UNIVERSIDAD POLITÉCNICA DE CARTAGENA

Escuela Técnica Superior de Ingeniería Industrial

Reconocimiento automático de locutor a través de aprendizaje automático mediante redes neuronales empleando el paquete de software libre Kaldi

TRABAJO FIN DE MÁSTER

MÁSTER EN INGENIERÍA INDUSTRIAL

Autor: **Rubén Jiménez Andreu**
Director: Francisco Periago Esparza
Codirector: Roberto Javier Font Ruiz

Cartagena, 18 de septiembre de 2017

Proyecto Fin de Máster: Reconocimiento automático de locutor a través de aprendizaje automático mediante redes neuronales empleando el paquete de software libre Kaldi

Rubén Jiménez Andreu [1]
Co-director: Francisco Periago Esparza[2]
Co-director: Roberto Javier Font Ruiz [3]

18 de septiembre de 2017

[1]Universidad Politécnica de Cartagena {rubenjmnz93@gmail.com}
[2]Universidad Politécnica de Cartagena {f.periago@upct.es}
[3]Biometric Vox S.L. {roberto.font@biometricvox.com}

Resumen

El objetivo del presente Proyecto Fin de Máster es presentar algunas técnicas para reconocimiento de hablante empleando redes neuronales y *deep learning*. Como referencia se emplean i-vectors y un modelo universal basado en mezclas gaussianas (GMM-UBM) como método del estado del arte.

El proceso de entrenamiento es realizado con la base de datos de audiolibros LibriSpeech. Esta base también se emplea para evaluar los modelos, junto con la de *Speakers in the Wild*, con locuciones más próximos a situaciones reales. El software *opensource* Kaldi será la herramienta empleada para crear los modelos, junto con Python y Octave.

Este proyecto se ha realizado en colaboración con la compañía de biometría de voz Biometric Vox S.L., localizada en Espinardo, Murcia. Esta empresa ofrece soluciones de reconocimiento por voz (CheckVox) y de firma biométrica de voz (FirVox). Además, ha contado con una beca de colaboración del Departamento Matemáticas Aplicada Estadística de Universidad Politécnica de Cartagena durante el curso 2015/2016.

Se encuentra estructurado en ocho capítulos. En el capítulo 1 se hace una breve introducción a la biometría de voz, sus ventajas e inconvenientes. En el capitulo 2 se explica en qué consiste el método de los i-vector y los pasos a seguir. Una descripción general de la redes neuronales para luego explicar algunos de sus usos en reconocimiento de hablante se hará en el capítulo 3. En el capítulo 4 se expondrán las distintas formas de analizar los resultados de la biometría y en el 5 se describen detalladamente las bases de datos y el software. Para terminar, en los capítulos 6 y 7 se presentan los experimentos realizados y sus resultados, respectivamente. Por último, en el capítulo 8 se exponen las conclusiones extraídas y las posibles nuevas líneas de investigación a seguir en futuros trabajos.

Índice general

Lista de Figuras

Lista de Tablas

Lista de acrónimos

ANN: *Artificial Neural Network*, Red Neuronal Artificial.

ASR: *Automatic Speech Recognition*, Reconocimiento Automático de Hablante.

CS: *Cosine Scoring*, puntuación basada en la distancia del coseno o ángulo entre dos vectores.

DCF: *Detection Cost Function*, Función Coste de la Detección.

DET: *Detection Error Tradeoff*, Error de Detección.

EER: *Equal Error Rate*, Tasa de Equierror.

GMM: *Gaussian Mixture Model*, Modelo de Mezclas Gaussianas.

MAP: *Maximum a Posteriori*

MFCC: *Mel Frequency Cepstral Coeficients*, Coeficientes Cepstrales en la Frequencia de Mel.

minDCF: *minimum Detection Cost Function*, Valor mínimo de la función coste.

NIST: *National Institute of Standards and Technology*, Instituto Nacional de Estándares y Tecnología.

PLDA: *Probabilistic Linear Discriminant Analysis*

POI: *Person Of Interest*, Persona De Interés.

SID: *Speaker Identification*, Identificación de hablante.

SITW: *Speakers In The Wild*

SITW SRC: *Speakers In The Wild Speaker Recognition Challenge*, Reto de Reconocimiento de hablante *Speakers In The Wild*.

TDNN: *Time Delay Neural Network*, Red Neuronal con Tiempo de Retraso.

UBM: *Universal Background Model*, Modelo Universal de Fondo.

WER: *Word Error Rate*, Tasa de Error de Palabra.

Capítulo 1

Introducción

En la era de la información y del *big data*, surge la necesidad de encontrar un método eficiente de proteger nuestros datos y de identificación personal. El método clásico de usuario y contraseña no aporta la confianza suficiente en muchos casos: puede ser usada por varias personas, hackearse o ser fácil de deducir (1234, el nombre de la mascota, la fecha de nacimiento de un ser querido, etc). Los métodos de identificación y reconocimiento basados en biometría ofrecen una solución a estos problemas.

Estos métodos utilizan características únicas de cada individuo para identificarlo unívocamente. Puede ser la huella dactilar, el rostro, la voz o cualquier otra que permita diferenciarlo de los demás. No existe una tecnología que pueda considerarse superior a las demás, cada una presenta sus ventajas e inconvenientes y puede ser adecuada para ciertas situaciones e ineficiente para otras. Las mejoras en la tecnología, el avance de las técnicas de aprendizaje profundo y una gran cantidad de datos para entrenar y probar los modelos han hecho que muchas compañías se estén interesando por este campo.

Los sistemas biométricos extraen parámetros numéricos a partir de muestras de la característica identificativa para construir modelos de los sujetos que los representen. Se pueden orientar hacia la resolución de dos problemas: el reconocimiento o la identificación.

Reconocimiento. Se tiene una muestra de la característica identificativa y un modelo del sujeto. El sistema ofrece a la salida si la muestra pertenece al sujeto o no. Esto suele hacerse extrayendo los parámetros numéricos de la muestra y comparándolos con el modelo, calculando una puntuación de semejanza. Si supera cierta umbral, se considera que la muestra pertenece al sujeto.

Identificación. En este caso se tiene una muestra y varios modelos de sujetos distintos. El sistema ofrece a la salida qué sujeto es el autor de la muestra. Como el caso anterior, se extraen unos parámetros de la muestra y se comparan con los modelos. Aquel que obtenga la mejor puntuación se considera el autor.

Uno de los mayores atractivos de la biometría de voz es que actualmente prácticamente cualquier persona tiene acceso a algún tipo micrófono a través de un teléfono, un *smartphone* o un ordenador. En los últimos años, algunos teléfonos inteligentes incorporan dispositivos de lectura de huella dactilar y están apareciendo nuevos modelos con sistemas de reconocimiento de rostro con cámaras especiales para visión en 3D, incluyendo APIs para que los desarrolladores puedan emplearlos, pero no están tan extendidos. La biometría de voz puede permitir, por ejemplo, la firma de documentos legales mediante la huella de voz a través de una llamada telefónica o acceder a las cuentas del banco a través de una app registrándonos con solo registrar una frase desde casi cualquier dispositivo. Incluso con el auge del internet de las cosas, que un dispositivo sea capaz de reconocer a quien le da la orden permitiría crear distintos perfiles o un control parental efectivo.

En contra tiene que la voz no es una representación estática de las características de una persona. Las locuciones de voz tienen una evolución temporal que influye en cómo los sonidos son pronunciados, con variaciones de duración, de tono y silencios intermedios. Resolver estos inconvenientes y discretizar la voz en forma de una serie de características fijas que identifiquen al hablante es el objetivo de este tipos de sistemas, la creación de una huella de voz del sujeto.

Una clasificación general de los sistemas de biometría de voz es si son dependientes o independientes del texto. Los sistemas dependientes tienen en cuenta lo que el hablante está diciendo en la locución. Los sistemas independientes no lo tienen en consideración. En general, para cada sistema los pasos a seguir son los siguientes:

1. **Entrenamiento.** Se entrena el modelo con una amplia base de datos. Suele buscarse que haya una gran variedad de hablantes.

2. **Enrolamiento.** Se registran los hablantes que se quieren identificar extrayendo su huella vocal. No tienen por qué formar parte del grupo de entrenamiento.

3. **Reconocimiento/identificación.** De una locución se extraen sus características y se comparan con las de los usuarios enrolados, indicando finalmente si es objetivo o no objetivo.

En este proyecto se emplearan distintas aproximaciones de reconocimiento de hablante independiente del texto basadas en el método de i-vectors. En este método se crea en primer lugar un modelo universal, *Universal Background Model* (UBM), habitualmente con Modelos de Mezclas Gaussianas, *Gaussian Mixture Model* (GMM), de donde obtener los estadísticos suficientes para extraer un vector identificativo de cada locución. Los i-vector son después comparados con alguna técnica como distancia del coseno o *Probabilistic Linear Discriminant Analysis* (PLDA). El objetivo es mejorar los resultados con el uso de *deep learning* y redes neuronales del tipo *Time Delay Neural Network* (TDNN) orientadas al reconocimiento del voz, *Automatic Speech Recognition* (ASR). Estas redes reciben como entrada una serie de parámetros extraídos de fragmentos

de locuciones en un contexto temporal y ofrece a la salida la probabilidad a posteriori o *posterior* de pertenencia a una determinada clase fonética.

Se tratan tres aproximaciones diferentes:

- **Detección de la actividad de voz, VAD.** Se usan los *posterior* extraídos de la red para reconocer aquellas partes de la locución que contienen audio de las que no. Es nuevo método de VAD ofrece mejores resultados que el tradicional basado en la energía de la locución, por lo que lo sustituye lo consiguiente lo sustituirá.
- **Entrenar un GMM supervisado.** Se emplean los *posterior* para modelar el contenido fonético de forma supervisada.
- **Crear un TDNN-UBM.** En este caso se emplea la red para extraer los estadísticos suficientes para hacer la extracción de los i-vector.

Los modelos son entrenados con la base de datos de audilibros LibriSpeech y evaluados tanto con LibriSpeech como con otra base llamada *Speakers In The Wild*, usando el software libre orientado a ASR Kaldi. El análisis de resultados se efectúa con la representación gráfica en curvas *DET*, un método muy usado en biometría, el cuál permite comparar cómo de buenos son distintos sistemas a la hora de hacer el reconocimiento.

Con cada nueva aproximación se consigue mejorar la identificación. Mientras que en Librispeech la diferencia es pequeña y depende de la zona de la curva es mejor o peor, es al aplicarlas a SITW, cuyas locuciones suponen un auténtico reto, donde las TDNN muestran su verdadero potencial. Se obtiene una mejora de la calidad enorme en un campo donde es difícil de obtener este tipo de resultados favorables y es de gran relevancia por contener locuciones extraídas de situaciones reales.

Capítulo 2

Método de los i-Vectors

En este capítulo se analizará la técnica de los i-vector como método del estado del arte en la identificación de hablante independiente del texto. La virtud de este método es que permite almacenar los parámetros que caracterizan al hablante en forma de vectores de longitud fija y reducida.

El proceso consta de los siguientes pasos (Figura 2.1):

1. Extracción de características de los archivos de audio originales.
2. Detección de silencios o VAD (*Voice Activity Detection*).
3. Entrenamiento de un modelo universal basado en un mezclas gaussianas (GMM-UBM).
4. Entrenamiento del extractor de i-vector.
5. Extracción de los i-vector.
6. Puntuación.

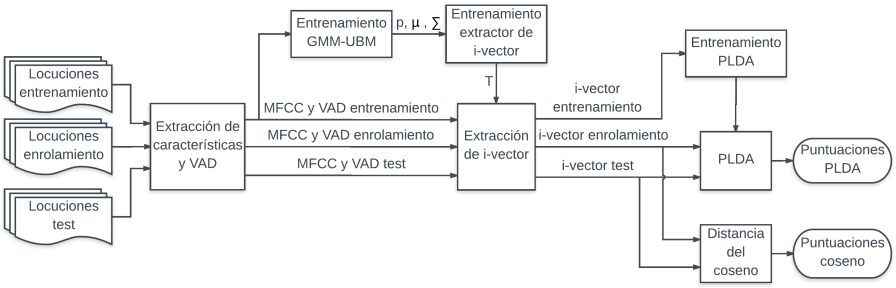


Figura 2.1: Diagrama de flujo de proceso de reconocimiento con el método de los i-vector.

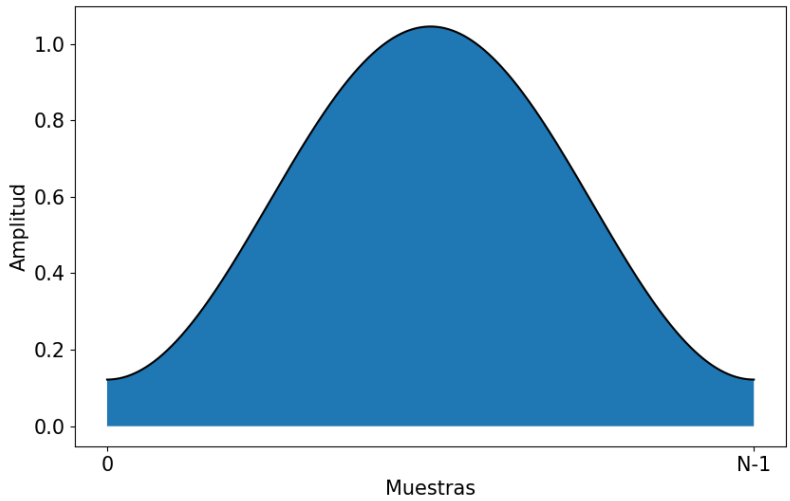


Figura 2.2: Ventana Hamming. Sigue la función $v(n) = 0,53836 - 0,46164 \cos\left(\frac{2\pi n}{N-1}\right)$, donde $n \in [0, N-1]$ y N es el número de muestras. Esta función se multiplica por el extracto de la señal de audio, dando más énfasis a la parte central que a los extremos.

En las siguientes secciones se presentarán en detalle los distintos pasos del proceso.

2.1. Extracción de características

El primer paso para la identificación del hablante es convertir las señales analógicas de audio en un formato digital que nos permita trabajar con ellas. Los *Mel Frequency Cepstral Coeficients* (MFCC) [3] son los coeficiente más empleados a la hora de hacer análisis de hablante ya que son capaces de representar las frecuencias del habla de forma compacta y adaptada a la capacidades del oído humano [12]. Además, cada uno de sus pasos está pensado para ser computacionamelte eficiente.

Los pasos para obtener estos coeficientes cepstrales de Mel son los siguientes:

1. En primer lugar, se divide el audio en fragmentos o *frames* habitualmente de 25ms y con 15ms de solapamiento entre ellos. Estos valores pueden variar, aunque son los más habituales. Cada frame se pasa por una función ventana, por ejemplo, de tipo Hamming (Figura 2.2); para evitar efectos indeseados en los bordes.

2. Sobre cada frame se calcula la Trasformada Rápida de Fourier para extraer

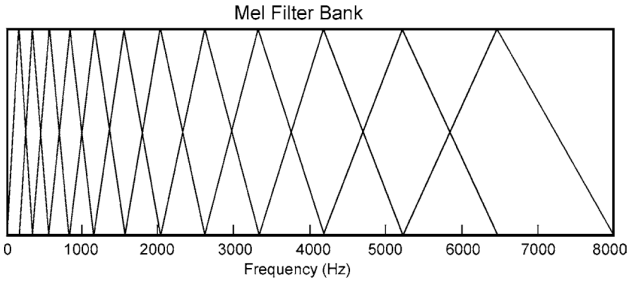


Figura 2.3: Banco de filtros de Mel.

sus frecuencias y estas son pasadas por un banco de filtros de Mel (Figura 2.3). Este banco se trata de un conjunto de funciones triangulares solapadas cuyos centros se encuentran igualmente espaciados en el dominio de las frecuencias de Mel. Esta escala de Mel es una escala musical que permite que intervalos de frecuencias espaciados exponencialmente sean percibidos como espaciados linealmente. La transformación de frecuencia f en hercios a Mel m se hace de la siguiente manera:

$$m = 1127,01048 \ln(1 + f/700) \tag{2.1}$$

3. Hecho esto, se calcula su logaritmo y se hace la transformada de coseno, obteniendo un vector de coeficientes por cada frame. El tamaño de dicho vector dependerá del número de coeficientes de la transformada discreta del coseno que se quieran calcular.

Como resultado, se tiene cada locución representada con un número variable de vectores de características MFCC, según su duración.

Es también útil calcular sus derivadas y las segundas derivadas, llamadas, respectivamente, *delta* (Δ) y *delta delta* ($\Delta\Delta$). Estas aportan información sobre las características dinámicas de la voz.

Al final del proceso, cada locución queda caracterizada por una serie de vectores, uno por cada frame, formados por la concatenación de los MFCC, los Δ y los $\Delta\Delta$.

2.2. *Voice Activity Detection* (VAD)

Durante un discurso es habitual que aparezcan pausas y periodos de silencio. Estos espacios sin voz no aportan nada a la hora de identificar al hablante y podría ser contraproducente tenerlos en cuenta durante el proceso debido al coste computacional que acarrearían. Es por ello que se realiza un proceso de detección de la actividad de la voz, VAD, para identificar aquellas partes donde se está hablando y aquellas donde no.

Puede hacerse una vez extraídos los MFCC analizando la energía de cada *frame*. Aquellos que no se superen cierto umbral serán considerados como silencio y no serán tenidos en cuenta en los cálculos posteriores.

2.3. Universal Background Model (GMM-UBM)

El siguiente paso es entrenar un UBM [19, 20]. Este modelo debe ofrecer una representación de la voz humana de la forma más general posible. Para el entrenamiento se emplean una gran cantidad de hablantes que aporten una amplia variedad de características de voz. No existe un método para determinar el número adecuado de hablantes para el entrenamiento, pero tiene que encontrarse un equilibrio entre subpoblaciones dentro del *dataset*, por ejemplo, hombres y mujeres. De esta forma se evita que haya una parcialidad y se sea lo más equitativo posible.

La manera más habitual de crearlo es mediante Modelos de Mezclas Gaussianas o GMM. La razón de su uso es que proporciona una aproximación con una forma suavizada a partir de las características del hablante, donde cada componente gaussiana individual representa una amplitud de clases acústicas que reflejan dependencia del tracto vocal del hablante. El modelo resultante recibe el nombre de GMM-UBM.

Está formado por la suma ponderada de M funciones gaussianas:

$$p(\vec{x}|\lambda) = \sum_{i=1}^{M} p_i b_i(\vec{x}) \tag{2.2}$$

donde $\vec{x}$ es un vector aleatorio D-dimensional, $b_i(\vec{x}), i = 1, ..., M$ son las densidades y $p_i, i = 1, ..., M$ son los pesos. Cada componente de densidad es una función gaussiana de la forma:

$$b_i(\vec{x}) = \frac{1}{(2\pi)^{D/2}|\Sigma_i|^{1/2}} \exp\left[-\frac{1}{2}(\vec{x} - \vec{\mu_i})'\Sigma_i(\vec{x} - \vec{\mu_i})\right] \tag{2.3}$$

con vector de medias $\vec{\mu_i}$ y matriz de covarianza Σ_i. Los pesos deben satisfacer que $\sum_{i=1}^{M} p_i = 1$. El modelo se puede expresar como:

$$\lambda = \{p_i, \mu_i, \Sigma_i\}, i = 1, ..., M \tag{2.4}$$

En el entrenamiento del GMM-UBM, el objetivo es encontrar los parámetros que mejor se adapten a los vectores de características, estimando la máxima verosimilitud. Para una secuencia de T vectores de entrenamiento $X = \vec{x_1}, ..., \vec{x_T}$, la verosimilitud es descrita como:

$$p(X|\lambda) = \prod_{t=1}^{T} p(\vec{x_t}|\lambda) \tag{2.5}$$

Sin embargo, la no linealidad hace que la maximización directa de λ no sea posible analíticamente. Hay que recurrir a métodos numéricos, empleando un

proceso iterativo conocido como algoritmo EM *Esperanza-Maximización*. Este algoritmo consiste en, partiendo de un modelo inicial λ, estimar un nuevo modelo $\overline{\lambda}$ tal que $p(X|\overline{\lambda}) > p(X|\lambda)$. En la siguiente iteración el nuevo modelo pasa a ser el modelo inicial.

En EM las siguientes funciones son utilizadas para asegurar una verosimilitud monótomamente creciente:

$$\overline{p_i} = \frac{1}{T}\sum_{t=1}^{T} p(i|\vec{x}_t, \lambda) \tag{2.6}$$

$$\vec{\overline{\mu_i}} = \frac{\sum_{t=1}^{T} p(i|\vec{x}_t, \lambda)\vec{x}_t}{\sum_{t=1}^{T} p(i|\vec{x}_t, \lambda)} \tag{2.7}$$

$$\overline{\sigma_i}^2 = \frac{\sum_{t=1}^{T} p(i|\vec{x}_t, \lambda)x_t^2}{\sum_{t=1}^{T} p(i|\vec{x}_t, \lambda)} - \overline{\mu}_i^2 \tag{2.8}$$

donde σ_i^2,x_t y μ_i son elementos de los vectores $\vec{\sigma_i^2}$,$\vec{x}_t$ y $\vec{\mu}_i$, respectivamente.

Factores críticos para la eficacia de este método es la selección del número de gaussianas del GMM y el modelo inicial para el algoritmo EM, no existiendo un método teórico para encontrar la mejor solución.

2.4. I-vectors

Los *identity vectors* o i-vectors buscan condensar toda la variabilidad presente en los datos de entrenamiento en un único vector de dimensión fija y reducida. De este modo, los i-vectors contienen no sólo información del hablante sino de otros factores como el idioma, el sexo o el canal (medio con el que se ha adquirido el audio). Técnicas como el PLDA (Sección 2.5.2) buscan eliminar esta información no deseada quedándose únicamente con la información referente al hablante.

A partir del modelo universal GMM-UBM formado por C componentes gaussianas, el vector de C medias puede ser adaptado mediante *Maximum a Posteriori* (MAP) a las características extraídas de una locución cualquiera, dando como resultado un supervector s (Figura 2.4).

La idea principal tras la metodología de los i-vector es que el supervector del GMM correspondiente a un hablante, s puede modelarse como:

$$s = \mu + \mathbf{T}w$$

donde $\mu = \left[\mu^{(1)'}, \ldots, \mu^{(C)'}\right]'$ es el vector de medias del modelo GMM-UBM de C componentes previamente calculado, $\mathbf{T} = \left[T^{(1)'}, \ldots, T^{(C)'}\right]'$ es una matriz de bajo rango que representa las M bases del espacio de variabilidad total reducido y w es un vector aleatorio de tamaño M que sigue una distribución normal $N(0; I)$. La matriz T es a veces llamada *i-vector extractor*.

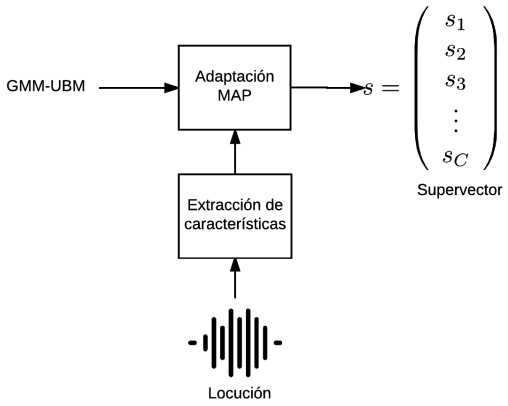


Figura 2.4: Extracción de supervectores a partir de GMM-UBM.

El i-vector es una estimación MAP (Maximum a Posteriori) de la variable w. A partir de una secuencia de observaciones de longitud variable $\chi = [x_1, \ldots, x_N]$, donde x_t es el vector de características (p. eg. MFCCs) del frame t de la locución, puede obtenerse una expresión en forma cerrada para el i-vector mediante los estadísticos suficientes de Baum-Welch de orden cero y de primer orden: $\mathbf{n}_\chi = \left[N_\chi^{(1)}, \ldots, N_\chi^{(C)}\right]'$ y $\mathbf{f}_\chi = \left[\mathbf{f}_\chi^{(1)'}, \ldots, \mathbf{f}_\chi^{(C)'}\right]'$ donde

$$N_\chi^{(c)} = \sum_t \gamma_t^{(c)} \tag{2.9}$$

$$\mathbf{f}_\chi^{(c)} = \sum_t \gamma_t^{(c)} x_t, \tag{2.10}$$

con $\gamma_t^{(c)}$ la probabilidad a posteriori de que el frame x_t haya sido generado por la componente c. De este modo, el i-vector viene dado por

$$\mathbf{w}_\chi = \mathbf{L}_\chi^{-1} \mathbf{T}' \mathbf{f}_\chi,$$

con

$$\mathbf{L}_\chi = \mathbf{I} + \sum_{c=1}^{C} N_\chi^{(c)} \mathbf{T}^{(c)'} \mathbf{T}^{(c)}.$$

Los $\gamma_t^{(c)}$ corresponden a *soft alignments* respecto a las C componentes; *soft* en el sentido de que a cada observación x_t se le asigna no una clase c, sino una probabilidad, en general no nula, de pertenecer a cada una de las C clases.

Podemos decir que las diferentes componentes gaussianas representan regiones del espacio acústico. De este modo (2.9) puede interpretarse, de forma poco rigurosa, como una estadística de paso por las diferentes regiones de esta partición del espacio acústico realizada por el modelo GMM-UBM a partir del cual se calculan los alineamientos.

2.5. Puntuación

Una vez extraídos los i-vector tanto del *enrollment* como del *test*, se necesita un método para comparar uno con otro y recibir una puntuación numérica de su semejanza. Con i-vector los métodos usados son *Cosine Distance Scoring* y *Probabilistic Linear Discriminant Analysis* (PLDA).

2.5.1. *Cosine Distance Scoring*

Este método obtiene una puntuación a partir del ángulo que forman dos i-vectors w_{taget} y w_{test}. Cuanto menor sea, más probabilidad hay de que pertenezcan al mismo hablante. Se define con la siguiente ecuación:

$$score(w_{target}, w_{test}) = \frac{w_{target} \cdot w_{test}}{||w_{target}||||w_{test}||} \tag{2.11}$$

Se fundamenta en que la magnitud de los i-vectors viene afectado por información no relevante sobre el hablante, como la sesión o el canal, siendo el ángulo el que realmente distingue a la persona. Es un método robusto, rápido y poco complejo.

2.5.2. *PLDA Scoring*

En el modelo PLDA [9, 18], dadas J locuciones de I hablantes con i-vector $x_{ij} : i = 1, ..., I; j = 1, ..., J$ de dimensión F. Estos vectores pueden descomponerse como:

$$x_{ij} = \mu + Fh_i + Gw_{ij} + \varepsilon_r \tag{2.12}$$

donde μ son las medias extraídas del modelo GMM-UBM, F es una matriz que representa el subespacio del hablante, G es una matriz que representa el subespacio dependiente de la sesión y ε_r una variable aleatoria con distribución normal con media cero y matriz de covarianza Σ representando el ruido. Por su parte, h_i y w_{ij} son vectores formados por variables aleatorias que contienen los factores del hablante y del canal.

Por tanto, dos locuciones tendrán más probabilidad de pertenecer al mismo hablante cuanto más parecidos entre sí sean sus vectores h.

Este método de puntuación es más complejo que el del coseno aunque ofrece mejores resultados en casos con más dificultad de identificación. Requiere antes del calculo de puntuaciones una fase de entrenamiento usando el algoritmo E-M.

Capítulo 3

Redes Neuronales Artificiales

En este capítulo se hará una introducción a las redes neuronales artificiales. Se expondrá la motivación y funcionamiento de la neurona artificial para luego ver cómo se combinan entre sí para formar redes. A continuación, se explica el proceso de entrenamiento de la red. Por último, veremos las redes neuronales profundas y algunos de sus usos en el reconocimiento de hablante.

3.1. Descripción de la neurona

Las redes neuronales artificiales surgieron en el campo de las matemáticas y la computación como una forma de imitación de las redes neuronales biológicas que forman el sistema nervioso de los seres vivos. El primer modelo matemático de red neuronal fue realizado por Warren McCulloch y Walter Pitts en 1943. Lo que ha hecho a este tipo de modelos populares es su capacidad de aprendizaje y el aumento en la potencia de computación de las CPU y GPU de los últimos años que permiten aumentar su complejidad.

Las neuronas biológicas están formadas por un axón, un núcleo y unas dendritas. El axón recibe señales electroquímicas provenientes de otras neuronas, las cuales son descodificadas e interpretadas en el núcleo para después generar otra señal electroquímica que es transmitida a través de las dendritas a otras neuronas, que realizan el mismo proceso. A partir de células sencillas se crean grandes redes que pueden realizar tareas complejas, tales como interpretar lo que se está viendo u oyendo, mover un brazo o desarrollar una teoría matemática.

Las redes neuronales artificiales, *Artificial Neural Network* (ANN), se basan en el mismo principio. Se tiene una unidad básica, llamada neurona o perceptrón (Figura 3.1), la cual recibe una serie entradas, bien exteriores o de otras neuronas. Estas entradas, que llamaremos $x_1, x_2, \ldots, x_n$, son sumadas en el cuerpo de la neurona, multiplicándolos por un peso $w_1, w_2, \ldots, w_n$:

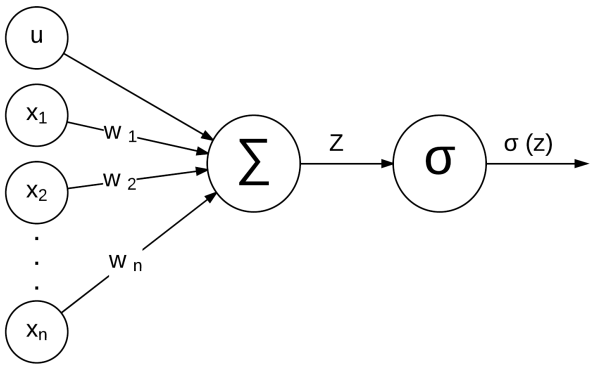


Figura 3.1: Representación gráfica de una neurona artificial. Las entradas x son multiplicadas por unos pesos w y sumadas entra sí junto con el umbral u. El resultado z pasa por una función de activación σ que devuelve el estado de la neurona.

$$z = x_1w_1 + x_2w_2 + \cdots + x_nw_n + u = \sum_{i=1}^{n} x_iw_i + u \tag{3.1}$$

donde u es un *bias* para la activación de la neurona. Otra forma de considerar este valor umbral es como una entrada $x_0 = 1$ con un peso w_0, así $u = 1 \cdot w_0$. Esta suma de productos z pasa después por una función de activación $\sigma(z)$ que representa el estado de la neurona. Este estado puede ser transmitido a otra neurona o servir como salida de la red.

Algunas de las funciones de activación empleadas son:

Función escalón: solo hay dos estados posibles, 0 o 1.

$$\sigma(z) = \begin{cases} 0; & z < 0 \\ 1; & z \geq 0 \end{cases} \tag{3.2}$$

o bien,

$$\sigma(z) = \begin{cases} -1; & z < 0 \\ 1; & z \geq 0 \end{cases} \tag{3.3}$$

Función lineal: solo permite resolver problemas lineales.

$$\sigma(z) = z \tag{3.4}$$

Puede añadirse saturación si no se quiere sobrepasar ciertos límites:

$$\sigma(z) = \begin{cases} 0; & z < 0 \\ z; & z \geq 0 \end{cases} \tag{3.5}$$

$$\sigma(z) = \begin{cases} -1; & z < -1 \\ z; & z \geq 0 \end{cases} \tag{3.6}$$

$$\sigma(z) = \begin{cases} 0; & z < 0 \\ z; & 0 \leq z < 1 \\ 1; & z \geq 1 \end{cases} \tag{3.7}$$

$$\sigma(z) = \begin{cases} -1; & z < -1 \\ z; & 0 \leq z < 1 \\ 1; & z \geq 1 \end{cases} \tag{3.8}$$

De especial importancia es la función 3.5 (Figura 3.2a), también llamada rectificador, muy empleada en aprendizaje profundo y con gran efectividad en redes convolucionales. A las neuronas con esta función se las conoce como ReLU (*rectified linear unit*, unidad lineal rectificada).

Función sigmoide logarítmica: Devuelve valores entre 0 y 1. Permite resolver problemas no lineales (Figura 3.2b).

$$\sigma(z) = \frac{1}{1 + e^{-z}} \tag{3.9}$$

Función sigmoide tangente logarítmica: Una variación de la anterior, permite obtener valores negativos (Figura 3.2c).

$$\sigma(z) = \frac{e^{z} - e^{-z}}{e^{z} + e^{-z}} \tag{3.10}$$

PNorm: Devuelve la integral desde $-\infty$ hasta z.

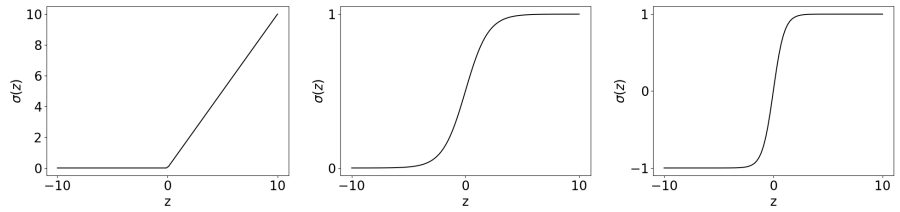
Softmax: Este tipo de activación se suele emplear en las neuronas de salida de la red. Convierte un vector de valores reales en un vector de probablidades. Para la j-neurona de la capa de salida:

$$\sigma(\vec{z})_j = \frac{e^{z_j}}{\sum_i e^{z_i}} \tag{3.11}$$

3.2. El perceptrón multicapa

El perceptrón multicapa es la estructura básica en cuanto a redes neuronales (Figura 3.3). Está formada por capas de varias neuronas cada una, donde todas las neuronas de una capa reciben de entrada el estado de todas las neuronas de la capa anterior. La información entonces se propaga hacia adelante, pasando capa a capa de la primera a la última. Este método es conocido como *propagación hacia adelante*.

Las capas existentes pueden ser de tres tipos:



(a) (b) (c)

Figura 3.2: Algunas funciones de activación de las neuronas. (a) función rectificador. (b) Función sigmoide logarítmica. (c) Función sigmoide tangente

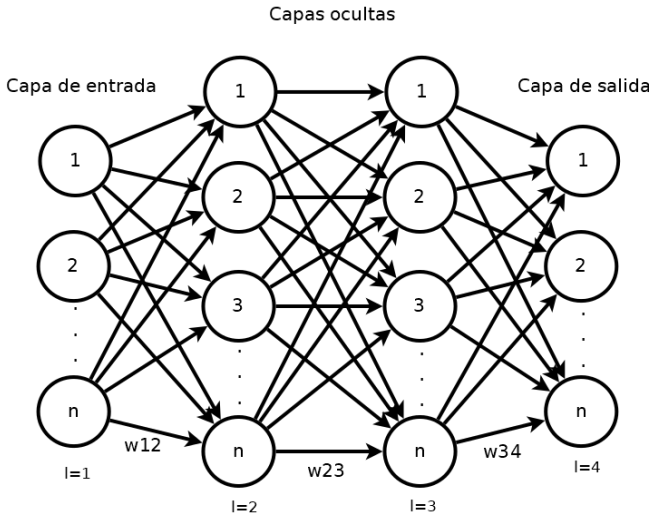


Figura 3.3: Perceptrón multicapa.

Capa de entrada: Es la primera capa. Las neuronas no tienen función de activación, su estado es el valor de la entrada.

Capas ocultas: Forman los niveles intermedios y pueden formar distintas estructuras según el tipo de red que se use. Puede haber una, varias o ninguna.

Capa de salida Es el último nivel y el estado de las neuronas es la salida de la red.

Dada una red con L capas y n_l; $l = l, \ldots, L$ neuronas por capa, la salida x_j^l de una neurona j de la capa l sería:

$$x_j^l = \sigma\left(\sum_{j=1}^{n_l} w_{ij}^l x_k^{l-1} + u_j^l\right) \tag{3.12}$$

donde w_{ij}^l es el peso de la señal que va de la neurona k de la capa $l-1$ a la neurona j de la capa l, x_k^{l-1} es la señal de salida de la neurona k de la capa $l-1$ y u_j^l los valores umbrales de la neurona j de la capa l.

Aunque existen recomendaciones, no hay ningún método que especifique el número de capas o neuronas a emplear, dejándose principalmente a la experiencia y la experimentación.

3.3. Entrenamiento

Durante el entrenamiento de la red, se realiza, de forma iterativa, el ajuste de los pesos de cada neurona w y de los valores umbrales u, hasta que los valores de salida y se correspondan con los esperados y', usando un conjunto de datos de entrenamiento.

Existen tres tipos de entrenamiento:

Entrenamiento supervisado: Además de los ejemplos de entrenamiento se tiene la solución al problema. Se entrena la red para ajustar las salidas obtenidas a las salidas esperadas.

Entrenamiento no supervisado: En este caso, solo se cuenta con los ejemplos de entrenamiento. Es la propia red la que busca patrones y características comunes entre los datos y ajusta sus pesos y umbrales según estos.

Entrenamiento por refuerzo: Además de los ejemplos, se tiene alguna forma de medir la idoneidad de la respuesta de la red, llamada *fitness*. La red se entrena para maximizar su *fitness*.

Hay que tener precaución con el entrenamiento. Adaptar la ANN demasiado bien a los ejemplos puede llevar a un caso de sobreentrenamiento u *overfitting*. Si se produce *overfitting*, la red quedará preparada para responder a unos rasgos

muy específicos que se encuentran en los datos de ejemplo y no será capaz de generalizar para hacer predicciones a partir de datos nuevos, lo cual es su objetivo. Es necesario llegar a una solución de compromiso entre ajuste y generalización.

Dentro del entrenamiento supervisado, el método comúnmente empleado para el entrenamiento de redes neuronales es la *propagación hacia atrás*.

3.3.1. Propagación hacia atras

En inglés, *backpropagation algorithm*, es un método de descenso del gradiente donde el error cometido se propaga hacia atrás desde la salida. El objetivo es minimizar una función coste C modificando los pesos y los umbrales de cada neurona capa a capa. Un ejemplo de función coste:

$$C = \frac{1}{2} \sum_j \|y_j - x_j^L\|^2 \tag{3.13}$$

donde y_j son las salidas esperadas de la red y x_j^L las salidas de las neuronas de la última capa.

En cada iteración los pesos y los umbrales son actualizados sumándoles un factor Δw_{jk}^l y Δu_j^l proporcional a su gradiente.

$$\Delta w_{jk}^l = -\lambda \frac{\partial C}{\partial w_{ij}^l} \tag{3.14}$$

$$\Delta u_j^l = -\lambda \frac{\partial C}{\partial u_j^l} \tag{3.15}$$

donde λ define el parámetro de aprendizaje. Se interpreta como el tamaño del paso en la dirección del gradiente negativo en cada iteración. Puede ser constante o variable si, por ejemplo, se quiere un paso grande en las primeras iteraciones para que la red aprenda rápido y pequeño al final del proceso para mayor precisión.

Usando la regla de la cadena, los gradientes pueden descomponerse de la siguiente manera:

$$\frac{\partial C}{\partial w_{ij}^l} = \frac{\partial C}{\partial x_j^l} \frac{\partial x_j^l}{\partial z_j^l} \frac{\partial z_j^l}{\partial w_{ij}^l} \tag{3.16}$$

$$\frac{\partial C}{\partial u_j^l} = \frac{\partial C}{\partial x_j^l} \frac{\partial x_j^l}{\partial z_j^l} \frac{\partial z_j^l}{\partial u_j^l} \tag{3.17}$$

Los dos primeros términos pueden agruparse en un único parámetro δ_j^l:

$$\delta_j^l = \frac{\partial C}{\partial x_j^l} \frac{\partial x_j^l}{\partial z_j^l} = \frac{\partial C}{\partial x_j^l} f'(z_j^l) \tag{3.18}$$

Y derivando el tercer término se obtiene:

$$\frac{\partial z_j^l}{\partial w_{ij}^l} = \frac{\partial}{\partial w_{ij}^l}\left(\sum_j w_{jk}^l x_k^{l-1} + u_j\right) = x_k^{l-1} \tag{3.19}$$

$$\frac{\partial z_j^l}{\partial u_j^l} = \frac{\partial}{\partial u_j^l}\left(\sum_j w_{jk}^l x_k^{l-1} + u_j\right) = 1 \tag{3.20}$$

Los gradientes quedan de esta forma:

$$\frac{\partial C}{\partial w_{ij}^l} = x_k^{l-1}\delta_j^l \tag{3.21}$$

$$\frac{\partial C}{\partial u_j^l} = \delta_j^l \tag{3.22}$$

El proceso a seguir es el siguiente:

1. Se inicializan los pesos y los umbrales de la red con valores aleatorios.
2. Se hacen pasar los ejemplos de entrenamiento por la red y se comparan los resultados obtenidos con los esperados calculando la función coste.
3. Desde la última capa L hasta la primera se calculan (3.14) y (3.15) y se actualizan los pesos y los umbrales.
4. Se calcula de nuevo la función coste.
5. Si C es menor que cierto valor límite o se supera un número máximo de iteraciones se termina el entrenamiento. Si no, se vuelve al paso 3.

El principal problema de este tipo de entrenamiento es que si la red tiene muchas capas la información del error cometido se irá perdiendo a medida que se retroceda. Las primeras capas apenas sufren cambios en cada iteración y los tiempos aumentan considerablemente.

3.4. Redes Neuronales Profundas

En *machine learning*, las redes suelen ser de pequeño tamaño, superficiales, con unas tres capas incluyendo la de entrada y salida. Es decir, que lo habitual es usar una sola capa oculta, lo cual ofrece buenos resultados para problemas de clasificación o de regresión lineales y no lineales. En el momento en el que se tiene más de una capa oculta se pasa al campo del *deep learning* y nuestra red pasa a llamarse Red Neuronal Profunda o DNN (*Deep Neural Network*).

Una de las características más importantes de este tipo es lo conocido como *jerarquía de características*. A medida que se avanza en la red, cada capa es capaz de identificar características cada vez más complejas, a partir de aquellas

más simples provenientes de la capa anterior. Esto es de gran utilidad a la hora de procesar imágenes, vídeo o audio.

Por ejemplo, suponiendo una red de reconocimiento de imágenes, donde la entrada son los píxeles que las componen. La primeras capas podrían identificar características sencillas tales como bordes, líneas, curvas o zonas de luz o sombra. Las capas siguientes encontrarían otras cada vez más complejas, como ojos, bocas, etc. Y las últimas identificarían rostros humanos completos.

Este tipo de redes permitirían extraer características y clasificar gran cantidad de datos sin etiquetar. Dado que la mayor parte de la información disponible es de este tipo, su aplicación es de interés en muchos campos. En entrenamientos con datos etiquetados, este tipo de redes puede procesar mucha mayor información de entrada que las redes convencionales, obteniendo resultados más precisos.

3.5. TDNN en reconocimento de hablante

El uso de *deep learning* y redes neuronales ha conseguido grandes mejoras en diversos campos. Dentro del campo del análisis de la voz, su mayor empleo ha sido en el reconocimiento del habla o ASR [6].

En estos sistemas, la red recibe como entrada un vector de características extraídas del audio x, los MFCC, aunque usando un número mayor de filtros que en los sistemas habituales. A este tipo de MFCC con mejor calidad los denominaremos *MFCC hires* (*HIgh RESolution*). La salida de la misma es otro vector con las probabilidades a posteriori o *posterior* de que una clase fonética q pertenezca al frame del que se han extraído las características: $p(q|x)$.

Estas clases fonéticas técnicamente son llamada senones. La pronunciación de un fonema viene influenciada por los fonemas anterior y posterior, por lo que son agrupados en grupos de tres, formando lo que se conoce como un trifonema. Cada fonema tiene tres estados. Un algoritmo, comúnmente un árbol de decisión, agrupa los estados similares reduciendo su número. Estas clases que representan uno o varios estados de uno o varios trifonemas son las clases que la red aprende a discriminar.

Un caso especial de DNN es la TDNN. En este tipo de redes, capa recibe como entrada la concatenación de varias salidas de la capa anterior. Es especialmente útil para el reconocimiento de patrones en largas secuencias temporales. Es por ello que se emplea en reconocimiento de voz, donde los sonidos rara vez son de una longitud uniformes y es preciso tener en cuenta la evolución temporal.

A continuación, se expone cómo este tipo de redes y los posterior obtenidos pueden emplearse para mejorar el método de reconocimiento de hablante basado en i-vector.

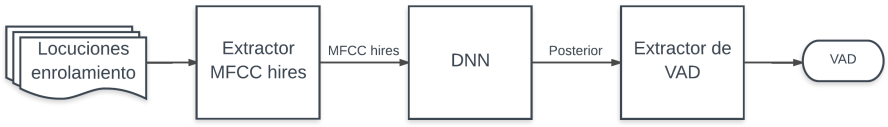


Figura 3.4: Diagrama de flujo de la extracción del VAD a partir de un DNN entrenada para ASR. El nuevo VAD sustituye al calculado por la energía en el diagrama de la Figura 2.1

.

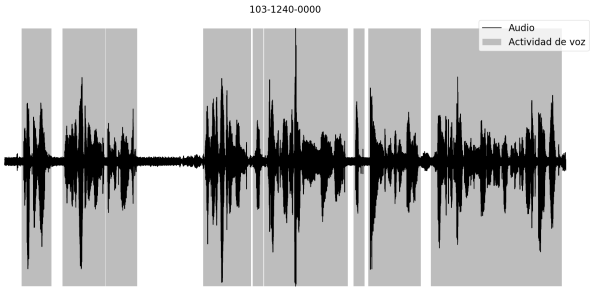


Figura 3.5: Ejemplo de VAD con DNN en el fichero de audio 103-1240-0100 de Librispeech.

3.5.1. VAD

Dado que entre las clases acústicas que la red puede reconocer se encuentran además los de "silencio" y "ruido", una primera aplicación sería identificar cuándo hay actividad de voz y extraer el VAD (Figura 3.4).

Como se ha comentado en el capítulo anterior, este paso es de gran importancia, pues permite desechar partes del audio que no contienen información relevante para centrarnos en aquellos con voz.

Por tanto, a partir de las salidas de red, se clasificarán los *frames* en dos categorías: voz y silencio/ruido; creando un nuevo VAD alternativo al generado a partir de la energía de la señal.

3.5.2. GMM-UBM supervisado

El modelo mezclas gaussianas clasifica las características en regiones acústicas que solo tienen sentido dentro del modelo que se está utilizando. En su lugar, pueden usarse los *posterior* de la red neuronal para que el modelo GMM los clasifique según su contenido fonético. De este modo se crea un GMM-UBM supervisado por la DNN [5, 11].

Un GMM venía definido por el vector de pesos p, el vector de medias μ y la matriz de covarianza Σ como se vio en la ecuaciones (2.2) y (2.3). Usando la DNN, estos valores son inicializados de la siguiente manera:

$$z_k^{(i)} = Pr(k|y_i) \tag{3.23}$$

$$p_k = \sum_{i=1}^{N} z_k^(i) \tag{3.24}$$

$$\mu_k = \frac{1}{p_k} \sum_{i=1}^{N} z_k^i x_i \tag{3.25}$$

$$\Sigma_k = \frac{1}{p_k} z_k^{(i)} (x_i - \mu_k)(x_i - \mu_k)^T \tag{3.26}$$

donde $Pr(k, y_i)$ es la probabilidad de que el fonema k pertenezca a las características y_i extraídas de la red en el frame i, es decir, los *posterior* extraídos de la DNN. x_i es el mismo vector de características del audio usado para el GMM-UBM tradicional.

El GMM-UBM tendrá un número de gaussianas igual al del número de clases acústicas detectadas por la red neuronal.

Los pasos siguientes son iguales que en el método de los i-Vectors que se presentó en el capítulo anterior.

3.5.3. DNN-UBM

En este caso, se emplea la TDNN para crear un UBM que directamente modela contenido acúsitco y para obtener el extractor de i-Vectors, T [5, 11]. Se elimina completamente el GMM-UBM y en su lugar un nuevo modelo universal al que llamaremos DNN-UBM es creado.

En el caso del modelo GMM-UBM, las regiones de partición del espacio acústico tienen un carácter puramente abstracto, en base a características espectrales. Cuando se emplea una red neuronal para ASR de modo que las clases tienen un carácter fonético, la hipótesis es que se realiza una más significativa partición de este espacio ya que las clases ya no tienen un carácter puramente abstracto sino que corresponden a características relativas a la pronunciación de diferentes fonemas.

Son necesarios las características MFCC del audio que se han estado usando el modelo tradicional de GMM-UBM y el GMM supervisado del método anterior para inicializar el modelo.

3.5.4. Red con cuello de botella

Si a la DNN se le añade una capa oculta intermedia la cual tenga un número reducido de neuronas, se fuerza a la red a concentrar la información en un número reducido de características. Al estrechamiento se le conoce como *cuello de botella* o *bottleneck* . Este tipo de redes son utilizadas con frecuencia para crear auto-encoders: se entrena la ANN para que la entrada y la salida sea la misma, así se puede utilizar la salida de las neuronas del cuello de botella como información comprimida, usando una parte de la red como encoder y la otra como decoder.

DNN de este estilo están mostrando buenos resultados en el campo de reconocimiento del habla y de hablante [10, 13]. En este caso, se usan dos redes con cuello de botella. Este tipo de configuración se está mostrando superior a otro tipo de arquitecturas a la de extraer características para el reconocimiento.

La primera red recibe como entrada las características extraídas del audio tras aplicarle la transformada de Fourier, aplicar el banco de frecuencias de Mel y hacerles el logaritmo. Ésta red tiene una capa oculta de cuello de botella cuyas salidas son la entrada de la segunda DNN. De forma intermedia, la salida puede ser apilada y muestreada de nuevo antes de entrar en la segunda red. Esta también cuenta con un cuello de botella cuya salida serán las características a emplear para el reconocimiento de hablante. Ambas redes son entrenadas para la identificación de clases acústicas.

El proceso, a continuación, realiza la identificación de hablante basada en i-vector con estas características. También ha demostrado tener buenos resultados con la combinación de las características de cuello de botella con los MFCC habituales.

Capítulo 4

Representación de resultados

Los distintos métodos planteados ofrecen como resultado una puntuación de lo parecidos que son los hablantes de dos locuciones, bien con la distancia del coseno, bien con PLDA. Se necesita alguna forma de analizar cómo de buenas son estas puntuaciones, si se diferencian bien los casos en los que son el mismo hablantes de los que no y bajo qué condiciones funcionan mejor o peor. En biometría, una forma extendida de hacerlo es mediante curvas *Detection Error Tradeoff* (DET), a partir de las cuales, además de una interpretación gráfica, se puede obtener información numérica relevante sobre el sistema la capacidad de discriminación del sistema, tales como el *Equal Error Rate* (EER) y el valor mínimo de la *Detection Cost Function* (DCF).

4.1. Tipos de errores

Para evaluar el sistema se ofrecen una serie de pruebas o *trials*. En cada *trial* se comparan los i-vectors de uno de los hablantes enrolados de una locución *test*. Junto con estos *trials* se indica si el locutor coincide, (*target*), o es diferente, (*non-target*). Los errores que se pueden dar son los siguientes:

Falso positivo: También llamado falsa alarma. Se reconoce que el test pertenece al hablante enrolado a pesar de no serlo.

Falso negativo: Es lo contrario al falso positivo, se considera el hablante del test no es el enrolado aunque sí lo sea.

La probabilidad de que ocurran falsos positivos y falsos negativos dependerá de en qué valor se sitúe el valor de corte que identifique si es *target* o *non-target*. Con un valor bajo se producirán muchos falsos positivos, lo que puede dar lugar a fallos en seguridad. Por otro lado, un valor alto dará lugar a muchos falsos negativos, haciendo del proceso de identificación algo tedioso. Se suele llegar a

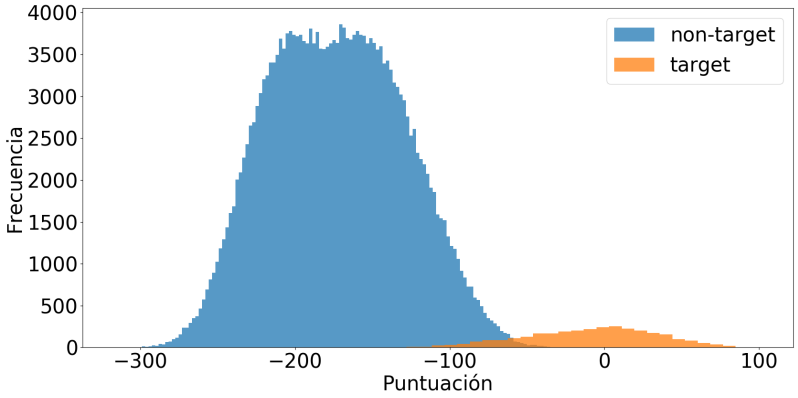


Figura 4.1: Histograma con las puntuciones obtenidas con GMM-UBM e i-vector con 512 componentes gaussianas. No existe una separación del 100 % entre targets y non-targets. Según dónde esté el valor umbral se tendrá una cierta cantidad de falsos negativos y falsos positivos.

una situación de compromiso colocando este punto de corte en un punto adecuado según la aplicación, priorizando seguridad o velocidad. El valor habitual suele estar en torno al 0,1 % de probabilidad de falso positivo.

4.2. Curvas DET

Pueden separarse las puntuaciones obtenidas en los *trials* en *target* y *non-target*. Si se representan sus densidades se observa que siguen distribuciones que se aproximan a la normal (Figura 4.1). Dada la complejidad y las distintas variables que afectan al proceso de identificación de hablante, estas curvas suelen estar solapadas. Esto significa que habrá casos en los que se considere que dos locuciones pertenecientes a hablantes distintos tengan mayor puntuación que otras que pertenezcan al mismo y viceversa.

Si situamos la puntuación, conforme aumenta su valor, la cantidad de non-targets tomados como targets (falso positivo) va disminuyendo, mientras que la cantidad de *target* tomados como non-targets va aumentando (falso negativo). A estos valores normalizados entre 0 y 1 se le llama probabilidad de falso positivo (P_{fp}) y probabilidad de falso negativo (P_{fn}).

En las curvas DET (Detection Error Trade-off) se representan P_{fp} (eje de abcisas) frente a P_{fn} (el eje de ordenadas). Los ejes están en escala de la función cuantil de la distribución normal, o función probit:

$$probit(p) = \sqrt{2} erf^{-1}(2p - 1) \tag{4.1}$$

donde p es la probabilidad, ya sea P_{fp} o P_{fn}, y erf^{-1} es la inversa de la fun-

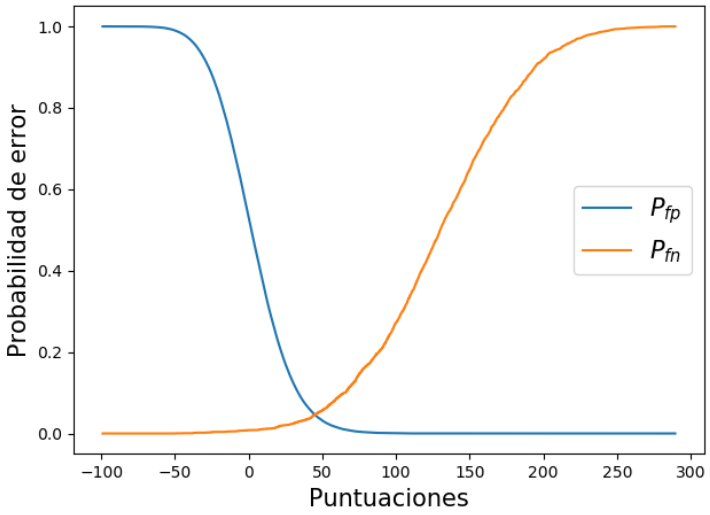


Figura 4.2: Probabilidad de los errores por falso positivo, P_{fp}, y falso negativo, P_{fn}, en función de la puntuación de los resultados mostrado en la Figura 4.1

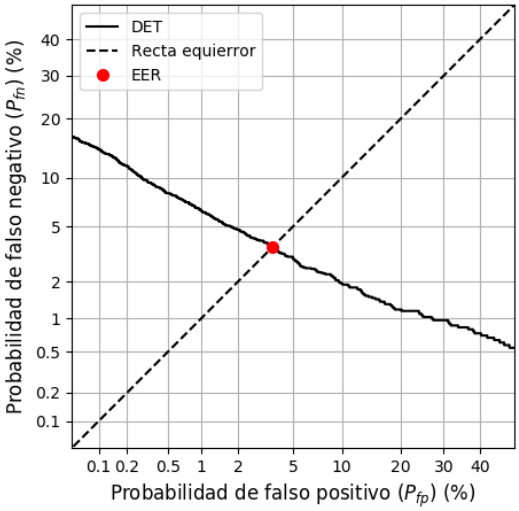


Figura 4.3: Ejemplo de curva DET con su EER representado.

ción de error. Este tipo de ejes tiene la cualidad de que si las puntuaciones de los *targets* y *non-targets* se aproximan a una distribución normal, la curva se aproximará una recta de pendiente $-\sigma_{non}/\sigma_{tar}$, con $\sigma_{(tar,non)}$ la desviación estándar.

Lo que hace que este tipo de curvas sean empleadas en reconocimiento de hablante frente a otras como las ROC, utilizadas de forma tradicional para representar esta información en una escala lineal, es que pequeños cambios en la separación de *targets* y *non-targets* producen una mayor separación de las curvas. De esta forma, las pequeñas mejoras introducidas de un método a otro son posibles de visualizar a simple vista.

4.3. Equal Error Rate

Existen algunas formas de resumir las curvas DET en un valor numérico representativo. Una de ellas es el *Equal Error Rate* o EER. Este es el punto en el que la probabilidad de falso positivo y de falso negativo se hacen iguales ($P_{fp} = P_{fn}$). O, dicho de otra forma, el punto donde la curva se cruza con la diagonal.

El EER es un indicador de la capacidad de discriminación del método empleado sencillo de obtener y fácilmente comprensible. Cuanto menor sea, menores serán los errores producidos y mejor separados estarán los *targets* de los *non-targets*. Sin embargo, no es una medida de calibración. No aporta información sobre donde situar el punto de corte.

Sin embargo, EER no siempre es un valor representativo de lo bueno que es nuestro sistema. Suele interesar ponerse del lado de la seguridad, dando más valor a tener una falsa probabilidad de acetar falsos positivos. Aunque dos sistemas tengan el mismo EER, uno puede ser mejor que otro. Se necesitan otros métodos para estudiar la bondad del sistema, tales como la función coste.

4.4. Detection Cost Function

La función de coste de detección o DCF permite obtener una puntuación de cómo de bueno es el sistema según la aplicación para la que sea necesario y los requerimientos de seguridad que exija.

Una función extendida en biometría de voz es la propuesta por el *National Institute of Standards and Technology* (NIST) de Estados Unidos en sus competiciones:

$$C_{Det}(P_{fn}, P_{fp}) = C_{fn}P_{fn}P_{tar} + C_{fp}P_{fp}(1 - P_{tar}) \tag{4.2}$$

donde C_{fn} es el coste de falso negativo, C_{fp} es el coste de falso positivo y P_{tar} es la probabilidad a priori de que *trial* sea *target*.

Para la del año 2010 propuso los siguientes valores de los parámetros:

Tabla 4.1: Valores para el cálculo de C_{det} según NIST10

C_{fn}	C_{fp}	P_{tar}
1	1	0,001

Previamente, hasta el año 2008, los valores propuestos eran estos:

Tabla 4.2: Valores para el cálculo de C_{det} según NIST08

C_{fn}	C_{fp}	P_{tar}
10	1	0,01

En nuestro caso, la función coste será calculada para ambos condiciones, indicando en cada caso si se trata del empelado en el año 2010 (DCF10) o en el 2008 (DCF08).

A partir de la DCF puede obtenerse el valor mínimo (minDCF10 y minDCF10), el cual representa cuánto de bueno puede llegar a ser el sistema a la hora de discriminar entre *target* y *non-target* dadas unas condiciones específicas de preferencia.

Capítulo 5

Bases de datos y software

En este capítulo se presentan las bases de datos con grabaciones de hablantes usadas para el entrenamiento y las pruebas de reconocimiento. Una vez expuestas, se hablará sobre el software libre Kaldi.

5.1. LibriSpeech

El corpus LibriSpeech [16] es un conjunto de datos de entrenamiento basado en el proyecto LibriVox, consistente en la creación de audiolibros de dominio público a través de lectores voluntarios. La mayoría de los libros proceden de textos del Proyecto Gutemberg, el cual ofrece un catálogo de libros electrónicos gratuitos libres de derechos de autor. El corpus contiene 1000 horas de audio muestreado a 16kHz.

Para la preparación del corpus, en primer lugar se ha hecho un primer reconocimiento del audio usando el decoder *gmm-decode-faster* de Kaldi, entrenado con el dataset VoxForge, y se han adaptado las transcripciones pasándolas a mayúscula, quitando símbolos de puntuación y expandiendo abreviaciones. A continuación, se ha hecho un primer alineamiento con el algoritmo de Smith-Waterman entre el texto reconocido y la transcripción y se ha segmentado el audio en fragmentos de 35 segundos o menos. En un segundo alineamiento, se filtran los resultados del anterior deshaciendose de aquellos que tengan una alta probabilidad de ser inadecuados. Finalmente, se segmentan de nuevo los audios: para datos de entrenamiento, en cualquier silencio con duración mayor que 0.3 segundos; en datos de test, cuando coincidan con el final de una oración.

Los hablantes del corpus son puntuados mediante una *Word Error Rate* (WER) según la semejanza entre el texto reconocido y su transcripción real. Aquellos con un WER bajo son catalogados como “clean”, y con un WER alto, como “other”. De “clean” se seleccionaron 20 hombres y 20 mujeres al azar y se asignaron al conjunto de desarrollo (“dev-clean”), y otros 20 hombres y 20 mujeres al azar para el conjunto de test (“test-clean”). El resto de hablantes se dividieron aleatoriamente en dos conjunto de entrenamiento de 100 y 360 horas

Tabla 5.1: Subconjuntos de datos en LibriSpeech.

Subconjunto	**Horas**	**Minutos por hablante**	**Hablantes femeninas**	**Hablantes masculinos**	**Hablantes totales**
dev-clean	5,4	8	20	20	40
test-clean	5,4	8	20	20	40
dev-other	5,3	10	16	17	33
test-other	5,1	10	17	16	33
train-clean-100	100,6	25	125	126	251
train-clean-360	363,6	25	439	482	921
train-other-500	496,7	30	564	602	1166

("train-clean-100" y "train-clean-360").

Para los datos del subconjunto "other", los hablantes se ordenaron de menor a mayor dificultad según su WER. Los hablantes para desarrollo y test fueron seleccionados al azar entre los del tercer cuartil de la lista ordenada. Con el resto se creó un único conjunto de entrenamiento.

5.2. Speakers in the Wild

El objetivo del *Speakers In The Wild Speaker Recognition Challenge* (SITW SRC) [15] es proveer una base de datos de locuciones de audio que permitan a los desarrolladores probar y mejorar sus sistemas de reconocimiento de hablante en condiciones reales. Incluye audios de enrolamiento y entrenamiento de un solo hablante, así como de multihablante. Esta base de datos nos servirá para comprobar nuestros modelos en condiciones más exigentes que las del corpus LibriSpeech.

Las locuciones de audio han sido extraídas de medios *open-source* donde aparecen hablando un gran número de figuras públicas conocidas, etiquetadas como *Person Of Interest* (POI). El objetivo del reto es identificar a estas POI por medio de un coeficiente de semejanza. Los audios incluyen entrevistas, sesiones de preguntas y respuestas en auditorios así como conversaciones casuales, incluyendo una amplia gama de condiciones acústicas y ambientales. No se ha añadido ningún efecto artificialmente, como puedan ser silencios, ruido, reverberación, etc.

Para la anotación y segmentación del audio se usa vídeo para identificar a la POI. En primer lugar, se identifica un fragmento donde el objetivo aparezca hablando más de 20 segundos. Este fragmento forma el segmento *core*. El audio alrededor del *core* se etiqueta para identificar otros hablantes y se define como *multi*. Por último, *assist* contiene todo el audio de la sesión.

Condiciones de enrolamiento. Consiste en audios de uno o varios hablantes formado por los fragmentos 'core' y 'assist'. No se han usado más de 10.000 hablantes para el enrolamiento. Las condiciones son las siguientes:

- ***Core*:** segmentos de audio de un solo POI. La duración será de entre 6 y 240 segundos.
- ***Assist*:** segmentos de audio de uno o más hablantes y al menos una POI. El tiempo total de discurso de la POI será de al menos 6 segundos hasta varios minutos y la duración total desde 40 segundos hasta 2 horas. Se incluye una anotación de desde donde y hasta cuando habla el objetivo.
- ***AssistClean:*** subconjunto de *assist* con audios en mejores condicione acústicas.

Condiciones de prueba. Los segmentos de audio para las pruebas están grabadas en las mismas condiciones que los de enrolamiento, aunque contienen más variabilidad. Las condiciones de entrenamiento son:

- ***Core:*** Ficheros de audios con un solo hablante. El tiempo de habla estará entre 6 y 240 segundos, aproximadamente.
- ***Multi:*** Ficheros de audio con uno o más hablantes. No se aportan etiquetas ni segmentación de los hablantes como en *Assist* y *Assist-Clean* del enrolamiento. El tiempo de habla total irá de 6 segundos a 10 minutos. Si hay alguna POI, el fichero contendrá al menos 6 segundos de habla del objetivo.

Además, los datos se encuentran separados en dos clases. Una es de desarrollo, destinada según las bases de la competición a la comprobación y mejora de los métodos de reconocimiento usados por los participantes. La otra es de evaluación, sin etiquetas y con POI distintas a las de desarrollo con objetivo de presentar resultados para el concurso. En caso de este proyecto, nos hemos centrado en el apartado de desarrollo.

5.3. Software Kaldi

El software Kaldi [17] se trata de un conjunto de herramientas orientadas al desarrollo de métodos de reconocimiento de voz, incluyendo herramientas útiles para el reconocimiento de hablante. Está escrito en lenguaje C++ y se encuentra bajo Licencia Apache v2.0, lo cual quiere decir que es software libre y gratuito que puede ser usado sin restricciones. Depende de dos librerías externas: *OpenFst* para trabajar con estados finitos una librería de álgebra numérico que puede ser "Basic Linear Algebra Subrutines" (BLAS) o "Linear Algebra PACKage" (LAPACK).

Kaldi provee una serie de códigos flexibles que pueden se combinados entre sí, ofreciendo una gran libertad a la hora de implementar métodos ya existentes o crear nuevos sin necesidad de modificar los códigos principales. Cada código tiene una función específica, por ejemplo, hay ejecutables distintos para acumular estadísticos, sumar acumuladores y actualizar el modelo GMM. Además, las salidas y entradas de estos ejecutables pueden ser enlazarse mediante tuberías para hacer más sencilla la concatenación.

Aunque está principalmente orientado para el uso de formas estándar de reconocimeinto, como son la extracción de MFCC, crear modelos GMM o i-vector; también ofrece la posibilidad de usar métodos experimentales, como redes neuronales. Kaldi ofrece tres tipos de DNN:

nnet1: Es mantenida por Karel Vesely. Puede entrenarse con una GPU, lo que hace que sea fácil de implementar y de modificar.

nnet2: Este tipo de red puede ser entrenada tanto en paralelo por varias CPU o GPU, recomendándose la segunda opción.

nnet3: Aun en desarrollo, pretende ser el siguiente nivel en cuanto a redes neuronales de Kaldi.

Para este proyecto se usará un DNN de tipo dnn2 entrenadas usando una GPU. Precisamente por las condiciones computacionales que se exigen, no es objeto de este proyecto entrenar la red. Esta ha sido proporcionada por Biometric Vox.

A partir de dichos bloques de código se han programado y ejecutado los modelos usando el lenguaje Bash. Adicionalmente se han creado algunos scripts en Python para tareas auxiliares y los datos han sido analizados mediante Octave.

5.4. Computación en la nube y control de versiones

Dada la duración de los experimentos y la potencia de cálculo requerida, se han usado servicios de computación en la nube para llevarlos a cabo. En concreto se ha empleado *Google Cloud Platform*, el cual tiene disponible una versión de gratuita 60 días o hasta agotar un crédito de 300$. Estos servicios permiten crear máquinas virtuales en los servidores la compañía que los oferte, en este caso Google, con las prestaciones de CPU, GPU y memoria que se requieran. Los costes normalmente vienen dados en función del tiempo que se estén usando las máquinas y serán mayores cuanto mejores prestaciones. Para este proyecto se ha usado una máquina Linux con Ubuntu 14.04 con 1 o 2 CPU para tareas sencillas y con 8, el máximo en la versión gratuíta, para los experimentos.

Para la gestión del software se usado Github. Github es un sistema de gestor de versiones tipo Hub que permite alojar los proyectos para que sean accesibles de forma pública, o privada si se tiene una cuenta de pago. Todos códigos empleados se encuentran accesibles en la dirección `https://github.com/rubjmnz93/Librispeech`.

Capítulo 6

Experimentos

En este capítulo se se explicarán los experimentos realizados, con información específica sobre los modelos empleados y cálculos empleados.

6.1. Reconocimiento basado en i-vector

En primer lugar se hará un reconocimiento de hablante basado en la técnica que forma el estado del arte basada en i-vector. Las puntuaciones serán calculadas tanto con distancia del coseno como con PLDA, cuyos resultados serán comparados para comprobar la utilidad de ambos métodos y su ámbito de aplicación. El modelo GMM-UBM se calculará usando **512** y **1024** componentes gaussianas para comprobar cómo afecta este parámetro.

Como conjunto de entrenamiento se usará el corpus de LibriSpeech y se evaluará tanto sobre Librispeech como sobre Speakers in the Wild (SITW). Para Librispeech, los datos de enrolamiento y entrenamiento separados en *clean* y *other* se combinan en uno solo sin hacer distinción entre uno y otro. Igualmente para los tres subconjuntos de entrenamiento, que se considerarán como uno solo de 960 horas, aproximadamente.

En la fase de extracción de características, los MFCC para los tres conjuntos (entrenamiento, enrollment y test) son extraídos usando un tamaño de frame de 25 ms con un solapmiento de 10 ms entre cada uno con un tamaño de 20 coeficientes cepstrales por frame. El VAD se calcula a partir del logaritmo de la energía de Mel, usando como valor umbral 5,5. Todo frame con una energía por debajo de ese valor será considerado vacío y no será tenido en cuenta para el reconocimiento.

El entrenamiento del GMM-UBM se hace en dos pasos. Primero, se entrena un GMM con una matriz de covarianza diagonal. Los datos de entrenamiento son los MFCC calculados, sus Δ y sus $\Delta\Delta$ (60 características por frame), usando una ventana de 3 frames para calcularlos. Para su inicialización, se calcula la medias de una serie de datos de entrenamiento elegidos aleatoriamente con los que se hacen una serie de iteraciones E-M en memoria. A partir de estos resultados, se

Tabla 6.1: Características del GMM-UBM diagonal.

Descripción	Valor
Número de iteraciones del entrenamiento en paralelo	4
Máximo número de frames en memoria para iniciar el modelo	500.000
Número de iteraciones E-M para iniciar el modelo	20
Proporción de gaussianas con los que empezar en la fase de inicio	50 %
Peso mínimo gaussiano a usar permitido en la inicialización del GMM	0.0001
Número de características delta	2
Número de frames de contexto para calcular las delta	3

procede al entrenamiento en paralelo. El resto de características de este GMM-UBM diagonal puede verse en la Tabla 6.1.

El GMM diagonal sirve de punto de partida para el entrenamiento de un GMM con matriz de covarianza completa. Este GMM-UBM completo ofrece una mejor representación de las características de la voz que el diagonal, aunque también es más costoso de calcular computacionalmente. Se entrena haciendo correr varios trabajos en paralelo durante un total de 4 iteraciones del algoritmo E-M.

El siguiente paso es el cálculo del extractor de i-vector, la matriz T, con 5 iteraciones en paralelo del algoritmo E-M. Una vez entrenado, se extraen los i-vector para los audios de entrenamiento, enrolamiento y test. Se considera un valor mínimo de 0.025 por el que los *posteriors* generados por debajo de este valor son recortados. Para el conjunto de enrolamiento, la huella de voz del hablante es calculada como la media de los i-vector de sus extractos de audio. Los i-vector tienen un tamaño de 400 elementos.

Con los i-vector de entrenamiento se entrena el modelo PLDA para el cálculo de puntuaciones. Los i-vector son normalizados para este propósito de forma que su longitud coincida con la raíz cuadrada dimensión, es decir,

$$|\omega| = \sqrt{\omega_1^2 + \omega_2^2 + \omega_2^2 + \cdots + \omega_{400}^2} = \sqrt{400} \tag{6.1}$$

Por último, se calculan las puntuaciones con la distancia del coseno y el modelo PLDA entrenado haciendo una comparación cruzada de todos los hablantes del enrolamiento con todos los audios del test, indicando en cada caso si se trata del mismo hablante o distintos.

6.2. VAD con TDNN

El siguiente paso será comprobar si un VAD basado en el reconocimiento de clases acústicas es más efectivo que uno basado en la energía, usando el mismo método de i-vector. Es decir, qué sistema diferenciará mejor las partes con habla de aquellas en silencio o con ruido.

La DNN ha sido proporciona por Biotric Vox S.L. y ha sido entrenada igualmente con la base de datos LibriSpeech. Los parámetros de entrada para esta

red son unos MFCC de mejor calidad que los usados en el método anterior, los cuales proporcionan 40 coeficientes cepstrales por frame.

La red es inicializada con una única capa oculta. Esta primera red recibe de entrada las características mencionadas, combinándolas con las de los dos frames anteriores y los dos posteriores. En total serían 5 frames o 200 características. Estas entradas son sometidas a una transformación LDA (*Linear Discrimination Analysis*). Mientras que el objetivo habitual del LDA es una reducción de la dimensión, en este caso se emplea para decorrelacionar las entradas manteniendo la dimensión, reduciendo la escala de las dimensiones que no aportan información y mejorando el funcionamiento de la red. La capa oculta tienen una dimensión de 350 y su función de activación es una *PNorm*. La última capa tiene una salida de 4135 elementos correspondientes a las clases acústicas identificadas y usa como función de activación *Softmax*. Después de la inicialización con esta red pequeña se pasa a otra con 5 capas ocultas con 350 elementos de entrada y de salida y función de activación *PNorm*.

La salida de la red es un vector de tamaño 4135 con las probabilidades de 0 a 1 de que se trate de dicho fonema, con una identificación como la de la Tabla 6.2. Para hacer el VAD se ha comparado la suma de las puntuaciones correspondientes a ruido y silencio con la suma del resto de clases acústicas que corresponderían a que realmente se está hablando en dicho frame. Según qué valor es el mayor, se clasifica como silencio con un 0 o como habla con un 1. Con esto se crea un fichero con la identificación del archivo de audio y un vector con la clasificación de cada frame, que se convierte al formato de Kaldi y sustituye al VAD generado con la energía.

Con este nuevo VAD se vuelven a realizar los experimentos anteriores con 512, 1024, 2048 y 4096 funciones gaussianas. Se comparan con los resultados base para comprobar si existe mejora.

6.3. GMM-UBM supervisado y TDNN-UBM

Usando la misma DNN que para extraer el VAD se han hecho pruebas usando un modelo de GMM-UBM supervisado y a partir de este un modelo DNN-UBM como los que se vieron en las secciones 3.5.2 y 3.5.3, respectivamente.

Se usa el VAD extraído de la red neuronal que en pruebas ha demostrado ser mejor que aquel basado en la energía. Los i-vector resultantes tendrán un tamaño de 400 para conservar la compatibilidad con el modelo anterior. Dado que el GMM supervisado tendrá un tamaño de 4135 gaussianas, se comparará su desempeño con los del i-Vector con GMM-UBM de 4096G.

6.4. Características de cuello de botella

Por último se va a hacer una comprobación de las características extraídas de una red de cuello de botella como la descrita en la Sección 3.5.4. Por motivos de tiempo y potencia de cómputo, solo se han usado como entrenamiento las

Tabla 6.2: Algunas clases acústicas extraídas de la DNN. Aquellos con NOI representan ruido (noise), los SIL son de silencio. El resto son distintos sonidos. El número de la derecha indica la salida de la red.

Fonema	Neurona de salida
NOI	1
NOI_B	2
NOI_E	3
NOI_I	4
NOI_S	5
SIL	6
SIL_B	7
SIL_E	8
SIL_I	9
SIL_S	10
SPN	11
SPN_B	12
SPN_E	13
SPN_I	14
SPN_S	15
⋮	⋮

del conjunto "train-clean-100" de LibriSpeech, combinadas con los coeficientes MFCC y sus Δ y $\Delta\Delta$ ((BN+MFCC)+Δ+$\Delta\Delta$). La evaluación también se hará sobre LibriSpeech y será comparado con el método de los i-vector habitual. El VAD será el extraído de la DNN usado en los casos anteriores.

Capítulo 7

Resultados

En cada experimento se generan las puntuaciones según la distancia del coseno (CS, *Cosine Scoring*) y PLDA. A partir de ellas se analiza el comportamiento del método gráficamente usando las curvas DET y numéricamente con EER, minDCF08 y minDCF10, según lo expuesto en el Capítulo 4.

7.1. Método de I-vector

Al comparar las puntuaciones con CS y PLDA obtenidas con LibriSpeech con las curvas DET (Figura 7.1), la distancia del coseno a pesar de ser un método más sencillo ofrece mejores puntuaciones para la zona de interés de baja P_{fp}, aunque luego sea superado por el PLDA. Numéricamente 7.1), el EER con PLDA es menos, pero $minDCF08$ y $minDCF08$ son menores para CS. Los resultados mejoran al aumentar el número de funciones gaussianas en el GMM-UBM.

Con las locuciones de SITW, más complejas que las de LibriSpeech, es cuando el método de PLDA muestra su verdadero potencial, superando ampliamente al CS (Figura 7.2, Tabla 7.2), con la excepción de $minDCF10$. Al aumentar el número de funciones gaussianas, el EER mejora y el $minDCF08$ con PLDA, pero $minDCF10$ empeora así como $minDCF08$ con CS.

Tabla 7.1: Equal Error Rate y valor mínimo de la función coste para la base de datos LibriSpeech con el método de los i-vector.

		EER(%)	**minDCF08x10^2**	**minDCF10x10^4**
512G	CS	3,621	1,257	2,986
	PLDA	3,300	1,497	3,378
1024G	CS	3,332	**1,244**	**2,837**
	PLDA	**3,107**	1,435	2,893

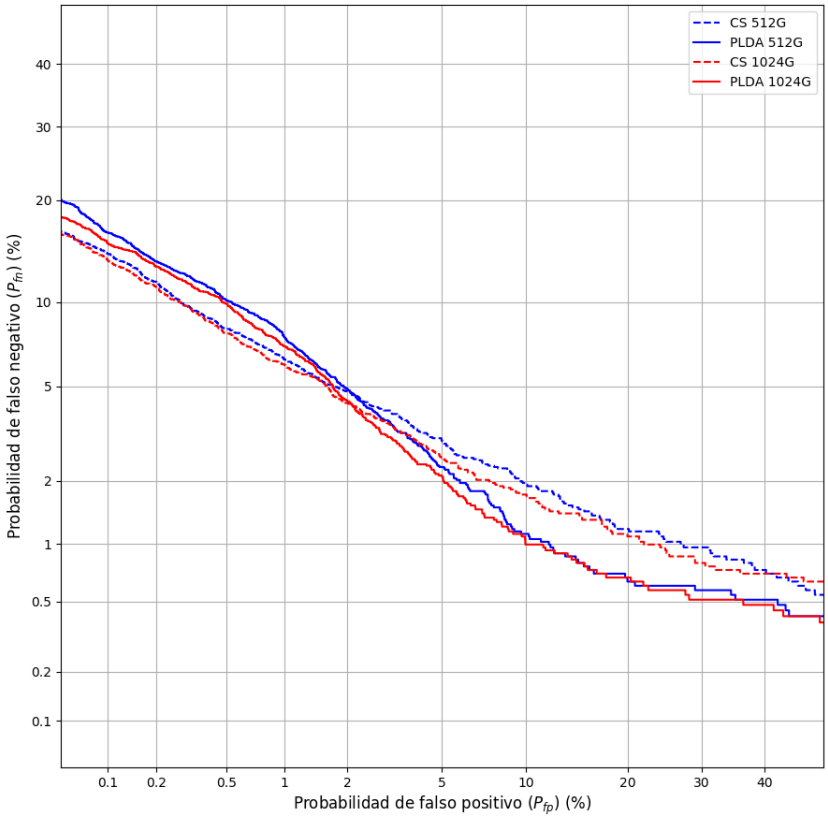


Figura 7.1: Curvas DET para la base de datos LibriSpeech con el método de los i-vector.

Tabla 7.2: Equal Error Rate y valor mínimo de la función coste para la base de datos LibriSpeech con el método de los i-vector.

		EER(%)	**minDCF08x10^2**	**minDCF10x10^4**
512G	CS	18,743	8,263	**9,981**
	PLDA	15,890	7,823	**9,981**
1024G	CS	18,600	8,316	9,985
	PLDA	**15,664**	**7,718**	9,988

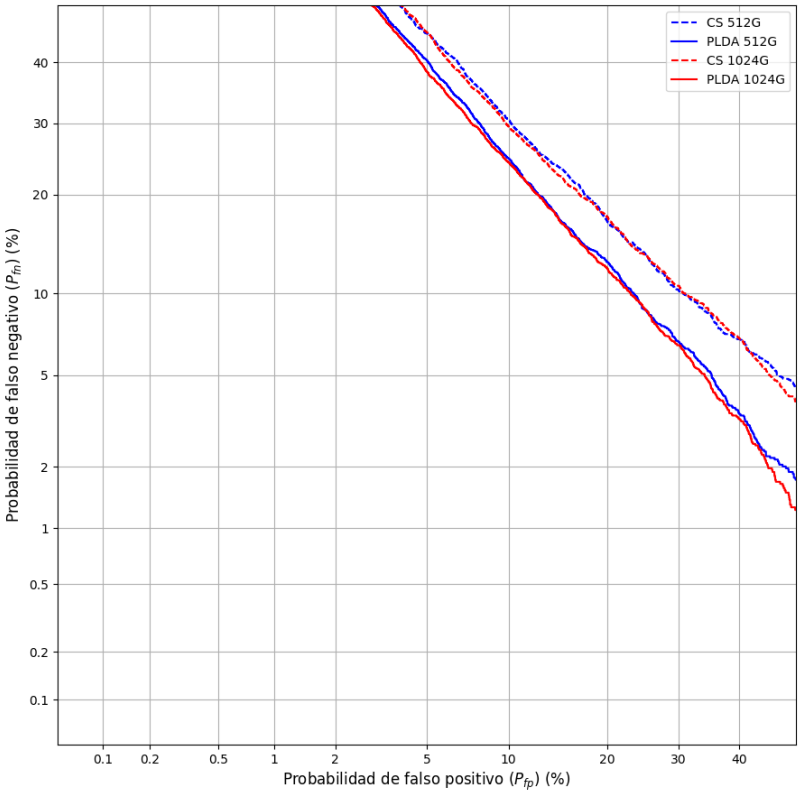


Figura 7.2: Curvas DET para la base de datos Speakers In The Wild con el método de los i-vector.

Tabla 7.3: Equal Error Rate y valor mínimo de la función coste para la base de datos LibriSpeech con el método de los i-vector extrayendo el VAD de la DNN.

		EER(%)	**minDCF08x10^2**	**minDCF10x10^4**
512G	CS	3,621	1,292	2,901
	PLDA	3,300	1,279	2,693
1024G	CS	3,460	1,230	2,812
	PLDA	3,076	1,225	2,579
2048G	CS	3,372	1,154	2,660
	PLDA	3,012	1,178	2,432
4096G	CS	3,108	1,137	2,667
	PLDA	**2,727**	**1,130**	**2,389**

7.2. VAD con TDNN

Al sustituir el VAD basado en la energía por el VAD basado en las clases acústicas extraídas por la red neuronal, las curvas DET con Librispeech (Figuras 7.3 y 7.4) las puntuaciones obtenidas con PLDA superan a las de CS, incluso en la parte de baja P_{fp}, que no varían demasiado de un sistema a otro. El EER apenas cambia (Tabla 7.3) y $minDCF08$ y $minDCF10$ tampoco sufren cambios importantes para CS. Es en la función coste de PLDA donde la mejora es palpable, superando esta vez a la de CS y mejorando respecto a la referencia.

Con SITW la mejora aportada por el nuevo VAD es más importante todavía, tanto para PLDA como CS, especialmente en la zona más sensible de baja P_{fp} (Figuras 7.5 y 7.6). Todos los parámetros mejoran respecto a la referencia (Tabla 7.4): el EER disminuye alrededor de un 1 %, minDCF08 alrededor de 1x10^2 puntos y minDCF10 sobre los 0,05x10^4 puntos.

Para terminar esta serie de experimentos, en la Figura 7.7 y en la Tabla 7.4 se comparan las curvas DET y los resultados numéricos obtenidos 512, 1024, 2048 y 4096 funciones gaussianas del GMM-UBM para LibriSpeech. En 7.8 y 7.4 se hace lo mismo con SITW. Por lo general, al aumentar el número mejoran los resultados, aunque también la complejidad y el tiempo de cálculo. Esta mejora es más visible con LibriSpeech.

7.3. GMM-UBM Supervisado y TDNN-UBM

Para LibriSpeech, las curvas DET (Figura 7.9) son peores que la referencia de 4096G para baja P_{fa} para luego ofrecer mejores resultados para alta P_{fa}. En la Tabla 7.3, el mejor EER para distancia del coseno se consigue con el GMM-UBM supervisado y para PLDA con DNN-UBM. Los mejores minDCF08 y minDCF10 se obtienen con las 4096G, empeorando con los otros dos métodos.

Para SITW, a simple vista se ve la superioridad del DNN-UBM respecto a los otros dos métodos, mejorando también el GMM-UBM supervisado al habitual para PLDA, en las curvas DET (Figura 7.10). En los resultados numéricos

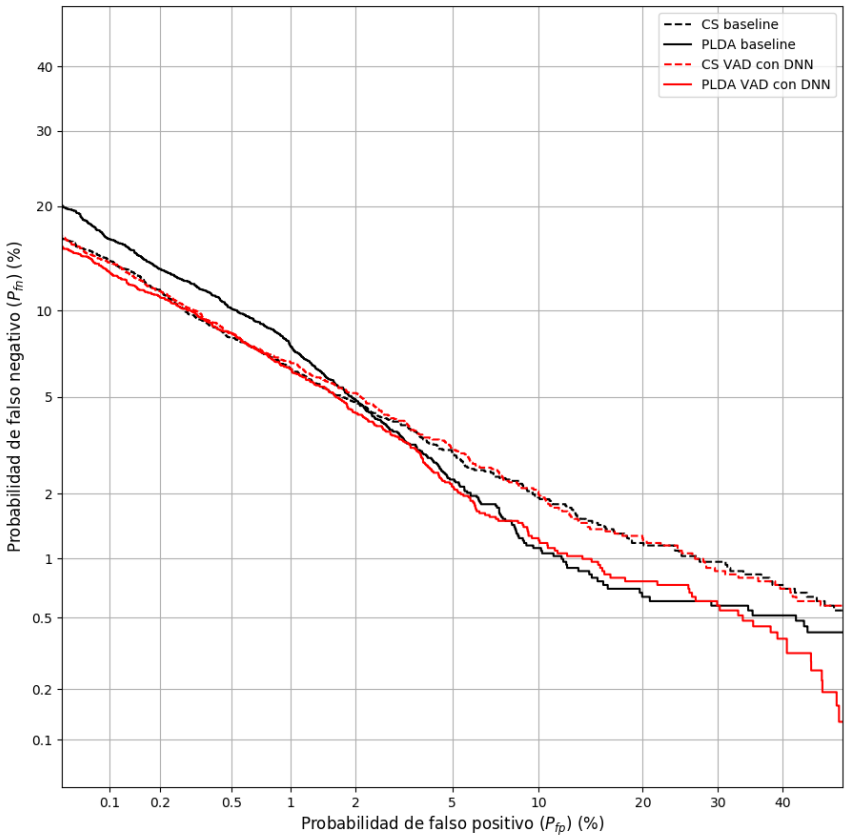


Figura 7.3: Curvas DET para la base de datos LibriSpeech comparando el VAD basado en la energía (*baseline*, en negro)con VAD extraído de la red neuronal para la detección de clases acústicas (rojo) para 512 funciones gaussianas en el GMM-UBM.

Tabla 7.4: Equal Error Rate y valor mínimo de la función coste para la base de datos SITW con el método de los i-vector extrayendo el VAD de la TDNN.

		EER(%)	**minDCF08x10^2**	**minDCF10x10^4**
512G	CS	17,392	7,269	9,93
	PLDA	15,078	7,132	9,93
1024G	CS	16,698	7,181	9,93
	PLDA	**14,622**	7,001	9,93
2048G	CS	16,790	7,226	9,90
	PLDA	14,737	7,040	**9,87**
4096G	CS	16,242	7,126	9,88
	PLDA	14,660	**6,970**	9,93

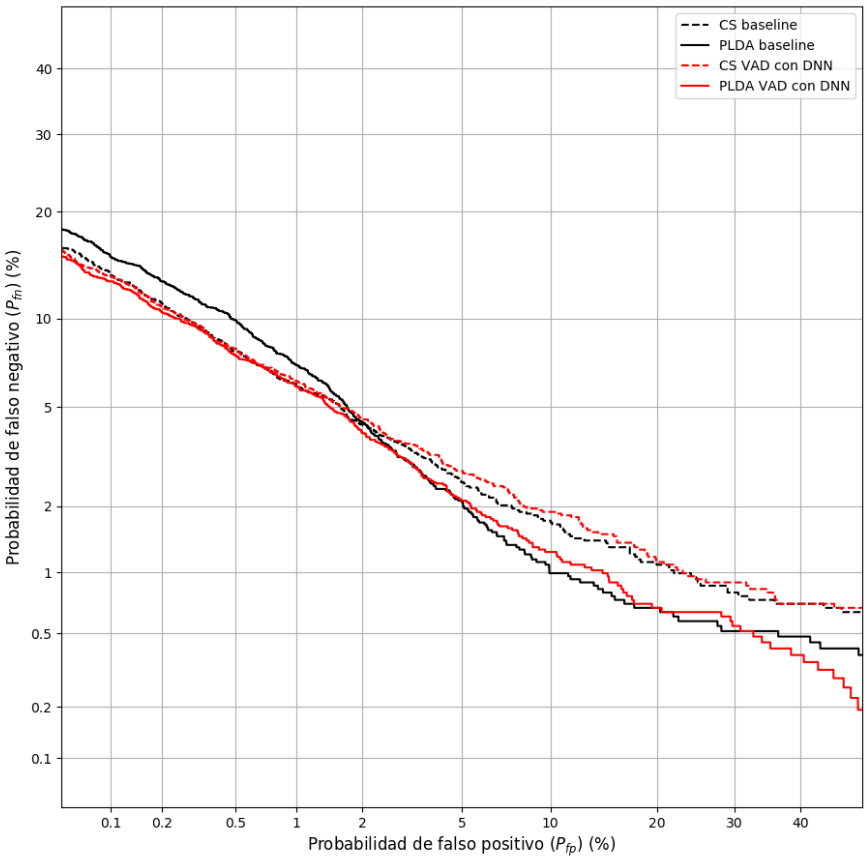


Figura 7.4: Curvas DET para la base de datos LibriSpeech comparando el VAD basado en la energía (*baseline*, en negro)con VAD extraído de la red neuronal para la detección de clases acústicas (rojo) para 1024 funciones gaussianas en el GMM-UBM.

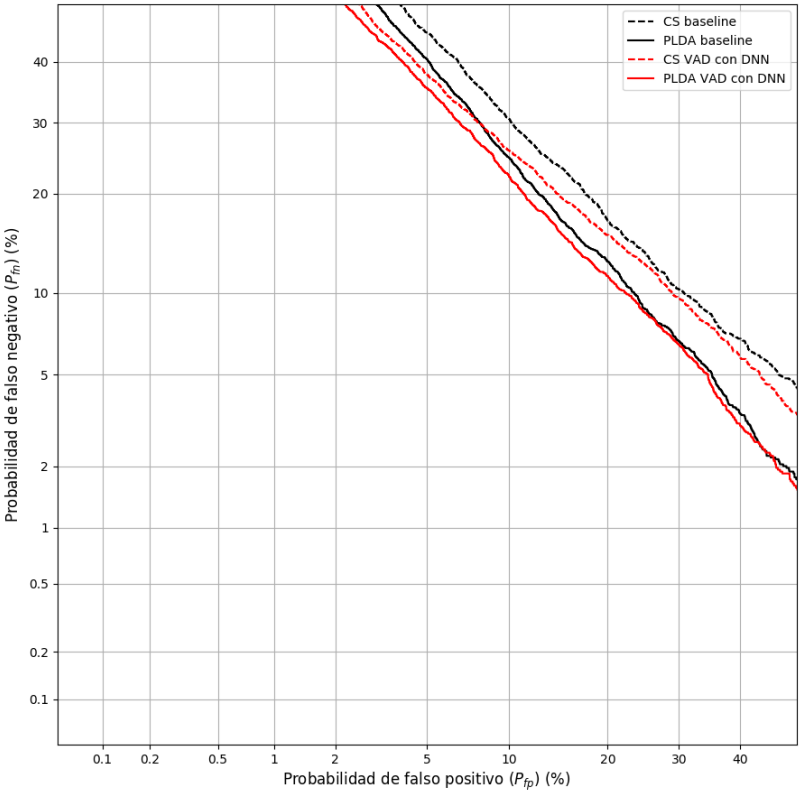


Figura 7.5: Curvas DET para la base de datos Speakers in the Wild comparando el VAD basado en la energía (*baseline*, en negro)con VAD extraído de la red neuronal para la detección de clases acústicas (rojo) para 512 funciones gaussianas en el GMM-UBM.

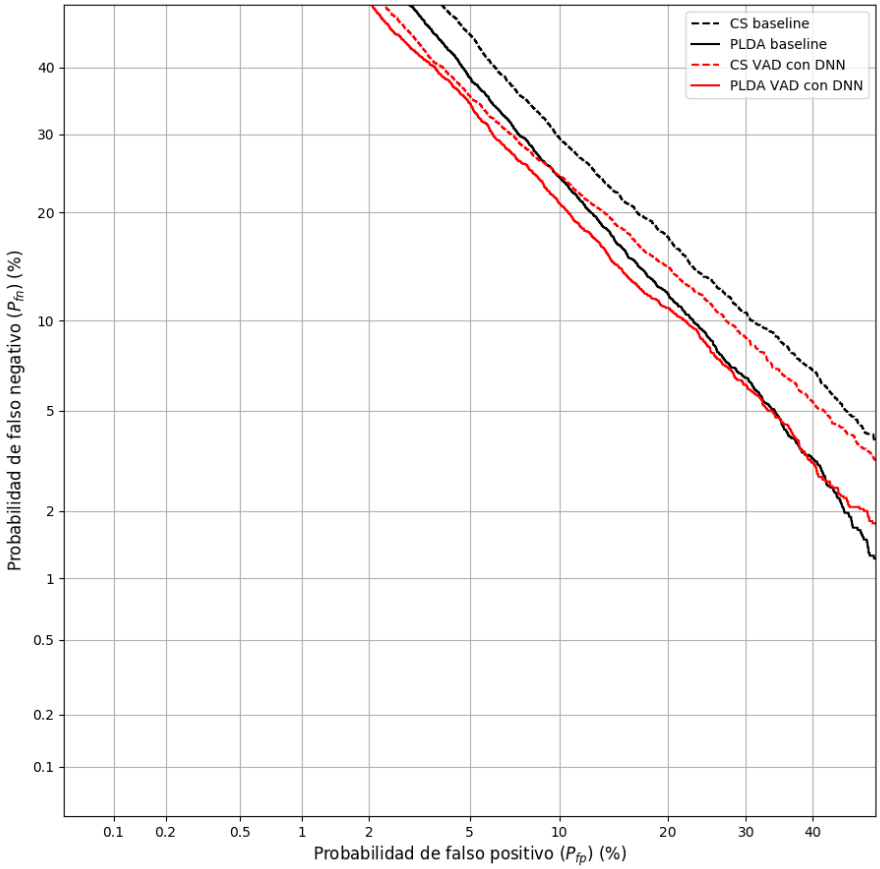


Figura 7.6: Curvas DET para la base de datos Speakers in the Wild comparando el VAD basado en la energía (*baseline*, en negro)con VAD extraído de la red neuronal para la detección de clases acústicas (rojo) para 1024 funciones gaussianas en el GMM-UBM.

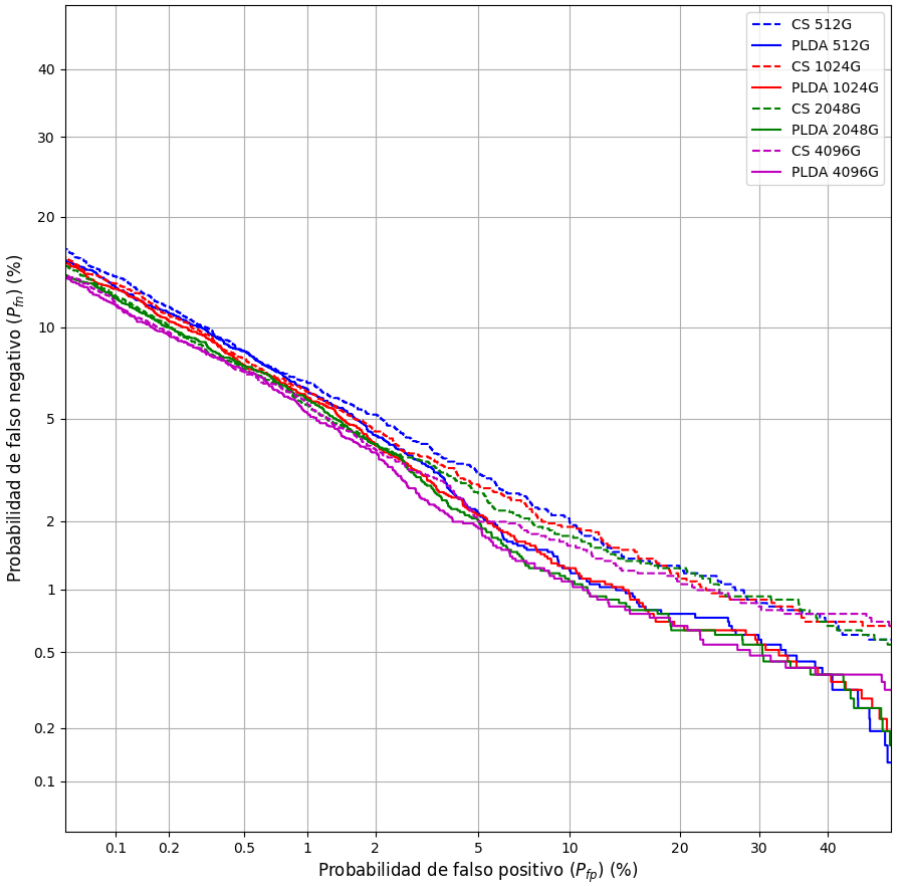


Figura 7.7: Curvas DET para la base de datos LibriSpeech con el método de los i-vector extrayendo el VAD con la DNN.

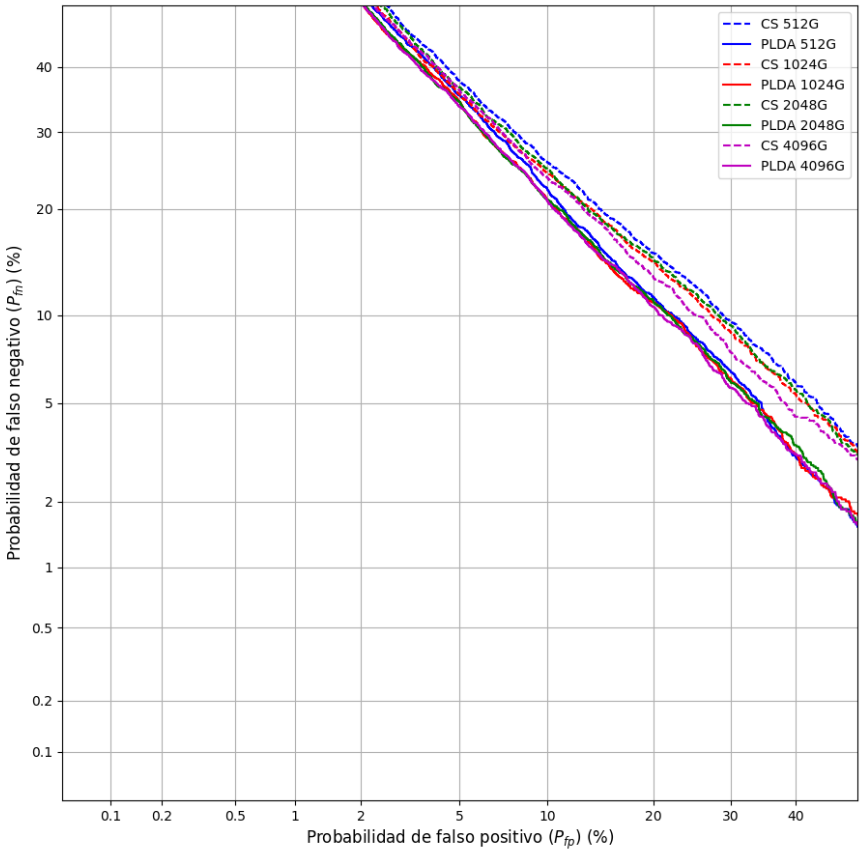


Figura 7.8: Curvas DET para la base de datos SITW con el método de los i-vector extrayendo el VAD con la DNN.

extraídos de las curvas (Tabla 7.6), el EER mejora aproximadamente 0,2 % con el GMM-UBM supervisado y cerca de un 1 % con el DNN-UBM. Usando distancia del coseno como puntuación, minDCF08 y minDCF10 empeora con los nuevos métodos; mientra que usando PLDA el mejor minDCF08 se consigue con DNN-UBM con una mejora de $0,5 \cdot 10^{-2}$ puntos y el mejor minDCF10 con GMM-UBM supervisado con $0,02 \cdot 10^{-4}$ puntos menos.

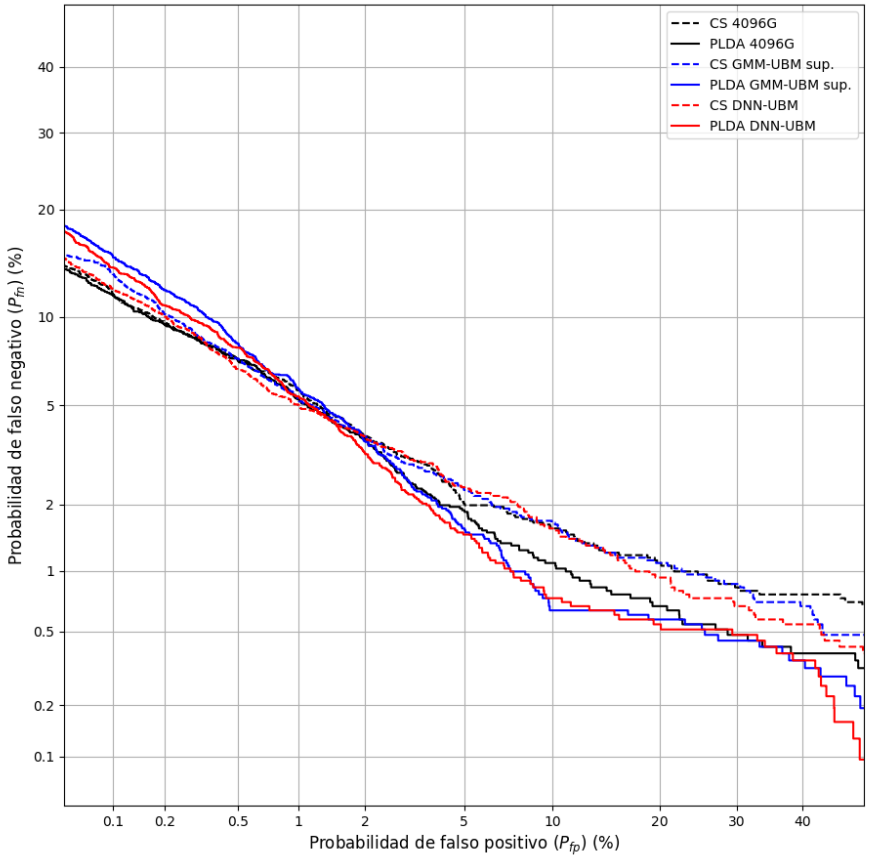


Figura 7.9: Curvas DET para la base de datos LibriSpeech usando 4096 gaussianas GMM-UBM de i-vector habitual, GMM-UBM supervisado y DNN-UBM. Puntuaciones calculadas con distancia del coseno (CS) y PLDA.

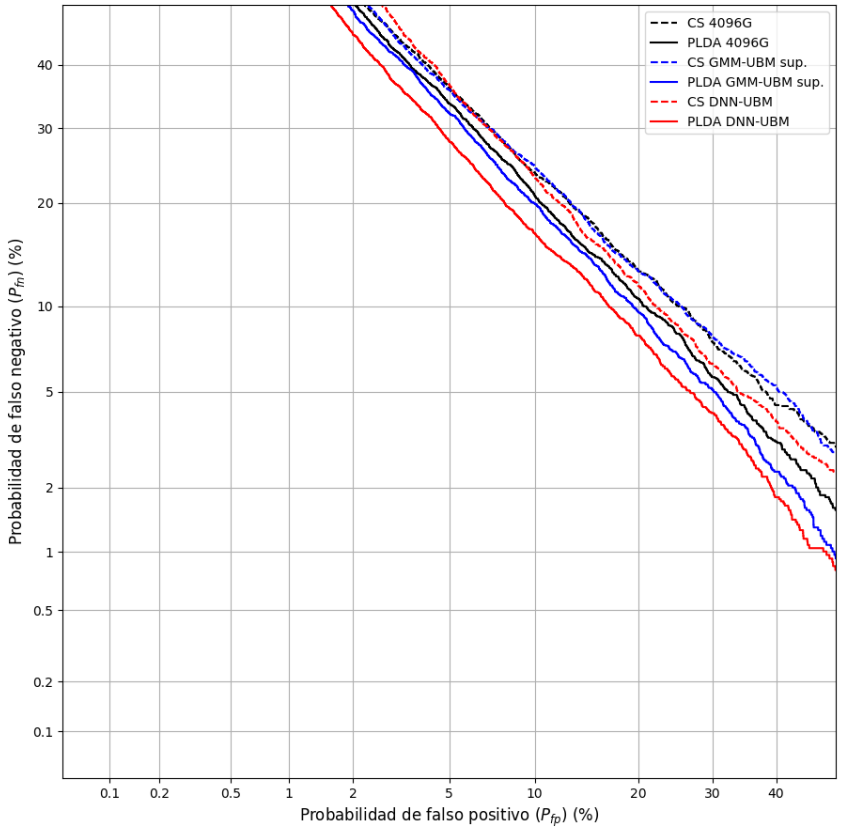


Figura 7.10: Curvas DET para la base de datos SITW usando 4096 gaussianas GMM-UBM de i-vector habitual, GMM-UBM supervisado y DNN-UBM. Puntuaciones calculadas con distancia del coseno (CS) y PLDA.

Tabla 7.5: Equal Error Rate y valor mínimo de la función coste para la base de datos LibriSpeech usando 4096 gaussianas GMM-UBM de i-vector habitual, GMM-UBM supervisado y DNN-UBM. Puntuaciones calculadas con distancia del coseno (CS) y PLDA.

		EER(%)	**minDCF08x10^2**	**minDCFx10^4**
4096G	CS	3,108	1,137	2,667
	PLDA	2,727	**1,130**	**2,389**
GMM-UBM sup.	CS	2,980	1,163	2,844
	PLDA	2,817	1,302	3,801
DNN-UBM	CS	3,082	1,138	2,776
	PLDA	**2,623**	1,252	2,983

Tabla 7.6: Equal Error Rate y valor mínimo de la función coste para la base de datos SITW usando 4096 gaussianas GMM-UBM de i-vector habitual, GMM-UBM supervisado y DNN-UBM. Puntuaciones calculadas con distancia del coseno (CS) y PLDA.

		EER(%)	**minDCF08x10^2**	**minDCF10x10^4**
4096G	CS	16,242	7,126	**9,884**
	PLDA	14,660	6,971	9,927
GMM-UBM sup.	CS	16,053	7,141	9,895
	PLDA	14,429	6,819	9,907
DNN-UBM	CS	15,432	7,608	9,977
	PLDA	**13,229**	**6,474**	9,972

7.4. Características de cuello de botella

En la Figura 7.11 se observa que el uso de las características de cuello ofrece peores resultados que el sistema de referencia, excepto para CS con alta P_{fp}. Hay que tener en cuenta que los datos usados durante este experimento han sido reducidos en comparación con los anteriores. En la Tabla 7.7 se puede ver numéricamente como este sistema empeora respecto a la referencia.

Tabla 7.7: EER, minDCF08 y minDCF10)para la base de datos LibriSpeech usando las características de cuello de botella junto a MFCC. Como referencia se ha usado solo los MFCC.

		EER(%)	**minDCF08x10^2**	**minDCF10x10^4**
Referencia	CS	4,582	**1,813**	**4,406**
	PLDA	6,634	2,341	4,454
Cuello de botella	CS	**4,425**	2,225	8,808
	PLDA	6,761	2,453	4,801

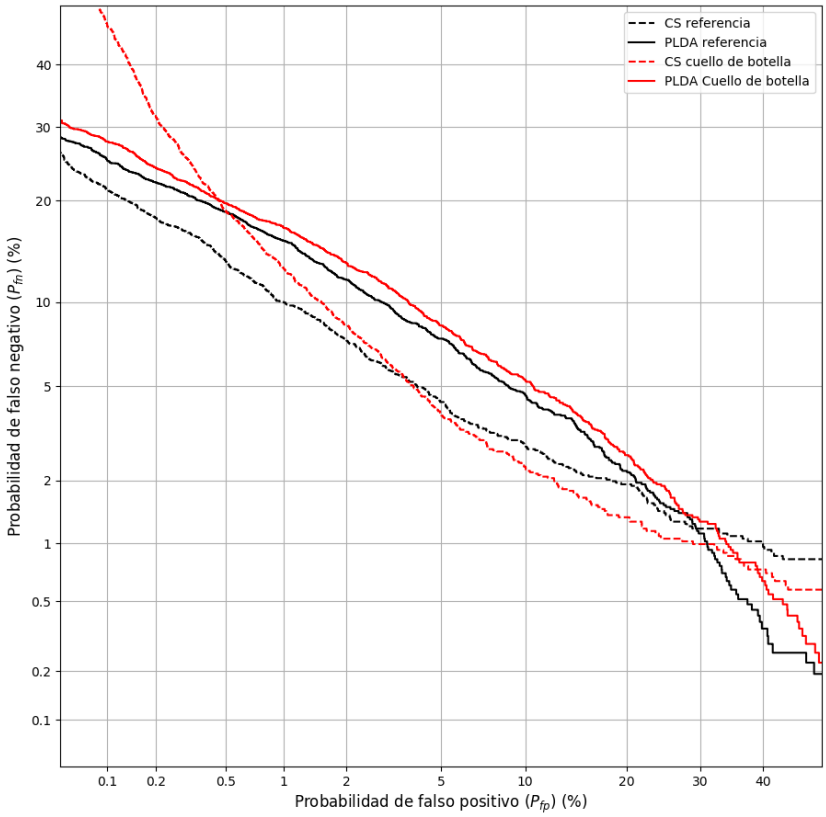


Figura 7.11: Curvas DET para las características de cuello de botella combinadas con MFCC, Δ y $\Delta\Delta$. GMM-UBM con 512G y puntuaciones calculadas con distancia del coseno (CS) y PLDA.

Capítulo 8

Conclusiones

En este trabajo hemos comprobado los resultados de aplicar un método de reconocimiento de hablante ampliamente usado como es el de i-vector y las posibilidades de mejorarlo a través de una red neuronal especializada en otra tarea como es la identificación de clases acústicas. Se pueden extraer las siguientes conclusiones:

- En el método de i-vector tradicional, las puntuaciones basadas en la distancia del coseno, que solo tienen en cuenta el ángulo que forman los vectores, ofrecen mejores resultados para casos sencillos, con locuciones con poco ruido y fácilmente comprensibles, con la ventaja de ser rápido y sencillo. En el caso de locuciones complejas, más próximas a casos reales, un método más complejo como PLDA mejora notablemente los resultados.
- Al aumentar el número de funciones gaussianas que componen el sistema GMM-UBM, mejoran los resultados, pero también aumenta la complejidad y el tiempo de cálculo.
- El VAD extraído a partir del contenido fonético detectado por la DNN mejora los resultados al sustituir al VAD basado en la energía. Especialmente en locuciones complejas.
- Para locuciones sencillas, emplear GMM-UBM supervisado o DNN-UBM no tiene por qué aportar mejoras significativas, mientras que sí lo hace en locuciones complejas.

En resumen, para identificar a un hablante mediante locuciones grabadas en condiciones favorables para ser fácilmente entendibles, métodos sencillos son perfectamente capaces de ofrecer resultados competitivos. Sin embargo, si se emplean locuciones obtenidas en un ambiente menos ideal, el aumentar la complejidad del sistema repercute en una mejora en la identificación.

Algunas vías de investigación para futuros proyectos basados en el presente podrían ser:

- Emplear distintos parámetros para la TDNN para optimizar los resultados o redes convolucionales, más orientadas al tratamiento de imágenes pero que pueden aportar buenos resultados.
- Evaluar el sistema empleando las características extraídas de una red con cuello de botella con el conjunto de entrenamiento completo de LibriSpeech y evaluar sobre SITW. Intentar nuevas configuraciones de la red para intentar mejorarlo.
- Aplicación práctica de los métodos presentados en este proyecto para desarrollar en un programa de reconocimiento de hablante.

Bibliografía

[1] David Bonomo Laynez. Sistemas de verifición automática de locutor, 2012.

[2] William M Campbell, Douglas E Sturim, Douglas A Reynolds, and Alex Solomonoff. Svm based speaker verification using a gmm supervector kernel and nap variability compensation. In *Acoustics, Speech and Signal Processing, 2006. ICASSP 2006 Proceedings. 2006 IEEE International Conference on*, volume 1, pages I–I. IEEE, 2006.

[3] Steven B. Davis and Paul Mermelstein. Comparison of parametric representation for monosyllabic word recognition in continuously spoken sentences. *IEEE Transactions on Acoustics, Speech and Signal Processing*, 28(4):357–366, 1980.

[4] Fernando Manuel Espinoza Cuadros et al. Identificación de hablantes a partir de trayectorias temporales en unidades lingüísticas sobre grandes bases de datos. B.S. thesis, 2012.

[5] Luciana Ferrer, Yun Lei, Mitchell McLaren, and Nicolas Scheffer. Study of senone-based deep neural network approaches for spoken language recognition. *IEEE/ACM Transactions on Audio, Speech and Language Processing (TASLP)*, 24(1):105–116, 2016.

[6] Geoffrey Hinton, Li Deng, Dong Yu, George E Dahl, Abdel-rahman Mohamed, Navdeep Jaitly, Andrew Senior, Vincent Vanhoucke, Patrick Nguyen, Tara N Sainath, et al. Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups. *IEEE Signal Processing Magazine*, 29(6):82–97, 2012.

[7] Marijn Anthonius Henricus Huijbregts. *Segmentation, diarization and speech transcription: surprise data unraveled.* PhD thesis, Centre for Telematics and Information Technology University of Twente, 2008.

[8] Pedro Iasasi Viñuela and Inés M. Galván León. *Redes neuronlaes Arficiales. Un enfoque práctico.* Pearson educación, S.A., 2004.

[9] Ye Jiang, Kong Aik Lee, and Longbiao Wang. Plda in the i-supervector space for text-independent speaker verification. *EURASIP Journal on Audio, Speech, and Music Processing*, 2014(1):29, Jul 2014.

[10] Martin Karafiát, Frantisek Grézl, Karel Veselý, Mirko Hannemann, and year=2014 Igor Szöke and Jan Cernocký, booktitle=INTERSPEECH. But 2014 babel system: analysis of adaptation in nn based systems.

[11] Yun Lei, Nicolas Scheffer, Luciana Ferrer, and Mitchell McLaren. A novel scheme for speaker recognition using a phonetically-aware deep neural network. In *Acoustics, Speech and Signal Processing (ICASSP), 2014 IEEE International Conference on*, pages 1695–1699. IEEE, 2014.

[12] Beth Logan et al. Mel frequency cepstral coefficients for music modeling. In *ISMIR*, 2000.

[13] Pavel Matějka, Ondřej Glembek, Ondřej Novotnỳ, Oldřich Plchot, František Grézl, Lukáš Burget, and Jan Honza Cernockỳ. Analysis of dnn approaches to speaker identification. In *Acoustics, Speech and Signal Processing (ICASSP), 2016 IEEE International Conference on*, pages 5100–5104. IEEE, 2016.

[14] P. Matějka, O. Glembek, F. Castaldo, M. J. Alam, O. Plchot, P. Kenny, L. Burget, and J. Černocky. Full-covariance ubm and heavy-tailed plda in i-vector speaker verification. In *2011 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 4828–4831, May 2011.

[15] Mitchell McLaren, Aaron Lawson, Luciana Ferrer, Diego Castan, and Martin Graciarena. The speakers in the wild speaker recognition challenge plan, 2015.

[16] Vassil Panayotov, Guoguo Chen, Daniel Povey, and Sanjeev Khudanpur. Librispeech: an asr corpus based on public domain audio books. In *Acoustics, Speech and Signal Processing (ICASSP), 2015 IEEE International Conference on*, pages 5206–5210. IEEE, 2015.

[17] Daniel Povey, Arnab Ghoshal, Gilles Boulianne, Lukas Burget, Ondrej Glembek, Nagendra Goel, Mirko Hannemann, Petr Motlicek, Yanmin Qian, Petr Schwarz, Jan Silovsky, Georg Stemmer, and Karel Vesely. The kaldi speech recognition toolkit. In *IEEE 2011 Workshop on Automatic Speech Recognition and Understanding*. IEEE Signal Processing Society, December 2011. IEEE Catalog No.: CFP11SRW-USB.

[18] Simon JD Prince and James H Elder. Probabilistic linear discriminant analysis for inferences about identity. In *Computer Vision, 2007. ICCV 2007. IEEE 11th International Conference on*, pages 1–8. IEEE, 2007.

[19] Douglas A Reynolds, Thomas F Quatieri, and Robert B Dunn. Speaker verification using adapted gaussian mixture models. *Digital signal processing*, 10(1-3):19–41, 2000.

[20] Douglas A Reynolds and Richard C Rose. Robust text-independent speaker identification using gaussian mixture speaker models. *IEEE transactions on speech and audio processing*, 3(1):72–83, 1995.