UNIVERSIDAD AUTÓNOMA DE MADRID
ESCUELA POLITÉCNICA SUPERIOR

TRABAJO FIN DE MÁSTER

Combinación de DNN y audio fingerprinting para detección de ataques de reproducción acústica multidispositivo de passwords habladas

Máster Universitario en Ingeniería de Telecomunicación

Autor: Beltrán Labrador Serrano
Tutor: Joaquín González Rodríguez

FECHA: FEBRERO 2019

COMBINACIÓN DE DNN Y AUDIO FINGERPRINTING PARA DETECCIÓN DE ATAQUES DE REPRODUCCIÓN ACÚSTICA MULTIDISPOSITIVO DE PASSWORDS HABLADAS

AUTOR: Beltrán Labrador Serrano
DIRECTOR: Joaquín González Rodríguez

AUDIAS
Dpto. de Tecnología Electrónica y de las Comunicaciones
Escuela Politécnica Superior
Universidad Autónoma de Madrid
FEBRERO 2019

Resumen

Resumen

En este Trabajo de Fin de Máster, se ha usado una combinación de sistemas DNN y *audio fingerprinting* para la detección de ataques de reproducción acústica multidispositivo de *passwords* habladas, con el fin de investigar la capacidad de detección de estos ataques, uno de los problemas de seguridad más graves de los sistemas de verificación de locutor.

Para ello se ha implementado un sistema de *audio fingerprinting* a partir de un prototipo para la detección de anuncios en programas *broadcast*; y se han desarrollado varios sistemas basados en redes neuronales profundas, para reconocer patrones acústicos en los audios de ataques de reproducción provocados por los diferentes dispositivos de grabación y reproducción que pueden ser utilizados en estos ataques.

El carácter multidispositivo de este problema, conlleva que los audios procedentes de estos ataques de reproducción puedan tener diferentes calidades, dependiendo de la fidelidad del dispositivo con el que se realice el ataque y las condiciones acústicas de grabación y reproducción. Los sistemas de *audio fingerprinting* y de reconocimiento de patrones, funcionan con diferente rendimiento según la calidad de estos audios, siendo los sistemas acústicos capaces de detectar mejor los ataques cuanto peor es la calidad, al reconocer los artefactos acústicos de diferente tipo provocados por la reproducción y grabación, que son más acuciados cuanto menor es la fidelidad del dispositivo. El sistema de *audio fingerprinting*, sin embargo, tendrá un mejor comportamiento cuanto más parecidos sean el audio del ataque y el audio original.

La complementariedad de estas estrategias ha sido probada en este trabajo, realizando una combinación de los sistemas, mejorando con la fusión el rendimiento de cualquiera de los sistemas individuales.

Palabras Clave

Redes neuronales profundas, huella de audio, reconocimiento de patrones, ASVSpoof2017, fusión de sistemas.

Abstract

In this final Master in Science Thesis, a combination of DNN and audio fingerprinting has been used, to be able to detect acoustic password replay attacks, on a multidevice environment, to assess the detection capabilities of this kind of attack, a major security problem on automatic speaker verification systems.

In order to achieve this objective, an audio fingerprinting system has been implemented, based on a prototipe used for commercial detection on broadcast shows; and several neural network based system have been developped, in pursuance to recognize acoustic patterns in the replay attack audios, caused by the multiple replay devices that can be used on these attacks.

In this multi-device environment, the sound recordings coming out from this attacks can have different qualities, depending on the device's fidelity and the acoustic conditions at the attack's time. The audio fingerprinting and the pattern recognition systems operate at different perfomances, depending on the quality of the audio. The acoustic systems have a better perfomence when the sound recording quality drops, being able to recognice acoustic patterns caused by the replay, which are more precise as the device's fidelity lowers down. On the other hand, the audio fingerprinting system, has a better efficiency when the genuine and spoof audio are more similar.

The complementarity of these strategies has been proved in this work, achieving with the system's fusion a better perfomance than any of the individual systems by their own.

Key words

Deep Neural Network, audio fingerprinting, pattern recognition, ASVSpoof2017, system fusion.

Agradecimientos

Quiero agradecer esta memoria a todos aquellos que han estado conmigo durante este Máster,

primeramente a mis compañeros de clase y amigos, Álvaro Iglesias, Carlos Lamas, Victor Pinazo, Javier Herranz, Esther Sánchez, Beatriz Cámara, Sergio Vivas, Álvaro Palomo, Pablo Ramírez. Con mención especial a Alejandro Martín, el único que se ha leído esta memoria. Este año, el mejor de la universidad, no habría sido el mismo sin vosotros,

también agradecérsela a aquellos amigos que no han estado en este Máster pero aún así han estado cerca, aunque lejos, Pablo Vicente y María de las Mercedes Soto Ramos,

también agradecérsela a mis compañeros de cafés y comidas, Luis Vaquero, Marta Blázquez y Ana Martín, por escuchar mis quejas y bromas sin perder la paciencia,

a mis amigos de Miraflores, año tras año ahí,

a Belén Sancho, siempre acompañándome en todas nuestras aventuras,

a mi familia por estar ahí,

y sobre todo a mis compañeros y profesores de Audias, pasados, presentes y futuros, Diego de Benito, Alicia Lozano, Rubén Zazo, Álvaro Escudero, Daniel Ramos, Javier Franco, Marcos Asenjo, Doroteo Torre, gracias por vuestros consejos, vuestras ideas y vuestro compañerismo; y especialmente a Joaquín González, por ayudarme todo lo que lo has hecho, por enseñarme tanto y por permitirme seguir aprendiendo contigo en esta nueva etapa que comienza ahora.

Índice general

Índice de figuras

Índice de tablas

1 Introducción

Los ataques de reproducción, donde un impostor reproduce una frase de un usuario genuino, son una de las mayores vulnerabilidades de los sistemas de verificación de locutor. Dos escenarios típicos en los que se pueden realizar este tipo de ataques son la grabación oculta del habla de intentos de verificación de usuarios legítimos, o usar los mismos audios genuinos en caso de acceso fraudulento a los canales de transmisión o a los dispositivos de almacenamiento del sistema de verificación.

Para detectar estos ataques de repetición, en el estado del arte se usan sistemas acústicos, basados en reconocimiento de patrones, capaces de encontrar las características acústicas que se forman en los aparatos al grabar y reproducir la voz. Estos sistemas acústicos actualmente están basados en su mayor parte en redes neuronales profundas, existiendo propuestas tanto con redes fully-connected, redes convolucionales y recursivas [21, 26, 34].

Por otro lado, otra forma de abordar el problema es mediante *audio fingerprinting*, comparando el audio del intento de acceso con todos los anteriores intentos de acceso al sistema, grabados en una base de datos. De esta manera, se puede detectar si el nuevo audio es una copia, y por lo tanto un intento de ataque de reproducción, en caso de que se encuentren grandes similitudes con un audio original; o un intento de acceso genuino, si el nuevo audio no se corresponde con ningún audio previamente guardado [14].

En caso de los sistemas acústicos, tienen mejor rendimiento cuanto más degradado esté el audio con el que se realice el ataque, y más artefactos y distorsiones hayan creado los dispositivos de grabación y reproducción, puesto que son estas alteraciones las que buscan los sistemas acústicos de detección de patrones. Sin embargo, con dispositivos de gran fidelidad, o en el supuesto de que se haya podido acceder al audio original, estos sistemas fallarán al no encontrar ninguna deformación acústica [19].

A su vez, los sistemas de *audio fingerprinting* tendrán un mejor funcionamiento cuanto mejor sea la calidad del sonido con el que se intente engañar al sistema, encontrándose mayores semejanzas con el audio original.

A la vista de la gran complementariedad de estas dos aproximaciones, es el objetivo de este trabajo de fin de Máster crear un sistema conjunto que permita usar las bondades de ambos planteamientos; ayudando a los sistemas acústicos, en las reproducciones de alta fidelidad, con un sistema de *audio fingerprinting*, que es en estos audios de gran calidad donde más capacidad de detección presenta.

1.1. Motivación del proyecto

Los sistemas de verificación por voz están a la orden del día como sistemas de verificación biométricos, aún no siendo uno de los sistemas más seguros, tiene ciertas ventajas, como el hecho de poder ser usado a través de una conversación telefónica para autenticarse a distancia y es un sistema con bastante aceptación por el usuario [11].

Uno de los problemas de seguridad que tienen estos sistemas de verificación por voz es la posibilidad de usar grabaciones ilícitas de intentos de verificación de usuarios genuinos. Es el objetivo de mucha investigación el diseñar sistemas de detección de ataques de repetición para reducir este problema de seguridad.

La principal motivación de este trabajo de fin de Máster, es continuar esta investigación, con el objetivo de ampliar las posibilidades de detección de estos sistemas, con el fin último de mejorar la seguridad de la verificación de usuario por voz.

1.2. Objetivos

El objetivo principal de este trabajo, es crear un sistema que una las bondades de los sistemas acústicos y de *audio fingerprinting*. Para ello se han de desarrollar e implementar tanto un sistema acústico de detección de ataques de reproducción, como un sistema de *audio fingerprinting*.

Respecto al sistema acústico, se estudiará el estado del arte, y se partirá de las descripciones de las propuestas con mejor rendimiento para implementar un sistema propio, buscando el mejor rendimiento posible en éste.

Por otro lado, en cuanto al enfoque del *audio fingerprinting*, se partirá de un sistema ya implementado en Matlab®, que se codificará en el lenguaje Python, para mejorar la velocidad y versatilidad, además de reconfigurarlo para adaptarlo a esta tarea, diferente de para la cual fue desarrollado inicialmente.

Por último, se unirán ambos sistemas, y se harán pruebas con el sistema conjunto, y con ambos sub-sistemas por separado para medir el rendimiento y contrastar las hipótesis.

1.3. Metodología y plan de trabajo

Este trabajo está segmentado en tres partes. Por un lado, el desarrollo e implementación del sub-sistema de *audio fingerprinting*; por otro, el mismo proceso con el sub-sistema acústico; y por último, el desarrollo del sistema completo como fusión de ambos sub-sistemas.

La metodología a seguir será la siguiente, diseñar y optimizar cada uno de los sub-sistemas para alcanzar el mejor rendimiento por separado y por último diseñar y optimizar la regla de fusión para crear el sistema final, realizando pruebas y analizando los resultados, en cada una de las etapas del diseño.

Posiblemente, el sistema conjunto no sea el óptimo con los dos sub-sistemas optimizados a alcanzar el mejor rendimiento en la tarea completa, ya que es posible que se pudiera obtener un mejor resultado global optimizando la complementariedad de ambos sub-sistemas, especializando cada uno de ellos, y no siendo tan generalistas. Sin embargo, no se llevará a cabo esta metodología de trabajo, puesto que resulta más interesante usar los recursos disponibles en desarrollar e implementar los dos sub-sistemas para que puedan funcionar con el mejor rendimiento posible por separado, de cara a usarlos como herramientas para otras investigaciones futuras en el ámbito.

1.4. Contribuciones

En este apartado se describen las contribuciones propias a este trabajo.

Por un lado, se ha desarrollado un sistema de *audio fingerprinting* en Python, a partir de un prototipo en Matlab®, diseñado para el reconocimiento de anuncios en programas *broadcast*. Se ha re-escrito el programa en Python, con relativa facilidad,teniendo en cuenta la similitud entre ambos lenguajes de programación, y la existencia de librerías en Python donde se disponen de funciones ideénticas a las de Matlab®. Además, se ha configurado el sistema para su funcionamiento en la tarea de la detección de ataques de reproducción, con longitudes de audio mucho más cortas que para lo que estaba diseñado originalmente el sistema.

Por otro lado, se han desarrollado varios sistemas acústicos; uno basado en GMM, como sistema de referencia; y dos basados en redes neuronales, diseñados a partir de otros sistemas publicados del estado del arte. Estos sistemas han sido desarrollados basándose en la información proporcionada en las publicaciones. No obteniéndose en una primera instancia los resultados deseados, se han probado diferentes técnicas y barridos de hiperparámetros, para mejorar los rendimientos de los sistemas, usando funciones de Keras [9] y, en algunos casos, programando directamente con Tensorflow [1]; consiguiéndose finalmente un rendimiento aceptable, para el objetivo de este trabajo.

Las características CQCC, usadas en algunos de estos sistemas acústicos se han extraído mediante un programa publicado, con pequeñas modificaciones [33]. Sin embargo, para obtener las características HFCC, ha sido desarrollado un extractor para este trabajo.

Finalmente, se ha utilizado el *toolbox* FoCal [6], para la fusión y calibración de las puntuaciones de los sistemas acústicos y de *audio fingerprinting*; y se han realizado las pruebas y obtenido los resultados descritos en el capítulo 5.

2

Estado del Arte

En este capítulo se describirá el estado del arte de las técnicas que se usarán en este trabajo, con el fin de establecer el punto de partida desde donde se abordará el diseño del sistema de detección de ataques de reproducción mediante combinación de DNN en detectores acústicos y *audio fingerprinting*.

Por un lado se describirá la técnica conocida como *audio fingerprinting*, y su evolución. El *audio fingerprinting* es una técnica que consiste en crear una huella acústica a partir del audio, habitualmente desde el espectrograma, con el fin de poder identificar ese audio, y poder ser encontrado en una base de datos; o para monitorizar el uso de ese audio y posibles vídeos asociados, en programas de *broadcast*, por ejemplo, u otros servicios; con el objetivo de controlar *copyrights*, licencias y monetización de contenidos.

Por otro lado, se detallarán los sistemas acústicos usados en el estado del arte, las arquitecturas utilizadas, y los rendimientos esperados. En general, son sistemas de *machine learning*, que aprenden a reconocer los patrones acústicos que se forman al reproducir un audio, como por ejemplo un *password* hablado, mediante un dispositivo, que debido a la acción de sus micrófonos y altavoces que introducen imperfecciones y artefactos acústicos. Cuanta mejor es la calidad de grabación y reproducción, menos acuciados son estos patrones y más complicado es detectar el ataque. Estos sistemas se entrenan mediante una base de datos donde están etiquetados los audios genuinos y las reproducciones ilícitas, de forma que aprendan los sistemas a distinguir automáticamente unos de otros.

2.1. Audio Fingerprinting

El *audio fingerprinting* es un tipo de algoritmo de ACR (*Automatic Content Recognition*). Es una representación determinista condensada de un audio que puede servir para identificar un audio en una base de datos, contando con varias ventajas respecto a comparar los dos audios en sí. Por un lado, reduce el tamaño de los datos a guardar, puesto que en la base de datos no es necesario guardar el audio completo, si no únicamente su *fingerprint* que por diseño es de un tamaño menor y la comparación se hace de forma eficiente, ya que el tamaño de la base de datos completa es más reducido. Además, al obtener el *fingerprint* del audio, se obtiene una representación robusta que elimina variaciones perceptuales como ruido o distorsiones sobre el audio [15].

Existen sistemas comerciales como Shazaam [10], que usan el *audio fingerprinting* para identificar rápidamente segmentos cortos de música grabados mediante un micrófono de teléfono móvil, en presencia de ruido, distorsiones, e incluso a partir del audio comprimido por un códec de voz de una llamada telefónica. Esto lo hacen buscando en una base de datos de millones de audios, eficientemente y de forma escalable.

Para poder identificar un audio, el sistema de *audio fingerprinting* requiere tener guardada, en su base de datos, una representación de ese audio exacto. No es posible reconocer un audio a partir de otro similar, como por ejemplo, intentar identificar una canción a partir de una actuación "en vivo" de esa misma canción, puesto que el tempo, la clave, y los instrumentos pueden ser distintos y la ejecución de la canción puede tener diferencias respecto al original, aunque podría darse el caso de identificar la canción a partir de una interpretación de una pieza clásica, por ejemplo, una obra de piano, debido a la exactitud y precisión del músico instrumental. Con voz cantada, sin embargo, es mucho más improbable replicar dos veces la misma forma de onda exacta [16].

Esto se debe a que el *audio fingerprinting* basa su funcionamiento en las características de bajo nivel del sonido. Mientras que un ser humano es capaz de reconocer una canción a partir de un "tarareo", o de una versión, este sistema no sería capaz, puesto que no recoge información de alto nivel si no las características tiempo-frecuenciales del segmento de audio.

En los siguientes apartados se describirá el funcionamiento de los algoritmos de *audio fingerprinting*, como por ejemplo Shazaam [10].

2.1.1. Principio básico de operación del *audio fingerprinting*

Para obtener el *fingerprint* de un audio, en primer lugar se parte del espectrograma. El espectrograma es una representación en tres dimensiones del audio, siendo una de las dimensiones la frecuencia, la otra, el tiempo y la tercera la energía de la señal en cada punto de tiempo y frecuencia. El espectrograma puede ser calculado mediante un banco de filtros paso banda, midiendo así la energía de cada banda frecuencial, pudiendo hacer esto en un sistema analógico. Por otro lado, se puede generar mediante procesado digital de la señal, a partir de la STFT (*Short-Time Fourier transform*), que divide la señal en ventanas (en general solapadas), y calcula el espectro de cada una de ellas. La expresión del cálculo del espectrograma a partir de la STFT es la siguiente:

$$spectrogram(t, w) = |STFT(t, w)|^2 [27] \tag{2.1}$$

A partir de este espectrograma, se obtienen las coordenadas de los máximos locales en amplitud, reduciendo el audio a una conjunto relativamente disperso de pares de tiempo-frecuencia. Esto reduce el problema de la búsqueda a uno similar al de la astro-navegación, en el cual se

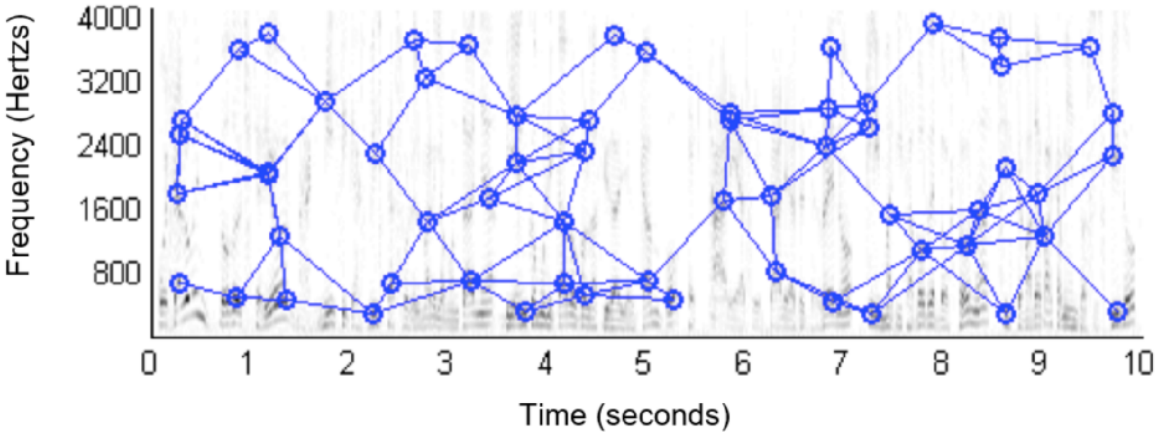

Figura 2.1: Espectrograma de una señal de audio, obtenida de [14]

debe buscar rápidamente una pequeña constelación obtenida del fragmento del audio, en el gran universo de puntos de la base datos.

Los picos se escogen buscando una densidad razonablemente uniforme en el espectrograma, para ello, se divide el espectrograma mediante una cuadrícula uniforme, y dentro de cada cuadrícula se escoge el máximo local. El objetivo de escoger el máximo de amplitud se debe a que estos puntos tendrán una mayor probabilidad de sobrevivir a las distorsiones y el ruido que puede haber presentes durante la captura del audio.

Una vez escogidos los picos del espectrograma, es necesario asociarlos en pares, como se muestra en la figura 2.1, para poder buscar el mismo patrón en audios coincidentes, manteniendo un instante temporal de referencia y otro relativo para cada emparejamiento, permitiendo la localización de un segmento de audio en cualquier posición temporal del audio de la base datos. A partir de estos emparejamientos, se realiza una combinación *hash*, generando un identificador inequívoco para cada uno de los emparejamientos de coordenadas tiempo-frecuenciales. La razón de generar estos *hashcodes*, es aumentar el rendimiento del sistema, mejorando la velocidad de búsqueda.

Por lo tanto, el sistema completo tiene dos partes; en un primer lugar, se genera una base de datos extrayendo de cada audio disponible todos sus picos característicos, emparejándolos y guardando sus coordenadas relativas en forma de *hashcode*. Posteriormente, cuando se tenga un audio nuevo desconocido que se quiera identificar, se debe realizar la misma extracción de las características, y con los *hashcodes* generados por este audio, buscar similitudes en la base de datos generada anteriormente. Estas similitudes se encuentran, a partir de los *hashcodes* iguales encontrados en la base de datos, asociando la coordenada temporal del audio de búsqueda con la coordenada temporal del audio de la base de datos. Las similitudes encontradas serán el número de emparejamientos iguales que mantienen la misma relación temporal.

Según el número de *hashcodes* (es decir, de emparejamientos de picos de tiempo-frecuencia característicos) iguales encontrados en cada audio de la base de datos, se puede tomar la decisión. Cuanto mayor sea el número de coincidencias con un audio, mayor será la confianza con la que se tomará la decisión de que ambos audios son el mismo. Esta decisión se puede tomar mediante un umbral, diseñado con el objetivo de reducir el número de falsos positivos pero permitiendo identificar segmentos de audio cortos [10].

2.2. Sistemas acústicos para la detección de ataques de reproducción

En el estado del arte de la detección de ataques de reproducción hay muchas aproximaciones diferentes, pero todas tienen en común que son sistemas de reconocimiento de patrones y *machine learning*. Dentro de estos son mayoría los sistemas basados en Modelos de Mezclas de Gaussianas (a partir de ahora, GMM, de su traducción al inglés, *Gaussian Mixture Models*) y en Redes Neuronales Profundas (a partir de ahora DNN, de su traducción en inglés, *Deep Neural Network*), siendo estas últimas las que, en general, obtienen mejor rendimiento.

Estos sistemas basan su funcionamiento, a la hora de detectar ataques de reproducción, en reconocer patrones que se forman cuando un audio es grabado y reproducido por algún dispositivo. Estos patrones son de diferente naturaleza, pudiendo ser algún tipo de filtrado de frecuencia, introducción de ruido, por ejemplo ruido de cuantificación, u otros artefactos acústicos de diversa índole. El objetivo es usar estos sistemas de *machine learning* para que aprendan automáticamente, sin necesidad de programación específica, a detectar estos patrones. Este aprendizaje se realiza enseñando al modelo diferentes ejemplos de audios, genuinos y provenientes de ataques de reproducción, etiquetados como tales. Así, el modelo aprenderá a reconocer estos patrones que sólo tienen los audios impostores.

Un Modelo de Mezclas de Gaussianas, o GMM [28], es una función de densidad de probabilidad representada como una suma ponderada de componentes gaussianas. Los GMMs son usados para realizar modelos paramétricos de distribuciones probabilísticas, y sus parámetros son estimados a partir de datos de entrenamiento, usando el algoritmo iterativo EM (*Expectation-Maximization*), o mediante una estimación MAP (*Maximum A Posteriori*) a partir de un modelo ya entrenado con el fin de adaptarlo a los nuevos datos.

Los sistemas basados en GMM en esta tarea, son actualmente utilizados como *baseline*, es decir, como sistema de referencia, para tener una noción del rendimiento que se puede conseguir, por ejemplo, usando diferentes tipos de características extraídas de los datos.

Por otro lado las redes neuronales profundas, son el nuevo paradigma de la inteligencia artificial, estando presentes en la mayor parte de los sistemas de *machine learning* de los últimos años, habiendo demostrado resultados impresionantes en muchas áreas de investigación, dentro del reconocimiento de patrones y fuera de él. Siendo una parte fundamental de este trabajo la implementación de diferentes arquitecturas de redes neuronales para conseguir unos resultados similares a los conseguidos por los sistemas del estado del arte, se realizará en los próximos apartados una breve introducción a las redes neuronales y las diferentes arquitecturas existentes.

2.2.1. Redes Neuronales Profundas

Las redes neuronales profundas es la técnica más extendida de resolución de problemas de reconocimiento de patrones. Como indica su nombre, están basadas, vagamente, en el funcionamiento del de las neuronas de los cerebros animales y su capacidad de aprender a partir de ejemplos.

Desde mediados del siglo XX [25, 35], se han propuesto diferentes modelos de neuronas artificiales, pero no fue hasta en los años 80, cuando se aceleró el interés en este ámbito con la publicación del algoritmo del *back-propagation* [29], usado para entrenar una red neuronal. *Back-propagation* ha sido el algoritmo más usado y estudiado sobre el entrenamiento de redes neuronales desde entonces y la mayor parte de los algoritmos para entrenar redes hoy en día se basan en él.

La estructura básica de una red neuronal artificial es un conjunto de nodos con alguna función de activación, que simularían las propias neuronas biológicas) unidas por conexiones ponderadas (representando las conexiones sinápticas entre neuronas). La red se estimula por una entrada y las conexiones ponderadas transmiten el impulso por toda la red hasta la salida, que sería la respuesta de la red a la entrada.

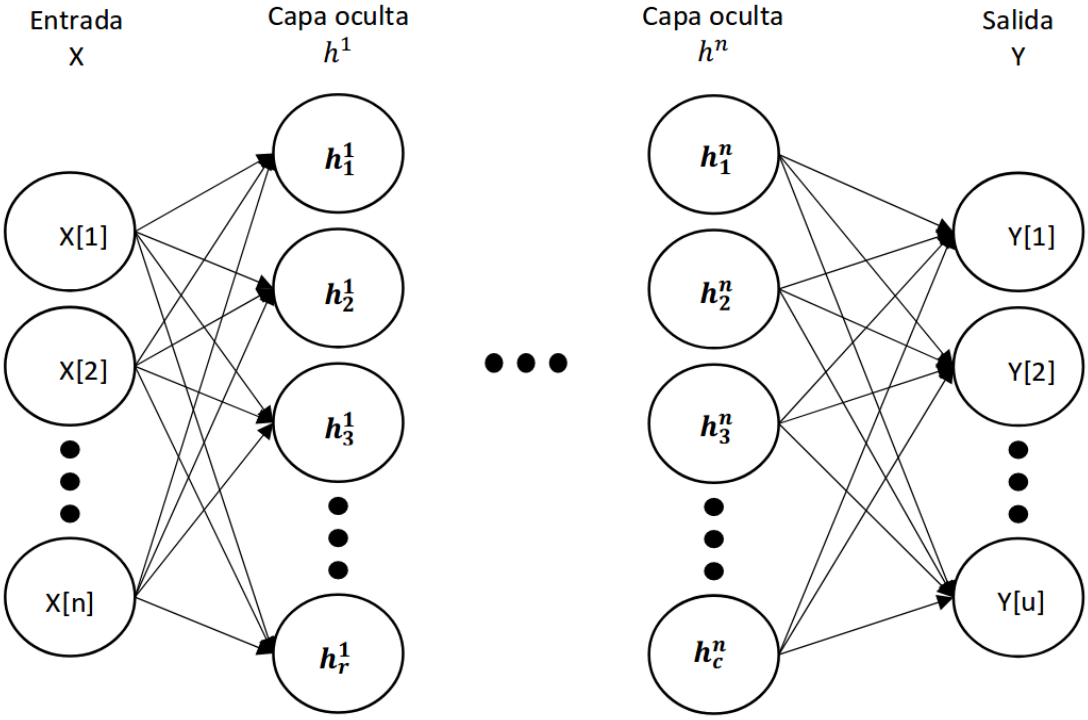

Figura 2.2: Estructura básica de una red neuronal, formada por una entrada de dimensión n, una o varias capas ocultas de diferentes o iguales dimensiones, unidas mediante conexiones ponderadas, y una salida con dimensión u.

La idea del algoritmo de *back-propagation* es encontrar los pesos que minimicen una función de error usando el método del descenso por gradiente. Por lo tanto, la solución del problema de aprendizaje es encontrar la combinación de pesos que minimiza la función de error para un conjunto de datos de entrenamiento dado. Calculando la derivada parcial del coste de la función, respecto a cada una de los pesos de la red, es posible establecer cuánto es de responsable cada peso del error total, pudiendo en cada iteración del algoritmo ir modificando estos pesos con el fin de reducir el coste, permitiendo encontrar los pesos óptimos que minimizan la función de coste. El hecho de que este método necesite el cálculo del gradiente de la error de coste, genera una restricción en las posibles funciones de activación de las neuronas.

La función de activación de las neuronas tienen el objetivo de realizar una transformación de la entrada de cada capa a la salida. Si la función de activación fuese lineal, teniendo en cuenta que cualquier combinación de operadores lineales es también un operador lineal, varias capas de una red neuronal se podrían resumir en una única capa con función de activación lineal. Sin embargo, si las funciones de activación son no lineales, usando varias capas de neuronas, consiguen una transformación mucho más potente, pudiendo crear fronteras de clasificación más complejas y modelar problemas no lineales. Es común buscar empíricamente el número óptimo de capas ocultas de una red, dependiendo de lo complejo que sea el sistema a resolver [39]. Típicas funciones de activación utilizadas en la literatura son: la tangente hiperbólica, la función sigmoide, lineal o la función ReLU (*Rectified Linear Unit*, utilizada principalmente por su bajo coste computacional) [20].

La función de coste en una red neuronal, como en cualquier problema de *machine learning*, permite medir el error de la predicción de nuestro modelo con respecto a la respuesta correcta. En problemas de clasificación es común usar la entropía cruzada, aunque hay más tipos de error comúnmente usados [4].

Según el problema a resolver, se han desarrollado diferentes arquitecturas de redes neuronales. Por un lado, las redes *fully-connected*, o totalmente conectadas, son el esquema básico de red neuronal que se muestra en la figura 2.2, en las cuales la entrada es un vector unidimensional y información se propaga por la red en la cual cada neurona de una capa está conectada con cada una de las neuronas de la capa anterior y posterior.

Por otro lado, las redes convolucionales, son un tipo de red neuronal, donde cada capa oculta está separada por lo general en dos partes, una capa convolucional y una capa de diezmado [22]. La capa convolucional funciona como extractor de características. Cada unidad en esta capa está conectado a un conjunto de unidades de la capa oculta anterior, según la forma de un filtro bidimensional; se calcula la activación de la capa convolucionando la entrada con el filtro (siendo estos los pesos entrenables durante el aprendizaje) y aplicando una función de activación no lineal. Por otro lado, la capa de diezmado reduce el tamaño de las representaciones obtenidas por la capa convolucional anterior, según una función de diezmado.

Las redes convolucionales suelen ser usadas con representaciones bidimensionales como entrada, imágenes por ejemplo [20], pero también pueden ser usadas con audio a partir de los vectores de características extraídos de él (vectores de MFCCs, o un espectograma) como en la figura 2.3. Esta arquitectura tiene la ventaja de ser de bajo coste en cuanto a términos de número de parámetros entrenables [24].

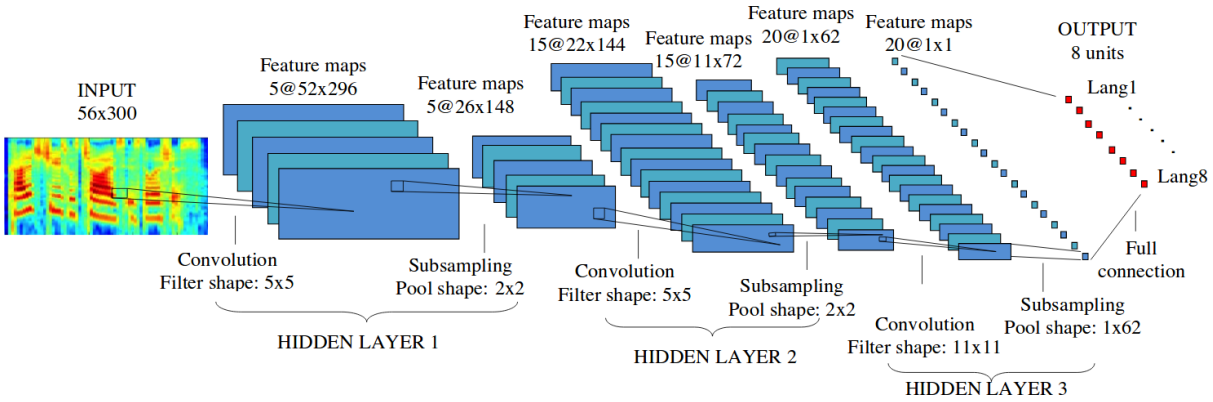

Figura 2.3: Arquitectura típica de una red convolucional, en este caso, aplicada al reconocimiento de idioma, extraída de [24].

2.2.2. Sistemas del estado del arte basados en redes neuronales profundas para la detección de ataques de reproducción

Una vez hecha una pequeña introducción de las redes neuronales profundas, es interesante estudiar, las diferentes arquitecturas y modelos usados en el estado del arte de la detección de ataques de reproducción mediante sistemas acústicos. En la tarea *ASVSpoof 2017*, descrita más adelante, los sistemas ganadores estaban basados en redes neuronales.

En el sistema implementado en [21], se consigue muy buenos resultados (EER= 7.37 %), con una red convolucional como extractor de características. En el sistema, se parte del audio en forma de espectrograma y se ha diseñado una red formada por 9 capas convolucionales, con una activación MFM (*Max-Feature-Map*) [37], que juega un papel de selector de características y permite reducir el tamaño de la red; una capa *fully-connected*, cuyo objetivo es realizar una

transformación del espacio para realizar la clasificación, acabando en una capa *softmax*, con dos posibles salidas (audio genuino o ataque de repetición). Después del entrenamiento de la red, se usa la salida de la capa *fully-connected* como *embedding*, es decir, usar los valores de salida de una capa de la red en forma de vector de características. Con este *embedding* se entrena un clasificador basado en GMM, con el fin de discriminar entre las dos clases en este espacio de baja dimensión a partir de las características obtenidas de la red.

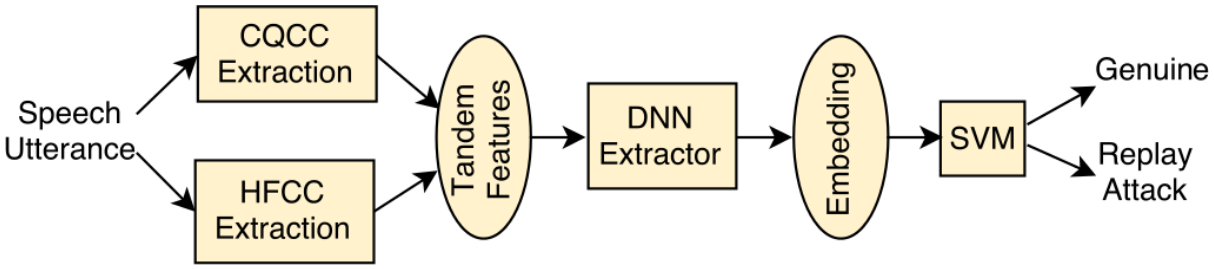

Figura 2.4: Sistema diseñado en [26], las caracterísiticas CQCC y HFCC se usan para entrenar una red neuronal *fully-connected*, que es utilizada como *embedding* para un SVM (*Support Vector Machine*) encargado finalmente de realizar la clasificación.

Por otro lado, en [26], cuyo sistema queda esquematizado en la figura 2.4, se obtienen también muy buenos resultados (EER= 11.5 %). En este sistema se parte de una combinación de coeficientes CQCC (*Constant-Q Cepstral Coefficients*) [32] y HFCC (*High-Frequency Cepstral Coefficient*) [26]. Los CQCC son unas características derivadas de la transformada CQT (*Constant-Q*), que pretende mantener una escala perceptual, inspirada en el oído humano, al estilo de los filtros Mel. Los HFCC se centran en la zona de alta frecuencia de la señal, argumentando que los dispositivos de grabación y reproducción, están en su mayor parte diseñados para habla telefónica, y pueden tener mayor predisposición a presentar artefactos acústicos en ciertas frecuencias fuera de la banda de voz (300-3400 Hz). Estos artefactos acústicos son presumiblemente más pronunciados en dispositivos de baja calidad, puesto que los dispositivos de alta calidad tienden a tener una respuesta en frecuencia más plana.

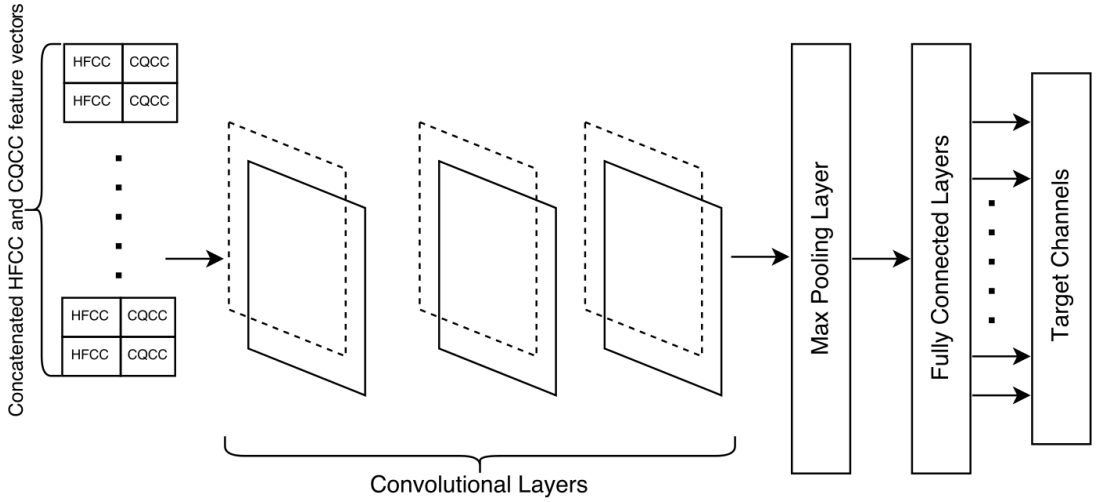

Figura 2.5: La red neuronal diseñada en [26], donde se usan tres capas convolucionales como extracción de características a partir de los coeficientes CQCC y HFCC, y cuya salida objetivo son las diferentes configuraciones de dispositivos, pretendiendo capturar en la capa anterior la información del canal para usarlo como embedding.

Una vez extraídas las características, los autores proponen una red formada por tres capas convolucionales y tres *fully-connected* como se muestra en la figura 2.5, considerando dos tipos de estrategias de clasificación como salida a la hora de entrenar la red, por un lado una estrategia de

clasificación binaria que distinga entre audios genuinos y ataques de reproducción, y por otro lado una estrategia de clasificación multi-clase, que distinga entre las diferentes condiciones de canal existente, según los dispositivos de grabación y reproducción disponibles en la base de datos de entrenamiento. Esta segunda aproximación es con la que mejor rendimiento consiguieron, permitiendo una mejor generalización a condiciones de canal no vistas durante el entrenamiento, probablemente porque se consigue capturar en los *embeddings* una mayor información del canal, que es la que supuestamente permite distinguir los audios genuinos de las reproducciones ilícitas.

Por último, con los *embeddings* obtenidos de la red, se entrena una frontera de decisión lineal mediante SVM, para discriminar entre las clases genuino e impostor.

Estos son los dos mejores sistemas de la evaluación *ASVSpoof 2017*. Por supuesto, existen muchos otros sistemas, basados en redes neuronales [7, 36, 2] con diferentes arquitecturas, o en otras estrategias de *machine learning*. Los dos sistemas de mejor rendimiento, tienen en común que usan las redes neuronales como extractor de características, utilizando *embeddings* a partir de ellas para entrenar otros clasificadores (GMM o SVM). Existen propuestas, como en [8], de diseñar un sistema *end to end*, basado exclusivamente en redes neuronales, obteniéndose en general peores rendimientos.

3

Entorno experimental

3.1. Objetivo

La verificación de locutor automática (ASV), es una tecnología usada en una creciente cantidad de aplicaciones, que requiere no sólo robustez a cambios acústicos para poder funcionar correctamente en diferentes condiciones, sino también resistencia a intentos de engañar al sistema. Entre otros posibles tipos de ataques, los ataques de repetición son un problema clave, puesto que pueden ser realizados fácilmente y su eficacia es bastante alta. Los ataques por reproducción se realizan usando grabaciones de la voz de un locutor objetivo, y reproduciéndolas en un sistema de ASV (*Automatic Speaker Verification*) en vez de la voz genuina del locutor. Un ejemplo de esto sería usar un dispositivo para reproducir la voz de un locutor para desbloquear un *smartphone* que use ASV como control de acceso.

Varios tipos de contramedidas han sido desarrolladas para proteger a los sistemas ASV de estas estrategias. Algunos de estos ejemplos son, requerir una frase distinta en cada intento de acceso, o pedir una serie de dígitos aleatorios. Estas medidas consiguen bastante protección, sin embargo, múltiples grabaciones pueden ser mezcladas para producir un ataque de reproducción que coincida con la frase pedida. El *audio fingerprinting* puede ser usado para detectar grabaciones de intentos anteriores, con la desventaja de necesitar el mantenimiento de una base de datos creciente con el uso del sistema. Por último, existe la posibilidad de detectar los ataques de reproducción usando exclusivamente las características acústicas del audio del intento de acceso. Esto tiene bastante dificultad, debido a la impredecible variabilidad de la calidad del ataque de reproducción. Las grabaciones, conseguidas posiblemente de manera encubierta, pueden contener ruido, o distorsiones, pudiendo convertir el problema de la detección en un problema de clasificación de ruido o distorsión de canal. Sin embargo, con grabaciones hechas con dispositivos de alta calidad, o incluso con accesos a la propia circuitería del sistema ASV, pudiendo inyectar copias exactas de los intentos de accesos anteriores, los ataques pueden ser indistinguibles de los accesos legítimos.

Con el fin de encontrar los límites prácticos en la detección de ataques de reproducción, se diseñó la evaluación *ASVSpoof 2017* [19]. Esta evaluación, fomenta la participación de universidades y centros de investigación, publicando una base de datos de desarrollo etiquetada para el diseño de los sistemas, y evaluando cada solución enviada sobre una base de datos de *test*, sin etiquetas públicas para los participantes del reto. Así, se consigue evaluar los diferentes sistemas

del estado del arte, midiendo el rendimiento y las capacidades de detección para esta tarea.

3.2. Base de datos

La base de datos consiste en una parte del corpus dependiente de texto *RedDots* [23], además de una versión grabada y reproducida por diferentes dispositivos de esos mismos datos [18]. Esta base de datos ha sido construida para la evaluación de los sistemas de detección de ataque por repetición.

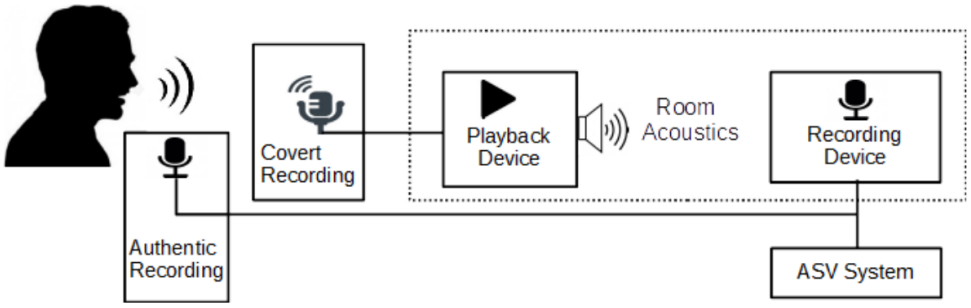

Figura 3.1: Arquitectura esquemática de un ataque de repetición, recreada para generar la base de datos, obtenida de [18].

En total, se han utilizado 3498 audios genuinos, una parte de la base de datos *RedDots*, consistente en diez frases cortas comunes, como podría ser *Ok, Google*, habladas por un total de 49 locutores distintos [18]; y un total de 14532 intentos de ataque de reproducción. Estos audios fueron creados a partir de la base de datos anterior, simulando un ataque de reproducción real esquematizado en la figura 3.1, usando una gran variedad de dispositivos de grabación y de reproducción, mostrados en la figura 3.2, para recrear diferentes escenarios de ataque de reproducción.

ID	Playback device	ID	Recording device
P1	ACER "Ferrari ONE" netbook	**R1**	AKG C562CM + Marantz PMD670
P2	All-in-one PC speakers	**R2**	BQ Aquaris M5 smartphone. Software: Smart voice recorder
P3	BQ Aquaris M5 smarphone	**R3**	Desktop Computer with headset and arecord
P4	Beyerdynamic DT 770 PRO headphones with PC	**R4**	H6 Handy Recorder
P5	Creative A60 connected to laptop	**R5**	Logitech C920 connected to Dell (SSD) notebook
P6	Dell (SSD) notebook + EdirolUA25 + XXX	**R6**	Nokia Lumia
P7	Dell laptop with internal speakers	**R7**	Røde NT2 microphone with a laptop
P8	Dynaudio BM5A Speaker connected to laptop	**R8**	Røde smartlav+ mic with a laptop
P9	HP Laptop speakers	**R9**	Samsung GT-I9100
P10	High-end GENELEC 8020C studio monitorss	**R10**	Samsung GT-P6200
P11	MacBook pro internal speakers	**R11**	Samsung Galaxy 7s
P12	PC with Altec lansing Orbit USB iML227 speaker	**R12**	Samsung Trend 2
P13	Samsung GT-I9100	**R13**	Samsung Trend 3
P14	Samsung GT-P6200	**R14**	ZoomHD1
P15	VIFA M10MD-39-08 Speakers with laptop	**R15**	iPhone 5c
		R16	iphone4

Figura 3.2: Diferentes dispositivos de grabación y reproducción utilizados en la evaluación *ASVSpoof 2017* [18].

Posteriormente, la base de datos fue dividida, con el fin de realizar la evaluación, en entrenamiento (3014 audios), desarrollo (1710 audios) y evaluación (13306), estando las etiquetas de esta última fracción, ocultas a los participantes del reto *ASVSpoof 2017*. Una vez finalizada la evaluación, se liberaron las etiquetas, permitiendo realizar los experimentos y pruebas en los que se basa este trabajo.

Los dispositivos de grabación y reproducción utilizados para cada simulacro de ataque de reproducción tienen diferentes calidades, además existen diversos posibles entornos de grabación y reproducción, con posibilidad de presencia de ruido, o uso de cámaras resonantes y anecoicas.

En el conjunto de entrenamiento hay cuatro configuraciones únicas, y en la de desarrollo 9 configuraciones posibles. Estas configuraciones han sido utilizadas para realizar el entrenamiento de una red, descrita en la sección 4.2.3, para capturar la información de canal.

El gran número de dispositivos y configuraciones, conlleva que la base de datos puede ser dividida en función de la calidad del audio del ataque de reproducción.

En el conjunto de evaluación, se puede distinguir un conjunto de alta calidad, denominado HQ1 con 1982 audios, de donde se puede extraer un subconjunto de muy alta calidad, denominado HQ2, con 361 audios. Estos conjuntos de alta calidad son especialmente útiles en este trabajo, puesto que es en ellos donde reside la hipótesis de complementariedad entre los sistemas basados en *audio fingerprinting* y los sistemas acústicos diseñados para la detección de ataques de reproducción.

3.3. Medidas de rendimiento

En este trabajo se ha utilizado la base de datos de la evaluación *ASVSpoof 2017* para realizar el entrenamiento de los sistemas y medir sus rendimientos. Existen muchos sistemas publicados del estado del arte que han sido diseñados para esta evaluación. La medida de rendimiento usada por la evaluación es el EER (*Equal Error Rate*). Esta figura de rendimiento indica el punto en el cual el FAR (*False Acceptance Rate*) y el FRR (*False Rejection Rate*), son iguales, y el punto óptimo de funcionamiento del sistema, en cuanto a error. Cuanto más bajo sea el EER del sistema, mejor será el rendimiento de éste, siendo robusto a datos no balanceados, como es el caso en la base de datos utilizada en *ASVSpoof 2017*, cuyo conjunto de datos de evaluación tiene 12008 audios de ataque de reproducción y 1298 audios genuinos (una proporción de 9.25 a 1 aproximadamente).

Para poder comparar el rendimiento de los sistemas publicados y los implementados en este trabajo, se ha usado el EER como medida de rendimiento durante el desarrollo, para optimizar el rendimiento en la tarea. Además se ha utilizado el EER para comparar el rendimiento de los sistemas desarrollados y la fusión entre estos.

4

Diseño Sistemas

En este capítulo se presentarán los diferentes sistemas diseñados durante este trabajo y sus implementaciones.

Por un lado, se describirá el sistema de *audio fingerprinting* desarrollado, su funcionamiento interno y la razón de elección de sus parámetros.

Además, se expondrán los diferentes sistemas acústicos de detección de ataques de reproducción implementados, con las características de cada uno de ellos, las características usadas.

Por último se detallará la fusión de ambos sub-sistemas, con el fin de implementar el sistema completo y comprobar las hipótesis de complementariedad entre ambas soluciones.

4.1. *Audio Fingerprinting*

Como se describió en el capítulo 2, el *audio fingerprinting* es una técnica utilizada para extraer una huella acústica de un audio, con el fin de poder identificarlo y poder ser encontrado en una base de datos. En el problema de los ataques de reproducción, la idea es usar un sistema de *audio fingerprinting* cuando se realiza un intento de verificación con el sistema, para buscar su huella acústica en una base de datos donde se guardan todas las huellas acústicas de los intentos exitosos anteriores. Si no coincide con ningún audio de la base de datos, se considera que es un audio genuino, se guarda su huella acústica en la base de datos, que crece dinámicamente con el uso del sistema, y se permite el paso al sistema de reconocimiento de locutor. Sin embargo, si la huella del audio del nuevo intento de verificación, coincide con algún otro intento guardado en la base de datos, se trata de un ataque de repetición, puesto que se está usando una grabación de un audio anterior.

En este trabajo, se ha partido de un sistema implementado en Matlab®, diseñado para la detección de anuncios en programas de *broadcast*, para desarrollar un sistema de detección de ataques de repetición, escrito en Python. Para ello, se ha estudiado el funcionamiento del sistema anterior y se ha programado en el nuevo lenguaje, cambiando los parámetros necesarios, para el uso eficiente en este tipo de problema. Debido a la relativa similitud entre ambos lenguajes de programación y la existencia de librerías en Python que contienen réplicas de las funciones de Matlab®, se ha podido realizar la re-escritura del código con relativa facilidad, pero se ha prestado atención en codificar de una forma más eficiente para mejorar la velocidad del sistema.

4.1.1. Descripción del sistema

El sistema de *audio fingerprinting* diseñado se basa en la extracción de puntos característicos tras un análisis de tiempo-frecuencia. Después se realiza un emparejamiento de esos puntos y se realiza una combinación *hashcode* para poder realizar una búsqueda rápida en una base de datos.

Para realizar el análisis de tiempo frecuencia, en primer lugar se realiza un sub-muestreo de la señal a 8 Khz, considerando que la señal va a tener más información entre 0 y 4KHz por la influencia de la voz, y consiguiendo aligerar computacionalmente el sistema. A partir de la señal filtrada, se calcula su espectograma usando STFT, realizando un enventanado de la señal con una ventana de tipo *hamming*, de longitud 64 ms y un solapamiento del 50 %. En el eje de la frecuencia, se realiza un sobremuestreo a 2048 puntos.

Desde este espectrograma, se puede realizar la extracción de puntos característicos que permiten la identificación de la señal de audio. Estos puntos son los máximos de energía en una región determinada, suponiendo que escoger estos puntos aportan más robustez frente a distorsiones y ruido que puedan degradar el audio [10]. Los puntos característicos, se escogen dividiendo el espectrograma linealmente en tiempo y en frecuencia, para conseguir una distribución uniforme de los puntos a lo largo del espectrograma. Así, se escoge como punto característico, el punto de mayor energía de una cuadrícula de un segundo en tiempo y 200 Hz en frecuencia, obteniéndose un total de 20 puntos característicos por segundo. En el sistema anterior, del cual se parte, al estar diseñado para anuncios en programas de *broadcast*, podía permitirse una densidad de puntos menor, al tener audios de búsqueda más largos. Sin embargo, en este problema, las *passwords* habladas pueden ser muy cortas, necesitándose así una mayor densidad de puntos, con lo que se consigue una huella acústica más detallada, pero ocupando más espacio en la base de datos y aumentando la carga computacional del sistema.

A partir de los puntos característicos, es necesario realizar emparejamientos entre ellos, manteniendo un instante temporal de referencia y un instante temporal relativo, para poder generar un patrón identificable y poder realizar la búsqueda de audios coincidentes, o en partes de estos audios. Además, se aumenta considerablemente la velocidad de la búsqueda en la base de datos [10]. Para realizar estos emparejamientos, se realiza un búsqueda alrededor de cada punto característico, emparejándolo con todos los puntos en una zona del espectrograma de 2 segundos en tiempo y 2KHz en frecuencia. Esta zona ha sido elegida empíricamente, pretendiendo conseguir un número de emparejamientos alto.

Nótese que si la probabilidad de que un pico sobreviva es p, la probabilidad de que un emparejamiento de dos picos sobreviva es de aproximadamente p^2 , por lo que se se necesitan bastantes emparejamientos por cada pico para no perder robustez, mientras que se mejora enormemente la velocidad del sistema usando *hashcodes* combinacionales [10].

A partir de la información de cada emparejamiento se realiza la combinación *hash*, calculándose un identificador inequívoco para cada emparejamiento. Este cálculo se realiza concatenando los bits de las coordenadas de los emparejamientos.

$$[F_1 : F_2 : \Delta T] : T_1 : ID \tag{4.1}$$

Donde F_1 es la frecuencia del primer punto característico, F_2 la del segundo punto característico, ΔT la diferencia de tiempos entre el primer y el segundo punto, T_1 el tiempo absoluto del primer punto característico y ID el identificador del fichero de audio. La información de cada coordenada se concatena en bits, formándose un identificador *hash*, de 64 bits, donde queda guardada la información completa del emparejamiento, mejorándose el rendimiento del sistema y la velocidad de búsqueda.

La base de datos se crea con los *hashcodes* extraídos de los audios conocidos, es decir, los intentos de acceso genuinos anteriores. Cuando se realiza un nuevo intento de acceso con el sistema ASV mediante una *password* hablada, se extrae su la huella de audio, en forma de *hashcodes* y se realiza una búsqueda en la base de datos, calculándose una puntuación para ese audio, que será usada para determinar si el intento de acceso es genuino, o ha sido fruto de un ataque de repetición.

Cada uno de los *hashcodes* obtenidos del nuevo intento de acceso, se busca en la base de datos, consiguiéndose un número de similitudes (*hashcodes* coincidentes), con cada uno de los audios guardados previamente, pudiendo ser un número muy pequeño, o incluso cero. El audio que mayor número de similitudes tenga será el escogido por el sistema, y este número será la puntuación del intento de acceso, a mayor puntuación, más posibilidades tiene el intento de acceso de ser un ataque de repetición, puesto que la puntuación está relacionada con el número de emparejamientos de puntos característicos iguales entre dos audios, y a mayor número, más probable es que los dos audios sean el mismo, o una copia del audio con algún ruido o distorsión.

4.1.2. Umbrales de decisión del sistema de *audio fingerpinting*

Para determinar, a partir de una puntuación obtenida del sistema, si dos audios son coincidentes, es necesario fijar una puntuación mínima de decisión, o umbral, con el fin de minimizar la aparición de falsos positivos y maximizar el número de detecciones correctas.

En este trabajo, se utiliza el EER, siendo esta figura de rendimiento el error en el umbral óptimo, puesto que indica lo mejor que puede funcionar un sistema en los datos de evaluación, al ser el punto donde el FAR y el FRR son iguales, como se describió en el capítulo 3.

4.2. Sistemas acústicos para la detección de ataques de reproducción

Los ataques de reproducción usados para intentar acceder a través de un sistema de ASV, se basan en la grabación de intentos de acceso legítimos y su reproducción, mediante dispositivos que pueden ser de diferentes calidades, posiblemente en presencia de ruido, y sin tener las mejores condiciones de grabación y reproducción, ya que en general, son realizados de forma oculta. Estas condiciones han intentado ser replicadas en la base de datos usada para el desarrollo del sistema, descrita en el capítulo 3, donde se usan diferentes dispositivos, frases y ambientes para obtener simulacros de posibles ataques de reproducción.

En estas circunstancias, es posible usar sistemas acústicos para la detección de estos ataques de reproducción. Estos sistemas acústicos se basan en técnicas de reconocimiento de patrones, con el fin de detectar aquellas distorsiones y artefactos acústicos que se generan al grabar y reproducir las *passwords* habladas, al realizar un intento de acceso ilícito.

En este trabajo, se han implementado varios sistemas, basándose en el estado del arte, descrito en el capítulo 2, donde se detallan los sistemas que mejor rendimiento tienen, a la hora de detectar ataques de reproducción. Como ya se describió, estos sistemas se basan en su mayor parte en DNN, pero se ha decidido implementar un sistema *baseline* basado en GMM, para comparar los rendimientos alcanzados respecto a una base de referencia.

4.2.1. Sistema de referencia basado en GMM

Un GMM es un modelo paramétrico creado a partir de una suma ponderada de componentes gaussianas [28]. Los parámetros a estimar son las medias de las gaussianas, las matrices de

covarianzas y la matriz de pesos de ponderación. Estos parámetros se estiman generalmente mediante el algoritmo iterativo EM [13].

El modelo implementado como sistema de referencia, se basa en el implementado en [12], utilizado como *baseline* en la evaluacion *ASVSpoof 2017*. En esta implementación, se usan dos mezclas de gaussianas de 512 componentes, modelando las clases genuina e impostora de los datos de entrenamiento.

Para mejorar el modelado mediante las mezclas de gaussianas, se han extraído de los datos los CQCC, usando el código publicado en [33], que sigue el esquema mostrado en la figura 4.3. Los CQCC han demostrado capturar muy bien la información necesaria para la detección de ataques de reproducción, por lo que son unas características usadas también en algunas implementaciones del estado del arte basadas en redes neuronales.

Estos coeficientes se extraen a partir de la CQT [38], una variante de la STFT, que busca una mayor similitud con el sistema auditivo humano. Como se muestra en la figura 4.1 y 4.2, en la transformada CQC, se busca tener una resolución temporal mayor en las altas frecuencias y una mejor resolución espectral en las bajas frecuencias, imitando el funcionamiento del oído humano.

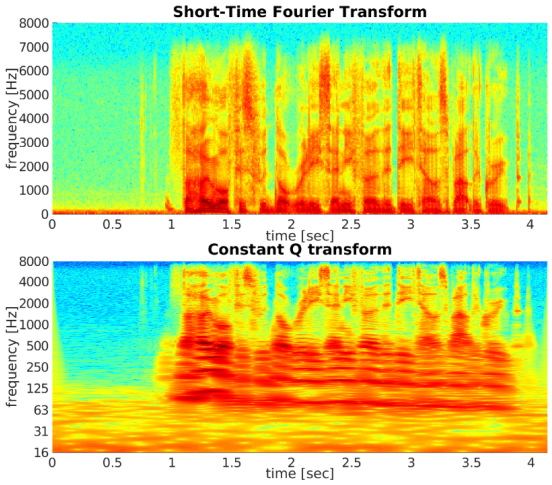

Figura 4.1: Comparación del espectrograma STFT y CQT, de la frase *"the woman is a star who has grown to love the limelight"* (parte de la base de datos de la evaluación *ASVSpoof 2017*). Se puede comprobar la mayor resolución de las bajas frecuencias que consigue la CQT. Figura obtenida de [33].

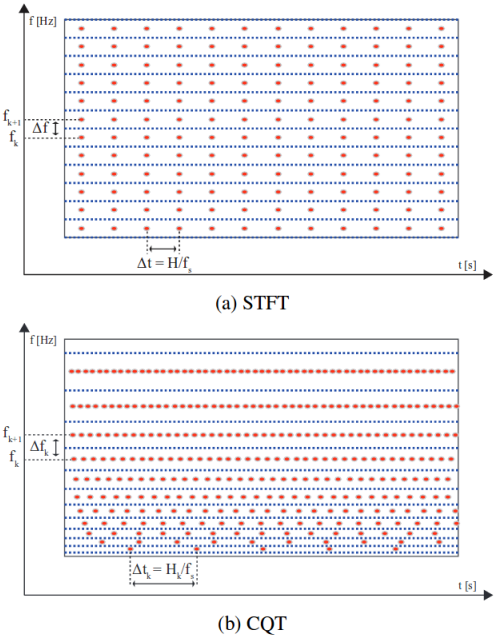

Figura 4.2: Diferencia entre la STFT y la CQC, consiguiendo en esta última transformada una mayor resolución frecuencial para las bajas frecuencias y una mayor resolución temporal para las altas frecuencias [33].

Con ello se consigue un tipo de espectrograma donde se tiene mucha más información espectral en las bajas frecuencias, siendo estas características más informativas para la detección de ataques de reproducción [33].

Para entrenar los modelos de mezclas de gaussianas, se han extraído de cada audio, recortándolo a un máximo de un segundo de longitud, las características CQCC, obteniéndose un total de 19 coeficientes CQCC por cada ventana más los coeficientes delta y de aceleración,

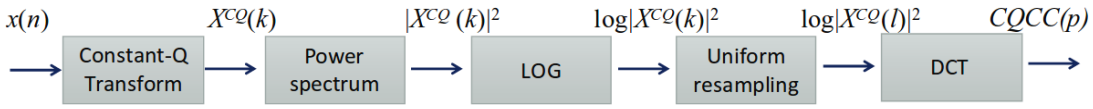

Figura 4.3: Diagrama de bloques de la extracción de los coeficientes CQCC. En un primer lugar se realiza la transformada CQC, de donde se saca el logaritmo del módulo al cuadrado. A partir de esa representación, se realiza un re-muestreado uniforme, ya que existen diferentes escalas, como se muestra en la figura 4.2. Este proceso queda detallado en [33]. Por último se realiza la transformada del coseno y se escogen los primeros coeficientes, de mayor energía.

obteniéndose por cada audio recortado, 117 ventanas de 57 coeficientes cada una. Para construir las etiquetas de entrenamiento por cada ventana, se replica la etiqueta de cada audio, creando una etiqueta por ventana.

Los GMM se entrenan usando el algoritmo de *Expectation-Maximization* [13], con inicialización aleatoria y utilizando matrices de covarianza diagonales para cada componente, entrenando un GMM para los audios de la clase impostor y otro GMM para los de la clase genuina.

Posteriormente, con los datos de test, tras la extracción de las características, se clasifica cada audio según la suma ponderada del *log-likelihood* de cada vector CQCC a cada uno de los modelos, asumiéndose independencia entre los vectores de características,

$$log\ p(X|\lambda) = \frac{1}{T}\sum_{t} log\ p(\bar{x}_t|\lambda) \tag{4.2}$$

donde X es el audio completo y $\bar{x}_t$ cada una de las ventanas. El audio se clasificaría según el modelo que más *log-likelihood* tenga, como genuino o impostor, y su puntuación será la resta de los *log-likelihood* de cada modelo [3].

Este sistema basado en GMM será usado como referencia de rendimiento para los sistemas basados en redes neuronales implementados, que se describirán en los siguientes apartados.

4.2.2. Sistema 1 basado en CNN + GMM a partir de espectrograma

En este apartado se describirá el sistema implementado, basado completamente en redes convolucionales, imitando al implementado en [21]. Las redes convolucionales son un tipo de red neuronal que se suelen usar con representaciones bidimensionales, como se describió en el capítulo 2. En el sistema aquí implementado, se partirá de un espectrograma, implementado a partir de la STFT, que la red convolucional interpretará como si se tratase de una imagen (para lo que originalmente fueron diseñadas).

Las redes convolucionales tienen un menor número de parámetros que las redes *fully-connected*, debido a que muchos de sus pesos se comparten. Además, al usarse habitualmente con capas de *pooling*, se consigue reducir la dimensión según se van incluyendo capas. Esto es útil para trabajar con problemas con pocos datos, como es nuestro caso, ya que se consigue reducir el *over-fitting* al reducir el número de parámetros entrenables.

Los datos de entrada son un espectrograma, descrito en el capítulo 2, usando ventanas de 65 ms con un solapamiento muy alto, de aproximadamente el 90 %. La razón de realizar así la extracción de características, ha sido la de implementar de la forma más parecida posible el sistema de [21]. Existiendo muchos parámetros de configuración y de entrenamiento que no están claros, o no se revelan en ese artículo, se han buscado empíricamente los parámetros a partir de las pruebas realizadas en el conjunto de desarrollo, como se comentará en el capítulo 5.

Esta red se ha implementado utilizando Keras y Tensorflow [9, 1]. La estructura de la red, mostrada en la figura 4.4, se compone de 9 capas convolucionales, que realizan una extracción

de características, y dos capas *fully-connected*, que realizan una transformación del espacio para realizar la clasificación en una última neurona, con función de activación sigmoide, clasificando en dos clases. Como función de coste se usa la entropía cruzada y como como optimizador Adam [17].

Type	Filter / Stride	Output	#Params
Conv1	5×5 / 1×1	$864 \times 400 \times 32$	832
MFM1	–	$864 \times 400 \times 16$	–
MaxPool1	2×2 / 2×2	$432 \times 200 \times 16$	–
Conv2a	1×1 / 1×1	$432 \times 200 \times 32$	544
MFM2a	–	$432 \times 200 \times 16$	–
Conv2b	3×3 / 1×1	$432 \times 200 \times 48$	7.0K
MFM2b	–	$432 \times 200 \times 24$	–
MaxPool2	2×2 / 2×2	$216 \times 100 \times 24$	–
Conv3a	1×1 / 1×1	$216 \times 100 \times 48$	1.2K
MFM3a	–	$216 \times 100 \times 32$	–
Conv3b	3×3 / 1×1	$216 \times 100 \times 64$	13.9K
MFM3b	–	$216 \times 100 \times 32$	–
MaxPool3	2×2 / 2×2	$108 \times 50 \times 32$	–
Conv4a	1×1 / 1×1	$108 \times 50 \times 64$	2.1K
MFM4a	–	$108 \times 50 \times 32$	–
Conv4b	3×3 / 1×1	$108 \times 50 \times 32$	9.2K
MFM4b	–	$108 \times 50 \times 16$	–
MaxPool4	2×2 / 2×2	$54 \times 25 \times 16$	–
Conv5a	1×1 / 1×1	$54 \times 25 \times 32$	544
MFM5a	–	$54 \times 25 \times 16$	–
Conv5b	3×3 / 1×1	$54 \times 25 \times 32$	4.6K
MFM5b	–	$54 \times 25 \times 16$	–
MaxPool5	2×2 / 2×2	$27 \times 12 \times 16$	–
FC6	–	32×2	332K
MFM6	–	32	–
FC7	–	2	64
Total	–	–	371K

Figura 4.4: Estructura de la red neuronal implementada en [21], que ha sido replicada para este trabajo. En total hay 371.000 parámetros entrenables.

Respecto a la función de activación de las capas, se usan en todas una activación MFM (*Max-Feature-Map*), mostrado en la figura 4.5, como se propuso originalmente en [21]. Este tipo de activación, suprime la mitad de las neuronas de cada capa mediante competición directa respecto a sus compañeras, desempeñando un papel de seleccionador de características.

Este tipo de arquitectura que usa MFM, se denomina LCNN *(Light-CNN)* [37], y permite un mejor rendimiento en este problema, respecto a usar otra función de activación como ReLU [21]. La función de activación MFM, no está dentro de las implementadas por Keras, por lo que ha sido necesario desarrollarla como función a medida, mediante una capa *lambda* en Keras.

Se ha usado la técnica *dropout* con diferentes valores, para reducir el *over-fitting*. Esta técnica consiste en, durante el entrenamiento, desconectar algunas neuronas y sus conexiones, con el fin de que las red no se adapte demasiado a los datos de entrenamiento. Es una técnica de regularización, que permite que las redes generalicen mejor en datos nuevos [31].

Algunos parámetros de esta red han sido modificados (como el tamaño del espectrograma de entrada y de algunas capas), para reducir el *over-fitting* e intentar conseguir un mejor rendimiento, como se detallará en el capítulo 5.

Una vez entrenada la red neuronal, se usa la datos de la última capa *fully-connected*, a modo

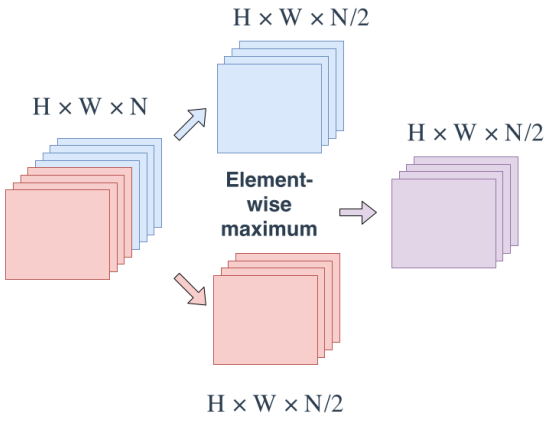

Figura 4.5: Esquema del funcionamiento de la función de activación *Max-Feature-Map*, donde se eliminan las activaciones de la mitad de las neuronas, reduciendo además, la dimensión de esa capa, obtenida de [21].

de *embedding* [24], para entrenar dos modelos de mezclas de gaussianas que, igual que en el apartado anterior, clasificar los datos, en vez de usar directamente la red para clasificar, con el fin de mejorar el rendimiento. Así, la red neuronal es utilizada para realizar una extracción o procesado de características a partir espectrograma, y se usa un sistema GMM para realizar la clasificación. Esta arquitectura es la misma que se utiliza en [21]. Hay intentos de replicar este sistema sin usar *embeddings*, es decir usando la red para clasificar directamente, como en [8], pero no se logra el mismo rendimiento.

4.2.3. Sistema 2 basado en DNN + SVM a partir de coeficientes CQCC y HFCC

En esta arquitectura, también basada en redes neuronales, se parte de características CQCC y HFCC. Los coeficientes CQCC, ya han sido descritos y detallados en la sección 4.2.1. Respecto a los coeficientes HFCC, propuestos en [26], extraidos siguiendo el esquema mostrado en la figura 4.6, son unos coeficientes cepstrales que enfocan su funcionamiento en las altas frecuencias de la señal. Según argumentan sus diseñadores, los dispositivos de grabación y reproducción diseñados para telefonía, con los cuales se puede realizar fácilmente un ataque de reproducción, pueden exhibir artefactos de canal en forma de atenuación o énfasis de determinadas frecuencias fuera de la banda de la voz (entre 300 y 3400 Hz). Estos artefactos acústicos podrían ser más pronunciados en dispositivos de baja calidad.

Para ello, los autores proponen realizar un filtrado paso-alto, previamente a la descomposición cepstral. El filtro paso alto propuesto es un filtro *butterworth* de segundo orden, con una frecuencia de corte a 3500 Hz. A partir de la señal filtrada, se realiza la STFT, para llegar a una representación del espectro la señal enventanada. Por último se toma el logaritmo de su módulo y se escogen los coeficientes de mayor energía de la DCT (*Discrete Cosine Transform*), con lo que se eliminaría la parte de la señal filtrada al tener menos energía. Finalmente se calculan los coeficientes *delta* y de aceleración.

Ambos tipos de características (CQCC y HFCC) son usados en conjunto en este sistema, aprovechando su complementariedad [26]. Para poder usar ambas características a la vez, ha sido necesario alinearlas en el dominio del tiempo. Debido a que la estructura temporal de las características CQCC proviene de un re-muestreo y no directamente de un enventanado en tiempo, se ha calculado el tamaño de ventana y el solapamiento de los HFCC para que ambas

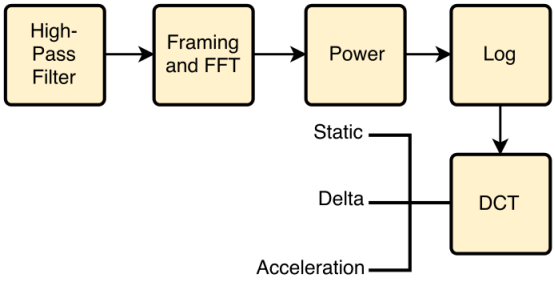

Figura 4.6: Esquema de la extracción de las características HFCC, según han sido diseñadas por [26].

características estén alineadas. Esto se ha conseguido con un tamaño de ventana de 25.5 ms y un solapamiento del 66.7 %.

A partir de este conjunto de caracterísiticas se construye un sistema de clasificación basado en una red neuronal y un SVM (*Support Vector Machine*). La red neuronal se comporta como extractor de características, y el SVM es el encargado de realizar la clasificación final, como se muestra en la figura 2.4.

La red neuronal formada por tres capas convolucionales y tres capas *fully-connected*, como queda reflejado en la figura 2.5. En esta red, las capas convolucionales desempeñan un papel de extracción de características, mientras que las capas *dense*, realizan una transformación del espacio consiguiendo mayor separabilidad de los datos [30].

En la capa de salida, se han probado para esta red, dos estrategias distintas. Por un lado, se ha usado una única neurona con función de activación *sigmoide*, para entrenar la red en hacer una clasificación binaria entre audios genuinos y ataques de reproducción. Por otro lado se ha llevado a cabo una estrategia de clasificación multi-clase, que distingue entre las diferentes condiciones de canal existentes. En el caso de la evaluación *ASVSpoof 2017*, en los datos de entrenamiento se proporcionaba la información de las condiciones de ataque de reproducción, (*Playback-Recording-Environment*).

El número de unidades en la capa de salida es igual al número de configuraciones únicas existentes en los datos de entrenamiento, utilizando una función de activación *softmax* en estas neuronas, una función que convierte un vector de números reales arbitrarios en un vector de valores en el rango de 0 a 1, que suma 1 como una distribución de probabilidad, teniendo a la salida de la red una probabilidad por cada clase [39].

La segunda estrategia tiene como objetivo capturar en las capas *fully-connected* anteriores de la salida, la información del canal, con el objetivo de usar esta información como *embedding* para luego realizar la clasificación en el SVM. Se ha probado empíricamente que esta estrategia funciona mejor y generaliza bien en condiciones de canal no vistas en los datos de test [26].

Las primeras tres capas convolucionales, tienen originalmente un tamaño de 128 filtros cada una, y las capas *fully-connected* un tamaño de 256 unidades, aplicando a estas últimas un *dropout* del 30 %, para evitar el *over-fitting*. El entrenamiento de la red ha sido realizado con diferentes tamaños de *batch* y número de *epochs*, buscando empíricamente el mejor rendimiento en los datos de desarrollo. Los *embeddings* se extraen de la últicа capa *fully-connected*, para realizar la clasificación en el SVM.

Un SVM (máquina de vector soporte), es un modelo que construye un hiperplano en un espacio, que puede ser de alta dimensionalidad, buscando separar de forma óptima los puntos de una clase de los de la otra, determinando el margen máximo entre los puntos [5]. Este hiperplano puede ser lineal, polinómico o tener otras funciones, definiéndose en la función *kernel* del SVM.

Se ha usado regularización, para permitir al SVM ajustarse más o menos a los datos, con la intención de reducir el *over-fitting* y permitir una mayor generalización.

En este problema, se ha usado un SVM de kernel lineal, buscando separar los datos con un hiperplano en la dimensión de la última capa *fully-connected*. El parámetro de regularización ha sido estimado empíricamente utilizando los datos de desarrollo.

El SVM ofrece puntuaciones de tipo blando, calculando la distancia de cada punto al hiperplano, cuanto mayor es esta distancia, más segura es la decisión de la clasificación de cada audio. La distancia al hiperplano puede ser positiva o negativa, dependiendo del lado en que está el punto, lo cual indica la clasificación calculada por el SVM.

4.3. Fusión de sub-sistemas

El objetivo final de este trabajo, es el de probar la complementariedad entre los sistemas de *audio fingerprinting* y los sistemas acústicos basados en reconocimiento de patrones.

Con este objetivo se han diseñado y desarrollado los sistemas anteriormente descritos. Para realizar la fusión, se ha elegido de cada categoría, el sistema con mejor rendimiento. Es posible que algún otro sistema o configuración del mismo, aun con menor rendimiento general, se obtenga una mayor complementariedad y mejores resultados en la fusión, pero en este trabajo se ha buscado utilizar los recursos disponibles en desarrollar dos sub-sistemas, que cada uno por su lado, tengan el mejor rendimiento posible, de cara a poder usarlo en investigaciones futuras.

La hipótesis de complementariedad, viene de las diferentes configuraciones posibles a la hora de realizar un ataque de reproducción. Los dispositivos que pueden ser usados y las estrategias de engaño al sistema son muy diversas y de muchos tipos, provocando que el audio resultante de un ataque de repetición pueda tener calidades muy dispares.

En aquellos ataques donde se usen dispositivos de grabación y reproducción de baja calidad, en presencia de ruido y graves atenuaciones, producirían audios que en el sistema de detección estarían muy degradados, con gran cantidad de ruido y artefactos acústicos. En este tipo de audios, un sistema de reconocimiento de patrones, como los descritos en la sección 4.2, tendría una mayor capacidad de detección, puesto que es con estas distorsiones con las que detecta el sistema si un audio proviene de un ataque de reproducción acústica.

Sin embargo, en estos mismos audios, un sistema de *audio fingerprinting*, tendría mayores dificultades, puesto que es más difícil recuperar los picos característicos de la señal en el espacio tiempo-frecuencial, debido a las distorsiones y al ruido, y la huella de audio podría diferir mucho entre el audio del ataque de reproducción y el guardado en la base de datos, aun siendo el mismo.

Por otro lado, también existe la posibilidad de que el ataque de reproducción sea realizado con dispositivos de alta fidelidad, en un escenario de poco ruido; o se puede dar incluso el acceso ilícito al sistema de autenticación y la captura los audios genuinos desde los medios de transmisión o almacenamiento del propio sistema. En este tipo de ataque, usando audios de gran calidad, los sistemas de reconocimiento de patrones tendrán una mayor dificultad, puesto que no existirán artefactos acústicos y distorsiones. No obstante, será en este tipo de audios donde el sistema de *audio fingerprinting* tendrá una mayor capacidad de detección, puesto que la huella acústica del audio del ataque de reproducción será muy similar a la del audio de la base de datos.

Para realizar la fusión de los sub-sistemas, se ha optado por una fusión a partir de una regresión logística lineal. Los pesos de la regresión logística son entrenados a partir de las puntuaciones de la mitad de la base de datos de evaluación y utilizados para realizar la fusión de la otra mitad y viceversa, una estrategia llamada *cross-validation*, como se muestra en la figura 4.7.

La razón por la que utiliza este esquema empleando los datos de evaluación para entrenar la fusión, en vez de desarrollo es porque en el sistema descrito en la sección 4.2.3, se utiliza el conjunto de desarrollo para su entrenamiento. Se ha empleado para el entrenamiento de la regresión logística el *toolbox* FoCal [6].

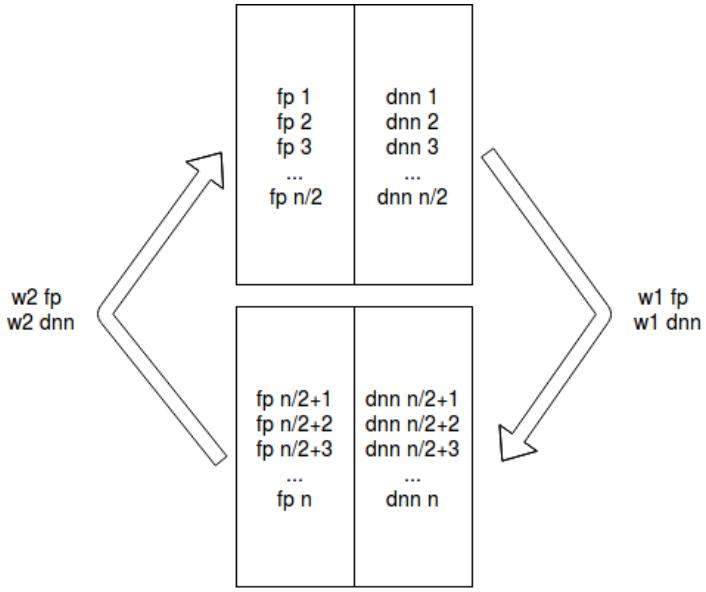

Figura 4.7: Estrategia de entrenamiento de los pesos de la regresión logística utilizada para la fusión de los sub-sistemas. Los pesos óptimos se calculan a partir de la mitad de la base de datos aplicándose en la fusión de la otra mitad de los datos, y viceversa.

5

Pruebas y resultados

En este capítulo se describirán las pruebas realizadas con los diferentes sistemas implementados y los resultados obtenidos en cada una de las pruebas. El objetivo de este trabajo es el de probar la hipótesis de complementariedad en la detección de ataques de reproducción de los sistemas acústicos basados en reconocimiento de patrones y la técnica del *audio fingerprinting*. En esta hipótesis, como se detalló en la sección 4.3, se espera que cada uno de las aproximaciones tenga un funcionamiento diferente en función de la calidad del audio del ataque de reproducción, exhibiendo el *audio fingerprinting* un mejor comportamiento en las grabaciones y reproducciones de alta fidelidad, mientras que sería más sencillo para los sistemas de reconocimiento de patrones trabajar en condiciones de baja calidad, debido a la creación de artefactos acústicos identificables.

Por un lado, se ha evaluado el sistema *audio fingerprinting*, midiendo su rendimiento en los diferentes conjuntos de la base de datos definidos por su calidad, descritos en el capítulo 3, utilizando las medidas de rendimiento definidas en ese mismo capítulo.

Por otro lado se han realizado pruebas de rendimiento en los diferentes sistemas implementados, comparándolos en base al sistema de referencia implementado. Además se ha analizado su rendimiento respecto a los sistemas publicados del estado del arte. Para ello se ha utilizado la medida de rendimiento EER.

Por último, se ha medido el rendimiento de los resultados de la fusión de ambos sistemas, comparándolo con el rendimiento de cada uno de los sistemas por separado, demostrando empíricamente la hipótesis de complementariedad.

5.1. Pruebas y resultados de *Audio Fingerprinting*

El sistema de *audio fingerprinting* se basa en que cada vez que se use el sistema de autenticación, se guarde el audio en una base de datos, que crece dinámicamente, con el fin de que cuando se realice un nuevo intento de acceso, se pueda comparar este nuevo audio con todos los intentos de accesos anteriores, para poder detectar un intento de acceso con la reproducción acústica de un audio anterior.

Para realizar las pruebas de rendimiento se ha construido una base de datos a partir de todos los audios de intentos de acceso genuinos disponibles. Un ataque, realizado a partir de la

grabación y reproducción de un intento de acceso legítimo, no puede suceder antes de que el propio audio genuino con el que se hace la reproducción, esté en la base de datos. Al estar la base de datos construida con todos los audios genuinos, cuando se utilice el sistema de *audio fingerprinting* para evaluar un audio genuino, no sería realista compararlo consigo mismo, puesto que ese audio todavía no debería de estar guardado en la base de datos dinámica del sistema. Por lo tanto, es necesario utilizar las etiquetas de test, para que, en caso de evaluar la puntuación de un audio genuino, el sistema devuelva la puntuación utilizando la base de datos completa pero eliminando ese audio en particular.

Con las puntuaciones del sistema por cada audio, se puede obtener el EER para cada los conjuntos de desarrollo, evaluación y los subconjuntos HQ1 y HQ2 (*High Quality*), de alta calidad, descritos en el capítulo 3, y sus conjuntos complementarios All-HQ1 y All-HQ2, que incluyen todos los audios, menos los de alta calidad.

En la siguiente tabla se muestran las figuras de error de este sistema en los diferentes conjuntos y subconjuntos de datos.

	DEV	EVAL			
EER (%)			All-HQ1	All-HQ2	HQ1	HQ2
Audio Fingerprinting	1,15	1,69	1,92	1,69	0,38	0,07

Tabla 5.1: Rendimiento del sistema *audio fingerprinting* en los diferentes conjuntos de datos.

En estos resultados se puede comprobar que el sistema de *audio fingerprinting* tiene un muy buen rendimiento en la tarea de la detección de ataques de reproducción consiguiéndose unos error muy bajo. Además se comprueba que cuanto más alta es la calidad del ataque (en los conjuntos de alta calidad HQ1 y HQ2), más capacidad de detección tiene el sistema.

5.2. Resultados sistemas acústicos

En los siguientes apartados se describirán los resultados obtenidos con los diferentes sistemas. Todos ellos han sido medidos en EER, midiendo su rendimiento en los diferentes conjuntos de datos.

5.2.1. Sistema *baseline* basado en GMM

El *baseline* utilizado en este trabajo, cuyo rendimiento se utilizará como referencia, está basado en modelos de mezclas de gaussianas (GMM), tal y como se describe en el capítulo 4. Se han realizado los experimentos previamente descritos obteniendo los siguientes resultados:

	DEV	EVAL			
EER (%)			All-HQ1	All-HQ2	HQ1	HQ2
GMM *baseline*	20,94	24,39	23,26	24,25	29,21	28,53

Tabla 5.2: Rendimiento obtenido con el sistema *baseline* basado en Modelos de Mezclas de Gaussianas.

Este sistema, optimizado para la base de datos de desarrollo, generaliza bastante bien sobre el conjunto de evaluación, aumentando en este conjunto el error un 16 % respecto al conjunto de desarrollo. Se comprueba como este sistema en los conjuntos de mayor calidad tiene más problemas para reconocer patrones acústicos, empeorándose considerablemente el rendimiento.

5.2.2. Sistema 1 basado en CNN + GMM a partir de espectrograma

En este sistema, desarrollado a partir del utilizado en [21]. Este último tiene unos rendimientos muy buenos, como se detallan en la tabla inferior, mucho mejores que los conseguidos en estos experimentos, con los que se obtienen con esta red, lo que parece ser un *overfitting* muy pronunciado, al tener un gran número de parámetros entrenables, en comparación con la cantidad de datos de entrenamiento disponibles. En el sistema publicado en [21], se da información sobre la arquitectura de la red, pero no ofrecen todos los hiperparámetros que han utilizado, los cuales han sido intentados de encontrar empíricamente sin éxito. En la tabla inferior se muestran los resultados obtenidos con la mejor de las redes desarrolladas en este trabajo, comparándose con el error del sistema original.

	DEV	EVAL				
EER (%)			All-HQ1	All-HQ2	HQ1	HQ2
Sistema 1 CNN	9,68	30,43	32,35	30,58	19,97	23,26
Sistema original	4,53	7,37	-	-	-	-

Tabla 5.3: Rendimiento obtenido con el sistema 1 basado en CNN, y del sistema original al partir del cual fue diseñado [21], cuyo rendimiento es muy superior.

En estos resultados se comprueba como este sistema no generaliza bien a nuevos datos, consiguiendo un buen rendimiento en el conjunto de desarrollo respecto al que ha sido optimizado, pero no pudiendo discriminar correctamente en datos con otra distribución, como es el conjunto de evaluación. Mientras tanto, el sistema original, sí consigue clasificar de forma correcta los datos de evaluación.

Una resultado sorprende en este sistema es el hecho de que funciona mejor en los subconjuntos de alta calidad que en el resto de la base de datos, pese a que en estos audios las imperfecciones acústicas son menores y más complicadas de detectar. Una razón posible para explicar este comportamiento es que en la base de datos de desarrollo, los audios tienen una mejor calidad, más parecida a la de los subconjuntos HQ1 y HQ2, y por ello la red neuronal consigue a discriminar alguna característica que tienen en común estos audios; fallando sin embargo en los audios de baja calidad.

5.2.3. Sistema 2 basado en DNN + SVM a partir de coeficientes CQCC y HFCC

Este sistema, basada en el desarrollado por [26], obtiene sus mejores resultados entrenando la red para discriminar entre las diferentes condiciones de canal. Como queda detallado en el capítulo 3, en el conjunto de entrenamiento existen sólo 4 condiciones únicas, mientras que en el conjunto de desarrollo hay 9 configuraciones. Para conseguir los mejores resultados siguiendo esta estrategia, se ha entrenado la red utilizando tanto el conjunto de entrenamiento como el de desarrollo, que además de tener más variabilidad de canal, se consiguen más datos para entrenar la red. Ambas redes tienen los mismos hiperparámetros, salvo el número de unidades en la capa de salida, que es mayor en la red que utiliza los datos de desarrollo, al existir en estos un mayor número de condiciones de canal. Se ha usado una estrategia de *cross-validation* para optimizar el número de épocas de entrenamiento.

En la siguiente tabla se muestran los resultados obtenidos con cada una de las estrategias, y de los sistemas originales a partir de los cuales fueron diseñados. De estos resultados se obtiene que el sistema con mejor rendimiento, en EER, es el sistema 2 entrenado con ambos conjuntos datos, de entrenamiento y desarrollo, por ello, es la puntuación de este sistema el que se ha utilizado para realizar las pruebas de fusión junto con la estrategia del *audio fingerprinting*.

	DEV	EVAL				
EER (%)			All-HQ1	All-HQ2	HQ1	HQ2
Sistema 2 DNN+SVM	18,42	31,35	31,35	31,24	31,13	34,34
Sistema 2 DNN+SVM añadiendo conjunto Dev	-	19,80	19,59	19,56	20,83	27,42
Sistema original	7.6	11.5	-	-	-	-

Tabla 5.4: Rendimiento del sistema 2 DNN+SVM con y sin utilizar el conjunto de desarrollo durante el entrenamiento, comparándose con el sistema original a partir del cual fue diseñado [26].

El sistema original consigue un mejor rendimiento que cualquiera de las estrategias implementadas. Añadiendo los datos de desarrollo al entrenamiento de la red, se consigue una enorme mejora, probablemente gracias a la mayor variabilidad que tienen estos datos respecto a la información de canal, que es lo que se intenta extraer en los *embeddings* de la red. Se comprueba que en los datos de muy alta calidad HQ2, se obtiene peor rendimiento, mientras que esta bajada de rendimiento no es tan grave en el subconjunto HQ1.

5.3. Resultados fusión de sistemas

En este apartado se detallará el rendimiento obtenido por la fusión de los sistemas, en los diferentes conjuntos de datos disponibles. Como se describió en el apartado 4.3, la fusión ha sido realizada entrenando una regresión logística lineal con una estrategia de *cross-validation*.

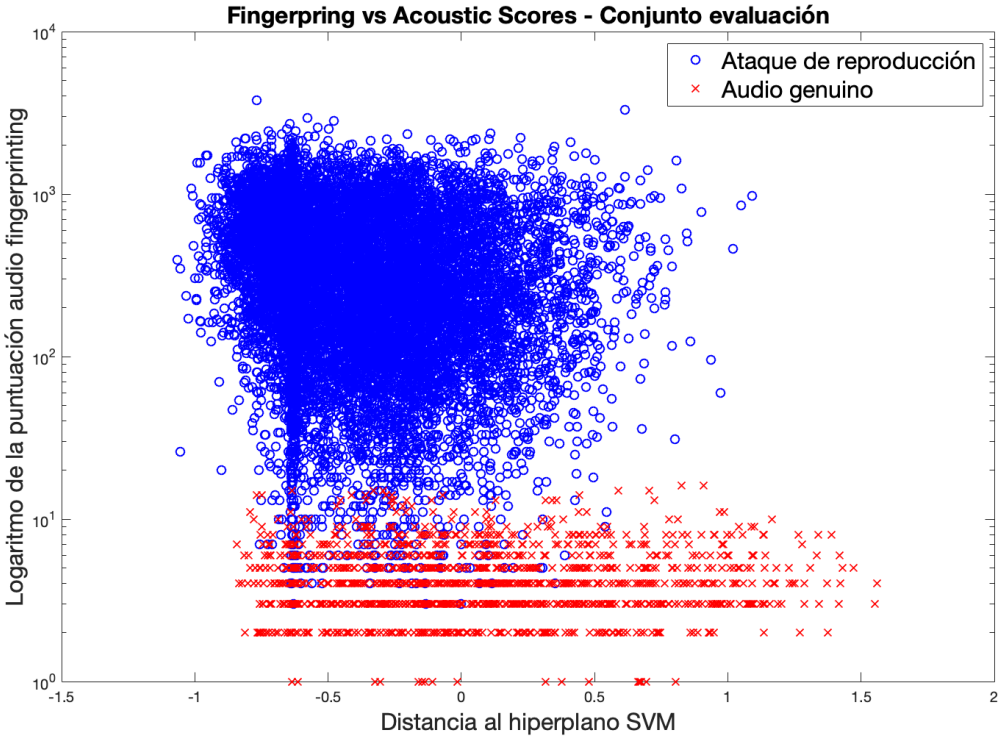

Figura 5.1: Cada audio del conjunto de evaluación se muestra a partir de la puntuación del sistema acústico enfrentada a la puntuación del sistema *audio fingerprinting*

Las puntuaciones de ambos clasificadores en el conjunto total de validación se muestran

enfrentadas en la figura 5.1. Se puede comprobar en la figura el alto grado de discriminación que consigue el sistema de *audio fingerprinting*. Además se puede intuir que el sistema acústico puede ayudar en discriminar los audios en los que se confunde el *audio fingerprinting*, aquellos de baja calidad donde la huella de audio del ataque de reproducción pierde la similitud al audio genuino del que proviene, produciendo una puntuación baja.

En la siguiente tabla se muestra el rendimiento obtenido en la fusión de los sistemas, junto con el de cada sub-sistema por separado. Además se añade el rendimiento del sistema de referencia basado en GMM para realizar la comparativa de los resultados.

	DEV	EVAL			
EER (%)			All-HQ1	All-HQ2	HQ1	HQ2
GMM *baseline*	20,94	24,39	23,26	24,25	29,21	28,53
DNN_SVM añadiendo conjunto Dev	-	19,80	19,59	19,56	20,83	27,42
Audio Fingerprinting	1,15	1,69	1,92	1,69	0,38	0,07
Fusión sub-sistemas DNN_SVM+*Audio Fingerprinting*	-	1,34	1,50	1,37	0,40	0

Tabla 5.5: Rendimiento de la fusión de sub-sistemas, comparado con el rendimiento de sistema individual y el sistema de referencia GMM.

En estos resultados se puede comprobar que ambos sistemas se complementan, y consiguen en la fusión un error menor. En el conjunto de evaluación, se consigue una mejora del 26 % en la fusión respecto al rendimiento obtenido con el *audio fingerprinting*, mejorándose del 1.69 % al 1.34 % en EER. Estos resultados demuestran la hipótesis de partida de complementariedad entre los dos tipos de sistemas. El sistema acústico permite mejorar la capacidad de clasificación del *audio fingerprinting* en los audios de baja calidad, donde el efecto de las distorsiones acústicas y el ruido pueden provocar que la huella de audio del ataque de reproducción se vea modificada.

6
Conclusiones

Los resultados obtenidos mediante la fusión de los sistemas desarrollados en este trabajo, demuestran la capacidad de combinación que tienen las dos estrategias de detección de ataques de reproducción acústica implementadas en este trabajo.

Por un lado, el sistema *audio fingerprinting*, desarrollado a partir de un sistema similar y adaptado a esta tarea; consigue unos resultados muy buenos, con una gran capacidad de detección de ataques de reproducción, con la desventaja de necesitar mantener una base de datos que crece dinámicamente con el uso, donde guardar la huellas de audio de los intentos de acceso al sistema.

Por otro lado, se han desarrollado varios sistemas acústicos de reconocimiento de patrones, diseñados a partir de los sistemas de mejor rendimiento de la evaluación *ASVSpoof 2017*, obteniéndose peores rendimientos que en los sistemas originales debido a la dificultad del entrenamiento de las redes neuronales en las que están basados; pero suficiente para ser utilizados junto al sistema de *audio fingerprinting* para comprobar la hipótesis de complementariedad.

Esta hipótesis reside en el hecho de que los ataques de reproducción acústica pueden ser realizados por varios dispositivos y en diversas condiciones acústicas. Esto conlleva que el audio fruto de la reproducción pueda tener diferentes calidades. En los audios de baja calidad, los sistemas acústicos, en general, tienen mayor capacidad de discriminación, debido a que esta baja calidad se debe a distorsiones, filtrados, ruido y otros artefactos acústicos, que son los patrones que el sistema es capaz de reconocer; mientras que el sistema de *audio fingerprinting* puede tener dificultades debido a que la huella de audio se ve modificada por la reproducción y se asemeja menos a la original. Sin embargo, en los audios de alta calidad, donde incluso los sistemas acústicos del estado del arte de tienen dificultades de detección, el sistema de *audio fingerprinting* tiene un error casi despreciable, debido a que la reproducción es muy similar al audio original.

Esta hipótesis se comprueba empíricamente en los resultados individuales de cada sistema, consiguiéndose en el sistema de *audio fingerprinting* mejor rendimiento en los subconjuntos de alta calidad de la base de datos de evaluación, mientras que los sistemas acústicos, en general, tienen peores resultados en esos subconjuntos.

En la fusión de ambos sistemas se consigue una mejora del 26 % respecto a la mejor puntuación de los sistemas individuales, demostrando la complementariedad de ambas estrategias.

Glosario de acrónimos

- **DNN**: Deep Neural Networks,
- **ACR**: Automatic Content Recogniticon
- **STFT**: Short Time Fourier Transform
- **GMM**: Gaussian Mixture Models
- **EM**: Expectation-Maximization
- **MAP**: Maximum A Posteriori
- **ReLU**: Rectified Linear Unit
- **MFCC**: Mel-Frequency Cepstral Coefficients
- **DCT**: Discrete Cosine Transform
- **EER**: Equal Error Rate
- **CQCC**: Constant-Q Cepstral Coefficient
- **HFCC**: High-Frequency Cepstral Coefficient
- **CQT**: Constant-Q Transform
- **SVM**: Support Vector Machine
- **ASV**: Automatic Speaker Verification
- **HQ**: High Quality
- **FAR**: False Acceptance Rate
- **FRR**: False Rejection Rate
- **ASR**: Automatic Speaker Recognition
- **MFM**: Max Feature Map
- **CNN**: Convolutional Neural Network
- **LCNN**: Light Convolutional Neural Network
- **FP**: Fingerprint

Bibliografía

[1] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dan Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. Software available from tensorflow.org.

[2] K. N. R. K. Raju Alluri, Sivanand Achanta, Sudarsana Reddy Kadiri, Suryakanth V. Gangashetty, and Anil Kumar Vuppala. Sff anti-spoofer: Iiit-h submission for automatic speaker verification spoofing and countermeasures challenge 2017. In *INTERSPEECH*, 2017.

[3] Frédéric Bimbot, Jean-François Bonastre, Corinne Fredouille, Guillaume Gravier, Ivan Magrin-Chagnolleau, Sylvain Meignier, Teva Merlin, Javier Ortega-García, Dijana Petrovska-Delacrétaz, and Douglas A. Reynolds. A tutorial on text-independent speaker verification. *EURASIP Journal on Advances in Signal Processing*, 2004(4):101962, Apr 2004.

[4] Christopher M. Bishop. Pattern recognition and machine learning (information science and statistics). Berlin, Heidelberg, 2006. Springer-Verlag.

[5] Bernhard E. Boser, Isabelle M. Guyon, and Vladimir N. Vapnik. A training algorithm for optimal margin classifiers. In *Proceedings of the Fifth Annual Workshop on Computational Learning Theory*, COLT '92, pages 144–152, New York, NY, USA, 1992. ACM.

[6] N Brümmer. Focal: Toolkit for fusion and calibration.

[7] Zhuxin Chen, Zhifeng Xie, Weibin Zhang, and Xiangmin Xu. Resnet and model fusion for automatic spoofing detection. In *INTERSPEECH*, 2017.

[8] Bhusan Chettri, Saumitra Mishra, Bob L Sturm, and Emmanouil Benetos. A study on convolutional neural network based end-to-end replay anti-spoofing. *arXiv preprint arXiv:1805.09164*, 2018.

[9] François Chollet et al. Keras. `https://keras.io`, 2015.

[10] Avery Li chun Wang and Th Floor Block F. An industrial-strength audio search algorithm. In *Proceedings of the 4 th International Conference on Music Information Retrieval*, 2003.

[11] Vitabile S. Conti V, Militello C. Biometric authentication overview: a fingerprint recognition sensor description. In *Int J Biosen Bioelectron*, pages 26–31, 2017.

[12] Hector Delgado, Massimiliano Todisco, Md Sahidullah, Nicholas Evans, Tomi Kinnunen, Kong Aik Lee, and Junichi Yamagishi. Asvspoof 2017 version 2.0: meta-data analysis and baseline enhancements. In *Speaker Odyssey 2018*, pages 296–303, 6 2018.

[13] A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum likelihood from incomplete data via the em algorithm. *JOURNAL OF THE ROYAL STATISTICAL SOCIETY, SERIES B*, 39(1):1–38, 1977.

[14] Joaquin Gonzalez-Rodriguez, Alvaro Escudero, Diego de Benito-Gorrón, Beltran Labrador, and Javier Franco-Pedroso. An audio fingerprinting approach to replay attack detection on asvspoof 2017 challenge data. In *Proc. Odyssey 2018 The Speaker and Language Recognition Workshop*, pages 304–311, 2018.

[15] Jaap Haitsma. A highly robust audio fingerprinting system. pages 107–115, 2002.

[16] Marko Helen and Tommi Lahti. Query by example methods for audio signals. pages 302 – 305, 07 2006.

[17] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980, 2014.

[18] T. Kinnunen, M. Sahidullah, M. Falcone, L. Costantini, R. G. Hautamäki, D. Thomsen, A. Sarkar, Z. Tan, H. Delgado, M. Todisco, N. Evans, V. Hautamäki, and K. A. Lee. Reddots replayed: A new replay spoofing attack corpus for text-dependent speaker verification research. In *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 5395–5399, March 2017.

[19] Tomi Kinnunen, Md Sahidullah, Héctor Delgado, Massimiliano Todisco, Nicholas Evans, Junichi Yamagishi, and Kong Aik Lee. The ASVspoof 2017 challenge: Assessing the limits of replay spoofing attack detection. In *INTERSPEECH 2017, Annual Conference of the International Speech Communication Association, August 20-24, 2017, Stockholm, Sweden*, Stockholm, SWEDEN, 08 2017.

[20] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. Imagenet classification with deep convolutional neural networks. In *Proceedings of the 25th International Conference on Neural Information Processing Systems - Volume 1*, NIPS'12, pages 1097–1105, USA, 2012. Curran Associates Inc.

[21] Galina Lavrentyeva, Sergey Novoselov, Egor Malykh, Alexander Kozlov, Oleg Kudashev, and Vadim Shchemelinin. Audio replay attack detection with deep learning frameworks. In *INTERSPEECH*, 2017.

[22] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. volume 86, pages 2278–2324, Nov 1998.

[23] Kong-Aik Lee, Anthony Larcher, Guangsen Wang, Patrick Kenny, Niko Brümmer, David A. van Leeuwen, Hagai Aronowitz, Marcel Kockmann, Carlos Vaquero, Bin Ma, Haizhou Li, Themos Stafylakis, Md. Jahangir Alam, Albert Swart, and Javier Perez. The reddots data collection for speaker recognition. In *INTERSPEECH*, 2015.

[24] Alicia Lozano Díez. Bottleneck and embedding representation of speech for dnn-based language and speaker recognition. 2018.

[25] Warren S. McCulloch and Walter Pitts. A logical calculus of the ideas immanent in nervous activity. volume 5, pages 115–133, Dec 1943.

[26] Parav Nagarsheth, Elie Khoury, Kailash Patil, and Matt Garland. Replay attack detection using dnn for channel discrimination. In *INTERSPEECH*, 2017.

[27] A. V. Oppenheim. Speech spectrograms using the fast fourier transform. volume 7, pages 57–62, Aug 1970.

[28] Douglas A. Reynolds. Gaussian mixture models. In *Encyclopedia of Biometrics*, 2009.

[29] D. E. Rumelhart, G. E. Hinton, and R. J. Williams. Parallel distributed processing: Explorations in the microstructure of cognition, vol. 1. pages 318–362, Cambridge, MA, USA, 1986. MIT Press.

[30] T. N. Sainath, O. Vinyals, A. Senior, and H. Sak. Convolutional, long short-term memory, fully connected deep neural networks. In *2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 4580–4584, April 2015.

[31] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15:1929–1958, 2014.

[32] Massimiliano Todisco, Héctor Delgado, and Nicholas W. D. Evans. A new feature for automatic speaker verification anti-spoofing: Constant q cepstral coefficients. In *Odyssey*, 2016.

[33] Massimiliano Todisco, Héctor Delgado, and Nicholas Evans. A new feature for automatic speaker verification anti-spoofing: Constant q cepstral coefficients. 06 2016.

[34] Giacomo Valenti, Héctor Delgado, Massimiliano Todisco, Nicholas Evans, and Laurent Pilati. An end-to-end spoofing countermeasure for automatic speaker verification using evolving recurrent neural networks. 2018.

[35] Bernard Widrow and Marcian E. Hoff. Neurocomputing: Foundations of research. pages 123–134, Cambridge, MA, USA, 1988. MIT Press.

[36] Marcin Witkowski, Stanislaw Kacprzak, Piotr Zelasko, Konrad Kowalczyk, and Jakub Galka. Audio replay attack detection using high-frequency features. In *INTERSPEECH*, 2017.

[37] Xiang Wu, Ran He, and Zhenan Sun. A lightened CNN for deep face representation. *CoRR*, abs/1511.02683, 2015.

[38] J. Youngberg and S. Boll. Constant-q signal analysis and synthesis. In *ICASSP '78. IEEE International Conference on Acoustics, Speech, and Signal Processing*, volume 3, pages 375–378, April 1978.

[39] Rubén Zazo Candil et al. Exploiting temporal context in speech technologies using lstm recurrent neural networks.