Arquitecturas de Redes Neuronales Profundas para mejora de voz: De los perceptrones multicapa a las redes completamente convolucionales (FCNs)

Laura Gámiz Pérez

Máster en Ingeniería de Telecomunicación

MÁSTERES
DE LA UAM
2018 - 2019

Escuela Politécnica Superior

UAM Universidad Autónoma de Madrid

UNIVERSIDAD AUTÓNOMA DE MADRID
ESCUELA POLITÉCNICA SUPERIOR

TRABAJO FIN DE MÁSTER

Arquitecturas de Redes Neuronales Profundas para mejora de voz: De los perceptrones multicapa a las redes completamente convolucionales (FCNs)

Máster Universitario en
Ingeniería de Telecomunicación

Autora: GÁMIZ PÉREZ, Laura

Tutor: TORRE TOLEDANO, Doroteo
Dpto. de Tecnología Electrónica de las Comunicaciones

SEPTIEMBRE 2019

Arquitecturas de Redes Neuronales Profundas para mejora de voz: De los perceptrones multicapa a las redes completamente convolucionales (FCNs)

Autora: GÁMIZ PÉREZ, Laura

Tutor: TORRE TOLEDANO, Doroteo

Grupo Audias - Audio, Data Intelligence And Speech
Dpto. Tecnología Electrónica y de las Comunicaciones
Escuela Politécnica Superior
Universidad Autónoma de Madrid
Septiembre de 2019

Arquitecturas de Redes Neuronales Profundas para mejora de voz: De los perceptrones multicapa a las redes completamente convolucionales (FCNs)

Laura Gámiz Pérez

Palabras clave: Mejora de voz, Redes neuronales, Aprendizaje profundo, Capa convolucional, Procesamiento de voz.

Resumen

En este proyecto se analizan e implementan distintos sistemas de reducción de ruido basados en redes neuronales profundas. La mejora del habla tiene como objetivo mejorar la inteligibilidad y la calidad perceptiva de la señal de voz degradada que forma parte de numerosas aplicaciones.

Recientemente, han sido propuestas algunas técnicas novedosas de aprendizaje profundo para dicho propósito. Sin embargo, los modelos descritos en estos sistemas aún presentan algunas deficiencias que impiden una correcta caracterización de la voz y una mejora eficiente de las señales.

Por este motivo, se proponen dos nuevos sistemas que usan arquitecturas de redes neuronales convolucionales (CNNs y FCNs), que han demostrado ser más adecuadas para modelar este tipo de señales. Diferentes configuraciones han sido comparadas en términos de PESQ y STOI, dos medidas de rendimiento ampliamente utilizadas para evaluar la calidad de la voz. Además, la complejidad de los modelos y el tiempo de cómputo de los sistemas han sido dos factores clave en la elección de los mismos. Como línea base en esta comparación se utilizará un sistema anterior que usa una red FC-DNN para mejorar la voz.

La base de datos ruidosa sobre la que se han aplicado estos sistemas ha sido generada mediante el software de adicción de ruido FANT, combinando la base de datos de voz limpia ALBAYZIN y la base de datos de ruidos HU. El software del proyecto se ha desarrollado e implementado en Python, haciendo uso de la librería Keras para la construcción de los modelos de redes neuronales.

Deep Neural Network Architectures for speech enhancement: From multilayer perceptrons to fully convolutional networks (FCNs)

Laura Gámiz Pérez

Keywords: Speech enhancement, Neuronal Networks, Deep learning, Convolutional layer, Speech processing

Abstract

Several noise reduction systems based on deep neural networks have been analysed and implemented within this project. Speech enhancement aims to improve both intelligibility and perceptive quality of the degraded voice signal that is part of a large number of applications.

Recently, some newflanged deep learning techniques have been proposed for that purpose. However, some inefficiencies still remain within the models described in these systems, preventing both an accurate voice characterization and an efficient signal improvement.

In this context, two novel systems that incorporate convolutional neural networks architectures (CNNs and FCNs) are presented. These architectures have proved to be more accurate in modelling these particular signals. Several configurations have been challenged in terms of PESQ and STOI: two performance measures widely used in voice quality assessment. Moreover, models complexity and computational time have played a key role in the model selection process. A previous system that used a FC-DNN in speech enhancement have been taken as a benchmark.

These systems have been applied to a noisy database generated by the noise addition software FANT, that combines the clean voice database ALBAYZIN and the noises database HU. Project's software has been developed and implemented in Python, using Keras library in the neural networks models building process.

Agradecimientos

A mi familia, amigas y amigos. En especial a mi hermana, por su ayuda y apoyo incondicional. Gracias Sara por abrirme las puertas del camino más bonito y por hacérmelo más fácil.

A mi tutor. Gracias Doroteo por ofrecerme la oportunidad de trabajar contigo y aprender de ello. Gracias por haberme ofrecido tu ayuda cuando la he necesitado, por tu paciencia infinita y por los consejos que me has dado.

Al Grupo AUDIAS, por confiar en mi, por su interés. Gracias por la oportunidad que me ofrecisteis desde el primer momento. Gracias por vuestra inestimable ayuda.

A mis compañeros de Máster, porque sin vuestra ayuda, seguramente, ahora no estaría escribiendo estas líneas. Por el día a día, por los buenos ratos y por los no tan buenos, por haber estado ahí y haberme acompañado durante todo este tiempo.

Índice general

Índice de figuras

Capítulo 1

Introducción

En este capítulo se presenta la motivación que ha impulsado la realización de este Trabajo Fin de Máster, así como los objetivos establecidos en el desarrollo del proyecto. Además, se muestra la estructura de la memoria, con el fin de facilitar la lectura y comprensión de la misma.

1.1. Motivación

Gran parte de las señales de voz de nuestro entorno presentan ruido aleatorio que dificulta en mayor o menor medida la comprensión e inteligibilidad de las mismas. Pero, reducir el ruido presente en este tipo de señales no es una tarea fácil, pues existe una amplia variedad de fuentes que lo provocan y puede ser el resultado de la combinación de varios factores. La mejora de voz, comúnmente conocido por sus siglas en inglés SE (*Speech Enhancement*), ha sido objeto de numerosas investigaciones a lo largo de la historia. Asimismo, sigue siendo de gran interés en la actualidad debido al elevado número de aplicaciones de las que forman parte este tipo de señales: comunicaciones móviles, codificación de voz, reconocimiento automático del habla, etc.

Entre las aproximaciones tradicionales para mejora de voz se incluyen la sustracción espectral, el filtrado de Wiener [1], el estimador de amplitud espectral basado en el error cuadrático medio mínimo (MMSE), etc. En los últimos años se han desarrollado métodos de aprendizaje automático basados en redes neuronales profundas [2] y se ha impulsado el uso de éstos en mejora de voz. En este ámbito, la mayoría de estos modelos de reducción de ruido se centran solo en el procesamiento del espectrograma de magnitud (por ejemplo, los espectros del logaritmo de potencia (LPS)) y dejan la fase en su forma ruidosa original, ya que no existe una estructura clara en un espectrograma de fase, por lo que estimar con precisión las fases limpias a partir de las ruidosas es difícil. En este ámbito encontramos las primeras aproximaciones de sistema de mejora de voz basados en redes neuronales profundas del tipo perceptrón multicapa [13].

A pesar de que se ha demostrado que superan los enfoques de mejora convencionales, todavía hay espacio para mejoras adicionales [19] [20]. Así pues, se proponen dos nuevas

alternativas basadas en redes neuronales que pretenden mejorar el rendimiento de los sistemas propuestos usando perceptrones multicapa. En este contexto se sitúa este Trabajo Fin de Máster, con el que se pretende encontrar técnicas y modelos novedosos que reduzcan el ruido presente en señales de voz.

Como primera mejora se propone el uso de redes neuronales convolucionales (*Convolutional Neural Network*, CNN), que permitan modelar eficientemente las estructuras locales del espectro temporal de un espectrograma y mejorar aún más el rendimiento.

Sin embargo, en la aproximación anterior aún es necesario mapear características a través de la Transformada de Fourier entre los dominios de tiempo y frecuencia para el análisis y la resíntesis. Además, estudios recientes han revelado la importancia de la fase cuando los espectrogramas se resintetizan nuevamente en formas de onda en el dominio del tiempo. Por este motivo, se propone el estudio de modelos basados en el aprendizaje profundo con entradas en su forma de onda original, sin procesar, teniendo en cuenta de esta forma la información de la fase. Por otro lado, se ha podido comprobar que el uso de redes neuronales con capas completamente conectadas (*fully connected*), pueden no preservar bien la información local para generar componentes de alta frecuencia y agregar un efecto negativo al proceso de aprendizaje. Por lo tanto, como segunda mejora, se estudiarán modelos de redes completamente convolucionales (*Fully Convolutional Networks*, FCN) para permitir que cada muestra de salida dependa localmente de las regiones de entrada vecinas.

Para la realización de experimentos se cuenta con una Base de Datos de voz limpia, ALBAYZIN, y una Base de Datos de ruidos, HU. Estas bases de datos serán combinadas para generar los datos de entrada a las redes, los cuales se emplearán para el entrenamiento de las mismas, y para generar los datos de evaluación de las redes mencionadas.

Con este Trabajo Fin de Máster se emplearán los conocimientos adquiridos en diferentes asignaturas del Máster de Ingeniería de Telecomunicación. Para el tratamiento y comprensión de las señales de voz se aplicará la base técnica adquirida en Tecnologías del Habla. Además, en relación a las técnicas de inteligencia artificial empleadas, se utilizarán los fundamentos aprendidos en Procesado Avanzado de Señales Multimedia. Por último, de forma más amplia, se aplicarán los procedimientos estudiados en Proyectos en Ingeniería de Telecomunicación.

1.2. Objetivos

El objetivo principal de este Trabajo Fin de Máster es analizar y estudiar diferentes sistemas de mejora de voz basados en diferentes arquitecturas de redes neuronales profundas. Se pretende mejorar los resultados obtenidos con el sistema *baseline* basado en una red tradicional del tipo perceptrón multicapa. Para ello, se han definido los siguientes objetivos parciales:

- Búsqueda y análisis de posibles vías de mejora del modelo *baseline*, a través del estudio de las limitaciones que presenta dicho sistema en el procesado y aprendizaje de señales de voz.
- Diseño e implementación de diferentes sistemas de mejora de voz, mediante la utilización de redes neuronales convolucionales (CNN y FCN); y comparación entre los mismos. Se pretende obtener modelos que mejoren el rendimiento de los sistemas actuales, en términos de tiempo de ejecución y valoración de la intelegibilidad del habla.

1.3. Planificación del proyecto

Para conseguir los objetivos propuestos, se dividirá el plan de trabajo en 4 fases, las cuales incluirán diferentes tareas. En la primera fase se realizará un estudio teórico de las técnicas y conceptos con los que se va a trabajar. La segunda fase consistirá en el diseño e implementación del sistema *baseline* y de nuevos sistemas de mejora de voz basados en redes neuronales. En la tercera fase se evaluarán los sistemas implementados. La última fase será destinada a la redacción de los procedimientos y del trabajo realizado en la memoria final. La secuencialidad de las tareas incluidas en cada fase y la descripción de las mismas aparece a continuación.

Plan de trabajo

- Estudio y documentación sobre técnicas avanzadas de aprendizaje automático, mostrando especial atención en redes neuronales.
- Preparación del entorno de trabajo: instalación de las herramientas y librerías en Python para la generación de modelos Deep Learning (Keras, TensorFlow, Theano).
- Estudio del estado del arte en mejora de voz. Análisis de trabajos previos y de las limitaciones que presentan.

Diseño e implementación de nuevos sistemas de mejora de voz basados en redes neuronales

- Formulación del problema y análisis detallado de las características de las señales de voz, que permita comprender en profundidad el problema descrito.
- Obtención de un pre-procesado óptimo para los conjuntos de entrenamiento de entrada a las redes neuronales; o análisis de los beneficios que supone prescindir del mismo.
- Generación de los conjuntos de entrenamiento de entrada a las redes neuronales.
- Diseño e implementación de sistemas de mejora de voz basados en redes neuronales CNNs.

- Diseño e implementación de sistemas de mejora de voz basados en redes neuronales FCNs.

Evaluación de los sistemas y análisis de resultados

- Generación de las bases de datos para test.
- Diseño e implementación de los sistemas de evaluación basados en redes neuronales.
- Realización de pruebas de evaluación y rendimiento de los modelos implementados.

Redacción de la memoria

- Elaboración del informe final del Trabajo Fin de Máster, en el cual se documente los procedimientos llevados a cabo y los resultados obtenidos. Presentación del proyecto.

1.4. Organización de la memoria

La presente memoria recoge el proceso de investigación y desarrollo realizado para la elaboración de este Trabajo Fin de Máster. Para ello, el informe ha sido estructurado en seis capítulos.

Capítulo 1: Introducción. Este capítulo recoge la motivación que ha dado impulso al desarrollo del Trabajo, los objetivos fijados, así como la planificación llevada a cabo y la estructura de la memoria.

Capítulo 2: Estado del Arte. En este capítulo se expone un breve estado del arte sobre las técnicas de mejora de voz y se describe el fundamento teórico de Redes Neuronales. El objetivo de este capítulo es familiarizar al lector con la terminología empleada en el resto de la memoria.

Capítulo3: Descripción de los sistemas. Presenta los motivos que han impulsado el desarrollo de sistemas novedosos para mejora de voz e investigación en este campo. Además, se describe detalladamente cada uno de los sistemas implementados.

Capítulo 4: Entorno experimental. Se describen las bases de datos utilizadas en los experimentos, así como las medidas de rendimiento utilizadas para la valoración de la eficiencia de los resultados propuestos. Además, se detallan las herramientas tecnológicas utilizadas en el desarrollo del Trabajo.

Capítulo 5: Resultados. Presenta los resultados obtenidos con los sistemas desarrollados y expone una valoración, comparación y discusión sobre los mismos.

Capítulo 6: Conclusiones y vías futuras. Se realizará una valoración global del trabajo realizado y de los objetivos alcanzados. Se describirán las dificultades encontradas en el

desarrollo del Trabajo. Además, se describirán unas posibles vías de trabajo futuro que puedan dar continuidad a la investigación en este campo.

Capítulo 2

Estado del arte

Este capítulo describirá el Estado del Arte sobre el que se sustenta este Trabajo Fin de Máster. En primer lugar se realizará una pequeña descripción de las señales objeto de estudio, que servirá como base teórica para la correcta comprensión de la memoria. A continuación, se realizará un repaso de las técnicas existentes en el contexto en el que se sitúa el presente Trabajo, con el fin de explicar los aportes al conocimiento que realiza el mismo al estado del conocimiento actual. Por último, se detallarán los principales conceptos sobre Redes Neuronales, pues éstas son la base de los sistemas implementados en este Trabajo.

2.1. Estado del arte en mejora de voz

2.1.1. Introducción a la mejora de voz

La voz es una onda acústica de presión generada por movimientos fisiológicos voluntarios de los órganos del aparato fonador humano [8]. En el procesado de señales de voz, y con el uso de un cada vez más robusto y fiable hardware digital en las computadoras actuales, se suele trabajar con señales de voz digitalizadas, es decir, secuencias de muestras discretas que toman un número finito de valores y que pueden representarse con un determinado número de bits.

Para un análisis preciso de la señal del habla en los distintos escenarios de aplicación, es necesario comprender la naturaleza de la misma, lo que hace patente la importancia de un preprocesado previo. Para ello, se suele hacer uso de distintas representaciones, que aprovechen al máximo la información presente en la señal de voz y que permitan obtener distintas propiedades matemáticas de la misma así como descubrir características fisiológicas del hablante. Además, en ocasiones, con esta extracción de características es posible reducir la dimensionalidad de los datos a la vez que se mantiene la información de interés o más relevante de la señal de voz, lo que disminuye el tiempo de cómputo en la ejecución del modelado posterior de los datos.

Sin embargo, la representación eficiente de las señales del habla en términos de un

pequeño número de parámetros es un problema de considerable importancia en la investigación del habla. Así pues, a continuación, se detallarán las representaciones más comunes de la señal de voz, destacando las principales características que pueden extraerse en los distintos dominios. El preprocesado de los datos consiste en extraer esta representación a partir de la señal digital de voz.

Extracción de características en la señal de voz

Existen diferentes formas de representar las señales de voz, las cuales se adaptan generalmente a las aplicaciones específicas de destino.

Sin embargo, independientemente de la aplicación, es conveniente tener en cuenta algunas consideraciones comunes en la extracción de características. Los parámetros deben ser perceptivamente significativos, lo que significa que deben ser análogos a los utilizados por el sistema auditivo humano y deben representar aspectos destacados de la señal del habla. Además, para garantizar la fiabilidad del sistema del que formen parte, éstos deben ser robustos a las variaciones en entornos como canales, micrófonos, altavoces e incluso, en la medida de lo posible, ruido. Asimismo, unos parámetros altamente significativos y representativos deberían capturar cambios de espectro con el tiempo y contener información contextual de los instantes de articulación del sonido.

Algunas de las representaciones que pueden ser adecuadas para el procesado de la señal de voz y para la correcta extracción de características son las siguientes:

- **Representación en el dominio del tiempo**.

  Esta versión de la señal de voz, sobre la que no se ha efectuado ningún tipo de procesamiento, es lo que se conoce como señal en el dominio del tiempo o forma de onda. Se trata de muestras acústicas en bruto, es decir, una secuencia discreta de números reales o enteros correspondientes a las muestras de la señal, muestreadas y cuantificadas tal y como se mencionó en párrafos anteriores. Esta representación de la señal de voz es la que encontramos, por ejemplo, en un archivo WAV codificado en PCM.

  Mediante la representación de la señal en el dominio del tiempo es posible obtener algunas de las características que definen una señal de voz, tales como la energía o la cantidad de cruces por cero. Además, con este tipo de representación es posible identificar instantes de silencio, útiles en aquellos casos en los que se quiera obtener una estimación del ruido de fondo presente en la señal.

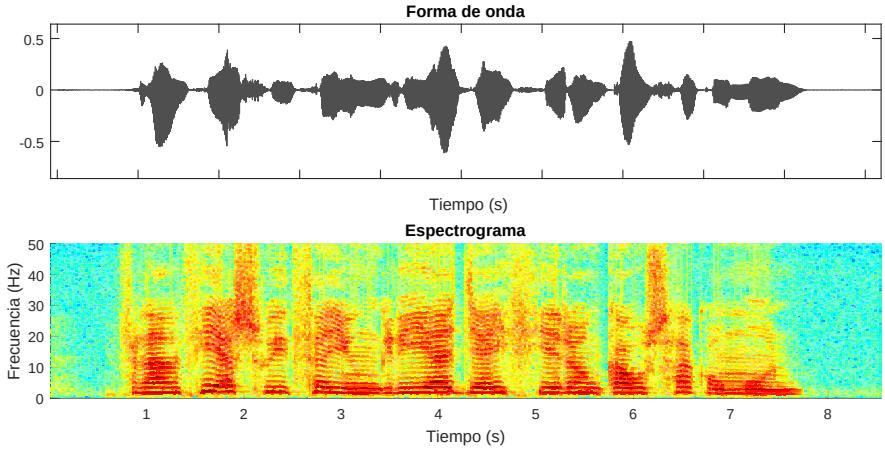

Figura 2.1: Representación en el dominio del tiempo (parte superior) y espectrograma (parte inferior) de una señal de voz.

En la parte superior de la Figura 2.1 aparece un ejemplo de una señal de voz en el dominio del tiempo.

- **Representación espectral**

  También es posible analizar una señal de voz en el dominio de la frecuencia, con el fin de conocer sus características espectrales. Para ello se hace uso de la Transformada Discreta de Fourier (DFT, del ingés *Discrete Fourier Transfom*), representada en la ecuación 2.1. La voz es una señal de banda ancha con frecuencias que varían desde los 100 Hz a los 8000 Hz aproximadamente. No obstante, la mayor parte de la energía está concentrada por debajo de los 4 kHz, por lo que la intelegibilidad de la voz tiene mayor peso en este rango de frecuencias.

$$DFT\{x_n\} = X_k = \sum_{n=0}^{N-1} x_n e^{-\frac{i2\pi}{N}kn} \qquad k = 0, ..., N-1 \tag{2.1}$$

  Resulta más conveniente utilizar la Transformada de Fourier a Corto Plazo (STFT, del ingés *Short Time Fourier Transform*) para representar la evolución temporal del espectro de la señal. Para ello, se divide la señal en tramas enventanadas, habitualmente con solapamiento, y se calcula la DFT a cada una de ellas. El uso de una ventana evitará las transiciones abruptas en los extremos de las tramas y, por tanto, la aparición de frecuencias indeseadas. La expresión de la STFT viene dada por la ecuación 2.2.

$$STFT\{x_n\} = X_{m,k} = \sum_{n=0}^{N-1} x[n]w[n-m]e^{-\frac{i2\pi}{N}kn} \qquad k = 0, ..., N-1 \tag{2.2}$$

Un espectrograma es la versión de magnitud al cuadrado de la expresión anterior, es decir, sin información de fase. Sin embargo, obtener una buena resolución simultáneamente en tiempo y en frecuencia es una tarea difícil, por lo que elegir en qué dominio establecer una mejor resolución depende de la aplicación de destino. Las ventanas de mayor duración tienen alta resolución en frecuencia y baja resolución en el dominio del tiempo, mientras que para ventanas cortas sucede lo contrario. Además, el tipo de ventana utilizado en el proceso de enventanado también influye en los resultados. Así pues, los espectrogramas se pueden construir utilizando diferentes tipos de ventanas con distintas longitudes. En la imagen inferior de la Figura 2.1 aparece el espectrograma correspondiente a la señal de voz de la imagen superior, construido utilizando una ventana de hamming de 20 ms.

Una característica importante de la señal de voz es su no-estacionariedad, pues sus componentes en tiempo y en frecuencia se encuentran en continuo cambio. Sin embargo, al tratarse de un proceso fisiológico no pueden variar bruscamente, por lo que las transiciones entre los niveles de amplitud de los distintos sonidos se producen de forma gradual. Por este motivo, puede considerarse que la señal de voz es estacionaria en cortos periodos de tiempo (20-30 ms) y suelen utilizarse tramas de esta duración para dividir a la misma. En estos intervalos se podrá analizar y procesar la señal de voz como una señal estacionaria.

Esencialmente, la representación de la voz en el dominio espectral es prácticamente lo mismo que utilizar las muestras en bruto de la representación en el dominio del tiempo, pues no existe pérdida de información en la transformación entre ambos dominios. Sin embargo, en algunas ocasiones, la representación en el dominio de la frecuencia puede percibirse de forma más clara para los humanos, pues permite asociar el contenido en distintas bandas de frecuencia con los diferentes fonemas.

- **Representación en la escala Mel**

  La escala Mel es una escala de frecuencia no lineal que pretende imitar el comportamiento psicoacústico del oído humano. Los MFCC (Coeficientes Cepstrales en escala de frecuencias Mel – *Mel Frequency Cepstral Coefficients*) son ampliamente utilizados para representar el habla y hacen uso de la escala Mel. Estos coeficientes permiten obtener características locales de la señal de voz similares, en resolución frecuencial, a la información que envía el oído al cerebro.

  Para el cálculo de los coeficientes MFCC se realiza, inicialmente, un proceso de preénfasis que realza las altas frecuencias, seguido de un enventanado de la señal de voz. A continuación, se calcula la DFT de la señal para pasarla al dominio de la frecuencia, de la cual se conserva sólo la amplitud y se descarta la fase. Es este punto cuando se multiplica dicha señal por un banco de filtros triangulares espaciados de acuerdo a la escala Mel. Finalmente, se calcula el logaritmo y mediante la DCT (Transformada Discreta del Coseno, del ingés Discrete Cosine Transform) obtenemos los coeficientes cepstrales en escala de frecuencias de Mel, dando lugar al vector de MFCCs.

  Las transformaciones tradicionales, como la Transformada de Fourier, se caracterizan por retribuirle la misma importancia a todo el espectro de frecuencias. Sin

embargo, la particularidad fundamental en los coeficientes MFCC es que las bandas de frecuencias están situadas logarítmicamente, según la escala Mel, asignando mayor relevancia a las bajas frecuencias. Esta escala está basada en el oído humano y se ha construido a través de experimentos fisiológicos. Diversos estudios han demostrado que el sistema auditivo humano procesa la señal de voz en el dominio espectral, caracterizándose por tener mayor resolución en bajas frecuencias y una menor resolución en frecuencias más altas.

2.1.2. Influencia del ruido en señales de voz

Existen distintos factores que pueden afectar a la señal de voz, impidiendo su correcta intelegibilidad y la extracción de características en ella. En todas las áreas de aplicación de la voz, resulta conveniente que la señal de voz se encuentre lo más limpia posible para facilitar su estudio y tratamiento. Por tanto, es necesario un procesado previo que mejore la señal y reduzca el ruido presente.

Las señales de voz pueden estar afectadas por distintos elementos perturbadores. Entre estos efectos nocivos se encuentran el ruido y la distorsión, los cuales aparecen en gran parte de las aplicaciones en las que está implicada la voz. La distorsión provoca una modificación de la señal. Sin embargo, el ruido es totalmente independiente de ésta última y puede producir una degeneración en ella, afectando a la calidad y a su intelegibilidad. Así pues, podemos definir el ruido como una señal no deseada que interfiere en la comunicación y que impide una comunicación fluida y el correcto procesado de la señal de voz.

En este Trabajo nos centraremos en el ruido aditivo, pues es el efecto negativo que afecta a las señales objeto de estudio. Puede considerarse ruido aditivo a todo aquel ruido que es añadido a la señal, dando lugar a una señal resultante que es la suma de dicho ruido y de la señal original. Este puede proceder de distintas fuentes y coexiste en el mismo medio acústico que la señal.

Para el proceso de reducción de ruido es fundamental conocer las características de la señal afectada, así como las del ruido y las fuentes que lo producen. Algunos tipos de ruidos predominantes en señales de voz del mundo real pueden ser ruido de tráfico, de alarmas y sirenas, de gente de fondo, sonidos medioambientales, etc. Así pues, la mejora del habla pretende mejorar la intelegibilidad y naturalidad de esta última.

2.1.3. Evolución de las técnicas en mejora de voz

La mejora de voz ha sido objeto de estudio de numerosas investigaciones a lo largo de la historia y se presenta como un problema grave en el procesamiento de señales de voz. Desde hace varias décadas, los investigadores han centrado más la atención en esta área debido al creciente número de aplicaciones del mundo real en las que está involucrada la señal de voz. Así pues, la mejora de voz se presenta como una tarea fascinante y desafiante para el reconocimiento de voz, comunicaciones móviles, sistemas de teleconferencia, diseño

de audífonos, etc. El objetivo es reducir el ruido y aumentar la relación señal a ruido[1] (SNR, del inglés *Signal to Noise Ratio*) de las señales de voz corrompidas por el ruido. Pero los resultados no siempre son satisfactorios en términos de calidad e inteligibilidad.

Numerosas técnicas de mejora del habla se han desarrollado en los últimos años. La mayoría de lo métodos tradicionales de reducción de ruido realizan la limpieza del habla contaminada sobre una transformación de la señal temporal original [3]. En este sentido, las técnicas clásicas han recurrido usualmente a representaciones en términos de la Transformada Discreta de Fourier (DFT, del inglés *Discrete Fourier Transform*), la Transformda Discreta en Cosenos (DCT, del inglés *Discrete Cosine Transform*) y la Transformada Wavelet Discreta (DWT, de inglés *Discrete Wavelet Transform*). Los algoritmos clásicos comprenden diferentes variantes de sustracción espectral [4], técnicas de filtrado óptimo y adaptativo [5], y la estimación de la señal limpia adoptando modelos estadísticos [6] [7]. Todas estas técnicas requieren conocer las características del ruido, las cuales se estiman durante los intervalos de silencio de la secuencia de voz.

La sustracción espectral es uno de los métodos más simples para eliminar el ruido aditivo en señales de voz monocanal y fue propuesto por Steven F. Boll en 1979 [4]. Este enfoque consiste en estimar la magnitud del espectro de la señal limpia y asignarle la fase de la señal contaminada. En cada tramo de análisis, la estimación de la magnitud de la voz limpia se obtiene restando a la misma una estimación del espectro de magnitud del ruido contaminante. El hecho de estar restando un estimador del ruido hace que aparezcan picos espectrales distintos en tramas consecutivas, que se convierten en tonos de muy corta duración al reconstruir la señal al dominio temporal, y cuyas frecuencias varían de trama a trama. Esta distorsión es conocida como “ruido musical”.

Las técnicas de filtrado óptimo y adaptativo se basan en pasar la señal contaminada a través de un filtro lineal que atenúa la componente ruidosa. El filtrado de Wiener se incluye dentro de las técnicas de filtrado óptimo e intenta minimizar el error cuadrático medio entre la señal limpia y la estimada. Otro enfoque consiste en estimar el espectro de la señal limpia de forma iterativa, modelando la voz como la respuesta de un sistema autorregresivo. En [5] el filtrado de Wiener iterativo se presentó utilizando un modelo de todos los polos. Sin embargo, el ruido musical es todavía un problema común en estos métodos tradicionales.

Otro trabajo notable fue el estimador de error cuadrático mínimo (MMSE) introducido por Ephraim y Malah [6]. Este método consiste en minimizar el error cuadrático medio sobre la magnitud o sobre el logaritmo de la magnitud de los coeficientes de Fourier de la voz limpia. Aunque estos métodos tradicionales basados en MMSE o Log-MMSE [7] pueden producir un menor ruido musical sin afectar a la calidad de la voz, sigue siendo necesario realizar una compensación para reducir la distorsión del habla y el ruido residual.

Con el fin de mitigar las deficiencias que presentan los métodos anteriores, en [8] se propusieron algoritmos de reducción de ruido basados en la Transformada Wavelet

[1]SNR, *Signal to Noise Ratio*. La SNR mide la relacion entre la potencia de una senal y la potencia del ruido que la corrompe, es decir, compara el nivel del ruido de fondo existente en una señal con el nivel de dicha señal.

Discreta. El objetivo de estos algoritmos es eliminar el ruido en el dominio transformado mediante un análisis multiresolución tiempo-frecuencia de las señales contaminadas. Otros métodos clásicos populares incluyen métodos basados en subespacios de señales [9] [10].

La mayoría de estos métodos no supervisados se basan en la naturaleza aditiva del ruido de fondo o en las propiedades estadísticas de las señales de voz y ruido. Sin embargo, a menudo no logran eliminar el ruido no estacionario para escenarios del mundo real en condiciones acústicas inesperadas. Por este motivo, las investigaciones recientes en mejora de voz centran sus estudios en modelos no lineales como las redes neuronales, las cuales modelan de forma eficiente la relación entre las señales de voz ruidosas y limpias. Los trabajos iniciales en este ámbito desarrollaron redes neuronales poco profundas (SNN, del inglés *Shallow Neural Network*) como filtros no lineales para predecir la señal limpia en el dominio del tiempo o la frecuencia. En [11] se propuso el uso de una SNN con una sola capa oculta para estimar las relaciones instantáneas de relación señal a ruido en los espectrogramas de modulación de amplitud, de forma que el ruido pudiera suprimirse de acuerdo con la SNR estimada. Sin embargo, el pequeño tamaño de la red y la poca profundidad de la misma no proporcionaban una buena estimación de la SNR. Además, un problema común observado para los algoritmos de mejora del habla basados en redes neuronales de este tipo era el bajo rendimiento en condiciones de ruido desconocido por el sistema. Un método simple, pero efectivo, para hacer frente a esta situación es incluir muchos tipos diferentes de ruido en el conjunto de entrenamiento [11] [12].

Recientemente, modelos basados en Redes Neuronales Profundas se han utilizado para tareas de mejora del habla. Las FC-DNN (*Fully-Connected Deep Neural Network*), en particular, han demostrado un buen rendimiento de generalización en condiciones acústicas desafiantes y pueden considerarse como el estado del arte inicial en este sector. Los modelos de mejora de voz basados en FC-DNN son modelos de regresión que aprenden un mapeo entre las características de entrada de voz ruidosas y un objetivo deseado, la voz limpia. Los enfoques comunes en este ámbito incluyen el mapeo espectral [13] [14], el enmascaramiento de tiempo-frecuencia (T-F) [15] [16] y enfoques de aprendizaje multitarea [26] [18]. En concreto, en el enfoque de mapeo espectral, la red neuronal predice espectros de potencia o magnitud de voz limpia a partir de espectros de voz ruidosa. Dentro de este grupo se sitúa el trabajo realizado en [13], en el que se propuso una compleja arquitectura de red FC-DNN para mejora de voz. El sistema propuesto en este último trabajo servirá como sistema baseline para los nuevos sistemas propuestos en este Trabajo Fin de Máster. Además, con el reciente auge de las Redes Neuronales Convolucionales (CNN, del ingés *Convolutional Neural Network*) algunas investigaciones en mejora de voz han destinado sus estudios a este tipo de redes [19] [20].

Sin embargo, las CNNs y sus variantes no han sido explotadas lo suficiente para mejora de voz. En este contexto se sitúa este trabajo fin de master, que pretende contribuir a la investigación en mejora de voz en este campo, explorando nuevas vertientes en el mapeo espectral y dando impulso a iniciativas que se alejen del mismo, rompiendo con la idea tradicional de supresión de ruido en el dominio transformado de Fourier.

2.2. Redes Neuronales Profundas

2.2.1. Introducción a las Redes Neuronales

El aprendizaje automático o *machine learning* es una aplicación de inteligencia artificial (IA, del inglés *Intelligence Artificial*) que proporciona a los sistemas la capacidad de aprender y mejorar automáticamente a partir de la experiencia sin ser programado explícitamente y sin intervención humana en el proceso. El aprendizaje automático se centra en el desarrollo de algoritmos que construyen modelos matemáticos o estadísticos a partir de datos de muestra, conocidos como datos de entrenamiento u observaciones, que permiten buscar patrones en los datos para hacer predicciones o tomar decisiones sobre datos desconocidos.

La forma más común de aprendizaje automático es el aprendizaje supervisado, en la que el algoritmo de entrenamiento aprende una función de mapeo entrada-salida a través de pares de entrada-salida de ejemplo. Además, el aprendizaje supervisado puede ser utilizado para problemas de clasificación o para problemas de regresión. La regresión se utiliza para predecir valores continuos mientras que la clasificación se utiliza para predecir de qué clase forma parte un punto de datos (valor discreto). En nuestro caso, donde el objetivo es eliminar el ruido presente en segmentos de voz, nos encontramos ante un problema de regresión en el que la salida puede tomar un rango continuo de valores.

Sin embargo, las técnicas convencionales de aprendizaje automático están limitadas en su capacidad para procesar datos naturales en su forma original. Durante décadas, la construcción de sistemas de aprendizaje automático ha requerido una ingeniería cuidadosa y una experiencia considerable en este sector para diseñar un extractor de características que transforme los datos sin procesar en una representación interna adecuada o en un vector de características desde el cual el subsistema de aprendizaje pueda detectar patrones y predecir valores a la salida. Por este motivo, la tecnología de *machine learning* hace uso, cada vez más, de técnicas de aprendizaje profundo o *Deep learning*.

El aprendizaje profundo permite que modelos computacionales compuestos de múltiples capas de procesamiento aprendan representaciones de datos con múltiples niveles de abstracción. Esta técnica utiliza los algoritmos *forward-propagation* y *backpropagation* para indicar cómo una máquina debe cambiar sus parámetros internos, los cuales se utilizan para calcular la representación en cada capa a partir de la representación en la capa anterior. En los siguientes apartados, se describirán con detalle los dos algoritmos mencionados.

Así pues, una de las representaciones de *Deep Learning* más habituales en la actualidad son los modelos basados en Redes Neuronales Profundas. Las redes neuronales son sistemas inspirados en el cerebro que pretenden replicar la forma en que los humanos aprendemos. El Dr. Robert Hecht-Nielsen definió una red neuronal como un sistema de computación compuesto por una serie de elementos de procesamiento simples, altamente

interconectados, que procesan información por su respuesta de estado dinámico a entradas externas [21].

Los modelos de redes neuronales son especificados por la topología o arquitectura de la red, es decir, la estructura y tipos de enlaces, las características de los nodos o neuronas y las técnicas de aprendizaje usadas para ajustar los pesos. En los siguientes apartados describiremos las dos arquitecturas de redes neuronales que se han usado en este Trabajo Fin de Máster, abordando desde los conceptos más básicos de los Perceptrones Multicapa, hasta conceptos más avanzados presentes en Redes Convolucionales.

2.2.2. El Perceptrón Multicapa (FC-DNN)

Arquitectura

El perceptrón multicapa (MLP, del inglés *Multi-Layer Perceptron*) es una de las topologías más difundidas y aceptadas y es también conocida como red *fully-connected Deep Neural Network* (FC-DNN). Está organizada en capas, las cuales se componen de un conjunto de nodos o neuronas que contienen una función de activación. A la primera capa se le denomina capa de entrada y el número de neuronas presentes en la misma debe establecerse en función de la dimensionalidad de los datos de entrada. Ésta capa va seguida de una o varias capas ocultas, las cuales forman un sistema de conexiones ponderadas que establecen la profundidad de la red. La última capa es la capa de salida y, al igual que en la capa de entrada, el número de neuronas debe fijarse previamente en función del problema a resolver. Los enlaces entre las distintas capas se establecen desde unidades en el nivel i a unidades en el nivel j.

Los elementos básicos que constituyen una arquitectura de red neuronal son los siguientes:

- **Entradas**. Son un conjunto de neuronas que reciben los datos o señales del exterior.
- **Pesos**. Son los coeficientes que proporcionan la importancia de la entrada a la función de agregación de la neurona. Existe un peso por cada enlace dentro de la red y éstos se modifican y adaptan en función de los ejemplos de entrenamiento y de las reglas establecidas para tal propósito. El proceso mediante el cual se determina el valor de los pesos de la red es lo que se conoce como entrenamiento de la misma.
- **Salidas**. Las salidas son aquellas neuronas que proporcionan la respuesta de la red neuronal, la cual está asociada a la función de activación seleccionada.

La Figura 2.2 muestra un ejemplo de red neuronal de tipo perceptrón multicapa.

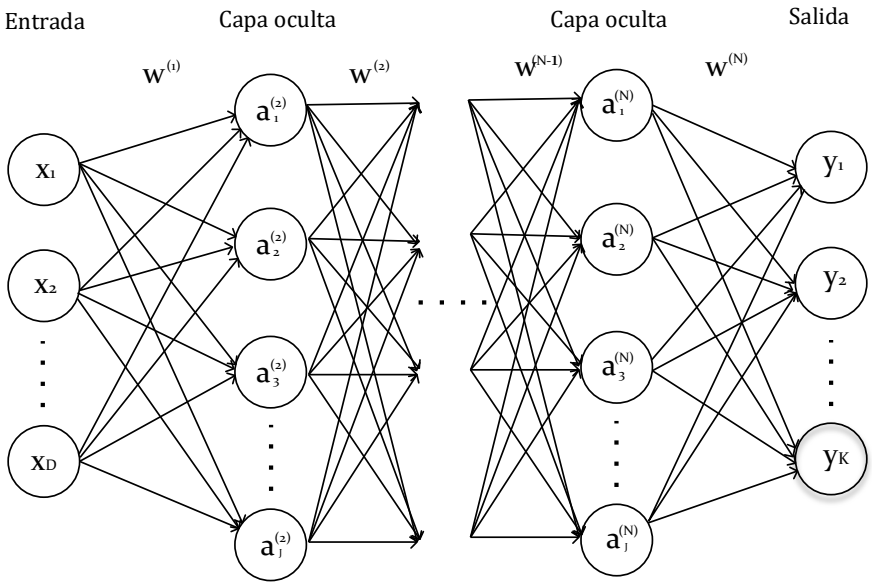

Figura 2.2: Perceptrón multicapa con D neuronas en la capa de entrada, $N-1$ capas ocultas de J neuronas y K neuronas en la capa de salida.

En este tipo de red, resulta interesante analizar lo referente a cuántas capas y neuronas por capa proporcionan el mejor resultado. Así pues, estos parámetros junto a otros que se describirán más adelante, como los tipos de funciones de activación o el *learning rate*, son lo que se conocen como hiperparámetros de la red. Estos parámetros no se utilizan para modelar directamente los datos, pero influyen en la capacidad de aprendizaje del modelo.

Algoritmo *Forward Propagation*

El algoritmo *Forward Propagation* es aquel mediante el cual se obtiene la salida de cada una de las neuronas que conforman una red neuronal. El modelo básico de red neuronal puede describirse como una serie de transformaciones funcionales. El objetivo en redes neuronales es conseguir que estas transformaciones dependan de los parámetros, para que éstos puedan ajustarse durante el entrenamiento.

Así pues, el valor que toma cada neurona de una determinada capa puede obtenerse mediante una operación no lineal de una combinación lineal de las entradas provenientes de las neuronas de la capa anterior, donde los coeficientes de la combinación lineal son parámetros adaptativos.

Para un modelo como el de la Figura 2.2, en el que la capa de entrada se compone de D neuronas, las $N-1$ capas ocultas constan de J neuronas y la capa de salida consta de K neuronas, vamos a usar la siguiente nomenclatura. Se ha denotado con x_i a cada una de las entradas, w^n a la matriz de pesos que controla el mapeo entre la capa n y la capa $n+1$, $w_{(0)}$ a los *bias*, a_j^n a la activación de la unidad j en la capa n, a_k^n a la activación de

la unidad k en la capa n y $g(.)$ a la función de activación de cada neurona.

Las activaciones de cada neurona en la primera capa oculta se obtienen mediante la expresión 2.3.

$$a_j^2 = \sum_{i=1}^{D} w_{ji}^1 x_i + w_{j0}^1 \tag{2.3}$$

Para obtener la salida de cada neurona basta con realizar una operación no lineal a las activaciones anteriores, de la forma $z_j^2 = g(a_j^2)$.

A continuación, las salidas anteriores se combinarían linealmente para dar lugar a las activaciones de las siguientes capas ocultas, las cuales se someterían, de nuevo, a una operación no lineal para obtener las nuevas salidas.

Para la capa de salida las activaciones se calcularían mediante la ecuación 2.4.

$$a_k^{N+1} = \sum_{j=1}^{J} w_{kj}^N z_j + w_{k0}^N \tag{2.4}$$

Finalmente, las salidas de la red se obtendrían a partir de una operación no lineal a los valores dados por 2.4, esto es, $z_k^{N+1} = y_k^{N+1} = g(a_k^{N+1})$.

Las operaciones no lineales que se llevan a cabo durante el proceso descrito se realizan mediante funciones de activación. Éstas definen la salida de una neurona, así como el rango de posibles valores de la misma. Además, representan el estado de activación de la neurona, de forma que la salida de ésta última sólo se propagará hacia el resto de neuronas si la neurona está activa. A continuación, se describen algunas de las funciones de activación más comunes.

- **Función lineal**. La salida devuelta por esta función es un valor proporcional a la entrada. El rango posible de salidas se extiende desde $-\infty$ hasta $+\infty$.

$$g(x_i) = cx_i \tag{2.5}$$

- **Función sigmoide**. Esta función solo devuelve valores positivos entre 0 y 1. El valor de m está relacionado con la pendiente de la función.

$$g(x_i) = \frac{1}{1 + e^{-mx_i}} \tag{2.6}$$

- **Función tangente hiperbólica**. Esta función es similar a la función sigmoide. Los valores de salida están comprendidos dentro del rango que va desde -1 a 1. El valor de m está relacionado con la pendiente de la función.

$$g(x_i) = \frac{e^{mx_i} - e^{-mx_i}}{e^{mx_i} + e^{-mx_i}} \tag{2.7}$$

- **Función ReLU (*Rectified Linear Unit*)**. Esta función devuelve el valor 0 si la entrada es negativa y el valor de la propia entrada en caso contrario. El rango de salida abarca desde 0 a $+\infty$. Esta función es la función de activación no lineal más simple y una de las más utilizadas.

$$g(x_i) = max(0, x_i) \tag{2.8}$$

- **Función Leaky ReLU**. Esta función es una modificación de la ReLU. En esta última, todos los valores de entrada negativos se vuelven cero a la salida, lo que disminuye la capacidad del modelo para ajustarse o entrenarse con los datos correctamente. Así pues, la LeakyReLU permite retener, en cierto modo, los valores negativos de la entrada, ofreciento una mayor flexibilidad al modelo que lo usa. El rango de salida se extiende desde $-\infty$ hasta $+\infty$.

$$g(x_i) = max(0.01x_i, x_i) \tag{2.9}$$

- **Función Softmax**. Esta función es a menudo utilizada en la última capa de una red neuronal utilizada para clasificación multiclase. La función Softmax devuelve la probabilidad de cada una de las clases del problema, por lo que todos los valores de la capa suman 1 para poder ser interpretados como probabilidades a posteriori de cada clase. Así pues, el rango de salida está comprendido entre 0 y 1.

$$g(x_i) = \frac{e^{x_i}}{\sum_i e^{x_i}} \tag{2.10}$$

Algoritmo *Backpropagation*

El algoritmo *Backpropagation* es el método utilizado para el entrenamiento de redes neuronales. Dada una red neuronal artificial y una función de error, este algoritmo calcula el gradiente de la función de error con respecto a los pesos de la red neuronal. El objetivo principal es obtener el valor de los pesos que minimizan el error entre la salida devuelta por el modelo y los valores reales que debería devolver.

Para una correcta comprensión del funcionamiento de este algoritmo, deben considerarse los siguiente parámetros. Un conjunto de datos formado por pares de entrada-salida (x_i, t_i), donde x_i es la entrada y t_i es la salida deseada para la entrada x_i. Al peso entre el nodo j de la capa $n+1$ y el nodo i de la capa n es denotado de la forma w^n_{ji}, y a los *bias* como $w_{(0)}$. Todos estos parámetros son colectivamente denotados por θ y a la función de error o de coste que define el error entre la salida deseada, t_i, y la estimada por la red, y_i, es nombrada como $E(X, \theta)$, donde X es el conjunto de todas las entradas.

El algoritmo *backpropagation* puede definirse en los siguientes pasos, asumiendo una determinada tasa de aprendizaje o *learning rate*, η, y una inicialización aleatoria de los pesos w^n_{ij}.

En primer lugar, debe aplicarse el algoritmo *Forward Propagation* descrito anteriormente para obtener las activaciones de todas las neuronas y las salidas de la red, y_i.

A continuación, debe calcularse el error a la salida para todas las neuronas, δ_k, empleando la expresión 2.11.

$$\delta_k = y_k - t_k \tag{2.11}$$

Estos errores deben propagarse hacia atrás, para obtener el error de las neuronas de las capas ocultas, δ_j. Para ello debe utilizarse la expresión 2.12, donde $g'(.)$ es la derivada de la función de activación. El desarrollo matemático de esta ecuación puede encontrarse en [2].

$$\delta_j = g'(a_j) \sum_k w_{kj}\delta_k \tag{2.12}$$

Seguidamente, deben combinarse los gradientes individuales para cada par entrada-salida para obtener el gradiente total del conjunto de datos completo, tal y como aparece en la siguiente expresión.

$$\frac{\partial E_n}{\partial w_{ji}} = \delta_j z_i \tag{2.13}$$

Finalmente, deben actualizarse los pesos de la red, cuya variación es proporcional al gradiente de la función de coste y la tasa de aprendizaje o *learning rate*, tal y como muestra la ecuación 2.14.

$$w^{\tau+1} = w^{\tau} - \eta \nabla E(w^{\tau}) \tag{2.14}$$

En el algoritmo descrito, una de las tareas fundamentales es definir la función de coste. Esta función mide el error entre la salida devuelta por la red y la salida deseada. Así pues, debe ser un valor positivo que disminuya a medida que se ajusta el modelo a los datos, es decir, durante el entrenamiento. A continuación, se mencionan algunas de las funciones de coste más usuales.

- **Mean Squared Error (MSE)**. Esta función calcula la diferencia al cuadrado entre la salida real y la estimada por la red y busca minimizar esta operación. Es la función de coste más utilizada en regresión lineal.

$$E = \frac{1}{n} \sum_{i=1}^{n} (t_{(i)} - y_{(i)})^2, \tag{2.15}$$

  donde n representa el número total de muestras de los vectores con los que se trabaja, $t_{(i)}$ representa la salida deseada y $y_{(i)}$ la salida estimada.

- **Mean Absolute Error (MAE)**. Esta función también mide diferencias entre los valores obtenidos por la red y los deseados. Al utilizar el valor absoluto para el cálculo de las diferencias, en lugar de elevarse al cuadrado como en el MSE, esta función es más robusta a valores grandes y atípicos.

$$E = \frac{1}{n}\sum_{i=1}^{n}|t_{(i)} - y_{(i)}|, \tag{2.16}$$

donde n representa el número total de muestras de los vectores con los que se trabaja, $t_{(i)}$ representa la salida deseada y $y_{(i)}$ la salida estimada.

- **Entropía cruzada**. Esta función de coste se utiliza en problemas de clasificación y trabaja con funciones de probabilidad. El empleo conjunto de la función de activación softmax y función de coste de entropía cruzada en una capa de salida permite obtener, en una red perfectamente entrenada, las probabilidades a posteriori de cada clase en las salidas de la red.

$$E = -\frac{1}{n}\sum_{i=1}^{n}[t_{(i)}log(y_{(i)}) + (1 - t_{(i)})log(1 - y_{(i)})], \tag{2.17}$$

donde n representa el número total de clases, $t_{(i)}$ representa la salida deseada y $y_{(i)}$ la salida estimada.

Por otro lado, tal y como puede comprobarse en la expresión 2.14, los pesos dependen de la tasa de aprendizaje o *learning rate*, η, la cual representa uno de los hiperparámetros más importantes en el entrenamiento de redes neuronales. Este parámetro es fundamental en los algoritmos de optimización, pues éstos se diferencian entre sí en la forma en la que lo utilizan.

Los algoritmos de optimización ayudan a minimizar la función de coste, ya al tratarse de una función no lineal es difícil encontrar su mínimo. Los algoritmos más utilizados para este propósito son aquellos basados en técnicas de descenso por gradiente. Estas técnicas minimizan una función objetivo parametrizada por los parámetros de un modelo, actualizando los parámetros en la dirección opuesta a la del gradiente de la función objetivo. Así pues, si el *learning rate* es bajo, la optimización del modelo tardará mucho tiempo, ya que los pasos hacia el mínimo de la función de coste serán muy pequeños. Por el contrario, si el *learning rate* es alto, el entrenamiento puede no converger, no alcanzándose nunca el mínimo.

A continuación, se enumeran algunos de los algoritmos de optimización más utilizados.

- **Stochastic Gradient Descent (SGD)**. Este algoritmo es una aproximación estocástica de la optimización del descenso de gradiente y viene dado por la expresión 2.14. Se diferencia del descenso por gradiente estándar en que las muestras son seleccionadas al azar, en lugar de como un solo grupo o en el orden en que aparecen en el conjunto de entrenamiento.

  La principal desventaja de este algoritmo es que requiere un *learning rate* fijo. Una solución estándar que funciona bien en la práctica es usar una tasa de aprendizaje constante lo suficientemente pequeña que proporcione una convergencia estable en durante las primeras épocas y luego reducir a la mitad su valor a medida que la convergencia disminuye.

- **Adaptive Gradient Algorithm (AdaGrad)**. Adagrad [22] es un optimizador con tasas de aprendizaje específicas para cada parámetro, que se adaptan en función de la frecuencia con la que un parámetro se actualiza durante el entrenamiento. Cuantas más actualizaciones reciba un parámetro, más pequeña será la tasa de aprendizaje empleada. Por el contrario, se utilizarán tasas de aprendizaje más altas con aquellos parámetros que se actualicen con menos frecuencia.

- **Adam**. Adam [23] es otro método que calcula las tasas de aprendizaje adaptativo para cada parámetro, a partir de los momentos de primer y segundo orden del gradiente de la función de coste.

  Este algoritmo es uno de lo más recientes y presenta numerosas ventajas frente a los tradicionales. Es un método sencillo de implementar, con pocos requisitos de memoria, adecuado para problemas con un elevado número de datos o parámetros y computacionalmente eficiente, lo que ha disparado su uso en los modelos actuales.

2.2.3. Redes Neuronales Convolucionales (CNN, FCN)

Introducción a las Redes Neuronales Convolucionales

A lo largo de la historia, dentro del aprendizaje profundo, han surgido una gran cantidad de arquitecturas y técnicas que pretenden dar una mejora a los resultados obtenidos con los tradicionales métodos de redes neuronales. En muchos casos, los datos de entrada muestran cierta relación temporal o espacial, como ocurre en las imágenes. Esta dependencia no se tiene en cuenta con los Perceptrones Multicapa previamente descritos.

Para aprovechar este tipo de dependencia y relación en los datos de entrada, surgieron las Redes Convolucionales. Éstas se inspiraron en investigaciones realizadas en el cerebro humano, entendiendo a este último como un conjunto de capas de neuronas. En particular, se centraron en las estructuras neuronales involucradas en la visión humana (córtex frontal). Cada grupo de neuronas puede estar diseñado específicamente para reconocer diferentes formas y pueden comunicarse entre sí para desarrollar una comprensión holística del objeto percibido. Así pues, el sistema se puede explicar como grupos jerárquicos de neuronas que detectan las características de bajo nivel de un estímulo de entrada y se comunican entre sí, siguiendo esa jerarquía, para desarrollar una detección de objetos de alto nivel.

Yann LeCun en [24] se inspiró en este modelo jerárquico del cerebro humano y desarrolló redes neuronales convolucionales para abarcar las siguientes nociones.

- Conexiones locales: en una imagen los píxeles cercanos están más fuertemente correlacionados que los píxeles distantes. Por tanto, deben extraerse características locales que dependen solo de pequeñas subregiones de la imagen. Cada capa o grupo de neuronas comparte una conexión a través de la cual transfieren las funciones aprendidas de un grupo a otro.

- Capas: existe una jerarquía obvia entre las diferentes capas o agrupaciones, que es análoga a decir que hay una jerarquía en el aprendizaje desde características de bajo

nivel a características de alto nivel. Por ejemplo, en un rostro las características de bajo nivel serían los ojos u orejas, mientras que las características de alto nivel sería la cara en su conjunto, es decir, la persona en cuestión.

- Invarianza espacial: los cambios en las entradas dan como resultado una salida igualmente modificada. Independientemente de cómo cambiemos la entrada, el modelo debería adaptarse y cambiar las salidas en consecuencia. Por ejemplo, los humanos tenemos la capacidad de reconocer un objeto incluso si está boca abajo o cambiado en una variedad de condiciones.

Arquitectura

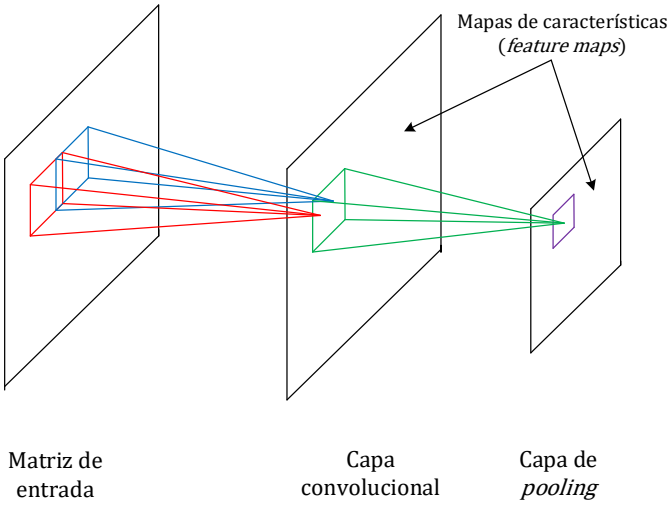

Figura 2.3: Diagrama que ilustra parte de una red neuronal convolucional: capa convolucional y capa de *pooling* con campos receptivos rectangulares. Pueden usarse varios pares sucesivos de estas capas.

Las ideas anteriores se introducen a las redes neuronales convolucionales a través de tres mecanismos:

- **Campo receptivo**. En redes convolucionales cada neurona está conectada solo a una región local del volumen de entrada, que se denomina campo receptivo. Dicho campo receptivo puede describirse por su ubicación central y su tamaño (*kernel size*). Sin embargo, no todos las muestras en un campo receptivo son igualmente importantes para su característica correspondiente, pues depende del conjunto de pesos de la red.
- **Filtros**. Las operaciones de convolución se realizan mediante filtros o *kernels*. Estos no están definidos previamente, sino que se aprenden durante el proceso de entrenamiento. Así pues, los pesos de los filtros serán los parámetros entrenables en una

capa convolucional. En redes convolucionales, se toman múltiples filtros para extraer diferentes características de la entrada. El tamaño de los mismos o *kernel size* debe coincidir con el del campo receptivo.

- **Reparto de pesos o *Weight sharing***. Dada la suposición general de que la entrada que va a ser procesada por la red se puede descomponer en un conjunto de regiones locales con la misma naturaleza, tal y como se ha descrito en el punto inicial, cada una de ellas podrá procesarse con el mismo conjunto de transformaciones, es decir, con los mismos filtros. Con el supuesto anterior, podríamos reducir la cantidad de parámetros en la red, en comparación con una red *fully connected*, y aumentar la capacidad de generalización de la misma.

- **Submuestreo o *pooling***. La operación de submuestreo o *pooling* reduce la dimensionalidad de los mapas de características extraídos. Éstos serán explicados en párrafos siguientes. Para ello aplica una ventana de un tamaño arbitrario y puede extraer la suma de las ventanas, el máximo o el promedio, según la especificación del usuario. Así pues, esta técnica nos ayuda a reducir la dimensionalidad al mismo tiempo que se preserva y mantiene la información relevante.

A continuación, vamos a relacionar los parámetros anteriores para una mejor comprensión de los mismos. La figura 2.3 muestra un diagrama de parte de una red convolucional, formado por una matriz de entrada, una capa convolucional y una capa de *pooling*.

Las neuronas en la primera capa no están conectadas a cada unidad en la matriz de entrada, como ocurre en las redes *fully connected*, sino solo a las unidades que forman parte de una pequeña subregión o campo receptivo. A su vez, si hubiera una segunda capa convolucional, ésta estaría conectada solo a las neuronas ubicadas en un pequeño rectángulo dentro de la primera capa. Esta arquitectura permite que la red se concentre en características de bajo nivel en la primera capa oculta, luego las agruparía en características de nivel superior en la siguiente capa oculta, y así sucesivamente. Al apilar capas convolucionales una encima de otra, puede obtenerse información más abstracta y profunda de la entrada.

Así pues una neurona ubicada en la fila i y columna j de una determinada capa está conectada a las salidas de las neuronas en la capa anterior ubicada en las filas i a $i + f_h - 1$ y columnas j a $j + f_w - 1$, donde f_h y f_w son las altura y ancho del campo receptivo (ver Figura 2.4 izquierda). Para que una capa tenga la misma altura y anchura que la capa anterior, es común agregar ceros alrededor de las entradas, como se muestra en el diagrama. Esto se denomina *zero padding*.

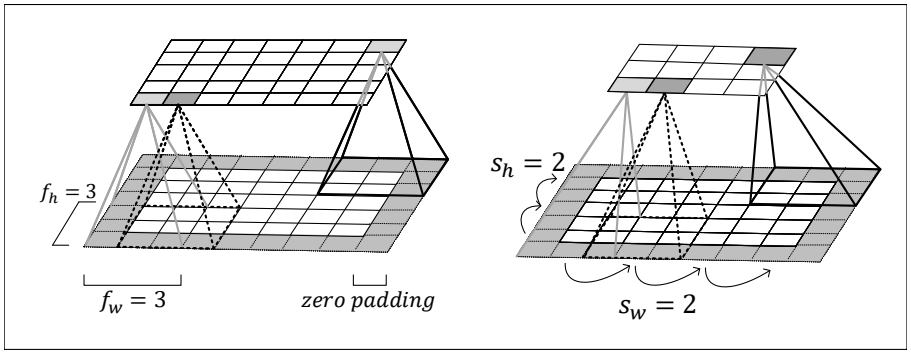

Figura 2.4: A la izquierda se muestra una representación gráfica del funcionamiento de los campos receptivos en una capa convolucional. A la derecha aparece una capa capa convolucional cuyo campo receptivo tiene un *stride* de 2.

También es posible conectar una capa de entrada grande a una capa mucho más pequeña espaciando el campo receptivo, como se muestra en la imagen derecha de Figura 2.4. La distancia entre dos campos receptivos consecutivos se denomina *stride*. En el diagrama, una capa de entrada de 5×7 (más *zero padding*) está conectada a una capa de 3×4, usando campos receptivos de 3×3 y un paso (*stride*) de 2. En este ejemplo, el paso es el mismo en ambas direcciones, pero no tiene que cumplirse obligatoriamente. Una neurona ubicada en la fila i y columna j en la capa superior está conectada a las salidas de las neuronas en la capa anterior ubicada en las filas $i \times s_h$ a $i \times s_h + f_h - 1$ y columnas $j \times s_w$ a $j \times s_w + f_w - 1$, donde s_h y s_w son los pasos verticales y horizontales.

En cada capa convolucional, las unidades se organizan en planos, cada uno de los cuales se denomina mapa de características o *feature map*. La evaluación de las activaciones de estas unidades es equivalente a una convolución entre las muestras de la matriz de entrada con un *kernel* que comprende los parámetros de los pesos. Además, todas las unidades en un mapa de características suelen compartir los mismos valores de pesos (*weight sharing*). Si pensamos en las unidades como extractores de características, entonces todas las unidades en un mapa de características detectan el mismo patrón pero en diferentes ubicaciones en la matriz de entrada, es decir, resaltan las áreas de la entrada que son más similares al filtro utilizado. Durante el entrenamiento, la red encuentra los filtros más útiles para su tarea y aprende a combinarlos en patrones más complejos.

Si las muestras en la matriz de entrada se desplazan, las activaciones del mapa de características se desplazarán en la misma cantidad, pero de lo contrario no se modificarán. Esto proporciona la base para la invariancia de la salida a distorsiones de la entrada.

Debido a que normalmente necesitaremos detectar múltiples características para construir un modelo efectivo, habrá múltiples mapas de características en la capa convolucional (ver Figura 2.5), determinados por el número de filtros o *kernels*, que debe fijarse previamente. Cada uno tendrá su propio conjunto de parámetros de peso y sesgo. El campo receptivo de una neurona es el mismo que el descrito anteriormente, pero se extiende a través de todos los mapas de características de las capas anteriores. Así pues, una capa convolucional aplica simultáneamente múltiples filtros, por lo que es capaz de detectar

múltiples características de sus entradas. Al número de filtros que se utiliza para la operación de convolución se denomina profundidad de la capa convolucional.

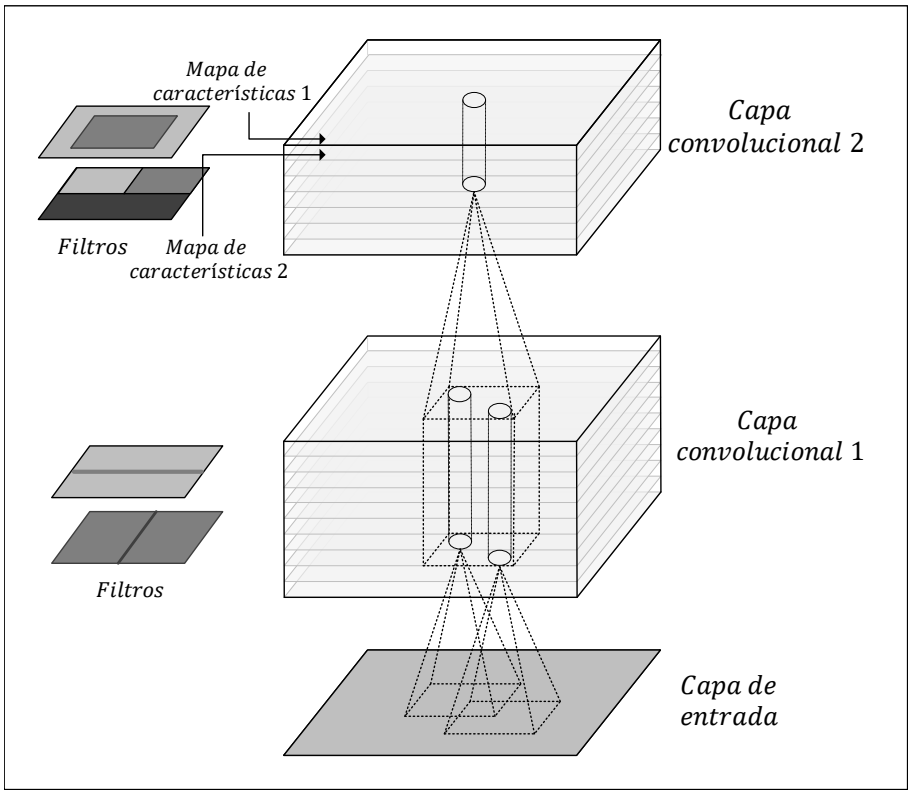

Figura 2.5: Múltiples mapas de características.

Específicamente, una neurona ubicada en la fila i y columna j del mapa de características k en una capa convolucional dada l está conectada a las salidas de las neuronas en la capa anterior $l-1$, ubicadas en las filas $i \times s_h$ a $i \times s_h + f_h - 1$ y las columnas $j \times s_w$ a $j \times s_w + f_w - 1$, en todos los mapas de características de la capa $l-1$. Además, todas las neuronas ubicadas en la misma fila i y columna j pero en diferentes mapas de características están conectadas a las salidas de las mismas neuronas en la capa anterior.

El hecho de que todas las neuronas en un mapa de características compartan los mismos parámetros reduce drásticamente el número de parámetros en el modelo. Además, debido al uso de campos receptivos, el número de pesos en la red es menor que para una red *fully connected*.

Una capa convolucional puede estar seguida de una capa de submuestreo o *pooling*, pero no necesariamente. Las salidas de las unidades en la capa convolucional formarían las entradas a la capa de *pooling* de la red. Para cada mapa de características en la capa convolucional, hay un plano de características en la capa de submuestreo y cada unidad recibe entradas de un campo receptivo pequeño en el mapa de características correspondiente de la capa convolucional. Los campos receptivos se eligen para ser contiguos y no

superpuestos. Normalmente, suelen elegirse de tamaño 2×2, de modo que haya la mitad del número de filas y columnas en la capa de submuestreo en comparación con la capa convolucional. De esta manera, la respuesta de una unidad en la capa de submuestreo será relativamente insensible a pequeños desplazamientos de la imagen en las regiones correspondientes del espacio de entrada.

En una arquitectura práctica, puede haber varios pares de capas convolucionales y de submuestreo, es decir, varios diagramas en serie como el de la figura 2.3. En cada etapa hay un mayor grado de invariancia a transformaciones de la entrada en comparación con la capa anterior. Puede haber varios mapas de características en una capa convolucional para cada plano de características en la capa de submuestreo anterior, de modo que la reducción gradual en la resolución espacial se puede compensar con un número creciente de características, esto es, número de filtros.

Existen diferentes arquitecturas predefinidas que usan capas convolucionales en el diseño de las mismas. Dos de las más utilizadas son las redes CNN (*Convolutional Neural Network*) y FCN (*Fully Convolutional Network*)

- Las CNNs constan de uno o más pares de capas convolucionales y capas de *max-pooling* [24]. Las capas convolucionales aplican un conjunto de filtros que procesan pequeñas partes locales de la entrada y que se replican a lo largo de todo el espacio de entrada, tal y como se ha mencionado anteriormente. Las capa de *max-pooling* genera una versión de menor resolución de las activaciones de la capa de convolución al tomar la activación máxima del filtro dentro de una ventana especificada. Esto da lugar a invariancia espacial y a tolerancia a pequeñas diferencias en la posición de objetos. Las capas superiores usan filtros más amplios que funcionan en entradas de menor resolución para procesar partes más complejas de la entrada. Las capas *fully-connected* finalmente combinan las entradas de todas las posiciones para proporcionar la salida final, tanto en problemas de regresión como en problemas de clasificación. Esta organización jerárquica genera buenos resultados en tareas de procesamiento de imágenes [28].
- Las FCNs presentan únicamente capas convolucionales en toda su estructura. A diferencia de las redes CNN, donde las últimas capas son *fully connected*, la capa de salida es también una capa convolucional. Esto hace que no se pierdan las relaciones locales en esta última capa y que se puedan conservar las relaciones locales en todo el proceso de aprendizaje.

Algoritmos *backpropagation* y *forward propagation*

Para el entrenamiento de la red, se puede utilizar el algoritmo *backpropagation*, descrito anteriormente, para evaluar el gradiente de la función de error, pero con algunas modificaciones.

La propagación *forward* se realiza mediante operaciones de convolución[2]. Además,

[2]Una convolución es una operación matemática entre dos funciones que produce una tercera función que expresa la forma en que una de las funciones es modificada por la otra. Las capas convolucionales en realidad usan correlaciones cruzadas, que son muy similares a las convoluciones.

las explicaciones anteriores pueden ser descritas matemáticamente. Así pues, la siguiente ecuación muestra cómo calcular la salida de una determinada neurona en una capa convolucional. En términos generales, dicha ecuación calcula la suma ponderada de todas las entradas más el término de sesgo.

$$z_{i,j,k} = b_k + \sum_{u=0}^{f_h-1} \sum_{v=0}^{f_w-1} \sum_{k'=0}^{f_{n'}-1} x_{i',j',k'} w_{u,v,k',k} \qquad \textit{siendo} \quad \begin{cases} i' = i \times s_h + u \\ j' = j \times s_w + v \end{cases}$$

- $z_{i,j,k}$ es la salida de la neurona localizada en la fila i y columna j en el mapa de características k de la capa convolucional l
- s_h y s_w son los pasos vertical y horizontal, f_h y f_w son la altura y el ancho del campo receptivo, y $f_{n'}$ es el número de mapas de características en la capa anterior (capa $l-1$)
- $x_{i',j',k'}$ es la salida de la neurona localizada en la cala $l-1$, fila i y columna j, mapa de características k.
- b_k es el término de *bias* para el mapa de características k (en la capa l)
- $w_{u,v,k',k}$ es la conexión de pesos entre cualquier neurona en el mapa de características k de la capa l y su entrada ubicada en la fila u y columna v (en relación con el campo receptivo de la neurona) y el mapa de características k'.

Las funciones de error y los optimizadores descritos previamente para una red de tipo Perceptrón Multicapa, así como el resto de conocimientos básicos, pueden ser aplicados de igual modo a este tipo de redes.

Capítulo 3

Diseño y descripción de los sistemas

En este capítulo se describirá el trabajo desarrollado y los sistemas implementados. Se detallarán las etapas en las que se divide un sistema basado en Redes Neuronales, así como los principales bloques que lo componen. Se explicarán las técnicas de procesado de señales empleadas en cada sistema y las distintas configuraciones y topologías de redes neuronales propuestas. Además, se mencionarán los motivos que han dado impulso al desarrollo de cada sistema. Así pues, la etapa de diseño es fundamental para que un proyecto cumpla con los objetivos establecidos.

3.1. Formulación del problema

Los sistemas implementados se enmarcan dentro de un enfoque novedoso en técnicas de mejora de voz, impulsado por las deficiencias de los métodos de reducción de ruido tradicionales. El trabajo pionero de Wiener y otros dan un enfoque óptimo del diseño de filtros que tienen a eliminar el ruido mientras dejan la señal deseada relativamente sin cambios. Sin embargo, el diseño de estos filtros requiere que la señal y el ruido sean estacionarios[1] y que las estadísticas de la señal sean conocidas a priori. En la práctica, estas condiciones rara vez se cumplen. Por este motivo, los métodos basados en Redes Neuronales han cobrado importancia en los últimos años.

3.2. Sistemas propuestos para mejora de voz

Antes de comenzar con la descripción detallada y el análisis de cada uno de los sistemas propuestos, debemos conocer cuál es la arquitectura general que presentan en común todos los sistemas.

[1] Una señal es estacionaria cuando las propiedades estadísticas del proceso que la genera no varían con el tiempo.

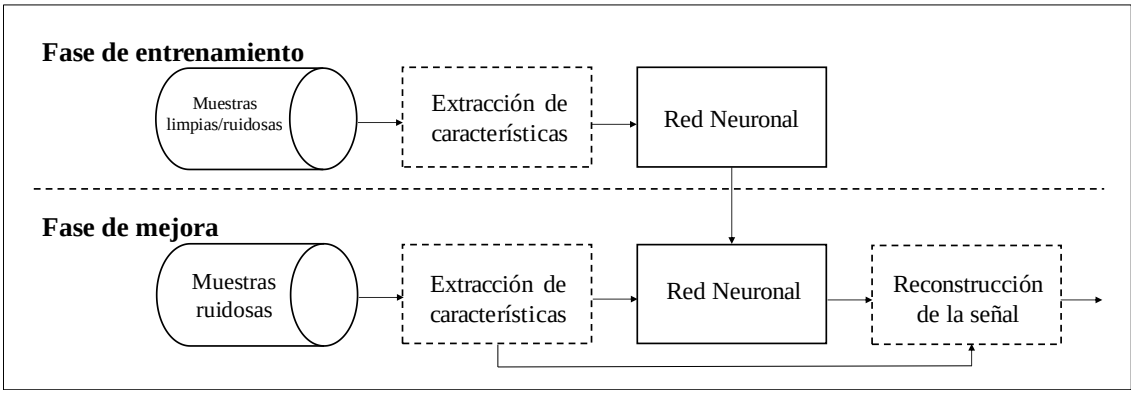

Figura 3.1: Arquitectura general de los sistemas propuestos para mejora de habla basados en redes neuronales.

Los sistemas propuestos se construyen mediante dos etapas: entrenamiento y mejora. La Figura 3.1 ilustra los diagramas de bloques de ambas etapas. En la fase de entrenamiento deben obtenerse las características de la voz limpia y sucia con las que serán representadas las señales de voz y que constituirán la entrada al modelo de red neuronal. Posteriormente, en la etapa de mejora, las características de voz ruidosas son procesadas por el modelo de red ya entrenado para predecir las características limpias del habla. Una vez obtenidas las características limpias, debe realizarse la reconstrucción de la señal al dominio original (temporal).

Para todos los sistemas implementados se seguirá el esquema general mostrado en la Figura 3.1. Sin embargo, en cada sistema se utilizará un modelo de Red Neuronal diferente, lo que permitirá compararlos e identificar el que mejor resultados ofrece. Así pues, el modelo de red neuronal utilizado en cada sistema determinará el planteamiento de reducción de ruido seguido en cada uno de ellos y la función objetivo. Además, el bloque de extracción de características debe adaptarse en función de los requisitos de entrada a la red neuronal. Las estructuras propuestas engloban redes *fully connected*, redes convolucionales y redes completamente convolucionales.

A continuación, para cada sistema se detallarán las fases de extracción de características, modelo de red neuronal y reconstrucción final de la señal. Asimismo, se mencionarán las mejoras que ofrecen los sistemas más novedosos con respecto al sistema baseline de referencia.

Además, se ha realizado un estudio empírico del valor óptimo de algunos parámetros con el objetivo de conseguir el mejor rendimiento en cada uno de los sistemas. Por tanto, se nombrarán los parámetros que se han estudiado para dicho fin.

3.2.1. Sistema basado en arquitecturas FC-DNN (*baseline*)

Este sistema fue propuesto en [13] [14] y será utilizado como sistema *baseline*. Servirá de apoyo para el diseño y desarrollo del resto de sistemas. Nuestro objetivo es mejorar el rendimiento del mismo, consiguiendo resultados más competitivos en este ámbito.

A continuación, se describirán el preprocesado o extracción de características utilizado

en este sistema, el modelo de red neuronal propuesto y la reconstrucción final de las señales de voz.

Extracción de características

La elección de las características que serán modeladas no es una tarea arbitraria, pues de ellas depende la capacidad de aprendizaje del modelo y el rendimiento del sistema completo. En el caso de mejora de voz, éstas deben ser lo suficientemente discriminatorias como para que el sistema sea capaz de distinguir entre la voz limpia (las características de la voz limpias) y el ruido (las características del ruido).

El diagrama de bloques de la Figura 3.2 muestra el proceso seguido en la extracción de características. Éste está orientado al cálculo de las componentes espectrales de la señal de voz (veáse sección 2.1.1), pues dan información de la amplitud de la misma, de la velocidad de variación y de la orientación. Además, éstas ofrecen parámetros perceptualmente relevantes [25]. A continuación se describe el proceso seguido para el cálculo de las mismas.

En primer lugar, debe realizarse un enventanado de la señal de voz. Se han seleccionado ventanas de 32 ms, lo que equivale a 512 muestras, siendo la frecuencia de muestreo de 16000 muestras/s. Además, para evitar discontinuidades en los bordes es común emplear un porcentaje de solapamiento entre tramas. En nuestro caso, se ha elegido un solapamiento de 16 ms, es decir, un 50 % del tamaño de la ventana, equivalente a 256 muestras. La ventana empleada ha sido tipo *Hamming*, cuya expresión aparece en 3.1. Además, estos valores han sido seleccionados de forma empírica.

$$w[n] = a_0 - a_1 cos(\frac{2\pi n}{N-1}) \tag{3.1}$$

donde $a_0 = 0.54$, $a_1 = 0.46$ y N el tamaño de la ventana.

A continuación, se aplica la DFT a cada una de las tramas, de las cuales se obtendrá el módulo y la fase. Finalmente, para obtener las características con las que se entrenará la red neuronal se calculará el logaritmo del módulo, dando lugar a los espectros de potencia o *log-power*. Puesto que el módulo de la DFT es simétrico, el número de muestras de entrada a la red neuronal será de 257, equivalente a la mitad del tamaño de ventana seleccionado (512 muestras). De este modo, se está reduciendo el tamaño del conjunto de entrenamiento sin pérdida de información relevante. La fase de la voz sucia será almacenada y se utilizará posteriormente en la reconstrucción final de la señal de voz. Nuestros oídos son insensibles a las pequeñas distorsiones de fase o los cambios espectrales globales [25], por lo que se ha diseñado un modelo de red neuronal basado únicamente en la estimación de la magnitud limpia del habla. No obstante, es importante señalar que las fases limpias y ruidosas son bastante diferentes a SNR bajas pero, desafortunadamente, es más difícil estimar la fase.

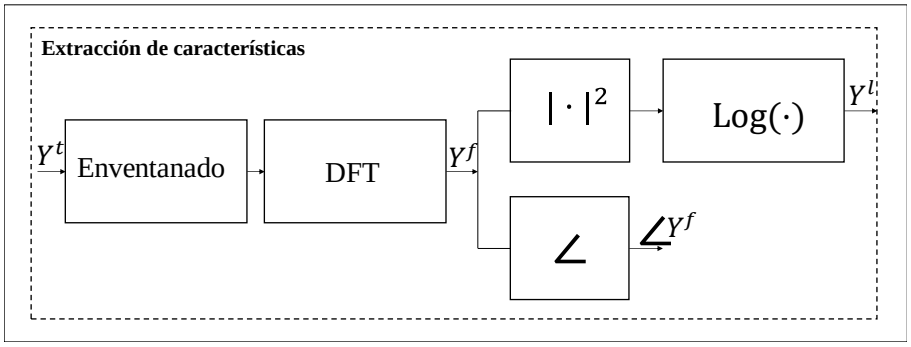

Figura 3.2: Diagrama de bloques del proceso de extracción de características.

Red Neuronal *feed-forward*

La arquitectura adoptada en este modelo es una red neuronal de tipo perceptrón multicapa que permite representar una función de regresión altamente no lineal que mapea las características de voz ruidosas en características de voz limpias. Para ello, utilizará pares de características espectrales de potencia de voz sucia y limpia, obtenidas siguiendo el proceso descrito anteriormente y que deben estar normalizadas a media cero y varianza unidad.

En la sección 2.2.2 se describieron los conceptos teóricos necesarios para comprender el funcionamiento de este tipo de redes, por lo que esta sección será destinada a detallar la estructura e hiperparámetros seleccionados para la red propuesta.

El número de neuronas de la capa de entrada debe ser fijo para el correcto funcionamiento del sistema. Tal y como se ha mencionado en el apartado anterior, cada trama está representada por 257 coeficientes espectrales. Sin embargo, en [13] se demostró que el uso de información de contexto disminuye las discontinuidades y mejora el rendimiento del sistema. Cuanta más información de contexto acústico mayor es el suavizado del habla mejorada y mejor es el sentido de la audición. Sin embargo, utilizar demasiadas tramas da lugar a vectores demasiado grandes, lo que dificulta el aprendizaje de la red neuronal y aumenta el tiempo de entrenamiento. En este caso, se han usado 10 tramas de contexto, las 5 tramas anteriores y las 5 tramas posteriores a la trama objetivo. Estas 11 tramas se concatenarán dando lugar a vectores de 2827 coeficientes espectrales. Así pues, el modelo será capaz de capturar información de contexto temporal, usando vectores de múltiples tramas, e información frecuencial, al usar características espectrales para el entrenamiento de la red. El tamaño de dichos vectores determina el número total de neuronas en la capa de entrada, esto es, 2827. Por su parte, la capa de salida debe tener 257 neuronas, que serán los 257 coeficientes espectrales limpios correspondientes a la trama central. Además, el modelo propuesto consta de 4 capas ocultas de 2048 neuronas.

La función de activación de la capa de entrada y de las capas ocultas es la *ReLU*. Esta función de activación no requiere ningún cálculo exponencial, por lo que asegura un menor cálculo numérico y un entrenamiento más rápido. En la capa de salida la función de activación es la función *Linear*, la cual permite obtener cualquier valor a la salida para las características de voz limpias.

Además, se ha fijado un *Dropout* de 0.3 en cada una de las capas. Esta técnica consiste en desactivar aleatoriamente un porcentaje de las neuronas durante el proceso de entrenamiento para evitar complejas coadaptaciones en los datos de entrenamiento que puedan dar lugar a un sobreajuste u *overfitting* de la red. Esto mejorará la capacidad de generalización del modelo propuesto.

La función de coste para el algoritmo *backpropagation* es el error cuadrático medio (MSE), que será calculado entre las características espectrales estimadas por la red y las características deseadas, de modo que la red se ajuste lo máximo posible al segundo conjunto mencionado. Además, el uso de esta función de coste en el dominio espectral de potencia logarítmica ha demostrado ser consistente con el sistema auditivo humano [26]. El optimizador seleccionado ha sido el Adam con *learning rate* de 0.0001. Este optimizador se caracteriza por ser computacionalmente eficiente y su buen comportamiento frente a datos no estacionarios. Además, funciona bien con grandes conjuntos de entrenamiento.

Este modelo será entrenado durante la etapa de entrenamiento y, posteriormente, será aplicado en la etapa de mejora para la limpieza de los audios. En la evaluación, la entrada a la red debe seguir el mismo formato, es decir, deben calcularse las características de la voz sucia siguiendo el proceso descrito en la Figura 3.2.

Así pues, el modelo propuesto es el sistema *baseline* que se desarrolló en [13][14], el cual se pretende mejorar con la investigación y desarrollo llevados a cabo en este Trabajo Fin de Máster. Por este motivo, se han propuesto dos métodos alternativos de uso de redes neuronales para mejora de voz, que serán abordados en las siguientes secciones.

Reconstrucción de la señal

El modelo de red neuronal descrito anteriormente da como resultado los 257 coeficientes espectrales limpios correspondientes a la trama objetivo. Dado que el módulo de la DFT es simétrico, en la fase de extracción de características se seleccionaron la mitad de los coeficientes espectrales, reduciendo de este modo el conjunto de datos de entrenamiento. Por este motivo, en primer lugar, será necesario recuperar la parte simétrica, mediante el duplicado de los coeficientes espectrales. Además, para el cálculo de la IDFT, debemos recuperar estos coeficientes espectrales en amplitud no logarítmica. A continuación, haciendo uso de las fases de la señal original ruidosa, se calcula la IDFT para cada vector de coeficientes de los que se compone la señal de voz. Finalmente, para reconstruir la señal en el dominio temporal, deben concatenarse las tramas obtenidas tras el cálculo de la IDFT, teniendo en cuenta el enventanado y solapamiento que se realizaron en la extracción de características. La Figura 3.3 muestra el proceso descrito.

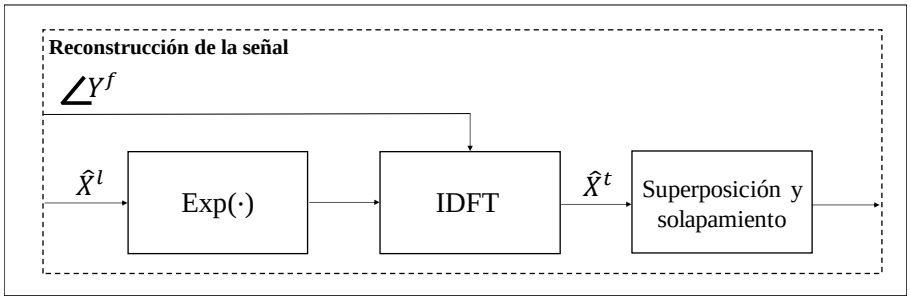

Figura 3.3: Diagrama de bloques del proceso de reconstrucción de la señal.

3.2.2. Sistemas basados en arquitecturas CNN

En la sección 2.2.3 se expuso la arquitectura y el funcionamiento básico de las capas convolucionales en una red neuronal. Un ejemplo típico del uso de capas convolucionales en una red neuronal es la arquitectura de tipo CNN, formada por capas convolucionales en las primeras capas y capas de tipo *fully-connected* en las últimas capas de la red. Las CNNs presentan algunos parámetros de diseño adicionales sobre las FC-DNN: campos receptivos, filtros locales, distribución de pesos y *max-pooling*. Así pues, se ha explorado e investigado el comportamiento de este tipo de redes para reducción de ruido en voz, lo que supone un estudio detallado de los parámetros anteriores y su aplicación en señales de habla. Este sistema se presenta como una primera mejora frente al modelo baseline descrito en el apartado anterior.

Las CNNs son atractivas en comparación con las FC-DNNs para su uso como modelos acústicos por diversas razones.

- En primer lugar, las FC-DNN no están diseñadas explícitamente para modelar la **invarianza espacial** dentro de las señales de voz, que puede existir debido a los diferentes estilos de habla o diferentes tipos de ruido [27]. No obstante, para capturar la invariabilidad espacial, una red FC-DNN debería ser de gran tamaño y usar conjuntos de entrenamiento lo suficientemente grandes. Por el contrario, las CNNs capturan la invariancia espacial con muchos menos parámetros al replicar los pesos de los filtros a través del tiempo y la frecuencia.

- En segundo lugar, las FC-DNNs ignoran las **conexiones locales espectro-temporales** existentes en las señales de voz. En las redes FC-DNN, la variabilidad a lo largo del eje de tiempo y la dependencia entre tramas de voz adyacentes se trata con una ventana de contexto temporal, en la que cada entrada a la red es un vector de tramas de características y la salida es la trama central limpia. En este tipo de redes, las entradas podrían presentarse en cualquier orden (fijo) sin afectar al rendimiento de la red [27], lo que hace que se pierda la conexión espectral. Sin embargo, las señales de voz tienen características de localidad a lo largo del eje de frecuencia, lo que significa que diferentes fonemas tienen concentraciones de energía en diferentes bandas locales a lo largo de dicho eje. Estas concentraciones locales de energía son determinantes para distinguir a los mismos, pero las FC-DNNs no están diseñadas para capturararlas.

> Los filtros que trabajan en regiones locales de frecuencia proporcionarán una manera eficiente de representar estas estructuras locales y sus combinaciones a lo largo de todo el eje de frecuencia se pueden utilizar para reconocer cada fonema. Otro beneficio de los filtros locales es el potencial para lograr una mayor robustez frente a los ruidos ambientales, especialmente cuando los ruidos se concentran solo en partes del espectro, donde los filtros locales en partes relativamente más limpias del espectro aún pueden detectar bien las características del habla para compensar la ambigüedad de las partes ruidosas. Además, si se usan filtros que trabajan en regiones locales espectro-temporales, se podrá detectar la variabilidad en ambos dominios. Las CNNs son capaces de modelar estas estructuras espectro-temporales al permitir que cada neurona de la capa convolucional reciba como entradas las características incluidas en un campo receptivo, lo que representaría un ancho de banda limitado de todo el espectro de voz y un intervalo temporal concreto. Así pues, las CNNs tienen en cuenta las regiones vecinas alrededor de cada unidad espectro-temporal y están diseñadas para centrarse en los patrones locales. Además, las representaciones espectrales del habla tienen fuertes correlaciones, lo que puede ser detectado de forma eficiente con redes CNN [28].

Por estos motivos, el uso de CNNs se presenta como una buena opción para abordar el problema de mejora de voz.

Extracción de características

Para poder capturar la variabilidad a lo largo del eje de frecuencia y detectar estructuras locales (fonemas) en este dominio, necesitamos representar las entradas de voz en una escala de frecuencia que se pueda dividir en varias bandas locales. Por lo tanto, las características espectrales de potencia calculadas en el sistema *baseline* son perfectas para el filtrado local en esta configuración CNN.

Estas características serán de nuevo obtenidas siguiendo el proceso descrito anteriormente. La diferencia con respecto al sistema anterior es la forma en la que se presentan estas características a la red neuronal. Para la FC-DNN las características forman vectores de 11 tramas de coeficientes espectrales, siendo la trama central la trama objetivo y 5 tramas de contexto a cada lado de la misma. Sin embargo, en la red CNN de este sistema la primera capa es una capa convolucional 2D, por lo que dichos vectores de coeficientes deben adaptarse al formato requerido por la misma. Así pues, son transformados en matrices de 2 dimensiones, que en realidad son los espectrogramas de potencia de pequeños segmentos de la señal de voz. En cada matriz de entrada a la red cada columna representará los coeficientes espectrales de una determinada trama. Éstas se formarán concatenando de forma paralela el vector de coeficientes espectrales de la trama objetivo y los vectores de coeficientes de las tramas de contexto, 5 vectores a cada lado de la trama central. A partir de los espectrogramas, la señal de voz puede conservar las conexiones espectro-temporales de la misma y la red podrá detectar los patrones existentes para separar la voz limpia de las muestras de entrada ruidosas de forma más eficiente. Esta transformación se realiza dentro del modelo de red neuronal, por lo que será descrita en la siguiente sección.

Red Neuronal CNN

En este sistema se han probado distintas configuraciones de arquitecturas CNN y se han comparado entre sí. Este tipo de redes tienen la estructura mencionada anteriormente, esto es, varias capas convolucionales a la entrada y una estructura de capas *fully-connected* a la salida. Así pues, introducen nuevos parámetros de diseño, detallados en la sección 2.2.3.

La Figura 3.4 muestra un diagrama de bloques de las estructuras propuestas. Así, los modelos estarán formados por:

- Una capa de ***Reshape***, que se encarga de cambiar la entrada a un formato aceptado por las capas convolucionales. Transforma los vectores de coeficientes en espectrogramas $2D$ de tamaño 257×11, es decir, 11 tramas temporales de 257 coeficientes espectrales cada una.
- **K bloques constituidos por una capa convolucional 2D seguida de una capa de *max-pooling***. En cada bloque se usarán un número f_i de filtros distintos de tamaño constante 3×3 en la capa convolucional. La capa de *max-pooling* tendrá un tamaño de rejilla constante de 2×2 en cada bloque k_i.
- Una capa ***Flatten***, que convierte los mapas de características en una sola columna que formará las entradas a las capas *fully-connected*.
- **Un bloque de capas *fully-connected***, formado por dos capas *Dense* de 2048 neuronas con un *Dropout* de 0.3 y una capa final de 257 neuronas, que actuará como capa de salida de la red propuesta. Para que las estructuras propuestas sean lo más comparables posible con el sistema *baseline*, el número de neuronas en las capas *Dense* y el porcentaje de *Dropout* se ha mantenido invariable. No obstante, se ha reducido el número de capas *Dense* frente al sistema *baseline*, que estaba compuesto por 4 capas ocultas de 2048 unidades, para que no se dispare el número de parámetros entrenables en el modelo final y porque la entrada a las capas *Dense* ya está previamente procesada por las capas convolucionales.

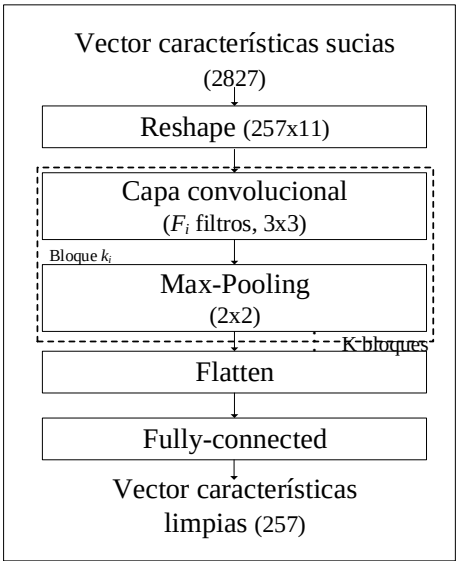

Figura 3.4: Diagrama de bloques de la arquitectura CNN.

En una CNN estándar, los pesos de los filtros locales son compartidos en el espacio de entrada, siguiendo el principio *weight sharing* descrito en la sección 2.2.3. En este caso, el cálculo de todas las unidades de un mapa de características puede verse como una convolución de los pesos de los filtros y las unidades de la capa anterior. Esta técnica funciona muy bien en procesado de imágenes, en la que los filtros pueden aplicarse de forma uniforme en todos los píxeles. Sin embargo, en las señales de voz, las estructuras locales que aparecen en diferentes bandas de frecuencia pueden comportarse de manera muy diferente, lo que sería bastante interesante que pudiera ser capturado por el sistema propuesto. Para abordar esta diferencia entre bandas espectrales, en [29] se propuso utilizar filtros con conjuntos de pesos distintos en cada banda descartando el principio de *weight sharing*. No obstante, eso aumenta significativamente el número de parámetros entrenables en la red y aumentar la complejidad de la red, perdiendo una de las principales ventajas que ofrece el uso de CNNs. Además, este tipo de enfoque limita la adición de capas convolucionales adicionales [30], ya que las salidas de los filtro en diferentes bandas de frecuencia no estarían relacionadas entre sí. Es por esto que este tipo de distribución de pesos normalmente se usa solo en la capa convolucional más alta de la red. Así pues, en [30] se demostró que utilizar filtros con pesos diferentes es equivalente al uso de varias capas convolucionales en cadena con gran cantidad de unidades ocultas para capturar las diferencias entre los componentes de baja y alta frecuencia. Por este motivo, se ha optado por incluir un número K de bloques compuestos por una capa convolucional y una capa de *max-pooling*, lo que permite conservar las ventajas de la técnica *weight sharing* y capturar patrones a alta y baja frecuencia en señales de voz.

La capa de *max-pooling* incluida en cada bloque k_i ayuda a eliminar la variabilidad en las unidades ocultas que existe debido a los distintos estilos del habla, distorsiones de canal, etc. Las capas completamente conectadas agregan la información local aprendida en las capas convolucionales para proporcionar los coeficientes limpios de la trama objetivo.

La función de activación en cada capa oculta de la red es la $ReLU$. De nuevo, en la

capa de salida la función de activación es la función *Linear*, que permite obtener cualquier valor para las características de voz limpias. La función de coste para el algoritmo *backpropagation* es el error cuadrático medio (MSE) y el optimizador seleccionado ha sido *Adam* con *learning rate* de 0.0001.

Para que la búsqueda de hiperparámetros sea asequible se han fijado el tamaño de los filtros de cada capa convolucional y el tamaño de rejilla en las capas de pooling dentro de cada bloque k_i. Así pues se ha modificado el número K de bloques de capas convolucionales y *max-pooling* y el número de filtros f_i en cada capa convolucional dentro de cada bloque k_i. Esto permite analizar cómo afecta la profundidad de la red y el número de mapas de características en los resultados obtenidos.

	Número de bloques K	**Número de filtros en cada capa convolucional**
CNN3x3$_1$	2	$f_1 = 32$ $f_2 = 64$
CNN3x3$_2$	2	$f_1 = 50$ $f_2 = 100$
CNN3x3$_3$	3	$f_1 = 16$ $f_2 = 32$ $f_3 = 64$

Tabla 3.1: Configuraciones de redes CNN. En todas ellas, el tamaño de los filtros de las capas convolucionales es 3x3 y el tamaño de rejilla de las capas de *max-pooling* es 2x2.

La tabla 3.1 muestra una descripción de las configuraciones diseñadas. Posteriormente, en el capítulo 5 se mostrarán los resultados obtenidos con cada una de ellas y cómo se consigue una mejora en el rendimiento en función de los distintos parámetros de diseño.

Reconstrucción

En este caso, la salida que devuelve la red neuronal mantiene el mismo formato que la del sistema *baseline*, es decir, la trama central del espectrograma. Además, para las entradas a la red se han usado 5 tramas de contexto a ambos lados de la misma. Esto significa que la reconstrucción de la señal al dominio temporal debe realizarse siguiente el proceso esquematizado en la Figura 3.3, y que fue descrito con detalle en la sección 3.2.1. Debe tenerse en cuenta el enventanado y solapamiento que se realizaron en el preprocesado de las señales de voz originales.

3.2.3. Sistemas basados en arquitecturas FCN

En el sistema anterior se ha propuesto un modelo de red neuronal con arquitectura CNN. Otro ejemplo típico de uso de capas convolucionales en redes neuronales son las redes FCN. Estas últimas son muy parecidas a las CNNs convencionales, con la diferencia de que las capas superiores de la red son capas convolucionales en lugar de capas *fully-connected*, por lo que estarán formadas por capas convolucionales en toda su arquitectura. Como segunda mejora al modelo *baseline* inicial se pretende implementar un modelo de

red neuronal FCN en este nuevo sistema, que reduzca las limitaciones encontradas en el sistema anterior.

Inicialmente, se proponen dos mejoras con respecto al sistema de referencia.

- La principal ventaja por la que se propuso el uso de CNNs en el sistema anterior es la capacidad de éstas para modelar estructuras espectro-temporales analizando regiones vecinas alrededor de cada unidad en un mapa de características. Sin embargo, en las CNNs aún se conservan capas *fully-connected* en las últimas capas de la red, que pueden no caracterizar con precisión la información local de las señales del habla, particularmente en componentes de alta frecuencia. Esto provoca que las CNNs tengan una capacidad limitada para restaurar componentes de alta frecuencia, disminuyendo de esta forma la inteligibilidad de la voz mejorada. Por el contrario, las FCNs solo constan de capas convolucionales, por lo que cada muestra de salida dependerá localmente de las regiones de entrada vecinas y las estructuras espectro-temporales locales de las señales del habla se pueden preservar de manera eficiente y efectiva con relativamente pocos pesos.
- Este tipo de redes, se ha usado recientemente en [31] para procesar los espectros de potencia de las señales de voz en tareas de mejora de habla. En estas redes, tal y como se detalló en el capítulo 2, aparecen implicadas convoluciones entre las muestras de entrada y los filtros de cada capa. Así pues, dado que el efecto de convolucionar una señal de dominio de tiempo $x(t)$ con un filtro $h(t)$ equivale a multiplicar su representación de frecuencia $X(f)$ con la respuesta de frecuencia del filtro $H(f)$ [32], puede ser innecesario mapear explícitamente la forma de onda en el dominio del tiempo al espectrograma de potencia para mejorar el habla. Por este motivo, se propone el uso de las señales de voz en su forma de onda original, dominio del tiempo, para el problema de mejora de voz.

Así pues, se ha diseñado un sistema *end-to-end* para mejora del habla que utiliza redes FCN para modelar las señales de voz y mapear las muestras ruidosas en muestras limpias.

Extracción de características

El objetivo de la tarea de mejora de voz es incrementar la inteligibilidad y la calidad de una señal de voz ruidosa. Debido a que las propiedades en el dominio log-espectral son más consistentes con el sistema auditivo humano, convencionalmente, se utiliza el logaritmo del espectro de potencia para modelos de eliminación de ruido basados en aprendizaje profundo [13][14]. Sin embargo, emplear el módulo de los espectros de potencia como características produce dos inconvenientes.

En primer lugar, la fase mantiene su forma de onda original, es decir, cuando la señal de voz mejorada se sintetiza de nuevo en el dominio del tiempo, las componentes de fase simplemente se toman prestadas de la voz ruidosa original, lo que puede degradar la calidad perceptiva de la voz mejorada. Además, estudios recientes han revelado la importancia de la calidad de la fase a la voz cuando la voz se vuelve a sintetizar nuevamente en las ondas de dominio del tiempo [33].

En segundo lugar, una gran cantidad de preprocesamiento y postprocesamiento son necesarios para el mapeo entre los dominios de tiempo y frecuencia, lo que aumenta la carga computacional y disminuye la eficiencia del sistema.

Así pues, se trata de un sistema de mejora de voz basado en la forma de onda original de la señal, sin procesar. Además, se propone el uso de los audios completos (sin un proceso de enventanado y división en tramas) como entradas a la red neuronal, ya que los resultados de convolución en tramas con relleno cero pueden ser inexactos y perderse información. Esto es posible a pesar de que los audios puedan tener longitudes diferentes. Como todas las capas *fully-connected* se eliminan en FCN, la longitud de las características de entrada no tiene que ser fijada para la multiplicación de matrices. Por otro lado, los filtros en las operaciones de convolución pueden procesar entradas con diferentes longitudes. Específicamente, si la longitud del filtro es l y la longitud de la señal de entrada es L (sin relleno), entonces la longitud de la salida filtrada es $L - l + 1$.

Esta aproximación no ha sido muy explorado en mejora de voz, por lo que este Trabajo Fin de Máster pretende innovar en este ámbito.

Red Neuronal FCN

En la sección anterior, se han expuesto los motivos por lo que las capas *fully-connected* pueden no modelar con precisión las señales de voz. Por lo tanto, en este sistema, se plantea aplicar redes FCN para llevar a cabo la tarea de mejora de voz. Se han propuesto distintas configuraciones de redes FCN que modelan el habla en su forma de onda original, dando lugar a sistema *end-to-end* para esta tarea.

La Figura 3.5 muestra un diagrama de bloques de las configuraciones propuestas. Así, los modelos estarán formados por:

- **K bloques constituidos por una capa convolucional 1D seguida de una de *Batch Normalization***. Esta última normaliza las activaciones de la capa convolucional anterior en cada entrenamiento, es decir, aplica una transformación que mantiene la activación media cerca de 0 y la desviación estándar alrededor de 1. Las capas convolucionales de los K bloques tendrán el mismo número de filtros y el mismo *kernel size*. Además, todos los bloques usan la función de activación $LeakyReLU$ en las capas convolucionales. La capa convolucional del primer bloque actuará como capa de entrada y será la encargada de recibir los audios completos sin procesar.
- **Una capa convolucional como capa de salida**, formada por un único filtro con un *kernel size* igual al de las capas anteriores. Ésta capa devolverá el audio limpio por la red.

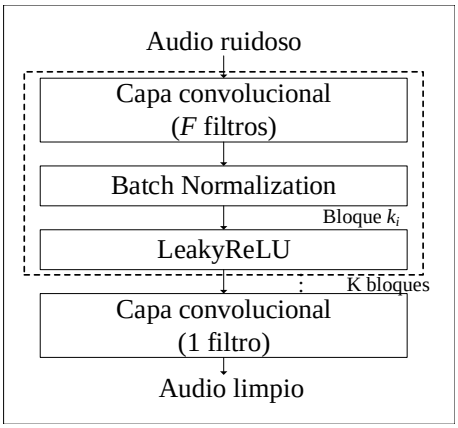

Figura 3.5: Diagrama de bloques de la arquitectura FCN.

El optimizador utilizado ha sido *Adam* con *learning-rate* de 0.0001 y función objetivo MSE. Además, para que el audio a la salida de la red mantenga la misma longitud que a la entrada se ha fijado en la capa convolucional el argumento *padding* igual a '*same*', que utiliza relleno con ceros para tal propósito.

Así pues, en las distintas configuraciones se ha modificado el número de bloques K, el número de filtros, las funciones de activación en la capa de salida y el tamaño de los filtros. Esto permitirá realizar un estudio del efecto de estos parámetros en el modelo. La tabla 3.2 muestra una descripción de las configuraciones diseñadas.

	Número de bloques K	**Número de filtros en cada capa convolucional**	***Kernel size***	**Función activación capa de salida**
FCN$_1$	8	30	55	Linear
FCN$_2$	8	30	55	tanh
FCN$_3$	8	90	55	Linear
FCN$_4$	8	90	55	tanh
FCN$_5$	4	30	110	Linear
FCN$_6$	5	30	110	Linear
FCN$_7$	16	30	27	Linear

Tabla 3.2: Configuraciones de redes FCN.

La ventaja de no usar capas *fully-connected* es que el número de parámetros en la red se puede reducir drásticamente, lo que hace que las FCNs sean particularmente adecuadas para implementaciones en dispositivos móviles con capacidad de almacenamiento limitada.

Reconstrucción

Dado que la capa de salida de la red propuesta es una capa convolucional con un único filtro, la salida de la misma será el audio limpio. Por este motivo, este sistema no requiere de un proceso de reconstrucción, lo que disminuye el tiempo de cómputo y aumenta la eficiencia del sistema.

Capítulo 4

Entorno experimental

En este capítulo se describirán las Bases de Datos que hemos usado para la evaluación de los sistemas desarrollados. Además, se expondrán las herramientas utilizadas para la implementación de los mismos. Por último, se realizará una breve descripción de las medidas de rendimiento elegidas para estudiar el comportamiento de los sistemas propuestos.

4.1. Bases de Datos

Para la evaluación y valoración de los sistemas diseñados se dispone de una base de datos de voz, ALBAYZIN, y de una base de datos de ruidos, HU. Estas bases de datos han sido combinadas para la generación de la base de datos de habla contaminada.

4.1.1. Base de datos de voz: ALBAYZIN

ALBAYZIN es una base de datos en español diseñada originariamente para tareas de reconocimiento de voz [35]. Fue producida en 1998 tras haber sido diseñada en 1991-93 por un consorcio de 6 grupos de investigación españoles liderados por el grupo de Procesamiento del Habla de la UPC (Universidad Politécnica de Catalunya) que actualmente se denomina Centro TALP (Centro de Tecnologias y Aplicaciones del Lenguaje y el Habla). El corpus está siendo distribuido por ELRA (*European Language Resources Association*).

Corpus fonético

El corpus de la base de datos fonética se divide en dos subconjuntos, que han sido diseñados de forma independiente.

- Subcorpus 1. Se genera mediante la elocución repetida de 200 oraciones fonéticamente equilibradas. En concreto, 4 locutores pronuncian las 200 frases anteriores y 160 locures pronuncian un subconjunto de 25. Este subcorpus está destinado a fines de entrenamiento de sistemas acústico-fonéticos.

- Subcorpus 2. Se genera a partir de un conjunto de 500 oraciones fonéticamente equilibradas. En concreto, 40 locutores pronuncian 50 frases cada uno. Este subcorpus está destinado principalmente a la evaluciación o test de los sistemas.

El corpus fonético está formado por un total de 6800 oraciones fonéticamente equilibradas, incluyendo 1000 con segmentación fonética, de 204 locutores diferentes.

Este ha sido el corpus seleccionado para el entrenamiento y valoración de los sistemas desarrollados en este Trabajo Fin de Máster.

Corpus geográfico

Este corpus está formado por frases correspondientes a consultas a una base de datos geográfica. Las frases son sometidas a una fuerte restricción semántica, al incluir información relativa a este sector. Este corpus también está dividido en dos subcorpus.

- Subcorpus 1. Se genera mediante la elocución repetida de 50 frases pronunciadas por 88 locutores. Está destinado al entrenamiento de sistemas.
- Subcorpus 2. Se genera a partir de un conjunto de 50 oraciones pronunciadas por 48 locutores. Este subcorpus está destinado principalmente a la evaluciación o test de los sistemas.

El corpus geográfico está formado por un total de 6800 oraciones de 136 locutores diferentes.

Corpus Lombard

La base de datos Lombard está compuesta por un subconjunto de las bases de datos mencionadas anteriormente y es pronunciada por 40 hablantes. Durante las grabaciones, se aplica una fuente de ruido a los altavoces a través de auriculares para producir el efecto Lombard.

Así pues, el total de frases ha sido pronunciado por 304 locutores diferentes, con igual número de hombres que de mujeres. El rango de edad de los participante va de 18 a 55 años. El número total de frases es de 15600 y tienen una duración promedio de aproximadamente 4 segundos. Además, están muestreadas a 16 kHz y codificadas con 16 bits.

4.1.2. Base de datos de ruido: HU

HU es una base de datos de sonidos ambientales que pueden usarse como ruidos en sistemas de segregación del habla. Fue recogido por Guoning Hu durante el desarrollo de

su tesis doctoral [34]. Esta base de datos puede descargarse de forma totalmente gratuita en un archivo con formato ZIP.

Está compuesta por un total de 100 sonidos. Algunos de ellos son: alarmas y sirenas, tráfico, sonidos de animales, sonido de agua, viento, ruido de máquinas, campanas, tos, aplausos, risas, marcación telefónica, etc. Además, la frecuencia de muestreo es de 16 kHz.

4.2. Herramientas

4.2.1. TensorFlow y Keras

TensorFlow es una biblioteca de software de código abierto para cálculo numérico y aprendizaje automático a gran escala. Fue desarrollada originariamente en noviembre de 2015 por el Equipo de Google Brain. TensorFlow puede entrenar y ejecutar redes neuronales profundas para tareas de clasificación, reconocimiento de imágenes, redes neuronales recurrentes, modelos de secuencia a secuencia para traducción automática, procesamiento de lenguaje natural, etc. Utiliza Python para proporcionar una API *front-end* para la creación de aplicaciones, mientras ejecuta esas aplicaciones en C++. Además, TensorFlow puede funcionar sobre múltiples CPUs y GPUs.

Keras es una biblioteca de alto nivel escrita en Python para el desarrollo de modelos de aprendizaje profundo. Para realizar operaciones de bajo nivel, Keras se basa en una biblioteca especializada y bien optimizada de manipulación de tensores que sirve como el "motor de fondo"de la misma. Keras permite conectarse a diferentes *backend*, tales como TensorFlow, CNTK (*Microsoft Cognitive Toolkit*) o Theano, para realizar las operaciones a bajo nivel. Está especialmente diseñada para la creación en poco tiempo de redes neuronales profundas, gracias a su formato modular, extensible y de fácil uso.

En este Trabajo Fin de Máster se ha utilizado Keras sobre TensorFlow en Python para el diseño e implementación de los sistemas propuestos. Además, se ha trabajado con dos GPUs, que han permitido un entrenamiento rápido y eficiente de los distintos modelos.

4.2.2. Matlab

MATLAB, abreviatura de *MATrix LABoratory*, es una herramienta de cómputo numérico que ofrece un entorno de desarrollo integrado (IDE) con un lenguaje de programación propio (lenguaje M). Este último permite trabajar en formato matricial y representaciones gráficas, entre otros.

En este Trabajo Fin de Máster se ha utilizado Matlab para la evaluación de los sistemas propuestos mediante el cálculo del STOI, el cual ha sido implementado en dicha herramienta. Además, gran parte de las representaciones se han realizado en este sistema.

4.2.3. Software FANT

FANT, es una herramienta que permite la adición de ruido a señales de voz a una determinada SNR, que es controlable por el usuario.

Deben usarse archivos de voz y ruido muestreados a 8 o 16 kHz con muestras almacenadas como enteros con signo de 16 bits. Además, deben estar en formato RAW. Los archivos de audio de salida se producirán con las mismas características. El software ha sido escrito en C. Las funcionalidades que permite se pueden controlar dando parámetros opcionales mediante la línea de comandos cuando se llama al programa.

Esta herramienta ha permitido la combinación de las Bases de Datos mencionadas en secciones anteriores, dando lugar a una Base de Datos de audios ruidosos. Así pues, se han utilizado *shell-scripts* para la llamada al software FANT y la adición de ruido en señales de voz.

4.3. Medidas de rendimiento

4.3.1. PESQ

La evaluación perceptiva de la calidad del habla (*Perceptual Evaluation of Speech Quality*, PESQ) [37], es un método objetivo para valorar la calidad subjetiva de la voz, definido en el estándar ITU-T P.862. Se basa en la comparación entre una señal limpia, $x[n]$, y una versión degradada de la misma, $y[n]$. La salida de PESQ será predicción de la calidad percibida por varios sujetos en una escucha subjetiva de $y[n]$. La intención de PESQ es reproducir la percepción del oído humano.

A continuación, se presenta una descripción de la estructura de PESQ y de los procesos clave que incluye. La Figura 4.1 muestra un diagrama de bloques de su funcionamiento.

1. El modelo alinea ambas señales a un nivel de escucha estándar (alineación de nivel) y las filtra con un filtro de entrada que modela un auricular de teléfono.

2. Se realiza una alineación temporal entre las señales $x[n]$ e $y[n]$. Para ello calcula unos retardos definidos por un punto de inicio y un punto de parada.

3. Una vez alineadas, se realiza una transformación auditiva mediante un modelo perceptual que mapea las señales en diferentes representaciones acústicas que intentan reproducir al sistema auditivo humano. La calidad de la señal distorsionada es calificada en base a estas representaciones.

4. Se extraen dos parámetros fundamentales basados en la diferencia entre las transformaciones de las señales, y que son agregados en frecuencia y tiempo.

5. Finalmente, el modelo cognitivo devuelve una distancia entre la señal original y la señal degradada, conocido como nota PESQ. Esta se corresponde al mismo tiempo con una predicción de la MOS subjetiva. La nota PESQ sigue una escala similar a la de MOS, comprendida entre 1 y 5. Así pues, a mayor valor de PESQ mayor similitud habrá entre las señales x[n] e y[n]. Un valor de 5 significa que la señal y[n] no tiene distorsión, mientras que un valor de 1 significa que existe una degracion severa.

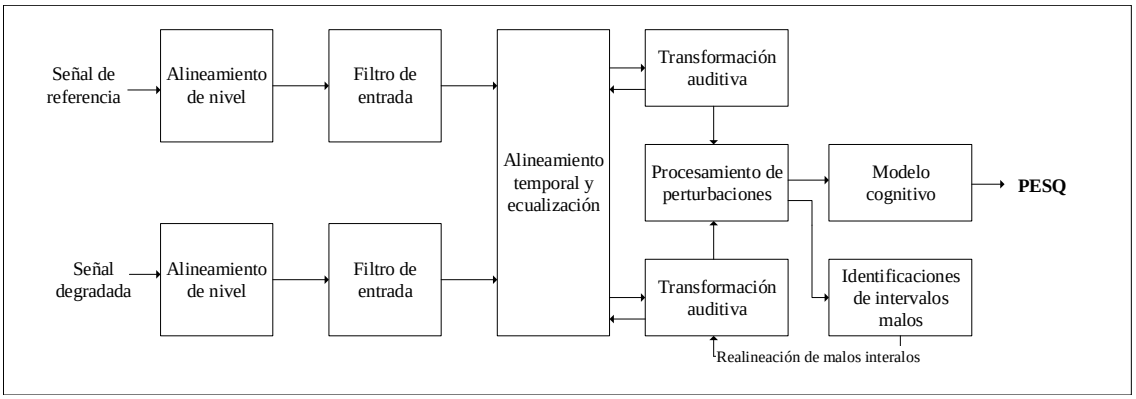

Figura 4.1: Estructura del modelo de evaluación perceptiva de la calidad del habla (PESQ).

4.3.2. STOI

La inteligibilidad objetiva a corto plazo (*Short-Time Objective Intelligibility*, STOI) [36] es una función del habla limpia y degradada. El puntaje STOI es una medida frecuentemente utilizada para predecir la inteligibilidad del habla ruidosa o procesada. La salida de STOI es un valor escalar, comprendido entre 0 y 1, y se espera que esté relacionado monotónicamente con la inteligibilidad promedio de varias pruebas de audición sobre y[n]. Por lo tanto, un valor más alto indica una mejor inteligibilidad del habla.

El proceso computacional general se ilustra como en la Figura 4.2 e incluye 5 pasos principales, que se describen brevemente a continuación:

1. Se eliminan los instantes de silencio. Dado que las regiones silenciosas no contribuyen a la inteligibilidad del habla, se eliminan antes de la evaluación.

2. Transformada de Fourier a corto plazo (STFT). Ambas señales se descomponen en la Transformada de Fourier para obtener una representación similar a las propiedades de representación del habla en el sistema auditivo. Esto se obtiene realizando un envetanado con solapamiento del 50 %. Cada trama tendrá una longitud de 256 muestras y se realizará relleno cero hasta las 512 muestras. La ventana utilizada será la ventana Hanning.

3. Análisis de la banda de un tercio de octava [1]. Esto se realiza simplemente agrupando los vectores de coeficientes espectrales. En total, se utilizan 15 bandas de un tercio de octava, donde la frecuencia central más baja se establece en 150 Hz y la más alta en aproximadamente 4.3 kHz. Se calcula la envolvente temporal a corto plazo de la señal limpia y degradada en cada una de esas bandas. La siguiente notación vectorial se usa para denotar la envolvente temporal a corto plazo de la voz limpia.

[1]Una banda de frecuencias es una zona del espectro definida por dos frecuencias límite, inferior y superior, y una frecuencia central. El ancho de dicha banda viene definido por la diferencia entre las dos frecuencias límite. En una banda de octavas la frecuencia superior es el doble de la inferior. Una banda de tercios de octava es la tercera parte de una banda de octava.

$$x_{j,m} = [X_j(m-N+1), X_j(m-N+2), ..., X_j(m)]^T \tag{4.1}$$

donde $X \in R^{15*M}$ es la banda de un tercio de octava, M es el número total de tramas, que dependerá de la longitud de la señal, m es el índice de una trama concreta, $j \in 1, 2, ..., 15$ es el índice de una determinada banda de octava y N=30, lo que equivale a una longitud de análisis de 384 ms. De mismo modo, $y_{j,m}$ denota la envolvente temporal a corto plazo de la señal de voz degradada.

4. Normalización y recorte. El objetivo del proceso de normalización es compensar las diferencias de nivel, que no deberían tener un fuerte efecto en la inteligibilidad del habla. El proceso de recorte asegura que la sensibilidad en la evaluación de STOI hacia una unidad espectral severamente degradada sea superior. Así pues, se obtiene la envolvente temporal normalizada y recortada de la señal degradada, $\overline{y}_{j,m}(n)$.

$$\overline{y}_{j,m}(n) = min(\frac{\| x_{j,m} \|}{\| y_{j,m} \|}, (1 + 10^{-\beta/20})x_{j,m}(n)) \tag{4.2}$$

donde x(n) denota el n-ésimo elemento de x, $n \in \{1, ..., N\}$, y $\beta = -15$ dB

5. Medida de inteligibilidad. Se calcula el coeficiente de correlación entre las dos envolventes temporales en cada banda y trama, que viene dado por la ecuacion 4.3).

$$d_{j,m} = \frac{(x_{j,m} - \mu_{x_{j,m}})^T(\overline{y}_{j,m} - \mu_{\overline{y}_{j,m}})}{\| x_{j,m} - \mu_{x_{j,m}} \| \| \overline{y}_{j,m} - \mu_{\overline{y}_{j,m}} \|} \tag{4.3}$$

Finalmente, STOI se calcula como el promedio de los coeficientes de correlación calculados en todas las bandas y tramas (veáse ecuación 4.4.

$$d = \frac{1}{JM} \sum_{j,m} d_{j,m} \tag{4.4}$$

donde J es el número total de bandas de un tercio de octava.

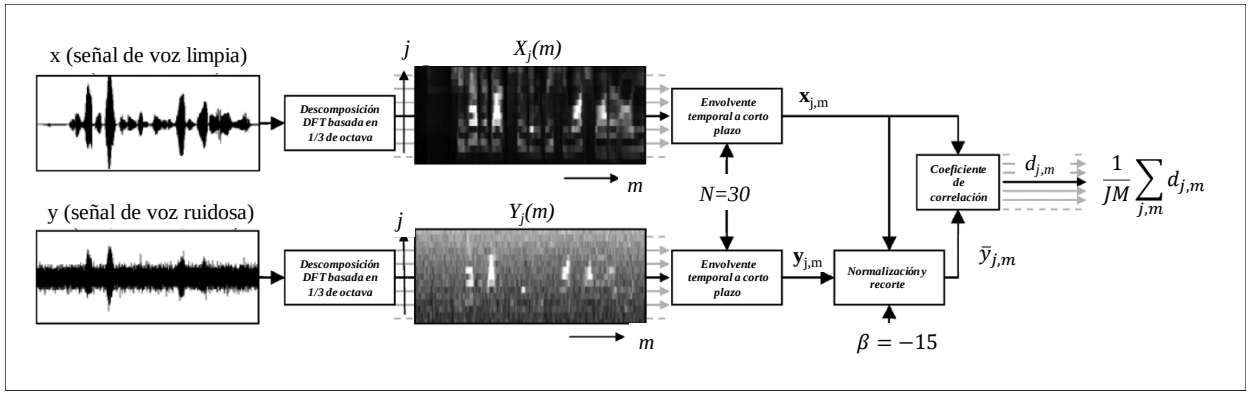

Figura 4.2: Estructura del modelo de cálculo de STOI.

Capítulo 5

Resultados

En este capítulo se mostrarán los resultados obtenidos al aplicar los sistemas diseñados sobre los audios de la Base de Datos ruidosa. Se utilizarán las medidas de rendimiento mencionadas en el capítulo anterior para realizar una comparación entre los mismos. Inicialmente, se evaluará el sistema *baseline* para generar el *benchmark* de referencia que deben superar los sistemas propuestos. Posteriormente, los dos sistemas propuestos se evaluarán sobre las mismas condiciones empíricas que el sistema anterior. De esta forma, se compararán los resultados obtenidos con las distintas configuraciones.

5.1. Descripción de los resultados obtenidos

El éxito de cualquier modelo de aprendizaje automático depende de la algoritmia empleada en el desarrollo del mismo, de la capacidad de automatización y del conjunto de datos de entrenamiento. Sin embargo, en última instancia, ningún nivel de sofisticación algoritmia es capaz de compensar un conjunto de datos pobre. La calidad de los datos y la variedad de los mismos son esenciales en el entrenamiento de cualquier modelo. Es necesario un gran conjunto de datos y una amplia representación de los posibles entornos ruidosos para que el modelo sea eficiente, preciso y con una alta capacidad de generalización.

Así pues, el proceso de preparación de los datos no es una tarea fácil. La forma en la que éstos se presentan al modelo y los parámetros seleccionados durante el entrenamiento también juegan un papel fundamental en el desempeño del mismo.

Uno de los desafíos en el diseño de sistemas de mejora del habla es abordar los posibles desajustes entre las condiciones de entrenamiento y test, causados por diferentes niveles de SNR, variabilidad del hablante, tipos de ruido, etc. Sin embargo, la falta de coincidencia en los tipos de ruido es el más difícil, ya que hay muchos tipos de entornos ruidosos en el mundo real. Por este motivo, en este Trabajo se ha optado por combinar oraciones de distintos locutores con una gran variedad de ruidos a distintos niveles de SNR, dando lugar a una base de datos estéreo de voz sucia con múltiples condiciones y su correspondiente base de datos de voz limpia. La muestra resultante podría ser lo suficientemente representativa de un entorno ruidoso real.

Para construir la base de datos de voz sucia que se utilizará en el entrenamiento de los modelos se han utilizado el subconjunto de entrenamiento del corpus fonético de ALBAYZIN y la base de datos de ruidos HU. Dado que la base de datos de voz limpia de entrenamiento está formada por 4.800 oraciones y disponemos de 100 ruidos que queremos mezclar con 5 niveles de SNR diferentes, el número de posibles combinaciones es de 2,4 millones. Así pues, para producir todas estas combinaciones se ha desarrollado un *software* de generación de datos sintéticos que usa el programa FANT (veáse la sección 4.2.3) y que permite seleccionar un determinado número de horas de entrenamiento, esto es, la cantidad de datos que queremos utilizar para el proceso de aprendizaje.

De esta forma, este software nos ha permitido generar cantidades de datos de hasta 1 Terabyte de espacio, que han sido almacenados en un disco duro de gran capacidad. Sin embargo, cargar todos estos datos en memoria RAM durante el entrenamiento no resulta eficiente por las limitaciones tecnológicas, pues provocaría retardos en los sistemas. Por este motivo, se ha hecho uso de la función *fit_generator*, disponible en la librería Keras, que va seleccionando subconjuntos de datos y los va proporcionando al modelo poco a poco, de modo que al finalizar cada época[1] (*epoch*) de entrenamiento el modelo haya sido capaz de recibir todo el conjunto de datos. Además, dicha función memoriza internamente cuál fue el último subconjunto que seleccionó, para no proporcionar al modelo datos repetidos durante una misma época.

Así pues, para entrenar los modelos propuestos en cada sistema será necesario adaptar la base de datos de voz sucia al formato y condiciones requeridas por cada uno de ellos. Para los modelos de los dos primeros sistemas será necesario calcular los coeficientes espectrales de la voz sucia y limpia, mientras que para el tercer sistema no se necesita ningún preprocesado previo.

Para la evaluación de los sistemas, también será necesario generar una base de datos de voz sucia y su correspondiente base de datos de voz limpia, sobre las que aplicaremos los distintos sistemas y las medidas de rendimiento mencionadas en el capítulo anterior. Para construir la versión ruidosa se realizará una combinación de la base de datos ALBAYZIN, subconjunto de test, y los ruidos de HU a los mismos niveles de SNR utilizados en el entrenamiento. Esto generará una base de datos de oraciones pronunciadas por diferentes hablantes con los mismos entornos ruidosos. Así pues, no se han cambiado las condiones ruidosas de entrenamiento y test, pues nuestro propósito es evaluar el funcionamiento de las diferentes arquitecturas y de los diferentes tipos de redes neuronales, y no tanto el rendimiento de las mismas frente a diferentes condicones ambientales en entrenamiento y test. Se han seleccionado conjunto de datos de entrenamiento lo suficientemente grandes como para que los modelos tengan una buena capacidad de generalización.

El objetivo de los experimentos llevados a cabo durante el desarrollo del Trabajo es encontrar la arquitectura de red neuronal que mejor funcione, en base a un equilibrio entre los siguientes requisitos:

- Proporcionar buenos resultados en términos de PESQ y STOI.
- Emplear un reducido tiempo de cómputo en preprocesado y postprocesado y requerir

[1] El número de épocas es un hiperparámetro que define el número de veces que el algoritmo de aprendizaje iterará sobre todo el conjunto de datos de entrenamiento. El modelo actualizará sus pesos al finalizar cada época de entrenamiento.

poco espacio y memoria.

- Estar formada por el menor número posible de parámetros entrenables, para disminuir la complejidad del modelo.
- Invertir poco tiempo en el proceso de entrenamiento.

A continuación, se detallará el procedimiento seguido y los resultados obtenidos con las diferentes configuraciones mencionadas en el capítulo 3.

5.1.1. Resultados obtenidos con el sistema *baseline*

Este sistema fue propuesto en [13] y ha sido un punto de transición entre las técnicas de mejora de voz tradicionales y los métodos más novedosos basados en redes neuronales. El modelo de red neuronal propuesto, descrito en la sección 3.2.1 es una red FC-DNN formada por 4 capas ocultas que trabaja sobre los coeficientes espectrales de las señales de voz.

El elevado número de hiperparámetros existentes en estos últimos métodos nos ha llevado a realizar un estudio previo que permita fijar el valor de algunos de ellos y reducir, de esta forma, el número posible de combinaciones en cada arquitectura de redes neuronales. Este estudio se ha realizado en este modelo, por ser el punto de partida para el desarrollo de este Trabajo. Se han realizado dos experimentos que permitirán fijar la cantidad de datos de entrenamiento necesaria para que el modelo sea eficiente (número de horas de entrenamiento) y el número de épocas, respectivamente. Una vez seleccionados estos dos parámetros, serán utilizados en el resto de experimentos, lo que hará que los mismos sean más comparables entre sí.

Selección del número de horas de datos de entrenamiento

Inicialmente variamos el número de horas de entrenamiento, para comprobar la influencia que tiene la cantidad de datos con la que es entrenado un modelo en los resultados obtenidos. En este primer experimento se ha fijado a 50 el número de épocas en las tres pruebas, y así comparar los resultados obtenidos. La Tabla 5.1 muestra los resultados de PESQ y STOI para los audios ruidosos a diferentes niveles de SNR, sin haberle aplicado ninguna mejora, y para los audios mejorados con tres configuraciones de la red FC-DNN propuesta, esto es, con entrenamientos de 10h, 100h y 1000h. En verde aparecen aquellos valores de PESQ y STOI que superan el valor de los mismos en los audios ruidosos, sin aplicar ningún modelo sobre estos. Por el contrario, en rojo aparecen los valores de PESQ y STOI de los conjuntos de audios limpios que, aun habiéndoles aplicado la red FC-DNN, no mejoran con respecto a los audios ruidosos, lo que quiere decir que la red degrada aún más los audios de entrada a la misma. Además, también aparece el número de parámetros entrenables con esta arquitectura de red neuronal.

	Audios ruidosos		**FC-DNN (50 epochs)**					
			10h		**100h**		**1000h**	
SNR (dB)	**PESQ**	**STOI**	**PESQ**	**STOI**	**PESQ**	**STOI**	**PESQ**	**STOI**
0	1,59	0,66	1,68	0,69	2,29	0,70	2,64	0,73
5	1,90	0,77	1,99	0,72	2,60	0,74	2,81	0,75
10	2,23	0,86	2,23	0,78	2,76	0,80	2,96	0,82
15	2,58	0,92	2,41	0,81	2,88	0,82	3,09	0,85
20	2,93	0,96	2,55	0,83	2,98	0,84	3,19	0,89
# parámetros entrenables			18.9M					

Tabla 5.1: Comparación del rendimiento del modelo FC-DNN con respecto a los valores de PESQ y STOI. El número de *epochs* en todos los casos es 50.

Analizando los resultados obtenidos (veáse Tabla 5.1), podemos comprobar que a mayor número de horas de datos empleado en el entrenamiento, mejores son los resultados en términos de PESQ y de STOI. No obstante, el tiempo de entrenamiento se dispara notablemente. Si seleccionamos una base de datos demasiado grande el tiempo dedicado a cada experimento será muy alto y estaremos incumpliendo uno de los requisitos impuesto en la selección del mejor modelo. Como se ha mencionado anteriormente, buscamos un modelo que funcione bien sobre los datos de entrenamiento sin que tenga que ser entrenado con conjuntos de datos muy grandes, esto es, que el tiempo de entrenamiento sea lo menor posible sin afectar al rendimiento del sistema. Además el objetivo de este Trabajo es valorar el funcionamiento de diferetes topologías de redes neuronales en la tarea de mejora de voz. Así pues, se han seleccionado 10h de base de datos para realizar experimentos con los dos siguientes sistemas, lo que nos permitirá probar más arquitecturas y configuraciones en los mismos. Asimismo, suponemos que el tamaño de la base de datos tiene el mismo comportamiento con el resto de topologías. No obstante, para comprobar la validez de esta hipótesis, sería necesario extender los experimentos para un mayor número de horas en cada uno de lo sistemas, pero esto no ha sido posible por el elevado coste computacional que requiere.

Selección del número de época de entrenamiento

Para fijar el número de épocas vamos a realizar un barrido entre 4 valores diferentes. Las Tabla 5.2 muestra los resultados de PESQ y STOI a distintos niveles de SNR para los audios ruidosos y para los audios mejorados con entrenamientos de 10, 20, 50 y 100 épocas. Para este experimento se ha hecho uso de la función *ModelCheckpoint* de Keras, que permite almacenar modelos intermedios tras un número seleccionado de épocas de entrenamiento. Esto hace que no tenga que repetirse el mismo entrenamiento cuatro veces distintas. De nuevo, en verde aparecen aquellos valores que mejoran con respecto a los audios ruidosos y en rojo aquellos que, aun habiéndoles aplicado la red neuronal, no mejoran con respecto a los mismos.

A mayor número de épocas mayor es el tiempo de entrenamiento, pero este aumento es menor que el que genera una mayor cantidad de datos. Por este motivo, se ha fijado a

100 el número de épocas, pues es el que mejores resultados ofrece para PESQ y para STOI (veáse Tabla 5.2). La red podrá actuar sobre todo el conjunto de datos de entrenamiento un número de veces suficiente y ser capaz de aprender los patrones y comportamientos presentes en los audios de entrada.

	Audios ruidosos		**FC-DNN (10h)**							
			10ep		**20ep**		**50ep**		**100ep**	
SNR (dB)	**PESQ**	**STOI**	**PESQ**	**STOI**	**PESQ**	**STOI**	**PESQ**	**STOI**	**PESQ**	**STOI**
0	1,59	0,66	1,50	0,66	1,59	0,67	1,68	0,69	1,79	0,71
5	1,90	0,77	1,88	0,68	1,90	0,69	1,99	0,72	2,06	0,79
10	2,23	0,86	2,12	0,71	2,14	0,75	2,23	0,78	2,30	0,85
15	2,58	0,92	2,26	0,73	2,32	0,79	2,41	0,81	2,50	0,88
20	2,93	0,96	2,40	0,76	2,46	0,81	2,55	0,83	2,65	0,90
# parámetros entrenables			18,9M							

Tabla 5.2: Comparación del rendimiento del modelo FC-DNN con respecto a los valores de PESQ y STOI. El número de horas de entrenamiento en todos los casos ha sido 10.

Además, con este número de épocas seleccionado la red no llega a producir sobreentrenamiento u *overfitting*, comportamiento que es importante tener en cuenta cuando se entrenan modelos de *deep learning*. Así pues, un modelo de red neuronal estará sobreentrenado cuando a partir de un determinado número de épocas, el coste sobre el conjunto de datos de entrenamiento disminuye, pero comienza a aumentar sobre el conjunto de validación. La gráfica de la Figura 5.1 muestra el comportamiento del modelo (red FC-DNN con 10h de datos de entrenamiento) sobre ambos conjuntos. Cuando el modelo ha alcanzado la época 100, la función de coste sigue disminuyendo en los dos casos, por lo que podemos seleccionar un valor de 100 épocas sin que el modelo llegue a producir *overfitting*. Cabe destacar que, tal y como ocurre en la imagen mencionada, normalmente el coste en los datos de entrenamiento es inferior al de los datos de validación.

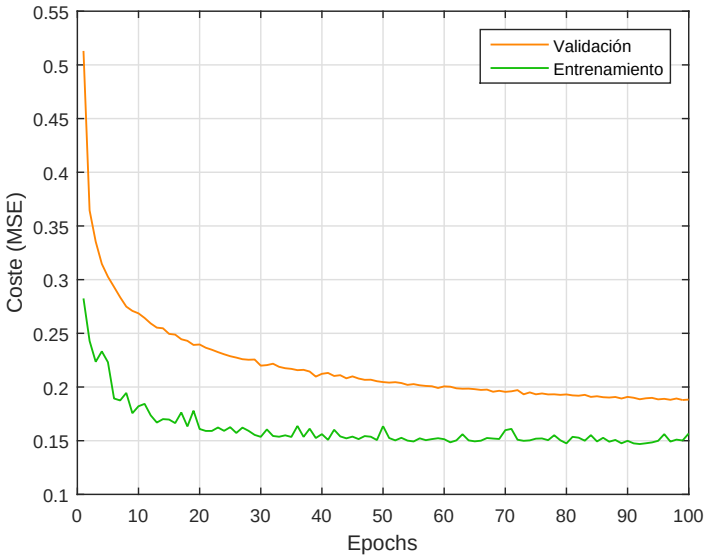

Figura 5.1: Entrenamiento vs validacion con el modelo *baseline*. El número de horas de datos de entrenamiento es 10h.

Para todos las pruebas se ha usado el algoritmo de optimización Adam, con función objetivo MSE y learning rate 0.0001. Este último parámetro está estrechamente relacionado con la convergencia del modelo y con el tiempo empleado en el proceso. En nuestro caso, se ha seleccionado este valor porque garantiza convergencia a un tiempo razonable. Por tanto, estos valores también se mantendrán en todos los entrenamientos.

Para el resto de experimentos se han seleccionado 10h de datos y 100 épocas de entrenamiento. No obstante, tal y como puede verse en la Tabla 5.2, para estos dos parámetros seleccionados algunos valores de PESQ y STOI de los audios limpios no superan los valores de PESQ y STOI de los audios sin mejorar, por lo que aún es posible encontrar algunas topologías de redes neuronales que superen y mejoren estos valores. Así pues, los resultados obtenidos con este modelo para el número de horas de entrenamiento y número de épocas seleccionados serán el *benchmark* de referencia que pretendemos superar con los nuevos sistemas propuestos.

5.1.2. Comparación de arquitecturas CNN con el sistema *baseline*

La topología de este modelo y las configuraciones diseñadas para este sistema fueron descritas en la sección 3.2.2. Dado que las capas convolucionales introducen nuevos parámetros en las redes neuronales, en esta sección realizaremos un estudio del comportamiento de éstos en la tarea de mejora de voz. Para ello, se han realizado dos experimentos que permitirán mostrar el efecto del número de filtros de cada capa convolucional y la profundidad de la red en el rendimiento del modelo final.

Efecto del número de filtros en las capas convolucionales

En primer lugar, comprobaremos el efecto que tiene el número de filtros de cada capa convolucional en los resultados finales, en términos de PESQ y STOI.

Para llevar a cabo este experimento se ha realizado una comparación entre dos configuraciones de redes CNN, en las que se ha mantenido invariante el número de bloques K, o lo que es lo mismo, el número de capas convolucionales seguidas de capas de *max-pooling*. En este caso se ha fijado $K = 2$. El número de filtros para la primera configuración ha sido $f_1 = 32$ y $f_2 = 64$ en la primera y segunda capa convolucional, respectivamente. En la segunda configuración los valores seleccionados han sido $f_1 = 50$ y $f_2 = 100$. Normalmente suelen escogerse el doble de filtros en la segunda capa que en la primera, para mantener una simetría en el número de características.

Tal y como se explicó en el capítulo 2, el número de filtros de cada capa convolucional define el número de mapas de características en la misma. Cada filtro en una capa convolucional encontrará los mismos patrones o características en la capa anterior. Así pues, a mayor número de filtros mayor será el número de características diferentes que esa capa puede encontrar en la capa inmediatamente anterior. En nuestro caso, a mayor número de filtros, mejor caracterizados estarán los audios ruidosos de entrada y será más fácil encontrar los patrones que diferencian a la voz limpia de la componente ruidosa prensente en la misma.

La Tabla 5.3 muestra los resultados obtenidos para las dos configuraciones de redes CNN mencionadas, así como para la red FC-DNN y para los audios ruidosos. Los valores de PESQ son superiores para todos los niveles de SNR que los obtenidos con los audios ruidosos, mientras que los valores de STOI aún se mantienen por debajo para los niveles mayores de SNR. Esto significa que cuando la componente ruidosa tiene menor efecto sobre la voz limpia, a la red neuronal le cuesta mas trabajo distinguir entre voz limpia y ruido. No obstante, en todos los casos, los valores superan a los de la red FC-DNN.

Por otro lado, podemos comprobar que el número de parámetros entrenables ha disminuido con respecto al sistema *baseline*, lo que significa que la complejidad del modelo también ha sido reducida. Aunque el número de parámetros de una red está estrechamente relacionado con su capacidad de aprendizaje, el elevado coste computacional que requiere en el entrenamiento una mayor cantidad de parámetros disminuye la eficiencia de ésta.

	Audios ruidosos		FC-DNN		CNN3x3 K=2 f1=32,f2=64		CNN3x3 K=2 f1=50,f2=100	
SNR (dB)	**PESQ**	**STOI**	**PESQ**	**STOI**	**PESQ**	**STOI**	**PESQ**	**STOI**
0	1,59	0,66	1,79	0,71	1,93	0,72	1,97	0,73
5	1,90	0,77	2,06	0,79	2,25	0,81	2,29	0,81
10	2,23	0,86	2,30	0,85	2,53	0,85	2,59	0,86
15	2,58	0,92	2,50	0,88	2,76	0,89	2,83	0,90
20	2,93	0,96	2,65	0,90	2,94	0,91	3,01	0,91
# parámetros entrenables			18,9M		12,9M		17,4M	

Tabla 5.3: Comparación del rendimiento del modelo FC-DNN con distintas configuraciones de arquitecturas CNN respecto a los valores de PESQ y STOI. Se han utilizado $K = 2$ bloques de capas convolucionales en las arquitecturas CNN y se han cambiado el número de filtros f_1 y f_2.

Efecto del número de capas convolucionales

En este experimento vamos a estudiar el efecto de la profundidad de la red en el rendimiento del sistema en términos de PESQ y STOI.

Se han propuesto dos arquitecturas de redes CNN para dicho estudio. La primera de ellas está compuesta por $K = 2$ capas convolucionales con $f_1 = 32$ y $f_2 = 64$ filtros respectivamente. La segunda de ellas está formada por $K = 3$ capas convolucionales de $f_1 = 16$, $f_2 = 32$ y $f_3 = 64$ filtros. Se ha mantenido el número de filtros en dos de las capas para que los resultados sean lo más comparables posible.

	Audios ruidosos		FC-DNN		CNN3x3 K=2 f1=32,f2=64		CNN3x3 K=3 f1=16,f2=32,f3=64	
SNR (dB)	**PESQ**	**STOI**	**PESQ**	**STOI**	**PESQ**	**STOI**	**PESQ**	**STOI**
0	1,59	0,66	1,79	0,71	1,93	0,72	1,98	0,74
5	1,90	0,77	2,06	0,79	2,25	0,81	2,30	0,82
10	2,23	0,86	2,30	0,85	2,53	0,85	2,59	0,87
15	2,58	0,92	2,50	0,88	2,76	0,89	2,83	0,90
20	2,93	0,96	2,65	0,90	2,94	0,91	3,02	0,92
# parámetros entrenables			18,9M		12,9M		37,5M	

Tabla 5.4: Comparación del rendimiento del modelo FC-DNN con dos configuraciones de arquitecturas CNN respecto a los valores de PESQ y STOI. Se han utilizado $K = 2$ y $K = 3$ bloques de capas convolucionales en las arquitecturas CNN, respectivamente, y se han mantenido el número de filtros en dos de las capas.

La Tabla 5.4 muestra los resultados obtenidos en este experimento. Podemos com-

probar que a mayor número de bloques K mejor es el rendimiento de los sistemas. No obstante, el número de parámetros entrenables se dispara, por lo que también aumenta la complejidad del modelo.

Con los dos experimentos llevados a cabo, hemos podido comprobar que al aumentar el número de filtros y el número de capas ocultas, mejoran los resultados obtenidos. En la sección 3.2.2 se afirmó que al aumentar el valor de estos parámetros podrían capturarse las diferencias entre las componentes de baja y alta frecuencia sin tener que usar diferentes conjuntos de pesos en un mismo mapa de características. Los resultados de PESQ y STOI obtenidos confirman esta suposición, aunque esto ya fue demostrado en [30].

Aunque los resultados mostrados son notablemente satisfactorios, la complejidad de los modelos no se ha visto disminuida. Tener un número de parámetros demasiado elevado puede dificultar la capacidad de generalización del sistema, generar un aumento del tiempo de entrenamiento y aumentar los requerimientos de memoria, requisitos clave para la selección de un modelo bueno y eficiente. Por este motivo, sería interesante encontrar aquellas configuraciones que, mejorando los resultados de PESQ y STOI del sistema *baseline*, no incumpliera estos dos requisitos.

5.1.3. Comparación de arquitecturas FCN con el sistema *baseline*

En esta sección se estudiará el comportamiento de las redes FCN para mejora de voz. La topología de este modelo y las configuraciones diseñadas para este sistema fueron descritas en la sección 3.2.3. Comprobaremos si este tipo de redes es capaz de satisfacer todos los requerimientos deseados, detallados anteriormente. Así pues, se han realizado varios experimentos que permitan mostrar el efecto del número de filtros en las capas convolucionales, de la función de activación en la capa de salida, del número de capas convolucionales en la arquitectura de la red y de la cantidad de datos de entrenamiento en el rendimiento de la misma.

Efecto del número de filtros en las capas convolucionales y de la función de activación de la capa de salida

El efecto del número de filtros ya ha sido descrito en la sección anterior para topologías de redes CNN. No obstante, en este experimento comproboramos si el efecto es el mismo en redes formadas únicamente por capas convolucionales. Además, dado que la última capa convolucional es la que genera el audio limpio, el cual no requiere un procesado posterior, estudiaremos el efecto de la función de activación en esta última capa.

	Audios ruidosos		FC-DNN		FCN55x1 (K=8)							
					F=30				F=90			
					linear		tanh		linear		tanh	
SNR (dB)	**PESQ**	**STOI**	**PESQ**	**STOI**	**PESQ**	**STOI**	**PESQ**	**STOI**	**PESQ**	**STOI**	**PESQ**	**STOI**
0	1,59	0,66	1,79	0,71	1,71	0,71	1,68	0,71	1,75	0,71	1,72	0,70
5	1,90	0,77	2,06	0,79	2,13	0,82	2,08	0,82	2,19	0,83	2,13	0,82
10	2,23	0,86	2,30	0,85	2,52	0,89	2,47	0,90	2,59	0,90	2,53	0,89
15	2,58	0,92	2,50	0,88	2,83	0,93	2,77	0,94	2,91	0,93	2,87	0,93
20	2,93	0,96	2,65	0,90	3,05	0,95	2,99	0,96	3,16	0,96	3,12	0,95
# parámetros entrenables			18,9M		300.931				2,7M			

Tabla 5.5: Comparación del rendimiento del modelo FC-DNN con distintas configuraciones de arquitecturas FCN respecto a los valores de PESQ y STOI. Se han utilizado $K = 8$ bloques de capas convolucionales y se ha variado el número de filtros F y la función de activación en la capa de salida.

La Tabla 5.5 muestra los resultados obtenidos en este test. Al aumentar el número de filtros en cada capa, aumentan los valores de PESQ y STOI, por lo que mejora la calidad de los audios mejorados. Además, en este tipo de redes, los resultados son mejores a niveles mayores de SNR, en comparación con los modelos anteriores.

Por otro lado, en [19] se propuso el uso de la función de activación *tanh* en la capa de salida, la cual genera valores comprendidos entre -1 y 1. No obstante, en nuestro caso, se ha propuesto el uso de la función *linear*, que puede devolver cualquier valor. Esto se adecua más a nuestro ejercicio, en el que la salida devuelta por la red es directamente el audio limpio en el dominio temporal. Además, en la Tabla 5.5 puede comprobarse que los resultados son ligeramente mejores cuando se usa esta última.

En este primer experimento con redes FCN, hemos podido comprobar que el número de parámetros entrenables ha disminuido considerablemente, y con ello la complejidad de las redes y los requisitos de memoria. Además, el rendimiento es bastante bueno en términos de PESQ y STOI, lo que hace que este tipo de redes sea más atractivo.

Efecto del número de capas convolucionales

El efecto del número de capas convolucionales, esto es, de la profundidad de la red, también ha sido estudiado en el sistema anterior. La Tabla 5.6 muestra que a mayor número de bloques K, mejores son los valores de PESQ y STOI, al igual que ocurría en la Tabla 5.4. Además, en este caso el número de parámetros entrenables no es tan alto como para las redes CNN, reduciéndose la complejidad de los modelos.

	Audios ruidosos		FC-DNN		FCN (F=30, linear)					
					(110x1)				(27x1)	
					K=4		K=5		K=16	
SNR (dB)	**PESQ**	**STOI**	**PESQ**	**STOI**	**PESQ**	**STOI**	**PESQ**	**STOI**	**PESQ**	**STOI**
0	1,59	0,66	1,79	0,71	1,71	0,71	1,72	0,72	1,68	0,70
5	1,90	0,77	2,06	0,79	2,11	0,81	2,12	0,82	2,07	0,82
10	2,23	0,86	2,30	0,85	2,50	0,88	2,51	0,89	2,45	0,89
15	2,58	0,92	2,50	0,88	2,84	0,92	2,83	0,93	2,77	0,93
20	2,93	0,96	2,65	0,90	3,10	0,94	3,05	0,95	3,00	0,96
# parámetros entrenables			18,9M		20.5051		304.201		344.071	

Tabla 5.6: Comparación del rendimiento del modelo FC-DNN con distintas configuraciones de arquitecturas FCN respecto a los valores de PESQ y STOI. Se han utilizado K=4, K=5 y K=16 bloques de capas convolucionales con F=30 filtros en cada una de ellas. El tamaño de los filtros se ha establecido a 110 o a 27.

En la Tabla anterior podemos comprobar que para $K = 16$ los valores de PESQ y STOI superan a los de los audios ruidosos para todos los niveles de SNR, con un número de parámetros entrenables relativamente pequeño (del orden de miles). Esto significa que los requerimientos de memoria necesarios para el entrenamieto de los modelos serán inferiores que para el resto de tipos de redes. Además, para este sistema no es necesario ningún tipo de preprocesado y postprocesado, pues los datos son introducidos en bruto a la red neuronal. Así pues, el tiempo de cómputo del sistema completo es muy bajo.

Efecto del número de horas de datos de entrenamiento

El objetivo de este experimento es comprobar si el número de horas de datos de entrenamiento se comporta igual que para el sistema *baseline* y verificar la hipótesis inicial en relación a este punto.

	Audios ruidosos		**FC-DNN**		**FCN55x1 (K=8, F=30, linear)**			
					10h		**100h**	
SNR (dB)	**PESQ**	**STOI**	**PESQ**	**STOI**	**PESQ**	**STOI**	**PESQ**	**STOI**
0	1,59	0,66	1,79	0,71	1,71	0,71	1,74	0,71
5	1,90	0,77	2,06	0,79	2,13	0,82	2,14	0,83
10	2,23	0,86	2,30	0,85	2,52	0,89	2,54	0,90
15	2,58	0,92	2,50	0,88	2,83	0,93	2,88	0,94
20	2,93	0,96	2,65	0,90	3,05	0,95	3,13	0,96
# parámetros entrenables			18,9M		300.931			

Tabla 5.7: Comparación del rendimiento del modelo FC-DNN con distintas configuraciones de arquitecturas FCN respecto a los valores de PESQ y STOI. Se han utilizado $K = 8$ bloques de capas convolucionales, $F = 30$ filtros en cada capa, función de activación *linear* en la capa de salida y se han variado el número de horas de datos de entrenamiento.

Para realizar este experimento se ha seleccionado una de las configuraciones anteriores ($K = 8$ bloques de capas convolucionales y $F = 30$ filtros en cada capa) y se ha entrenado con una mayor cantidad de datos de entrenamiento (100h). En la Tabla 5.7 puede comprobarse que a mayor cantidad de datos mejores son los resultados, por lo que la suposición inicial también se cumple para este tipo de redes.

Representación de los filtros aprendidos por la red

Como se ha mencionado en líneas anteriores, el número de mapas de características viene definido por el número de filtros en cada capa convolucional. Así pues, cada filtro de una capa intentará encontrar patrones en la capa anterior y resaltar algunas características determinadas. Así pues, los pesos de los filtros son los parámetros entrenables en las capas convolucionales de una red neuronal.

En concreto, los filtros de la primera capa de una red FCN extraen información directamente de los audios de entrada. Por este motivo, resulta interesante representar dichos filtros y analizar qué características de los audios de entrada está aprendiendo la red neuronal. Además, esto tiene especial interés en este sistema, en el que no se habían extraído características de los audios de entrada, sino que éstos se habían introducido en bruto a la red, en el dominio temporal. Los distintos filtros intentarán representar las componentes de alta y baja frecuencia de los audios ruidosos y poder distinguir entre la voz limpia y el ruido.

Para realizar este análisis, se ha seleccionado la red FCN con $K = 16$ capas convolucionales, *kernel size* 27×1 y $F = 30$ filtro en cada capa, por ser la que mejores resultados ofrece (veáse Tabla 5.8).

A continuación, se representan 12 de los $F = 30$ filtros que componen la primera capa convolucional. Éstos han sido elegidos al azar.

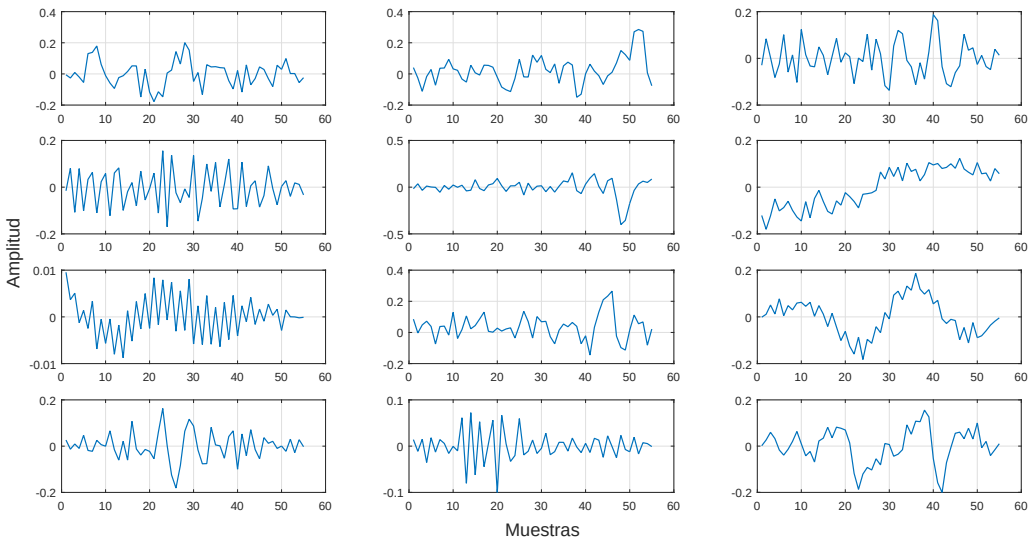

Figura 5.2: Representación de 12 filtros de la primera capa convolucional de la red de tipo FCN con $K = 16$ y *kernel size* 27×1.

En la Figura 5.2 podemos apreciar filtros con características muy diversas (periodicidad en el tiempo, oscilaciones a altas frecuencias, etc). Esta heterogeneidad permite que cada uno de ellos se centre en determinadas peculiaridades y propiedades de los audios de entrada, capturando diferentes patrones y comportamientos en los mismos. Por este motivo, estos filtros son capaces de representar de forma eficiente las componentes de alta y baja frecuencia de la señal de voz y distinguir entre la voz limpia y el ruido.

5.2. Análisis y discusión de resultados. Selección del mejor modelo.

En esta sección se va a realizar una comparación entre las mejores configuraciones anteriores de los distintos tipos de redes. Esto permitirá evaluar el comportamiento de las diferentes arquitecturas en la tarea de mejora de voz e identificar la que genere mejor rendimiento.

La Tabla 5.8 muestra una comparativa entre la red FC-DNN del sistema de referencia, una de las configuraciones de redes CNN y dos configuraciones de redes FCN. La red CNN tiene $K = 2$ capas convolucionales con $f_1 = 50$ y $f_2 = 100$ filtros en cada una de ellas, respectivamente. Además el tamaño de los filtros es de 3×3. Con respecto a las redes FCN, están formadas por $K = 8$ y $K = 16$ capas convolucionales con $F = 30$ filtros cada una de ellas. El *kernel size* es de 55×1 para la primera red y de 27×1 para la segunda.

	Audios ruidosos		**FC-DNN**		**CNN**		**FCN**			
					CNN3x3		**FCN55x1**		**FCN27X1**	
					f1=50,f2=100		**F=30**			
					K=2		**K=8**		**K=16**	
SNR (dB)	**PESQ**	**STOI**	**PESQ**	**STOI**	**PESQ**	**STOI**	**PESQ**	**STOI**	**PESQ**	**STOI**
0	1,59	0,66	1,79	0,71	1,97	0,73	1,71	0,71	1,68	0,70
5	1,90	0,77	2,06	0,79	2,29	0,81	2,13	0,82	2,07	0,82
10	2,23	0,86	2,30	0,85	2,59	0,86	2,52	0,89	2,45	0,89
15	2,58	0,92	2,50	0,88	2,83	0,90	2,83	0,93	2,77	0,93
20	2,93	0,96	2,65	0,90	3,01	0,91	3,05	0,95	3,00	0,96
# parámetros entrenables			18,9M		17,4M		300.931		344.071	

Tabla 5.8: Comparación del rendimiento del modelo FC-DNN con distintas configuraciones de arquitecturas CNN y FCN respecto a los valores de PESQ y STOI.

Basándonos en los requisitos impuestos para la selección de la mejor arquitectura, el modelo con mejor comportamiento es aquel con arquitectura FCN de $K = 16$ capas convolucionales de $F = 30$ filtros y con *kernel size* de 27×1. Los valores de PESQ y STOI son superiores a los de los audios ruidosos para todos los niveles de SNR. Además, los resultados experimentales muestran que este modelo de red no solo recupera eficazmente las formas de onda, sino también supera el *benchmark* de la red FC-DNN en términos de inteligibilidad objetiva a corto plazo (STOI) y evaluación perceptiva de la calidad del habla (PESQ). Asimismo, este sistema no requiere de ningún preprocesado ni postprocesado, por lo que el tiempo de cómputo y los requerimientos de memoria serán bajos. El tiempo de entrenamiento en redes FCN es mucho más bajo que para las arquitecturas CNN y FC-DNN. Esto se debe al reducido número de parámetros entrenables que tienen las redes FCN y a las capas de *Batch Normalization* incluidas en esta arquitectura. Este tipo de capas acelera la convergencia del modelo, mejora el rendimiento y afianza la estabilidad de las redes neuronales artificiales.

Capítulo 6

Conclusiones y trabajo futuro

En este capítulo se realizará una valoración global del trabajo realizado y de los objetivos alcanzados. Se analizará el grado de cumplimiento de los objetivos fijados y se describirán las dificultades encontradas y los conocimientos adquiridos durante el desarrollo del Trabajo. Además, se expondrán unas posibles vías de trabajo futuro que puedan dar continuidad a la investigación en este campo.

6.1. Conclusiones

El objetivo principal de este Trabajo Fin de Máster ha sido diseñar e implementar diferentes sistemas de mejora de voz basados en modelos de redes neuronales que mejoren los resultados obtenidos en trabajos anteriores.

Inicialmente, se ha dedicado un periodo de tiempo al estudio y análisis de las técnicas existentes, para poder encontrar las limitaciones de las mismas y la forma de solventarlas. Además, la voz es una señal compleja, por lo que conocer sus propiedades y características ayudará a encontrar la forma más eficiente de tratarla y trabajar con ella. En este análisis nos hemos apoyado en libros destinados al procesado de la voz y en artículos científicos, que han servido de sustento para afrontar el problema de la mejora de voz.

El punto de partida en el desarrollo de nuevos sistemas ha sido el trabajo realizado en [13] [14], donde se propuso una red FC-DNN para reducción de ruido. Para poder diseñar modelos novedosos ha sido necesario un análisis profundo de dicho trabajo, que permita evaluar las aportaciones y desventajas del mismo y reflexionar sobre las mejoras que deben incluirse en un nuevo sistema. Los resultados obtenidos con este sistema han sido la línea base que nos hemos propuesto superar con el diseño de nuevos sistemas.

No obstante, durante este estudio inicial surgieron dos inconvenientes. En primer lugar, los modelos de redes neuronales requieren una gran cantidad de datos de entrenamiento. Además, en nuestro caso, es necesario disponer de una gran variedad de condiciones acústicas que representen un entorno ruidoso real y den lugar a una base de datos lo suficientemente robusta y representativa. Esto hace que los requerimientos de memoria aumentan considerablemente. Para afrontarlo, se ha tenido que implementar un software

que sea capaz de trabajar con grandes volúmenes de datos y con las restricciones en procesamiento del hardware disponible en el laboratorio.

En segundo lugar, el estudio previo del sistema *baseline* y las limitaciones encontradas en el mismo, ha dado impulso al estudio de nuevas arquitecturas de redes neuronales. Las redes propuestas han sido aquellas que usan capas convolucionales en su arquitectura, las cuales no habían sido estudiadas en este campo de aplicación en nuestro Grupo. Así pues, esto ha requerido un estudio teórico de este tipo de redes, que ha sido crucial en el desarrollo del Trabajo. Además, el tiempo de cómputo empleado en el preprocesado y postprocesado del sistema anterior era demasiado elevado, por lo que se ha propuesto prescindir del mismo, impulsando el desarrollo de un sistema novedoso *end to end* que no necesite extracción de características de los datos de entrada.

Así pues, se han propuesto dos sistemas distintos que utilizan diferentes topologías de redes neuronales convolucionales. Además, dado que no existe una configuración predefinida de red neuronal que se adecue a un problema concreto, se han realizado diferentes experimentos que permitan encontrar aquellos parámetros que mejor se adapten a nuestra tarea. Por este motivo, se han comparado distintas configuraciones modificando algunos de los hiperparámetros más característicos de las redes neuronales convolucionales.

Los resultados se han comparado en términos de PESQ y STOI, prestando especial interés al número de parámetros entrenables, complejidad de la red y tiempo de entrenamiento. Se ha buscado un equilibrio entre los requisitos mencionados para seleccionar las mejores configuraciones.

Los resultados empíricos han sido satisfactorios en los dos sistemas propuestos, pues los valores de PESQ y STOI han superado a los de línea base. Además, el tiempo de cómputo de los sistemas completos y el número de parámetros también ha sido más bajo. Por tanto, se ha logrado cumplir los objetivos fijados al inicio del proyecto.

Por último, el desarrollo de este Trabajo Fin de Máster ha supuesto una implicación en tareas de investigación, lo que ha llevado a una ampliación de los conocimientos técnicos adquiridos durante el máster. Se ha trabajado con técnicas y algoritmos realmente novedosos, que están experimentando un gran auge en el mundo empresarial actual.

6.2. Trabajo futuro

La tarea de mejora de voz lleva siendo objeto de estudio durante décadas, por la gran cantidad de aplicaciones en las que está implicada el habla. Además, el creciente auge de las tecnologías y del análisis de datos impulsa a una continua investigación en este campo. Como posibles vías de trabajo futuro se proponen las siguientes:

- Ampliación de la base de datos, usando más ruidos ambientales y oraciones de otros hablantes. Sería interesante mezclar contenido en varios idiomas para aumentar la capacidad de generalización de los modelos.

- Aplicación de redes neuronales híbridas. Se propone el uso de varias redes neuronales en paralelo que puedan generar resultados más precisos y fiables. Además, la introducción de características adicionales que representen al tipo de hablante o al ruido (*metadata*) también podría resultar atractivo para el modelo, que lo utilizaría como información complementaria.

Estas líneas de trabajo futuro podrían dar continuidad a la investigación de sistemas de aprendizaje profundo en mejora de voz, pues promueven el estudio de nuevos enfoques.

Referencias bibliográficas

[1] SCALART, Pascal, et al. Speech enhancement based on a priori signal to noise estimation. En *Acoustics, Speech, and Signal Processing, 1996. ICASSP-96. Conference Proceedings., 1996 IEEE International Conference on.* IEEE, 1996. p. 629-632.

[2] BISHOP, Christopher M. Pattern Recognition and Machine Learning. Springer, 2006.

[3] TOMASSI, D. R.; ARONSON, L.; MARTÍNEZ, C. E.; MILONE, D: H.; TORRES, M. E.; RUFINER, M. E. Evaluación de técnicas clásicas de reducción de ruido en señales de voz. En *Revista Argentina de Bioingeniería*, Paraná, Entre Ríos, Argentina, 2005.

[4] BOLL, S. F. Suppression of acoustic noise in speech using spectral subtraction. En *IEEE Trans. Acoustics, Speech, and Signal Processing*, 1979, vol. 27(2), p. 521-534.

[5] LIM, J. S.; OPPENHEIM, A. V. All-pole modeling of degraded speech. En *IEEE Trans. Acoustics, Speech, and Signal Processing*, 1978, vol. 26(3), p. 197-210.

[6] EPHRAIM, Y.; MALAH, D. Speech enhancement using a minimum mean-square error short-time spectral amplitud estimator. En *IEEE Trans. Acoustics, Speech, and Signal Processing*, 1984, vol. 32(6), p. 1109-1121.

[7] EPHRAIM, Y.; MALAH, D. Speech enhancement using a minimum mean-square error log-spectral amplitud estimator. En *IEEE Trans. Acoustics, Speech, and Signal Processing*, 1985, vol. 33(2), p. 443-446.

[8] GÁMIZ, Laura. Uso de wavelets para reducción de ruido: aplicación en señales de voz y señales sísmicas. *Universidad de Granada*, 2017.

[9] EPHRAIM, Y.; VAN TREES, H. L. A signal subspace approach for speech enhancement. En *IEEE Trans. Speech and Audio Processing*, 1995, vol. 3(4), p. 251-266.

[10] HU, Y.; LOIZOU, P. C. A generalized subspace approach for enhancing speech corrupted by colored noise. En *IEEE Trans. Speech and Audio Processing*, 2003, vol. 11(4), p. 334-341.

[11] TCHORZ, J.; KOLLMEIER, B. SNR estimation based on amplitude modulation analysis with applications to noise suppression. En *IEEE Trans. Speech and Audio Processing*, 2003, vol. 11(3), p. 184-192.

[12] WANG, Y. X.; WANG, D. L. Towards scaling up classification-based speech separation. En *IEEE Trans. Audio, Speech and Language Processing*, 2013, vol. 21(7), p. 1381-1390.

[13] LU, X.; DU, J.; DAI, L. R.; LEE, C. H. A regression approach to speech enhancement based on deep neural networks. En *IEEE Trans. Audio, Speech and Language Processing*, 2015, vol. 23(1), p. 7-19.

[14] LU, X.; DU, J.; DAI, L. R.; LEE, C. H. An experimental study on speech enhancement based on deep neural networks. En *IEEE Signal Processing Letter*, 2014, vol. 21(1), p. 65-68.

[15] ZHAO, Y.; WANG, D.; MERKS, I.; ZHANG, T. DNN-based enhancement of noisy and reverberant speech. En *IEEE International Conference on Acoustics, Speech and Signal Processing*, 2016, p. 6525-6529.

[16] WANG, Y.; NARAYANAN, A.; WANG, D. On training targets for supervised speech separation. En *IEEE/ACM transactions on audio, speech and language processing*, 2014, vol. 22(12), p. 1849-1858.

[17] GAO, T.; DU, J.; XU, Y.; LIU, C.; DAI, L. R.; LEE, C. H. Improving deep neural network based speech enhancement in low snr environments. En *International Conference on Latent Variable Analysis and Signal Separation*, Springer 2015, p. 75-82.

[18] FU, S. W.; TSAO, Y.; LU, X. SNR-Aware convolutional neural network modeling for speech enhancement. En *INTERSPEECH*, 2016, p. 3768-3772.

[19] FU, S. W.; TSAO, Y.; LU, X.; KAWAI, H. Raw waveform-based speech enhancement by fully convolutional networks. En *Proc Asia, Pac. Signal Inf. Process. Assoc. Annu. Summit Conf*, 2017, p. 6-12.

[20] FU, S. W.; TSAO, Y.; LU, X. End-to-End Waveform Utterance Enhancement for Direct Evaluation Metrics Optimization by Fully Convolutional Neural Networks. En *IEEE/ACM transactions on audio, speech and language processing*, 2018, vol. 26(9) p. 1570-1584.

[21] R. Hecht-Nielsen Neurocomputing: Picking the human brain. En *IEEE Spectrum*, Mar. 1988, p. 36-41.

[22] DUCHI, J.; SINGER, Y.; HAZAN, E. Adaptive Subgradient Methods for Online Learning and Stochastic Optimization. En *Journal of Machine Learning Research*, 2011, p. 2121-2159.

[23] KINGMA D. P.; JIMMY L. B. Adam: A method for stochastic optimization. En *arXiv preprint arXiv:1412.6980*, 2014.

[24] LECUN, Y.; BOTTOU, L.; BENGIO, Y.; HAFFNER, P. Gradient-based learning applied to document recognition. En *Proceedings of the IEEE*, 1998, p. 2278–2324.

[25] WAN, E.; NELSON, A. Networks for speech enhancement. En *Handbook of Neural Networks for Speech Processing*, 1998, Norwell, MA, USA.

[26] XIE, F.; COMPERNOLLE, D. V. A family of MLP based nonlinear spectral estimators for noise reduction. En *Proc. ICASSP*, 1994, p. 53-56.

[27] LeCun, Y.; Bengio, Y. Convolutional Networks for Images, Speech, and Time-series. En *The Handbook of Brain Theory and Neural Networks*, 1995.

[28] LeCun, Y.; Huang, F.; Bottou, L. Learning Methods for Generic Object Recognition with Invariance to Pose and Lighting. En *Proc. CVPR*, 2004.

[29] Abdel-Hamid, O.; Mohamed, A.; Jiang, H.; Penn, G. Applying Convolutional Neural Network Concepts to Hybrid NNHMM Model for Speech Recognition. En *Proc. ICASSP*, 2012.

[30] Abdel-Hamid, O.; Mohamed, A.; Brian K.; Bhuvana R. DEEP CONVOLUTIONAL NEURAL NETWORKS FOR LVCSR. Department of Computer Science, University of Toronto, Canada.

[31] S.W. Fu; Y. Tsao, X. Lu; H. Kawai Raw waveform-based speech enhancement by fully convolutional networks. En *Proc. Asia, Pac. Signal Inf. Process. Assoc. Annu. Summit Conf.*, 2017, p. 6-12.

[32] A. V. Oppenheim Discrete-time signal processing. Pearson Education India, 1999.

[33] K. Paliwal; K. Wójcicki; B. Shannon The importance of phase in speech enhancement. En *speech communication*, 2011, vol. 53, p. 465-494.

[34] Hu G.; Wang D.L A tandem algorithm for pitch estimation and voiced speech segregation. En *IEEE Transactions on Audio, Speech, and Language Processing*, Sep. 2011, vol. 19, no. 7, p. 2125–2136.

[35] Moreno, A; Poig, D.; Bonafonte, A.; Lleida, E.; Llisterri, J.; Mariño, J.B.; Nadeu, C. Albayzin speech database: design of the phonetic corpus. En *uropean Conference on Speech Communication and Technology*, 1993, p. 175-178.

[36] C. H. Taal; R. C. Hendriks; R. Heusdens; J. Jensen An algorithm for intelligibility prediction of time–frequency weighted noisy speech. En *IEEE Trans. Audio, Speech, Lang. Process.*, Sep. 2011, vol. 19, no. 7, p. 2125–2136.

[37] A. Rix; J. Beerends; M. Hollier; A. Hekstra Perceptual evaluation of speech quality (PESQ), an objective method for end-to-end speech quality assessment of narrowband telephone networks and speech codecs. En *Int. Telecommun. Union, T Recommendation*, 2001, Art. no. 862.