UNIVERSITAT POLITÈCNICA DE VALÈNCIA

_ TELECOM ESCUELA TÉCNICA VLC SUPERIOR DE INGENIERÍA DE TELECOMUNICACIÓN

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Escuela Técnica Superior de Ingeniería de Telecomunicación

Análisis de audio mediante técnicas de aprendizaje profundo

Trabajo Fin de Grado

Grado en Ingeniería de Tecnologías y Servicios de Telecomunicación

AUTOR/A: Pastor Naranjo, Francisco

Tutor/a: Piñero Sipán, María Gemma

Cotutor/a: Amor del Amor, María Rocío del

CURSO ACADÉMICO: 2021/2022

Resumen

El análisis de la señal de audio puede definirse como el proceso de extracción de información relevante a partir de las muestras de la señal. En la mayoría de los casos, las señales de voz y música son captadas por equipos electrónicos para su posterior análisis, pero también son alteradas por ecos, ruidos, etc., que dificultan dicha tarea. En este sentido, es fundamental el desarrollo de algoritmos que preprocesen la señal para reducir al máximo las fuentes externas que la alteran. Este TFG se centra en soluciones basadas en Deep Learning (DL) para dos aplicaciones de audio: la cancelación del eco acústico en sistemas de teleconferencia y la detección de emociones en la voz de una persona. Para la cancelación del eco, se utilizarán algoritmos basados en Redes Generativas Adversariales Condicionales (Conditional Generative Adversarial Networks, cGAN), que han demostrado tener un mejor rendimiento en el campo de la mejora del habla que otros modelos de DL. En cuanto al problema de la detección de emociones, se explorarán técnicas híbridas basadas en redes neuronales convolucionales (Convolutional Neural Networks, CNN) y redes de memoria larga a corto plazo (Long Short-Term Memory , LSTM), así como nuevos modelos DL como los Transformers.

Resum

L'anàlisi de la senyal d'àudio pot definir-se com el procés d'extracció d'informació rellevant a partir de les mostres de la senyal. En la majoria dels casos, les senyals de veu i música són captades per equips electrònics per al seu posterior anàlisi, però també són alterades per ecos, sorolls, etc., que dificulten aquesta taska. En eixe sentit, és fonamental el desenvolupament d'algoritmes que preprocessen la senyal per a reduir al màxim les fonts externes que l'alteren. Aquest TFG es centra en solucions basades en Deep Learning (DL) per a dues aplicacions d'àudio: la cancel·lació de l'eco acústic en sistemes de teleconferència i la detecció d'emocions en la veu d'una persona. Per a la cancel·lació d'eco, s'utilitzaran algoritmes basats en xarxes generatives adversarials condicionals (Conditional Generative Adversarial Networks, cGAN), que han demostrat tenir un millor rendiment en el camp de la millora de la parla que altres models de DL. En quant al problema de la detecció d'emocions, s'exploraran tècniques híbrides basades en xarxes neuronals convolucionals (Convolutional Neural Networks, CNN) i xarxes de memòria llarga a curt termini (Long Short-Term Memory , LSTM), així com nous models DL com els Transformers.

Abstract

Audio signal analysis can be defined as the process of extracting relevant information from signal samples. In most cases, speech and music signals are captured by electronic equipment for further analysis, but they are also altered by echoes, noise, etc., which hinder such task. In this sense, it is essential to develop algorithms that preprocess the signal to reduce as much as possible the external sources that alter it. This TFG focuses on Deep Learning (DL) based solutions for two audio applications: acoustic echo cancellation in teleconferencing systems and emotion detection in a person's speech. For echo cancellation, algorithms based on conditional Generative Adversarial Networks (cGAN), which have been shown to perform better in the field of speech enhancement than other DL models, will be used. For the emotion detection problem, hybrid techniques based on Convolutional Neural Networks (CNN) and Long Short-Term Memory (LSTM) networks will be explored, as well as new DL models such as Transformers.

Quiero agradecer a mis tutoras, Gema y Rocío, que me han ayudado y han estado apoyándome todo este tiempo para realizar el trabajo. Sin vosotras habría sido imposible.

A mis amigos, por estar ahí siempre aunque la vida nos lleve por caminos distintos.

A Ana, mi ratilla, porque estar con un cabezón no es fácil y por quererme cada día como lo haces, te adoro.

A mi hermano, por aguantarme y por ser la persona con quien más cosas comparto, te quiero mucho.

A mi padre por estar siempre ahí y quererme como lo haces. Gracias por cuidarme así.

Y por último, a mi madre, que es la más top del poli y además es mi madre. Gracias por guiarme. Eres un ejemplo para mí.

Índice general

Índice de figuras

Índice de tablas

Listado de siglas empleadas

AA Aprendizaje Automático.

AEC Acoustic Echo Cancellation.

AWGN Additive White Gaussian Noise.

cGAN conditional Generative Adversarial Network.

CNN Convolutional Neural Network.

CPU Control Processing Unit.

CVBLab Computer Vision and Behaviour Analysis Laboratory.

D3Net Densely-Connected Multidilated DenseNet.

DCNN 1-Dimensional Convolutional Neural Network.

Deep-FSMN Deep Feedforward Sequential Memory Networks.

DFT Discrete Fourier Transform.

DL Deep Learning.

EEG Electroencefalograma.

EmoDB Berlin Database of Emotional Speech.

ERLE Echo Return Loss Enhacement.

FFT Fast Fourier Transform.

FN Far-end Noisy.

FnN Far-end not Noisy.

FT Fourier Transform.

GAN Generative Adversarial Network.

GCCRN Gate Complex Convolutional Recurrent Network.

GPU Graphics Processing Unit.

GRU Gated Recurrent Unit.

LFLB Local Feature Learning Block.

LSTM Long Short-Term Memory.

MFCC Mel Frequency Cepstral Coeficient.

MLP Multi Layer Perceptron.

N+FnL Noisy + Far-end not Lineal.

NAS Network-Attached Storage.

NN Near-end Noisy.

NnN Near-end not Noisy.

PCNSE Parallel Convolutional layers integrated with Squeeze-and-Excitation Network.

PESQ Perceptual Evaluation of Speech Quality.

RAM Random Acess Memory.

RAVDESS Ryerson Audio-Visual Database of Emotional Speech and Song.

RBSC Residual Block Skip Connections.

ReLU Rectified Linear Unit.

RES Residual Echo Supressors.

RIR Room Impulse Response.

RMSE Root Mean Square Error/Energy.

RNN Recurrent Neural Network.

SADRN Self-Attention Dilated Residual Network.

Seq-L Sequence-Learning.

SER Signal to Echo Ratio.

STFT Short Time Fourier Transform.

SVM Support Vector Machine.

TFG Trabajo Fin de Grado.

TQWT Tunnable Q Wavelet Transform.

ViT Vision Transformer.

ZCR Zero Crossing Rate.

Parte I

Memoria

Capítulo 1

Introducción

1.1. Motivación y descripción del problema

El procesado de señal consiste en el mapeo una señal de entrada a una salida a través de algoritmos que la transforman. Existen muchas maneras de tratar estas señales en función del objetivo que se persiga. Aquellos que más interés suscitan y que presentan mayor actividad en el estado del arte son reconocimiento de voz, mejora de la voz, detección de emociones en voz, cancelación de eco o supresión de eco, entre otros. Este proyecto se enfoca en la implementación de técnicas de inteligencia artificial basadas en aprendizaje profundo para dos de estas aplicaciones: cancelación de eco acústico y detección de emociones en voz.

Cancelación de eco acústico

Los sistemas de cancelación de eco acústico fueron desarrollados en primera instancia por los AT&T Bell Labs [1] y estaban basados en el modelo mostrado en figura1.1, donde el bloque AEC (Acoustic Echo Cancellation) fue implementado mediante un filtro adaptativo. El objetivo de los sistemas AEC es eliminar el eco indeseado producido por el acoplamiento acústico entre el altavoz y el micrófono, normalmente contenidos en el mismo dispositivo. Tradicionalmente, la cancelación de eco se consigue identificando la respuesta al impulso de la sala (RIR) entre el altavoz y el micrófono, denotado como $h(n)$ en la figura1.1, y substrayendo la señal estimada del eco $x(n) * h(n)$ de la señal del micrófono $y(n)$ con el fin de obtener la voz limpia $\hat{s}(n)$. En la práctica, un cancelador debe lidiar también con escenarios no estacionarios, efectos no lineales sobre las señales y, probablemente, con ruido ambiente denotado por $v(n)$ en la figura1.1. Para combatir estos inconvenientes, habitualmente se incluyen en el diseño de sistemas AEC supresores residuales de eco (RES) [2] y detectores de actividad para las señales del extremo lejano (far-end) y cercano (near-end) [3].

Recientemente, sistemas AEC basados en técnicas de inteligencia artificial han superado los métodos clásicos basados en filtros adaptativos, especialmente cuando tienen que lidiar con escenarios no estacionarios. En 2021 y 2022 Microsoft propuso tres *challenges* sobre cancelación de eco acústico [4] donde sistemas AEC competían entre sí y eran evaluados empleando una base de datos desconocida por los participantes. La forma de evaluación se realizó mediante una puntuación perceptual [5]. Esto, como se verá posteriormente en la sección 1.2, resulta de gran utilidad para la

comparación de modelos y el desarrollo de los mismos.

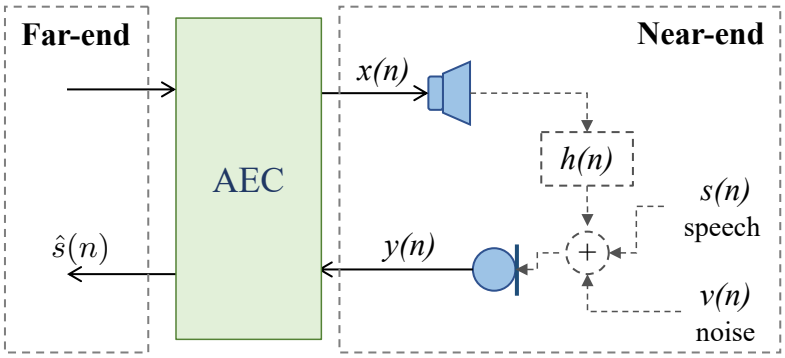


Figura 1.1: Modelo de un sistema de cancelación de eco acústico

Detección de emociones en voz

En cuanto a la detección de sentimientos, este es también uno de los campos más en boga en el procesado de señal y es lo que se conoce como *Cognitive Computing*. Conforme los algoritmos de aprendizaje profundo y la tecnología han ido evolucionando, los servicios que incluyen algún tipo de inteligencia artificial han aumentado considerablemente. En consecuencia, las técnicas de detección de emociones en voz se han vuelto mucho más habituales al estar presentes en aplicaciones del día a día como pueden ser las encuestas de satisfacción de cliente [6], servicios sanitarios por teléfono [7], centros de llamadas [8], asistencia en carretera [9] o interacciones humano-máquina [10]. La detección de emociones se está empezando a emplear también en aplicaciones de asistencia psicológica, pudiendo detectar enfermedades como depresión, adicciones o incluso identificación de trastorno del espectro autista en personas no diagnosticadas [11]. Es fundamental disponer de modelos y sistemas capaces de reconocer de forma precisa la información intrínseca del discurso para poder dar una respuesta lo más apropiada posible en cada momento. Y es que, cuando nos comunicamos, gran parte de la información que aportamos no procede del texto que estamos reproduciendo, sino de la forma de expresarnos, de la intención, del énfasis que realizamos sobre ciertas palabras, en definitiva, de los sentimientos que mostramos al hablar.

Este trabajo se enfoca principalmente en el procesado de voz pero existen diversas maneras de enfocar el problema de detección de emociones en función del tipo de datos que se utilicen para la clasificación [12]:

- Voz: influyen el tono, ritmo de palabras, duración de los silencios, volumen, entre otros.
- Texto: se centra en el significado de la frase en sí.
- Señales fisiológicas: se emplean mediciones como electroencefalogramas (EEG) o sensores corporales que monitorizan las constantes del sujeto.
- Vídeo: se observan las facciones y gestos durante el discurso e incluso la expresión corporal.
- Multimodal: se combinan varias de las características anteriores para mejorar la capacidad del modelo.

El presente proyecto, por tanto, está dividido en dos bloques. Por un lado, se ha desarrollado una red neuronal generativa capaz de tomar audios de entrada, eliminar el eco presente en la grabación y mantener la información relevante. Por otro, se ha construido un clasificador, también basado en aprendizaje profundo, capaz de discernir entre distintas emociones según el discurso de un orador. Para la cancelación de eco y la detección de sentimientos, el preprocesado aplicado sobre las señales es similar; se realiza un cambio de dominio de tiempo a tiempo-frecuencia utilizando un tipo de espectrograma determinado en función de las necesidades de la aplicación. En el caso de la cancelación de eco, se han obtenido los espectrogramas para tamaños de ventana pequeños, mientras que en la detección de emociones, se han obtenido espectrogramas de Mel para cada muestra completa.

1.2. Estado del arte

1.2.1. Cancelación de eco acústico

En cuanto a los trabajos recientes sobre cancelación de eco, y observando los resultados obtenidos en los *Challenges* de Microsoft, destacan dos vías de entre las mejores soluciones presentadas. Por un lado, hay trabajos que emplean redes neuronales en combinación con canceladores adaptativos para eliminar el eco residual [13, 14, 15] y, por otro, hay soluciones para la cancelación de eco acústico que sólo emplean modelos de aprendizaje profundo [16, 17]. Por tanto, se pueden encontrar modelos tales como *Gated Recurrent Units* (GRU) [13], *Deep Feedforward Sequential Memory Networks* (Deep-FSMN) [14], *Gate Complex Convolutional Recurrent Networks* (GCCRN) [15] o redes *Long Short Term Memory* (LSTM) [16, 17].

A parte de las soluciones presentadas a los AEC Challenges, otros canceladores basados en redes neuronales han sido presentados recientemente [18, 19, 20, 21]. En [18], los autores proponen una *Multiscale Attention Neural Network* para cancelación de eco como implementación extremo a extremo, el cual combina convoluciones temporales para transformar la forma de onda en espectrogramas y módulos de atención para obtener características empleando unidades LSTM. En [19], construyen una U-Net con múltiples encoders para suprimir el eco residual. Además, en [20], implementan una red muy pequeña basada en una *densely-connected multidilated DenseNet* (D3Net) para trabajar en tiempo real, eliminando la necesidad de realizar pooling gracias a su construcción de bloques. Más recientemente, se ha estudiado el uso de *Generative Adversarial Networks* (GANs) para cancelación de eco [21]. Sin embargo, en este caso, el generador, basado en un *autoencoder* simple, genera máscaras tiempo-frecuencia que, en un segundo paso, sirven para resintetizar la señal mejorada. Por tanto, la GAN desarrollada en [21] no es capaz de mapear una señal ruidosa a una señal limpia de forma integral, en un solo paso.

Sin embargo, no hay una solución definitiva para el problema de cancelación de eco acústico, por lo que este campo de investigación permanece abierto. La contribución realizada en este proyecto es un sistema AEC extremo a extremo basado en una *Generative Adversarial Network* condicional (cGAN). La cGAN es una versión mejorada de la GAN que ha superado los resultados del estado del arte para distintas tareas como mejora de voz [22], generación y reconocimiento de fuente [23] y generación de imagen [24], entre otros. Sin embargo, esta es la primera vez que una cGAN se emplea para cancelación de eco acústico. Este tipo de modelo se compone de dos bloques: el generador, encargado de procesar la señal grabada por el micrófono $y(n)$ y sintetizarla en una nueva señal sin eco, $\hat{s}(n)$, y el discriminador, que asegura y mejora la calidad de la señal sintetizada $\hat{s}(n)$.

Este último bloque determina si la señal de entrada es real o sintetizada, evitando que el generador cree salidas no realistas, como será detallado más adelante.

1.2.2. Detección de emociones en voz

En lo referido a la computación afectiva, debido a que este proyecto está basado en el tratamiento de voz para detectar emociones, los trabajos recogidos en el estado del arte emplearán únicamente esta fuente de información para identificar sentimientos. Se puede hacer una distinción en dos grupos en función de si la extracción de características del audio se realiza de manera automática, métodos de aprendizaje profundo, o si previo al clasificador se requiere una etapa enfocada expresamente a realizar esta función (aprendizaje *hand-crafted*).

Estudios basados en extracción automática de características

En [25], los autores proponen un método para solventar el problema de la generalización de los modelos cuando estos se aplican a otras bases de datos distintas a la de entrenamiento sin tener que emplear cantidades enormes de muestras etiquetadas. Realizan lo que se llama aprendizaje multi tarea (*multi-task learning* en inglés), donde enseñan a la red a realizar tareas secundarias a parte de la principal para mejorar el rendimiento. Además, lo hacen de forma no supervisada, evitando así la necesidad de disponer de muchas muestras etiquetadas. Mientras que la tarea principal es la predicción de atributos emocionales mediante regresión, la auxiliar es la reconstrucción de características intermedias empleando un *autoencoder* para reducción de ruido, con lo que mejoran el rendimiento mostrado en el estado del arte. En [26], los autores proponen un modelo extremo a extremo basado en una red neuronal convolucional unidimensional (DCNN) para realizar detección de emociones en tiempo real. En este caso, emplean la estrategia de aprendizaje híbrido, extrayendo de forma paralela las características espaciales de la señal y las dependencias temporales de la voz a largo plazo. Para realizar esas tareas se emplean bloques residuales con conexiones de salto (RBSC) y un módulo de aprendizaje de secuencias (Seq-L), respectivamente. Por último, se combinan las características extraídas para realizar la detección final de emociones. En [27], los autores plantean una alternativa que permita mantener el contexto temporal a largo plazo. Demuestran la mejora que aporta el utilizar un sistema de convoluciones paralelas integradas con una red *squeez-and-excitation* (PCNSE) para extraer las relaciones temporales y frecuenciales de un espectrograma 3D. Para la clasificación en sí, estas características extraídas sirven de entrada de una *self-attention Dilated Residual Network* (SADRN). En [28], implementan la versión 1D y 2D de una red capaz de adquirir de forma automática características tanto locales como globales. Para extraer las características locales, emplean cuatro bloques de aprendizaje de características locales (LFLB), realizando cada uno de ellos una convolución y diezmado. Para obtener las características globales, incluyen una capa LSTM previa a la clasificación.

Estudios basados en aprendizaje *hand-crafted*

En [29] realizan un modelo híbrido de clasificación que combina características extraídas de forma automática y características *hand-crafted*. En cuanto a las *hand-crafted*, seleccionan algunas como *Root-Mean Square Energy* (RMSE), *Mel-Frequency Cepstral Coefficients* (MFCC) o *Zero-crossing Rate* (ZCR). Para las profundas, utilizan arquitecturas preentrenadas como la VGG16 y

por último clasifican empleando una *Support Vector Machine*. En [30], se realiza una modificación en la obtención de los *Mel Frequency Magnitude Coefficients*. Únicamente empelando esta característica junto con una *Support Vector Machine* (SVM), obtienen resultados muy buenos en varias bases de datos. En [31], los autores implementan un modelo no lineal multinivel para la generación de características que emplea estructura criptográfica. Es un enfoque distinto donde emplean la *Tunnable Q Wavelet Transform* (TQWT) en lugar de espectrogramas, se seleccionan las características más relevantes de un sistema de generación y se clasifica.

En el caso del *cognitive computing*, debido al interés que suscita la automatización de servicios y de las interacciones humano máquina, su desarrollo es mucho mayor que el de la cancelación de eco. Es por ello que se hace más complicado aportar innovación y mejora respecto a los trabajos del estado del arte. Sin embargo, aunque no mejora sus resultados, en el presente proyecto se propone una idea innovadora que consiste en una red híbrida que combina las características extraídas con dos tipos de redes: una red convolucional y un *Vision Transformer*. Quizás con el tiempo suficiente y una investigación apropiada, esta arquitectura podría superar el rendimiento de los modelos actuales, lo cual queda como trabajo futuro de este proyecto.

1.3. Objetivos del proyecto

El objetivo principal de este trabajo fin de grado es diseñar, implementar y validar técnicas de aprendizaje profundo para análisis de audio, probando la utilidad de estas técnicas para dicho fin. Se pretende mostrar así la versatilidad de este tipo de tecnología implementándolo tanto en tareas de generación, como puede ser cancelación de eco, como en clasificación, en el caso de la detección de sentimientos en voz.

Las unidades básicas de operación son comunes a ambas tareas. Las convoluciones, funciones de activación, normalización y capas de modificación de la dimensionalidad son, en esencia, las mismas, aunque con modificaciones en cuanto al número de parámetros, capas y disposición para adaptarse a la funcionalidad.

Para solventar el problema de cancelación de eco acústico, se busca desarrollar una *Generative Adversarial Network* condicional (cGAN) que limpie la señal de voz sin modificarla ni introducir artefactos. Para la clasificación de sentimientos, se busca implementar una CNN combinada con un *Vision Transformer* capaz de identificar las emociones en el habla.

Para la realización del proyecto, se ha dividido el objetivo principal en varios objetivos secundarios. De esta manera, se ha establecido el alcance del trabajo y la secuenciación de las tareas para alcanzar resultados:

- Estado del arte y recopilación de los trabajos más recientes en los dos ámbitos de procesado de señal relativos a este trabajo (cancelación de eco y detección de sentimientos).
- Selección del método a emplear con distintas propuestas de mejora respecto al trabajo original.
- Selección de las bases de datos según criterios de calidad de las muestras y popularidad en la comunidad científica (mayor capacidad de comparación).
- Desarrollo de un modelo inicial: U-Net para cancelación de eco y CNN para detección de sentimientos.

- Comparación con los resultados del estado del arte.
- Inclusión de innovación y mejora respecto a lo obtenido previamente: cGAN para cancelación de eco y VGG16+*Vision Transformer* para detección de emociones.
- Comparación con los resultados del estado del arte y los modelos iniciales.
- Publicación de un artículo en revista o congreso si se supera el rendimiento de los modelos del estado del arte.

1.4. Estructura del documento

En el capítulo 2 se describen todas las bases de datos empleadas para cancelación de eco acústico y detección de emociones. También se comentan los programas y librerías empleadas para construir los modelos así como los dispositivos proporcionados por el grupo CVBLab. En el capítulo 3 se describe, en primer lugar, toda la teoría sobre la que se fundamenta tanto el preprocesado de la señal como diferentes técnicas de inteligencia artificial (los modelos de clasificación y aprendizaje profundo). Posteriormente, se muestran los métodos propuestos, tanto en cancelación de eco como en detección de sentimientos. En ambos casos se detalla el preprocesado de las grabaciones, la arquitectura o arquitecturas propuestas y la forma en la que se han optimizado. En el capítulo 4 se exponen las métricas de evaluación empleadas, los experimentos de ablación llevados a cabo, los resultados experimentales obtenidos y se compara el rendimiento de los modelos desarrollados con otros trabajos del estado del arte. En el capítulo 5, se cierra la memoria con una conclusión final y se proponen distintas posibilidades para seguir investigando en este campo.

Capítulo 2

Materiales

2.1. Bases de datos

2.1.1. Cancelación de eco

En cancelación de eco, normalmente se trabaja con bases de datos simuladas por los propios creadores de los modelos, pero no se suelen hacer públicas y esto perjudica considerablemente a la hora de comparar el rendimiento entre trabajos del estado del arte. Es por ello que se ha considerado la utilización de una base de datos pública y completa como es la de Microsoft para el entrenamiento del modelo presentado en este trabajo. De esta forma, es sencillo comparar los resultados del modelo planteado con los de otras publicaciones.

2.1.1.1. Microsoft synthetic database

La base de datos elegida para entrenar el modelo propuesto de cancelación de eco es el *synthetic dataset*[1] publicado por Microsoft para el AEC Challenge de 2021 [32]. Consta de 10.000 muestras sintéticas compuestas por escenarios *single-talk*, *double-talk*, ruido en el extremo cercano, ruido en el extremo lejano y varias situaciones de distorsión no lineal. Cada muestra incluye voz de extremo lejano, la señal de eco, la señal de extremo cercano captada por el micrófono y la voz del extremo cercano que será considerado el *ground truth*. Para las 10.000 muestras sintéticas, la relación señal a eco (SER) está uniformemente distribuida entre -10 y 9 dB en pasos de 1 dB. Para la evaluación del modelo, las primeras 400 muestras se han seleccionado tal y como se recomendaba en el *challenge*. El resto de grabaciones, excluyendo esas primeras 400, conforman el conjunto de entrenamiento.

[1] `https://github.com/microsoft/AEC-Challenge`.

Tabla 2.1: Resumen de la base de datos para cancelación de eco acústico.

Nombre	Synthetic dataset
Número de muestras	10.000
Duración muestras	10 s
Frecuencia muestreo	48 kHz
Idioma	Inglés
Año	2021

2.1.2. Detección de emociones

Como en la mayoría de trabajos relacionados con el *cognitive computing*, la profundidad y capacidad de generalización de los modelos depende en gran medida de la base de datos empleada y su heterogeneidad. Sin embargo, en el caso de detección de emociones mediante voz es complicado disponer de bases de datos suficientemente completas que permitan extrapolar los resultados por diversos motivos. Por un lado, no es una tarea trivial la selección de emociones. Si bien el psicólogo Paul Ekman [33] categorizó las emociones en 6 tipos básicos (furia, disgusto, felicidad, tristeza, miedo y sorpresa), no siempre resulta evidente seleccionar una de ellas. Surgen alternativas o categorizaciones complementarias como el modelo del reloj de arena [34] o la evaluación continua en valencia, excitación y dominancia [35], pero no existe unanimidad. Por otro lado, es relevante la forma de grabación ya que el ruido y la distorsión que pueda haber durante la captación de la voz afecta a la respuesta de la red neuronal. El hecho de emplear actores también perjudica a la generalización del modelo ya que en muchas ocasiones se exagera la emoción que tratan de mostrar. Por último, dependiendo de la cultura en la que nos encontremos inmersos, los sentimientos se expresan de forma distinta, complicando todavía más la tarea.

Teniendo todo esto en cuenta y considerando la disponibilidad y calidad de las bases de datos, se han elegido dos de ellas: EmoDB y RAVDESS

Tabla 2.2: Resumen de las bases de datos para detección de emociones en voz.

Nombre	Nº muestras	Actores	Duración media	Nº clases	Idioma	Año
EmoDB	500	5H+5M	2,7 s	7	Alemán	1999
RAVDESS	2,452	12H+12M	3 s	8	Inglés	2018

2.1.2.1. EmoDB

Una de las bases de datos elegidas para entrenar el modelo de detección de sentimientos en voz es *Berlin Database of Emotional Speech* (EmoDB) publicada en 1999. Dispone tanto de audios etiquetados como de las transcripciones de texto. Las grabaciones se realizaron en alemán por 10 actores (5 hombres y 5 mujeres) generando más de 500 archivos. Estos tienen una duración media de entorno a 2,7 segundos. Distingue entre 7 sentimientos: enfadado, aburrido, disgustado, asustado, feliz, triste y neutro.

2.1.2.2. RAVDESS

La otra de las bases de datos elegidas para entrenar el modelo de detección de sentimientos en voz es *Ryerson Audio-Visual Database of Emotional Speech and Song* (RAVDESS) publicada en 2018. Dispone tanto de audio como de vídeos etiquetados. Las grabaciones se realizaron en inglés de norte América por 24 actores (12 hombres y 12 mujeres) generando un total de 1440 archivos hablando y 1012 cantando. Estos tienen una duración media de 3 segundos. Distingue entre 8 sentimientos: calmado, feliz, triste, enfadado, asustado, sorprendido, disgustado y neutro.

2.2. Software

Para el desarrollo y la depuración de los modelos en este Trabajo Fin de Grado, se ha utilizado *Python*. Se trata de un *software* libre creado por Guido van Rossum a finales de la década de los 80. Es un lenguaje de programación de alto nivel que se centra en la legibilidad y estructura visual del código. Los desarrolladores enfocan sus esfuerzos en que las ideas puedan expresarse de la forma más clara y sencilla posible, lo cual se trata de su característica diferencial. En la actualidad, se utiliza en numerosos campos debido a su gran versatilidad, de entre los que podría destacar la bioinformática, neurofisiología, física, matemáticas o la representación 3D [36]. Para las aplicaciones relacionadas con la Inteligencia Artificial son imprescindibles las librerías mostradas en los apartados que siguen.

2.2.1. Numpy

Se trata de una librería de uso libre que permite trabajar de forma eficiente con vectores, siendo una herramienta muy potente para la computación. Dispone de todo tipo de funciones, constantes numéricas, transformadas de Fourier, etc. que simplifican y optimizan estas operaciones al trabajar con grandes cantidades de datos en forma vectorial. Es compatible además con muchos tipos de *hardware*, siendo posible implementarlo sobre la GPU. Su núcleo es en lenguaje de programación C.

2.2.2. Pandas

Es una librería también de uso libre que tiene como función el análisis y manipulación de datos en Python. Es una herramienta muy flexible que permite generar objetos *dataframe* rápidos y eficientes para manipulación de bases de datos o realizar etiquetados de dichas bases de datos, entre otros. Su núcleo es en lenguaje de programación C.

2.2.3. Matplotlib

Matplotlib es una librería muy completa que permite crear gráficas y visualizaciones estáticas, animadas e interactivas en Python. Utilizando esta herramienta es posible representar los datos, información y resultados de forma sencilla y clara. Resulta clave para analizar y comparar de forma rápida con otros trabajos en el estado del arte. Entre las funciones más destacadas se encuentran la posibilidad de crear representaciones de calidad y personalizadas, figuras interactivas sobre las

que aplicar zoom y actualizaciones instantáneas, o exportar dichas gráficas a cualquier formato para su almacenamiento.

2.2.4. Librosa

Librosa es una librería de Python que implementa una gran cantidad de funciones para el procesado de música y audio. De esta forma, se simplifica en gran medida la extracción de información y características de este tipo de señales. Se evita así tener que programar complicadas funciones matemáticas y poseer un amplio conocimiento del procesado de audio para ser capaz de utilizar las características que considero más relevantes.

2.2.5. Tensorflow

TensorFlow es una plataforma de código abierto de extremo a extremo para el Aprendizaje Automático (AA). Es el elemento clave para la implementación de técnicas de aprendizaje automático y mediante la cual se crean los modelos. Dispone de una API intuitiva de alto nivel denominada Keras que permite ejecutar y depurar de forma sencilla modelos iterativos de AA. Funciona tanto de forma local como en la nube e incluso en el navegador, sin importar el lenguaje empleado. Esta fluidez y flexibilidad hace que sea una de las herramientas más utilizadas en innovación e investigación en el campo del AA.

2.2.6. PyTorch

PyTorch es una librería que trabaja con tensores de forma optimizada para aplicaciones de aprendizaje profundo sobre GPU y CPU. Contiene distintas funcionalidades que facilitan y optimizan todos los procesos relacionados con el desarrollo de técnicas de aprendizaje profundo avanzadas. Es una herramienta muy potente que se emplea como alternativa a TensorFlow, aunque presenta algunas diferencias. Pythorch está más enfocada a una implementación orientada a objetos y es más reciente, se creó en septiembre de 2017, mientras que TensorFlow tiene más variedad posiblidades y empezó su desarrollo en 2011 aunque hasta noviembre de 2015 no se convirtió en un *software* libre.

2.3. Hardware

Debido a la complejidad computacional de las técnicas que se han desarrollado en este proyecto, CVBLab, el grupo de investigación en el que se ha realizado el trabajo, ha proporcionado parte de su infraestructura de computación de altas prestaciones como soporte. El actor principal de esta infraestructura de inteligencia artificial es el sistema NVIDIA DGX A100. Dicha máquina está compuesta por 8x NVIDIA A100 Tensor Core GPUs ofreciendo 5 petaFLOPS de potencia de cálculo y 320GB de memoria RAM. Adicionalmente, CVBLab dispone de seis servidores i7 @4.20GHz con 32GB de RAM que albergan potentes GPUs. En concreto, (1x) NVIDIA A100 Tensor Core GPU de 40GB, (4x) tarjetas gráficas NVIDIA Titan XP y (4x) tarjetas gráficas NVIDIA Titan V. El grupo también dispone de dos servidores NAS (Synology DS416 y Synology DS918+) en los que se alojan los datos y la documentación de los diferentes proyectos que se desarrollan con 32 y

16 TB de capacidad, respectivamente. Para llevar a término este TFG, sólo ha sido necesario hacer uso de uno de los servidores i7 @4.20GHz con 32 GB de RAM, su respectiva GPU NVIDIA Titan V y los servidores NAS.

Capítulo 3

Metodología

El núcleo metodológico del sistema extremo a extremo AEC propuesto es la cGAN mostrada en la figura 3.16. El método empleado para detección de sentimiento es una VGG16+ViT observable en la figura 3.19. A continuación, se muestra la teoría sobre la que se fundamenta el preprocesado de la voz y las redes neuronales y una descripción detallada de los diferentes bloques y procesos de ambos métodos.

3.1. Técnicas de preprocesado

Tradicionalmente, para realizar tareas de procesado de audio se han empleado o bien las señales en crudo, o bien características extraídas de las mismas y elegidas por un experto en la materia en función de la aplicación concreta. Esta forma de operar limita en gran medida la capacidad de los métodos implementados y dificulta la inclusión de técnicas de aprendizaje profundo. Por un lado, hacer uso de las señales sin aplicar ninguna transformación provoca que se pierda una gran cantidad de información. Por ejemplo, las características frecuenciales no son tan evidentes en las señales en bruto, por lo que es complicado sacar el máximo rendimiento a la hora de emplear redes neuronales. Por otra parte, si lo que se lleva a cabo es una selección manual de las características, es necesario disponer de los conocimientos adecuados para poder elegir con criterio. En ocasiones no resulta evidente qué componentes de la señal aportan mayor información para la tarea que se está llevando a cabo. Además, si se requieren implementaciones con limitaciones en latencia, hay que acotar todavía más las diferentes opciones de extracción de información.

Una solución válida, que es la que se aborda en este trabajo, es la obtención de espectrogramas. De esta forma, incluimos de manera detallada la información frecuencial además de la temporal y generamos una imagen de la señal. Esto permite a su vez la aplicación de capas convolucionales que extraigan las características de la señal automáticamente, por lo que es más sencillo seleccionar aquellas que sean óptimas y logren buenos resultados en la tarea objetivo.

A continuación, se describen el proceso de generación de un espectrograma y sus distintas variantes [37].

3.1.1. Short Time Fourier Transform en tiempo discreto

La transformada de Fourier (FT) es una transformación matemática que pasa una señal del dominio temporal al dominio frecuencial y es invertible. Dada una señal $s(t)$ definida en el tiempo, su transformada de Fourier $S(f)$ se obtiene mediante la siguiente ecuación:

$$S(f) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{+\infty} s(t) e^{-2j\pi ft} dt \tag{3.1}$$

Sin embargo, cuando se llevan a cabo técnicas de procesado de señal no se realiza la FT directamente. En su lugar, se emplea la *Discrete Fourier Transform* (DFT), el equivalente de la transformada de Fourier pero aplicado sobre secuencias discretas de duración finita. Dada una secuencia de N números complejos $x_0, x_1, \dots x_{N-1}$, es posible obtener su DFT a partir de la siguiente expresión:

$$S[k] = \sum_{n=0}^{N-1} x[n] e^{-j2\pi \frac{kn}{N}} \qquad k = 0, \dots, N-1 \tag{3.2}$$

Existe una aproximación a la DFT que permite calcularla de forma eficiente para aplicaciones de procesado de señal. Recibe el nombre de *Fast Fourier Transform* (FFT) y es la que realmente se utiliza en algoritmos computacionales.

La *Short Time Fourier Transform* (STFT) o TF dependiente del tiempo es una matriz que almacena información de magnitud y fase para cada punto en tiempo y frecuencia. Para generarla, se define un tamaño de ventana temporal y un paso, se divide la señal en tramas, generalmente con un cierto solape entre ellas, y se aplica sobre cada una la transformada de Fourier (a efectos prácticos es la FFT). Por tanto, la STFT se expresa como:

$$X[n, \lambda) = \sum_{m=-\infty}^{+\infty} w[m] x[n+m] e^{-j\lambda m} \tag{3.3}$$

donde n es una variable discreta que representa el instante de tiempo sobre el que se calcula la TF, mientras que λ es una variable continua que representa la pulsación. La $w[m]$ simboliza la ventana que divide la señal en tramas y la $x[n+m]$, la señal en sí.

Por tanto, la STFT es la transformación que permite generar una imagen tiempo-frecuencia. Por último, cabe destacar la importancia de seleccionar un tamaño de ventana adecuado. A mayor tamaño de ventana, peor resolución temporal y mejor resolución frecuencial y viceversa. Se aprecia claramente este efecto en la figura 3.2.

3.1.2. Espectrograma

El espectrograma es el módulo de la STFT, por lo que el proceso de generación es el mismo que el del apartado anterior. Dada una señal en el tiempo como la de la figura 3.1, tras conformar el espectrograma completo, se obtiene como resultado lo mostrado en la figura 3.2. En esta, se observa además el efecto que el tamaño de la ventana tiene sobre el mismo. En (a) se presenta el espectrograma de la señal de la figura 3.1 utilizando una ventana de tamaño 1024 puntos, en (b) la ventana usada es de tamaño 512 puntos y en (c) 256. Si observamos el efecto producido, se

puede apreciar que, a medida que disminuimos el tamaño de la ventana, aumenta la resolución en el tiempo pero, por contra, disminuye la resolución frecuencial y viceversa.

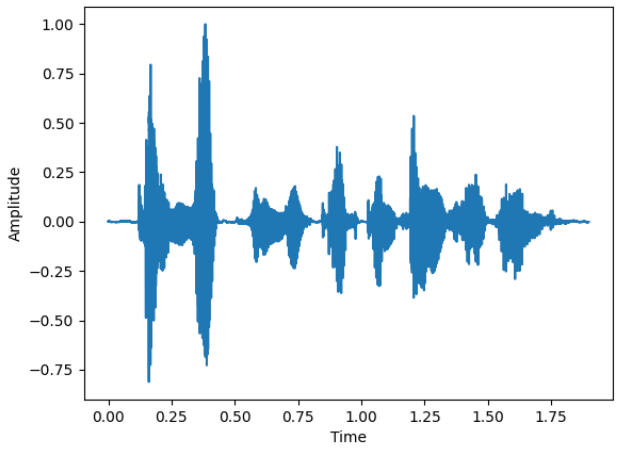
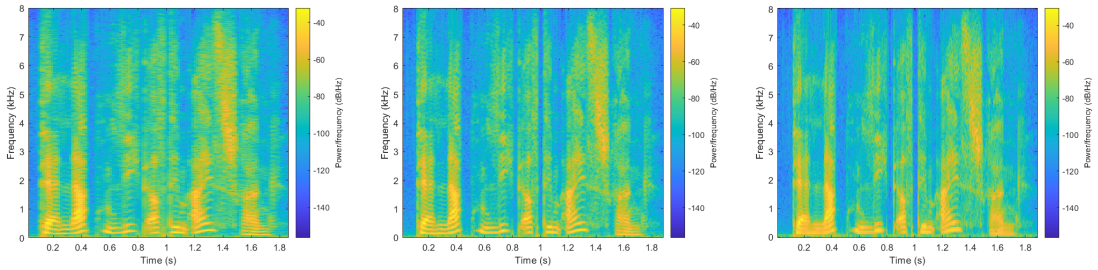


Figura 3.1: Señal en en el tiempo.

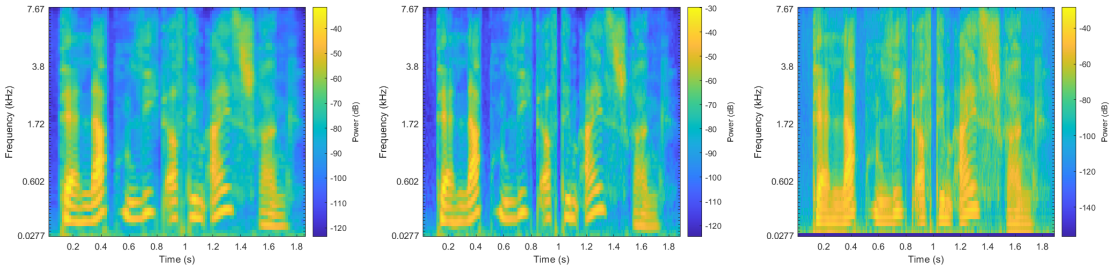


(a) **(b)** **(c)**

Figura 3.2: Espectrogramas de la señal en la figura 3.1. (a) Ventana de 1024 puntos, (b) Ventana de 512 puntos, (c) Ventana de 256 puntos.

3.1.3. Espectrograma de Mel

Este tipo de transformación es una adaptación del espectrograma clásico. Se trata de un banco de filtros que convierten el eje de frecuencias del espectrograma a escala de Mel. Resulta muy útil y es muy común en métodos de aprendizaje automático en aplicaciones del habla ya que dicha escala se adapta mejor a la voz humana. Interpreta el espectro de la misma forma que lo hacemos las personas, con una mayor resolución en frecuencias bajas que en frecuencias altas y detectando la intensidad del sonido de forma logarítmica (en dB). Se puede observar en la figura3.3.



(a) **(b)** **(c)**

Figura 3.3: Espectrogramas de Mel de la señal en la figura 3.1. (a) Ventana de 1024 puntos, (b) Ventana de 512 puntos, (c) Ventana de 256 puntos.

Para calcularlo, el procedimiento es similar al de la STFT. En primer lugar, la entrada de audio se almacena en tramas de un número determinado de muestras (del tamaño de la ventana). Las tramas se superponen tantos puntos como se le indique. A continuación, la ventana especificada se aplica sobre cada trama y cada una se convierte al dominio de la frecuencia. Posteriormente, cada trama del dominio frecuencial pasa a través de un banco de filtros de Mel. Por último, se suman todos los valores espectrales obtenidos a la salida del banco de filtros y los canales se concatenan de forma que cada trama se corresponde con un vector columna. Este proceso se puede observar en la figura 3.4.

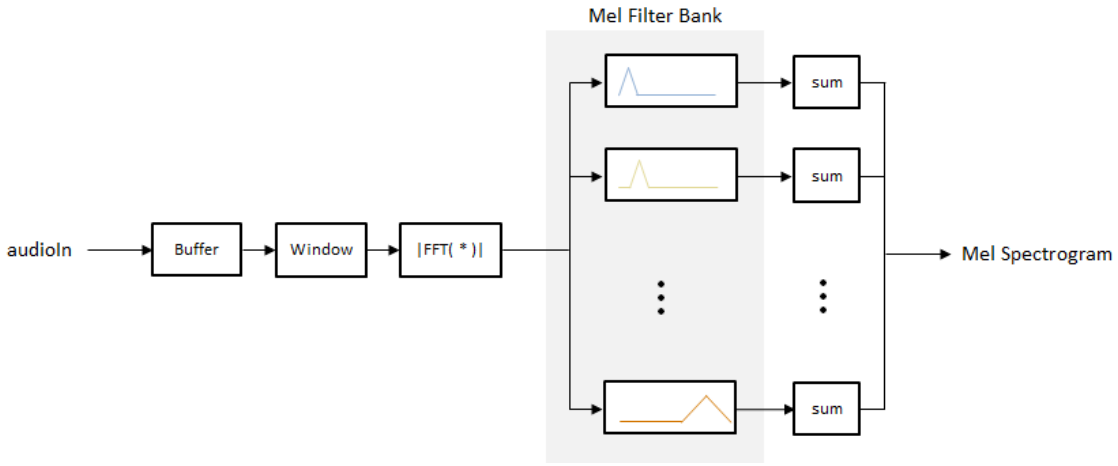


Figura 3.4: Obtención del espectrograma de Mel [38].

Al igual que ocurría en la STFT, el tamaño de ventana también determina el nivel de resolución temporal y frecuencial obtenido en la transformación. A mayor tamaño de ventana peor resolución temporal y mejor frecuencial y viceversa.

En cuanto al banco de filtros, está diseñado como un conjunto de N filtros triangulares solapados un $50\,\%$ e igualmente espaciados en la escala Mel, tal y como se puede observar en la figura 3.5

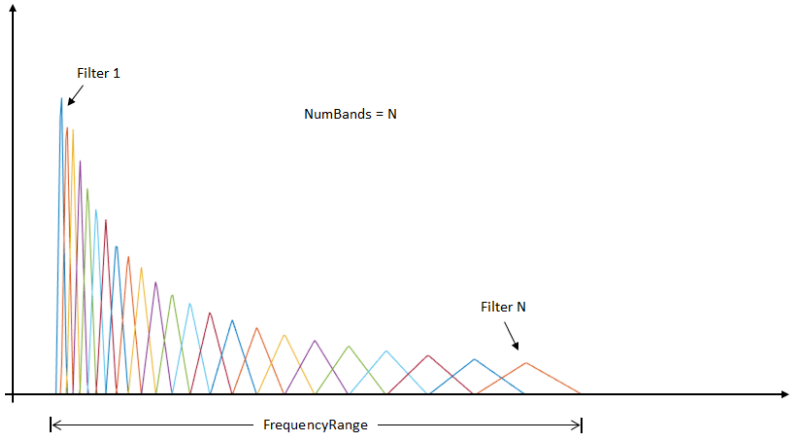


Figura 3.5: Banco de filtros de Mel [38].

3.2. Técnicas de *data augmentation*

El *data augmentation* es una técnica muy extendida en el ámbito de la inteligencia artificial que permite incrementar el número de muestras disponibles durante el entrenamiento generando nuevos datos a partir de los que ya hay. De esta forma, el conjunto de entrenamiento es más completo y con las clases más balanceadas. Existen muchas formas de llevar a cabo esta tarea cuando se trabaja con señales, como variando el tono o la intensidad. Sin embargo, al estar aplicando esta técnica en detección de emociones, los parámetros que se pueden modificar son escasos, ya que la red utiliza las características de la voz como información para identificar el sentimiento. Por tanto, se han empleado dos de ellas, desplazamiento temporal y adición de ruido.

3.2.1. Desplazamiento temporal

Este método consiste en desplazar la señal una cantidad de tiempo aleatoria ya sea hacia la derecha o hacia la izquierda. Posteriormente se concatenan ceros en el lado contrario hacia el que se ha realizado el desplazamiento para mantener las dimensiones iniciales.

3.2.2. Adición de ruido blanco gaussiano

El ruido blanco gaussiano o AWGN es una señal aleatoria e incorrelada cuya función de densidad sigue una distribución normal. De esta forma, si se suma con la señal de interés, aporta aleatorización y variación en todas las frecuencias sin enmascarar la información relevante.

3.3. Técnicas de aprendizaje profundo

Las técnicas de aprendizaje profundo son algoritmos de aprendizaje automático que permiten realizar una tarea sin la necesidad de que esta haya sido expresamente programada. A partir de un conjunto de datos, estos algoritmos son capaces de extraer la información relevante para llevar a

cabo la tarea que se les ha designado. Para ello, se construyen modelos secuenciales formados por capas de operadores lineales y sus parámetros se optimizan mediante métodos de entrenamiento. Dependiendo de la función que tienen designada y de las bases de datos disponibles, se puede realizar la optimización de parámetros mediante técnicas supervisadas, semisupervisadas o no supervisadas. Por tanto, las aplicaciones y posibilidades que tiene esta tecnología son infinitas así como el número de modelos diferentes que se pueden llegar implementar. Esta memoria se centrará solo en aquellos métodos y arquitecturas que han sido utilizadas a lo largo de este proyecto como son las redes convolucionales, el perceptrón multicapa, la LSTM, la U-Net y el Vision Transformer.

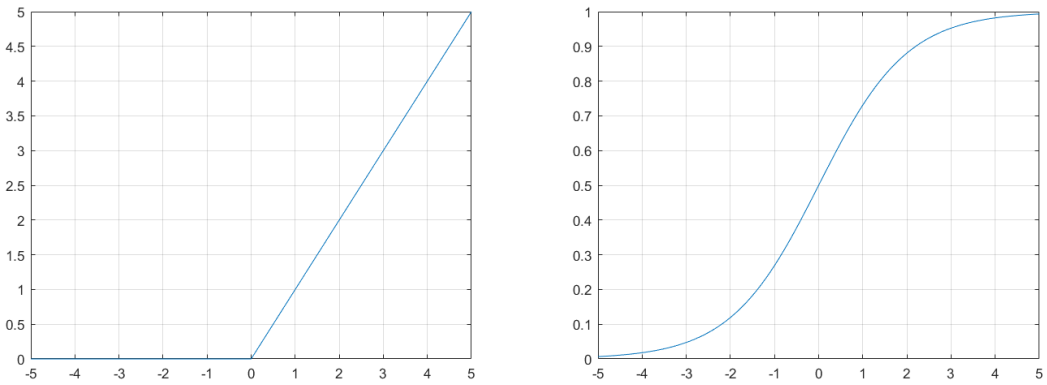
3.3.1. Perceptrón multicapa

El perceptrón es un algoritmo que fue diseñado para funcionar como una máquina de reconocimiento de patrones emulando a los humanos. Su unidad básica es la neurona, un nodo que realiza una suma ponderada de sus entradas y una función de activación que umbraliza dicha suma. La idea, tal y como se comenta en [39], es crear una máquina capaz de conceptualizar entradas físicas como son la luz, el sonido, la temperatura, etc. y ser capaz de interpretarlo para llevar a cabo un objetivo concreto.

Con esta umbralización tras la función de activación, el perceptrón puede emplearse como un modelo de clasificación binario. Este umbral determina la frontera de decisión lineal que minimiza la distancia entre los puntos y dicha frontera.

Habitualmente, como método de optimización para determinar el hiperplano que divide el espacio de cada clase se utiliza el descenso estocástico por gradiente. Esta función optimiza tanto la función de activación como la suma ponderada para que se ajuste de la mejor manera al conjunto de datos de entrada.

En cuando a la función de activación, en un principio se empleaba la sigmoide. La forma se adapta al objetivo de mapear la entrada a 0 o 1 mediante la apliación de una función no lineal. Sin embargo, actualmente se utiliza casi siempre la Rectified Linear Unit (ReLU) de la figura 3.6a debido a que se adapta mejor al descenso estocástico por gradiente y es mucho más eficiente computacionalmente.



(a) **(b)**

Figura 3.6: Funciones de activación. (a) ReLU, (b) Sigmoide.

Sin embargo, no es posible aplicar este mecanismo a conjuntos de datos no lineales. Como solución, surge lo que se denomina como perceptrón multicapa.

El perceptrón multicapa es una red neuronal compuesta por una capa de entrada, una o varias capas ocultas y una capa de salida que se utiliza tanto para clasificación binaria como multiclase. Entrando más en detalle, las capas ocultas son una agrupación de neuronas en paralelo, alimentadas por las capas anteriores y que disponen de sus propios pesos optimizables. De forma iterativa, la salida de cada capa, llamada activaciones, alimenta a la siguiente hasta llegar a la última de ellas, la capa de salida, que aplica una función de activación para efectuar la clasificación en sí. Se puede apreciar claramente su estructura en la figura 3.7.

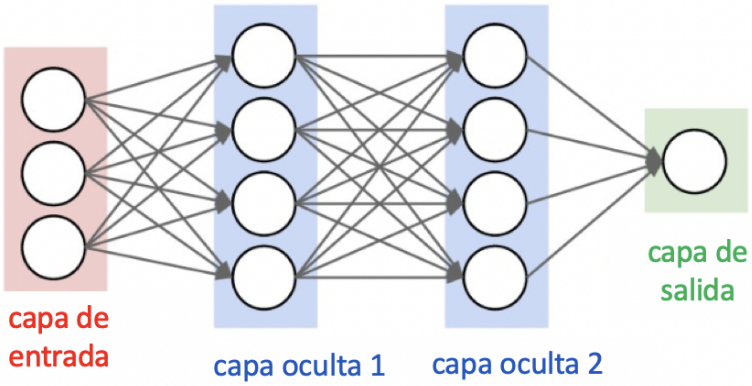


Figura 3.7: Perceptrón multicapa

En cuanto a la función de activación de la capa de salida, varía en función de si el problema es de clasificación binaria o si es de clasificación multiclase. En el caso de clasificación binaria se emplea una sigmoide definida a partir de la eq. 3.4. Se puede observar en la figura 3.6b que, si los valores de entrada son negativos, la predicción será la clase 0 mientras que si son positivos, será la clase 1.

$$S(x) = \frac{1}{1 + e^{-x}} \tag{3.4}$$

Por otro lado, si se trata de clasificación multiclase, se utiliza la función de activación softmax definida a partir de la eq. 3.5. Esta normaliza un conjunto de K valores reales en una distribución de probabilidad que suma 1. Se aprecia que para $K = 2$, las funciones softmax y sigmoide son equivalentes.

$$S(x) = \frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_i}} \tag{3.5}$$

Las capas que componen el MLP se llaman densas o completamente conectadas ya que cada una de las neuronas de una capa están conectadas con todas las neuronas de la capa siguiente. Para optimizar los pesos de las neuronas de cada capa se llevan a cabo dos tareas, *forward* y *backpropagation*. En primer lugar se realiza el proceso de *forward*, que consiste en la obtención de una

predicción tras introducir los datos de entrada en la red. En este proceso se calculan las activaciones de todas las capas utilizando los pesos de las neuronas. Posteriormente, se realiza el proceso de *backpropagation*, que actualiza los pesos de cada una de las neuronas de forma iterativa con el objetivo de minimizar una función de coste. Para ello, la función debe ser diferenciable por lo que un ejemplo válido podría ser el error cuadrático medio. Para optimizar los pesos en sí, una vez se ha realizado el proceso de inferencia, se calcula el gradiente de la función de coste para cada par de entrada-salida de cada neurona y, con este valor, se optimizan los parámetros de las mismas.

Se puede observar el proceso descrito anteriormente en la figura 3.8.

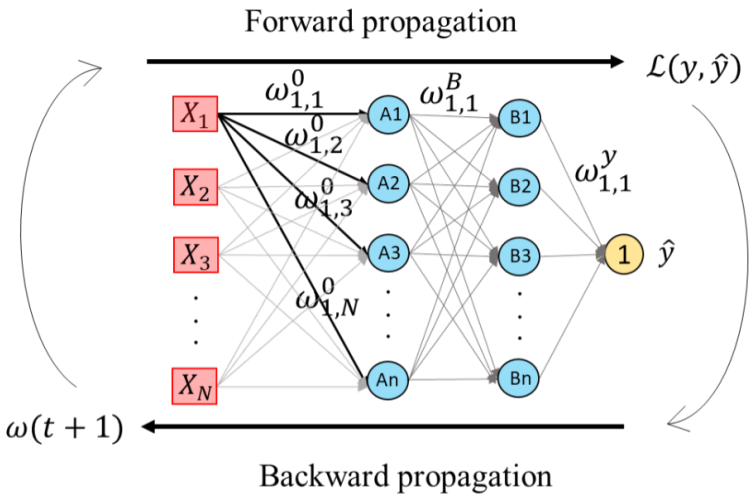


Figura 3.8: Proceso *forward-backward*.

donde $L(y, \hat{y})$ hace referencia a la función de coste a minimizar, ω a los pesos actuales y $\omega(t+1)$ a los pesos actualizados.

Una función de coste habitual en tareas de clasificación es la entropía cruzada. Este algoritmo mide la diferencia entre dos distribuciones de probabilidad. Por tanto, dadas dos distribuciones cualesquiera P y Q, ante un evento x, la entropía cruzada se calcula como:

$$H(P, Q) = -\sum_{x} P(x) \log Q(x) \tag{3.6}$$

Para evitar el sobreentrenamiento, es habitual colocar entre las capas densas lo que se denomina como capas *dropout*. Estas desactivan algunas conexiones entre neuronas de forma aleatoria durante la optimización de parámetros evitando así que se especialicen para el conjunto de entrenamiento.

3.3.2. Red neuronal convolucional

Las redes neuronales convolucionales, CNN por sus siglas en inglés, son algoritmos de aprendizaje profundo que, a partir de una imagen de entrada, son capaces de extraer información mediante

filtros optimizados por entrenamiento. De esta forma, es posible reducir la dimensionalidad de las mismas y caracterizarlas sin perder información relevante.

La arquitectura de las CNN es análoga al patrón de conectividad de las neuronas de un cerebro humano. Su organización se inspira en la del córtex visual, donde cada neurona responde a estímulos en una única región y, uniéndolas, se cubre todo el área visual.

Para tareas de clasificación, si bien es verdad que se podría transformar la imagen en un vector e introducirlo en el perceptrón multicapa directamente, esto resulta bastante ineficiente. El perceptrón por sí solo no es capaz de identificar las relaciones existentes entre píxeles para imágenes complejas por lo que la precisión sería bastante baja, además que podrían generarse vectores enormes. La red convolucional, por contra, sí que captura las dependencias espaciales mediante la aplicación de filtros. Esto permite reducir el número de parámetros necesarios y utilizar eficientemente bases de datos extensas.

Estos filtros se denominan kernels. Son pequeñas matrices que suelen tener un tamaño de entorno a 3×3, que se convolucionan con la imagen de entrada. De forma iterativa, aplican operaciones a un grupo de píxeles y se desplazan al siguiente grupo.

Entre las capas de una red convolucional es posible distinguir tres tipos distintos:

- **Capas convolucionales**. Formadas por bloques de kernels, son las encargadas de extraer la información relevante. Las primeras capas de este tipo extraen características de bajo nivel, mientras que las últimas, extraen las de alto nivel. Realizan la convolución entre la matriz de entrada y el kernel tantas veces como kernels haya. Los coeficientes de los kernels son los parámetros entrenables de la red.
- **Capas de activación**. Compuestas por una función matemática, aplican una transformación no lineal a la entrada para provocar la convergencia del modelo.
- **Capas de diezmado**. Estas capas sirven para reducir la dimensionalidad de la imagen. Disponen de un kernel que, para cada iteración del filtrado, toma o bien el máximo de los píxeles que engloba o bien la media, disminuyendo así el tamaño de la matriz de entrada. Permiten simplificar y seleccionar la información relevante.
- ***Flatten***. Tras aplicar varias capas convolucionales y de diezmado, se convierte la matriz resultante en un vector para introducirlo en el perceptrón multicapa. Es esta capa la que se encarga de realizar la vectorización de la matriz de salida para posteriormente clasificar.

De esta forma, para construir una red convolucional se deben disponer de forma secuencial las capas anteriores tal y como se puede ver en la figura 3.9. Se van alternando capas convolucionales, capas de activación y capas diezmado para la extracción de características y, tras el *flatten*, se realiza la clasificación. Suele ser interesante incluir capas de normalización, pues ayudan a la convergencia del modelo.

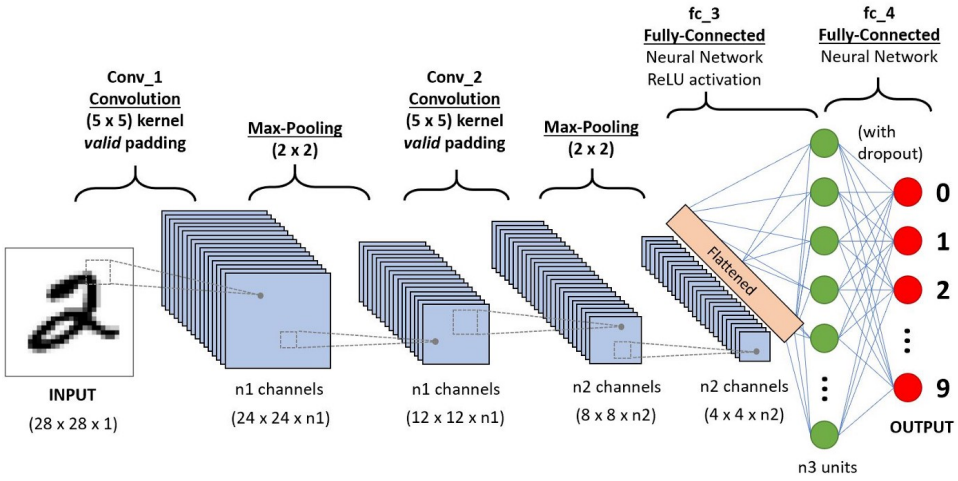


Figura 3.9: Red neuronal convolucional [40].

3.3.3. *Generative Adversarial Networks*

Las redes generativas adversariales, GAN por sus siglas en inglés, son modelos generativos basados en aprendizaje profundo. Estos algoritmos no tratan de resolver el problema típico de clasificación, por lo que funcionan de diferente manera que los métodos de aprendizaje máquina clásicos. Crean una salida nueva a partir de la entrada de forma que no sea posible diferenciar una muestra generada de una real. Esto permite, además, incorporar técnicas de aprendizaje no supervisado que completen y potencien el modelo.

Las GANs (figura 3.10a) se componen de dos bloques:

- **Generador**. Encargado de realizar la tarea generativa. Normalmente se implementa con *autoencoders* o arquitecturas similares como U-Nets.
- **Discriminador**. Trata de diferenciar si la entrada es una muestra generada o una real. Normalmente se implementa con CNNs.

En este TFG se ha implementado una red generativa adversarial condicional (figura 3.10b). El término condicional hace referencia a la información adicional que se introduce como entrada del modelo y que resulta imprescindible para las tareas tanto de generación como de discriminación. En el caso de la cancelación de eco, la entrada del modelo sería el audio a limpiar, mientras que la información condicional, el eco en sí.

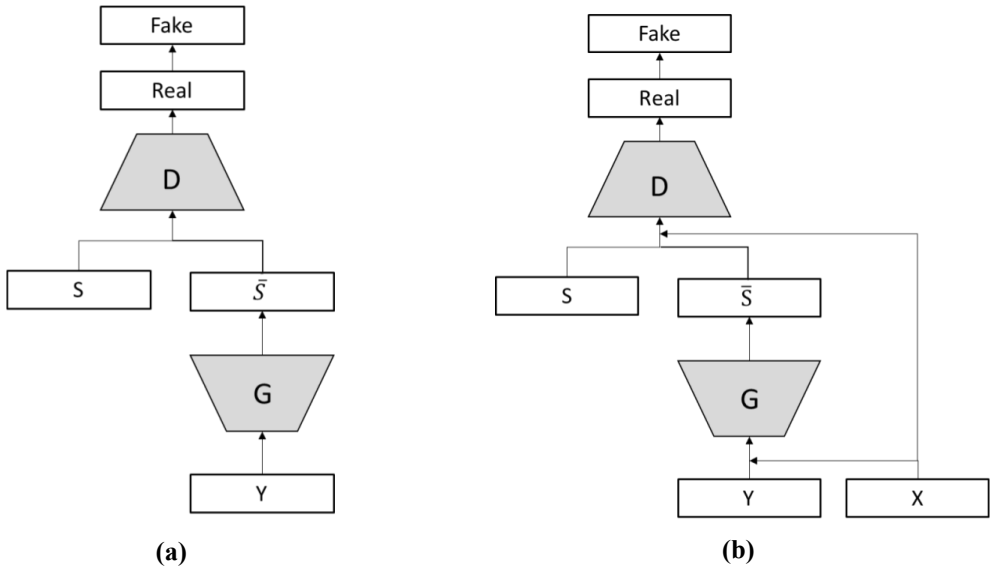


Figura 3.10: Esquemas de redes generativas adversariales. (a) GAN, (b) cGAN.

3.3.4. U-Net

Se trata de una arquitectura empleada principalmente para segmentación semántica. Se compone de dos ramas, una contractiva y una expansiva.

La fase contractiva sigue la estructura de una red neuronal convolucional convencional; se apilan secuencialmente bloques compuestos por una capa convolucional, una de activación lineal y una de diezmado. Generalmente, se construyen de forma que la capa convolucional de cada bloque tenga el doble de filtros que la del bloque anterior, mientras que las capas de diezmado suelen tener un factor de redimensionado de $\frac{1}{2}$. A este proceso también se le denomina *downsampling*.

Tras atravesar esta primera fase contractiva (o *encoder*), la imagen ha quedado reducida a un vector de características, y es lo que se conoce como vector de estado latente.

A continuación, prosigue la parte expansiva, es decir, la que realiza la función generativa. Esta fase presenta la estructura inversa a la contractiva, aunque en lugar de incluir capas de diezmado, implementa capas de interpolación (*upsampling*). Estas capas redimensionan la matriz de entrada por un factor de 2, revirtiendo la reducción aplicada previamente. Después, se toma la salida del bloque homólogo de la fase contractiva, se concatena con la entrada del bloque actual y se procesa de igual forma que durante el *downsampling*.

De esta forma, se han extraído las características más importantes que definen la imagen de entrada y se ha reconstruido con las dimensiones que tenía inicialmente. Además, durante la reconstrucción es posible generar la segmentación, suprimir el ruido, eliminar el eco o cualquier tarea que le sea designada.

Existe una variación de la U-Net que consiste en incluir bloques residuales en lugar de convolucionales. Funcionan de la misma forma que los bloques convolucionales, pero antes de realizar el diezmado se suman la matriz procesada y la matriz de entrada al bloque. De esta manera, se mantiene la información de la imagen de entrada y se suavizan en gran medida los problemas de

desvanecimiento por gradiente. Esto se debe a que las capas convolucionales no tienen que predecir por completo la salida sino el gradiente, que al sumarlo con la entrada, da un resultado óptimo.

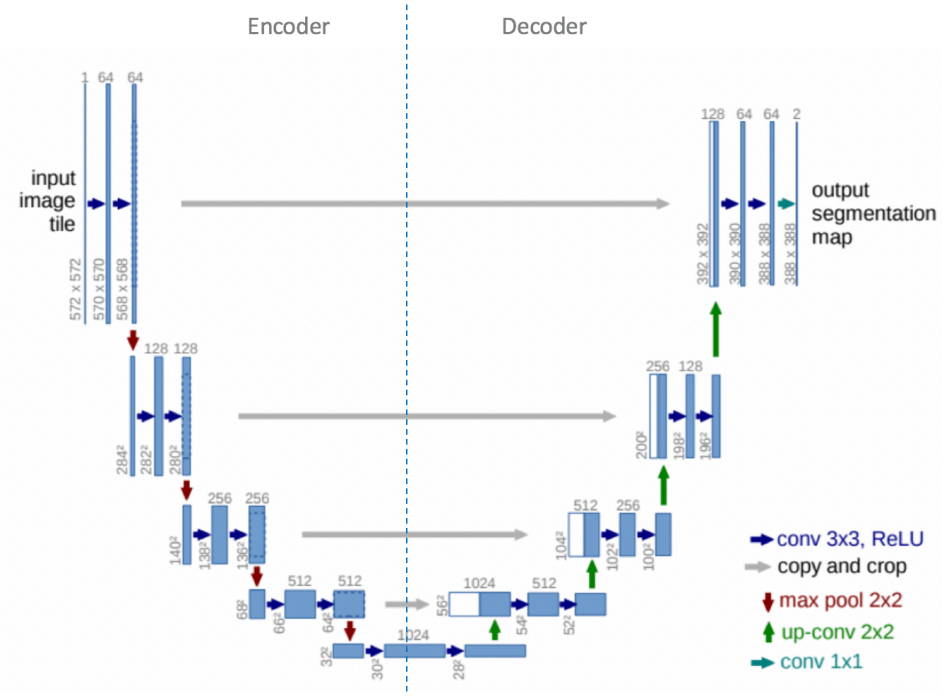


Figura 3.11: U-Net [41].

3.3.5. Long Short-Term Memory

Las *Long-Short Term Memory* (LSTM) son redes neuronales recurrentes con la capacidad de manejar dependencias a largo plazo. Están diseñadas de forma que reducen los problemas de desvanecimiento por gradiente que tienen las redes recurrentes (también llamadas persistentes).

Por ejemplo, cuando estás viendo una película, utilizas la información de *frames* consecutivos para comprender lo que está ocurriendo en la escena ya que, con uno solo, no es posible leer el movimiento o la intención de los personajes. Para dependencias temporales a corto plazo, las redes recurrentes pueden llegar a funcionar, pero cuando se buscan correlaciones a largo plazo, los gradientes provocan que la información se vaya perdiendo, por lo que no tienen la relevancia que deberían en instantes posteriores.

Para tratar de resolver estos problemas surgen las LSTMs. El núcleo de una red LSTM está compuesto por un sistema de tres celdas denominadas puertas: puerta del olvido, puerta de entrada y puerta de salida. Además, dispone de dos estados ocultos, el estado del instante anterior, $H(t-1)$, y el del actual $H(t)$, y de dos estados de celda, el del instante actual y del anterior, $C(t)$ y $C(t-1)$. El estado oculto se corresponde con la memoria a corto plazo, mientras que el estado de celda, con la memoria a largo plazo.

Las puertas son los elementos que modifican los estados oculto y de celda en función de la entrada y cada una tiene una función específica:

- Puerta del olvido. Esta puerta determina si la información procedente del instante anterior

se recuerda o si se olvida. La ecuación que rige esta parte es:

$$f_t = \sigma(x_t \cdot U_f + H_{t-1} \cdot W_f), \;\; con \;\; f_t \, \epsilon \, [0, 1] \tag{3.7}$$

donde

- $X_t =$ Entrada en el instante actual
- $U_f =$ Pesos de la entrada
- $H_{t-1} =$ Instante anterior del estado oculto
- $W_f =$ Pesos de la capa oculta

Si el valor de f_t es 0 se olvida todo y si es 1, se mantiene.

- Puerta de entrada. Esta puerta intenta aprender nueva información de la entrada de la celda.

$$i_t = \sigma(x_t \cdot U_i + H_{t-1} \cdot W_i), \;\; con \;\; i_t \, \epsilon \, [0, 1] \tag{3.8}$$

donde

- $X_t =$ Entrada en el instante actual
- $U_i =$ Pesos de la entrada
- $H_{t-1} =$ Instante anterior del estado oculto
- $W_i =$ Pesos de la capa oculta

Este parámetro se utiliza para determinar cuánto de la nueva información se añade o se quita mediante la siguiente ecuación.

$$N_t = \tanh{(x_t \cdot U_c + H_{t-1} \cdot W_c)}, \;\; con \;\; N_t \, \epsilon \, [-1, 1] \tag{3.9}$$

donde

- $X_t =$ Entrada en el instante actual
- $U_c =$ Pesos de la entrada
- $H_{t-1} =$ Instante anterior del estado oculto
- $W_c =$ Pesos de la capa oculta

Si el valor de N_t es negativo, se substrae información y si es positivo, se añade. Tomando estos parámetros, la actualización del estado de celda sería:

$$C_t = f_t \cdot C_{t-1} + i_t \cdot N_t \tag{3.10}$$

- Puerta de salida. Esta puerta pasa la información actualizada del instante actual al instante siguiente. Para ello, realiza las siguientes operaciones:
  - Ecuación de la puerta de salida: $o_t = \sigma(x_t \cdot U_o + H_{t-1} \cdot W_o), \;\; con \;\; o_t \, \epsilon \, [0, 1]$
  - Actualización del estado oculto actual: $H_t = o_t \cdot \tanh C_t$
  - Salida del instante actual: $Output = softmax(H_t)$

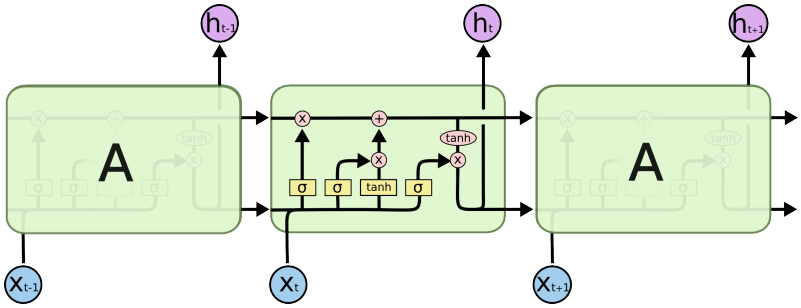


Figura 3.12: LSTM [42]. En este esquema, se observa cómo la información va atravesando la LSTM repetidas veces a lo largo del tiempo y cómo se procesa una vez se encuentra en su interior. La h_t hace referencia al estado oculto y la x_t, a la información de entrada.

3.3.6. Vision Transformer

En el paradigma de las técnicas de aprendizaje máquina para secuencias, las primeras aproximaciones estuvieron basadas en redes neuronales recurrentes (RNN) en una arquitectura *encoder-decoder*. Sin embargo, tiene una gran limitación cuando se trabaja con secuencias extensas. Debido a los problemas de desvanecimiento por gradiente, se va perdiendo información de las primeras muestras conforme se incorporan nuevos elementos y como el *decoder* sólo tiene acceso al último estado, es difícil establecer relaciones entre muestras alejadas. Las LSTM pretenden solventarlo añadiendo el estado de celda comentado anteriormente, pero no siempre se comportan de forma idónea.

Surgen entonces como alternativa los mecanismos de atención. Estos permiten que, en lugar de prestar atención al último estado obtenido por el *encoder*, sea posible observar todos los estados al mismo tiempo. De esta forma, se puede extraer información de la secuencia completa mediante una suma ponderada de los estados anteriores y darle el peso correspondiente a cada uno en función de la entrada. Sin embargo, este método sólo es capaz de procesar secuencias de una en una, no es posible paralelizar esta operación, por lo que para bases de datos muy extensas, el coste computacional y temporal es excesivamente elevado.

Es este contexto nacen los Transformers, cuando en 2017 se publica *Attention is all you need* [43]. El modelo Transformer extrae características mediante mecanismos de *self-attention* por los que se establecen relaciones de cada elemento de la secuencia con el resto. Para ello, no utilizan ningún tipo de recurrencia, sino que son todo sumas ponderadas y funciones de activación, por lo que es posible paralelizar y acelerar procesos en gran medida.

Como se puede apreciar en la figura 3.13, esta arquitectura se compone de un *encoder*, a la izquierda, y un *decoder*, a la derecha. Ambas partes disponen de bloques comunes como son *multi-head attention*, *feed forward* y *positional encoding*.

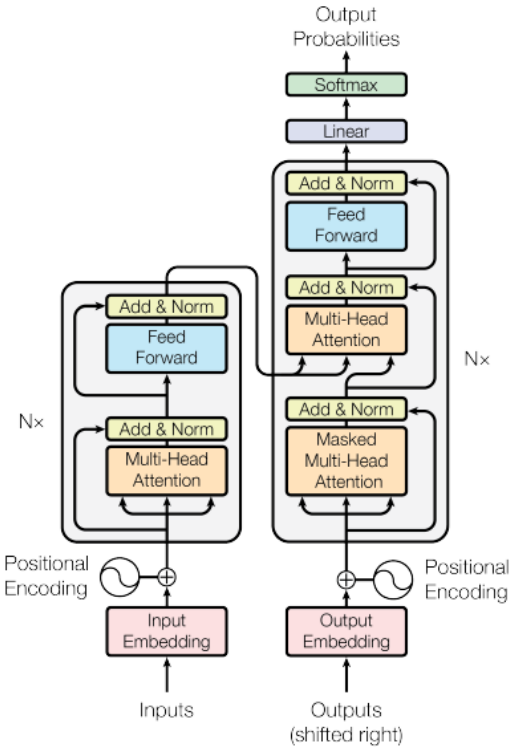


Figura 3.13: Transformer [44]. En este esquema se observa claramente el bloque del *encoder*, a la izquierda, el del *decoder*, a la derecha y el *positional encoding*, a la entrada de ambos bloques. Además, dispone de un *top model* compuesto por una capa lineal y una *softmax* a la salida del *decoder*.

Para empezar, es preciso comprender en qué consiste el mecanismo de *self-attention*. Es un proceso que realiza un cambio de subespacio sobre la secuencia de entrada y la manera más sencilla de llevarlo a cabo es mediante el llamado *scaled dot-product attention* 3.14. Para ello, primero se multiplica cada vector de entrada por tres matrices de pesos, K, Q y V, que dan como resultado los vectores denominados *Queries*, *Keys* y *Values*, respectivamente. Los pesos o coeficientes de estas matrices se aprenden durante el entrenamiento. Estos vectores permiten determinar el peso que tienen el resto de elementos de la secuencia sobre el que se está evaluando mediante una serie de operaciones. A continuación, el vector *Query* de un elemento de la secuencia se multiplica con el vector *Keys* del resto de elementos. Esto da como resultado lo que se conoce como *Scores*. Posteriormente, se normaliza y se aplica una *softmax* sobre dicho vector de *Scores* y este se multiplica por el vector de *Values* de todos los elementos de la secuencia. Por último, se suman todos los resultados de los productos, obteniendo como salida la *self-attention* del vector en cuestión. Y así con cada uno de los vectores de entrada que componen la secuencia.

Una vez sabido esto, es posible comprender el bloque de *multi-head attention* de la figura 3.14. Un *attention head* es el elemento que realiza el proceso de *self-attention* comentado anteriormente. Por tanto, en el mecanismo de *multi-head attention*, lo que se hace es aplicar múltiples *attention heads* al mismo tiempo y de forma paralela. La salida obtenida de todos ellos se concatena y finalmente se multiplica por una matriz de pesos obteniendo así la salida del bloque.

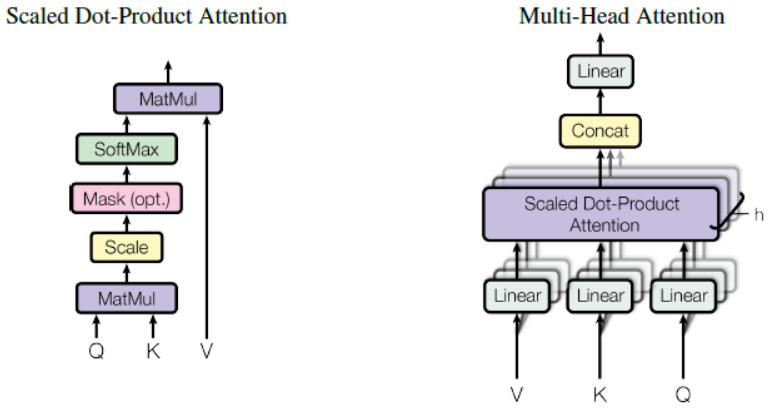


Figura 3.14: Multi-head attention [44]. El esquema de la izquierda representa el proceso de obtención de los vectores *Queries*, *Keys* y *Values*, es decir, el *scaled dot product attention*. El de la derecha, muestra la arquitectura del *multi-head attention*, con varios bloques de los anteriores en paralelo que, a su salida, se concatenan y se aplica una capa lineal.

Otro elemento clave en el Transformer es el *positional encoding*. Al procesar ahora la señal en paralelo mediante los mecanismos de atención y no secuencialmente, se pierde el orden. Con el fin de introducir de nuevo el orden en el procesado, se suma a la entrada un valor que indica la posición. La forma más común de *positional encoding* es la sinusoidal dada por la ecuación:

$$P(k, 2i) = \sin\left(\frac{k}{n^{\frac{2i}{d}}}\right) \tag{3.11}$$

$$P(k, 2i+1) = \cos\left(\frac{k}{n^{\frac{2i}{d}}}\right) \tag{3.12}$$

aunque el valor de posición se puede también aprender en el entrenamiento. La k hace referencia a la posición del objeto en la secuencia de entrada, la i se utiliza para mapear a los índices de las columnas, la d es el tamaño del vector de características y la n es un escalar arbitrario del orden de 10.000.

Por último, el bloque *feed forward* permite seleccionar y adaptar la salida del bloque *multi-head attention* mediante transformaciones no lineales para el siguiente bloque *multi-head attention*. Si esto no se hiciera, simplemente se estarían reevaluando los vectores de *Values* continuamente y el modelo no convergería.

Una vez explicados los bloques más importantes de la figura 3.13 por separado, sólo faltaría ver la disposición de los mismos. Para la codificación y decodificación se sigue un núcleo común. En primer lugar se aplica el *multi-head attention* y se suma la salida de este bloque con la entrada del mismo. Después, esta matriz se introduce en el bloque *feed forward* y de nuevo se suma la salida del bloque con la entrada del mismo. Este proceso se repite N_x veces.

El *Vision Transformer* funciona de la misma forma pero adaptado a imágenes. Se necesita un proceso previo que codifique la imagen o partes de ella en vectores de características(*embedings*). En este caso, los vectores de entrada son parches de 16×16 directamente convertidos en un vector mediante una proyección lineal o *flatten*. En el ejemplo de la figura 3.15, el *top model* es un

perceptrón multicapa (MLP) que realiza la tarea de clasificación. Estos se introducen en el *encoder* del Tranformer y finalmente se añade un *top model* que podría tener cualquier forma, ya sean estructuras generativas o clasificadores.

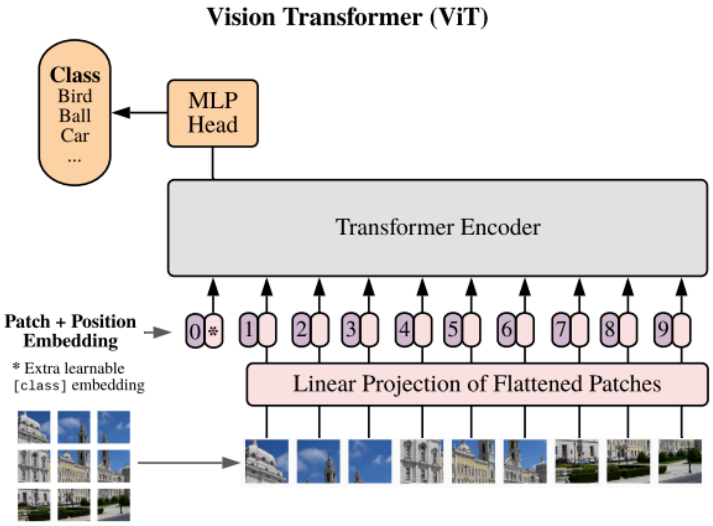


Figura 3.15: Vision transformer [45]

3.4. Métodos propuestos

3.4.1. Cancelación de eco acústico

3.4.1.1. Procesado de la señal

En el contexto del sistema AEC mostrado en figura1.1, la señal de micrófono $y(n)$ puede expresarse como una mezcla de la voz del extremo cercano $s(n)$ y la voz del extremo lejano $x(n)$ según:

$$y(n) = s(n) + f(x(n)) + v(n)\,, \tag{3.13}$$

donde $v(n)$ denota el ruido ambiente y $f(x(n)) = d(n)$ es la señal del eco. La eq. (3.13) describe un escenario *double-talk* genérico cuando $s(n) \neq 0$ o, alternativamente, un escenario *single-talk* cuando $s(n) = 0$. Respecto al eco $f(x(n))$, normalmente se asume que se trata de una función lineal tal que $d(n) = x(n) * h(n)$. Sin embargo, se han considerado efectos no lineales sobre $x(n)$ como recortes o deterioros debido a la respuesta de la sala al impulso (RIR) variante en el tiempo.

Como se muestra en la figura 3.16, la entrada a la cGAN son segmentos de 155 ms de la señal del micrófono, $y(n)$, y de la señal de extremo lejano $x(n)$ (resaltado con un recuadro rojo en la parte inferior izquierda de la figura3.16). En el segmento de 155ms a analizar, el primer bloque de 135ms se corresponde con información previa, el siguiente bloque de 10ms conforma el *frame* de interés, mientras que el último bloque de 10ms es información posterior. A continuación, se obtiene la *Short Time Fourier Transform* (STFT) de los segmentos de 155ms empleando una ventana *Hanning* de $w = 318$ muestras y 75 % de solapamiento. Después, se obtienen los espectrogramas

normalizados $Y(k,m)$ y $X(k,m)$ de tamaño 160×32 a partir de las señales $y(n)$ y $x(n)$, respectivamente. Se hace de esta forma para no tener un gran retardo, ya que esta aplicación está pensada para tiempo real. Los módulos de los espectrogramas complejos se utilizan como características de entrada al modelo propuesto. Remarcar que, durante la fase de inferencia, el proceso de normalización se invierte, obteniendo la predicción del módulo del espectrograma de la voz del extremo cercano, $|\hat{S}(k,m)|$. Después, se utiliza la fase de $y(n)$ para calcular $\hat{s}(n)$ como la STFT inversa de $|\hat{S}(k,m)|e^{\angle j\,Y(k,m)}$. De cada predicción, solo el *frame* de interés de 10ms del segmento de 155ms es seleccionado para construir la voz en el extremo cercano $\hat{s}(n)$ estimado. En el caso de escenarios *single-talk* donde $s(n)$ no está presente, la salida del modelo no tendrá ningún tipo de contenido de voz, aunque también se denotará como $\hat{s}(n)$ por claridad. Para simplificar la notación, el operador de módulo será omitido cuando haga referencia a espectrogramas para el resto del documento.

3.4.1.2. cGAN para cancelación de eco

Dado $\mathcal{T} = \{(S_1, X_1, Y_1), (S_2, X_2, Y_2), ..., (S_N, X_N, Y_N)\}$, compuesto por N tríos de espectrograma limpio (S), espectrograma ruidoso (Y) y espectrograma del extremo lejano (X), el problema de cancelación de eco es encontrar un mapeo tal que $f(Y) : Y \longmapsto S$. De acuerdo al principio de las GAN, la cGAN propuesta tiene su generador (G) encargado de realizar el mapeo a una señal sin eco y un discriminador (D) que refina las señales sintetizadas. Sin embargo, en el *framework* propuesto, G y D reciben información condicional de entrada para controlar la salida. Una vez presentado Y junto con la representación del eco X (la información condicional), G produce la señal mejorada $\hat{S} = G(Y, X)$. Posteriormente, el discriminador (D) recibe un par de señales de entrada, $\{S \cup X\}$ y $\{\hat{S} \cup X\}$, y D aprende a clasificar los pares (S, X) como real y $(\hat{S}, X)$ como falso, mientras que G intenta engañar a D de forma que clasifique $(\hat{S}, X)$ como real.

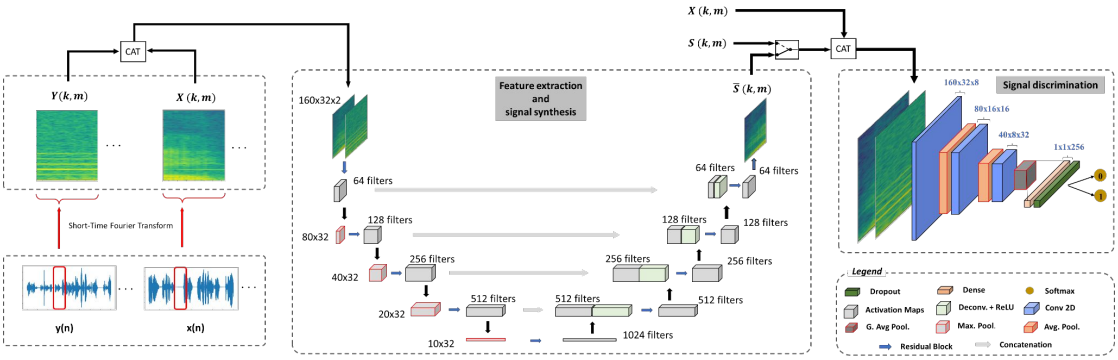


Figura 3.16: Modelo propuesto para realizar la cancelación de eco acústico. Los espectrogramas de la señal de micrófono (Y) y de la voz del extremo lejano (X) se concatenan y alimentan al generador (*Extracción de características y síntesis de señal*). A continuación, el espectrograma predicho $(\hat{S})$ se concatena con el espectrograma de la voz del extremo lejano (X) y alimenta al discriminador (*Discriminación de la señal*).

3.4.1.3. Generador cGAN para sintetización de señales

El generador está realizando una tarea de traslación imagen a imagen. Normalmente se emplean modelos basados en *auto-encoders* para este problema pero, debido al *downsampling*, mucha in-

formación puede perderse. Además, el flujo de información de imagen pasa a través de todas las capas, incluyendo el cuello de botella. Así, a veces se intercambian muchas características redundantes no deseadas (entradas y salidas comparten muchos píxeles). Por esta razón, en este trabajo se emplean conexiones de salto siguiendo la estructura de una residual U-Net [46]. La residual U-Net está compuesta por dos ramas: *encoder*, encargado de obtener características relevantes de la imagen y *decoder*, a cargo de la reconstrucción de la imagen objetivo.

Como se muestra en la figura 3.16, los espectrogramas de la señal del micrófono de extremo cercano (Y) y del extremo lejano (X) se utilizan como entradas, cada uno como imagen en escala de grises que componen un volumen de entrada de dos canales de tamaño $160 \times 32 \times 2$. La rama de codificación se compone de bloques convolucionales apilados, forzando a la red a centrarse en correlaciones temporalmente cerradas sobre la señal de entrada. Estas conexiones han mejorado la optimización de los modelos de aprendizaje profundo, evitando problemas de desvanecimiento por gradiente en otras aplicaciones U-Net. Respecto a la operación de diezmado entre bloques convolucionales, únicamente se realiza en la dimensión frecuencial utilizando filtros de tamaño 2×1. Debido al pequeño tamaño de los *frames* de entrada, un diezmado temporal podría suavizar el espectrograma y degradar la representación de la voz.

En la rama *decoder*, los bloques convolucionales son deconvoluciones que recuperan la dimensión espacial progresivamente. La red G también se caracteriza por sus conexiones de salto, que conectan cada capa de codificación a su capa homóloga de decodificación y pasan información detallada del espectrograma de entrada al *decoder*. Además, ofrecen mejor comportamiento en entrenamiento, ya que los gradientes pueden fluir de forma más profunda a través de toda la estructura. Finalmente, una capa convolucional de tamaño 1×1 reconstruye el espectrograma estimado de la voz en el extremo cercano, $\hat{S}$.

3.4.1.4. Discriminador cGAN (síntesis vs real)

El discriminador (D) se encarga de transmitir la información a G de qué es real y qué es falso. De esta forma, G puede corregir ligeramente su salida hacia una distribución realista, eliminando las componentes del eco que la delatan como falsa. Por tanto, D puede expresarse como el aprendizaje de cierto tipo de pérdidas para que la salida de G parezca real. En este caso, D está compuesto por un clasificador implementado mediante una *Convolutional Neural Network* (CNN) con dos entradas. el espectrograma predicho por el generador ($\hat{S}$) y la información condicional (X). $D : \{\hat{S} \cup X\} \rightarrow Z$ mapea los espectrogramas generado $\hat{S}$ a un vector de *embedding* Z, donde la etapa de clasificación se aborda en un espacio de menor dimensión. Al final de la red convolucional. Una capa densa de activación mediante *softmax* se aplica para efectuar la clasificación (falso vs real).

3.4.1.5. Optimización de la cGAN

El generador y discriminador se entrenan de forma adversarial, donde ambas redes juegan al tira y afloja, e intentan maximizar su propia función de utilidad. El generador intenta engañar al discriminador produciendo muestras muy cercanas a las del conjunto de entrenamiento, mientras que el discriminador intenta clasificar correctamente entre datos reales y falsos.

Por un lado, durante la fase de entrenamiento del generador, el aprendizaje se lleva a cabo mediante el error entre los espectrogramas limpio y generado y las pérdidas proporcionadas por el discriminador para cada iteración. Denotemos el módulo del espectrograma de la señal objetivo

como $S(k,m)$, donde m denota el índice del *frame*, $m = 1,\dots,M$, y k es el *bin* de frecuencia. La función de pérdidas que mide el error entre espectrogramas se define como *Root Mean Square Error* (RMSE):

$$\mathcal{L}_{RMSE} = \sqrt{\frac{1}{M(w/2+1)} \sum_{m=1}^{M} \sum_{k=0}^{w/2} \left[\hat{S}(k,m) - S(k,m)\right]^2}. \tag{3.14}$$

La función de pérdidas elegida para el discriminador es la *categorical cross-entropy* y se espera que sea mínima cuando la pareja $\{\hat{S}, X\}$ sea real:

$$\mathcal{L}_D = \sum_{m=1}^{M} D(\hat{S}, X) \cdot \log_{10} \hat{a}\,, \tag{3.15}$$

donde $\hat{a}$ denota el *ground-truth* del discriminador. Por tanto, la función objetivo usada en el *back propagation* es una combinación ponderada de las pérdidas anteriores:

$$\mathcal{L}_G = \mathcal{L}_{RMSE} + \alpha \cdot \mathcal{L}_D\,. \tag{3.16}$$

Por otro lado, durante la etapa de entrenamiento del discriminador, el proceso de aprendizaje es únicamente llevada a cabo por las pérdidas del propio discriminador cada dos iteraciones. La función de pérdidas es también la *categorical-crossentropy* (3.15) pero, en este caso, se espera que prediga la pareja $\{\hat{S}, X\}$ como falso y $\{S, X\}$ como real.

3.4.2. Detección de sentimientos

En esta sección se describe, en primer lugar, el procesado de señal llevado a cabo para obtener la imagen de entrada de las diferentes configuraciones que se proponen para la detección de emociones. A continuación, se describen las dos configuraciones utilizadas para la extracción local de características del audio: una CNN diseñada y construida ad-hoc para este problema y el *fine tuning* de una arquitectura preentrenada, la VGG16. Una vez descritas las dos formas de extraer características, se presentarán las dos estrategias de clasificación planteadas: la más simple, un perceptrón multicapa, y una modificación de la misma insertando previamente al perceptrón una capa LSTM. Finalmente se presenta una arquitectura híbrida que combina las características extraídas localmente con las obtenidas con un *Vision Tansformer*, clasificando el vector final con un perceptrón multicapa.

3.4.2.1. Procesado de la señal de audio

La entrada a todas las arquitecturas que se van a estudiar van a ser los espectrogramas de Mel de los audios completos de las bases de datos respectivas, no se realiza una segmentación como en el caso anterior, ya que en este caso no tenemos el condicionante de tiempo real. Esto resulta más interesante ya que mantener las correlaciones y la información de la muestra completa conlleva un mejor rendimiento del modelo y una mayor velocidad de inferencia. Sin embargo, debido a la variabilidad en la duración de las grabaciones, se ha fijado el tamaño de las mismas a 80000

puntos, es decir, 5 segundos para una frecuencia de muestreo constante de 16 kHz. Posteriormente, se obtiene el espectrograma de Mel de los audios empleando una ventana *Hanning* de $w = 2048$ muestras, 75 % de solapamiento y 128 filtros de Mel. Después, se realiza una conversión de la amplitud de los espectrogramas a dB y, finalmente, se obtienen los espectrogramas normalizados $X(k, m)$, de tamaño 128×157 a partir de la señal $x(n)$. Al tratarse de un problema de clasificación, la inferencia no requiere de un postprocesado, el propio modelo obtiene el resultado final.

3.4.2.2. Extracción de características locales

Mediante CNN ad-hoc

Tal y como se ha explicado en la sección 3.3.2, las primeras capas de una red neuronal convolucional extraen características de bajo nivel, mientras que las últimas, de alto nivel. Esta sección del modelo tiene por objetivo extraer y seleccionar las características locales de la imagen de entrada, por lo que interesa que la red sea profunda y disponga de gran cantidad de filtros. Para ello, se ha realizado una arquitectura de 4 bloques como muestran la figura 3.17-(a), donde cada uno de ellos dispone de una capa convolucional, una de normalización, una de activación y una de diezmado. Sin embargo, aunque la estructura es constante, la parametrización de las capas varía tal y como se observa en la Tabla 3.1.

Tabla 3.1: Tamaño y paso de los filtros de la CNN

Blocks	Convolution			Pooling	
	filters	kernel_size	stride	kernel_size	stride
Block 1	64	(3,3)	(1,1)	(2,2)	(2,2)
Block 2	64	(3,3)	(1,1)	(2,2)	(2,2)
Block 3	128	(3,3)	(1,1)	(2,2)	(2,2)
Block 4	128	(3,3)	(1,1)	(4,4)	(4,4)

El hecho de que los filtros aumenten de forma progresiva es una práctica habitual en los modelos basados en aprendizaje profundo. Conforme se accede a capas más profundas, es posible identificar más características, por lo que el número de *kernels* se adapta para que el modelo trabaje de forma más eficiente. Finalmente, se añade una capa de linealización o *flatten* que reorganiza las características de salida como un vector unidimensional.

Mediante *fine tuning* de VGG16

Si bien es verdad que es posible construir una CNN desde cero, la gran cantidad de parámetros no asegura que se alcance un punto de convergencia óptimo y de forma eficiente. En 2014, la red que ganó la competición de reconocimiento de imagen ILSVR(Imagenet) fue la VGG16. Con 16 capas y cerca de 138 millones de parámetros, demostró un rendimiento excelente en tareas de procesado y clasificación de imagen por lo que, desde que fue publicado, se ha utilizado en numerosas ocasiones como base para *transfer learning* y *fine tunning*. Estas técnicas permiten, empleando la misma arquitectura, tomar como valores iniciales los parámetros que se alcanzaron tras entrenar la red con la base de datos Imagenet. En el *transfer learning* se congelan (no se entrenan) todos los pesos de la red convolucional y se añade un *top model* para realizar la clasificación y en el *fine tunning*

se descongelan (se reentrenan con las nuevas imágenes) parte de los parámetros y se añade un *top model*. De esta forma, es posible alcanzar resultados buenos sin tener que realizar entrenamientos muy largos ni tener que disponer de bases de datos extensas.

La estructura de la red, mostrada en la figura 3.17-(b), presenta una capa de entrada que debe ser una imagen de 3 canales. A continuación, le siguen dos bloques compuestos por dos capas convolucionales con función de activación ReLU y una capa de diezmado. Por último, dispone de tres bloques, cada uno compuesto por tres capas convolucionales con función de activación ReLU y una capa de diezmado. Como se puede observar en la Tabla 3.2, el número de filtros de las capas convolucionales es creciente, al igual que en el caso anterior, y el factor de redimensionado de las capas de diezmado es constante, de $\frac{1}{2}$.

Tabla 3.2: Tamaño y paso de los filtros de la VGG16

Blocks	Convolution			Pooling	
	filters	kernel_size	stride	kernel_size	stride
Block 1	64	(3,3)	(1,1)	(2,2)	(2,2)
Block 2	128	(3,3)	(1,1)	(2,2)	(2,2)
Block 3	256	(3,3)	(1,1)	(2,2)	(2,2)
Block 4	512	(3,3)	(1,1)	(2,2)	(2,2)
Block 5	512	(3,3)	(1,1)	(2,2)	(2,2)

En las implementaciones en las que se incorpora esta arquitectura, ha sido necesario incluir una capa convolucional previa que transformara el espectrograma en una imagen de tres canales, ya que solo tiene uno. Además, en todo momento se han congelado las dos primeras capas, es decir, no se han entrenado, para preservar los pesos originales. Por último, en la parte final del modelo, se ha añadido una capa de *global max pooling* para linealizar las características. Esta capa toma el valor máximo de cada punto de la imagen para cada canal, generando así un vector unidimensional de tamaño $1 \times 1 \times n_{caracterstcias}$.

3.4.2.3. Estrategias de clasificación

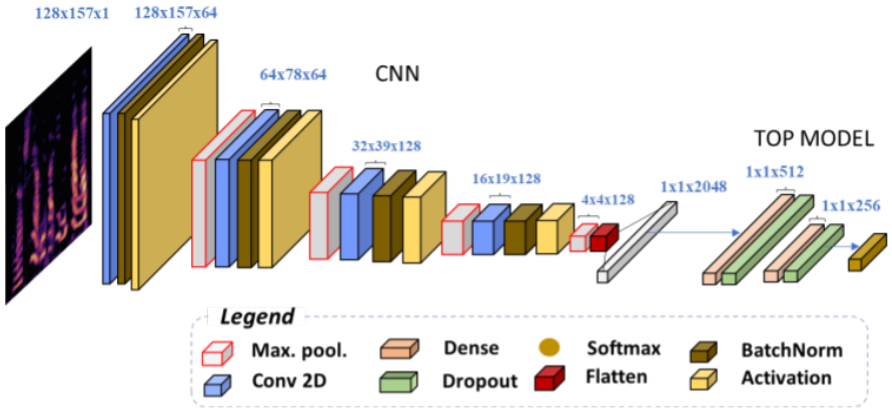
Como se ha dicho anteriormente, las estrategias de clasificación planteadas con las características extraídas usando ambos métodos propuestos en la sección 3.4.2.2 son: un perceptrón multicapa y un perceptrón multicapa con una LSTM previa. A continuación presentamos ambas arquitecturas.

Perceptrón multicapa

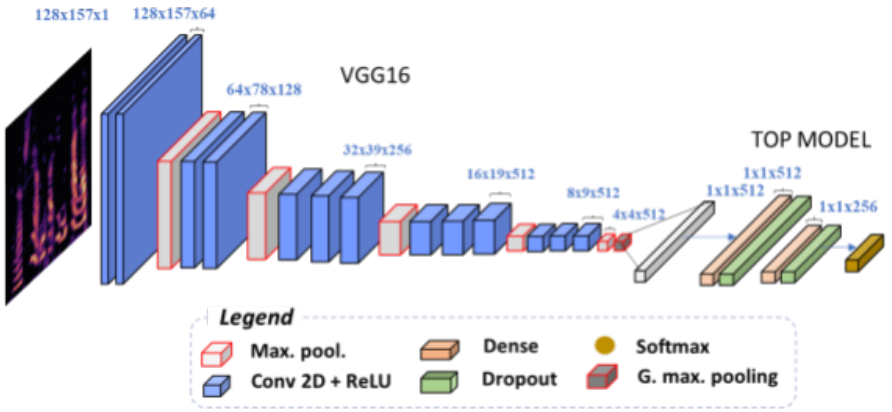
Tras obtener los vectores de características extraídos por los bloques anteriores, se introduce el resultado en el perceptrón multicapa, encargado de realizar la clasificación final. Para los métodos propuestos, se ha encontrado óptima la disposición que se observa en la figura 3.17, con dos capas densas de 512 y 256 neuronas intercaladas con capas *dropout*. De esta forma, se consigue suficiente precisión y se evita en la medida de lo posible el sobreentrenamiento. Como primera aproximación, se ha considerado oportuno establecer una base de referencia empleando una arquitectura simple y así poder apreciar el valor que aportan los distintos bloques. Para ello, en primer lugar se ha construido un modelo compuesto por una red neuronal convolucional y un perceptrón multicapa.

Por tanto, las dos opciones que se han evaluado en este caso son:

- La CNN de la sección 3.4.2.2 junto con el perceptrón multicapa de la sección 3.4.2.3. Obsérvese en la figura 3.17-(a).
- La VGG16 de la sección 3.4.2.2 junto con el perceptrón multicapa de la sección 3.4.2.3.Obsérvese en la figura 3.17-(b).



(a)



(b)

Figura 3.17: El espectrograma de Mel calculado se introduce en (a) la red convolucional ad-hoc o (b) la VGG *fine tuned*. El vector de características resultante sirve como entrada para el clasificador final, el perceptrón multicapa

.

Perceptron multicapa+LSTM

Para poder establecer relaciones temporales a largo plazo, se ha considerado oportuno incorporar a las arquitecturas anteriores una LSTM en serie, con la salida de la red neuronal convolucional como entrada e introduciendo su salida en el perceptrón multicapa. Esta capa permite establecer relaciones temporales a largo plazo dentro de una secuencia temporal. Por tanto, tomando como entrada el vector de características resultante de las redes neuronales convolucionales, resulta interesante añadir una LSTM que determine estas dependencias internas. Debido al gran número

de parámetros que requiere, seleccionar un número de unidades elevado puede comprometer el tiempo de inferencia del modelo. Se ha seleccionado entonces un valor de 512 unidades para la LSTM debido a que ofrece un compromiso entre rendimiento y coste computacional. Se pretende comprobar así si las relaciones y la información que esta capa aporta mejoran el rendimiento del modelo. En este caso se ha evaluado la combinación VGG16 de la sección 3.4.2.2 junto con la LSTM de la sección 3.4.2.3 y el perceptrón multicapa de la sección 3.4.2.3. Dicha configuración puede observarse en la figura 3.18. En este caso, la capa de *global max. pooling* se modifica para que el vector de características disponga de contexto temporal. La combinación con la CNN-adhoc se ha descartado ya que, como se verá en el capítulo de resultados, se demuestra que la VGG16 es la arquitectura que extrae las mejores características.

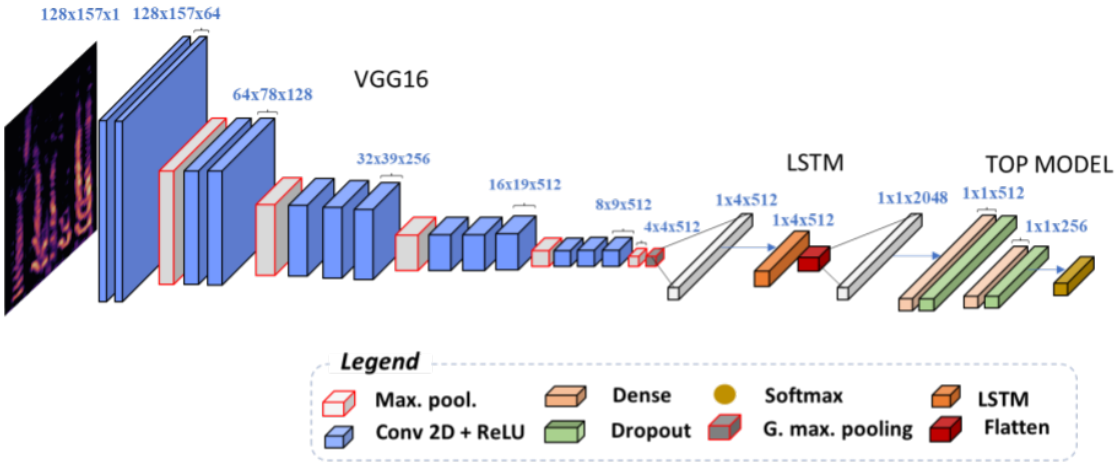


Figura 3.18: El espectrograma de Mel calculado se introduce en la VGG16 *fine tuned*. El vector de características resultante sirve como entrada para la LSTM. La salida de esta sirve a su vez como entrada para el clasificador final, el perceptrón multicapa

.

3.4.2.4. Introduciendo *Vision Transformer* en la detección de emociones

En este método, con el objetivo de establecer relaciones a nivel global se ha optado por combinar las características extraídas por un Transformer *encoder* con las extraídas por una VGG16. El transformer actúa en paralelo a la red convolucional, teniendo como entrada el espectrograma de Mel y concatenando su salida con la de la VGG16. Este vector generado es el que se introduce en el perceptrón multicapa para efectuar la clasificación final. Esta arquitectura es la mostrada en la figura 3.19.

El *Vision Transformer*, como se ha comentado previamente, permite establecer las relaciones tanto espaciales como temporales dentro del espectrograma. Esto es posible gracias al funcionamiento intrínseco del Transformer; se vectoriza la imagen por parches generando una secuencia y se establecen relaciones ponderadas entre los diferentes parches que componen la imagen. Por tanto, esta sección del modelo recibe una imagen de entrada de 128×157 correspondiente al espectrograma de Mel del audio, la parchea y la codifica mediante el Transformer *encoder*. Los parámetros de los bloques que conforman la arquitectura y que ofrecen un resultado óptimo son los siguientes:

- Para la secuenciación de la imagen, según los autores que propusieron el *Vision Transformer*

en [45], el tamaño de parches recomendado es 16x16. Se ha estudiado la posibilidad de modificar este valor pero los mejores resultados se han obtenido con parches de este tamaño.

- La codificación posicional se realiza sumando un parámetro optimizable a cada uno de los vectores generados durante el parcheo.
- Para el mecanismo de atención y *feed forward*, se ha utilizado un bloque predefinido e implementado en el artículo [47]. En él, los autores comparan su método con los referentes en el estado del arte, donde muestran la validez de su método de *self-attention*.

Finalmente, se promedian las características obtenidas para cada uno de los vectores. De esta forma, el espectrograma permanece caracterizado y se reduce la dimensionalidad en gran medida, disminuyendo así el tiempo de inferencia.

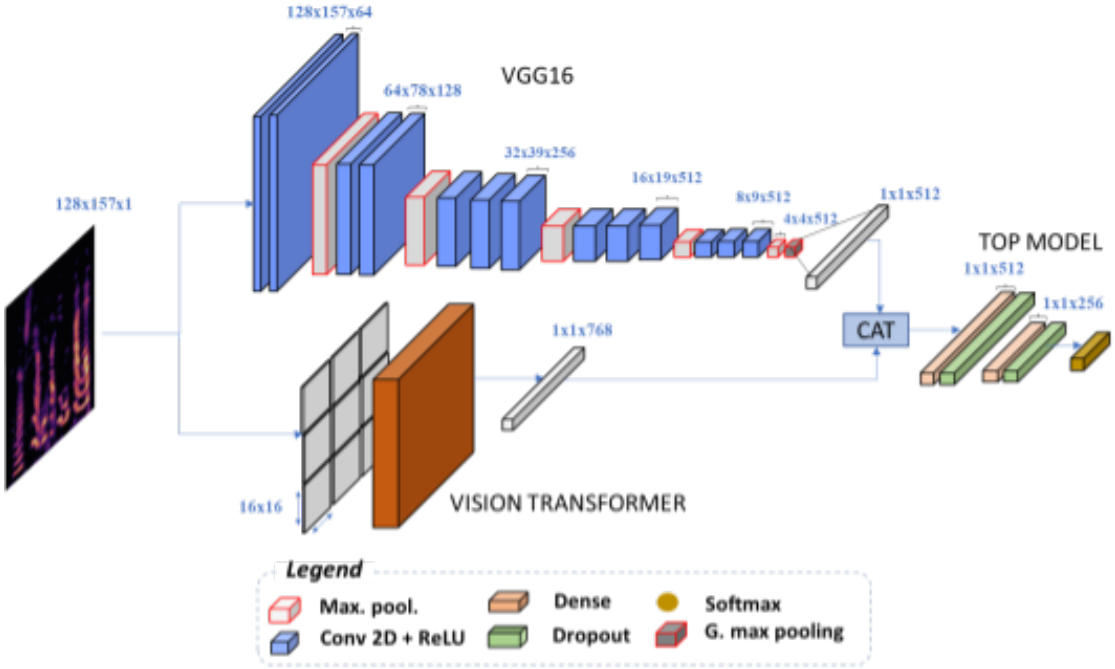


Figura 3.19: El espectrograma de Mel calculado se introduce en la rama superior en la VGG16 *fine tuned*, y en la rama inferior en el Transformer. Los vectores de características resultantes de ambas ramas se concatenan y sirven como entrada para el clasificador final, un perceptrón multicapa

.

3.4.2.5. Optimización de los modelos

En este caso, el proceso de entrenamiento y optimización de los modelos sigue una estructura mucho más estándar en comparación con la cancelación de eco. Durante la fase de entrenamiento, la función de pérdidas empleada es la *categorical cross-entropy* para todos los modelos propuestos. Cada iteración en la que se evalúa un *batch*, se propagan las pérdidas a través de la arquitectura para actualizar los pesos de las diferentes capas. Para lograr el mejor resultado posible, se monitoriza la

precisión en validación y el modelo que se almacena finalmente es aquel que ha logrado un mejor resultado, independientemente de la época.

Capítulo 4

Resultados

4.1. Cancelación de eco

4.1.1. Métricas de evaluación

Se han empleado el *Echo Return Loss Enhacement* (ERLE) y la *Perceptual Evaluation of Speech Quality* (PESQ) [48] como las métricas de evaluación de rendimiento. El ERLE evalúa la reducción de eco conseguida durante los periodos de *single-talk*, cuando la voz en el extremo cercano no está presente, de la siguiente forma:

$$\text{ERLE} = 10\log_{10}\frac{E[y^2(n)]}{E[\hat{s}^2(n)]} \approx 10\log_{10}\frac{\sum_n y^2(n)}{\sum_n \hat{s}^2(n)}\,, \tag{4.1}$$

donde la operación de esperanza estadística $E[\cdot]$ se estima calculando la media temporal sobre la duración del discurso evaluado. PESQ es un método objetivo para predecir la calidad subjetiva de la voz (hasta 3,1 kHz) inicialmente propuesto en 2001 para evaluar la distorsión de los códecs de voz de banda estrecha que se usan en telefonía (voz sobre IP) [48]. Con los años se ha convertido en una métrica estándar para, en general, medir la calidad de voz obtenida en comparación con la voz limpia. En este proyecto, se usará para medir la calidad de voz *near-end* después de pasar por el cancelador de eco, $\hat{s}(n)$, respecto a la voz original $s(n)$, es decir, en escenarios *double-talk*. La puntuación de PESQ varía de $-0{,}5$ a $4{,}5$, donde los valores más altos se corresponden con una mayor calidad del discurso.

4.1.2. Experimentos de ablación

Con tal de optimizar los parámetros de la cGAN, se han llevado a cabo diversos experimentos. Utilizando la opción de entrenamiento, se ha testeado de forma cruzada distintos valores de $\alpha = \{0{,}02, 0{,}05, 0{,}1, 0{,}5\}$, el factor de ponderación de las pérdidas del discriminador en la ecuación de pérdidas totales (3.16), y se han obtenido y representado sus respectivas puntuaciones de PESQ de las señales sintetizadas del discurso, mostradas en la figura 4.1.

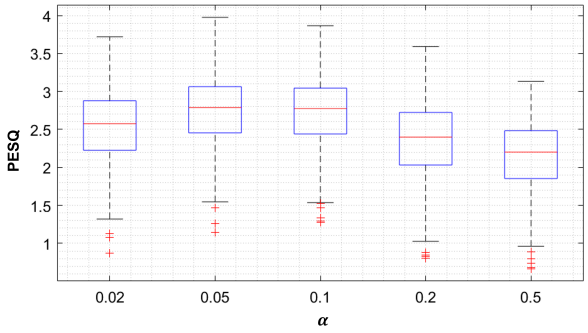


Figura 4.1: Estudio de ablación sobre las pérdidas del discriminador. Hiperparámetros estudiados para α en (3.16) basándose en la puntuación de PESQ.

Los resultados muestran que la inclusión del término de pérdidas del discriminador en el entrenamiento completo mejora el rendimiento de la sintetización del discurso. Sin embargo, utilizar una pendiente demasiado grande una vez se ha alcanzado cierto nivel de convergencia puede empeorar los resultados. Por tanto, se ha seleccionado $\alpha = 0{,}05$ debido a su buen comportamiento respecto al PESQ en la figura 4.1. El resto de parámetros utilizados para entrenar el modelo propuesto son: 130 épocas con un tamaño de *batch* de 128, tasa de aprendizaje $\mu = 5 \cdot 10^{-4}$ para el generador y $\mu = 5 \cdot 10^{-5}$ para el discriminador. Ambas tasas sufren un decremento exponencial desde la época 80 para optimizar el modelo lo máximo posible.

4.1.3. Resultados experimentales

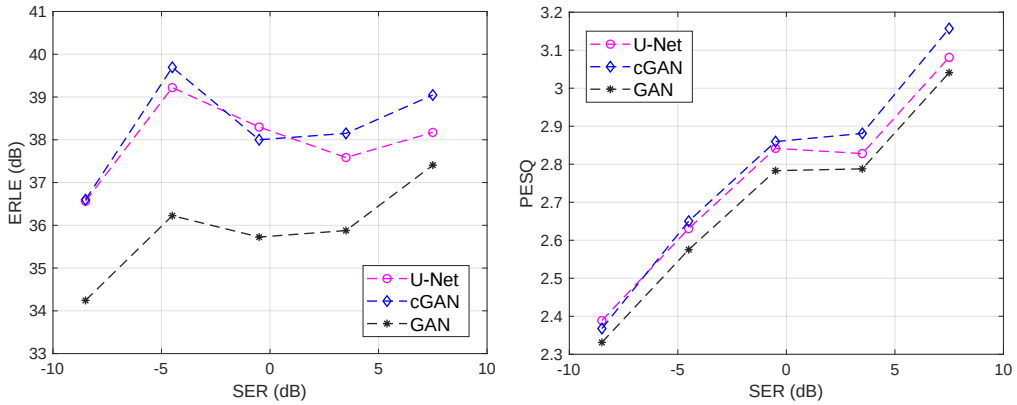
Tras entrenar el modelo con la base de datos mencionada y los parámetros optimizados, se ha evaluado la metodología en el conjunto de testeo por medio de las métricas PESQ y ERLE. De forma adicional, para evaluar la mejora en el rendimiento de la cGAN propuesta, se han comparado los resultados con dos modelos de referencia: una Residual U-Net y una GAN. La estructura principal es la misma para las tres redes, manteniendo el número de filtros y capas del generador.

En el caso de la U-Net residual, se omite el bloque del discriminador, por lo que únicamente está compuesto por el generador. Por otra parte, en la GAN, la entrada del discriminador es una imagen de un solo canal, por lo que el espectrograma del extremo lejano, X, no se utiliza como condición. Los valores de ERLE y PESQ de los tres modelos se muestra en la Tabla 4.1 para diferentes escenarios y para todo el rango de valores de relación señal a eco (SER) entre $-10\,\text{dB}$ y $9\,\text{dB}$. La Tabla 4.1 muestra que la cGAN ofrece el mejor rendimiento en ambas métricas para cada uno de los escenarios.

Tabla 4.1: Puntuaciones de PESQ y ERLE para distintos escenarios. All: todo el conjunto de test; FN: extremo lejano con ruido; FnN: extremo lejano sin ruido; NN:extremo cercano con ruido; NnN: exremo cercano sin ruido; N+FnL: Ambos ruidosos + extremo lejano no lineal.

Metrics	Models	Testing parameters					
		All	**FN**	**FnN**	**NN**	**NnN**	**N+FnL**
PESQ	GAN	2.68	2.66	2.7	2.6	2.75	2.58
	U-Net	2.72	2.71	2.74	2.65	2.79	2.63
	cGAN	**2.76**	**2.74**	**2.78**	**2.69**	**2.82**	**2.66**
ERLE	GAN	34.24	33.73	34.78	33.62	34.82	33.43
	U-Net	35.94	35.57	36.32	35.36	36.47	35.46
	cGAN	**36.31**	**36.11**	**36.51**	**35.85**	**36.73**	**36.28**

Para mostrar el buen funcionamiento de los modelos presentados en este TFG, es posible comparar sus valores de PESQ con aquellos reportados por [16, 15, 49], cuyas soluciones AEC obtuvieron la cuarta, tercera y segunda posición respectivamente en el "*Acoustic Echo Cancellation Challenge - ICASSP 2021*" [4]. Sin embargo, no ha sido posible comparar el modelo propuesto con el que obtuvo el primer puesto [13] porque no miden su rendimiento con una métrica implementable. En primer lugar, los autores en [15] utilizan 100 muestras elegidas de forma aleatoria del *synthetic dataset* de Microsoft como su conjunto de testeo, reportando un PESQ de 2,61, inferior al PESQ en la columna "ALL" de la Tabla 4.1 correspondiente a los modelos implementados en este trabajo. En segundo lugar, Westhausen et al. [16] utilizan un conjunto de testeo alternativo también proporcionado por Microsoft (llamado *test set* en [32]), el cual presenta características similares al *synthetic dataset*. Su puntuación de PESQ es de 2,53 para la voz con ruido del extremo cercano (comparable con los valores de PESQ de la columna "NN" de la Tabla 4.1), 2,73 para la voz ruidosa de extremo lejano (comparable a los valores de la columna "FN") y 2,43 cuando ambas señales son ruidosas(comparable a los valores de la columna "N+FnL"). En tercer lugar, en [49] los autores utilizan las primeras 500 grabaciones del *synthetic dataset* para evaluar su modelo, alcanzando una puntuación de PESQ de 2,07. Puede apreciarse que el PESQ obtenido por la cGAN presentada supera el rendimiento de las soluciones AEC previas en todas las condiciones de SER.



(a) **(b)**

Figura 4.2: Media de valores de ERLE, figura (a), y PESQ, figura (b), para diferentes intervalos de SER del conjunto de test.

Por último, se ha estudiado la influencia de la relación señal a eco en el rendimiento del modelo

respecto a los valores de PESQ y ERLE en conjuntos uniformes de acuerdo a su SER: bajo SER en el rango $[-10, -7]$ dB, medio-bajo en el rango $[-6, -3]$ dB, SER medio en el rango $[-2, 1]$ dB, medio-alto en el rango $[2, 5]$ dB y SER alto en el rango $[6, 9]$ dB. La figura4.2 muestra la media del ERLE (parte superior) y la media del PESQ (parte inferior) de las señales sintetizadas para cada conjunto, donde los valores de SER en la figura4.2 se toman como el punto medio del intervalo. Se puede apreciar que los valores de la cGAN tanto para PESQ como para ERLE son más elevados que en los otros modelos para la mayoría de intervalos de SER. Sin embargo, las ganancias de PESQ y ERLE obtenidas por la cGAN respecto a la U-Net son más elevadas para valores de SER intermedios y elevados, mientras que para valores bajos de SER, el discriminador no mejora la señal obtenida a la salida del generador. Respecto al comportamiento de la GAN, la U-Net y la cGAN superan de forma clara a la GAN en todos los intervalos de SER. Es destacable que el rendimiento de la GAN es peor que de la U-Net, lo que muestra la importancia de construir un discriminador condicional robusto.

4.2. Detección de sentimientos

En esta sección se describen los experimentos realizados y los resultados obtenidos en la tarea de detección de sentimientos.

4.2.1. Métricas de evaluación

Para evaluar la tarea de detección de sentimientos con las diferentes aproximaciones propuestas, se ha empleado la precisión y las pérdidas del conjunto de test como métricas de evaluación de rendimiento. Estas medidas permiten, si se calculan sobre el conjunto de test, determinar la capacidad de generalización del modelo con un subconjunto de datos que no ha sido utilizado para el entrenamiento. Además de entrenar las redes, utilizando el conjunto de datos de entrenamiento, se necesita otro subconjunto de datos, que no haya sido utilizado para el entrenamiento pero diferente al conjunto de test, que nos permita encontrar los hiperparámetros óptimos necesarios en los modelos (número de épocas, tamaño del *batch*, tasa de aprendizaje, etc.). Este conjunto adicional es el conjunto de validación. Cuando se trabaja con bases de datos pequeñas, como es el caso en la tarea de detección de sentimientos, una mala distribución de las muestras puede provocar que el rendimiento parezca peor o mejor de lo que realmente es, falseando la percepción de los modelos. Para evitar este problema, se ha llevado a cabo un procedimiento denominado validación cruzada (*K-fold cross validation*).

Este método consiste en dividir la base de datos en K particiones y entrenar K modelos distintos. El entrenamiento de cada uno de estos modelos hace uso de la base de datos al completo, pero la distribución de los datos es distinta para cada uno de ellos, tal y como se aprecia en la figura 4.3. Se realiza de forma que, una vez generadas las K divisiones, cada muestra se encuentra una sola vez en uno de los conjuntos de validación mientras que, para el resto de divisiones, se encuentran en el conjunto de entrenamiento. Con estos K modelos se constituye un comité de clasificadores que evaluarán a cada muestra de test, para la cual se decide como predicción la clase mayoritaria predicha por el comité. De esta forma, se independiza el resultado de la elección de una división de datos concreta, pues esta puede resultar especialmente favorable o desfavorable.

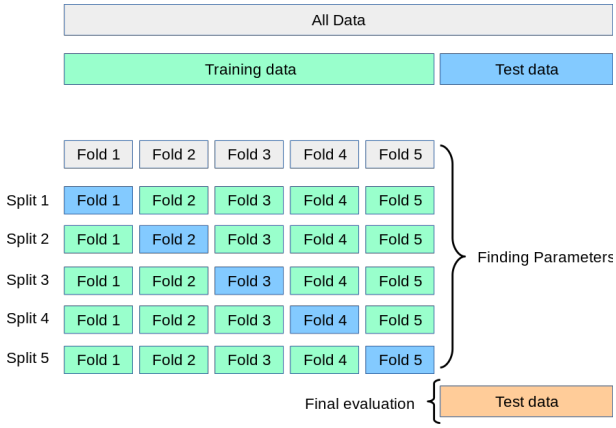


Figura 4.3: Validación cruzada K-Fold [50]

4.2.2. Experimentos de ablación

Para ajustar los hiperparámetros que optimicen el rendimiento del modelo, se han llevado a cabo diversos experimentos. Utilizando la opción de entrenamiento, se han testeado mediante validación cruzada distintos valores de la profundidad del ViT, $deep = 1, 2, 4, 8, 12$, de tasa de aprendizaje, $\tau = 1e-3, 1e-4, 1e-5$, número de épocas, $epochs = 10, 20, \ldots, 190, 200$ y tamaño de *batch*, $batchsize = 4, 8, 16, 32$. La profundidad se refiere a la cantidad de bloques (conformados por *multi-head attention* y *feed forward*) que se encuentran implementados en el *encoder* del ViT.

Los resultados muestran que, para las dos bases de datos, el rendimiento mejora conforme se van añadiendo más bloques. Por tanto, se ha mantenido un valor constante de $deep = 12$. En cuanto a los parámetros de entrenamiento, el tamaño del *batch* óptimo para todas las configuraciones resultó ser $batchsize = 16$. Sin embargo, el resto de parámetros varían en función de si el modelo optimizado utiliza como extractor de características la CNN ad hoc o la VGG16. Esto se debe a que unos valores mal seleccionados, provocan la no convergencia del modelo. En el caso de la CNN ad hoc, ha sido necesario utilizar una tasa de aprendizaje $\tau = 1e-4$, con un número de épocas elevado, del orden de 70 épocas. Con la VGG16, por contra, se debe usar una tasa de aprendizaje menor, del orden de $\tau = 1e-5$ y también necesitaba menos periodos de entrenamiento para alcanzar la convergencia, entorno a 40 épocas.

4.2.3. Resultados experimentales

Para evaluar el modelo VGG16+ViT de forma adecuada, se ha entrenado utilizando el método de validación cruzada K-Fold. Además, para comprobar que las distintas partes de la arquitectura incrementan el rendimiento, se han evaluado otros tres modelos del estado del arte como referencia. Se trata una CNN ad hoc (sección 3.4.2.2), una VGG16 entrenada mediante *fine-tuning* (seccion 3.4.2.2) y una VGG16+LSTM (sección 3.4.2.3). Con las dos primeras se determina qué extractor de características ofrece un mejor rendimiento y se establece una base de referencia. Con la VGG16+LSTM, se busca implementar una de las arquitecturas recurrentes más populares del estado del arte, pudiendo ver así la mejora que aporta respecto a la base de referencia y permitiendo la comparación de dicha arquitectura con el modelo propuesto bajo las mismas condiciones. Todas las configuraciones se han completado con un *top model*, que realiza la clasificación, consistente en un perceptrón multicapa (sección 3.4.2.3).

Este proceso se ha seguido para cada una de las bases de datos destinadas a detección de sentimientos comentadas en la sección 2.1. Es posible observar así la superioridad del modelo presentado para distintos escenarios.

Resultados en EmoDB

La base de datos EmoDB, dispone de pocas muestras por clase y además se encuentran desbalanceadas. Por ello, se ha considerado conveniente aplicar *data augmentation* sobre el conjunto de entrenamiento para aumentar el número de muestras. Concretamente, se han utilizado las técnicas de desplazamiento temporal y adición de ruido blanco gausiano en todas las clases, para después reducir el número de muestras de ciertas clases para que quede balanceado. De esta forma, es posible aportar más variedad a la hora de optimizar los parámetros mejorando así la capacidad de generalización del modelo.

A continuación, en la Tabla 4.2 se muestra el rendimiento de los distintos modelos a evaluar.

Arquitectura	Precisión en test (%)
CNN	69,16 %
VGG16	73,83 %
VGG16+LSTM	76,64 %
VGG16+ViT	**77,57 %**

Tabla 4.2: Resultados de los modelos evaluados sobre la base da datos EmoDB.

Se observa en la Tabla 4.2 que la red neuronal convolucional ad hoc presenta un buen comportamiento por sí sola. Sin embargo, congelando las dos primeras capas de la VGG16 y reentrenando el resto, se obtienen mejores resultados y de forma mucho más eficiente; el tiempo de optimización es considerablemente menor, del orden de un 50 % más corto. Por tanto, utilizando como extractor de características la VGG16, se ha tratado de mejorar el rendimiento añadiendo nuevos bloques. Por un lado, se ha colocado en serie con la VGG16 una LSTM capaz de retener información a largo plazo. Se establecen así relaciones temporales entre las características finales, lo que resulta en una mayor precisión en test. Por otro, se incorpora un *Vision Transformer* en paralelo a la VGG16, ya que tiene la capacidad de establecer relaciones tanto espaciales como temporales entre los distintos parches del espectrograma. De esta forma, se complementan la extracción de características de una CNN y la información obtenida por el *Vision Transformer*. Con esta disposición se obtiene una precisión final del $77,57\,\%$, superando al resto de modelos del estado del arte presentes en la Tabla 4.2.

Para ahondar en cómo se distribuye la precisión por clases para el modelo VGG16+ViT, se han extraído las matrices de confusión de la figura 4.4. De esta forma, es posible saber en qué clases el modelo ofrece un mejor rendimiento o cuales confunde.

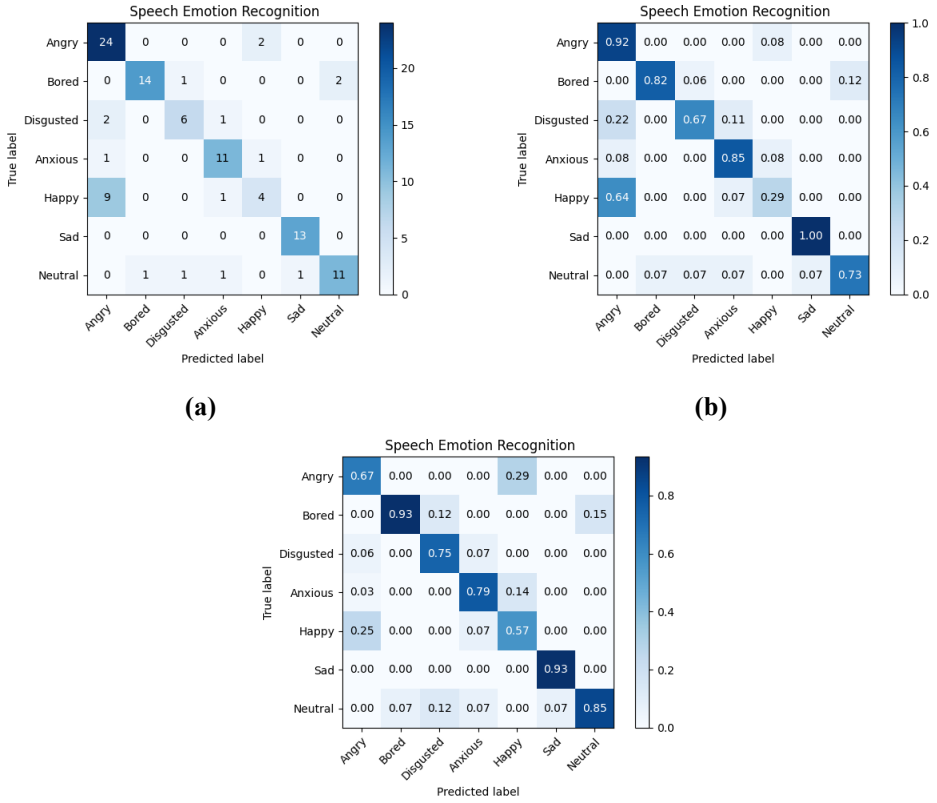


Figura 4.4: Matrices de confusión del ViT en EmoDB. (a) sin normalizar, (b) normalizado por filas, (c) normalizado por columnas.

Los resultados muestran que la mayoría de muestras que confunde se debe al *arousal*, es decir, a la excitación o nivel de intensidad que proyectamos cuando expresamos nuestros sentimientos. De hecho, lo que más confunde es enfado con alegría pues en ambos casos el nivel de excitación es elevado. La clase neutro la confunde con todas las emociones que muestran un nivel de intensidad bajo, como son la tristeza, ansiedad, disgusto y aburrimiento. Sin embargo, la clase neutro es de las más complicadas de predecir según se observa en el estado del arte, corroborando los resultados del modelo propuesto. También ocurre que confunde parte de la clase disgustado no sólo por su intensidad, sino también por el tipo de sentimiento, si es positivo o negativo. Esto se observa claramente en las matrices de confusión, pues el modelo predice parte de la clase disgustado como enfadado y parte como ansioso.

Resultados en RAVDESS

A continuación, se ha evaluado la base de datos RAVDESS. En cuanto a la distribución de las clases, se encuentra mejor preparada que EmoDB, todas tienen aproximadamente el mismo número de muestras por lo que no hay ninguna especialmente desbalanceada. No ha sido entonces necesario aplicar técnicas de *data augmentation* durante el entrenamiento. En la Tabla 4.3 se observa el rendimiento de los modelos comparados.

Arquitectura	Precisión en test (%)
CNN	68,78 %
VGG16	70,41 %
VGG16+LSTM	72,04 %
VGG16+ViT	**72,65 %**

Tabla 4.3: Resultados de los modelos evaluados sobre la base de datos RAVDESS.

En este apartado, se ha querido corroborar la tendencia mostrada por los modelos evaluados en la base de datos anterior. De nuevo, es preferible utilizar la VGG16 frente a la CNN ad hoc por su rendimiento y eficiencia de optimización. Congelándose las dos primeras capas de la VGG16, se obtienen mejores resultados y en menos tiempo. A continuación, se ha comprobado que, en efecto, la incorporación de información temporal mediante una LSTM en serie con el extractor de características mejora el rendimiento del modelo. Las dependencias temporales que establece sí que son relevantes para la clasificación de las muestras. Por último, se ha probado que el *Vision Transformer* en paralelo con la VGG16 también ofrece un mejor rendimiento que los otros modelos. Como muestra la Tabla 4.3, la arquitectura VGG16+ViT alcanza una precisión en test del $72,65\,\%$, superior a los otros métodos evaluados.

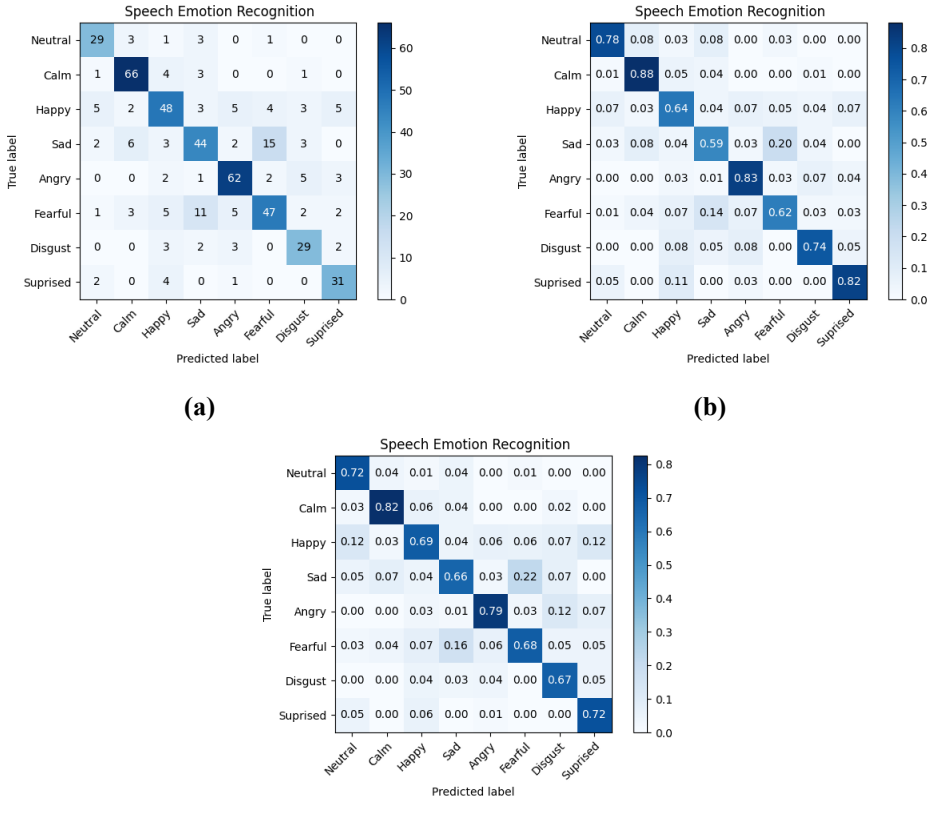


Figura 4.5: Matrices de confusión de la VGG16+ViT en RAVDESS. (a) sin normalizar, (b) normalizado por filas, (c) normalizado por columnas.

Para profundizar en el rendimiento real del modelo, se han obtenido las matrices de confusión de la figura 4.5 para esta base de datos.

En este caso, la matriz de confusión se encuentra más balanceada. Las clases se predicen con un rendimiento similar, aunque sí que se aprecian tendencias como la mostrada en EmoDB. Concretamente, las clases miedo y triste a veces las confunde; ambas expresan un sentimiento negativo y, en la mayoría de casos, con un nivel de intensidad bajo. El modelo también tiene ciertos problemas para diferenciar la clase feliz; predice en su lugar otros sentimientos por su nivel de intensidad, como pueden ser enfadado, otros los confunde por el grado de sentimiento, como ocurre con neutral y otros por ambos motivos, como la clase sorprendido.

Capítulo 5

Conclusiones y propuesta de futuro

En este trabajo se presentan dos arquitecturas principales; una para realizar cancelación de eco y otra para detección de emociones. Por un lado, se ha desarrollado una *Generative Adversarial Network* condicional (cGAN) que limpia la señal objetivo eliminando el eco. Por otro, se ha implementado un modelo híbrido de VGG16+ViT que realiza la identificación de emociones.

En primer lugar, se ha desarrollado el sistema AEC basado en la cGAN. Para el entrenamiento y testeo del modelo, se ha empleado una base de datos sintética proporcionado por Microsoft. El rendimiento de esta arquitectura ha sido evaluado para escenarios tanto *single-talk* como *double-talk* mediante las métricas ERLE y PESQ, respectivamente. También se han comparado los resultados de la propuesta con otros dos métodos, una U-Net y una GAN, y con otros trabajos del estado del arte, mostrando los beneficios del modelo presentado. Debido a los buenos resultados que ofrece la arquitectura, este trabajo [51] ha sido aceptado en la conferencia internacional 30^{th} *European Signal Processing Conference* (EUSIPCO 2022) que tendrá lugar en Belgrado del 29 de agosto al 2 de septiembre de 2022.

En cuanto a las mejoras que se podrían incluir en esta tarea, una podría ser la creación de bases de datos más realistas y variadas. Si bien es verdad que la proporcionada por Microsoft es bastante buena, añadir más variabilidad en cuanto a las no linealidades, simular un entorno más realista e incrementar el número de muestras mejoraría sin duda el rendimiento y la capacidad de generalización del modelo. En cuanto a la arquitectura en sí, aplicar algoritmos recurrentes en el espacio latente es bastante habitual en el estado del arte, por lo que podría ser una buena opción. En este caso no se ha implementado porque la latencia se incrementaba demasiado. Por último, reducir el tiempo de inferencia es clave para aplicaciones en tiempo real. Reducir el número de parámetros de la arquitectura actual combinándolo con operaciones en el espacio latente mantendría el rendimiento reduciendo la latencia.

Posteriormente, se ha diseñado el sistema de detección de emociones basado en la red híbrida VGG16+ViT. Para el entrenamiento y testeo del modelo, se han empleado dos bases de datos públicas etiquetadas, *EmoDB* y *RAVDESS*. El rendimiento de la arquitectura se ha evaluado mediante la precisión en test tras optimizar el modelo utilizando técnicas de validación cruzada k-fold. Se han comparado los resultados de la propuesta con otros tres métodos, una CNN ad hoc, una VGG16 y una VGG16+LSTM y se ha mostrado la superioridad del modelo. Esto demuestra parte del potencial que tienen los *Transformers* y *Vision Transformers* para procesado de audio, ya que la información que son capaces de aportar es relevante aún utilizando pocos datos de entrenamiento.

Y es que si algo caracteriza a los Transformers, es que muestran su verdadero potencial cuando se utilizan grandes bases de datos.

Para mejorar esta propuesta, una de las opciones también hace referencia a las bases de datos. A pesar de que existe cierta variedad, generar nuevas bases de datos en otros idiomas o en ambientes diferentes podría enriquecer y potenciar el desarrollo de modelos más completos, así como permitir su utilización en más ámbitos. Esto se debe a que, en cada cultura y entorno, nos comportamos y expresamos emociones de forma diferente. Por ejemplo, no existe una base de datos etiquetada en español por lo que, disponer de una, abriría las puertas a implementar un sistema de detección de emociones en aplicaciones de psicología en nuestro idioma, como es uno de los objetivos del grupo de investigación CVBLab. Otra opción muy interesante y que se utiliza en el estado del arte es la clasificación multimodal. Combinar varios tipos de datos como pueden ser la voz, el contenido de la frase y los gestos siempre aporta más información y mejora los resultados. En este caso el tiempo de inferencia es bastante más rápido y además no necesariamente es tan crítico como en la cancelación de eco.

Por último, debe hacerse mención a la revolución que está teniendo lugar en el campo de la inteligencia artificial. Y es que, hasta la fecha, los algoritmos y modelos que se presentaban estaban destinados y preparados para realizar una función concreta. Sin embargo, con la llegada de los *Transfomers*, esas redes capaces de establecer relaciones dentro de cualquier tipo de secuencias, los modelos están empezando a ser capaces de desarrollar varias funciones cuando estas pertenecen al mismo ámbito. Por ejemplo, para el caso del procesado de audio, es posible entrenar el *encoder* del *Transformer* de forma que sea capaz de interpretar un espectrograma. Con esta capacidad, realizar tareas como detección de palabras clave, detección de emociones o clasificación de eventos en voz se pueden realizar con el mismo núcleo de extracción de características. Sin embargo, es necesario una enorme cantidad de datos para alcanzar un buen rendimiento, por lo que es necesario recurrir a técnicas de aprendizaje autosupervisado. Y esto es lo que proponen en [52] los autores del trabajo. Todavía es posible mejorar y optimizar el método que emplean por lo que la propuesta de futuro ideal tanto para detección de emociones como para cancelación de eco sería la implementación de un *Vision Transformer* generalista capaz de realizar ambas funciones de manera fiable.

Bibliografía

[1] M. M. Sondhi. “An adaptive echo canceller”. En: *The Bell System Technical Journal* 46.3 (1967), págs. 497-511.

[2] S. Gustafsson, R. Martin y P. Vary. “Combined acoustic echo control and noise reduction for hands-free telephony”. En: *Signal Proc.* 64.1 (1998), págs. 21-32.

[3] J. Sohn, N.S. Kim y W. Sung. “A statistical model-based voice activity detection”. En: *IEEE Signal Proc. Letters* 6.1 (1999), págs. 1-3.

[4] Microsoft Research. *Acoustic Echo Cancellation Challenge - ICASSP 2021 [Online]*. Available at `https://www.microsoft.com/en-us/research/academic-program/acoustic-echo-cancellation-challenge-icassp-2021/`. accessed on March 4th, 2022 de 2022.

[5] B. Naderi y R. Cutler. “An Open source Implementation of ITU-T Recommendation P.808 with Validation”. En: *arXiv preprint arXiv:2005.08138* (2020). arXiv: `2005.08138`. URL: `http://arxiv.org/abs/2005.08138`.

[6] Fuji Ren y Changqin Quan. “Linguistic-based emotion analysis and recognition for measuring consumer satisfaction: an application of affective computing”. En: *Information Technology and Management* 13.4 (2012), págs. 321-332.

[7] Georgios N Yannakakis. “Enhancing health care via affective computing”. En: (2018).

[8] Javier Hernandez, Rob R Morris y Rosalind W Picard. “Call center stress recognition with person-specific models”. En: *International Conference on Affective Computing and Intelligent Interaction*. Springer. 2011, págs. 125-134.

[9] EV Polyakov y col. “Investigation and development of the intelligent voice assistant for the Internet of Things using machine learning”. En: *2018 Moscow Workshop on Electronic and Networking Technologies (MWENT)*. IEEE. 2018, págs. 1-5.

[10] Roger K Moore. “Is spoken language all-or-nothing? Implications for future speech-based human-machine interaction”. En: *Dialogues with Social Robots*. Springer, 2017, págs. 281-291.

[11] Samaneh Madanian y col. “Automatic Speech Emotion Recognition Using Machine Learning: Digital Transformation of Mental Health”. En: (2022).

[12] M Maithri y col. “Automated Emotion Recognition: Current Trends and Future Perspectives”. En: *Computer Methods and Programs in Biomedicine* (2022), pág. 106646.

[13] J.M. Valin y col. “Low-Complexity, Real-Time Joint Neural Echo Control and Speech Enhancement Based On Percepnet”. En: *Proc. of the 2021 IEEE Int. Conf. on Acoustics, Speech and Signal Proc. (ICASSP)*. 2021, págs. 7133-7137.

[14] Z. Wang y col. “Weighted Recursive Least Square Filter and Neural Network Based Residual ECHO Suppression for the AEC-Challenge”. En: *Proc. of the 2021 IEEE Int. Conf. on Acoustics, Speech and Signal Proc. (ICASSP)*. 2021, págs. 141-145.

[15] R. Peng y col. “ICASSP 2021 Acoustic Echo Cancellation Challenge: Integrated Adaptive Echo Cancellation with Time Alignment and Deep Learning-Based Residual Echo Plus Noise Suppression”. En: *Proc. of the 2021 IEEE Int. Conf. on Acoustics, Speech and Signal Proc. (ICASSP)*. 2021, págs. 146-150.

[16] N.L. Westhausen y B.T. Meyer. “Acoustic Echo Cancellation with the Dual-Signal Transformation LSTM Network”. En: *Proc. of the 2021 IEEE Int. Conf. on Acoustics, Speech and Signal Proc. (ICASSP)*. 2021, págs. 7138-7142.

[17] H. Zhang y D. Wang. “A Deep Learning Approach to Multi-Channel and Multi-Microphone Acoustic Echo Cancellation”. En: *Proc. of Interspeech 2021*. 2021, págs. 1139-1143.

[18] L. Ma y col. “Multi-Scale Attention Neural Network for Acoustic Echo Cancellation”. En: *arXiv:2106.00010* (2021).

[19] E. Kim, J.-J. Jeon y H. Seo. “U-Convolution Based Residual Echo Suppression with Multiple Encoders”. En: *ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Proc. (ICASSP)*. IEEE. 2021, págs. 925-929.

[20] K.N. Watcharasupat y col. “End-to-End Complex-Valued Multidilated Convolutional Neural Network for Joint Acoustic Echo Cancellation and Noise Suppression”. En: *arXiv preprint arXiv:2110.00745* (2021).

[21] Y. Zhang y col. “Generative Adversarial Network Based Acoustic Echo Cancellation.” En: *INTERSPEECH*. 2020, págs. 3945-3949.

[22] S. Routray y Q. Mao. “Phase sensitive masking-based single channel speech enhancement using conditional generative adversarial network”. En: *Computer Speech & Language* 71 (2022), pág. 101270.

[23] R. Liu y col. “SCCGAN: style and characters inpainting based on CGAN”. En: *Mobile Networks and Applications* 26.1 (2021), págs. 3-12.

[24] X. Cao. “Image-to-Image Translation with Application to Biometrics”. Tesis de mtría. Schulich School of Engineering, 2022.

[25] Srinivas Parthasarathy y Carlos Busso. “Semi-supervised speech emotion recognition with ladder networks”. En: *IEEE/ACM transactions on audio, speech, and language processing* 28 (2020), págs. 2697-2709.

[26] Soonil Kwon y col. “MLT-DNet: Speech emotion recognition using 1D dilated CNN based on multi-learning trick approach”. En: *Expert Systems with Applications* 167 (2021), pág. 114177.

[27] Ziping Zhao y col. “Combining a parallel 2d cnn with a self-attention dilated residual network for ctc-based discrete speech emotion recognition”. En: *Neural Networks* 141 (2021), págs. 52-60.

[28] Jianfeng Zhao, Xia Mao y Lijiang Chen. “Speech emotion recognition using deep 1D & 2D CNN LSTM networks”. En: *Biomedical signal processing and control* 47 (2019), págs. 312-323.

[29] Mehmet Bilal Er. “A novel approach for classification of speech emotions based on deep and acoustic features”. En: *IEEE Access* 8 (2020), págs. 221640-221653.

[30] J Ancilin y A Milton. “Improved speech emotion recognition with Mel frequency magnitude coefficient”. En: *Applied Acoustics* 179 (2021), pág. 108046.

[31] Turker Tuncer, Sengul Dogan y U Rajendra Acharya. “Automated accurate speech emotion recognition system using twine shuffle pattern and iterative neighborhood component analysis techniques”. En: *Knowledge-Based Systems* 211 (2021), pág. 106547.

[32] K. Sridhar y col. “ICASSP 2021 Acoustic Echo Cancellation Challenge: Datasets, Testing Framework, and Results”. En: *ICASSP 2021 - 2021 IEEE International Conference on Acoustics, Speech and Signal Proc. (ICASSP)*. 2021, págs. 151-155. DOI: 10.1109/ICASSP39728.2021.9413457.

[33] Paul Ed Ekman y Richard J Davidson. *The nature of emotion: Fundamental questions*. Oxford University Press, 1994.

[34] Yosephine Susanto y col. “The hourglass model revisited”. En: *IEEE Intelligent Systems* 35.5 (2020), págs. 96-102.

[35] Gyanendra K Verma y Uma Shanker Tiwary. “Affect representation and recognition in 3D continuous valence–arousal–dominance space”. En: *Multimedia Tools and Applications* 76.2 (2017), págs. 2159-2183.

[36] Ivet Challenger-Pérez, Yanet Díaz-Ricardo y Roberto Antonio Becerra-García. “El lenguaje de programación Python”. En: *Ciencias Holgúin* 20.2 (2014), págs. 1-13.

[37] Boualem Boashash. *Time-frequency signal analysis and processing: a comprehensive reference*. Academic press, 2015.

[38] Lawrence Rabiner y Ronald Schafer. *Theory and applications of digital speech processing*. Prentice Hall Press, 2010.

[39] Frank Rosenblatt. *The perceptron, a perceiving and recognizing automaton Project Para*. Cornell Aeronautical Laboratory, 1957.

[40] Sumit Saha. *https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53*. Mayo de 2022.

[41] ArcGIS Developers. *https://developers.arcgis.com/python/guide/how-unet-works/*. Mayo de 2022.

[42] Cristopher Olah. *https://colah.github.io/posts/2015-08-Understanding-LSTMs/*. Mayo de 2022.

[43] Ashish Vaswani y col. “Attention is all you need”. En: *Advances in neural information processing systems* 30 (2017).

[44] Eduardo Muñoz. *https://towardsdatascience.com/attention-is-all-you-need-discovering-the-transformer-paper-73e5ff5e0634*. Mayo de 2022.

[45] Alexey Dosovitskiy y col. “An image is worth 16x16 words: Transformers for image recognition at scale”. En: *arXiv preprint arXiv:2010.11929* (2020).

[46] O. Ronneberger, P. Fischer y T. Brox. “U-Net: Convolutional Networks for Biomedical Image Segmentation”. En: *Medical Image Computing and Computer-Assisted Intervention – MICCAI 2015*. 2015, págs. 234-241.

[47] Yunyang Xiong y col. “Nyströmformer: A Nyström-based Algorithm for Approximating Self-Attention”. En: (2021).

[48] ITU-T. “Perceptual evaluation of speech quality (PESQ): An objective method for end-to-end speech quality assessment of narrow-band telephone networks and speech codecs”. En: *ITU-T Recommendation P.862* (2001).

[49] Ziteng Wang y col. “Weighted recursive least square filter and neural network based residual echo suppression for the aec-challenge”. En: *ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Proc. (ICASSP)*. IEEE. 2021, págs. 141-145.

[50] Scikit learn. *https://scikit-learn.org/stable/modules/cross$_v alidation.html$*. Mayo de 2022.

[51] Julio Silva-Rodríguez Miguel Ferrer Gema Piñero Fran Pastor-Naranjo Rocío del Amor y Valery Naranjo. “Conditional generative adversarial networks for acoustic echo cancellation”. En: *Proc. of the 30th European Signal Processing Conference (EUSIPCO 2022). Accepted for publication.* 2022. URL: `https://www2.securecms.com/EUSIPCO2022/Papers/ViewSession.asp?Sessionid=1067`.

[52] Yuan Gong y col. “Ssast: Self-supervised audio spectrogram transformer”. En: *arXiv preprint arXiv:2110.09784* 4 (2021).

Parte II

Presupuesto

Capítulo 1

Presupuesto

1.1. Objetivo

El objetivo de este apartado es realizar una valoración de los costes necesarios para desarrollar e implementar las técnicas de aprendizaje profundo para procesado de audio presentadas en este proyecto.

1.2. Presupuestos parciales

Para poder analizar y cuantificar mejor el presupuesto del proyecto, se ha considerado oportuno dividirlo en tres presupuestos parciales: costes de personal 1.2.1, costes de *hardware* 1.2.2 y costes de *software* 1.2.3.

1.2.1. Costes de personal

Estos costes hacen referencia a los recursos humanos que han sido necesarios para la realización del proyecto. Se trata del coste de la mano de obra, por lo que se debe tener en cuenta tanto las horas empleadas por cada persona involucrada como la remuneración que debería recibir cada una de ellas. En la tabla 1.1 se encuentra desglosado el coste por cada uno y su valor total. Los participantes de este TFG son:

- Dª María Gema Piñero Sipán, catedrática de universidad y tutora de este trabajo.
- Dª María Rocío del Amor del Amor, ingeniera biomédica y cotutora de este trabajo.
- D. Francisco Pastor Naranjo, estudiante del grado en Ingeniería de Tecnologías y Servicios de Telecomunicaciones y autor del proyecto.

Descripción	Uds.	Cantidad	Precio unitario (€/h)	Coste imputable
Catedrática	h	33	35	1.155,00
Graduada	h	40	25	1.000,00
Estudiante	h	300	13.5	4.050,00
		TOTAL		6.205,00

Tabla 1.1: Desglose de los costes de personal

1.2.2. Costes de hardware

En cuanto a los costes de *hardware*, estos se refieren al valor de la amortización de cada uno de los dispositivos empleados durante la consecución del proyecto. Esto significa que, en función del valor de cada producto, de su tiempo de uso y del tiempo que ha sido utilizado para este trabajo, supone un coste a considerar.

Para la depuración del código así como para administrar toda la información del proyecto y elaborar la memoria se ha utilizado un ordenador personal. Concretamente se trata del modelo *Dell XPS 15 9550*. Para entrenar y evaluar el modelo, el ordenador personal no dispone de la suficiente memoria RAM ni capacidad de procesamiento, por lo que ha sido necesario recurrir a los servidores de computación de que dispone el grupo CVBLab. Por último, para almacenar los audios utilizados para la optimización de los parámetros, se ha hecho uso de un servidor de datos también perteneciente a CVBLab. De esta forma, los costes de *hardware* se encuentran desglosados en la tabla 1.2. Comentar que, al tratarse de productos electrónicos, en España disponen de un 21 % de IVA.

Descripción	Cantidad	Coste por ud. sin IVA (€)	Periodo de amortización (meses)	Intervalo amortizado (meses)	Coste imputable sin IVA (€)
DELL XPS 9550 Intel(R) Core(TM) i7-6700HQ, @2.6GHz, 16GB RAM, NVIDIA GeForce GTX 960M	1	1405,86	72	5	97,63
Procesador Intel i7 @4.20GHz	1	344,00	48	3	21,50
Tarjeta gráfica NVIDIA Titan V	1	3300,00	48	3	206,25
NAS Synology DS918	1	77,00	48	3	4,81
		TOTAL			330,20

Tabla 1.2: Coste de *hardware*

1.2.3. Costes de software

A continuación, se detallan los costes adicionales ocasionados por las licencias de programas necesarios en el proyecto. Para la elaboración de este documento se ha empleado la versión gratuita de *overleaf*. Para representar espectrogramas y gestionar las bases de datos se ha empleado MATLAB

R2020b, cuyo coste anual es de 840€ con un 21 % de IVA por ser un servicio digital. Por último, para la implementación del modelo se ha usado Python, un *software libre*, y diversas librerías de código abierto como son keras, matplotlib o pytorch. Los costes de *software* se encuentran desglosados en la tabla 1.3.

Descripción	Cantidad	Coste sin IVA (€)	Periodo de amortización (meses)	Intervalo de uso (meses)	Coste imputable sin IVA (€)
Overleaf	1	0	12	8	0
MATLAB R2020b	1	840,00	12	8	694,21
Python	1	0	12	3	0
Librerías python	1	0	12	3	0
		TOTAL			694,21

Tabla 1.3: Costes de *software*

1.3. Presupuesto Total

Por último, tomando todos los presupuestos parciales, se tiene que para la realización de este proyecto se requiere de un presupuesto total de $7.229,41$€