Universidad Internacional de La Rioja (UNIR)

ESIT

Máster Universitario en Inteligencia Artificial

Aumento de audio y compresión de modelos para clasificación de escenas acústicas

Trabajo Fin de Máster

Presentado por: González Poy, Eduard Marcel

Director/a: Fernández Isabel, Alberto

Ciudad: Tortosa
Fecha: 14 de setiembre de 2022

Resumen

Este proyecto consiste en el desarrollo de un modelo de baja complejidad para clasificación de escenas acústicas; basado en el reto de la tarea 1 de DCASE 2022. Para conseguir la baja complejidad, se propone la destilación de conocimiento con una aproximación “maestro-alumno” y posterior cuantificación de los parámetros de la red al tipo entero de 8 bits. La red maestro se basa en redes convolucionales residuales y, la red alumno, es una red convolucional lineal. Se propone una metodología de aumento de datos de audio basada en la convolución entre la señal original del conjunto de datos propuesto y respuestas impulsiones de dispositivos no presentes en la captura de esos datos. El modelo propuesto supera al de referencia obteniendo un *log loss* de 1,415, dentro de la limitación de complejidad establecida.

Palabras Clave: Aumento de audio, clasificación de escenas acústicas, cuantificación de redes neuronales, destilación de conocimiento, redes convolucionales residuales.

Abstract

In this project we have developed a low complexity model for acoustic scene classification; based on the challenge of task 1 of DCASE 2022, where a benchmark model to overcome is proposed. To achieve low complexity, we propose knowledge distillation with a "master-student" approach and subsequent quantization of the network parameters to the 8-bit integer type. The master network is based on residual convolutional networks and the student is a linear convolutional network. We present an audio data augmentation methodology, based on convolution between the original signal (of the proposed data set) and impulse responses from devices not present at the recording of the data. Our model outperforms the reference model by obtaining a log loss of 1,415, within the set complexity constraint.

Keywords: Audio augmentation, acoustic scene clasification, neural network quantization, knowledge distillation, residual convolutional networks

Índice de contenidos

Índice de tablas

Índice de figuras

1 Introducción

El trabajo final de máster (TFM) es la culminación de los estudios realizados. Por lo tanto, se ponen en práctica los conocimientos adquiridos a través de la investigación y el desarrollo del proyecto.

Este TFM, enmarcado en el área del *Deep Learning*, consiste en la implementación de una metodología que permita la ejecución de algoritmos de aprendizaje para la clasificación de escenas acústicas (*Acoustic Scene Clasification,* ASC) en dispositivos de baja complejidad (i. e. microcontroladores). En consecuencia, existe un componente de investigación sobre las técnicas de procesamiento de la señal de audio y del aprendizaje automático. Así como, la optimización de estos mecanismos para el despliegue en sistemas con características computacionales reducidas.

1.1 Motivación y justificación

El objetivo principal de la Inteligencia Artificial (IA), entendida como disciplina científica, es investigar y desarrollar tecnologías que permitan a las máquinas entender el entorno e interactuar con este de forma comparable a como lo hacen los humanos. Los animales (en particular los humanos) tenemos la capacidad de crear modelos del mundo gracias a los procesos cognitivos; recabamos información a través de los sistemas sensoriales (auditivo, visual, olfativo, somatosensorial, …) y la integramos para crear un contexto del entorno. Este contexto es crucial para la interacción de manera adecuada y, en última instancia, para la adaptación y supervivencia.

Lograr que los dispositivos tengan la capacidad de contextualizar el entorno y, en consecuencia, mejorar la interacción humano-máquina (*Human Machine Interaction*, HMI), es un reto de la IA. De ello, se derivan distintos campos de estudio basados en la modalidad sensorial (y el proceso cognitivo subyacente), como la visión por computador o el *machine listening* (capacidad de las máquinas para percibir y comprender el contenido del audio). En este segundo campo, es donde se enmarca este proyecto. Concretamente, en la línea de investigación que tiene como objetivo caracterizar el entorno acústico proveniente de un audio, proporcionándole una etiqueta semántica (Aucouturier et al., 2007); es decir: la clasificación de escenas acústicas o ASC.

Existen numerosos casos de uso de la ASC, como los dispositivos *wearables* y *hearables* conscientes del contexto, los audífonos, la atención médica remota, sistemas de vigilancia y

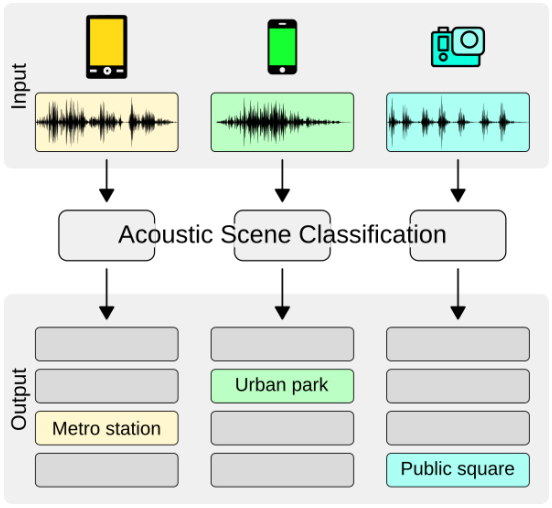

*Figura 1 : Descripción general del sistema de clasificación de escenas acústicas. Fuente: (*Low-Complexity Acoustic Scene Classification - DCASE*, 2022)*

seguridad, la monitorización de la vida salvaje en el hábitat natural, las ciudades inteligentes, el IoT y la navegación autónoma (Abeßer, 2020).

La mayor parte de estas aplicaciones se basan en el paradigma actual de la *Edge AI*; la inteligencia artificial desplegada en dispositivos cercanos al usuario, muchas veces sin necesidad de conexión a internet y con la capacidad de ejecutar los algoritmos de aprendizaje y procesamiento sin recurrir a servidores externos de cómputo. Además, gran parte de los dispositivos son portables (móviles, cámaras, relojes inteligentes, …). Esto implica que muchos de los dispositivos, ideales para la aplicación de la ASC, no tienen acceso a la potencia proporcionada por la red eléctrica y, por ello, están diseñados con componentes para la realización de tareas específicas (microcontroladores); que requieren menos consumo de potencia en detrimento de la reducción en la capacidad computacional.

En consecuencia, es necesario el desarrollo de algoritmos para la ASC capaces de ser ejecutados en dispositivos de baja complejidad.

1.2 Planteamiento del trabajo

El trabajo se plantea siguiendo la línea proporcionada por la tarea 1 de la octava edición (2022) del evento DCASE (*Low-Complexity Acoustic Scene Classification - DCASE*, 2022). Esta tarea, consiste en la creación de modelos para la clasificación de escenas acústicas (Figura 1). Además, añade el reto de implementar la solución de forma optimizada en base a los requerimientos de los sistemas en dispositivos de baja complejidad.

En la tarea se describe una metodología basada en el estado del arte, conforme a anteriores realizaciones del evento. Se propone un sistema de referencia basado en una red neuronal convolucional (CNN) y caracterización de la señal de audio basada en la energía de las bandas frecuenciales *Log-mel* (*Marmoi/Dcase2022_task1_baseline: Baseline System for DCASE 2022 Task 1*, 2022). La solución, implementada en Python, es un punto de partida para el desarrollo de sistemas que permitan mejorar el rendimiento; tanto a nivel de precisión de la clasificación, como a nivel de optimización de la red neuronal.

Se proporciona un conjunto de datos etiquetado que contiene audios de distintos escenarios acústicos de diferentes ciudades de Europa (se describe con detalle en los siguientes capítulos). Además, se ha predefinido el conjunto de entrenamiento y de prueba para validar la solución respecto al modelo de referencia.

Para validar la compresión de la red neuronal, y comprobar que puede ejecutarse en dispositivos de baja complejidad, se ofrece un script para calcular el tamaño, en cuanto a número de parámetros y MMACs (*million multiply-accumulate operations*), para modelos basados en Keras y PyTorch (*AlbertoAncilotto/NeSsi: Keras/Pytorch Neural Network Size, Operations and Parameters Counter*, 2022).

1.3 Estructura de la memoria

En el siguiente capítulo (Contexto y estado del arte), se repasa el estudio de la ASC. Se resume la historia y las distintas aproximaciones para solucionar el problema. Se hace especial énfasis en el estado del arte, basando en *Deep learning*. Se enumeran distintas soluciones para cada uno de los procesos subyacentes en la ASC: preparación y preprocesamiento de los datos, redes neuronales y optimización de los algoritmos para su ejecución en microcontroladores.

El capítulo 3 Objetivos y metodología de trabajo enuncia los distintos objetivos a alcanzar y la forma de alcanzarlos.

Los siguientes capítulos (4, 5 y 6) describen de forma exhaustiva el planteamiento de la comparativa, el desarrollo del sistema y la experimentación. También, se describe el conjunto de datos y su análisis y, se explica el algoritmo utilizado como referencia.

Finalmente, en el capítulo 7, se reportan las conclusiones obtenidas en base a los objetivos y resultados y las líneas de trabajo futuro.

2 Contexto y estado del arte

2.1 Antecedentes de la ASC

La clasificación de escenas acústicas (ASC) es una tarea técnica bien definida que pertenece al campo del análisis de escenas acústicas asistido por ordenador (CASA, *Computational Auditory Scene Analysis*) y cuyo objetivo es caracterizar el entorno acústico de un flujo sonoro dándole una etiqueta semántica (Rouat, 2008).

El análisis acústico de escenas asistido por ordenador es un campo interdisciplinar que se deriva del campo del análisis acústico de escenas (ASA, *Auditory Scene Analysis*). ASA describe e investiga la organización perceptiva de la audición en el contexto de sonidos simultáneos y/o paralelos. Un rasgo importante del sistema acústico es su capacidad para discriminar mezclas entre sonidos y agrupar características acústicas de un mismo evento o fuente sonora. Bregman presenta los principios del ASA y su visión de la percepción acústica de las mezclas de sonidos (Bregman & McAdams, 1994). Utilizando analogías con el análisis visual de escenas, aporta un marco completo para el análisis acústico de escenas. Los sistemas CASA intentan aplicar y/o explorar algunos de los principios de ASA mediante el uso de motores de desarrollo basados en estos modelos computacionales.

Fruto del avance en este campo de investigación y, en paralelo con la mejora tecnológica en los sistemas informáticos, surgen numerosos estudios de los sistemas ASC; aplicando distintas metodologías. Con el fin de homogeneizar este conocimiento y compartir los resultados, nace el evento DCASE (*Detection and classification of acoustic scenes and events*). En la primera edición se propusieron tres tareas: la clasificación de escenas acústicas o paisajes sonoros, la identificación de eventos sonoros únicos sin sonidos superpuestos y la identificación de eventos sonoros en situaciones con muchos sonidos (Giannoulis et al., 2013).

Las primeras soluciones propuestas se basaron en la parametrización de la señal de audio: preprocesamiento, extracción y descripción estadística de características acústicas como MFCC (*Mel Frecuency Cepstrum Coefficients*) y, la aplicación de técnicas de aprendizaje automático como GMM (*Gaussian Mixture Model*), HMM (*Hidden Markov Model*) o SVM (*Support Vector Machine*). En el estudio de Barchiesi (Barchiesi et al., 2014), se compara la capacidad humana para clasificar escenas acústicas frente a estos métodos. Concluyendo que el algoritmo con mejores resultados obtiene una precisión media que coincide con la precisión media obtenida por los humanos, y los pares de clases comunes son clasificados

erróneamente tanto por los ordenadores como por los humanos. Sin embargo, todas las escenas acústicas son clasificadas correctamente por al menos algunos individuos, mientras que hay escenas que son clasificadas erróneamente por todos los algoritmos (Barchiesi et al., 2014). En un artículo similar (Mesaros et al., 2017), que compara los métodos de las ediciones posteriores al 2013 del evento DCASE, se probó que el rendimiento humano, evaluado a través de un experimento auditivo, era significativamente más bajo que el rendimiento de la máquina. Los sujetos de prueba exhibieron diferentes comportamientos a lo largo del experimento, lo que generó diferencias significativas en el rendimiento entre los grupos de sujetos. Los oyentes expertos capacitados para el trabajo recibieron la misma precisión que el promedio de los sistemas presentados, en comparación con estudios previos sobre la capacidad humana para comprender escenas de audio cotidianas.

2.2 Estado del arte

En (Abeßer, 2020) se describe la estructura típica de un sistema de ASC. Constituido básicamente por: (1) Adquisición del audio, (2) Preparación de los datos y, (3) Modelado de los datos. En este trabajo se incluye un cuarto elemento: (4) la compresión de redes neuronales profundas para su ejecución en dispositivos de baja complejidad. Con la finalidad de dar respuesta a uno de los retos que se plantean en la revisión de Abeßer.

2.2.1 Adquisición del audio

El primer punto es importante por dos razones: la calidad del audio y como esta afecta al proceso de preparación de los datos y, a su vez, al modelado, así como el entono de ejecución y el despliegue en aplicaciones reales.

El audio digital es la representación codificada del audio analógico; este, a su vez, es la representación eléctrica del sonido. Por lo tanto, para obtener el audio digital son necesarios un sensor (como un micrófono) y un conversor analógico-digital (AD). El sonido, en física, se describe como la vibración provocada por la propagación de ondas mecánicas a través de un medio. El sensor suele ser tener una membrana o material sensible a esta perturbación y un transductor mecánico-eléctrico que permite traducir el sonido a una señal eléctrica analógica. Finalmente, esta señal eléctrica es digitalizada mediante un conversor AD. En este trabajo no se describe este proceso de forma exhaustiva. Pero, hay que tener presente cómo funciona la conversión del sonido a audio digital y como pueden afectar los elementos y procesos a la calidad de sonido percibida.

Otro aspecto que afecta a este primer punto es la privacidad y, es por ello por lo que los últimos esfuerzos en investigación sobre el ASC se centran en obtener modelos ejecutables en dispositivos cercanos al usuario final, sin necesidad de procesar los datos en centros de datos que podrían verse comprometidos a nivel de seguridad y afectar a la privacidad.

2.2.2 Preparación de los datos

El segundo punto incluye la representación de la señal, el preprocesado y técnicas de aumento de datos. La línea general en los sistemas ASC es el procesado de audio representado en formato mono o estéreo. Aun así, se han aplicado técnicas más sofisticadas como el trabajo de Zirlinski y Lee combinaron características espaciales de grabaciones biaurales con características espectro temporales para caracterizar la distribución del sonido en escenas acústicas (Blauert, 2013). Las transformaciones tiempo-frecuencia más utilizadas son la transformada de Fourier en tiempo corto (STFT), el espectrograma de Mel y el espectrograma wavelet (Zheng et al., 2018).

A nivel de preprocesamiento, el primer paso es la normalización de las características, que permite acelerar la convergencia de los algoritmos de descenso de gradiente. Otra aplicación es el escalado logarítmico, que permite reducir el rango dinámico de las escenas de audio. Otros de los retos más importantes se refieren al ruido y cancelación de ecos. Los algoritmos de ASC con mejores resultados en los últimos desafíos de DCASE utilizaron casi exclusivamente representaciones de espectrogramas basadas en el espaciado logarítmico de las frecuencias (Martín-Morató et al., 2021) y siguen centrándose en representaciones fijas de la señal basadas en el espectrograma en lugar de en transformaciones de la señal.

Las técnicas de augmento de datos se han vuelto de vital importancia en los modelos actuales de aprendizaje profundo basados en redes convolucionales; donde se requiere de grandes cantidades de datos. En algunos campos de la inteligencia artificial, como en el de la visón por computador, existen conjuntos de datos enromes, como el caso de ImageNet con alrededor de 14 millones de imágenes y 21000 clases (Russakovsky et al., 2015). El *machine listening*, queda por detrás, aunque existe el conjunto AudioSet (Gemmeke et al., 2017), que contiene 2,1 millones de fragmentos de audio divididos en 527 clases. Los algoritmos ASC recientemente propuestos utilizan métodos de aumento de datos de mezcla o basados en GAN para aumentar la cantidad disponible de datos de entrenamiento.

Para el desarrollo de este trabajo se han estudiado las representaciones del audio digital en forma de espectrogramas y algunas técnicas de aumento de datos.

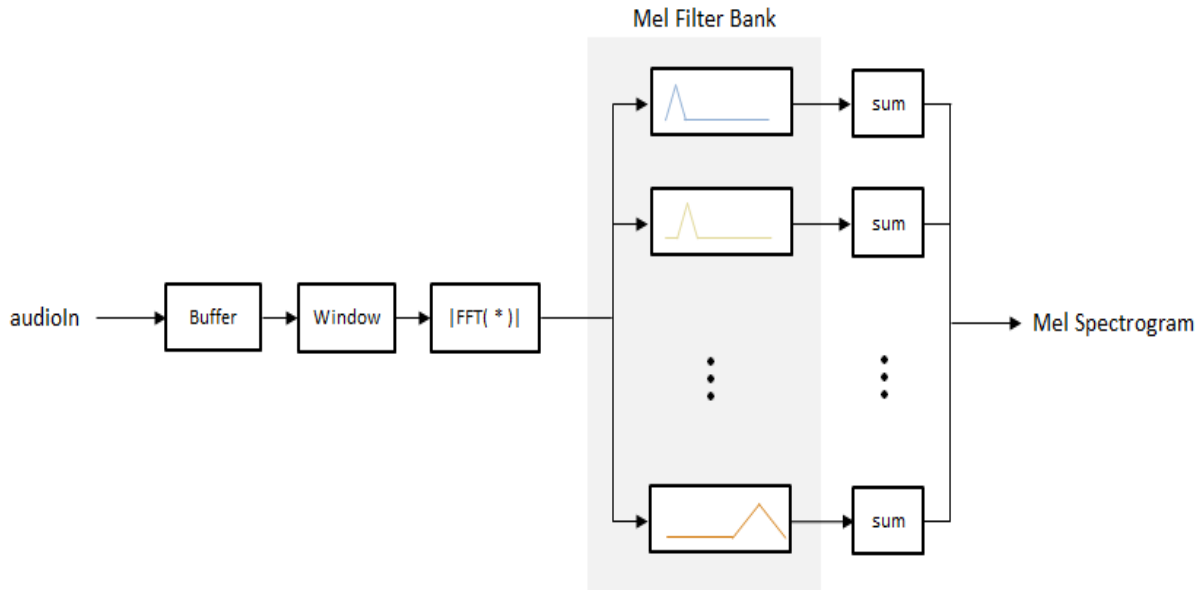

Figura 2: Diagrama de cómputo para la representación del espectrograma Mel. Fuente: (https://es.mathworks.com/help/audio/ref/melspectrogram_1.png)

2.2.2.1 Representación del espectrograma

En la Figura 2, se describe el proceso de cómputo del espectrograma en bandas de energía logarítmica de Mel, propuesto por Rabiner (Rabiner, 2010)

En este algoritmo, la entrada de audio se almacena primero en tramas con un número de muestras (determinado por la duración del fragmento y la frecuencia de muestreo aplicada). Las tramas se solapan con el número de muestras (o duración); normalmente se solapan un 50% de las muestras entre ventanas. La ventana especificada se aplica a cada trama y, a continuación, la trama se convierte al dominio frecuencial según un número de puntos determinado por la transformada rápida de Fourier (FFT).

Es importante saber que la FFT, que se calcula para el análisis espectral de cada segmento de la señal, trabaja sobre un número de muestras igual a una potencia de 2. Si el número de muestras del segmento no llega a una potencia de 2, la FFT añade ceros (*zero padding*) hasta el siguiente número «potencia de 2». Por tanto, es importante escoger un tiempo de ventana o número de muestras que, en función de la frecuencia de muestreo, determine un número de muestras el más aproximado a una potencia de 2 sin sobrepasarla.

Otro aspecto importante para tener en cuenta es que las frecuencias de muestreo tienen que cumplir con el teorema de Nyquist. Es decir, la frecuencia de muestreo debe ser como mínimo el doble del ancho de banda de interés. Por ejemplo, si deseamos obtener información sobre

Frecuencia de muestreo (Hz)	Duración de la ventana (ms)	Muestras por ventana	Muestras FFT	Zero padding
44100	46	2028	2048	20
	40	1764	1764	0
	23	1014	1024	10
	10	441	441	0
22050	46	1014	1024	10
	23	507	512	5
	20	441	441	0

Tabla 1: Número de muestras de FFT recomendados por duración de ventana y frecuencia de muestreo

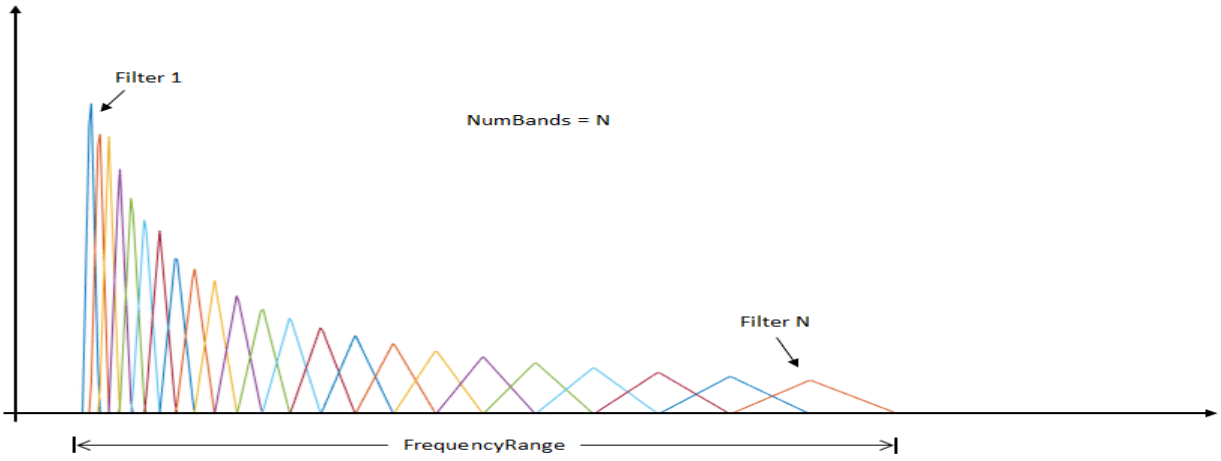

Figura 3: Banco de filtros Mel. Fuente: (https://es.mathworks.com/help/audio/ref/melspectrogram_2.png)

las frecuencias hasta 20000 Hz, la frecuencia de muestreo debe ser como mínimo de 40000 Hz. En este caso la frecuencia estandarizada para audio es de 44100 Hz.

La Tabla 1 muestra los valores recomendados según la frecuencia, duración de la ventana y muestras de la FFT.

Finalmente, cada fragmento de la representación en domino frecuencial pasa por un banco de filtros Mel. Estos bancos (Figura 3) representan una escala perceptiva de tonos tienen la misma distancia entre sí desde el punto de vista de la percepción auditiva humana. El punto de referencia entre esta escala y la medición de frecuencia normal se define asignando un tono perceptivo de 1000 mels a un tono de 1000 Hz, 40 dB por encima del umbral del oyente.

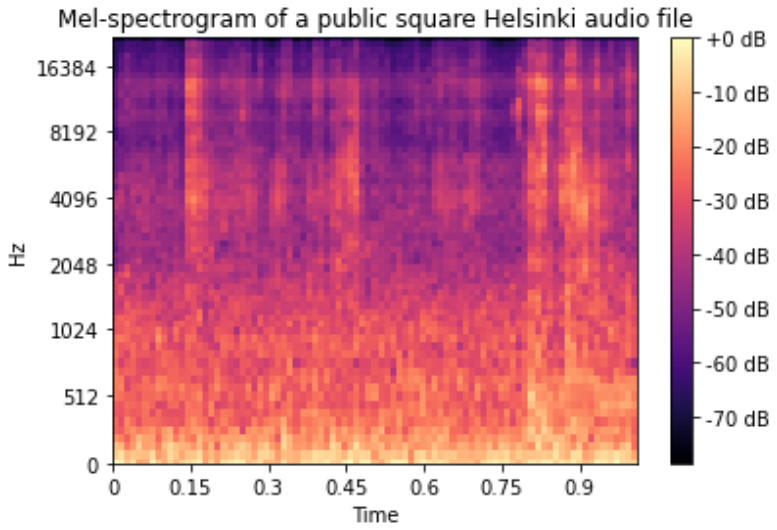

Figura 4: Espectrograma Mel del audio: public_square-helsinki-112-3243-6-c.wav (frecuencia de muestreo: 44100 Hz, tiempo de ventana de 23 ms con solapamiento de 11,5 ms y con 64 filtros Mel. La matriz resultante es de 64x87 pixeles). Fuente: propia

Por encima de aproximadamente 500 Hz, los oyentes juzgan que los intervalos cada vez más grandes producen incrementos de tono iguales (Stevens, 1937).

Los valores espectrales emitidos por el banco de filtros Mel se suman y, a continuación, los canales se concatenan de modo que cada fotograma se transforma en un vector columna (Figura 4).

2.2.2.2 Técnicas de aumento de datos de audio

En este apartado se describen algunas de las técnicas comentadas. En concreto las más comunes que hacen referencia al aumento de audio aplicado directamente sobre la señal digital (Salamon, 2017) y, el aumento aplicado sobre el espectrograma propuesto en 2019 por investigadores de Google Brain y que es aplicado en la mayoría de los trabajos del estado del arte de los sistemas ASC (Park et al., 2019).

A. Aumento de la señal de audio (Salamon, 2017)**.**

A.1 Inyección de ruido gaussiano: Una técnica muy común es la suma de ruido gaussiano a la propia señal. Este aumento se sustenta en el hecho de que algunos de los elementos implicados en la captura del sonido (ver apartado 2.2.1 Adquisición del audio), pueden introducir ruidos debido a interferencias entre componentes electrónicos y corrientes indeseadas. La idea es dotar al modelo con la capacidad de generalizar mejor frente a captaciones que contienen ruido.

A.2 Estiramiento del tiempo. Se ralentiza o acelera el audio manteniendo el tono

A.3 Estiramiento del tono. Se sube o baja el tono manteniendo la duración

A.4 Compresión de rango dinámico. Se comprime el rango dinámico de la muestra. Es decir, se reduce la diferencia de nivel de presión sonora existente entre los sonidos más bajos y los sonidos más altos. Esta presión se mide en decibelios (dB).

B. Aumento del espectrograma (Park et al., 2019).

Este método trata el problema directamente sobre la representación visual; es decir sobre el espectrograma en lugar de modificar el audio. La política de aumento consiste en deformar las características, enmascarar bloques de frecuencia y enmascarar bloques de tiempo.

El enmascaramiento temporal (Figura 5) se aplica de forma que se enmascaran los pasos temporales consecutivos [t0, t0+t].

De forma análoga, el enmascaramiento frecuencial (Figura 6) se aplica a bandas de frecuencia consecutivas [f0, f0+f]

Park et al., proponen una serie de políticas de aplicación de estas mascaras. Algunas de ellas se solapan ofreciendo distintas tipologías de aumentos. En la Figura 7, se representa la aplicación de la política LD, donde se enmascara 2 segmentos distintos, tanto para tiempo como para frecuencia.

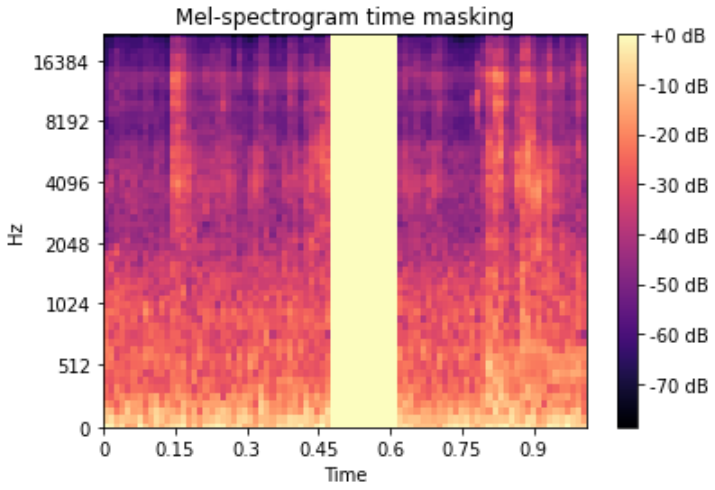

Figura 5: Representación de enmascaramiento temporal del espectrograma Mel. Fuente: propia

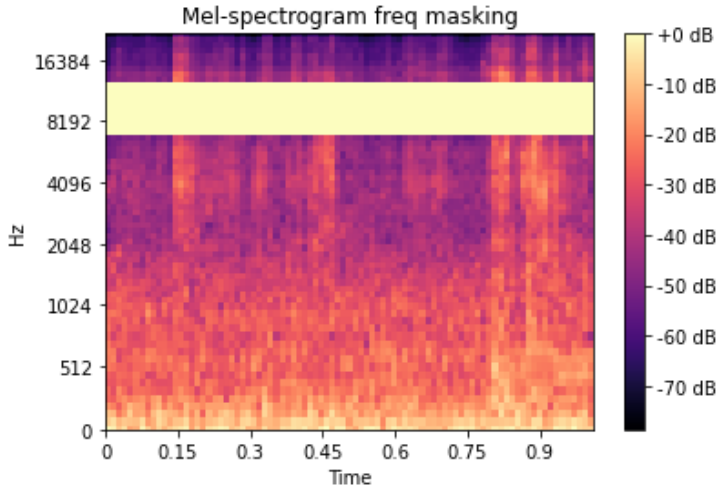

Figura 6: Representación de enmascaramiento frecuencial del espectrograma Mel. Fuente: propia

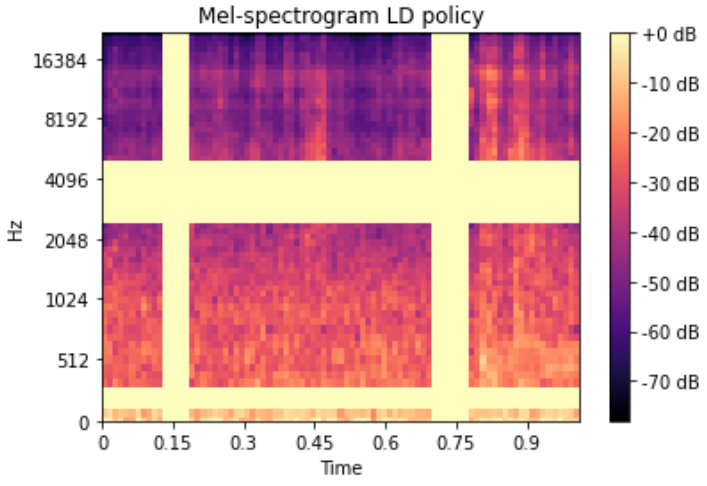

Figura 7: Representación de SpecAugment con política LD. Fuente: propia

2.2.3 Modelado

Desde la aportación de las redes convolucionales profundas (CNN) al campo de la visión por computador y su clara mejora respecto a los modelos clásicos de aprendizaje, los algoritmos ASC suelen utilizar arquitecturas de red basadas en CNN. Aun así, el uso de esta técnica en el campo del *machine listening*, no llego hasta unos años después con el trabajo de algunos investigadores que presentaron sus propuestas en el DCASE de 2016 (Bae et al., 2016; Battaglino et al., 2016), donde implementaban arquitecturas hibridas entre LSTM y CNN.

En paralelo, las arquitecturas CNN convencionales, basadas en capas consecutivas y lineales han ido quedando obsoletas en el campo de la percepción computacional. Siendo substituidas por redes CNN que se constituyen a base de bloques y siguen arquitecturas no lineales como las Inception (Szegedy et al., 2014) o las ResNet (He et al., 2015).

Recientemente, el uso de redes atencionales basadas en Transformers se ha incrementado debido a su rendimiento en las tareas del procesamiento del lenguaje natural. Desde que se propuso esta arquitectura (Vaswani et al., 2017), basada en mecanismos atencionales que solucionan el problema del desvanecimiento de gradiente en redes recurrentes, se ha aplicado en la mayoría de los campos de la inteligencia artificial. En (Baevski et al., 2020) se emplea esta arquitectura para generar un modelo (Wav2Vector), inspirado en los modelos de PNL, para la tarea de reconocimiento de voz.

Fruto de este avance, los investigadores han trabajado en modelos de visión aplicando Transformes (Dosovitskiy et al., 2021), también llamado ViT. Recientemente, se han desarrollado modelos basados en combinaciones de CNN y Vit (Mehta & Apple, 2021). Estos trabajos han servido de inspiración para modelos Transformers aplicados al campo del audio como los AST (*Audio Spectrogram Transformres*) (Chung et al., 2021). La limitación de estas redes es la gran cantidad de datos que precisan y el número de parámetros. Estas propuestas son muy novedosas. En consecuencia, las técnicas de compresión aplicadas en estas arquitecturas son muy complejas y, por el momento, el conocimiento respecto a esto es limitado.

En los próximos puntos se explican más en detalle algunos de los modelos citados anteriormente, empleados en las tareas de ASC de baja complejidad de eventos DCASE en años anteriores.

2.2.3.1 CNN: Arquitectura lineal

La idea fundamental detrás de las redes CNN es la de simular el proceso cognitivo de la percepción visual humana. El modelo descrito por Hubel y Wiesel en 1959 (Hubel & Wiesel, 1959), describe un conjunto de capas en la corteza visual, que, de forma jerárquica, determinan las propiedades de los objetos percibidos (bordes, forma, …).

Esta idea llevó a implementar un modelo computacional basado en capas de filtros convolucionales que permiten obtener, de forma similar a la percepción humana, las características y propiedades de las imágenes. Las CNN se popularizaron en el campo de la visión por computador después de poder aplicar esta implementación en unidades GPU (Cires¸an et al., 2012), superando los resultados obtenidos hasta entonces con redes neuronales clásicas.

La arquitectura consiste en un conjunto de capas convolucionales seguidas de una o varias capas de decisión. Las capas convolucionales son matrices que actúan como filtros que realizan operaciones sobre la imagen. El tamaño del filtro (*Kernel size*) así como el desplazamiento, sobre la imagen, medido en píxeles (*stride*) y, el número de filtros son hiperparámetros de la red. El valor de los filtros es lo que se aprende en el proceso de *backpropagation*.

Entre las distintas capas convolucionales, suelen añadirse otros filtros como *MaxPool*; este permite reducir el tamaño de la matriz obteniendo el valor máximo calculado sobre el mapa de características que cubre el filtro. De forma análoga, se decide el tamaño del filtro y el desplazamiento. Aunque, los parámetros de estos filtros no se entrenan puesto que el cálculo es fijo.

Finalmente, se añade una capa *Flatten* para secuenciar los valores y pasarlos a las capas densas de clasificación. EN la Figura 8, se muestra, de manera gráfica, el conjunto de capas descritos.

2.2.3.2 CNN: ResNet

Las redes residuales convolucionales o ResNet (*Residual Networks*) propuestas por Kaiming et al. (2015), se inspiran en la capacidad biológica de las neuronas en realizar conexiones con otras neuronas situadas en capas no necesariamente contiguas. Esta arquitectura permite entrenar CNN muy profundas controlando el problema del desvanecimiento del gradiente. La propuesta gano el desafío ImageNet de 2015.

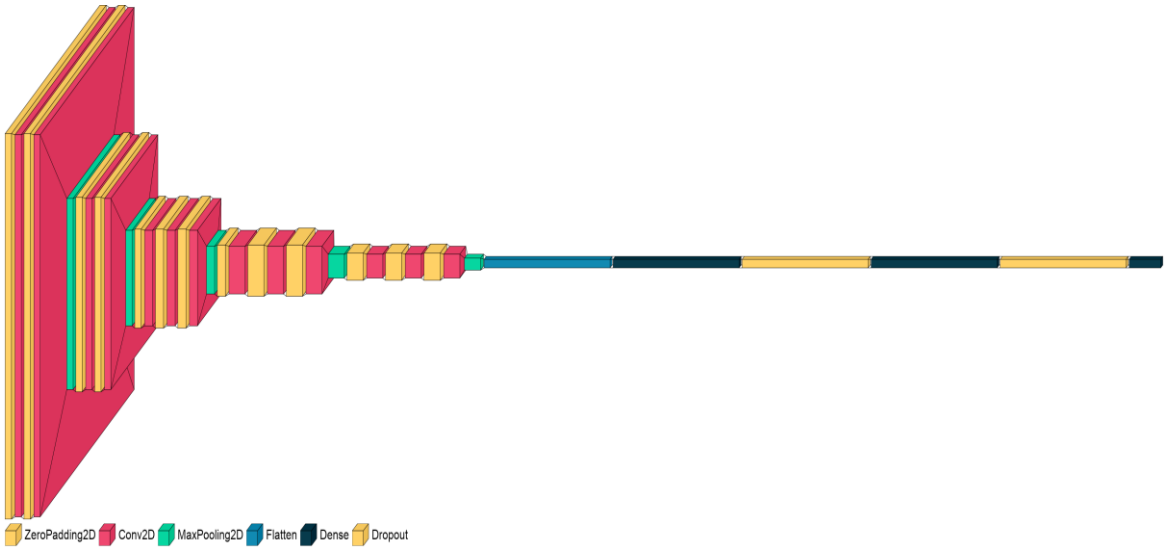

Figura 8: Arquitectura gráfica de una CNN. Fuente: (https://github.com/paulgavrikov/visualkeras/raw/master/figures/vgg16_legend.png)

La figura 9 muestra la idea principal de esta arquitectura. Donde la salida del bloque residual es la suma entre el mapa de características *F(x)* y el mapa identidad *x*. Su hipótesis es que dejar que las capas apiladas se ajusten a un mapeo residual es más fácil que dejar que se ajusten directamente al mapeo subyacente deseado (He et al., 2015).

Para implementar este procedimiento, se emplean conexiones "conexiones de acceso directo" (*shortcut conections, SC*). Estas no añaden parámetros ni complejidad computacional, y las redes pueden seguir siendo entrenadas con los algoritmos de *backpropagation* y desarrolladas con librerías habituales.

Las propuestas con mejores resultados en ediciones anteriores para resolver esta tarea (clasificación de escenas acústicas con modelos de baja complejidad), se basan en arquitecturas ResNet. En la Tabla 2 se muestra la comparativa de los mejores modelos del 2021 (Martín-Morató et al., 2021).

2.2.4 Compresión de redes neuronales

Una de las limitaciones más importantes de las redes profundas es la gran cantidad de parámetros. Esto influye en la complejidad computacional y, a su vez, en la velocidad de inferencia que, implica la incapacidad de despliegue en aplicaciones a tiempo real y en dispositivos de baja complejidad.

La potencial aplicabilidad de la IA en este campo ha llevado a investigadores a desarrollar métodos de compresión y optimización de redes profundas que permiten reducir la complejidad sin comprometer del rendimiento.

Normalmente, se refieren a estos sistemas con el término "*Edge inteligence*" (Liu et al., 2020). Como referencia a las disciplinas confluyen para tal fin: la inteligencia artificial junto con el "*Edge computing*". En su revisión de los métodos más novedosos que aparecen en la literatura, Liu et al., enumera distintas técnicas que buscan solucionar el problema: modelos hechos a mano, compresión de modelos, búsqueda de arquitectura neuronal consciente del hardware y modelos de aprendizaje profundo adaptativo.

En este trabajo se estudian los métodos relacionados con la compresión y optimización, basados en la reducción de parámetros redundantes en redes sobreparametrizadas.

2.2.4.1 Cuantificación

La gran cantidad de parámetros en redes profundas producen una gran cantidad de resultados intermedios en las capas de activación y mapas de características de las redes CNN. Esto se traduce en una cantidad de memoria inasumible en dispositivos como microcontroladores. Por ejemplo, la arquitectura ResNet-50 (He et al., 2015) tiene 26 millones de pesos de parámetros, genera 16 millones de activaciones en una inferencia, requiere alrededor de 168 MB de espacio de memoria y necesita al menos 3 GB/s de ancho de banda de memoria.

La cuantificación ("*quantization*") afronta este problema convirtiendo los pesos de las redes de precisión alta (float32) a pesos de precisión menor (int8, float16) conservando el rendimiento a nivel de exactitud. La función de cuantificación de describe como:

$$\mathrm{Q(r)} = \mathrm{Int}\left(\frac{\mathrm{r}}{\mathrm{S}}\right) - \mathrm{Z}\,,$$

donde, *Q* es el operador de cuantificación, *r* es un valor real de entrada (activación o peso), S es el factor de escala, y Z es un entero "*zero point*". Como muestra la figura 10, esta función

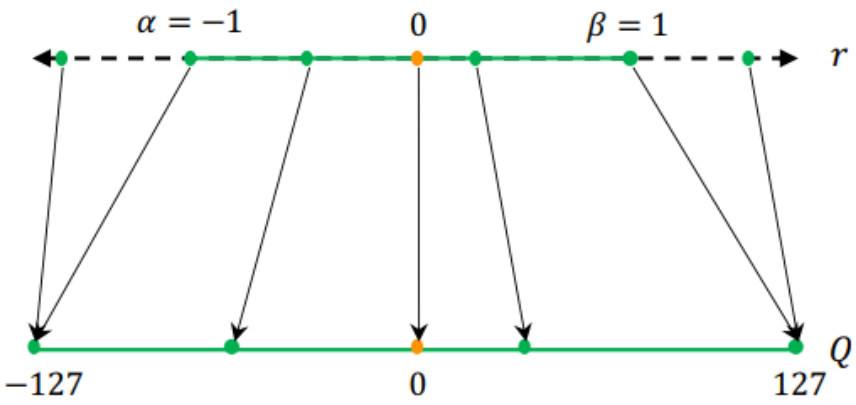

Figura 9: Ejemplo con rango restringido a valores reales entre [-127, 127]. Fuente: (Gholami et al., 2021)

es un mapeo de valores reales r a algún valor entero (Gholami et al., 2021). La elección del factor S de escalado es lo que determinará el rango final de valores:

$$S = \frac{\beta - \alpha}{2^b - 1},$$

donde [α, β] denota el rango de recorte, un rango limitado con el que estamos recortando los valores reales, y b es el ancho de bits de cuantificación.

La importancia de poder aplicar estos métodos se traduce en la sencillez de uso, con librerías como Tensorflow; quienes han desarrollado TFLite: un marco de código abierto para la inferencia y despliegue de modelos de aprendizaje profundo en sistemas integrados (David et al., 2020). TFLite permite obtener binarios ejecutables de forma que puedan ser embebidos en microcontroladores.

2.2.4.2 Destilación de conocimiento

La destilación de conocimiento (*Knowledge Distillation, KD*) no es un método de compresión en sí mismo (como la cuantificación). No se optimiza la red desarrollada directamente, si no que se transfiere la capacidad de generalización de una red mayor (red maestra) a una red de menor tamaño (red alumno).

Existen multitud de técnicas y algoritmos de KD diferenciadas por la representación del conocimiento, métodos de destilación o arquitecturas maestro-alumno (Gou et al., 2021).

La técnica de KD más popular para tareas de clasificación de imágenes (Hinton et al., 2015), en la que se centra este proyecto, se basa en obtener las predicciones de los datos producidas

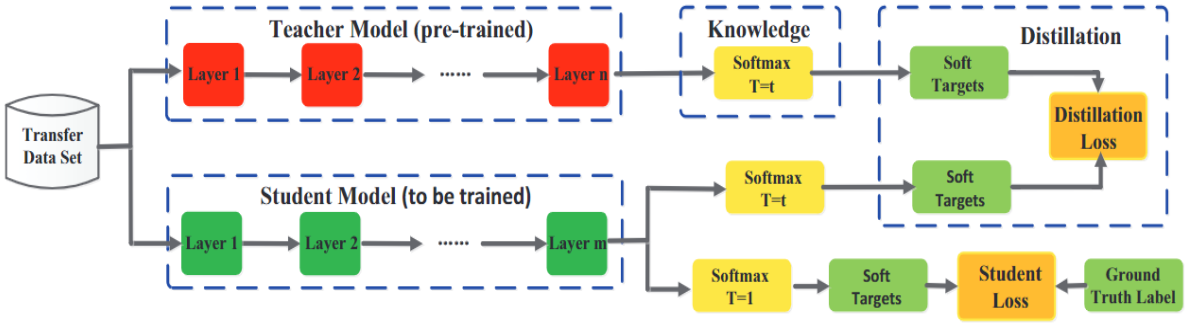

Figura 10: Proceso de KD basado en la propuesta de Hinton et al., Fuente: (Gou et al., 2021)

por la red maestra y tratarlas como conocimiento para la red alumna (destilación). A diferencia de la técnica de transferencia de aprendizaje, no se transfiere el valor de los pesos, si no este conocimiento abstracto. Es decir, se sustenta en la representación del conocimiento derivada de la salida del modelo maestro; el alumno debe "imitar" la salida del maestro.

Esto se puede formular como:

$$L_{ResD}(z_t, z_s) = L_R(z_t, z_s),$$

donde, z_t y z_s representan los *logits* de la red maestra y estudiante respectivamente y, L_R representa la función de coste de divergencia entre esos *logits*. Los *logits*, son la salida de la última capa. Esta salida acuñada por Hinton et al., como *soft targets*, son las probabilidades de que la entrada pertenezca a las clases y se pueden estimar mediante una función *softmax* como:

$$p(z_i, T) = \frac{exp\left(\frac{z_i}{T}\right)}{\sum_j \exp\left(\frac{z_j}{T}\right)},$$

donde z_i es el *logit* para la *i*-ésima clase, y se introduce un factor de temperatura (T) que controla la distribución de probabilidad. En las activaciones *softmax* por defecto este valor es *T=1*, y se obtienen distribuciones más suaves a medida que aumenta el valor de T.

Con esto se puede reformular la ecuación de coste de divergencia como:

$$L_{ResD}\big(p(z_t, T), p(z_s, T)\big) = L_R\big(p(z_t, T), p(z_s, T)\big),$$

Es decir, para destilar el conocimiento, se entrena la red maestra con un valor de T > 1. Esta distribución de probabilidad suavizada se emplea en la función de divergencia junto con el entrenamiento de la red alumno con el mismo valor de T. Finalmente, una vez el alumno se ha entrenado, se vuelve al valor de T = 1 por defecto. En la Figura 11 se resume el procedimiento.

2.3 Conclusiones sobre el estado del arte

Para llevar a cabo el desarrollo de modelos de baja complejidad para la clasificación de escenas acústicas, es necesario aplicar conocimientos de diferentes disciplinas: procesamiento de la señal digital (en concreto, audio), aprendizaje profundo y, compresión y optimización de redes profundas.

Por el alcance del trabajo final de máster, no es asumible profundizar en cada uno de estos campos. Esto es un riesgo que se ha valorado y se ha solventado recopilando la información relevante para la tarea en concreto.

Por consiguiente, se ha estudiado lo relacionado con el procesamiento de audio y representación de los espectrogramas Mel. Estas dos áreas no se perciben como un riesgo importante debido al conocimiento previo sobre la materia.

En cuanto al aprendizaje profundo, se han estudiado, por un lado, las técnicas de aumento de datos aplicadas al audio y los modelos implementados en otras tareas similares. Además, se han estudiado otros modelos más novedosos relacionados con el propósito del proyecto, como son los Transformers ViT y las redes MobileViT. Esta parte del estudio necesita platearse rigurosamente ya que existen multitud de modelos y, su aplicación está estrechamente relacionada con las herramientas y equipos disponibles. Se añade, además, una complicación extra: en los modelos propuestos, se deben poder aplicar las técnicas de compresión. Por esta razón, como se comenta en el siguiente apartado (3 Objetivos y metodología de trabajo), las fases de desarrollo serán paralelas e iterativas.

La tercera área de estudio relacionada con este proyecto es, quizás, la más compleja; dado que requiere de un conocimiento sobre redes profundas muy asentado. Después del estudio se valora como un riesgo importante. Por este motivo, se aplicarán las técnicas que aparecen en la literatura sobre la temática en concreto de este trabajo. Una de ellas, la compresión por cuantificación es un requerimiento en sí para cumplir con el objetivo.

3 Objetivos y metodología de trabajo

Los objetivos y como llegar a estos es clave en un proyecto de investigación y desarrollo. En los siguientes apartados se describe cual es la intención principal, las necesidades teóricas y técnicas para afrontar el desarrollo y, la forma de realizar con éxito este trabajo.

3.1 Objetivo general

El objetivo principal es implementar un modelo que consiga superar el rendimiento del modelo base de la tarea 1 de DACSE 2022 y, que esté dentro del límite de complejidad computacional propuesto.

Se entiende por rendimiento la capacidad de generalización de la red. Y, se medirá según el *log loss (multiclass cross-entropy*). De hecho, se penaliza esta métrica agregando, en el conjunto de prueba, captaciones de dispositivos no empleados en el conjunto de entrenamiento (ver 4.1 Conjunto de datos).

Además, se añaden dos requisitos referentes a la complejidad computacional para cumplir con el objetivo de poder desplegar este modelo en dispositivos de baja complejidad.

Los valores de entrenamiento obtenidos sobre el modelo de desarrollo son:

1. *log los*s: 1,575
2. accuracy: 42,9%

Requisitos de complejidad computacional:

3. Número de parámetros: máximo de 128.000, con el tipo de variable INT8
4. MACs (multiply-accumulate operations): máximo 30 millones (30 MMAC)

3.2 Objetivos específicos

Para conseguir el propósito general se deberán cumplir los siguientes puntos:

- Estudiar el estado del arte sobre ASC.
- Estudiar técnicas de compresión de redes neuronales.
- Diseñar modelos que permitan obtener el rendimiento esperado
- Comprimir dichos modelos para ejecutarlos en dispositivos de baja complejidad, sin comprometer el rendimiento en cuanto a precisión.
- Evaluar el rendimiento y la complejidad computacional del modelo comprimidos.

3.3 Metodología de trabajo

3.3.1 Fase de estudio del problema

Durante esta fase se realizarán las tareas de investigación y búsqueda de literatura relacionada con el problema a tratar. También, se identificarán los elementos técnicos para el desarrollo de la solución y las librerías de software útiles para el desarrollo; para el cual se utilizará el lenguaje de programación Python.

3.3.2 Fases iterativas

Se propondrá una posible solución basada en los conocimientos adquiridos en la fase previa. Se implementará y se evaluarán los resultados según el cumplimiento de los objetivos.

Esta fase se realizará de forma iterativa, siguiendo un modelo de desarrollo ágil, modularizando el desarrollo y separando las tareas en *sprints* (en este caso obteniendo resultados parciales cada semana). Estos módulos siguen la arquitectura planteada en el estado del arte, y para cada uno de ellos se plantean diversas tareas. Hay que tener en cuenta que el alcance de cada uno de los módulos puede ser modificado según avance el proyecto, aunque el objetivo final está claramente definido y no se valorarán nuevas tareas que añadan un desvió demasiado arriesgado para cumplirlo.

Módulo 1: Procesamiento y representación del audio

- Análisis de los datos
- Procesamiento de audio y representación del espectrograma Mel
- Aplicación de las técnicas de aumento

Módulo 2: Modelado

- Recreación y entrenamiento del modelo de referencia
- Propuestas de modelos en base a los estudiados
- Sesiones de entrenamiento y validación
- Refinamiento del modelo propuesto

Módulo 3: Compresión

- Entrenamiento del modelo alumno
- Aplicación de técnicas de destilación

- Cuantificación

El desarrollo de cada módulo no es de ninguna manera secuencial. Siguiendo el paradigma de metodología ágil, en cada *sprint* se incluirán tareas de cada uno de los módulos para conseguir subobjetivos parciales cada semana. Y, se incrementará de forma gradual el alcance del proyecto según se vayan analizando los distintos resultados obtenidos al final de cada desarrollo.

3.3.3 Análisis de resultados y conclusiones

La última parte del proyecto servirá de análisis retrospectivo tanto del resultado de desarrollo como del trabajo realizado.

Se valorará el cumplimiento de los objetivos, las posibles mejoras técnicas y las posibles vías de resolución del problema no exploradas.

También, se realizará una crítica personal teniendo en cuenta la adquisición de conocimientos y el desempeño en las distintas tareas. Se analizarán los problemas encontrados y los riesgos que no se tuvieron en cuenta al inicio y, se valorarán las propuestas aplicadas para solventar dichas complicaciones.

4 Planteamiento de la comparativa

En este capítulo se describe el conjunto de datos con el que se trabajará y que determinará la clasificación y, a su vez, el procesamiento de datos y el modelado.

Se describen los modelos desarrollados: el modelo de referencia y el modelo propuesto. Posteriormente, se enumeran distintos métodos para la reducción de parámetros en las redes neuronales que se desarrollarán para realizar el estudio comparativo; el cual se medirá en base a las métricas y complejidad computacional que se describen en este capítulo. También, se diseñarán un conjunto experimentos a llevar a cabo para comparar los resultados obtenidos. Finalmente, se explica el entorno y librerías empleadas para el desarrollo.

4.1 Conjunto de datos

El conjunto de datos de desarrollo para esta tarea es TAU Urban Acoustic Scenes 2022 Mobile (Heittola et al., 2020). Este, contiene grabaciones de 12 ciudades europeas en 10 escenas acústicas diferentes utilizando 4 dispositivos distintos. Además, se crearon datos sintéticos para 11 dispositivos móviles basados en las grabaciones originales. El conjunto de datos tiene exactamente el mismo contenido que TAU Urban Acoustic Scenes 2020 Mobile, pero los archivos de audio tienen una duración de 1 segundo (por tanto, hay 10 veces más archivos que en la versión de 2020).

Las grabaciones se realizaron con cuatro dispositivos que capturaron el audio simultáneamente. El dispositivo de grabación principal consiste en un Soundman OKM II Klassik/studio A3; un micrófono biaural electret y una grabadora de audio Zoom F8 que utiliza una tasa de muestreo de 48kHz y una resolución de 24 bits, denominado dispositivo A. Los otros dispositivos son dispositivos de cliente habituales: el dispositivo B es un Samsung Galaxy S7, el dispositivo C es un iPhone SE y el dispositivo D es una GoPro Hero5 Session.

Los datos de audio se grabaron en Ámsterdam, Barcelona, Helsinki, Lisboa, Londres, Lyon, Madrid, Milán, Praga, París, Estocolmo y Viena. El conjunto de datos fue recogido por la Universidad Tecnológica de Tampere entre 05/2018 - 11/2018. La recopilación de datos recibió financiación del Consejo Europeo de Investigación, acuerdo de subvención 637422 EVERYSOUND.

Además, se simulan 11 dispositivos móviles S1-S11 utilizando el audio grabado con el dispositivo A, junto con respuestas al impulso grabadas con dispositivos reales y una compresión de rango dinámico adicional, con el fin de simular grabaciones realistas. Una

grabación del dispositivo A se procesa mediante convolución con la respuesta al impulso seleccionada y, a continuación, se procesa con un conjunto seleccionado de parámetros para la compresión de rango dinámico (específicos del dispositivo). Las respuestas al impulso son datos patentados y no se publicarán.

El conjunto de datos de desarrollo comprende 40 horas de datos del dispositivo A, y cantidades más pequeñas de los otros dispositivos. El audio se proporciona en un formato de un solo canal de 44,1 kHz y 24 bits.

- Escenas acústicas/etiquetas semánticas (10):
- Aeropuerto - airport
- Centro comercial interior - shopping_mall
- Estación de metro - metro_station
- Calle peatonal - street_pedestrian
- Plaza pública - public_square
- Calle con tráfico medio - street_traffic
- Viajar en tranvía - tram
- Viajar en autobús - bus
- Viajar en metro - metro
- Parque urbano - park

En los siguientes apartados, se describen los conjuntos de datos empleados para la realización de los distintos entrenamientos y comparativas. Estos conjuntos se refieren a ficheros *.csv*; en ningún caso a la representación de audio que se aplicará posteriormente. Es decir, en estos ficheros se incluye la ruta al fichero de audio (*.wav*), la etiqueta semántica (“scene_label”), el dispositivo con el que se capturo la escena acústica (“source_label”) y la localización (“location”).

Posteriormente, el capítulo: 4.2 Representación de los datos, se especifica como se han procesado y representado los datos de audio.

4.1.1 Conjunto de entrenamiento

El conjunto de entrenamiento contiene 139.620 observaciones distribuidas como se muestra en las Figura 12 y Figura 13.

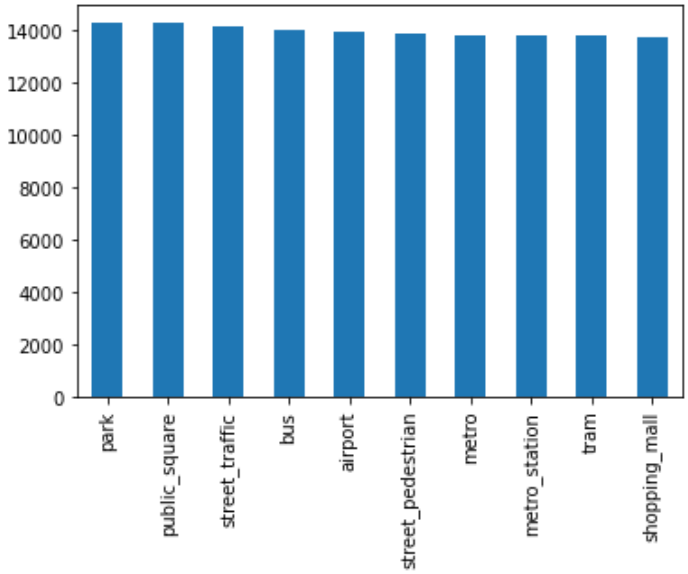

Figura 11: Distribución de observaciones del conjunto de entrenamiento por escenas acústicas. Fuente: propia

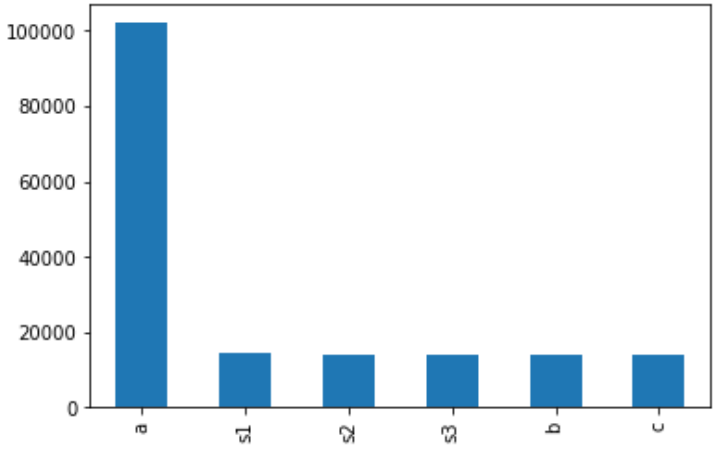

Figura 12: Distribución de observaciones del conjunto de entrenamiento por dispositivo. Fuente: propia

Como se puede observar, el conjunto de entrenamiento esta desbalanceado en cuanto a observaciones por dispositivo.

4.1.2 Conjunto de prueba

El conjunto de prueba contiene 29.680 observaciones distribuidas. Se muestra en las Figura 14 y Figura 15.

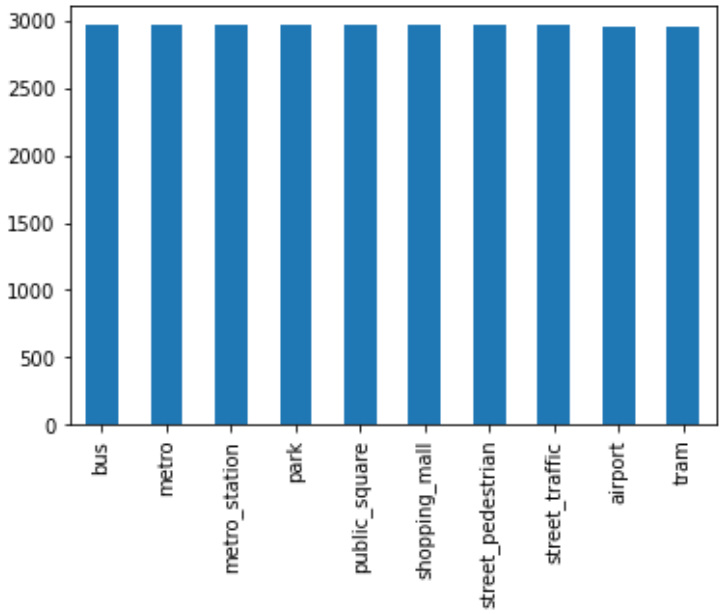

Figura 13: Distribución de observaciones del conjunto de prueba por escenas acústicas. Fuente: propia

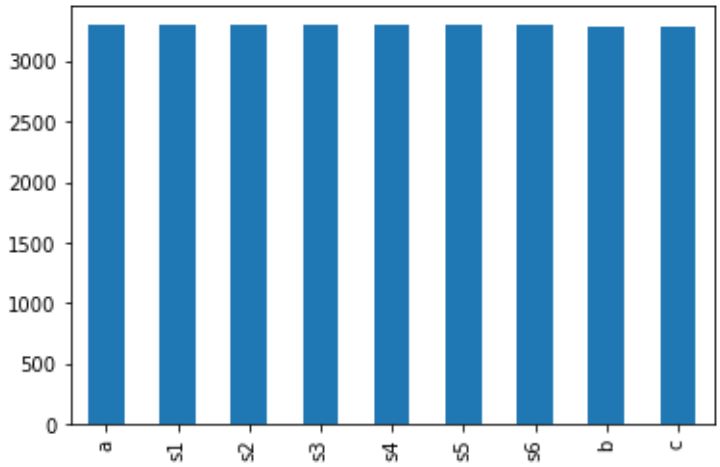

Figura 14: Distribución de observaciones del conjunto de prueba por dispositivo. Fuente: propia

Como se puede apreciar, en este se incluyen dispositivos que no existen en el conjunto de entrenamiento (s4, s5 y s6). Con esto se pretende penalizar la evaluación de los modelos poco robustos y que no sean capaces de generalizar el aprendizaje.

4.1.3 Conjunto de datos para aumento

Además, se ha generado un conjunto de datos con un número de observaciones menor respecto al conjunto de entrenamiento base; en los cuales se aplicará el método de aumento de datos que se explica en el siguiente punto (*4.2.2 Aumento de datos basado en la respuesta*

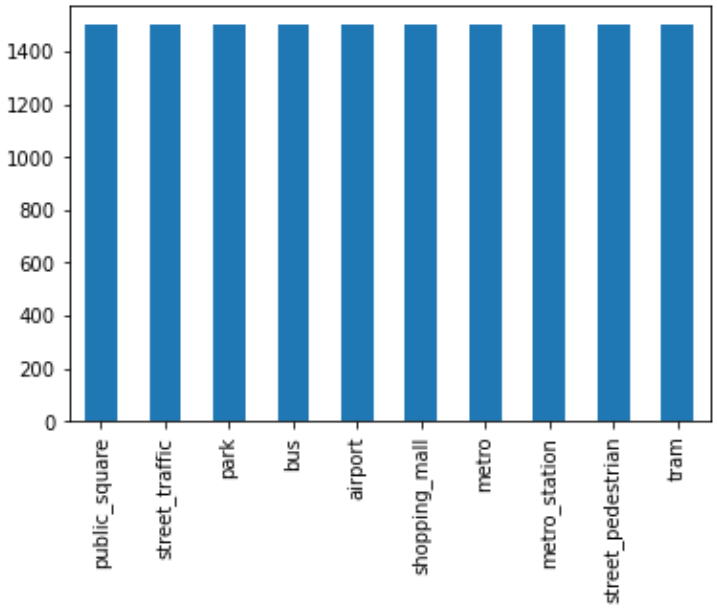

Figura 15: Distribución de observaciones del conjunto de aumento por escenas acústicas. Fuente: propia

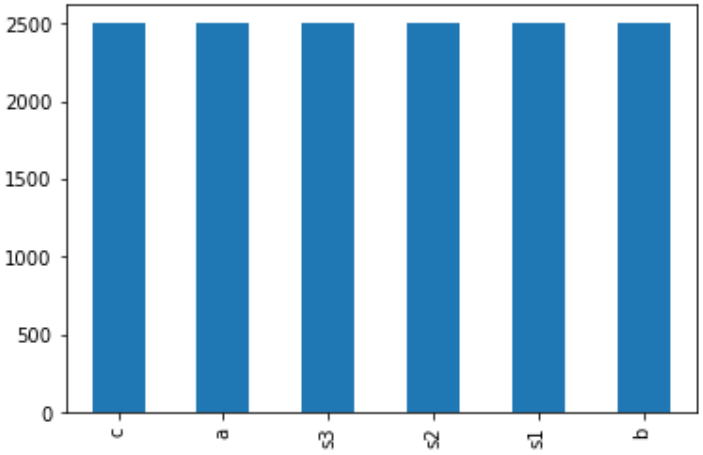

Figura 16: Distribución de observaciones del conjunto de aumento por dispositivo. Fuente: propia

al impulso de dispositivos reales no observados). Esto es debido a que la cantidad de audios asciende a 30GB y, por limitaciones en equipamiento y capacidad de cómputo, aplicar el aumento de datos sobre esta cantidad de no es asumible.

Para generar este nuevo conjunto se ha agrupado todas las observaciones según los atributos: "source_label", "location", "scene_label". De cada uno de estos grupos se han recogido 50 muestras aleatorias. Obteniendo en total 15000 muestras (Figura 16). Cada uno de estos subconjuntos, por lo tanto, ha quedado balanceado en cuanto a escenas acústicas, localizaciones y dispositivos de grabación (Figura 17). Finalmente, los datos para cada subconjunto se han mezclado de forma aleatoria.

4.2 Representación de los datos

A continuación, se explica el procesado del audio en base al conocimiento adquirido en el estudio del estado del arte. Se generan distintas representaciones de los conjuntos de datos descritos anteriormente según distintos parámetros del espectrograma Mel y métodos de aumento (apartado 2.2.2 Preparación de los datos).

4.2.1 Espectrograma de Mel

Siguiendo la línea de los trabajos estudiados, el audio se ha representado en espectrogramas de Mel. Para la posterior comparación de rendimiento se han creado estas representaciones con distintos parámetros:

Un primer caso se han procesado los datos de entrenamiento y test teniendo en cuenta el modelo de referencia, a 44100 Hz de muestreo y 40 filtros Mel. Según la definición del desarrollo de referencia, la ventana es de 40 ms; por lo tanto 1764 muestras para la FFT y 882 muestras para el solapamiento (20 ms). Otro conjunto se ha generado con estas especificaciones exceptuando el número de filtros; el cual se ha incrementado 80.

De acuerdo con lo establecido en la tabla: “Tabla 1: Número de muestras de FFT recomendados por duración de ventana y frecuencia de muestreo”, se han generado otros conjuntos de procesado según la frecuencia de muestreo, duración de la ventana y número de filtros Mel.

Un conjunto a 22050 Hz. Para este el número de muestras elegido es de 1024, con un solapamiento de 512. Lo que implica que cada segmento de audio donde se aplicará el banco de filtros Mel, es de 46 ms de ventana, con un solapamiento de 23 ms. El número de filtros Mel escogido en este caso es de 64.

También, se ha procesado los mismos datos con la frecuencia de muestreo original, de 44100 Hz. En este caso aplicando el mismo número de muestras de la FFT y solapamiento. Con lo que la ventana tiene una duración de 23 ms y un solapamiento de 11,5 ms. En este segundo caso, se ha aplicado 64 y 128 filtros Mel.

En resumen, se ha obtenido las representaciones en espectrograma de Mel (para el conjunto de entrenamiento y test) que se muestran en la Tabla 3.

conjunto	Frec. Muestreo (Hz)	Muestras FFT	Muestras solape	Ventana (ms)	Solape (ms)	Núm Mel	Imagen (pixeles)
base	44100	1764	882	40	20	40	40x51
Procesado_1	44100	1764	882	40	20	80	80x51
Procesado_2	22050	1024	512	46	23	40	40x44
Procesado_3	44100	1024	512	23	11.5	40	40x87
Procesado_4	44100	1024	512	23	11.5	80	80x87

Tabla 2: Resumen del conjunto de representaciones del audio

4.2.2 Aumento de datos basado en la respuesta al impulso de dispositivos reales no observados

Uno de los retos a superar en este proyecto es la capacidad de generalización del modelo. Es por ello por lo que en el conjunto de prueba se incluyen dispositivos no observados en el conjunto de entrenamiento. Cada dispositivo ofrece una respuesta frecuencial distinta; un hecho que sucedería en caso de desplegar el modelo en una aplicación real.

Los autores de esta tarea, del evento DCASE, han generado “dispositivos ficticios” a partir de la convolución de algunas captaciones con respuestas al impulso de dispositivos reales (ver apartado 4.1 Conjunto de datos). Aunque, como puntualizan, las respuestas al impulso de esos dispositivos están patentadas y no son públicas.

Siguiendo esta línea, se ha generado un nuevo conjunto de datos partiendo del conjunto para aumento. Para ello se ha empleado respuestas al impulso de diversos micrófonos de estudio generadas por Stewart Tavener y compartidas bajo licencia *creative commons* en su blog (*Microphone Impulse Response Project: About MicIRP*, 2022).

Las respuestas al impulso se generaron con un método de barrido sinusoidal en una pequeña cabina, tratada con espuma acústica, situada a unos 20 o 30 cm de la fuente. Y, con el software de análisis Fuzzmeasure[1], se exportaron los datos como respuestas al impulso.

[1] https://www.rodetest.com/

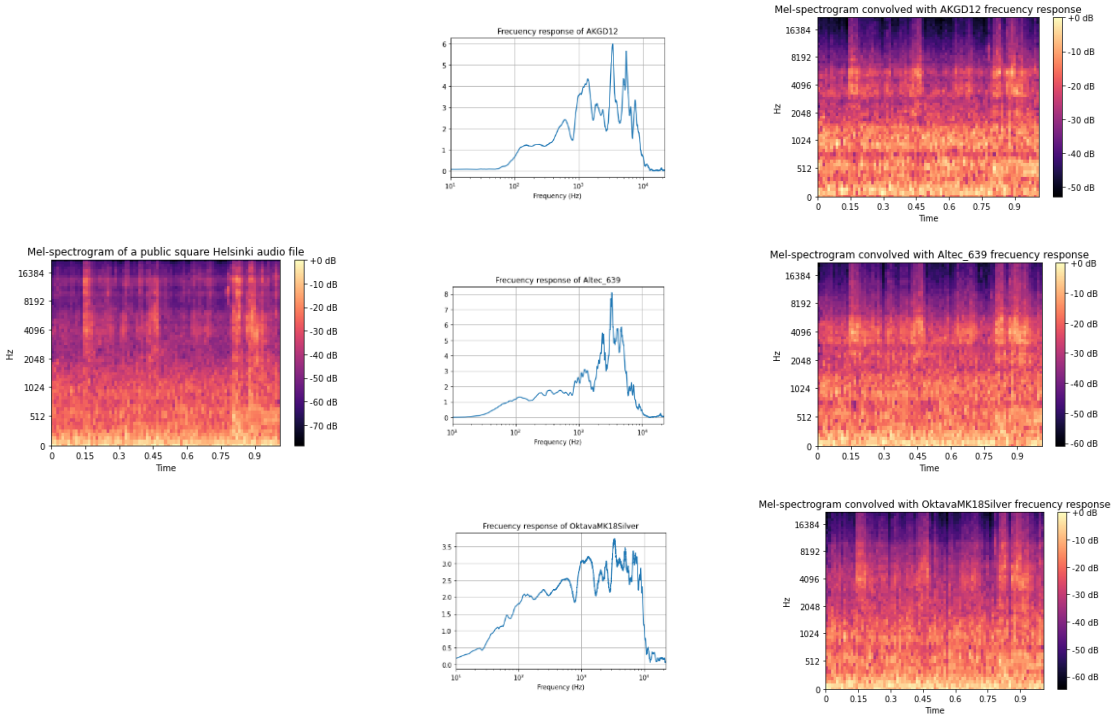

Figura 17: Resultado de convolución entre audio original y respuesta al impulso de micrófonos reales. Fuente: propia

En concreto, se han utilizado 8 micrófonos distintos: AKG D12, Altec 639, American R331, Astatic 77, Beyer M260, FosterDynamic DF1, Oktava MK18 Silver y Sony C37Fet. Para cada observación del conjunto de datos de aumento se ha calculado la convolución entre el audio

y la respuesta al impulso y se ha comprimido la dinámica para no saturar la salida del sistema. Se han creado 120.000 muestras nuevas (15.000 datos para aumento por 8 respuestas al

impulso). En la Figura 18, se muestra la transformación del espectrograma Mel entre un audio original y la aplicación de las respuestas de algunos micrófonos.

4.2.3 Aumento de datos sobre el espectrograma

Para cada uno de los conjuntos propuestos en la Tabla 2 se han generado los subconjuntos con aumento sobre el espectrograma. Siguiendo el trabajo propuesto por Park et al., se ha aplicado una política de aumento LD; aplicando 2 enmascaramientos en distintos rangos de frecuencia y 2 enmascaramientos en distintas ventanas de tiempo. Estos rangos: [t0, t0+t] y [f0, f0+f], se han decidió de manera aleatoria para cada ejemplo procesado.

Etapa	Capa	Filtros	Kernel/Pool Size
1	Input	-	-
2	Conv2D BatchNorm Activation Relu	16 - -	7 - -
3	Conv2D BatchNorm Activation Relu MaxPoll2D Dropout (0,3)	16 - - - -	7 - - (5,5) -
4	Conv2D BatchNorm Activation Relu MaxPoll2D Dropout (0,3)	32 - - - -	7 - - (4,10) -
5	Flatten Dense (100) Dropout (0,3) Dense (10)	- - - -	- - - -

Tabla 3: Arquitectura de la red CNN de referencia.

4.3 Modelo de referencia

El sistema de referencia implementa un enfoque basado en una red neuronal convolucional (CNN) que utiliza energías de banda mel logarítmica extraídas para cada señal de 1 segundo. En concreto: 40 filtros Mel, con una ventana de 40 ms y un solapamiento de 20 ms, procesado a una frecuencia de muestreo de 44100 Hz. Representación en espectrograma Mel de 40x51 píxeles.

La red consta de tres capas CNN y una capa totalmente conectada para asignar etiquetas de escena a las señales de audio; como se muestra en la Tabla 4.

El entrenamiento se ha realizado durante 200 épocas con lotes de 16 ejemplos. Se ha utilizado la función de optimización Adam (con una tasa de aprendizaje del 0,001) y la función de coste: *categorical-crossentropy*. El resultado (como se menciona en los objetivos, a superar) es de *log los*: 1,575 y *Accuracy:* 42,9%. Después de la cuantificación con TFlite, el número de parámetros y millones de MACS de este modelo es de 46.512 y 29,23 respectivamente.

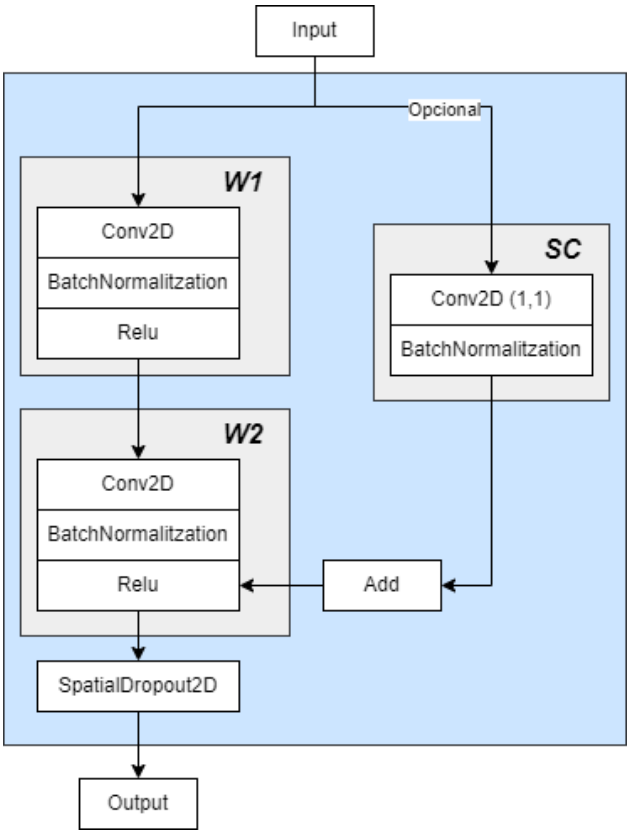

Figura 18: Bloque Residual de la red maestra. Fuente: propia

4.4 Propuesta de modelado propio

Se propone una estrategia “maestro-alumno” basada en la destilación de conocimiento. Para la red maestra se emplea una adaptación de ResNet siguiendo la línea propuesta por Kaiming He et al. (He et al., 2015). La red maestra consta de 5 etapas constituidas por distintos bloques residuales y capas MaxPool. La red alumno es una red lineal CNN, también con 5 etapas, emulando la arquitectura de la red maestro.

Debido a las limitaciones de la GPU utilizada para entrenar (ver apartado 4.6 Entorno de desarrollo), la ResNet implementada es una versión más sencilla que las propuestas en el trabajo original.

Primero, se ha definido un bloque residual (Figura 19), que se reutilizara en las distintas etapas. Consta de 2 capas convolucionales seguidas de capas de normalización por *batch* y de activación Relu. Se incluye la SC (capa de conexión directa) de forma opcional y una capa SpatialDropout2D en la salida; esta versión de *dropout* descarta mapas de características 2D completos en lugar de elementos individuales (Tompson et al., 2015).

Etapa	Capa	Filtros	Kernel / Pool Size
1	Input Conv2D BatchNormalization	- 128 -	- 5 -
2	Bloque 2.1 MaxPool2D Bloque 2.2 Bloque 2.3 Bloque 2.4	128 - 128 128 128	W1 = 3, W2 = 1 (2,2) W1 = 3, W2 = 3 W1 = 3, W2 = 3 W1 = 3, W2 = 3
3	Bloque SC 3.1 Bloque 3.2 Bloque 3.3 Bloque 3.4	256 256 256 256	W1 = 3, W2 = 1 W1 = 1, W2 = 1 W1 = 1, W2 = 1 W1 = 1, W2 = 1
4	Bloque SC 4.1 Bloque 4.2 Bloque 4.3 Bloque 4.4	512 512 512 512	W1 = 1, W2 = 1 W1 = 1, W2 = 1 W1 = 1, W2 = 1 W1 = 1, W2 = 1
5	Conv2D BatchNormalization GlobalAveragePooling2D Dense (10)	10 - - -	1 - - -

Tabla 4: Arquitectura red maestra propuesta

Se ha etiquetado como W1 y W2 (haciendo referencia al bloque residual de la figura 9) las capas con pesos y SC (*ShortCut*) la capa de conexión directa. Para W1 y W2 se pueden configurar el *kernel* por separado. SC siempre tendrá un *kernel* identidad (es decir de 1), y el número de pasos del filtro (*strides*) siempre será de un píxel. El resto de las configuraciones se muestran en a la Tabla 5. Los bloques residuales; que contienen la capa SC, se han nombrado “Bloque SC”. El resto de los bloques no contienen capa SC. El número de parámetros asciende a 3.980.960.

La red alumno, también de 5 etapas, mantiene la entrada y salida con la misma configuración, exceptuando la capa convolucional de la entrada, en la cual se ha reducido el número de filtros. Las etapas 2, 3 y 4 siguen la estructura de un bloque residual sin SC y, las etapas 2 y 3 están precedidas de una capa MaxPool. Este modelo tiene un total de 57.856 parámetros. En la Tabal 6 se resume la arquitectura del modelo alumno.

Etapa	Capa	Filtros	Kernel/Pool Size
1	Input Conv2D BatchNormalization	- 32 -	- (6,5) -
2	MaxPoll2D Conv2D BatchNormalization Conv2D BatchNormalization	- 32 - 32 -	(2,2) 3 - 3 -
3	MaxPoll2D Conv2D BatchNormalization Conv2D BatchNormalization	- 32 - 32 -	(5,2) 3 - 3 -
4	Conv2D BatchNormalization Conv2D BatchNormalization	32 - 32 -	3 - 3 -
5	Conv2D BatchNormalization GlobalAveragePooling2D Dense (10)	10 - - -	1 - - -

Tabla 5: Arquitectura red alumno propuesta

4.5 Métricas de evaluación

Aunque el objetivo principal, en el que se basa este desarrollo, es superar la capacidad de generalización de la red (medida en términos de *log loss*), se añaden dos requisitos relacionados con la complejidad computacional para conseguir una solución apropiada.

4.5.1 Rendimiento y evaluación

El rendimiento se medirá por la capacidad de generalización de la red. Esto implica minimizar la función de perdida *cross-entropy* (o, *Log Loss*).

Esta es la función de pérdida se definida como la verosimilitud logarítmica negativa de un modelo logístico que devuelve probabilidades (predicción sobre datos de prueba) para sus datos de entrenamiento (datos de prueba verdaderos).

Esta función está implementada en la librería *sklearn* como *log_loss*, y se describe con la siguiente ecuación:

$$L_{\log}(y,p) = -(y\log(p) + (1-y)\log(1-p))$$

4.5.2 Complejidad computacional

La complejidad computacional se medirá en términos de recuento de parámetros y MMACs (millones de operaciones de multiplicación-acumulación). Los límites están modelados según los dispositivos Cortex-M4 (por ejemplo, STM32L496@80MHz o Arduino Nano 33@64MHz); los límites máximos permitidos son los siguientes:

1. **Número máximo de parámetros 128K**, y el tipo de variable utilizada se fija en INT8, contando todos los parámetros (incluyendo los de valor cero).

2. **Número máximo de MACS por inferencia: 30 MMAC (millones de MACs).** El límite se aproxima en función de la potencia de cálculo de la clase de dispositivo objetivo. La longitud del segmento de análisis para la inferencia es de 1 s. El límite de 30 MMAC imita el encaje de los búferes de audio en la SRAM (memoria interna de acceso rápido) del dispositivo de destino y permite cierto espacio de cabeza para el cálculo de características (por ejemplo, la FFT), suponiendo que las características más utilizadas caben en ella.

4.6 Entorno de desarrollo

En un principio se ha querido utilizar el equipo personal básico para el desarrollo. Pero, esto no ha sido posible por las limitaciones en cuanto a GPU. Como segunda opción se ha probado Google Colab básico compartiendo los ficheros por Google Drive. Pero, el tamaño del conjunto de datos original más el conjunto de audio aumentados es de aproximadamente 40 GB (archivos de audio en formato *.wav*), así que la limitación de Google Drive (10 GB) excluye esta posibilidad.

Finalmente se ha decidió emplear el entorno local Spyder IDE 5.1.5 para procesar los datos, convertirlos a espectrogramas Mel (imágenes) y guardarlos en formato *pickle*. En este caso, para el procesado 4 (el más voluminoso) el tamaño total es de 8,60 GB; suficiente para compartirlo vía Drive.

Con esto, se ha podido utilizar Google Colab. Pero, aun definiendo funciones que leen de uno en uno los ficheros *pkl* y, concatenan el contenido a la variable correspondiente (con lo que

se optimiza la RAM del entorno), al cargar los datos de aumento se cerraba la sesión. Por lo que ha sido necesario obtener la versión de pago Google Colab Pro.

Llegados a este punto se ha podido trabajar de forma (más o menos) cómoda. Guardando siempre en carpetas compartidas de Drive los puntos de control después de cada época de entrenamiento; ya que al cabo de unas horas la sesión de Colab se cierra por inactividad.

4.6.1 Librerías utilizadas

En este apartado se citan las librerías empleadas para el desarrollo de cada tarea listada en el apartado 3.3 Metodología de trabajo.

- **Análisis de los datos:** pandas, numpy, matplotlib.
- **Procesamiento de audio y representación del espectrograma Mel**: librosa, pickle
- **Aplicación de las técnicas de aumento:** waves, soundfile.
- **Recreación y entrenamiento del modelo de referencia**: tensorflow, keras, sklearn
- **Propuestas de modelos en base a los estudiados**: tensorflow, keras, sklearn
- **Sesiones de entrenamiento y validación**: tensorflow, keras, sklearn
- **Refinamiento del modelo propuesto**: tensorflow, keras, sklearn
- **Entrenamiento del modelo alumno**: tensorflow, keras, sklearn
- **Aplicación de técnicas de destilación:** código inspirado en (*Knowledge Distillation*, 2022)
- **Cuantificación**: tflite
- **Métricas:** sklearn, tflite, nessi (*AlbertoAncilotto/NeSsi: Keras/Pytorch Neural Network Size, Operations and Parameters Counter*, 2022)

4.7 Propuestas descartadas

Como se ha descrito en el capítulo 3 Objetivos y metodología de trabajo, las fases de desarrollo han sido graduales. Por lo que algunos modelos propuestos al inicio se han descartado por distintas razones.

Una primera propuesta ha sido implementar un *Vision Transformer* (ViT). El desarrollo se ha conseguido con éxito. Aunque, la necesidad de reducir el número de parámetros ha degradado el rendimiento. Se ha conseguido un *loss* = 1,725 con un "*encoder"* de 4 capas, cada una de ellas de 4 capas atencionales "*multi-head* ". Se ha aplicado poda en las últimas capas densas (de 40 y 20 nodos cada una), permitiendo reducir el número de parámetros. Aun así, la cantidad de MMACS supera lo permitido. Esto implicaría reducir más aún la red y, sumado a que no cumple con el objetivo principal se ha descartado.

5 Desarrollo de la comparativa

En este capítulo se describe el proceso que se ha seguido para encontrar un modelo que consiga el objetivo principal de este estudio. Además, se compara los resultados obtenidos en función del procesado del audio y representación del espectrograma Mel. Finalmente, se compara el resultado obtenido después de la optimización y compresión del modelo alumno con el modelo de referencia.

De ahora en adelante, se hará referencia a la Tabla 3: Resumen del conjunto de representaciones del audio para denotar los conjuntos empleados en los experimentos realizados.

5.1 Proceso de entrenamiento del modelo maestro

La arquitectura de la red maestra, propuesta en el capítulo anterior, es la culminación de varios experimentos realizados. En este capítulo se enumeran los pasos seguidos para llegar a ese modelo. Para buscar una configuración idónea, se parte de los datos procesados “base”; con la intención de encontrar un modelo maestro que supere al modelo de base sin alterar el procesado los datos de entrada utilizados como referencia.

Para el entrenamiento (a no ser que se puntualice lo contrario) se ha dividido el conjunto de datos base en entrenamiento (70%) y validación (30%). El valor de *loss* servirá de comparativa (pues es el valor para tener en cuenta para alcanzar el objetivo principal)

A continuación, se enumeran los pasos seguidos:

- Configuración de capas y bloques
- Funciones de optimización y número de épocas de entrenamiento
- Número de lotes.
- Funciones de activación

A. Configuración de capas y bloques

Después de varios intentos de reproducir modelos ResNet idénticos a los propuestos en (He et al., 2015) , se ha visto que no era asimilable la cantidad de parámetros debido al coste temporal y de RAM que implica; sumado a la limitación del entorno de entrenamiento Google Colab Pro. Por lo que se han ido eliminando bloques en distintas etapas y reduciendo la

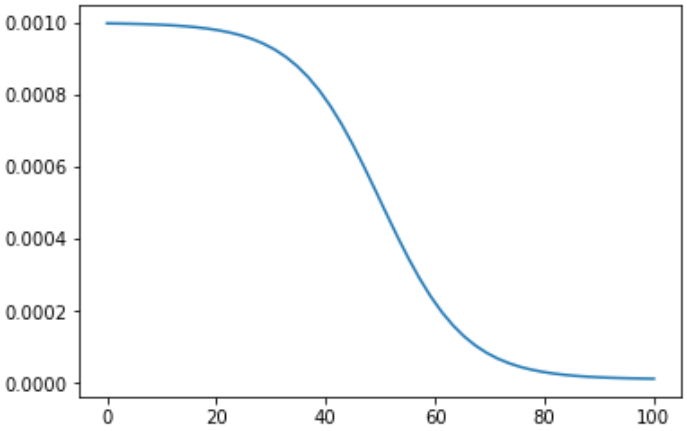

Figura 19: Función de optimización sigmoide

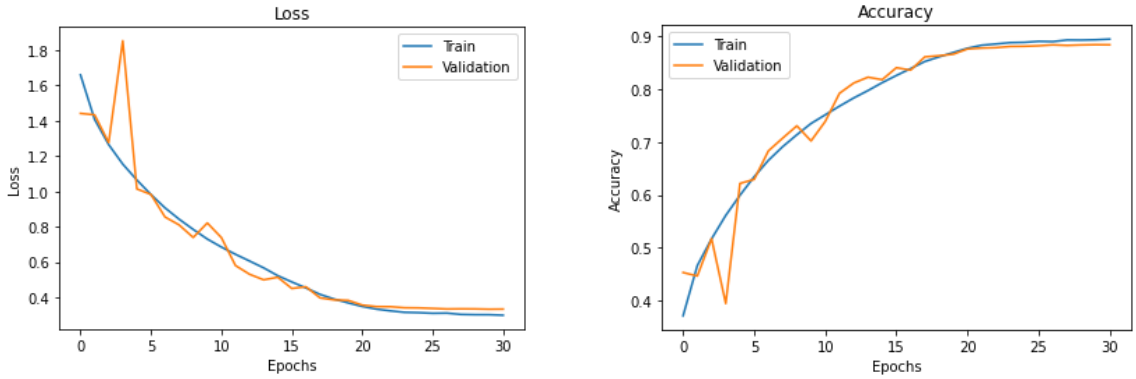

Figura 20: Histórico de loss y accuracy para el modelo propuesto. Fuente: propia

cantidad de filtros en las capas convolucionales hasta conseguir un modelo asumible en cuanto a recursos.

B. Funciones de optimización y número de épocas de entrenamiento

Empleando la función de optimización Adam con una tasa de aprendizaje de 0,001 (como en la propuesta base), la red no se estabiliza correctamente. Por lo que se ha estudiado otras técnicas basadas es aprendizajes programados. En este caso se ha empleado un aprendizaje en base a la función sigmoide (Macêdo et al., 2021) (Figura 20). La idea es dividir la convergencia en 3 fases: (1) encontrar rápidamente el mínimo local donde converger después de pasar por distintos puntos de silla (tasa de aprendizaje elevada), (2) una vez allí reducir progresivamente la tasa para converger en el centro de ese mínimo y, (3) reducir rápidamente la tasa a valores muy bajos y mantenerse en esos valores para estabilizarse en el punto crítico.

datos	Tamaño de lote	épocas	loss
base	32	30	1,906
		10	1,720
		5	**1,682**
		3	1,702

Tabla 6: Comparación de entrenamiento por número de épocas

datos	épocas	Tamaño de lote	loss
base	5	16	1,685
		32	1,682
		64	**1,638**

Tabla 7: Comparación de entrenamiento por número de ejemplos por lote

La cantidad de parámetros (casi 4 millones) respecto al número de observaciones del conjunto para entrenamiento (93545, después de dividir en 70/30 para validación) repercute en que el modelo aprenda de los datos muy rápidamente y lleve al sobreajuste después de varias épocas; la Figura 21 muestra el histórico de un entrenamiento con más de 30 épocas:

En este caso se percibe la rapidez de convergencia de la red y se constata con los resultados obtenidos (Tabla 7), donde los mejores valores de *loss* se consiguen con tan solo 5 épocas.

Por el momento se ha fijado el número de épocas a 5. Siendo susceptible de cambiar según sigan los experimentos. Por ejemplo, añadir los datos de aumento incrementaría la cantidad de datos y podría implicar la necesidad de aumentar el número de épocas para conseguir esta convergencia.

C. Número de ejemplos por lote

Se han realizado distintos experimentos que se resumen en la Tabla 8.

D. Funciones de activación

Dejando fijados los mejores valores se ha procedido a comparar en entrenamiento con distintas funciones de activación de las capas W1 y W2 de cada bloque residual (Tabla 9).

datos	épocas	Tamaño de lote	Activación	loss
base	5	64	Relu	1,638
			Elu	**1,556**
			Swish	1,653

Tabla 8: Comparación de entrenamiento por función de activación

datos	Resumen de procesado (frec., ventana, mels), imagen (mxn)	épocas	loss
base	(44100 Hz, 40 ms, 40) (40x51)	5	1,556
Procesado_1	(44100 Hz, 40 ms, 80) (80x51)	5	1,562
		8	**1,477**
		11	1,484
Procesado_2	(22050 Hz, 46 ms, 40) (40x44)	5	1,621
Procesado_3	(44100 Hz, 23 ms, 40) (40x87)	5	1,639
		8	1,642
Procesado_4	(44100 Hz, 23 ms, 80) (80x87)	5	1,534
		8	1,541
		11	1,531

Tabla 9: Comparación de entrenamiento por función del procesado de los datos

Las funciones elegidas son Relu (Fred Agarap, 2019), Elu (Clevert et al., 2015) y Swish (Ramachandran et al., 2017), dado que las 3 han sido empleadas en clasificación de imágenes. La función Relu, es la que se ha utilizado en los entrenamientos hasta ahora.

5.2 Comparativa en función del procesado de los datos

Llegados a este punto, la red maestra entrenada ya generaliza mejor que el modelo de referencia. Aun así, falta destilarlo y comprimirlo. Tampoco, se han empleado el resto de los conjuntos procesados ni, tampoco, el conjunto de aumento; con lo que se pretende mejorar la solución.

En este apartado se muestran (Tabla 10) los resultados obtenidos entrenando la red con los distintos conjuntos de la Tabla 3 y con la configuración del apartado anterior (64 ejemplos por lote y activación Elu), modificando el número de épocas; algunos de los procesados contienen más cantidad de datos por imagen lo que implica que la convergencia puede ser más lenta.

La elección de los distintos procesados no es arbitraria. Se pretende obtener una visión de cómo afecta:

(1. Base contra Procesado_2) la frecuencia de muestreo sin modificar el tiempo de ventana; en la mediad de lo posible según el número de muestras de la FFT. Tampoco se modifica el número de filtros Mel.
(2. Base contra Procesado_1) el número de filtros Mel, sin modificar nada más.

(3. Base contra Procesado_3, y Procesado_1 contra Procesado_4) la duración de la ventana en relación con el número de filtros sin modificar la frecuencia de muestreo.

5.3 Comparativa con aumento de datos

Para este paso del desarrollo se ha comparado los distintos aumentos de datos empleando el conjunto del apartado anterior que mejor resultado obtiene. Los datos para este experimento se han agrupado de la siguiente forma:

(1) Datos base con espectrograma aumentado. Se sustituye los datos base por los datos aumentados tal y como figura en el apartado 4.3.2 Aumento de datos sobre el espectrograma. La división de entrenamiento-validación es de 70-30, respectivamente (como hasta ahora).

(2) Datos base más datos aumentados con el método propuesto en el apartado 4.3.1 Aumento de datos basado en la respuesta al impulso de dispositivos reales no observados. Se ha separado el conjunto base en 70-30 para entrenamiento y validación respectivamente. Para el conjunto aumentado se ha seguido este mismo proceso. Finalmente se han concatenado y mezclado aleatoriamente los datos de entrenamiento de cada conjunto y, también, los datos de validación. En este caso existen datos de aumento y base en la misma proporción en el conjunto de entrenamiento y en el de validación.

(3) Datos base más datos aumentados con el método propuesto en el apartado 4.3.1 Aumento de datos basado en la respuesta al impulso de dispositivos reales no observados. El 70% datos aumento se ha utilizado para validación (84.000 observaciones), sin añadir, en este, datos base. Para el entrenamiento se utilizan todos los datos base más el 30 % de datos aumentados restantes (un total de 175.620 observaciones) mezclados de forma aleatoria. La idea es que la red generalice lo mejor posible, por ello, en la validación solo se añaden datos aumentados.

La Tabla 11 resume los resultados obtenidos con cada uno de estas distribuciones en los datos.

Datos aumento	épocas	loss
(1)	5	1,710
	8	1,683
	11	1,641
(2)	5	1,499
	8	1,463
	11	1,486
(3)	5	1,468
	8	**1,432**
	11	1.484

Tabla 10: Comparación de entrenamiento con aumento de datos

Temperatura (T)	épocas	loss
2	6	2,604
	10	1,545
	22	**1.416**
3	6	2,238
	10	2,043
	22	1,558
4	6	4,000
	10	2,073
	22	1,616
5	6	2,670
	10	1,798
	22	1,609

Tabla 11: Resultados de la destilación en función de la temperatura

5.4 Destilación y compresión por cuantificación INT8

Para terminar el desarrollo de la comparativa, es necesario destilar el conocimiento del modelo maestro al modelo alumno y cuantificarlo al tipo de dato INT8.

loss	accuracy	Número de parámetros	MMACs
1,415	48.73%	57.660	26,116

Tabla 12: Métricas finales del sistema propuesto

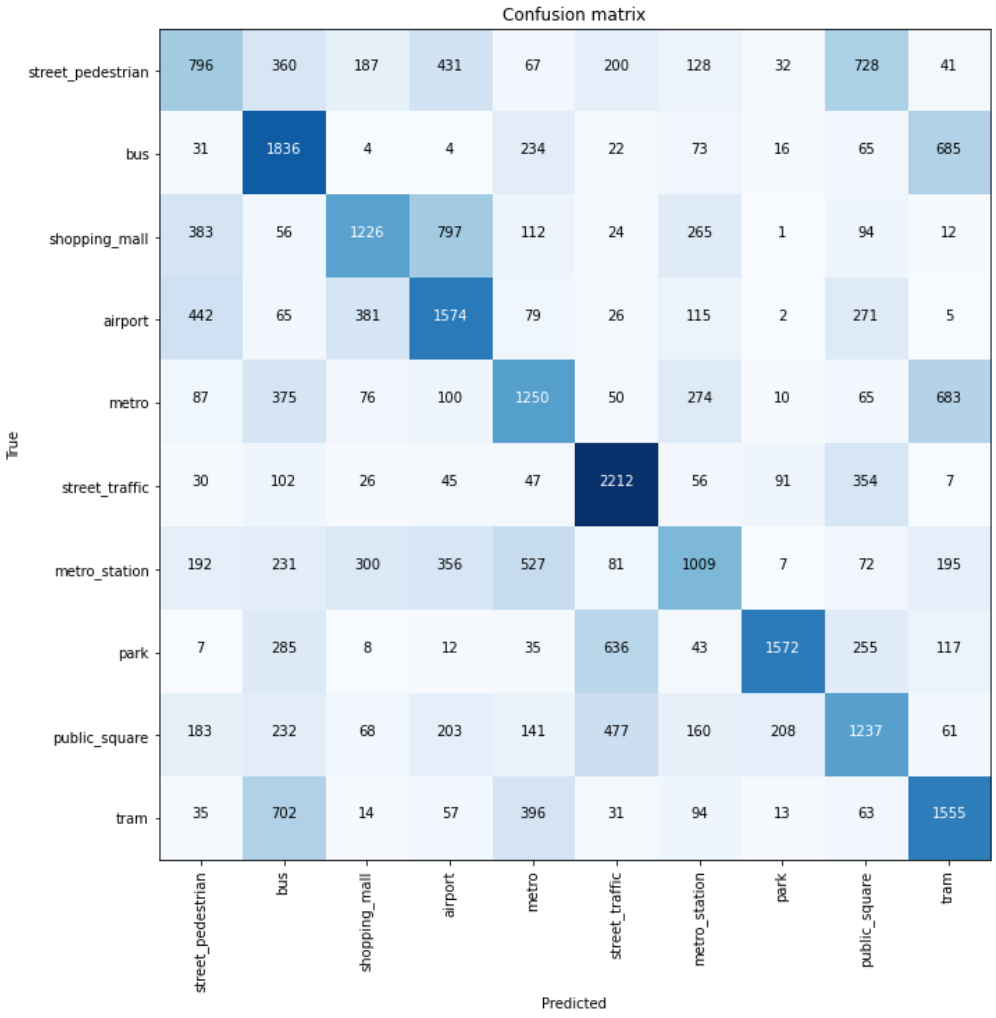

Figura 21: Matriz de confusión del modelo propuesto. Fuente: propia

Para la destilación se ha seguido el esquema de entrenamiento estudiado (ver Figura 11: Proceso de KD basado en la propuesta de Hinton et al.). El modelo alumno se ha entrenado con varios valores de temperatura (T) y número de épocas. Los resultados se muestran en la Tabla 12.

En este caso el modelo maestro se ha compilado obteniendo una salida con *logits*; sin función de activación *softmax* y añadiendo el parámetro correspondiente en la función de coste ("*from_logits = True"*). Después, el modelo alumno se ha compilado de esta misma forma para el proceso de destilación.

Por último, los pesos del mejor entrenamiento se han cargado sobre un modelo alumno. Para terminar, se ha cuantificado este modelo para reducir el tamaño y obtener las métricas de rendimiento y complejidad que se muestran en la Tabla 13.

Se añade la matriz de confusión generada tras la predicción sobre los datos de prueba por parte del modelo final propuesto (Figura 22).

6 Discusión de los resultados

En este apartado se discuten los resultados obtenidos durante el entrenamiento y experimentación con datos realizado en el capítulo anterior; el cual, se ha dividido en 3 fases:

- modelar una red y entrenarla con el mismo procesamiento de datos que el proyecto de referencia. Con la finalidad de obtener un mejor rendimiento que el del modelo base.

- Experimentar con los distintos procesados del audio y aumento de los datos con la intención de mejorar el resultado anterior.

- Destilar y comprimir el modelo para obtener el resultado final y compararlo con el modelo de referencia respecto al objetivo principal.

Cada una de estas 3 fases se analizará a continuación.

6.1 Análisis del modelo propuesto

Después de estudiar los últimos avances y técnicas aplicadas a la ASC con modelos de baja complejidad, se ha diseñado una red convolucional basada en bloques residuales que será destilada a una red convolucional lineal, que posteriormente se cuantificará valores de 8 bits.

Para el desarrollo del modelo maestro ha habido una ilimitación impuesta por el entorno de desarrollo y no se ha podido modelar una red con más parámetros o mayor número de filtros en las capas convolucionales. Sin embargo, el modelo propuesto tiene casi 4 millones de parámetros. Una cantidad suficiente como para aprender los patrones de las observaciones muy rápidamente.

De hecho, este es el primer punto para analizar. Con la cantidad de datos del conjunto base, respecto a la cantidad de parámetros y, utilizando la función de optimización Adam con una tasa de aprendizaje de 0,001, la red converge rápidamente, pero de forma muy poco estable. Para pailar este problema se ha incorporado el aprendizaje programado utilizando la función sigmoide. El resultado ha mejorado considerablemente en cuanto a estabilidad y velocidad de convergencia, pudiendo analizar el comportamiento fácilmente.

Lo primero que se ha visto es lo rápido que converge la red propuesta (en 5 épocas obtiene un *loss* de 1,683) respecto al modelo de referencia (200 épocas, 1,575 de *loss*). Esto

demuestra la capacidad del modelo de aprender muy bien los patrones. Pero, esta facilidad deriva en un sobreajuste (también) demasiado rápido. Por esta razón, después de 5 épocas, el modelo obtiene peores resultados. Este sobreajuste se tratará añadiendo datos con más información y aumentando los datos de audio con distintas técnicas.

Esto se puede corroborar con los dos experimentos siguientes: número de ejemplos por lote y función de activación.

Un número pequeño de ejemplos en cada iteración implica mejor capacidad de aprendizaje y una convergencia más rápida. Con el mismo número de épocas y distinto valor de tamaño del lote, se obtiene una capacidad de generalización mayor con 64 ejemplos por lote que 32.

Por otro lado, en lo referente a la función de activación, aunque tanto Relu como Swich son más novedosas y sus autores han validado su mejor desempeño frente a Elu (entre otras) con redes muy profundas y gran cantidad de datos como ImageNet; para nuestro caso, Elu consigue un resultado considerablemente mejor; un *loss* de 1,556 frente a 1,638 y 1,653 para Relu y Swish respectivamente.

Una posible explicación es que el modelo necesita “aprender menos” de los ejemplos de entrenamiento y, añadir más ejemplos por lote o funciones de activación que en realidad frenan la velocidad de convergencia, ayuda en este aspecto.

6.2 Análisis del procesado de audio

Cuando representamos el audio digital procesado (en función de la frecuencia de muestreo, número de muestras de la FFT y número de bandas Mel) en forma de espectrograma de Mel, estamos convirtiendo esa información en un conjunto de píxeles con unos valores determinados. Al entrenar la red con esos valores como inputs y filtrarlos con las matrices convolucionales, estamos obteniendo información sobre cuán importante es la relación frecuencia-tiempo para clasificar correctamente un ejemplo dado.

Los experimentos realizados sobre los distintos procesados de audio y representación de los espectrogramas de Mel aportan información interesante acera de las características para tener en cuenta sobre el audio para la clasificación es escenas acústicas.

En el apartado 5.2 Comparativa en función del procesado de los datos se enumeran las comparativas que se pueden obtener en base a los resultados obtenidos, anotados en la Tabla 10.

Si comparamos el resultado del conjunto base con el Procesado_1, se aprecia como afecta la elección del número de filtros Mel. Un valor de Mels mayor implica una resolución frecuencial más grande (cada banda contiene un rango frecuencial menor); hay más granularidad en la información frecuencial. Los resultados muestran que añadir el doble de bandas Mel ayuda a mejorar la generalización de la red. Esto concluye en que la relación de cambios de frecuencia de las escenas acústicas es relevante para la clasificación.

Comparando el procesado base con el Procesado_2, donde el único cambio es la frecuencia de muestreo (Fm), podemos determinar que una resolución mayor de muestras para caracterizar el audio es mejor. Como se ha explicado, según Nyquist la Fm debe ser el doble de la frecuencia de interés. Por lo tanto, con una Fm=22050 Hz se está obviando la información a partir de, aproximadamente, 11.000 Hz hasta los, aproximadamente, 22.000 Hz que se procesan con una Fm=44100 Hz. La conclusión es que la información que aportan las frecuencias altas, en las escenas acústicas captadas, es de interés para los sistemas ASC.

Aunque, hay que señalar que el procesado con 1024 muestras de la FFT para una ventana de 46ms y una Fm de 22050 Hz, añade 10 muestras a cero. Lo mismo sucede con los Procesado_3 y Procesado_4. En estos dos últimos casos, la intención es comparar cuanto afecta la resolución temporal. Cruzando los resultados entre base y Procesado_3, y Procesado_1 y Procesado_4, se corrobora que una ventana de 40ms funciona mejor que con una ventana de 23 ms. Por lo tanto, la granularidad a nivel temporal no aporta más información, si no que parece confundir más al modelo. O, añadir ceros en la FFT afecta de forma desfavorable a la representación del espectrograma.

El mejor resultado se consigue con el Procesado_1. Lo que ratifica las conclusiones anteriores: (1) la resolución frecuencial es importante, (2) las frecuencias altas aportan información útil para la ASC y, (3) la resolución temporal demasiado baja, o un número de muestras de la FFT no ajustado (exactamente) a una potencia de 2, degradan el resultando del entrenamiento.

6.3 Análisis del aumento

Se han aplicado dos métodos para el aumento de datos. En el apartado 5.3 Comparativa con aumento de datos, se muestran los resultados obtenidos, anotados en la Tabla 11.

El primer método, estudiado en el estado del arte (SpecAugment), no ha dado los resultados esperados. En la literatura se confirma como un método que funciona en la mayoría de las propuestas de sistemas ASC. El hecho de no obtener esos resultados puede residir en que la

política aplicada no sea apropiada (ver 4.2.3 Aumento de datos sobre el espectrograma) o, en la forma en la que se ha aplicado el método. Normalmente, se aplica como una capa extra en el propio modelo. Pero, en este trabajo se ha procesado aparte para obtener el conjunto y realizar los experimentos de forma individual. Este hecho se apunta como posible mejora en el siguiente capítulo.

La segunda propuesta de aumento es un método propio inspirado en la recreación sintética de los datos de dispositivos no observados; propios del conjunto original. Esta aportación ha mejorado la capacidad de generalización del modelo. Para validar la bondad de la aportación, se han realizado 2 experimentos distintos según la distribución de los datos entre entrenamiento y validación (en la tabla 11 se enumeran como 2 y 3).

El mejor resultado se consigue con la configuración 3; donde el conjunto de validación solo contiene datos aumentados (70%) y, el de entrenamiento, los datos originales más una parte (30%) de datos aumentados. En este caso, el modelo generaliza mejor debido a que está aprendido características de los espectrogramas que no están tan relacionadas con la respuesta frecuencial del dispositivo; tal como se había previsto.

6.4 Análisis de rendimiento y complejidad

La Tabla 14 muestra la comparativa final entre el modelo de referencia y la propuesta desarrollada en este trabajo.

El modelo propuesto obtiene mejor valor de *loss*. Lo que confirma que este modelo generaliza mejor que el de referencia. De hecho, al destilar el conocimiento del modelo maestro (*loss* = 1,432) al modelo alumno, observamos que mejora más este valor. La explicación a esto reside en que el modelo alumno debe configurar los pesos de acuerdo con la salida del maestro y, al tener menos parámetros los establece para las características que mejor optimizan la función de coste.

El valor de *accuracy* también es mejor para el modelo propuesto (+ 5,83%). El hecho de generalizar mejor aumenta la exactitud del modelo; ya que es capaz de etiquetar más clases pertenecientes a dispositivos no observados del conjunto de prueba.

El modelo propuesto tiene un total de 11.148 parámetros más que el modelo de referencia. En ambos casos, aun así, queda margen para escalar hasta los 128.000 permitidos.

Por el contrario, el número de MACs del modelo propuesto frente al modelo de referencia es menor; en concreto 3,114 millones menos. Esto se debe al número final de capas densas. El

	loss	accuracy	Número de parámetros	MMACs
Modelo de referencia	1,575	42,90%	**46.512**	29,230
Modelo propuesto	**1,415**	**48.73%**	57.660	**26,116**

Tabla 13: Comparativa de métricas finales entre modelos

modelo de referencia contiene dos capas densas de 100 y 10 nodos, mientras que el modelo propuesto solo una de 10 nodos. El máximo MMACs permitido es de 30 millones.

Si analizamos estos dos últimos valores (número de parámetros y MMACs), observamos que, aunque ambos podrían “crecer” en número de parámetros, el modelo de referencia está limitado por el valor de MMACs; siendo poco probable la opción de escalar el modelo. En cambio, el modelo propuesto tiene cierto margen para la mejora.

Por lo tanto, podemos considerar que el modelo propuesto, aparte de alcanzar el objetivo propuesto, supera el modelo de referencia y, además, es escalable.

7 Conclusiones

En este último capítulo se valora el trabajo realizado y las conclusiones que derivan de este. Se divide en dos secciones. En la primera se analiza el éxito en cuanto al cumplimiento de los objetivos marcados y la metodología utilizada, así como el resto de las implicaciones técnicas y teóricas. Finalmente, se listan algunas de las líneas de trabajo futuro, tanto a nivel práctico como teórico.

7.1 Cumplimiento de los objetivos

En este proyecto se planteó un objetivo principal y un conjunto de objetivos específicos. El primero, era obtener un modelo capaz de generalizar mejor que el modelo de referencia, en la tarea de clasificación de escenas acústicas; además, añadiendo el requisito de cumplir con limitaciones computacionales. Se ha conseguido el objetivo y se ha mejorado en exactitud y se ha reducido la cantidad de MMACS. Para ello, los objetivos específicos también se han logrado siguiendo la metodología explicada. Además, durante el desarrollo del proyecto, se ha propuesto una metodología propia para aumento de datos de audio que ha permitido mejorar los resultados iniciales.

Por otro lado, algunos de los riesgos planteados después del estudio de la temática (ver 2.3 Conclusiones sobre el estado del arte) se han podido solventar de forma adecuada. En este aspecto, la literatura que ha llevado más tiempo comprender ha sido la relacionada con la compresión de redes profundas.

A nivel técnico, el problema más complejo de solventar ha sido la limitación de prestaciones de equipamiento. Lo que pone de manifiesto la dificultad técnica y económica que conllevan este tipo de proyectos a mayor escala. Para solucionarlo, se ha gestionado el desarrollo del software entre el equipo personal y el servicio de Google Colab. Esto, ha añadido complicaciones y retrasos en la planificación inicial.

7.2 Líneas de trabajo futuro

Existen muchas opciones para la continuidad de este trabajo. A nivel de investigación se puede profundizar en cada una de las disciplinas que convergen para dar solución al problema planteado. Por ejemplo:

- El procesamiento del audio y como los dispositivos afectan a la captación para mejorar la capacidad de generalizar.

- Estudiar y mejorar el método propuesto de aumento de audio. Por ejemplo, añadir respuestas frecuenciales nuevas.

- Explorar la capacidad de los modelos basados en Transformers.

- Investigar a fondo los métodos de compresión y optimización de redes profundas

A nivel técnico, se identifican una serie de mejoras para el proyecto:

- Estudiar la distribución del conjunto de datos para entrenamiento y validación. Por ejemplo, distribuir por ubicación o localización de la captura del audio, puesto que esta característica podría afectar a la capacidad de generalizar.

- Con equipos con mejores prestaciones, se puede incrementar el número de filtros Mel y la cantidad de muestras por ventana; esto permitiría generar y valorar más resultados.

- También, con acceso a más y mejores GPUs, se podrán entrenar modelos más grandes y añadir más cantidad de datos aumentados con otras técnicas.

Por último, una línea de trabajo muy interesante sería el despliegue de este modelo en un microcontrolador o dispositivo portable. Permitiendo aplicar la solución en un entorno real y, en consecuencia, analizar el trabajo desde otra perspectiva.

8 Bibliografía

Abeßer, J. (2020). A review of deep learning based methods for acoustic scene classification. *Applied Sciences (Switzerland)*, *10*(6). https://doi.org/10.3390/app10062020

AlbertoAncilotto/NeSsi: Keras/Pytorch neural network size, operations and parameters counter. (2022). https://github.com/AlbertoAncilotto/NeSsi

Bae, S. H., Choi, I., & Kim, N. S. (2016). *Detection and Classification of Acoustic Scenes and Events*.

Baevski, A., Zhou, H., Mohamed, A., & Auli, M. (2020). *wav2vec 2.0: A Framework for Self-Supervised Learning of Speech Representations*. https://github.com/pytorch/fairseq

Barchiesi, D., Giannoulis, D., Stowell, D., & Plumbley, M. D. (2014). *Acoustic Scene Classification*.

Battaglino, D., Lepauloux, L., & Evans, N. (2016). *Detection and Classification of Acoustic Scenes and Events*.

Blauert, J. (2013). The technology of binaural listening. En *The Technology of Binaural Listening*. Springer Berlin Heidelberg. https://doi.org/10.1007/978-3-642-37762-4

Bregman, A. S., & McAdams, S. (1994). Auditory Scene Analysis: The Perceptual Organization of Sound . *The Journal of the Acoustical Society of America*, *95*(2), 1177-1178. https://doi.org/10.1121/1.408434

Chung, Y.-A., Gong, Y., & Glass, J. (2021). *AST: Audio Spectrogram Transformer*. https://github.com/YuanGongND/ast.

Cires¸an, D. C. C., Meier, U., Masci, J., Gambardella, L. M., & Urgen Schmidhuber, J. ¨. (2012). *Flexible, High Performance Convolutional Neural Networks for Image Classification*.

Clevert, D.-A., Unterthiner, T., & Hochreiter, S. (2015). *Fast and Accurate Deep Network Learning by Exponential Linear Units (ELUs)*. http://arxiv.org/abs/1511.07289

David, R., Duke, J., Jain, A., Reddi, V. J., Jeffries, N., Li, J., Kreeger, N., Nappier, I., Natraj, M., Regev, S., Rhodes, R., Wang, T., & Warden, P. (2020). *TensorFlow Lite Micro: Embedded Machine Learning on TinyML Systems*. http://arxiv.org/abs/2010.08678

Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S., Uszkoreit, J., & Houlsby, N. (2021). *AN IMAGE*

IS WORTH 16X16 WORDS: TRANSFORMERS FOR IMAGE RECOGNITION AT SCALE. https://github.com/

Fred Agarap, A. M. (2019). *Deep Learning using Rectified Linear Units (ReLU)*. https://github.com/AFAgarap/relu-classifier.

Gemmeke, J. F., Ellis, D. P. W., Freedman, D., Jansen, A., Lawrence, W., Channing Moore, R., Plakal, M., & Ritter, M. (2017). *AUDIO SET: AN ONTOLOGY AND HUMAN-LABELED DATASET FOR AUDIO EVENTS*.

Gholami, A., Kim, S., Dong, Z., Yao, Z., Mahoney, M. W., & Keutzer, K. (2021). *A Survey of Quantization Methods for Efficient Neural Network Inference*. http://arxiv.org/abs/2103.13630

Giannoulis, D., Benetos, E., Stowell, D., Rossignol, M., Lagrange, M., Plumbley, M., & Plumbley, M. D. (2013). *Detection and classification of acoustic scenes and events: An IEEE AASP challenge 2013 IEEE Workshop on Applications of Signal Processing to Audio and Acoustics, New Paltz, NY DETECTION AND CLASSIFICATION OF ACOUSTIC SCENES AND EVENTS: AN IEEE AASP CHALLENGE*. https://doi.org/10.1109/WASPAA.2013.6701819ï

Gou, J., Yu, B., Maybank, S. J., & Tao, D. (2021). *Knowledge Distillation: A Survey*.

He, K., Zhang, X., Ren, S., & Sun, J. (2015). *Deep Residual Learning for Image Recognition*. http://image-net.org/challenges/LSVRC/2015/

Heittola, T., Mesaros, A., & Virtanen, T. (2020). *Acoustic scene classification in DCASE 2020 Challenge: generalization across devices and low complexity solutions*. http://arxiv.org/abs/2005.14623

Hinton, G., Vinyals, O., & Dean, J. (2015). *Distilling the Knowledge in a Neural Network*. http://arxiv.org/abs/1503.02531

Hubel, D. H., & Wiesel, T. N. (1959). 59I RECEPTIVE FIELDS OF SINGLE NEURONES IN THE CAT'S STRIATE CORTEX. En *J. Physiol. (I959)* (Vol. 48).

Knowledge Distillation. (2022). https://keras.io/examples/vision/knowledge_distillation/

Liu, D., Kong, H., Luo, X., Liu, W., & Subramaniam, R. (2020). *Bringing AI To Edge: From Deep Learning's Perspective*. https://doi.org/10.1016/j.neucom.2021.04.141

Low-Complexity Acoustic Scene Classification - DCASE. (2022). https://dcase.community/challenge2022/task-low-complexity-acoustic-scene-classification

Macêdo, D., Dreyer, P., Ludermir, T., Member, S., & Zanchettin, C. (2021). *Training Aware Sigmoidal Optimizer*.

marmoi/dcase2022_task1_baseline: Baseline system for DCASE 2022 task 1. (2022). https://github.com/marmoi/dcase2022_task1_baseline

Martín-Morató, I., Heittola, T., Mesaros, A., & Virtanen, T. (2021). *Detection and Classification of Acoustic Scenes and Events 2021 LOW-COMPLEXITY ACOUSTIC SCENE CLASSIFICATION FOR MULTI-DEVICE AUDIO: ANALYSIS OF DCASE 2021 CHALLENGE SYSTEMS*. http://dcase.community/challenge2021/task-acoustic-scene-

Mehta, S., & Apple, M. R. (2021). *MobileViT*. https://github.com/apple/ml-cvnets.

Mesaros, A., Heittola, T., & Virtanen, T. (2017). *ASSESSMENT OF HUMAN AND MACHINE PERFORMANCE IN ACOUSTIC SCENE CLASSIFICATION: DCASE 2016 CASE STUDY*.

Microphone Impulse Response Project: About MicIRP. (2022). http://micirp.blogspot.com/p/about-micirp.html

Park, D. S., Chan, W., Zhang, Y., Chiu, C.-C., Zoph, B., Cubuk, E. D., & le Google Brain, Q. v. (2019). *SpecAugment: A Simple Data Augmentation Method for Automatic Speech Recognition*.

Rabiner, L. R. , and R. W. Schafer. (2010). *Theory and Applications of Digital Speech Processing: Vol. Upper Saddle River* (NJ: Pearson).

Ramachandran, P., Zoph, B., & Le, Q. v. (2017). *Searching for Activation Functions*. http://arxiv.org/abs/1710.05941

Rouat, J. (2008). Computational Auditory Scene Analysis: Principles, Algorithms, and Applications (Wang, D. and Brown, G.J., Eds.; 2006) [Book review]. *IEEE Transactions on Neural Networks*, *19*(1), 199-199. https://doi.org/10.1109/tnn.2007.913988

Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., Berg, A. C., & Fei-Fei, L. (2015). *ImageNet Large Scale Visual Recognition Challenge*. http://image-net.org/challenges/LSVRC/

Salamon, J. B. J. (2017). Deep Convolutional Neural Networks and Data Augmentation for Environmental Sound Classification. *IEEE Signal Processing Letters. Vol. 24, Issue 3*.

Stevens, S., & Volkmann, J. (1937). A Scale for the Measurement of the Psychological Magnitude Pitch. *The Journal of the Acoustical Society of America 8, 185*.

Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V., & Rabinovich, A. (2014). *Going Deeper with Convolutions*. http://arxiv.org/abs/1409.4842

Tompson, J., Goroshin, R., Jain, A., Lecun, Y., & Bregler, C. (2015). *Efficient Object Localization Using Convolutional Networks*.

Vaswani, A., Brain, G., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., & Polosukhin, I. (2017). *Attention Is All You Need*.

Zheng, W., Mo, Z., Xing, X., & Zhao, G. (2018). *CNNs-based Acoustic Scene Classification using Multi-Spectrogram Fusion and Label Expansions*.

9 Anexos

9.1 Anexo. Artículo de investigación

Aumento de audio y compresión de modelos para clasificación de escenas acústicas

Eduard Marcel González Poy

Universidad Internacional de la Rioja, Logroño (España)

14 de setiembre de 2022

Resumen

Este proyecto consiste en el desarrollo de un modelo de baja complejidad para clasificación de escenas acústicas; basado en el reto de la tarea 1 de DCASE 2022, donde se propone un modelo de referencia al que superar. Para conseguir la baja complejidad, se propone la destilación de conocimiento con una aproximación "maestro-alumno" y posterior cuantificación de los parámetros de la red al tipo entero de 8 bits. La red maestro se basa en redes convolucionales residuales y, la red alumno, es una red convolucional lineal. Se propone una metodología de aumento de datos de audio basada en la convolución entre la señal original del conjunto de datos propuesto y respuestas impulsiones de dispositivos no presentes en la captura de esos datos. El modelo propuesto supera al de referencia obteniendo un *log loss* de 1,415, dentro de la limitación de complejidad establecida.

Palabras Clave

Aumento de audio, clasificación de escenas acústicas, cuantificación de redes neuronales, destilación de conocimiento, redes convolucionales

I. Introducción

Este trabajo consiste en el desarrollo de un modelo para la clasificación de escenas acústicas (ASC) basado en la línea proporcionada por la tarea 1 de la octava edición (2022) del evento DCASE [1]. La solución debe cumplir con los requerimientos de los dispositivos de baja complejidad. La ASC es una tarea bien definida que consiste en caracterizar el entorno acústico proveniente de un audio, proporcionándole una etiqueta semántica [2]. Existen numerosos casos de uso de la ASC, como los dispositivos *wearables* y *hearables* conscientes del contexto, los audífonos, la atención médica remota, sistemas de vigilancia y seguridad, la monitorización de la vida salvaje en el hábitat natural, las ciudades inteligentes, el IoT y la navegación autónoma [3]. La mayor parte de estas aplicaciones se basan en el paradigma actual de la *Edge AI*; la inteligencia artificial (IA) desplegada en dispositivos cercanos al usuario. Diseñados con componentes para la realización de tareas específicas (microcontroladores); que requieren menos consumo de potencia en detrimento de la capacidad computacional. En consecuencia, es necesario el desarrollo de algoritmos para la ASC capaces de ser ejecutados en estos dispositivos.

II. Estado del Arte

En [3] se describe la estructura típica de un sistema de ASC. Constituido básicamente por: (A) Adquisición del audio, (B) Preparación de los datos y, (C) Modelado de los datos. En este trabajo se incluye (D) la compresión de redes neuronales profundas para su ejecución en dispositivos de baja complejidad. Este apartado se organiza siguiendo estos 4 puntos.

A. Adquisición del audio

El propósito de este artículo no es estudiar este punto, pero hay que tener presente que la conversión del sonido a audio digital puede afectar a la señal percibida.

Otro aspecto que afecta a este primer punto es la privacidad. Motivo por el que los últimos esfuerzos en investigación sobre el ASC se centran en obtener modelos ejecutables en dispositivos cercanos al usuario final, sin necesidad de procesar información en centros de datos que podría verse comprometida a nivel de seguridad y privacidad.

B. Procesamiento de los datos

Los algoritmos de ASC con mejores resultados en los últimos desafíos de DCASE utilizaron representaciones de espectrogramas basadas en el espaciado logarítmico de las frecuencias [4] (Figura 1). El proceso de cómputo del espectrograma en bandas de energía logarítmica de Mel, propuesto por Rabiner [5] se basa en una etapa de enventanado de la señal de audio según un número determinado de muestras (normalmente las ventanas se solapan entre ellas a razón del 50% de la duración).

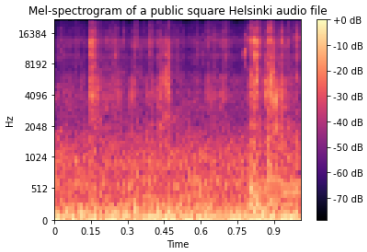

Figura 1: representación del audio en espectrograma Mel

Posteriormente, se aplica la transformada rápida de Fourier (FFT) para obtener la representación en el dominio frecuencial. Cada fragmento obtenido se pasa por un banco de filtros según la escala Mel [6]. Los valores espectrales emitidos por el banco de filtros Mel se suman y se concatenan de modo que cada fotograma se transforma en un vector columna.

Las técnicas de augmento de datos son de gran importancia en los modelos de aprendizaje profundo basados en redes convolucionales (CNN). En este apartado se describen las técnicas más comunes en referencia al aumento de audio aplicado directamente sobre la señal digital [7] y, el aumento aplicado sobre el espectrograma; aplicado en la mayoría de los trabajos del estado del arte de los sistemas ASC [8] (*SpecAugment*).

Salamon [7], propone 4 técnicas distintas: (1) inyección de ruido gaussiano, (2) Estiramiento del tiempo, (3) Estiramiento del tono y (4) Compresión de rango dinámico.

El *SpecAugment* trata el problema directamente sobre la representación visual. Consiste en deformar las características, enmascarar bloques de frecuencia y enmascarar bloques de tiempo. El su estudio, Park et al., proponen distintas políticas de aplicación. En la Figura 2

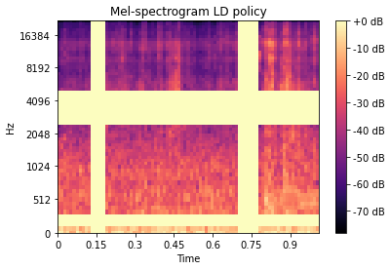

Figura 2: Representación de SpecAugment con política LD

se muestra la política LD, donde se enmascaran 2 segmentos distintos de frecuencia y tempo.

C. Modelado

Las propuestas con mejores resultados en ediciones anteriores de DACSE [4], se basan en redes ResNet [9]. Estas redes se constituyen por bloques no lineales. Las ResNet añaden una capa de conexión directa (*shortcut*, SC) entre la entrada al bloque convolucional y su salida. Su hipótesis es que: "dejar que las capas apiladas se ajusten a un mapeo residual es más fácil que dejar que se ajusten directamente al mapeo subyacente deseado" [9].

Recientemente, el uso de Transformers [10] se ha popularizado y extrapolado a tareas de VC con las redes Vit [11] o mobileViT [12], y clasificación de audio con los AST (*Audio Spectrogram Transformers*) [13]. La complejidad subyacente en estos modelos hace que la tarea de compresión y optimización no sea trivial. Debido al alcance de este trabajo, no es asumible el riesgo que implica el desarrollo de Transformers de baja complejidad.

D. Compresión de redes profundas

La gran cantidad de parámetros en las redes neuronales profundas (DNN), se traduce en una cantidad de memoria inasumible en dispositivos portables y aplicaciones a tiempo real. En la revisión de Liu et al., [14] se enumeran distintas técnicas que buscan solucionar el problema: modelos hechos a mano, compresión de modelos, arquitectura neuronal consciente del hardware y modelos de aprendizaje profundo adaptativo. En este trabajo se estudian los métodos relacionados con la compresión y optimización, basados en la reducción de parámetros redundantes en redes sobreparametrizadas; como la cuantificación y la destilación de conocimiento (KD, *Knowledge Distillation*). La cuantificación afronta este problema convirtiendo los pesos de las redes de precisión alta (float32) a pesos de precisión menor (int8, float16) conservando el rendimiento a nivel de exactitud [15]. KD no es un método de compresión en sí mismo, si no que busca transferir la capacidad de una red grande (maestra) a una red pequeña (alumna). Existen multitud de técnicas de KD diferenciadas por la representación del conocimiento, métodos de destilación o arquitecturas maestro-alumno [16]. La técnica de KD más popular para tareas de clasificación de imágenes [17], se basa en obtener las predicciones producidas por la red maestra y tratarlas como conocimiento para la red alumna empleando funciones de divergencia.

III. Objetivos y Metodología

El objetivo es implementar un modelo que consiga superar el rendimiento del modelo base de la tarea 1 de DACSE 2022 y, que esté dentro del límite de complejidad computacional propuesto. Esto es: un *log loss* de 1,575 y una exactitud (*accuracy*) de: 42,9%. La complejidad

computacional se mide en términos de cantidad de parámetros del tipo *int8* (8 bits con signo) y MMACs(*million multiply-accumulate operations*); siendo 128K y 30 millones los máximos respectivamente. El modelo de referencia ha sido entrenado durante 200 épocas con lotes de 16 ejemplos. El número de parámetros y MMACs de este modelo es de 46.512 y 29,23 respectivamente.

Para lograr este hito, se han seguido los puntos B, C y D estudiados en el estado del arte. Modularizando el desarrollo para conseguir paso a paso mejorar la solución propuesta. Se ha iterado sobre los módulos siguientes: (1) Procesamiento y representación del audio, (2) Propuesta de modelo y, (3) Destilación y compresión. En el siguiente apartado se profundiza en la aportación sobre cada uno de estos módulos.

IV. Contribución

A. Conjunto de datos

El conjunto de empleado (TAU Urban Acoustic Scenes 2022 Mobile) [18] está proporcionado por los organizadores de DACSE. Contiene grabaciones de 12 ciudades para 10 escenas acústicas distintas, capturadas con 4 dispositivos. Además, se han añadido dispositivos sintéticos emulando 11 dispositivos móviles. Cada fichero de audio dura 1 segundo, con una frecuencia de muestreo de 44.100 Hz y 24 bits. Se proporcionan los archivos *.csv* con el contenido de los audios para entrenamiento y prueba; este último conjunto contiene dispositivos que no aparecen en el conjunto de

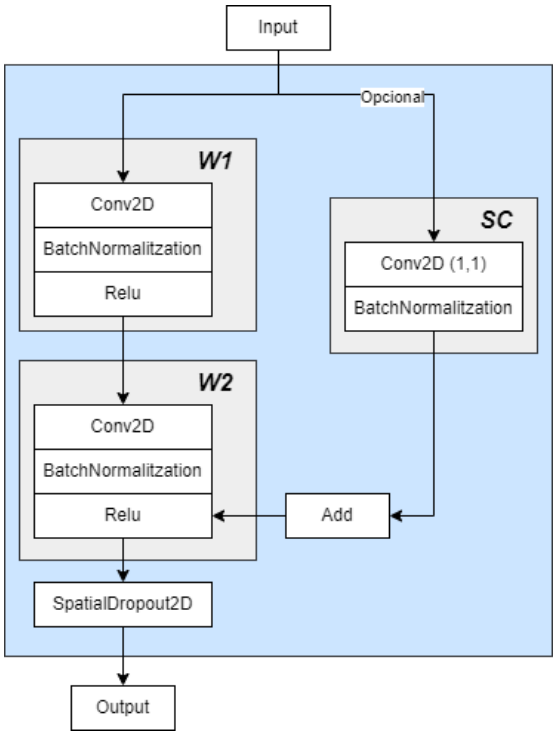

Figura 3: Bloque Residual de la red maestra

entrenamiento para penalizar la capacidad de generalización del modelo. En este trabajo se ha añadido un subconjunto para aumento con 15.000 ejemplos balanceados por dispositivo y extraídos del conjunto de entrenamiento.

B. Propuesta de modelo

La solución propuesta se basa en un enfoque maestro-alumno empleando la técnica KD y, posterior cuantificación del modelo alumno. Para la red maestra se ha desarrollado una red CNN residual siguiendo la línea de [9], con modificaciones para reducir su complejidad. Consiste en 5 etapas constituidas por bloques residuales (Figura 3). La tabla 1 describe la arquitectura implementada. Se ha substituido la capa *dropout* estándar por una capa *dropout* espacial [19].

TABLA 1: ARQUITECTURA DE LA RED MAESTRA

Etapa	Capa	Filtros	Kernel / Pool Size
1	Input	-	-
	Conv2D	128	5
	BatchNormalization	-	-
2	Bloque 2.1	128	W1 = 3, W2 = 1
	MaxPool2D	-	(2,2)
	Bloque 2.2	128	W1 = 3, W2 = 3
	Bloque 2.3	128	W1 = 3, W2 = 3
	Bloque 2.4	128	W1 = 3, W2 = 3
3	Bloque SC 3.1	256	W1 = 3, W2 = 1
	Bloque 3.2	256	W1 = 1, W2 = 1
	Bloque 3.3	256	W1 = 1, W2 = 1
	Bloque 3.4	256	W1 = 1, W2 = 1
4	Bloque SC 4.1	512	W1 = 1, W2 = 1
	Bloque 4.2	512	W1 = 1, W2 = 1
	Bloque 4.3	512	W1 = 1, W2 = 1
	Bloque 4.4	512	W1 = 1, W2 = 1
5	Conv2D	10	1
	BatchNormalization	-	-
	GlobalAveragePooling2D	-	-
	Dense (10)	-	-

Se ha etiquetado como W1 y W2 las capas con pesos y SC la capa de conexión directa. Se ha elegido Elu [20] como función de activación de las capas W1 y W2 tras varios experimentos frente a Relu [21] y Swish [22]. El número de parámetros asciende a 3.980.960.

La red alumno es una CNN lineal. Emulado la red maestro, está compuesta por 5 etapas. La etapa de entrada es análoga a la red maestro con menos filtros (32). La salida es idéntica a la red maestro. Las etapas 2, 3 y 4 consisten en una capa MaxPool seguida de un bloque residual sin SC. Tiene un total de 57.856 parámetros.

Para el entrenamiento de la red maestro se ha empleado el método de aprendizaje programado que sigue una función sigmoide [23], mejorando la estabilidad en la pérdida respecto la función de optimización Adam.

TABLA 2: VALORES DE LOSS EN FUNCIÓN DEL PROCESADO

datos	f_m (Hz)	W_s (ms)	M	imagen (píxeles)
base	44100	40	40	40x51
P_1	44100	40	80	80x51
P_2	22050	46	40	40x44
P_3	44100	23	40	40x87
P_4	44100	23	80	80x87

C. Representación y aumento de datos

Los datos se han procesado (Tabla 2) en función de la frecuencia de muestreo (f_m), la duración de la ventana (W_s) y el número de filtros Mel (M).

Para el aumento de datos se propone un método propio. Se aplica la convolución con respuestas frecuenciales (o, IR) de micrófonos reales no utilizados en la captación original. A continuación, se aplica una compresión dinámica para evitar la saturación en la salida del sistema. La intención es que el modelo aprenda características no relacionadas con la IR de los dispositivos. Las IR se pueden encontrar bajo licencia *creative commons* en [24]. Se han utilizado las IR de 8 micrófonos: AKG D12, Altec 639, American R331, Astatic 77, Beyer M260, FosterDynamic DF1, Oktava MK18 Silver y Sony C37Fet. La figura 4 muestra un ejemplo de esta aplicación.

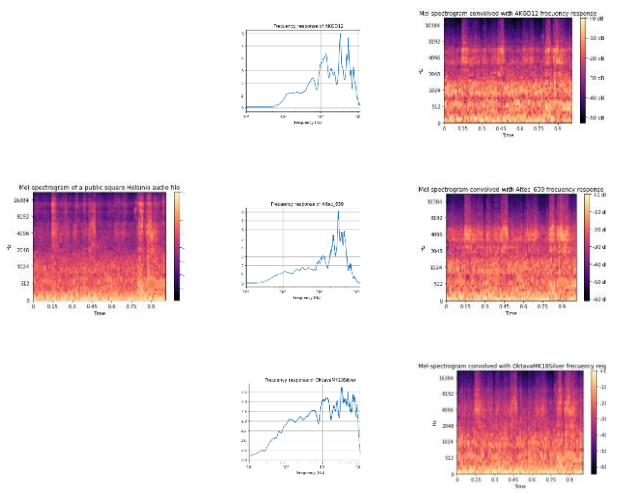

Figura 4: Resultado de convolución entre audio original y respuesta al impulso de micrófonos reales

D. Destilación y compresión

La KD se ha aplicado siguiendo el código en [25] y, la cuantificación empleando la librería TFlite [26].

V. Evaluación y Resultados

Para entrenar el modelo maestro se ha seguido los siguientes pasos: (1) buscar el mejor modelo con los datos base, (2) comparar el *loss* con otros procesados (Tabla 2), y, (3) comparar con el aumento de datos propuesto.

En el primer paso se ha conseguido un *loss* = 1.556; con 5 épocas y 64 ejemplos por lote. En el segundo paso, un *loss* = 1,477 con los datos P_1 (Tabla 3). Por último, se ha aplicado los datos de aumento siguiendo dos distribuciones entre entrenamiento y validación: A1 y A2. La tabla 4 resume los resultados.

(A1). Se han dividido los datos originales y aumentados a razón de 70-30% para entrenamiento y validación respectivamente; cada división se ha concatenado y mezclado aleatoriamente.

Tabla 3: Resultados del entrenamiento en función del procesado de los datos

datos	épocas	loss
base	5	1,556
P_1	5	1,562
	8	**1,477**
	11	1,484
P_2	5	1,621
P_3	5	1,639
	8	1,642
P_4	5	1,534
	8	1,541
	11	1,531

Tabla 4: Resultado del entrenamiento con datos aumentados

aumento	épocas	loss
A1	5	1,499
	8	1,463
	11	1,486
A2	5	1,468
	8	**1,432**
	11	1.484

Tabla 5: Resultados de la destilación en función de la temperatura

T	épocas	loss
2	6	2,604
	10	1,545
	22	**1.416**
3	6	2,238
	10	2,043
	22	1,558
4	6	4,000
	10	2,073
	22	1,616

(A2). El 70% de los datos de aumento se ha utilizado para validación sin añadir datos originales. Para el entrenamiento se han usado todos los datos originales más el 30 % de datos aumentados restantes mezclados de forma aleatoria.

Con el mejor modelo maestro, se ha aplicado la destilación al alumno con distintos valores de temperatura (T) (Tabla 5).

TABLA 6: MÉTRICAS DEL MODELO PROPUESTO

loss	accuracy	Número de parámetros	MMACs
1,415	48.73%	57.660	26,116

El resultado final, después de la cuantificación se muestra en la tabla 6.

VI. DISCUSIÓN

El modelo propuesto obtiene mejores resultados en *loss*, *accuracy* y MMACs. Tiene 11.148 parámetros más que el modelo de referencia. Aunque, aún está por debajo de los 128.000 permitidos. Asimismo, restan 3,884 MMACs para el límite. En consecuencia, el modelo propuesto permite cierta escalabilidad.

El mejor resultado de la Tabla 3 se consigue con el procesado P_1. Podemos deducir que: (1) la resolución frecuencial es importante, (2) las frecuencias altas aportan información útil para la ASC y, (3) la resolución temporal demasiado baja degrada el resultando del entrenamiento.

El mejor resultado de la tabla 4 se obtiene con A2. El modelo generaliza mejor debido a que está aprendido características de los espectrogramas que no están tan relacionadas con la IR del dispositivo; tal como se había previsto.

VII. CONCLUSIONES

En este trabajo se describe el desarrollo de un modelo capaz de generalizar mejor que el modelo de referencia, en la tarea de clasificación de escenas acústicas; añadiendo el requisito de cumplir con limitaciones computacionales. Además, se ha conseguido mejorar en exactitud y se ha reducido la cantidad de MMACs. También, se ha propuesto una metodología propia para aumento de datos de audio que ha permitido mejorar los resultados iniciales.

Se han identificado una serie de mejoras para futuras líneas de trabajo. Estudiar la distribución del conjunto de datos para entrenamiento y validación. Por ejemplo, distribuir por ubicación o localización de la captura del audio, puesto que esta característica podría afectar a la capacidad de generalizar. Incrementar el número de filtros Mel y la cantidad de muestras por ventana; esto permitiría generar y valorar más resultados. O, añadir más cantidad de datos aumentados con otras técnicas.

Por último, una línea de trabajo muy interesante sería el despliegue de este modelo en un microcontrolador o dispositivo portable. Permitiendo aplicar la solución en un entorno real y, en consecuencia, analizar el trabajo desde otra perspectiva.

REFERENCIAS

[1] "Low-Complexity Acoustic Scene Classification - DCASE," 2022. https://dcase.community/challenge2022/task-low-complexity-acoustic-scene-classification (accessed Apr. 06, 2022).

[2] J.-J. Aucouturier, B. Defreville, and F. Pachet, "The bag-of-frames approach to audio pattern recognition: A sufficient model for urban soundscapes but not for polyphonic music," *J Acoust Soc Am*, vol. 122, no. 2, pp. 881–891, Aug. 2007, doi: 10.1121/1.2750160.

[3] J. Abeßer, “A review of deep learning based methods for acoustic scene classification,” *Applied Sciences (Switzerland)*, vol. 10, no. 6, Mar. 2020, doi: 10.3390/app10062020.

[4] I. Martín-Morató, T. Heittola, A. Mesaros, and T. Virtanen, “Detection and Classification of Acoustic Scenes and Events 2021 LOW-COMPLEXITY ACOUSTIC SCENE CLASSIFICATION FOR MULTI-DEVICE AUDIO: ANALYSIS OF DCASE 2021 CHALLENGE SYSTEMS,” 2021. [Online]. Available: http://dcase.community/challenge2021/task-acoustic-scene-

[5] L. R. , and R. W. Schafer. Rabiner, *Theory and Applications of Digital Speech Processing*, NJ: Pearson., vol. Upper Saddle River. 2010.

[6] S. Stevens and J. Volkmann, “A Scale for the Measurement of the Psychological Magnitude Pitch,” *The Journal of the Acoustical Society of America 8, 185*, 1937.

[7] J. B. J. Salamon, “Deep Convolutional Neural Networks and Data Augmentation for Environmental Sound Classification,” *IEEE Signal Processing Letters. Vol. 24, Issue 3*, 2017.

[8] D. S. Park *et al.*, “SpecAugment: A Simple Data Augmentation Method for Automatic Speech Recognition,” 2019.

[9] K. He, X. Zhang, S. Ren, and J. Sun, “Deep Residual Learning for Image Recognition,” 2015. [Online]. Available: http://image-net.org/challenges/LSVRC/2015/

[10] A. Vaswani *et al.*, “Attention Is All You Need,” 2017.

[11] A. Dosovitskiy *et al.*, “AN IMAGE IS WORTH 16X16 WORDS: TRANSFORMERS FOR IMAGE RECOGNITION AT SCALE,” 2021, Accessed: Sep. 04, 2022. [Online]. Available: https://github.com/

[12] S. Mehta and M. R. Apple, “MobileViT,” 2021, Accessed: Aug. 11, 2022. [Online]. Available: https://github.com/apple/ml-cvnets.

[13] Y.-A. Chung, Y. Gong, and J. Glass, “AST: Audio Spectrogram Transformer,” 2021. [Online]. Available: https://github.com/YuanGongND/ast.

[14] D. Liu, H. Kong, X. Luo, W. Liu, and R. Subramaniam, “Bringing AI To Edge: From Deep Learning’s Perspective,” Nov. 2020, doi: 10.1016/j.neucom.2021.04.141.

[15] A. Gholami, S. Kim, Z. Dong, Z. Yao, M. W. Mahoney, and K. Keutzer, “A Survey of Quantization Methods for Efficient Neural Network Inference,” Mar. 2021, [Online]. Available: http://arxiv.org/abs/2103.13630

[16] J. Gou, B. Yu, S. J. Maybank, and D. Tao, “Knowledge Distillation: A Survey,” 2021.

[17] G. Hinton, O. Vinyals, and J. Dean, “Distilling the Knowledge in a Neural Network,” Mar. 2015, [Online]. Available: http://arxiv.org/abs/1503.02531

[18] T. Heittola, A. Mesaros, and T. Virtanen, “Acoustic scene classification in DCASE 2020 Challenge: generalization across devices and low complexity solutions,” May 2020, [Online]. Available: http://arxiv.org/abs/2005.14623

[19] J. Tompson, R. Goroshin, A. Jain, Y. Lecun, and C. Bregler, “Efficient Object Localization Using Convolutional Networks,” 2015.

[20] D.-A. Clevert, T. Unterthiner, and S. Hochreiter, “Fast and Accurate Deep Network Learning by Exponential Linear Units (ELUs),” Nov. 2015, [Online]. Available: http://arxiv.org/abs/1511.07289

[21] A. M. Fred Agarap, “Deep Learning using Rectified Linear Units (ReLU),” 2019. [Online]. Available: https://github.com/AFAgarap/relu-classifier.

[22] P. Ramachandran, B. Zoph, and Q. v. Le, “Searching for Activation Functions,” Oct. 2017, [Online]. Available: http://arxiv.org/abs/1710.05941

[23] D. Macêdo, P. Dreyer, T. Ludermir, S. Member, and C. Zanchettin, “Training Aware Sigmoidal Optimizer,” 2021.

[24] “Microphone Impulse Response Project: About MicIRP,” 2022. http://micirp.blogspot.com/p/about-micirp.html (accessed Sep. 11, 2022).

[25] “Knowledge Distillation,” 2022. https://keras.io/examples/vision/knowledge_distillation/ (accessed Sep. 11, 2022).

[26] R. David *et al.*, “TensorFlow Lite Micro: Embedded Machine Learning on TinyML Systems,” Oct. 2020, [Online]. Available: http://arxiv.org/abs/2010.08678