de 512 unidades. Como se puede ver en la figura 12, el comportamiento de estas dos redes es muy parecido, similarmente a cómo ocurría en los anteriores experimentos con CNNs, estas redes son capaces de alcanzar buena tasa de acierto desde la primera época de entrenamiento (89 % *accuracy*). En la tabla 5.6 se muestran los resultados obtenidos para los 4 experimentos elaborados sobre CNNs, en términos de DCF sobre el conjunto de test las que mejor resultado ofrecen son las empleadas en los experimentos 5 y 8.

En la tabla del apéndice 5 se muestran las matrices de confusión correspondientes a los experimen-

Sistema	Experimento	Pmiss	Pfa	DCF
CNN-1	5	0.137	0.052	**0.095**
CNN-1	6	0.208	0.044	0.126
CNN-2	7	0.138	0.057	0.097
CNN-LSTM	8	0.132	0.058	**0.095**

Tabla 5.5: Resultados CNN

tos 5 y 6, de aquí se puede deducir que la resolución que mejores resultados da es la correspondiente a 20 frames (TP 43.2 % TN 47.2 %), es decir 0.2 segundos, ya que en el experimento 6, la reducción de la resolución provoca que la red aumente notablemente la tasa de falsos negativos (TP 39.6 %, TN 47.6 %).

Por último, en la misma tabla, se puede ver cómo el comportamiento de ambas redes es muy similiar, por lo que las últimas capas de LSTM añadidas no aportan más capacidad a la red para aprender a reconocer dichos patrones temporales.

5.1.4. Sistemas basados en Random Forest

Como último experimento para la tarea VAD se propone el uso del algoritmo Random Forest, que, como ya se ha comentado en 2, se trata de uno de los algoritmos de *Machine Learning* que mejores resultados tiene en la tarea VAD.

En la figura 5.2 se muestra la distribución de la energía por trama del conjunto IARPA Babel empleado para entrenar el algoritmo.

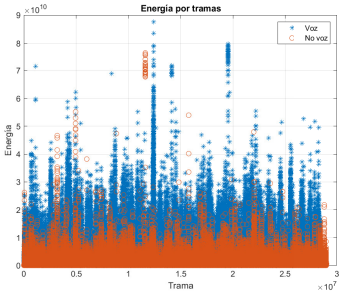

Figura 5.2: Energía por trama IARPABABEL

Experimento	Pmiss	Pfa	DCF
9	0.111	0.090	0.100

Tabla 5.6: Resultados RF

Como se puede ver en la tabla de resultados 5.6 y en los valores de la matriz de confusión en

la tabla 5 del apéndice, los resultados son bastante comparables con el resto de Redes Neuronales empleadas, incluso funcionando mejor que algunos experimentos de las CNN.

5.1.5. Experimentos fusión

Con el objetivo de minimizar el error de predicción todo lo posible, se propone fusionar los mejores sistemas descritos previamente que mejores resultados han dado. En la sección .3 del apéndice se encuentra un ejemplo de predicción para estos sistemas de un audio del conjunto de test de IARPA Babel. Teniendo en cuenta esto, los sistemas empleados han sido:

- Experimento 1: DNN sin Data Augmentation, empleando como características MFCCs.
- Experimento 4: LSTM con Data Augmentation, también se hace uso de MFCCs.
- Experimento 6: CNN-1, con resolución temporal correspondiente a 0.02 segundos.
- Experimento 7: CNN-LSTM con el audio limpio como señal de entrenamiento, sin preprocesamiento.
- Experimento 8: *Random Forest* con uso de la energía por trama como característica.

Para realizar la fusión de los sistemas, en primer lugar se han calculado las predicciones de cada uno de estos sistemas, la predicción final se decide por mayoría simple ponderada, donde las predicciones de la DNN y LSTM cuentan el doble que las de los demás sistemas por ser las que mejores resultados ofrecen.

El resultado final de porcentaje de fallos y falsa aceptación se puede observar en la tabla 5.7. Se puede ver cómo la suma de estos sistemas ofrece una ligera mejora respecto a los resultados individuales.

Experimento	Pmiss	Pfa	DCF
Fusion	0.115	0.047	0.081

Tabla 5.7: Resultados fusion

Por último, en la tabla 5 se puede ver la matriz de confusión correspondiente a la fusión de los sistemas.

5.2. Sistemas Clasificador de Género

En el siguiente apartado se van a describir los resultados para la tarea de clasificación de Género mediante el uso de GMMs, que es sin duda el algoritmo que mejores resultados nos ha dado teniendo en cuenta Accuracy de entrenamiento, validación y tiempo de entrenamiento total.

Se procede a visualizar los datos tal y como se puede ver en la figura 5.5, donde asimismo se puede ver que los ejes representan las diferentes medidas del pitch empleadas por trama (media, mediana y

desviación típica).

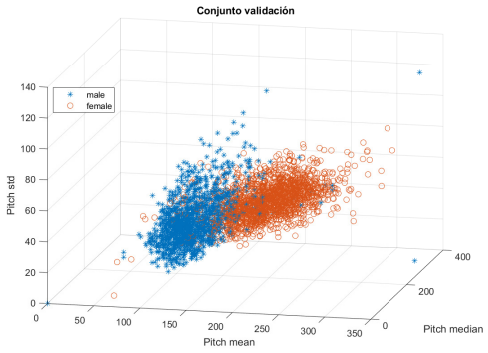

Figura 5.3: Etiquetas género Reales

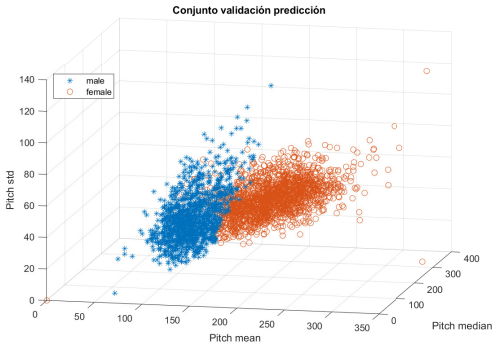

Figura 5.4: Etiquetas género Predichas

Figura 5.5: Etiquetas conjunto de validación

Como ya se ha mencionado previamente, para la tarea de clasificación de género se hará uso de GMMs, ya que es el algoritmo que mejores resultados ha dado. También se han implementado un sistema basado en *Naive Bayes* y otro basado en SVM (*Support Vector Machines*), sin embargo han dado peores resultados en términos de *accuracy*.

En nuestro caso, constamos de la base de datos SRE10 que cuenta con este tipo de etiquetado, por lo que la hemos dividido en un conjunto de entrenamiento y validación tal y como se muestra en la tabla 5.8. Por contra, no constamos de conjunto de test para evaluar finalmente el clasificador, sin embargo en el siguiente apartado se verá una demostración sobre datos de test etiquetados manualmente.

Conjunto	Nº Muestras	Porcentaje	Accuracy
Train	8983	75	0.8696
Validation	3000	25	0.8693

Tabla 5.8: Cantidad de muestras por conjunto

La GMM empleada consta de dos componentes, una por clase y se ha establecido el tipo de covarianza como *full* (completa).

5.3. Segmentador de Audio

Como demostración de los sistemas desarrollados se ha probado el sistema fusión VAD y el clasificador de género con un audio grabado a 44.1 KHz de 20 segundos. Se ha tenido que reducir la frecuencia de muestreo a 8 KHz para el sistema VAD y a 16 KHz para el clasificador de género, puesto que estas son las frecuencias de muestreo con las que se han entrenado.

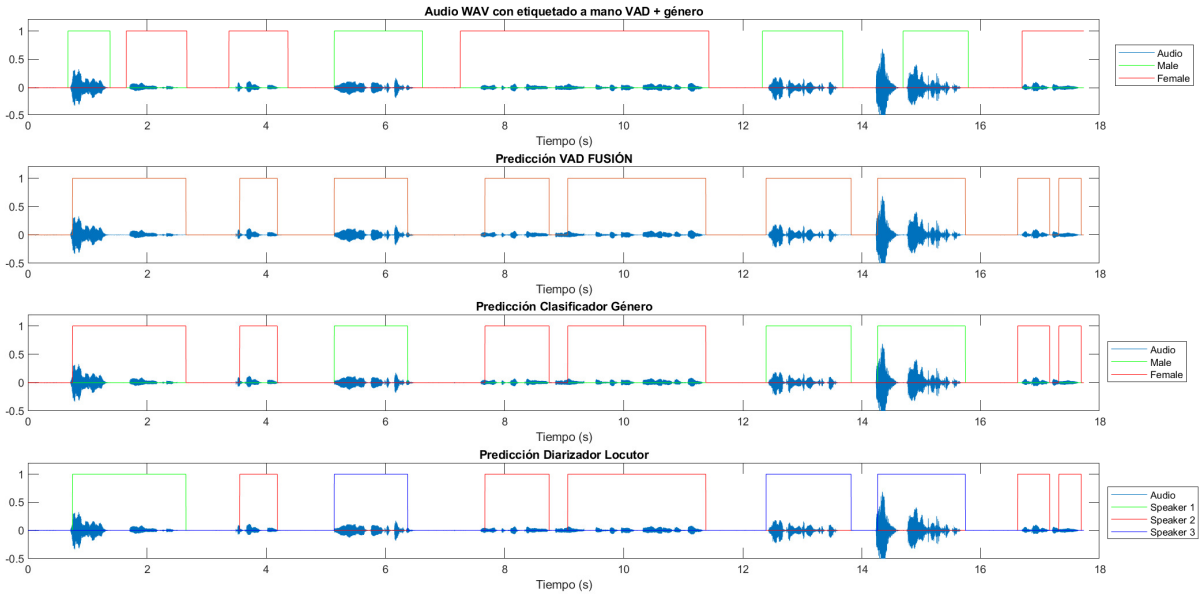

Figura 5.6: Segmentador de Audio

Se puede observar en la figura 5.6 en primer lugar una grabación de audio en la que conversan dos locutores, un hombre y una mujer. A continuación, se pasa el audio por el sistema VAD, eliminando aquellos *frames* correspondientes a silencio. Como se puede ver, la primera detección se corresponde con dos interacciones de los locutores en realidad, sin embargo, al detectarse como una sola, inducirá a error en el caso del clasificador de género y del diarizador, reconociéndose como un locutor extra. El resto de etiquetado se predice satisfactoriamente con muy buena precisión.

6

Conclusiones y trabajo futuro

6.1. Conclusiones

Como conclusión general del proyecto se puede decir que se ha conseguido implementar un Segmentador de Audio empleando diferentes técnicas de Machine Learning, dos temas punteros en la investigación actual y con numerosas aplicaciones en entornos reales. Este segmentador está compuesto por 3 subsistemas, cada uno encargado de una tarea en concreto.

En primer lugar, se ha desarrollado un sistema de Detección de Actividad de Voz basado en redes neuronales y en otros métodos de clasificación de Machine Learning como Random Forest. Para ello, se ha tomado como *baseline* una DNN implementada previamente con la librería de Python, Keras. Además, esta red se ha comparado con otra LSTM, con tres CNN (cada una diseñada para una resolución o característica) y un sistema Random Forest. Se han obtenido precisiones de hasta un 91.8 % en clasificación por tramas en el conjunto de test fusionando los mejores experimentos, en el caso de la mejor red LSTM se ha obtenido un 90.8 % en el conjunto de test.

Para concluir acerca de cuál ha sido el mejor, se han mostrado unas comparaciones y valoraciones de los valores más representativos del rendimiento de cada sistema: DNN, LSTM con/sin *Data Augmentation*, CNN-1, CNN-2, CNN-LSTM y *Random Forest*.

En los resultados expuestos en el apartado *Pruebas y Resultados* se observa que la red con peor DCF es la CNN-1 entrenada con melgramas con una resolución temporal de 0.05 segundos, sin embargo en el caso de una resolución de 0.2 segundos la medida mejora. Se han empleado dos resoluciones distintas ya que en la tarea de Detección de Actividad de Voz la información importante es la del intervalo de cambio de voz a no voz y viceversa, por lo que resultaba interesante evaluar la resolución que permitiera a la red clasificar mejor esta zona crítica. Se observa que el mejor experimento se trata de la LSTM entrenada con datos a los cuales se les ha aplicado un *Data Augmentation* con ruido de fondo. Esto quiere decir que la arquitectura de LSTM empleada es capaz de separar mejor las dos clases. En el caso del *accuracy*, la mayoría de los experimentos están en torno al 90 % de precisión en el conjunto de test. De nuevo, los experimentos con LSTM siguen dando muy buenos resultados, aunque las CNN entrenadas con melgramas y la forma de onda del audio muestran resultados muy similares.

En segundo lugar, se ha implementado un sistema para la tarea de clasificación de género mediante el uso de GMMs. En este caso, no se dispone de conjunto de test, por lo que se ha evaluado en función del *accuracy* en el conjunto de validación. Se han obtenido precisiones de hasta un 86.93 % mediante el sistema basado en GMM entrenado con el *pitch* de la señal de audio. Este sistema se ha implementado mediante la librería de Python, *Sci-kit learn*. También se han evaluado otros sistemas diseñados con otros métodos de *Machine Learning* como *SVM* o *Naive Bayes*, además entrenados con otros tipos de características como los MFCC, sin embargo, el que mejores resultados ha ofrecido ha sido la GMM, gracias en parte a la distribución del *pitch* por género representada en la figura 5.5.

Por último, se ha empleado un sistema diarizador de locutor implementado previamente mediante la librería *Sci-kit learn* y entrenado con *i-vectors*. Este diarizador se encarga de reconocer los distintos locutores que pueda haber en el audio. Como prueba del funcionamiento del segmentador completo, en la figura 5.6 se muestra una demostración. Para ello, en una primera instancia el audio se segmenta según el sistema VAD fusión y posteriormente estos segmentos se pasan tanto al clasificador de género como al diarizador.

Completados los distintos experimentos y teniendo en cuenta los resultados, se puede afirmar los siguiente. En el caso de la tarea VAD, la red neuronal recurrente LSTM empleada con Data Augmentation es la que mejores resultados da en cuanto a DCF. Aún así, los distintos modelos de CNN empleados ofrecen resultados muy similares, incluso con menor tiempo de entrenamiento y menor coste computacional. También cabe destacar que el algoritmo Random Forest ofrece mejores resultados que la DNN de la que se partía, lo que nos indica que para nuestra base de datos se pueden emplear técnicas de *Deep Learning*, sin embargo algoritmos más simples nos pueden aportar resultados cercanos en términos de precisión. En el caso de la tarea de clasificación de género, el modelo que mejores resultados da en cuanto a *accuracy* es la GMM, principalmente debido a la distribución de los datos del *pitch* en hombres y mujeres.

6.2. Trabajo Futuro

Existen muchos modelos de ANN y de *Machine Learning* para los que se puede implementar las tareas de Detección de Actividad de Voz, clasificación de género y diarización de locutor. En este Trabajo de Fin de Grado, para el caso de la tarea VAD se han usado seis tipos de modelos (DNN, LSTM, CNN-1, CNN-2, CNN-LSTM y *Random Forest*), por lo que quedan muchas arquitecturas que explorar e hiperparámetros que variar, así como nuevas metodologías que desarrollar para la tarea.

En el caso del sistema clasificador de género, se han evaluado distintos algoritmos de Machine Learning, GMMs, SVMs y *Naive Bayes*. Como trabajo futuro se podría explorar mediante *Deep Learning* esta tarea, evaluando como se ha hecho para VAD el funcionamiento de diferentes arquitecturas de redes y variando los hiperparámetros para encontrar los que mejor se adecúen a la tarea.

Bibliografía

[1] N. S. Ibrahim and D. A. Ramli, "I-vector Extraction for Speaker Recognition Based on Dimensionality Reduction," *Procedia Computer Science*, vol. 126, pp. 1534–1540, 2018.

[2] O. Vinyals, S. V. Ravuri, and D. Povey, "Revisiting recurrent neural networks for robust ASR International Computer Science Institute , Berkeley , CA , USA EECS Department , University of California - Berkeley , Berkeley , CA , USA," *Network*, pp. 4085–4088, 2012.

[3] T. Kristjansson, S. Deligne, and P. Olsen, "Voicing features for robust speech detection," *9th European Conference on Speech Communication and Technology*, vol. 10601, no. 2, pp. 369–372, 2005.

[4] W. Xiong, F. A. L. Wu, X. H. J. Droppo, and A. Stolcke, "The Microsoft 2017 conversational speech recognition system ," pp. 5934–5938, 2018.

[5] D. de Benito-Gorron, A. Lozano-Diez, D. T. Toledano, and J. Gonzalez-Rodriguez, "Exploring convolutional, recurrent, and hybrid deep neural networks for speech and music detection in a large audio dataset," *Eurasip Journal on Audio, Speech, and Music Processing*, vol. 2019, no. 1, pp. 1–18, 2019.

[6] D. Ying, Y. Yan, J. Dang, and F. K. Soong, "Voice Activity Detection Based on an Unsupervised Learning Framework," vol. 19, no. 8, pp. 2624–2633, 2011.

[7] H. Ding, K. Yamamoto, and M. Akamine, "Comparative evaluation of different methods for voice activity detection," *Proceedings of the Annual Conference of the International Speech Communication Association, INTERSPEECH*, pp. 107–110, 2008.

[8] W. S. McCulloch and W. Pitts, "A logical calculus of the ideas immanent in nervous activity," *The Bulletin of Mathematical Biophysics*, vol. 5, no. 4, pp. 115–133, 1943.

[9] F. Rosenblatt, "Principles of Neurodynamics: Perceptrons And the Theory of Brain Mechanisms," *Washington: Spartan Books*, vol. 1, no. 2, pp. 118–148, 1962.

[10] Yoshua Bengio, Patrice Simard, and Paolo Frasconi, "Learning Long-term Dependencies with Gradient Descent is Difficult," 2014.

[11] S. Hochreiter and J. Schmidhuber, "Long Short-Term Memory," *Neural Computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[12] S. J. Pan and Q. Yang, "A survey on transfer learning," *IEEE Transactions on Knowledge and Data Engineering*, vol. 22, no. 10, pp. 1345–1359, 2010.

[13] A. Escudero-Barrero, A. Lozano-Diez, R. Zazo, J. Franco-Pedroso, D. T. Toledano, and J. González-Rodríguez, "Speech activity detection combining DNN-trained and rule-based voice and music detectors at NIST OPENSAT 2017," pp. 3–5, 2017.

[14] R. Zazo, T. N. Sainath, G. Simko, and C. Parada, "Feature learning with raw-waveform CLDNNs for Voice Activity Detection," *Proceedings of the Annual Conference of the International Speech*

Communication Association, INTERSPEECH, vol. 08-12-September-2016, pp. 3668–3672, 2016.

[15] G. B. Wang and W. Q. Zhang, "A Fusion Model for Robust Voice Activity Detection," *2019 IEEE 19th International Symposium on Signal Processing and Information Technology, ISSPIT 2019*, 2019.

[16] G. B. Wang and W. Q. Zhang, "An RNN and CRNN based approach to robust voice activity detection," *2019 Asia-Pacific Signal and Information Processing Association Annual Summit and Conference, APSIPA ASC 2019*, no. November, pp. 1347–1350, 2019.

[17] A. L. Wiener and M., "Classification and Regression by randomForest. R News 2," vol. 3, no. December 2002, pp. 18–22, 2003.

[18] L. Breiman, "Using iterated bagging to debias regressions," *Machine Learning*, vol. 45, no. 3, pp. 261–277, 2001.

[19] F. Saki and N. Kehtarnavaz, "Background noise classification using random forest tree classifier for cochlear implant applications," *ICASSP, IEEE International Conference on Acoustics, Speech and Signal Processing - Proceedings*, pp. 3591–3595, 2014.

[20] A. T. Ruiz, J. Equihua, S. Martínez, E. Robredo, G. Palm, and F. Schwenker, "Detection of bat acoustics signals using voice activity detection techniques with random forests classification," in *Progress in Pattern Recognition, Image Analysis, Computer Vision, and Applications* (R. Vera-Rodriguez, J. Fierrez, and A. Morales, eds.), (Cham), pp. 253–261, Springer International Publishing, 2019.

[21] H. Phan, M. Maaß, R. Mazur, and A. Mertins, "Random regression forests for acoustic event detection and classification," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 23, no. 1, pp. 20–31, 2015.

[22] D. G. Childers, Ke Wu, K. S. Bae, and D. M. Hicks, "Automatic recognition of gender by voice," in *ICASSP-88., International Conference on Acoustics, Speech, and Signal Processing*, pp. 603–606 vol.1, April 1988.

[23] J. H. Chang, Nam Soo Kim, and S. K Mitra, "Pitch estimation of speech signal based on adaptive lattice notch filter," *Signal Processing*, vol. 85, no. 3, pp. 637–641, 2005.

[24] P. Ghahremani, B. Babaali, D. Povey, K. Riedhammer, J. Trmal, and S. Khudanpur, "A Pitch Extraction Algorithm Tuned for ASR," *Icassp 2014*, 2014.

[25] Y. L. Shue and M. Iseli, "The role of voice source measures on automatic gender classification," *ICASSP, IEEE International Conference on Acoustics, Speech and Signal Processing - Proceedings*, pp. 4493–4496, 2008.

[26] G. Aggarwal and R. Vig, "Acoustic Methodologies for Classifying Gender and Emotions using Machine Learning Algorithms," *Proceedings - 2019 Amity International Conference on Artificial Intelligence, AICAI 2019*, pp. 672–677, 2019.

[27] Y. M. Zeng, Z. Y. Wu, T. Falk, and W. Y. Chan, "Robust GMM based gender classification using pitch and RASTA-PLP parameters of speech," *Proceedings of the 2006 International Conference on Machine Learning and Cybernetics*, vol. 2006, no. August, pp. 3376–3379, 2006.

[28] X. Bao and J. Zhu, “A novel voice activity detection based on phoneme recognition using statistical model,” *Eurasip Journal on Audio, Speech, and Music Processing*, vol. 2012, no. 1, pp. 1–10, 2012.

[29] J. Pribil, A. Přibilová, and J. Matousek, “Gmm-based speaker gender and age classification after voice conversion,” pp. 1–5, 07 2016.

[30] S. Misra, T. Das, P. Saha, U. Baruah, and R. H. Laskar, “Comparison of MFCC and LPCC for a fixed phrase speaker verification system, time complexity and failure analysis,” *IEEE International Conference on Circuit, Power and Computing Technologies, ICCPCT 2015*, pp. 1–4, 2015.

[31] D. Snyder, G. Chen, and D. Povey, “MUSAN: A Music, Speech, and Noise Corpus,” 2015. arXiv:1510.08484v1.

[32] V. Sze, Y. H. Chen, T. J. Yang, and J. S. Emer, “Efficient Processing of Deep Neural Networks: A Tutorial and Survey,” *Proceedings of the IEEE*, vol. 105, no. 12, pp. 2295–2329, 2017.

Acrónimos

ANN Artificial Neural Network.

CNN Convolutional Neural Network.

DCF Detection Cost Function.

DNN Deep Neural Network.

DT Decision Trees.

FFT Fast Fourier Transform.

GMM Gaussian Mixture Model.

HMM Hidden Markov Model.

LPC Linear Prediction Coefficients.

LPCC Linear Prediction Cepstrun Coefficients.

LSTM Long Short Term Memory.

MFCC Mel Frequency Cepstral Coefficients.

NCCF Normalized Cross Correlation Function.

RNN Recurrent Neural Network.

SNR Signal To Noise Ratio.

STAM Short-Time Average Magnitude.

STFT Short-Time Fourier Transform.

STZR Short-Time Zero Crossing Rate.

SVM Support Vector Machine.

VAD Voice Activity Detection.

Apéndices

.1. Modelos Redes Neuronales

En este apartado del anexo se procede a mostrar para cada uno de los modelos empleados de Redes Neuronales sus correspondientes curvas de entrenamiento así como la composición del modelo empleado.

A continuación se representan figuras y tablas correspondientes a los experimentos con DNN.

Curvas de entrenamiento correspondientes a los experimentos 1 y 2

Figura 1: Curva entrenamiento Exp.1

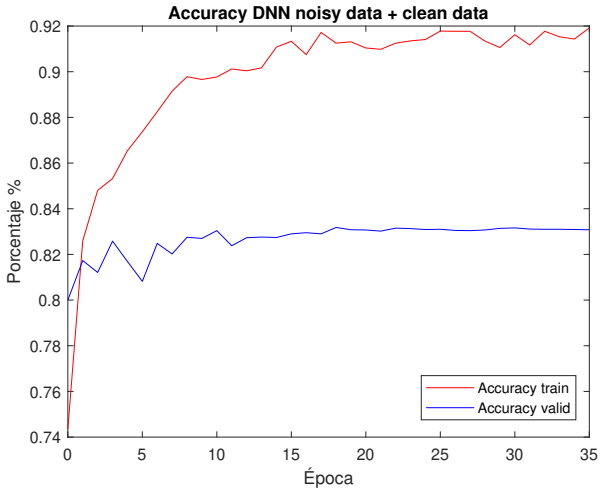

Figura 2: Curva entrenamiento Exp.2

Figura 3: Accuracy DNN

Capas empleadas, función de activación asociada y logaritmo del número de parámetros

Capa	Dimensión salida	p	Activación
Dense_1	(0,500)	5.49	Sigmoid
Dense_2	(0,500)	5.39	ReLu
Dense_3	(0,500)	5.39	ReLu
Dense_4	(0,500)	3	SoftMax

Tabla 1: Modelo DNN

A continuación se representan figuras y tablas correspondientes a los experimentos con LSTM y por último con CNN.

Curvas de entrenamiento correspondientes a los experimentos 3 y 4

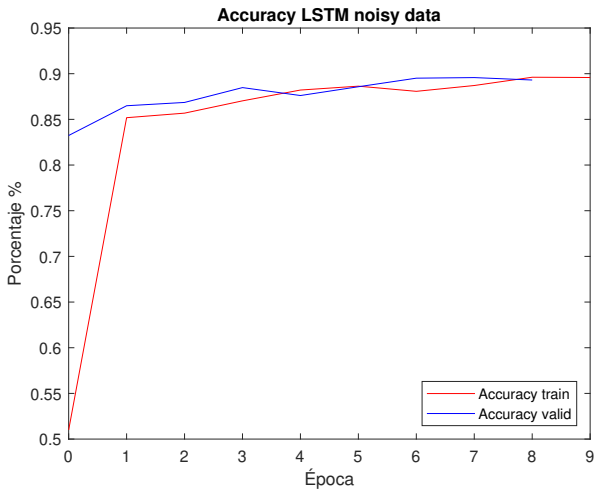

Figura 4: Curva entrenamiento Exp.3

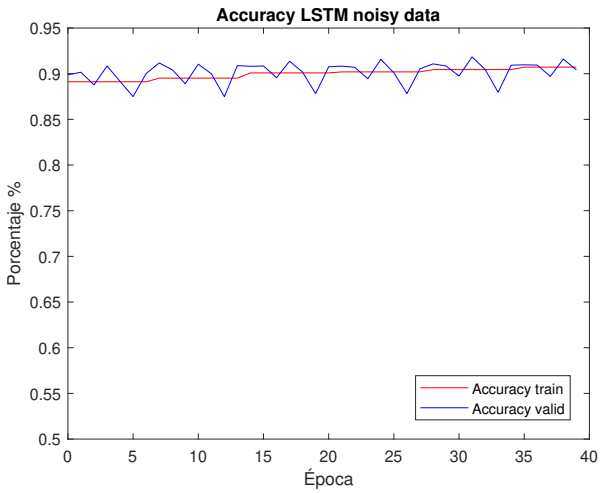

Figura 5: Curva entrenamiento Exp.4

Figura 6: Accuracy LSTM

Capas empleadas, función de activación asociada y logaritmo del número de parámetros

Capa	Dimensión salida	p	Activación
$LSTM_1$	(0,0,500)	6.01	Sigmoid
$LSTM_2$	(0,0,500)	6.30	Sigmoid
$LSTM_3$	(0,0,500)	6.30	Sigmoid
Time distributed	(0,0,2)	3	SoftMax

Tabla 2: Modelo LSTM

Capas empleadas, función de activación asociada y logaritmo del número de parámetros

Capa	Dimensión salida	p	Activación
$Conv2D_1$	(0,126,20/5,64)	2.58/2.42	ReLu
$MaxPooling2D_2$	(0,124,20/5,64)	-	-
$Conv2D_3$	(0,122,20/5,64)	4.78/4.05	ReLu
$MaxPooling2D_4$	(0,120,20/5,64)	-	-
$Conv2D_5$	(0,118,20/5,64)	4.78/4.09	ReLu
$MaxPooling2D_6$	(0,116,20/5,64)	-	-
$Dense_7$	(0,512)	7.88/7.03	ReLu
$Dense_8$	(0,2)	3.01	SoftMax

Tabla 3: Modelo CNN-1

Curvas de entrenamiento correspondientes a los experimentos 5 y 6

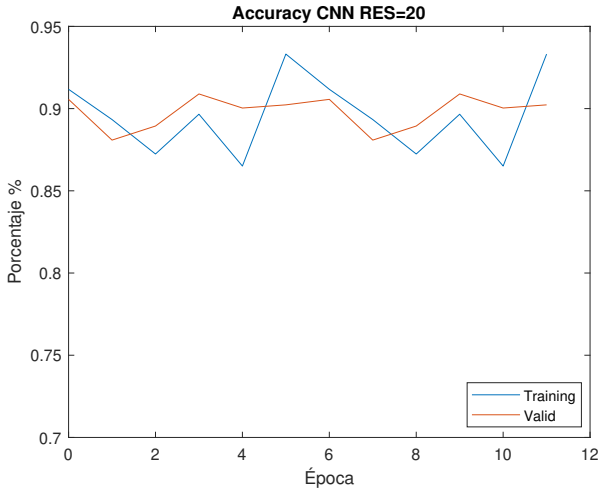

Figura 7: Curva entrenamiento Exp.5

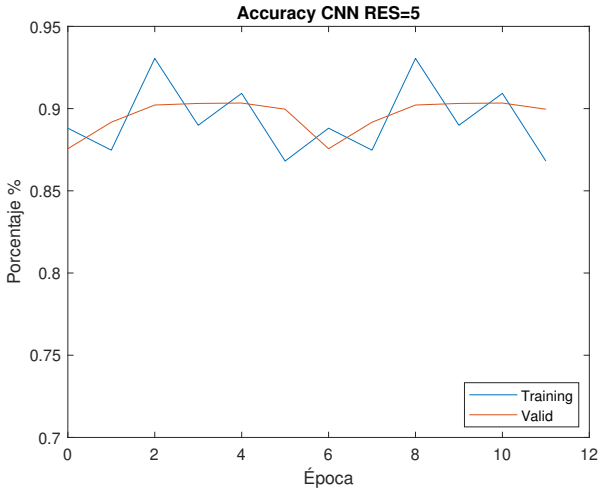

Figura 8: Curva entrenamiento Exp.6

Figura 9: Accuracy CNN-1

Curvas de entrenamiento correspondientes a los experimentos 7 y 8

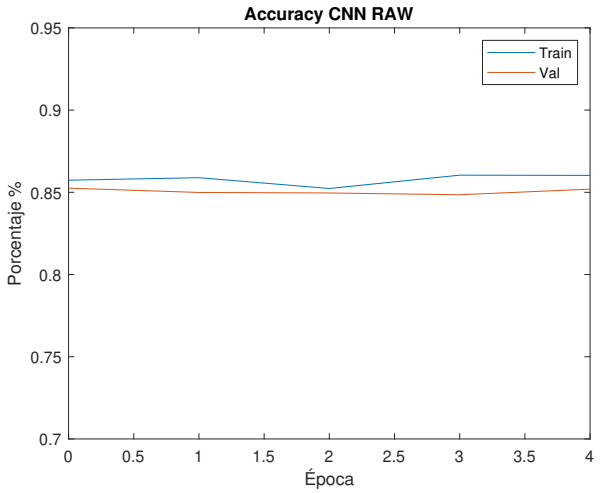

Figura 10: Curva entrenamiento Exp.7

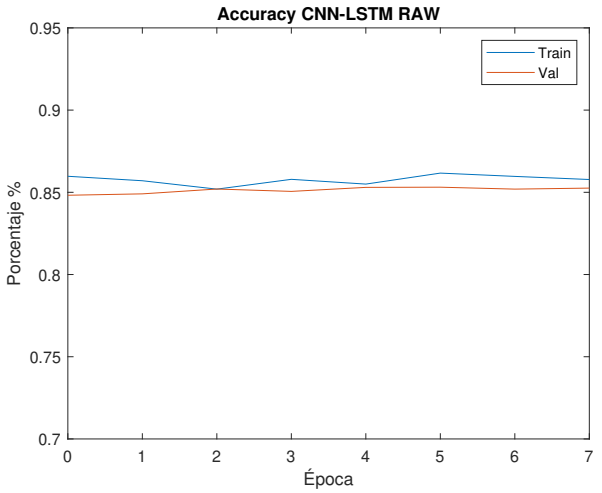

Figura 11: Curva entrenamiento Exp.8

Figura 12: Accuracy CNN-RAW

Capas empleadas, función de activación asociada y logaritmo del número de parámetros

Capa	Dimensión salida	p	Activación
Conv1D_1	(None, 77, 128)	2.80	ReLu
MaxPooling1D_2	(None, 76, 128)		-
Conv1D_3	(None, 69, 128)	5.11	ReLu
MaxPooling1D_4	(None, 69, 128)		-
Conv1D_5	(None, 54, 128)	5.41	ReLu
MaxPooling1D_6	(None, 54, 128)		-
LSTM_7	(None, 54, 128)	5.11	Sigmoid
LSTM_8	(None, 54, 128)	5.11	Sigmoid
Dense_9	(None, 54, 512)	4.81	ReLU
Flatten_{10}	(None, 27648)	-	-
Dense_{11}	(None, 2)	4.74	SoftMax

Tabla 4: Modelo CNN-LSTM

.2. Resultados

En la tabla siguiente se muestran las matrices de confusión correspondientes a los experimentos.

Sistema	Experimento	TP	FP	TN	FN
DNN	1	44.4	5.7	44.9	4.9
DNN	2	44.7	10.3	39.6	5.3
LSTM	3	44.9	4.4	45.4	5.1
LSTM	4	**45.9**	4.9	44.9	**4.1**
CNN-1	5	43.2	2.6	47.2	6.9
CNN-1	6	39.6	**2.2**	**47.6**	10.4
CNN-2	7	43.8	3.1	46.7	6.2
CNN-LSTM	8	43.6	3	46.8	6.4
RF	9	44.5	4.5	45.4	5.5
FUSIÓN	10	44.3	2.3	47.5	5.7

Tabla 5: Matrices confusión

En la tabla siguiente se muestra el *accuracy* para cada conjunto de datos para todos los experimentos.

Exp.	Red/Algoritmo	Data Augmentation	Features	Acc. Train	Acc. Val.	Acc. Test
1	DNN	No	MFCC's	92.08	90.49	89.3
2	DNN	Sí	MFCC's	91.92	83.08	84.3
3	LSTM	No	MFCC's	89.58	86.72	**90.8**
4	LSTM	Sí	MFCC's	92.18	**91.55**	90.3
5	CNN-1	No	Melgramas	92.04	89.04	90.4
6	CNN-1	No	Melgramas	**92.29**	90.51	87.2
7	CNN-2	No	RAW-Audio	86.50	85.50	90.5
8	CNN-LSTM	No	RAW-Audio	87.44	85.75	90.4
9	RF	No	Energía	87.1	84.56	89.9

Tabla 6: Accuracy

.3. Representaciones Gráficas

En este apartado se incluyen las distintas representaciones gráficas de las predicciones para cada uno de los sistemas sobre un archivo del conjunto de test del conjunto IARPABABEL para la tarea VAD, así como la misma predicción para el sistema fusión final. Los experimentos representados son:

- Experimento 1, el cual consiste en una DNN entrenada con MFCCs.
- Experimento 4, LSTM con Data Augmentation entrenada con MFCCs.
- Experimento 6, CNN-1 entrenada con melgramas.
- Experimento 7, CNN-LSTM entrenada con la señal de audio RAW.
- Experimento 8, Random Forest entrenado con la energía por trama.

Forma de onda, etiquetado real, probabilidad y etiquetado predicho del experimento 1

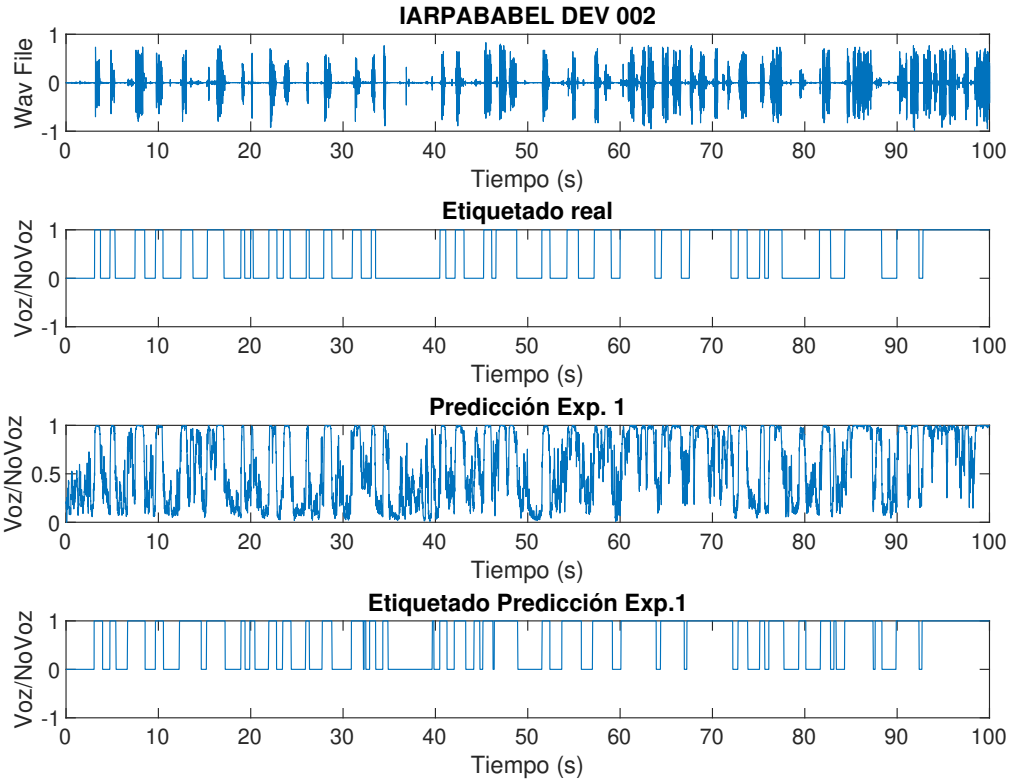

Figura 13: Predicción IARPABABEL DEV-002 Exp.1

Forma de onda, etiquetado real, probabilidad y etiquetado predicho del experimento 4

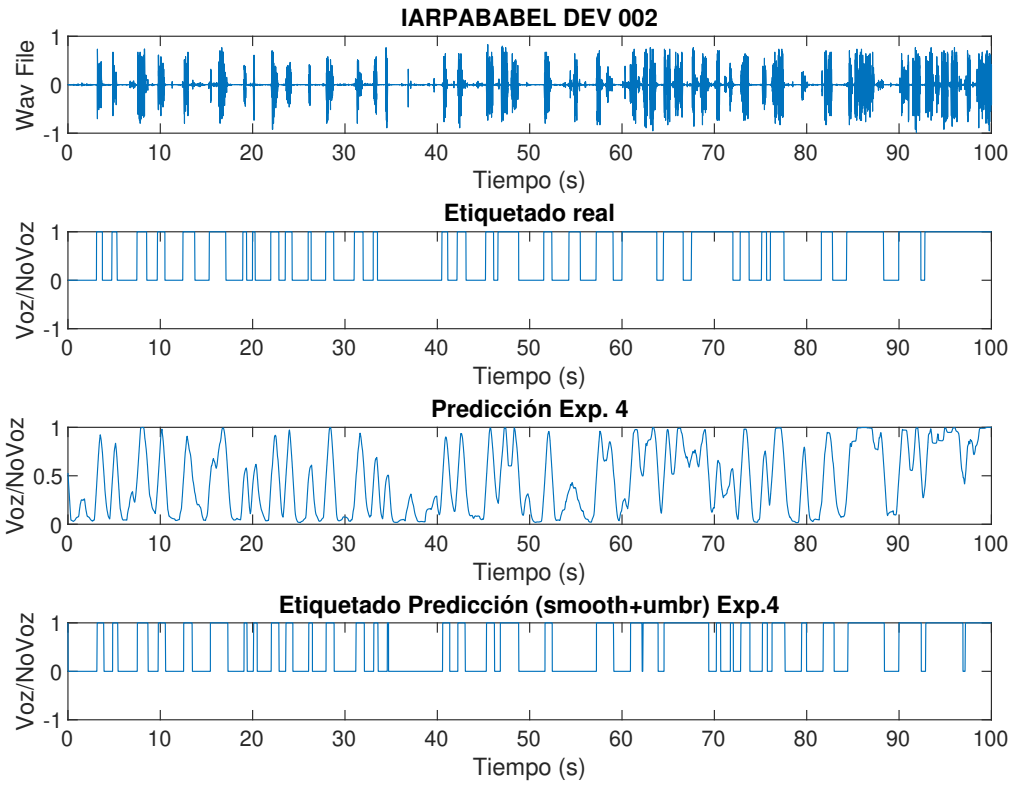

Figura 14: Predicción IARPABABEL DEV-002 Exp.4

Forma de onda, etiquetado real, probabilidad y etiquetado predicho del experimento 6

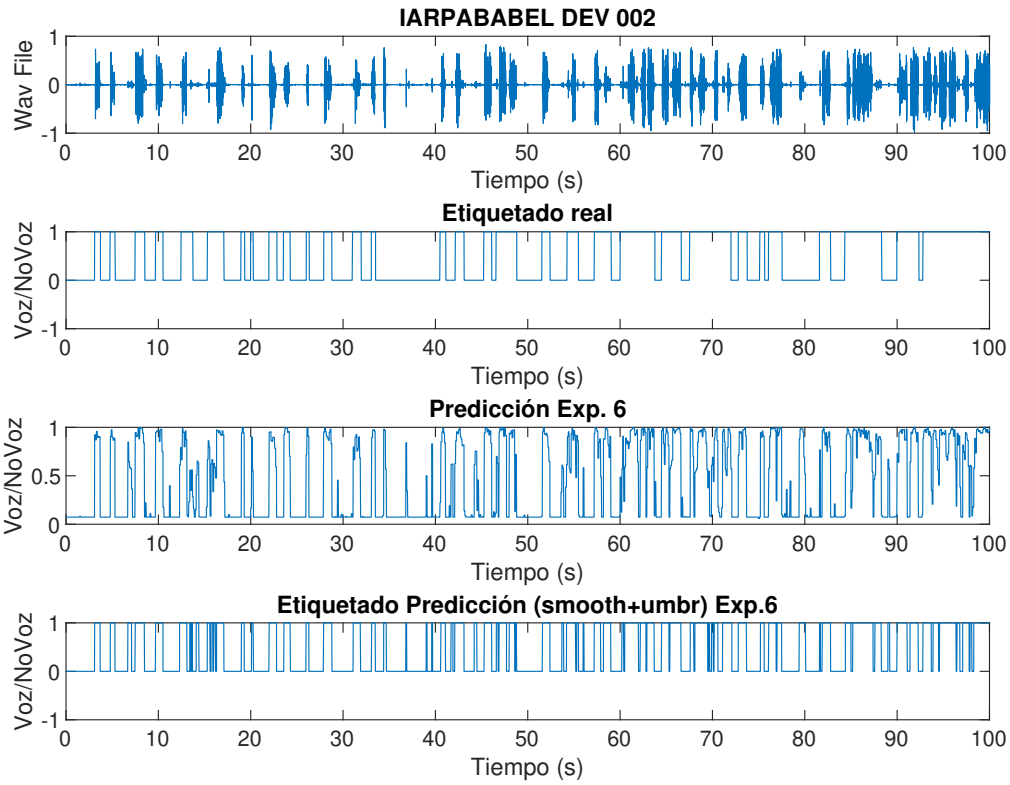

Figura 15: Predicción IARPABABEL DEV-002 Exp.6

Forma de onda, etiquetado real, probabilidad y etiquetado predicho del experimento 7

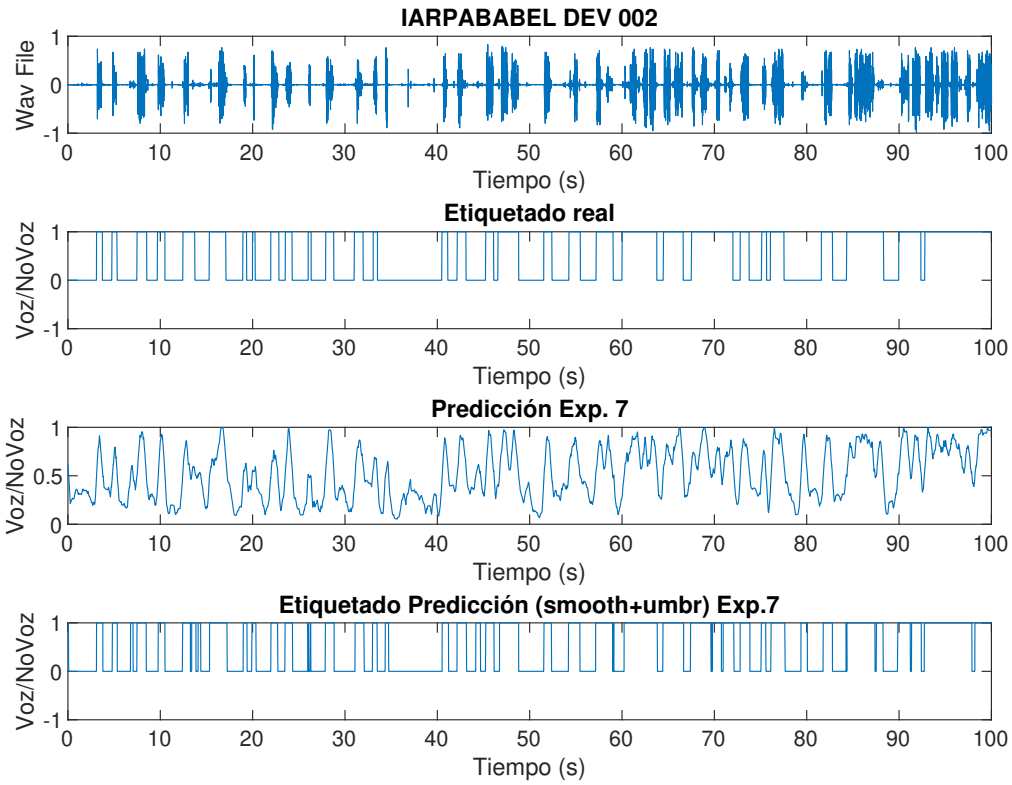

Figura 16: Predicción IARPABABEL DEV-002 Exp.7

Forma de onda, etiquetado real, probabilidad y etiquetado predicho del experimento 8

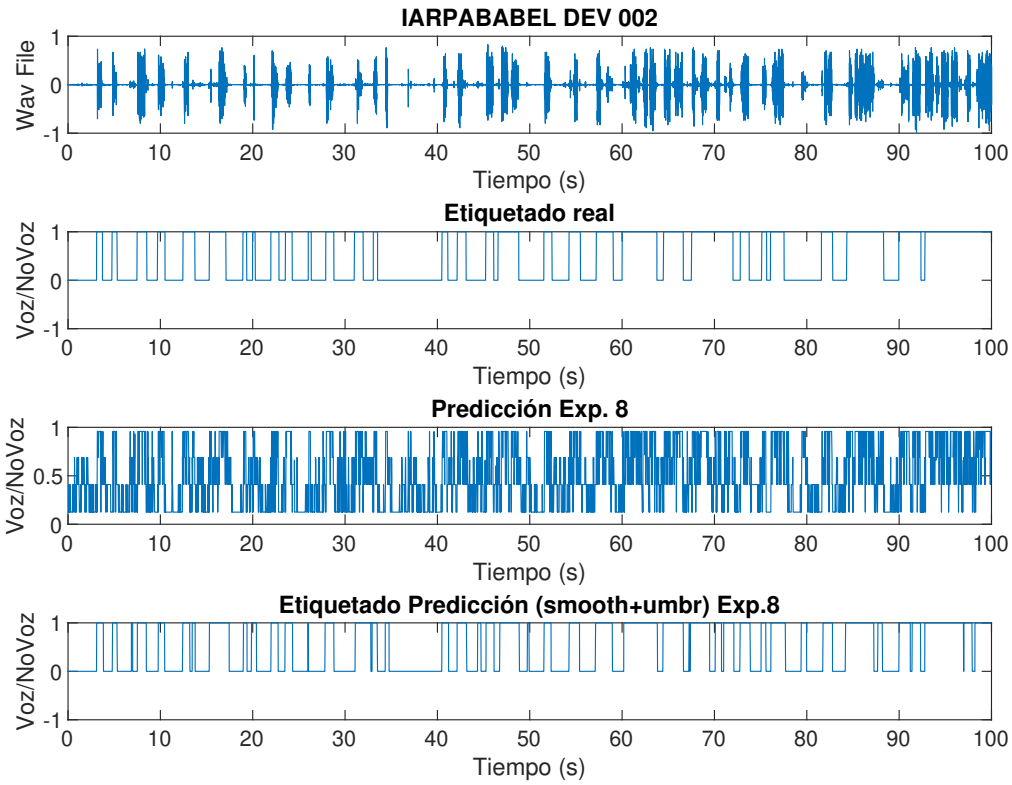

Figura 17: Predicción IARPABABEL DEV-002 Exp.8

Forma de onda, etiquetado real y etiquetado predicho del experimento fusión

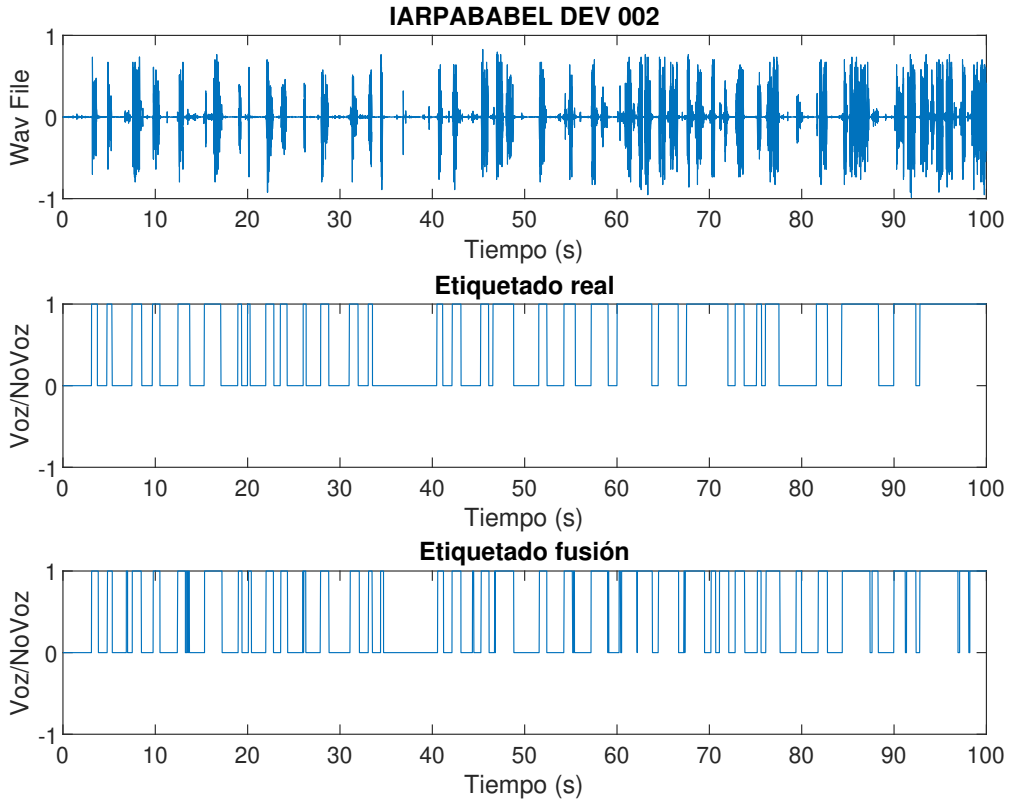

Figura 18: Predicción IARPABABEL DEV-002 Exp.Fusión