Fusión de redes neuronales para la detección de eventos acústicos asociados a la seguridad ciudadana

Ricardo Tangarife González

Trabajo de investigación presentado para optar por el título de
Magíster en Ingeniería

Director
PhD. Ricardo Andrés Velásquez Vélez

Co-Director
PhD. Fredy Alexander Rivera Vélez

Universidad de Antioquia
Facultad de Ingeniería
Maestría en Ingeniería
Medellín, Antioquia, Colombia

2024

Cita	Tangarife Gonzalez [1]
Referencia Estilo IEEE (2020)	[1] R. Tangarife Gonzalez, “Fusión de redes neuronales para la detección de eventos acústicos asociados a la seguridad ciudadana”, Tesis de maestría, Maestría en Ingeniería, Universidad de Antioquia, Medellín, Antioquia, Colombia, 2024.

Maestría en Ingeniería,

Grupo de Investigación Sistemas Embebidos e Inteligencia Computacional (SISTEMIC).

Agradecimientos

Agradecimientos a mi director de tesis, Ricardo Andrés Velásquez, y a mi co-director, Fredy Alexander Rivera, por su asesoría y orientación a lo largo de todo el proceso de esta investigación. También deseo reconocer al grupo de investigación SISTEMIC y a mis compañeros de maestría, Fabian Stiven Duque y Laura Saldarriaga, por su colaboración y apoyo.

Agradezco a mi madre, mi hermana, mi familia y mis amigos, quienes estuvieron a mi lado brindándome ánimo y apoyo en cada etapa de este proceso académico y personal.

Gracias a la Universidad de Antioquia y al proyecto de Seguridad Ciudadana del Sistema General de Regalías (SGR) identificado con el código BPIN 2020000100044 por el respaldo financiero brindado.

Resumen

La seguridad ciudadana se ha convertido en un eje central para garantizar la calidad de vida de los habitantes de las diferentes ciudades del mundo, apoyándose de las herramientas tecnológicas como las cámaras de video se han podido mitigar situaciones de riesgo mediante mecanismos de monitoreo y control. Los sonidos que se encuentran en el ambiente urbano son una fuente de información que ha venido ganando relevancia para detectar situaciones de peligro para los ciudadanos. Así, a partir de eventos acústicos es posible generar alertas para que sean atendidas por las entidades encargadas. Para el procesamiento de sonidos urbanos es común utilizar redes neuronales profundas, ya que permiten identificar con gran precisión los eventos de riesgo. El cómputo en el borde como alternativa al cómputo en la nube es otra tendencia que busca mejorar la eficiencia de los sistemas de monitoreo. La confluencia de los modelos de aprendizaje profundo y la computación en el borde ha generado nuevos retos como la ejecución eficiente de los modelos de aprendizaje profundo en los dispositivos de borde, estos últimos con limitaciones en cuanto a capacidad de cómputo, memoria y almacenamiento. Para mejorar la eficiencia en la ejecución de modelos neuronales en el borde existen diferentes técnicas a nivel tanto de modelo como de implementación. La fusión de modelos de redes neuronales es una de estas técnicas ya que permite la combinación de dos o más modelos en uno solo, tal que, el nuevo modelo permita las inferencias de los modelos iniciales, pero reduciendo los requisitos de cómputo, memoria y almacenamiento. En este trabajo de maestría se plantea una estrategia que utiliza la fusión de redes neuronales para la detección de eventos acústicos asociados a la seguridad ciudadana, en el contexto de ejecución en un dispositivo de borde. Encontramos que mediante la estrategia de fusión planteada en costes computacionales como huella en memoria, peso del modelo y tiempo de ejecución en un contexto de dispositivo de borde beneficia en 2x la ejecución del modelo fusionado, versus la ejecución de los modelos individuales, sin afectar el desempeño en F1-Score del modelo. Además, se encontraron relaciones cruciales entre los parámetros MFCC y el rendimiento del modelo, proporcionando pautas para optimizar la eficiencia y la precisión.

Palabras clave

Seguridad Ciudadana, Computación en el borde, Aprendizaje profundo, Detección de eventos acústicos, Fusión de redes neuronales

Índice general

Índice de figuras

Índice de tablas

Capítulo 1

Introducción

Como ciudadanos, necesitamos sentirnos seguros y protegidos en el lugar y la zona en la que vivimos. Actualmente, las entidades gubernamentales tratan este tema con el término Seguridad Pública, que se define como la protección universal de los ciudadanos, garantizando su vida, integridad, libertad y patrimonio [1]. En los Objetivos de Desarrollo Sostenible (ODS) adoptados a partir del 2015 por la Organización de las Naciones Unidas (ONU), se encuentra la seguridad ciudadana en el objetivo número 16 de Paz y Justicia, en el cual contemplan que los conflictos, la inseguridad, las instituciones débiles y el acceso limitado a la justicia continúan suponiendo una grave amenaza para el desarrollo sostenible [2]. Localmente, los resultados arrojados por las encuestas anuales de percepción ciudadana Medellín Cómo Vamos hasta el 2023, muestran que durante los últimos tres años los ciudadanos se sienten cada vez menos seguros en la ciudad y en el barrio. Entre enero y agosto de 2023 se denunciaron más de 43 mil casos de victimización en Medellín por delitos como hurtos, lesiones personales, violencia intrafamiliar, delitos sexuales, homicidios, feminicidios, entre otros. En 2023, el 18 % de los ciudadanos encuestados dijo haber sido víctima de algún delito, siendo esta la cifra más alta desde 2008 [3].

Diversos mecanismos y sistemas de seguridad han sido desarrollados a lo largo de los años. T. D. Räty et al. en [4] clasificaron los sistemas de seguridad en 3 generaciones, de acuerdo con las tecnologías implementadas para su objetivo. La primera generación comprende los dispositivos empleados entre 1960 y 1980, los cuales fueron principalmente cámaras de vídeo analógicas. El avance de la tecnología de vídeo abrió paso a la segunda generación de sistemas de seguridad, entre los años 1980 y 2000. Las comunicaciones digitales permitieron la compresión, transmisión y reducción de ancho de banda de los sistemas de vigilancia basados en video, incrementado su desempeño y ampliando su utilidad en múltiples sectores públicos y privados. La tercera generación, se caracteriza por el incremento en el número de lugares de monitoreo, creando sistemas de adquisición de datos distribuidos, que realizan reportes a un sistema central de monitoreo y procesamiento. Así mismo, implementando modelos estadísticos para la detección y reconocimiento de objetos y personas; esta generación comprende el periodo entre los años 2000 y 2011. De allí en adelante hemos visto un crecimiento tecnológico continuo, con el surgimiento del Internet de las cosas (IoT - Internet of Things) que posibilita el monitoreo continuo y en tiempo real, no solo de video sino de otras fuentes de información como audio, ruido, telemetría, entre otras. De esta forma es posible aumentar el cubrimiento y monitoreo en los principales focos

de inseguridad en las diferentes ciudades del mundo. Es posible enmarcar estas tendencias como una cuarta generación de sistemas de seguridad en la cual se incluyen los modelos predictivos de aprendizaje de máquina aplicados en cada una de las variables monitoreadas.

Si bien el video ha centrado la atención del sector de vigilancia automática, las señales acústicas se abren camino y ofrecen nuevas posibilidades para el control, la vigilancia y el apoyo en materia de seguridad ciudadana. Las cámaras de video tienen desventajas ante condiciones ambientales adversas como la lluvia, la niebla, o la oscuridad de la noche. Sin embargo, los eventos acústicos permiten identificar situaciones de riesgo, incluso si se presentan algunas de las condiciones ambientales mencionadas anteriormente. A lo largo de las ciudades se generan sonidos que permiten detectar situaciones de alerta, prevención, o peligro. Múltiples señales acústicas han sido utilizadas para la alerta temprana y protección de la vida, integridad y seguridad de las personas, tales como gritos humanos [5–7], explosiones [8; 9], disparos [10–13], llanto de bebés [14; 15] y choques de autos [16; 17]. Las señales acústicas también han sido utilizadas para estudiar el entorno y detectar situaciones particulares en él, tales como el ruido de Aeronaves No Tripuladas (UAV - Unmanned Aerial Vehicle) [18; 19], lluvia [20; 21], bocinas [22; 23] y aves [24]. Otras señales como ladridos de perros [22; 23; 25], alarmas [26; 27], sirenas [28; 29] y rompimiento de vidrio [30; 31], han sido utilizadas para detectar situaciones de peligro y delitos contra el patrimonio de los ciudadanos. Es por esto que la relevancia de cada uno de estos sonidos depende del contexto y de la aplicación a la que se destine. En particular, para efectos de seguridad ciudadana es muy importante identificar señales de situaciones de alerta y/o peligro que puedan afectar la integridad, seguridad, y patrimonio de los ciudadanos.

1.1 Planteamiento del Problema

Actualmente, el cómputo en la nube es la plataforma de procesamiento más utilizada para el procesamiento de señales de audio y video en la tarea de detección de eventos asociados a la seguridad. Si bien este tipo de cómputo proporciona toda la capacidad de procesamiento requerida para realizar estas tareas, también tiene grandes desventajas y desafíos en cuanto a escalabilidad. Las implicaciones de enviar los datos a un nodo de cómputo centralizado, de un número cada vez mayor de sensores, son baja eficiencia energética, altos costos, considerable latencia y riegos a la privacidad de los ciudadanos. El ancho de banda para la transmisión de audio y video es considerable, lo cual tiene un gran impacto en los costos recurrentes del sistema. Por otro lado, debido a la gran cantidad de datos es común utilizar técnicas de compresión con pérdidas para reducir el ancho de banda requerido, lo cual tiene implicaciones en la calidad de los datos con los que trabajan los algoritmos de detección de eventos en la nube. Así mismo, la latencia también puede tener implicaciones en cuanto a la pronta respuesta del sistema a eventos críticos de seguridad que requieran atención inmediata. Adicionalmente, los riesgos a la violación de la privacidad de los ciudadanos es otra desventaja de los sistemas basados en la nube. Información sensible de los ciudadanos es transmitida a través de la Internet, y almacenada en servidores donde los ciudadanos no tienen ningún control. Sin embargo, la información puede ser utilizada de forma malintencionada ya sea

por las compañías que ofrecen los servicios de cómputo, o por un tercero malintencionado que pueda violar la seguridad de los sistemas de comunicación o del sistema de almacenamiento. Finalmente, el consumo energético es otra implicación que no debe ser menospreciada en la actualidad. La transmisión y procesamiento de estos grandes volúmenes de datos tienen altos consumos energéticos. La computación en el borde es un paradigma reciente de computación, diferente a la computación en la nube, que consiste en tener una arquitectura de cómputo distribuida que permite el procesamiento de los datos cerca de la fuente. Esta arquitectura, permite resolver o mitigar algunas de las desventajas y desafíos previamente mencionados para el cómputo en la nube. El ancho de banda requerido para transmitir los datos disminuye. Si los datos se procesan en el borde, entonces no es necesario enviarlos al sistema central, así solo el resultado del procesamiento es transmitido. La latencia, la eficiencia energética y la privacidad de los datos también mejoran con el paradigma de cómputo en el borde.

El aprendizaje profundo (DL - Deep Learning) es un enfoque del aprendizaje de máquina compuesto por múltiples capas encargadas de aprender representaciones de datos con diferentes niveles de abstracción, con el objetivo de predecir, clasificar o detectar información. Por su capacidad de aprender modelos complejos, el aprendizaje profundo ha tenido éxito en muchos dominios, incluyendo tareas de visión, procesamiento del lenguaje natural y reconocimiento del habla [32]. De igual forma, se ha empleado en las recientes tareas de clasificación y detección de eventos acústicos [33–35]. La construcción y ejecución de los modelos de aprendizaje profundo representan una elevada complejidad computacional por la cantidad de operaciones y parámetros que se emplean. Algunos de los modelos pre-entrenados para la clasificación de múltiples sonidos se listan en la Tabla 1.1. Para cada modelo se presenta el número de parámetros, la precisión y el número de operaciones flotantes necesarias para ejecutar una instancia del modelo en MFLOPS (Millions of Float Point Operations). Estos modelos son evaluados con el conjunto de datos UrbanSound8k [36] que contienen diez clases diferentes de eventos acústicos urbanos.

Tabla 1.1: Características modelos pre entrenados para clasificación múltiples sonidos

Modelo	**Precisión (%)**	**MFLOPS**	**Número Parámetros (10^6)**
YAMNet	96.16	69	3.7
VGGish	96.70	360.72	73.54
ShuffleNet	87.18	149	2
SqueezeNet	83.97	833	1.3
GoogleNet	87.06	150	7

Al momento de trasladar los modelos de aprendizaje profundo al paradigma de computación en el borde nos encontramos con un conjunto de restricciones debidas al tipo de dispositivos utilizados. Un ejemplo de ello son los computadores de una sola tarjeta (SBC - Single-Board Computer), ampliamente usados por sus características de bajo costo y consumo de energía. Los SBCs son usados en situaciones en las que un computador estándar no sería adecuado y los requisitos de procesamiento no pueden ser satisfechos por microcontroladores. Comparado con un servidor o computador personal, los SBCs tienen restricciones en el tamaño de memoria, la velocidad y la

capacidad de procesamiento. El rango de restricciones para la memoria RAM se encuentra entre 512MB y máximo 8GB, por parte de la velocidad de los núcleos está en el rango de 700MHz a los 2GHz, con un máximo de 8 núcleos por procesador [37]. Si bien el almacenamiento nativo está restringido a unas decenas de gigas, es posible expandirlo por medio de memorias SD, módulos M2, y discos externos. Conocidas las restricciones de los dispositivos de borde, se han desarrollado múltiples estrategias para la ejecución eficiente de estos modelos predictivos en el borde. Algunas de éstas utilizan técnicas como compresión [38; 39], cuantización [40; 41] o poda [42; 43] de parámetros del modelo. La reducción en la precisión aritmética de las operaciones también ha sido utilizada [44; 45].A partir de esto, surge el planteamiento del problema de cuál es la forma más eficiente de detectar múltiples eventos acústicos asociados a la seguridad empleando dispositivos de borde y basados en las técnicas de ejecución eficiente de estos modelos de aprendizaje profundo, teniendo en cuenta el dinamismo que se puede dar en las zonas urbanas, con respecto a los eventos de seguridad ciudadana de interés. Una de las técnicas para la ejecución eficiente, es la fusión de modelos de redes neuronales, mediante la cual se hace la combinación o mezcla de dos o más modelos, para compartir una porción de la arquitectura del modelo final implementado, es decir una porción de los parámetros. Esta técnica busca reducir el tamaño en memoria y el número de operaciones matemáticas del modelo fusionado comparado al uso de modelos independientes [46]. Por tanto, se espera que a través de la implementación de la fusión de modelos específicos previamente entrenados para la detección de eventos acústicos permitan reducir el número de parámetros y mantener la precisión, comparado con la utilización de un único modelo para la detección de los múltiples eventos como los citados en la Tabla 1.1.

En síntesis, la detección de eventos acústicos como apoyo a la seguridad ciudadana está generando un amplio campo de interés por los aportes que se pueden generar entorno al bienestar y la calidad de vida de los ciudadanos. Las tecnologías emergentes como la computación en el borde permiten obtener beneficios sobre las tecnologías en las que se han estado implementando los sistemas de seguridad vigentes como lo es la computación en la nube, pero igualmente representa un desafío para la ejecución eficiente bajo recursos limitados. Como se expondrá en el estado del arte, las técnicas de fusión para ejecución eficiente de modelos de aprendizaje profundo se han estudiado poco dentro del paradigma de computación en el borde. Así mismo, en la literatura tampoco hemos encontrado evidencia de que estas técnicas hayan sido aplicadas a la detección de eventos acústicos en el contexto de seguridad ciudadana. Por otra parte, no se encontraron reportes en la literatura en los que la fusión se realice a partir de modelos previamente entrenados y efectivamente se reduzca el costo computacional del modelo fusionado. Las conclusiones presentadas en los artículos [46] y [47] han mostrado un camino a explorar el cual presenta buenos resultados al fusionar algunos modelos de redes neuronales, en términos de precisión del modelo y tamaño en memoria del modelo fusionado, respectivamente. Este trabajo de investigación busca definir e implementar una estrategia para la detección de múltiples eventos acústicos asociados a la seguridad ciudadana utilizando técnicas de fusión de redes neuronales y su ejecución en un dispositivo de borde SBC. Partiendo de la hipótesis de que la fusión de modelos de aprendizaje profundo reducirá los requisitos de memoria, cómputo y almacenamiento del modelo fusionado comparado a los modelos independientes, y posibilitará de es-

ta forma su implementación en dispositivos de borde sin una reducción apreciable en la exactitud de los modelos. Además de facilitar la integración dinámica de diferentes modelos de eventos acústicos de interés.

1.2 Objetivos

1.2.1 Objetivo General

Definir e implementar una estrategia para la detección de diferentes eventos acústicos asociados a la seguridad ciudadana usando técnicas de fusión de redes neuronales y dispositivos de borde, para reducir el tamaño en memoria y la latencia del modelo fusionado sin comprometer su precisión con respecto a los modelos individuales.

1.2.2 Objetivos Específicos

1. Seleccionar los modelos pre entrenados de redes neuronales para la implementación del reconocimiento de al menos dos eventos acústicos en el ámbito de la seguridad ciudadana, evaluando métricas de latencia, tamaño en memoria y precisión.

2. Definir e implementar una estrategia utilizando técnicas de fusión de redes neuronales que permita la detección de los eventos acústicos seleccionados y su ejecución eficiente en un dispositivo de borde.

3. Evaluar los resultados obtenidos de la estrategia definida mediante métricas de tamaño en memoria, latencia y precisión, teniendo como referencia comparativa la ejecución individual de los modelos.

1.3 Contribución del Trabajo de Investigación

En este trabajo de maestría se investigaron y se propusieron diversas estrategias de fusión de redes neuronales convolucionales. Nuestro enfoque se centró en la creación de un modelo fusionado a partir de dos modelos individuales, alcanzando, en algunos casos, igualar o mejorar el rendimiento en F1-Score promedio en comparación con las tareas individuales. Este resultado se logró sin sacrificar métricas críticas como la huella en memoria, el peso del modelo y el tiempo de ejecución, en relación a la ejecución simultanea de los modelos individuales, aspectos cruciales en entornos de dispositivos de borde.

Nuestra investigación también incluyó la exploración del espacio de diseño en los hiperparámetros de la extracción de características del audio mediante los Coeficientes Cepstrales de Frecuencia Mel (MFCC - Mel Frequency Cepstral Coeficients). Descubrimos que los modelos que utilizan parámetros más altos, como coeficientes MFCC, tienden a ofrecer un mejor rendimiento en el reconocimiento de eventos acústicos. No obstante, estos modelos, aunque más precisos, son computacionalmente más costosos

y requieren más tiempo para entrenarse y evaluar. En consecuencia, enfatizamos el equilibrio entre el número de parámetros y el rendimiento del modelo, considerando cuidadosamente las limitaciones de hardware y tiempo. De esta exploración de espacio de diseño presentamos un artículo nombrado "*Performance Evaluation of CNN Models in Urban Acoustic Event Recognition through MFCC Hyperparameter Search*" (DOI 10.1109/CSCE60160.2023.00035), presentado en *The 25th International Conference on Artificial Intelligence* (ICAI'23) llevada a cabo entre el 23 y 27 de Julio de 2023 en Las Vegas, USA.

1.4 Organización del Documento

El resto de este trabajo de investigación de maestría está organizado de la siguiente manera:

- **Capítulo 2:** Presenta el marco teórico con conceptos fundamentales en los temas de sistemas de Detección de Eventos Acústicos, aprendizaje profundo y fusión de redes neuronales.
- **Capítulo 3:** Presentamos la exploración del espacio de diseño para la extracción de características con coeficientes cepstrales de frecuencia de Mel. De manera previa a la experimentación, se presentan las bases de datos utilizadas, los eventos de seguridad ciudadana a tratar y la selección de la arquitectura convolucional.
- **Capítulo 4:** Aquí definimos, proponemos y evaluamos diferentes estrategias de fusión. También se presentan los modelos neuronales que se usan como referencia.
- **Capítulo 5:** En este capítulo presentamos parte esencial de este trabajo de investigación, donde exponemos por secciones los resultados de las métricas de desempeño de ejecución. Comparando los modelos fusionados con los individuales y biclase, al ejecutarse en un dispositivo de borde.
- **Capítulo 6:** Contiene las conclusiones generales del trabajo de investigación y los futuros trabajos de investigación que podrían explorarse a partir de estos resultados.

Capítulo 2

Marco Teórico

En la Sección 2.1 se presentan algunos conceptos sobre la detección de eventos acústicos y algunas aplicaciones reportadas en la literatura. La Sección 2.2 sintetiza los principales eventos acústicos que se han reportado en la literatura. A continuación, se discuten algunos conceptos sobre aprendizaje profundo dando una visión de las tipologías de redes neuronales profundas usadas recientemente para la detección de los eventos acústicos. La sección 2.4 presenta algunas estrategias para la ejecución eficiente de redes neuronales profundas. Finalmente, en la sección 2.5 presentamos los conceptos básicos de fusión de redes neuronales y el estado del arte de esta rama de investigación.

2.1 Reconocimiento de Eventos Acústicos

La definición de evento está asociada a circunstancias particulares que involucran o están relacionadas con las personas. Este concepto se aplica en campos como las ciencias aplicadas y las ciencias de la computación, incluyendo sensores inteligentes, administración de salud e inteligencia de negocios. También se utiliza en situaciones de crisis, investigaciones criminales, medios de comunicación y ciencias sociales [48]. Así, el Reconocimiento de Eventos Acústicos (AER - Acoustic Event Recognition) se refiere a la tarea de identificar eventos específicos que ocurren en un fragmento de audio. Esta tecnología tiene un amplio uso en áreas de seguridad, asistencia a personas e interacción humano-computador, utilizando técnicas como el Reconocimiento Automático del Habla (ASR - Automatic Speech Recognition), la Identificación del Hablante (SID - Speaker Identification) y la Recuperación de Información Musical (MIR - Music Information Retrieval) [36].
La implementación de aplicaciones AER ha utilizado métodos de modelado estadístico clásicos para la clasificación de eventos. Xia et al. en [49], y Phan et al. en [50], utilizaron técnicas de bosques aleatorios (RF Random Forest) para la detección de eventos acústicos. Diment et al. en [51] y Schröder et al. en [52] realizaron la implementación a través de modelos ocultos de Markov (HMM - Hidden Markov Models). Otra técnica utilizada en este campo son las máquinas de soporte vectorial (SVM - Support Vector Machine) utilizada por los autores Nogueira et al. en [53].
Recientemente se han implementado modelos basados en técnicas de aprendizaje profundo debido al éxito demostrado en tareas como el procesamiento de lenguaje natural

y visión por computadora. Los modelos de aprendizaje profundo logran mayor precisión comparado con los clasificadores clásicos. Tres arquitecturas de redes neuronales profundas (DNNs - Deep Neural Networks) se han usado principalmente en los trabajos reportados en la literatura. Kong et al. en [33] implementan una DNN completamente conectada para la detección de eventos acústicos. Chou et al. en [34] proponen una estrategia para la detección de eventos acústicos basado en redes neuronales convolucionales (CNN - Convolutional Neural Networks). La tercera arquitectura basada en redes neuronales convolucionales recurrentes (CRNN Convolutional Recurrent Neural Networks) utilizada por Cakir et al. en [35] en este caso para la detección específica del evento acústico de sonidos de pájaros. Otro trabajo de detección de eventos acústicos, pero ejecutándose en dispositivos de borde es el de Wei, Y et al. en [54] que presenta un modelo de CRNN, denominado EdgeCRNN. Este trabajo se enfoca en la detección de palabras clave. Logrando la detección de 12 palabras clave y obteniendo un procesamiento de 11,1 audios por segundo en una Raspberry Pi 3B+ como dispositivo de borde, su tasa de precisión alcanza el 98,05 %. Similarmente, en el ámbito de salud, Zhilibayev en [55], presenta un sistema para la monitorización de emociones humanas utilizando el habla en tiempo real. En el estudio se evalúan diferentes modelos CNN para la detección de 8 clases de emociones. Finalmente, realizan pruebas con una CNN unidimensional utilizando una extracción de características de sonido con los MFCC. Obtuvieron una precisión del 82,3 % y un tiempo de respuesta total de 0,261seg al ejecutarse en una placa Raspberry Pi 3B+.
Una tarea relacionada a la detección de eventos acústicos es la localización de los eventos acústicos o fuentes de sonido. Esta consiste en encontrar la fuente de sonido a través de un arreglo espacial de micrófonos. La localización del sonido aporta ampliamente en campos como la salud en el apoyo de personas discapacitadas de la escucha. Aporta en la navegación, a través del Sonar para fijar el rumbo de embarcaciones. De igual manera, hace aporte en la creación de espacios de realidad virtual, y en los campos de la seguridad ciudadana, teleconferencias, reconocimiento del habla y aplicaciones militares [56]. En este campo, Belloch, J. et al. en [57] demostraron un método para utilizar las GPU de bajo costo de los dispositivos móviles para la localización de eventos acústicos. En su implementación, utilizaron la GPU del dispositivo Samsung Galaxy 5 para la localización del sonido a través de un sistema de 12 micrófonos. Por parte de Kotus, J. et al. en [58], presentaron la determinación automática de una fuente de sonido en un espacio cerrado en tres dimensiones. Finalmente, Park, J. et al. en [59] obtuvieron la localización acústica de sonidos de disparos aplicando técnicas de diferencias de nivel interaural combinadas con máquinas de soporte vectorial.

2.2 Eventos Acústicos Estudiados en Apoyo a la Seguridad Ciudadana

En la Tabla 2.1 se presenta los eventos acústicos que se han reportado en diferentes estudios para su detección frente a situaciones que puedan significar un sinónimo de alerta o riesgo para la ciudadanía. En esta tabla se presentan las técnicas empleadas para la detección del evento acústico. Adicionalmente, para cada trabajo se especifica el nivel de implementación, haciendo referencia al tipo de dispositivo de cómputo en el cual se implementa la solución. Por último, se clasifica si el trabajo detecta un único

evento o múltiples eventos acústicos.

2.3 Aprendizaje Profundo

Los algoritmos de aprendizaje profundo basados en redes neuronales se originaron en la década de 1940, dando lugar a la primera ola de algoritmos de inteligencia artificial (AI - Artificial Intelligence). Inicialmente partieron de la creación del perceptrón de una sola capa (SLP - Single Layer Perception) y el perceptrón multicapa (MLP - Multilayer Perceptron) [61; 62]. Para mostrar la relación que existe entre AI, el aprendizaje de máquina (ML - Machine Learning), las redes neuronales y el aprendizaje profundo se ilustra el diagrama de Venn de la Figura 2.1.

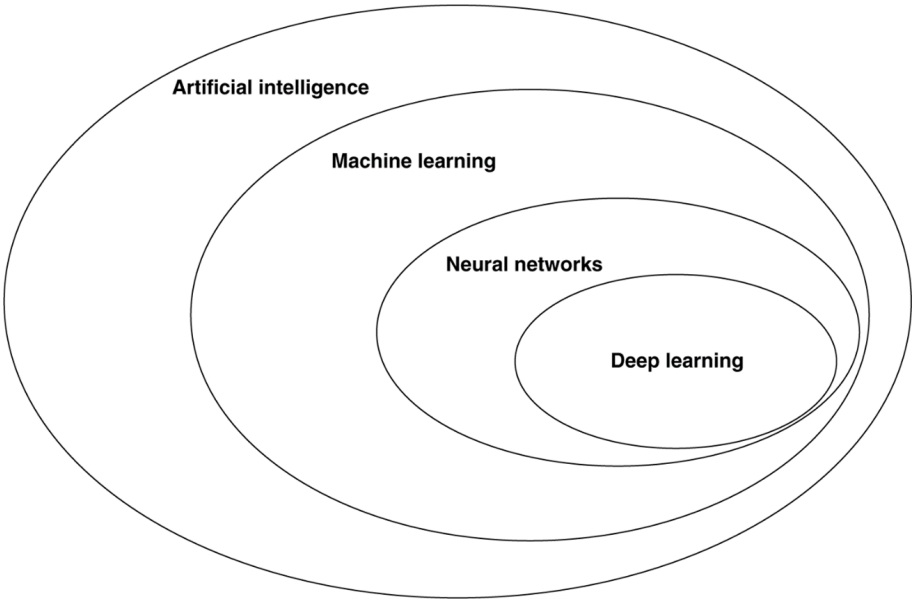

Figura 2.1: Relación entre la AI, el aprendizaje de máquina, las redes neuronales y el aprendizaje profundo.

Como se observa en la Figura 2.1, el aprendizaje de máquina es un subcampo de la inteligencia artificial. El aprendizaje profundo es un subcampo del aprendizaje de máquina, y las redes neuronales constituyen el eje principal de los algoritmos de aprendizaje profundo. En efecto, es el número de capas, o también llamada la profundidad de las redes neuronales, lo que distingue a una red neuronal simple (SLP o MLP) de una DNN, la cual debe tener más de tres capas [63]. El objetivo principal de estos algoritmos, es enseñar a la máquina a realizar predicciones y clasificaciones de datos a través de modelos estadísticos, en este caso, utilizando las redes neuronales artificiales (ANN Artificial Neural Networks). Estas redes, están inspiradas en la morfología de una red neuronal biológica, a la neurona en el caso de las ANN se le denomina perceptrón. En la Figura 2.2 se muestra de forma gráfica el modelamiento que recibe el perceptrón basado en las neuronas biológicas [64].

A continuación, se presentarán de manera general dos tipos de arquitecturas de redes neuronales profundas que han demostrado mejores resultados en la tarea de detección de eventos acústicos; estas son las redes neuronales convolucionales y las redes

Tabla 2.1: Principales eventos acústicos estudiados en la detección de situaciones de riesgo para la seguridad ciudadana

Evento Acústico	Referencia	Técnicas Usadas	Nivel de Implementación	Múltiples Eventos
Vehículos aéreos no tripulados (UAVs)	Jamil S et al. [18]	MFCC CNN	Servidor Local	X
	M. Z. Anwar et al. [19]	MFCC SVM	Servidor Local	X
Rompimiento de vidrios	Agarwal, S. et al. [30]	CNN	Servidor Local	✓
	Vozáriková, E. et al. [31]	MFCC HMM	Servidor Local	✓
Gritos humanos	Huang, W. et al. [5]	MFCC SVM	Servidor nivel Gateway	X
	Nanjo, H. et al. [6]	MFCC HMM	Servidor Local	X
	Park, J., & Kim, S. [7]	HMM	Servidor Local	X
Explosiones	Arslan [8]	LPC WLP	Servidor Local	✓
	Ntalampiras, S. et al. [9]	MFCC HMM	Servidor Local	✓
Disparos	Morehead, A. et al. [10]	CNN	Dispositivo de Borde	X
	Rahman, S. et al. [11]	SVM KNN	Servidor Local	X
	Freire, I. et al. [12]	MFCC HMM	Servidor Local	X
	Sigmund, M. [13]	MFCC DNN	Servidor Local	X
Sirenas	Arce, P. et al. [28]	CNN	Dispositivo de Borde	✓
	Pramanick, D. et al. [29]	FDM CNN	Servidor Local	X
Choque de automóviles	Li, Y. et al. [16]	MFCC BLSTM	Servidor Local	X
	Almaadeed, N. et al. [17]	MFCC SVM	Servidor Local	X
Intrusos/Ruido	Zieger, C.et al. [60]	SVF CSP	Dispositivo de Borde	X

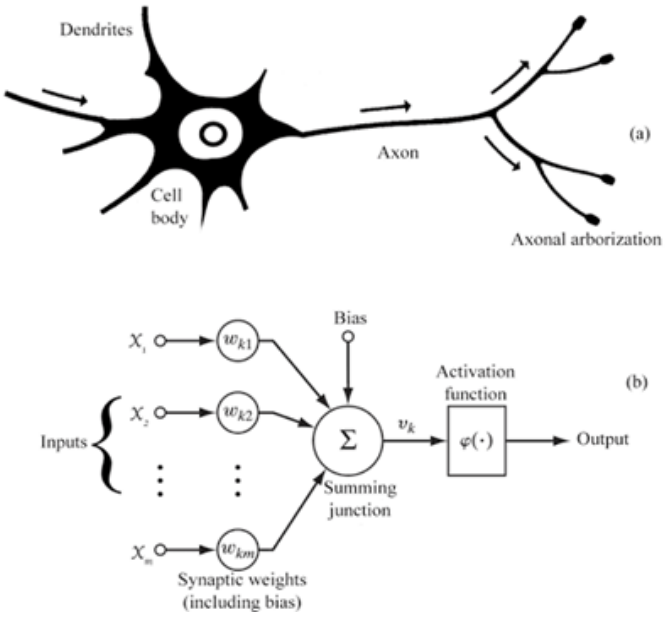

Figura 2.2: Modelamiento del Perceptrón (b), basado en la Neurona biológica (a).

neuronales recurrentes del tipo memoria a corto plazo (LSTM - Long Short Term Memory).

2.3.1 Redes Neuronales Convolucionales (CNN)

Esta arquitectura está diseñada para localizar, modelar y predecir con mayor exactitud los patrones presentes en datos de entrada como imágenes y videos. Son empleadas principalmente para la clasificación de imágenes, la autenticación de rostros y la segmentación semántica de imágenes. Las CNN utilizan una técnica de filtros que se deslizan de forma iterativa por pequeñas regiones de los datos de entrada, permitiendo trasladar propiedades de una región a una capa posterior de la red. En el caso de imágenes, los filtros o también llamados kernel son matrices cuadradas que se multiplican por segmentos cuadrados de los datos de entrada para extraer características presentes en ellos. Este proceso de mapeo de las características de los datos de entrada a un espacio de características diferente, se repite hasta la capa de salida de la red. En la capa de salida se obtienen valores de probabilidad correspondientes a las posibles clases de salida a las que pueden pertenecer los datos de entrada [10]. Una representación gráfica de esta arquitectura es presentada en la Figura 2.3 [65].

2.3.2 Redes Recurrentes de Memoria a corto plazo

Las redes neuronales recurrentes (RNN - Recurrent Neural Network) son un tipo de red neuronal artificial destinada a comprender patrones en datos continuos (datos de texto, datos de voz, secuencias de datos de sensores numéricos, datos de series temporales de finanzas, entre otras). Las redes neuronales recurrentes se diferencian de las redes neuronales convencionales DNN o CNN, en tener un lazo de retroalimentación, el cual está asociado a las salidas anteriores, así tomando las salidas como entradas para el siguiente instante de tiempo. Dentro de las RNN se encuentra el tipo LSTM, la

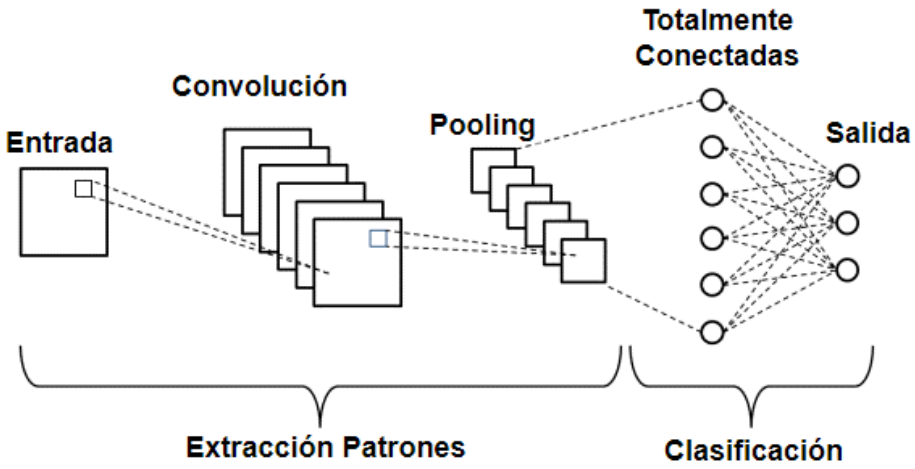

Figura 2.3: Estructura de una CNN.

arquitectura de una red LSTM es especialmente eficaz cuando se utiliza en una configuración de red profunda, ya que, resuelve el problema de desvanecimiento de coeficientes asociado al algoritmo de gradiente descendente de las redes neuronales recurrentes tradicionales. En la estructura LSTM, la capa oculta recurrente consiste en un conjunto de subredes conectadas denominadas bloques de memoria. Cada bloque de memoria incluye una o más neuronas de memoria auto conectadas y tres compuertas multiplicativas para controlar el flujo de información. En la Figura 2.4 se puede observar el diagrama de bloques de esta arquitectura de red [66]. En la primera compuerta, se enfoca en lo que se necesita olvidar de los datos (compuerta de olvido); en la segunda compuerta, la nueva información es almacenada en el estado de la neurona a lo largo de todo el proceso (compuerta de entrada); y en la compuerta final, se produce la nueva salida, basada en lo que se ha decidido (compuerta de salida) [67].

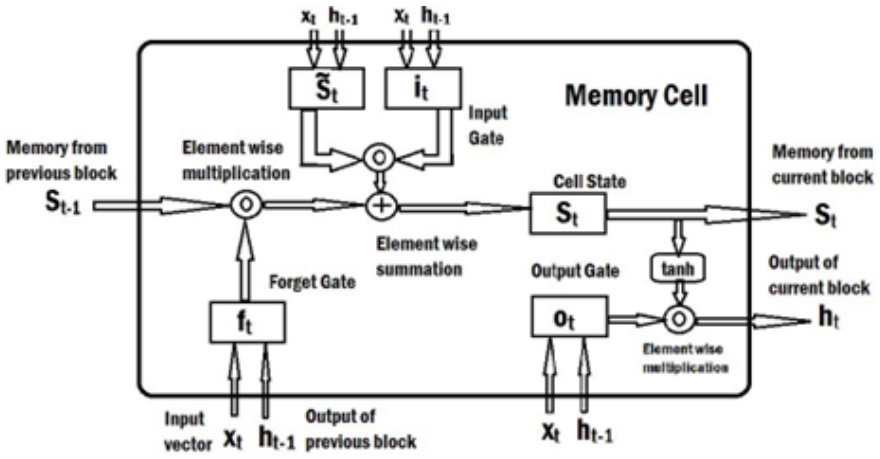

Figura 2.4: Estructura de una LSTM.

2.4 Ejecución eficiente de modelos DNN

En la actualidad, tanto el entrenamiento de una red profunda como la tarea de inferencia suelen realizarse en servidores y GPUs, normalmente ubicados en la nube. Sin embargo, tendencias tecnológicas como la computación en el borde optan por trasladar la inferencia de los modelos neuronales de la nube a dispositivos de cómputo cercanos a los sensores en el borde de la red. Lo anterior, para mitigar algunas desventajas del cómputo en la nube como baja eficiencia energética, altos costos, considerable latencia y riegos a la privacidad de los ciudadanos. Sin embargo, los dispositivos de borde se encuentran restringidos en capacidades de cómputo que dificultan la inferencia de redes neuronales profundas. Los recientes modelos de redes neuronales profundas para el procesamiento de imágenes o del habla requieren entre 100 giga-operaciones (GOP/s) a 1 tera-operaciones (TOP/s), así como la necesidad de consultar millones de parámetros de red (pesos del núcleo y sesgos) por cada evaluación de la red. La energía que se consume en estas numerosas operaciones y en la obtención de datos es el principal cuello de botella de la inferencia en el borde. Superar este cuello de botella es posible, pero requiere una estrecha interacción entre la optimización algorítmica (modificando la topología de la red) y la optimización del hardware (modificando las arquitecturas de procesamiento) [68]. Es por ello, que la investigación sobre el aprendizaje profundo eficiente en recursos ha sido estudiada desde diversas comunidades de investigación, incluidas las de aprendizaje de máquina, aritmética computacional y sistemas informáticos. Estas comunidades han centrado sus esfuerzos en optimizaciones a diferentes niveles. A nivel de modelo utilizando técnicas como la compresión [38; 40; 42] de los modelos de aprendizaje profundo. A nivel de implementación, desarrollando arquitecturas de aceleradores de hardware [69; 70] para la ejecución de los modelos. Y, en un nivel intermedio, se han propuesto técnicas de optimización a nivel aritmético [44; 45]. A continuación, se enmarcan las técnicas en los diferentes niveles, que han sido estudiadas para la ejecución eficiente en recursos de los modelos de aprendizaje profundo. Estas técnicas se han centrado en la optimización de un solo modelo.

2.4.1 Eficiencia a nivel de modelo

Las técnicas de eficiencia en recursos a nivel de modelo, tienen como objetivo reducir el tamaño de los modelos de las redes neuronales profundas, de tal manera que puedan ser adaptados a sistemas de recursos limitados, como IoT, dispositivos móviles, entre otros, incluidos dentro de los dispositivos de borde. Dentro de las técnicas a nivel de modelo se encuentran la cuantización de pesos (Weight Quantization), la poda (Pruning), la destilación del conocimiento (Knowledge Distillation), entre otras. La cuantización de pesos consiste en cuantizar los pesos de la red neuronal con un número menor de bits, en busca de reducir cantidad de memoria ocupada y no perder considerablemente la precisión de la red, [40; 41] muestran estudios recientes centrados en esta técnica. La técnica de poda se ha implementado para eliminar pesos [42; 43], neuronas [71; 72], filtros [73] o canales [74; 75] de la red neuronal que no son importantes o no aportan en la predicción del modelo. Con la técnica de poda se busca reducir el número de operaciones computacionales del modelo, sin reducir drásticamente la precisión del modelo. Por último, los artículos [76; 77] se han centrado en la destilación de co-

nocimiento. La destilación de conocimiento se enfoca en transferir el conocimiento de un modelo entrenado, de gran escala y con alto rendimiento, denominada red profesora (teacher network), a un modelo de red neuronal compacto, denominado red estudiante (student network). La destilación de conocimiento busca mejorar la precisión de una red con número de parámetros y número de operaciones reducidas.

2.4.2 Eficiencia a nivel aritmético

Las técnicas de eficiencia a nivel aritmético hacen uso de aritmética de menor precisión, para con ello reducir el tamaño en memoria y los tiempos de transferencias de datos entre buses e interconexiones. Estas técnicas dependen principalmente del tipo de formato aritmético utilizado, en las cuales han implementado formatos estándar de precisión media, simple, o doble de representación punto flotante [44]. Otros estudios, se han centrado en proponer formatos específicos fuera del estándar, es así como en [45], proponen el formato BFloat16 que consiste en un formato de punto flotante con 8 bits de exponente y 7 bits de mantisa, este formato es soportado específicamente por las arquitecturas de los procesadores Intel Cooper Lake Xeon y las unidades de procesamiento tensorial (TPU Tensor Processing Unit) de Google. Otro de los formatos propuestos está en [78], en este estudio proponen otro formato de punto flotante de 16 bits, lo nombraron DLFloat, consistiendo de 6 bits de exponente y 9 bits de parte significativa, buscando obtener un mejor rango dinámico de datos y precisión frente a los otros formatos.

2.4.3 Eficiencia a nivel de implementación

A nivel de implementación las técnicas se han centrado en mejorar la eficiencia energética y la velocidad computacional para ejecutar las operaciones del modelo. En [79] se propuso una arquitectura llamada Eyeriss. En esta arquitectura emplearon la técnica de clock-gating para suspender las operaciones de convolución, cuando se detectara que los pesos o activaciones del modelo son cero y así ahorrar potencia computacional. La técnica de salida temprana [80–82] (Early Exiting) también se ha propuesto dentro de este nivel. La salida temprana consiste en obtener la predicción o clasificación de salida del modelo tan pronto como sea posible, esto, a través de múltiples puntos de salida de un solo modelo. Este método minimiza el uso de los recursos de cómputo y la latencia. Por último, otra de las técnicas a este nivel, es el cambio de modelo en tiempo de ejecución (Model Switching). En [83], proponen la selección del mejor modelo que se adapte a la detección de objetos según el contenido dinámico que tenga el video de entrada, a través de un preprocesamiento de la entrada. De forma similar en [84], hacen la selección del modelo adecuado en tiempo de ejecución, pero en este caso, no basados en la entrada; sino en la carga computacional del sistema sobre el cual se ejecuta el modelo.

En el siguiente apartado se aborda la técnica de fusión de redes neuronales, la cual hace parte de las optimizaciones a nivel de modelo, se enfatizará en su definición y el estado del arte actual en el que se encuentra.

2.5 Fusión de Redes Neuronales

El termino fusión de redes neuronales no es utilizado en su sentido más amplio en la literatura de redes neuronales. Si bien es posible encontrar diversos trabajos con el término fusión multimodal, creemos que este es solo un caso particular del concepto general de fusión de redes neuronales. La fusión de redes neuronales es una técnica donde múltiples modelos neuronales se combinan para compartir una porción de la arquitectura del modelo final implementado, es decir una porción de los parámetros, de ahí el termino fusión. En este contexto, es posible establecer múltiples estrategias de fusión de acuerdo con el conjunto de características de entrada y el número de tareas que deba realizar la red neuronal fusionada.

- Un conjunto de características una tarea.
- Múltiples conjuntos de características una tarea.
- Un conjunto de características múltiples tareas.

2.5.1 Un conjunto de características una tarea

En esta estrategia un conjunto de características alimenta múltiples modelos todos realizando la misma tarea. Tiene como propósito adaptar el modelo predictivo a cambios dinámicos en los datos de entrada, en la capacidad de cómputo disponible, o en el tiempo disponible para realizar la tarea. Aproximaciones a este tipo de fusión de redes neuronales son las estrategias de salida temprana [80–82] y el intercambio de modelo [83; 84]. Si bien la salida temprana es vista como un único modelo, cada salida requiere de una capa softmax, lo cual convierte cada una de estas salidas en un modelo predictivo independiente de los demás. Cada salida tiene asociada una porción de los parámetros del modelo completo. Por otro lado, en la técnica de intercambio de modelos la fusión se hace efectiva en la toma de decisión acerca de cuál modelo debe actuar en un momento determinado. En este caso los modelos no comparten parámetros solo comparten la toma de decisiones.

2.5.2 Múltiples conjuntos de características una tarea

Este tipo de fusión emplea las características de múltiples sensores o fuentes de información complementaria que se fusionan para mejorar la predicción de una única tarea. La fusión multimodal es la estrategia de fusión de redes neuronales más reportada en la literatura [85–90]. La fusión multimodal ha sido utilizada en tareas como el reconocimiento de emociones y análisis del sentimiento, usando características de entradas de tipo imagen, video, audio y texto [91]. En recientes estudios como en [47], se ha utilizado la fusión de redes neuronales como método de optimización para la ejecución de múltiples redes neuronales, pues al evaluar el tamaño en memoria del modelo resultante, encontraron una buena proyección del espacio ocupado versus el disponible en los dispositivos de borde. La fusión multimodal de redes neuronales se clasifica según el nivel dentro de su arquitectura donde se realiza la fusión.

2.5.2.1. Fusión a nivel de entrada o características

En el nivel de características o enfoque de fusión temprana, las características extraídas de los datos de entrada se combinan primero y luego sigue el proceso de aprendizaje profundo estándar. Aquí, las características se refieren a algunas propiedades distinguibles de un flujo de datos. Por ejemplo, una unidad de fusión de características podría fusionar las características multimodales, como el color de la piel y las señales de movimiento, en un vector de características más grande, tomándose como entrada este vector para la unidad de detección de rostros con el fin de detectarse uno de los rostros en específico [87].

2.5.2.2. Fusión a nivel intermedio o híbrido

En la fusión de características de nivel intermedio, los datos de cada modalidad o conjunto de datos se introducen primero en las redes unimodales de mejor rendimiento. Así, independientemente aprenden las características intermedias de cada uno de los datos. Posteriormente, los pesos intermedios de estas redes unimodales se fusionan y se añaden en capas, usualmente, de tipo totalmente conectadas. El objetivo de este entrenamiento de la red es poder capturar las interacciones que existen entre los conjuntos de datos de entrada [87].

2.5.2.3. Fusión a nivel de decisión o tardía

La fusión de modelos a nivel de decisión (o fusión tardía) utiliza un clasificador independiente para ponderar las decisiones de las DNN unimodales. El propósito, es que la combinación de los resultados unimodales pueda mejorar la fiabilidad del modelo. La forma más directa de hacer la fusión a nivel de decisión es entrenar clasificadores o modelos separados y ponderar sus resultados. Estas ponderaciones pueden ser aprendidas por otro clasificador, o establecidas experimentalmente. En la fusión a nivel de decisión no se realiza ninguna concatenación, en comparación con la fusión a nivel intermedio, que utilizaba subredes para extraer características intermedias, en este caso, solo se obtiene la decisión de cada modelo [87].

En la Figura 2.5 se puede observar de manera gráfica los anteriores tipos de fusión descritos según el nivel de fusión [92].

Las técnicas de fusión de redes neuronales no solo dependen de los niveles en el que se fusionen los modelos, sino también, de cuál será el patrón o método utilizado para realizarse la combinación o mezcla de las características o pesos de cada modelo. Dentro de estos métodos se encuentran la suma de pesos, el promedio, efectuándose mediante una suma ponderada de los pesos, el máximo, a través de la elección de las características con máximo valor en sus pesos, la concatenación, que apila las características según los diferentes canales de las capas de la red, y la fusión bilineal, que realiza un producto matricial entre pares de características. Los anteriores son los principales métodos que se han implementado a lo largo de las investigaciones [92].

Los estudios recientes en esta rama investigativa se han centrado en presentar diferentes combinaciones de técnicas en problemas específicos, algunos otros, han mos-

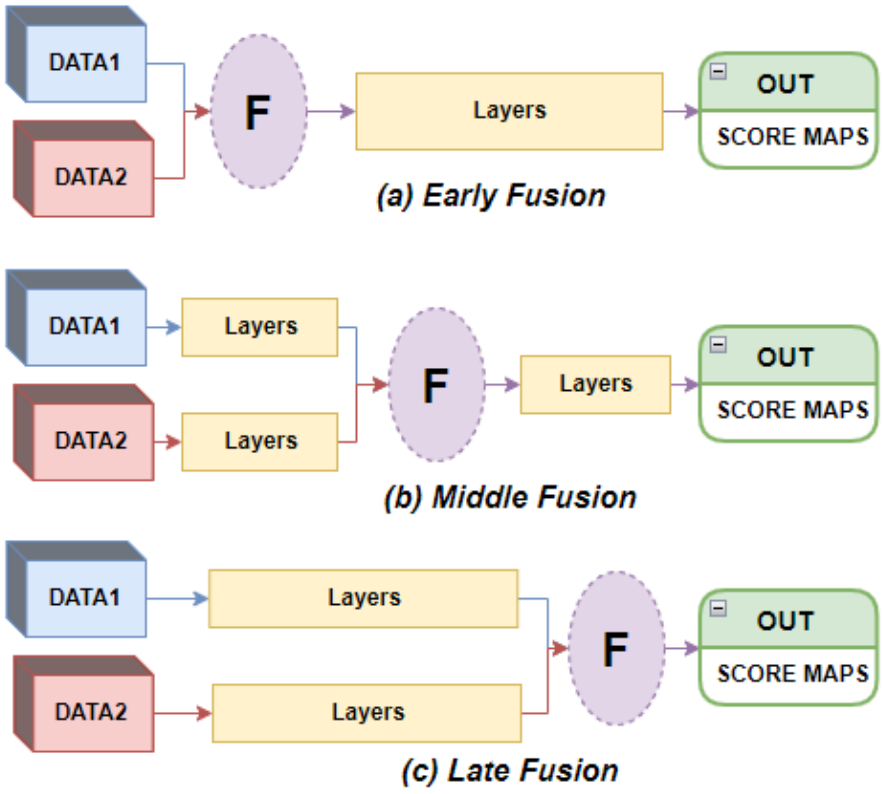

Figura 2.5: Fusiones según los diferentes niveles.

trado nuevos métodos de fusión basados en arquitecturas específicas de sus modelos. Es así como Choi, J. & Lee en [85] presentaron un método para la fusión de los modelos nombrado EmbraceNet, que realiza la fusión a un nivel intermedio, mediante el cual realizan una reducción intermedia del tamaño de los canales de sus capas para posteriormente realizar la combinación de los pesos, la realizan de manera enfocada a la clasificación multimodal de gestos humanos y reconocimiento de la actividad humana, sus resultados mostraron mayor robustez de predicción ante alguna falta de los datos de entrada multimodal. Por otro lado, en [86] Rachmadi et al. investigaron el rendimiento de fusiones de redes neuronales convolucionales a niveles temprano y tardío para el problema de verificación de parentesco basado en imágenes, en ambos de los modelos obtuvieron una precisión mayor al 60 % y con el método que mayor precisión obtuvieron fue con la fusión tardía, obteniendo un 62.66 %. Williams et al. en [87] evaluaron los tres niveles de fusión de modelos de redes neuronales para la predicción de sentimientos a través de entradas de audio, video y texto, con ello demostraron que obtuvieron mejores resultados de precisión de 72.4 %, 74 %, 72.5 % en nivel temprano, intermedio y tardío, respectivamente, versus los resultados de clasificación unimodal.

2.5.3 Un conjunto de características múltiples tareas

Es común encontrar fuentes de información que sirven como entrada a múltiples modelos neuronales. Por ejemplo, una cámara de video en la calle puede servir para detectar transeúntes, identificar placas de automóviles, estimar congestión vehicular,

identificar personas por su forma de caminar, etc. En este contexto, podemos definir el tercer tipo de fusión de redes neuronales, donde a partir de un conjunto de características de entrada se alimentan múltiples modelos neuronales. La estrategia aquí consistirá en compartir los parámetros de las primeras capas del modelo neuronal que permiten detectar características generales de la entrada, para luego especializar el modelo de acuerdo a las tareas objetivo. Técnicas reportadas en la literatura dentro del marco de este tipo de fusión de redes neuronales incluyen: aprendizaje multitarea (MTL - Multi-Task Learning) [93], transferencia de aprendizaje (transfer learning) [94], aprendizaje multi-etiqueta (multi-label learning) [95], y regresión multi-salida (multi-output regression) [96]. El aprendizaje multitarea busca principalmente aprender múltiples tareas conjuntamente, tal que el conocimiento o aprendizaje adquirido por una de las tareas pueda ser aprovechado por las demás. Así, el aprendizaje multitarea resulta más efectivo pues utiliza más datos de diferentes tareas de aprendizaje, comparado con el aprendizaje de una sola tarea. Por otro lado, la transferencia de aprendizaje, se diferencia del aprendizaje multitarea en que en este último el objetivo es mejorar el rendimiento de todas las tareas, mientras que, en la transferencia de aprendizaje el objetivo es mejorar el rendimiento de una tarea en específico con la ayuda del aprendizaje de las demás tareas. Por último, en el aprendizaje multi-etiqueta y la regresión multi-salida cada dato se encuentra asociado con múltiples etiquetas, que pueden ser categóricas o numéricas, dependiendo el caso. Y el objetivo principal en estos casos, es predecir simultáneamente las múltiples etiquetas o valores objetivo. Una variante posible en esta categoría de fusión de redes neuronales es utilizar modelos entrenados en el mismo conjunto de características y buscar redundancias entre las correspondientes capas de los modelos buscando reducir la complejidad computacional o intentando mejorar la precisión de las tareas. En [46] Mallouh et al. realizaron la fusión de dos modelos de redes neuronales entrenados de forma independiente para la detección de la edad del hablante y otro para la detección del género del hablante. La fusión consistió en concatenar las características de la última capa de cada modelo y agregar softmax independientes para cada modelo, pero alimentadas por las características concatenadas. El resultado fue un incremento de 8 puntos porcentuales en la precisión del modelo fusionado, respecto a los modelos independientes. Este trabajo no explota redundancias entre modelos. No fue posible encontrar trabajos adicionales en esta variante de fusión de redes neuronales. Sin embargo, pensamos que puede tener un gran potencial en el ámbito de la computación en el borde en caso de demostrar una reducción significativamente en el costo computacional del modelo fusionado respecto al costo de ejecutar los modelos independientes de forma simultánea.

Capítulo 3

Modelos AER para Disparos, Sirenas y Gritos

En línea con la tendencia de la comunidad del AER, se ha observado una adopción creciente de modelos basados en técnicas de aprendizaje profundo. Esta evolución responde a su éxito en tareas de procesamiento de lenguaje natural y visión por computadora, destacando por su capacidad para alcanzar niveles de precisión superiores en comparación con los clasificadores tradicionales. En la literatura reportada [97], se documenta el uso de múltiples arquitecturas de DNN y sus combinaciones para la extracción de características y la clasificación de los eventos.

Resulta oportuno mencionar tres arquitecturas que se han utilizado para esta tarea. Kong et al. [33] implementaron un DNN tradicional completamente conectada para la detección de eventos acústicos. Por su parte, Chou et al. [34] propusieron una estrategia basada en CNN para la detección de eventos acústicos. La tercera arquitectura, centrada en CRNN, fue empleada por Cakir et al. [35] específicamente en la detección de sonidos de aves.

Adicionalmente, algunos autores han explorado la relación entre el número de MFCC utilizado como extractor de características del sonido y los modelos basados en CNN. Jiang et al. lograron un valor promedio de puntuación F1 del 97.9 % en el reconocimiento de eventos acústicos relacionados con la monitorización de tuberías de gas y petróleo usando 12 MFCC y una arquitectura de CNN de dos capas [98]. Cao et al., por su parte, encontraron que la mejor combinación para la clasificación de sonidos urbanos fue MFCC+CNN, empleando 40 MFCC [99]. Otros estudios han evaluado la influencia de parámetros de los MFCC, incluyendo el tipo de ventana y el número de coeficientes [100], [101].

Dado que no se encontraron disponibles públicamente modelos a gran escala específicamente en el ámbito del reconocimiento de eventos acústicos urbanos con enfoque en la seguridad ciudadana, y considerando que nuestros objetivos incluyen la ejecución en dispositivos de borde, donde las numerosas dimensiones de estos modelos presentan desafíos significativos, hemos optado por crear nuestros propios modelos. Esta elección se fundamenta en la necesidad de adaptar los modelos a nuestras necesidades y restricciones, además de asegurar su eficiencia en términos de consumo de recursos y latencia en la inferencia.

3.1 Selección de Eventos

En el marco de nuestro enfoque en la seguridad ciudadana, la identificación y análisis de eventos acústicos adquieren relevancia. Estos eventos, en forma de sonidos distintivos, a menudo encierran información valiosa que puede ser utilizada para prevenir incidentes y responder de manera eficaz a situaciones peligrosas. En los entornos urbanos existe una variedad de señales acústicas utilizadas con fines de advertencia temprana para proteger vidas, integridad y seguridad, destacan los gritos humanos, explosiones, disparos, llantos de bebés y choques de automóviles. Además, las señales acústicas se han aplicado para estudiar el entorno y detectar situaciones específicas, como el ruido de vehículos aéreos no tripulados (UAV), la lluvia, bocinas de automóviles y cantos de aves. También se han empleado señales como ladridos de perros, alarmas, sirenas y cristales rotos para identificar situaciones peligrosas y delitos contra la propiedad ciudadana. Así, nos centramos en identificar y analizar eventos acústicos específicos que tienen un impacto significativo en entornos urbanos. Estos eventos no solo ofrecen información valiosa para la detección temprana de situaciones peligrosas, sino que también contribuyen a la prevención de incidentes y al bienestar de la comunidad. Hemos seleccionado tres categorías principales de eventos acústicos que son de particular interés para nuestros análisis: **disparos, gritos y sirenas.**

La elección de estos eventos se basa en su relevancia crítica en el ámbito de la seguridad ciudadana y en su identificación clara y distintiva en grabaciones de audio. Los disparos, como indicadores potenciales de situaciones de peligro, los gritos, que pueden señalar emergencias o agresiones, y las sirenas, que advierten sobre la presencia de vehículos de emergencia, representan situaciones de alto impacto que requieren una respuesta inmediata y precisa. Al enfocarnos en estos eventos específicos, buscamos proporcionar herramientas efectivas para la identificación y clasificación en escenarios urbanos, que contribuyen a la protección de la seguridad y bienestar de los ciudadanos.

3.2 Construcción del Dataset

El entrenamiento de nuestros modelos de AER se fundamentó en la utilización de conjuntos de datos relevantes y adecuados para la tarea. Principalmente, empleamos el conjunto de datos UrbanSound8k [102], el cual organiza una variedad de sonidos urbanos en diez categorías, tales como aire acondicionado, bocina de automóvil, juego de niños, taladro, ladrido de perro, motor, disparo, martillo neumático, sirena y música callejera. Este conjunto de datos está compuesto por 8732 archivos de audio, cada uno con una duración de hasta 4 segundos, y grabados con una frecuencia de muestreo de 22.05 KHz.

Para el entrenamiento del modelo AER destinado al reconocimiento de sirenas, equilibramos el número de muestras utilizado en relación al conjunto de datos UrbanSound8k para nuestra clase de interés, el sonido de sirenas. De manera equitativa y aleatoria, etiquetamos las otras nueve categorías restantes en el conjunto de datos como eventos no de interés. El número total de muestras equilibradas utilizadas para crear el modelo fue de 929 para eventos de sirenas y 929 para eventos de no interés.

En el caso del modelo AER para la detección de disparos, enriquecimos el conjunto de datos base con sonidos de explosiones en la clase de no interés, con el propósito de

aumentar la robustez del modelo. Esto, teniendo en cuenta que existen sonidos impulsivos como fuegos artificiales, explosiones de bombas, explosión de transformadores eléctricos, entre otros, que pueden darse en entornos urbanos y podrían causar falsas detecciones de los eventos de interés que son los disparos. Para lograrlo, extrajimos sonidos de explosiones del conjunto de datos Sound Events for Surveillance Applications (SESA) [103], que cuenta con sonidos explosivos en entornos urbanos. Los sonidos recién agregados fueron normalizados a una duración máxima de 4 segundos. Posteriormente, realizamos el mismo proceso de equilibrio que en el caso de los sonidos de sirenas, seleccionando la clase de disparos en UrbanSound8k como la clase de interés y eligiendo muestras equilibradas y aleatorias de las clases restantes para usarlas como eventos de no interés, incluyendo las explosiones. Empleamos 748 muestras para crear el modelo de reconocimiento de disparos, con 374 muestras para el evento de interés y 374 para eventos de no interés.

Finalmente, para el entrenamiento del modelo AER destinado a reconocer gritos, utilizamos un conjunto de datos de gritos humanos disponible en la plataforma Kaggle [104]. Llevamos a cabo la limpieza y el preprocesamiento de los sonidos que contribuyen a la construcción del modelo, con una duración máxima normalizada de 4 segundos para cada una de las muestras de interés. Siguiendo el enfoque de modelos anteriores, empleamos las categorías del conjunto de datos UrbanSound8k como eventos de no interés y las seleccionamos de manera equilibrada y aleatoria. Utilizamos 782 muestras para crear el modelo de reconocimiento de gritos, con 391 muestras equilibradas para el evento de interés de los sonidos de gritos y 391 para eventos de no interés.

La elección y preparación de estas bases de datos se realizó con el propósito de desarrollar modelos AER precisos y confiables para la identificación de eventos acústicos específicos en entornos urbanos asociados a la seguridad ciudadana. En la Tabla 3.1 se agrupan las clases utilizadas para cada evento acústico, las bases de datos utilizadas y el número de muestras.

3.3 Selección de Arquitectura CNN

Diversos estudios han empleado arquitecturas de redes neuronales convolucionales para llevar a cabo la clasificación y detección eficiente de eventos acústicos. Algunas de estas arquitecturas se han desarrollado basándose en investigaciones y resultados de los desafíos DCASE [105; 106]. Por lo general, estas arquitecturas constan de varios bloques de capas convolucionales, capas de activación que utilizan la función de unidad lineal rectificada (ReLU), capas de pooling y regularización mediante dropout.

Tsalera et al. en [36] evaluaron arquitecturas de gran escala de convoluciones, como GoogLeNet, SqueezeNet, ShuffleNet, VGGish y YAMNet, utilizando el conjunto de datos UrbanSound8k. La arquitectura VGGish obtuvo el mejor rendimiento con una precisión del 96.7 %. Sin embargo, VGGish requiere alrededor de 360.72 MFLOPS (Millones de Operaciones en Punto Flotante) para la inferencia y clasifica múltiples sonidos simultáneamente. Dado que nuestro objetivo es encontrar modelos de redes neuronales convolucionales con requisitos computacionales más bajos, una arquitectura a gran escala como VGGish no es la mejor elección para reconocer un único evento acústico. Por lo tanto, algunos artículos, como [107], han presentado modelos reducidos basados en arquitecturas similares a VGGish para reconocer un conjunto más pequeño

Tabla 3.1: Base de datos para detección de eventos acústicos seleccionados relacionados a la seguridad ciudadana

Evento Acústico	**Base de Datos**	**Clases**	**Número de Muestras**
Disparo	UrbanSound8k[102]	Disparos	374
No-Disparo	UrbanSound8k[102]	Aire acondicionado, bocina, niños jugando, ladridos de perros, perforaciones taladro, sonido de motor, martillo neumático, sirenas, música de calle	374
	Sound Events for Surveillance Applications (SESA)[103]	Explosiones	
Sirena	UrbanSound8k[102]	Sirenas	929
No-Sirena	UrbanSound8k[102]	Aire acondicionado, bocina, niños jugando, ladridos de perros, perforaciones taladro, sonido de motor, disparos, martillo neumático, y música de calle	929
Grito	Screams Dataset Kaggle[104]	Gritos	391
No-Grito	UrbanSound8k[102]	Aire acondicionado, bocina, niños jugando, ladridos de perros, perforaciones taladro, sonido de motor, disparos, martillo neumático, sirenas, y música de calle	391

de eventos acústicos.

En nuestras experimentaciones, tomamos como referencia la arquitectura convolucional con activación ReLU presentada en [107]. Los autores evaluaron directamente esta arquitectura en el conjunto de datos UrbanSound8k para reconocer las diez clases del conjunto y obtuvieron un rendimiento del modelo con una precisión del 91 %. La arquitectura de referencia se compone de 4 bloques convolucionales, en los cuales cada bloque consta de una capa convolucional 2D con un tamaño de kernel de (2x2), seguida de una activación ReLU, un max pooling con un tamaño de pool de (2x2) y una regularización dropout de 0.2. Los filtros convolucionales aumentan para cada bloque: 16, 32, 64 y 128. Finalmente, se incluye una capa de pooling promedio global, seguida de una capa aplanada conectada a una capa densa con activación softmax. La Figura 3.1 ilustra la arquitectura general utilizada para cada uno de nuestros modelos.

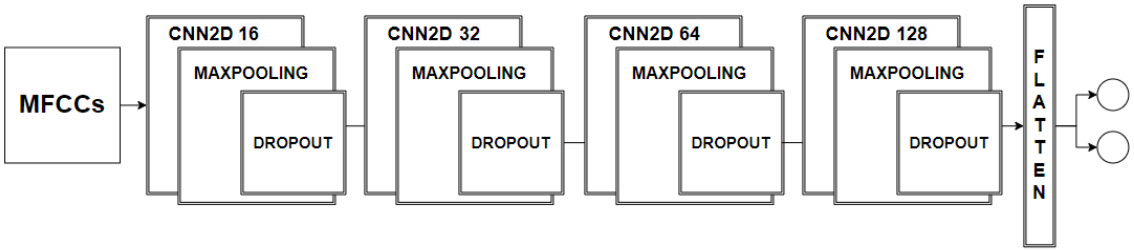

Figura 3.1: Arquitectura CNN.

3.4 Extracción de Características del Sonido

Las características de coeficientes cepstrales en frecuencia Mel (MFCC) han sido ampliamente utilizadas debido a su efectividad comprobada en la descripción de las características y estructuras de audio. Este método ha sido de gran utilidad en el reconocimiento automático del habla, la recuperación de información musical, la recuperación de sonidos ambientales, y la detección, clasificación y reconocimiento de eventos acústicos.

En el contexto de los sonidos, su representación a través de los coeficientes MFCC encapsula las características principales del sonido. Los MFCC nos permiten formar matrices bidimensionales que pueden interpretarse como imágenes. Y así, aprovechar las ventajas de las redes neuronales convolucionales para el procesamiento de este tipo de datos. La eficacia de las redes neuronales convolucionales (CNNs) en el AER ha quedado demostrada en los resultados de los desafíos de la comunidad de Detección y Clasificación de Escenas y Eventos Acústicos (DCASE) de los últimos diez años [108; 109]. Estos desafíos han utilizado características extraídas del sonido mediante técnicas como los MFCC [110; 111].

El proceso de cálculo de los MFCC se desarrolla de la siguiente manera [98; 112]:

1. En primer lugar, dividimos la señal en tramas cortas mediante segmentación y aplicación de ventanas.
2. A continuación, calculamos el espectro de potencia utilizando la transformada rápida de Fourier (FFT) de cada ventana.
3. El espectro de potencia se transforma a la escala Mel mediante la aplicación de una bancada de filtros Mel según la Ecuación 3.1.
4. Se calcula la energía total de cada bancada de filtros y se obtiene su logaritmo.
5. Luego, aplicamos la transformada discreta del coseno (DCT) y extraemos los coeficientes MFCC.

Finalmente, organizamos cada ventana de tiempo de MFCC en una matriz. La Figura 3.2 muestra un diagrama de este proceso [113; 114].

La escala Mel está definida por la ecuación:

$$f_{mel} = 2595 * log_{10}(1 + \frac{f}{700}) \tag{3.1}$$

donde f_{mel} es la escala logarítmica de la frecuencia normal representada por f.

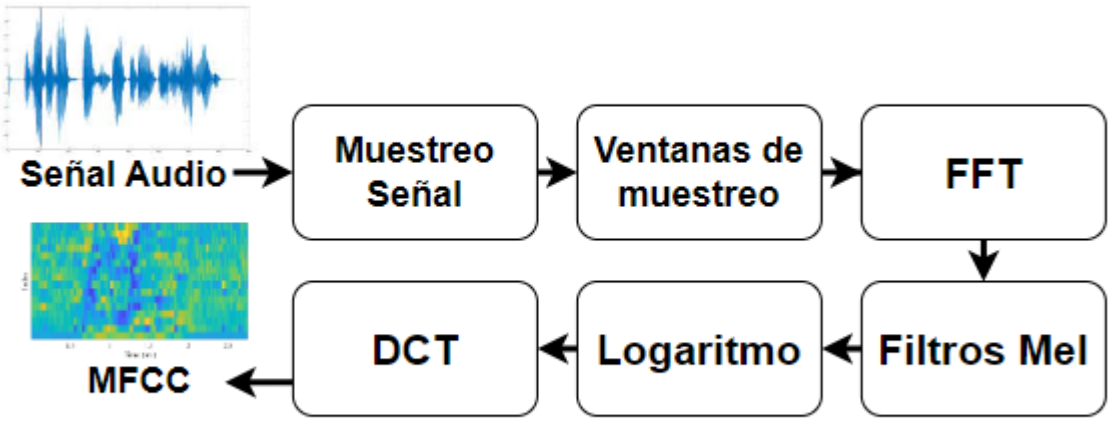

Figura 3.2: Proceso extracción de características del audio MFCC.

En la búsqueda de optimizar la extracción de características de los Coeficientes Cepstrales de Frecuencia de Mel para el reconocimiento de eventos acústicos urbanos, encontramos investigaciones previas relacionadas con la influencia de los parámetros de los MFCC en tareas similares.

A. Jain y O. Sharma [100] investigaron la influencia del tipo de ventana utilizada para el muestreo en la extracción de MFCC para la identificación de voz. Evaluaron ventanas como Hamming, Hanning, Blackman, Kaiser y Rectangular, obteniendo resultados prometedores con la ventana Hamming. En otro estudio realizado por M. Sadeghi y H. Marvi [101], se encontró que es posible describir características de audio con tan solo 5 MFCCs, lo que resultó en una buena precisión en la identificación del hablante. Además, A. Benba et al. [115] investigaron el comportamiento del número de coeficientes Cepstrales de Frecuencia de Mel con respecto al tipo de kernel utilizado en un clasificador SVM para clasificar pacientes con enfermedad de Parkinson.

A pesar de estos hallazgos, a nuestro conocimiento, no existen estudios específicos que evalúen el rendimiento de modelos convolucionales que realicen una búsqueda exhaustiva de hiperparámetros en la extracción de características de MFCC, especialmente en el contexto de eventos acústicos urbanos. Por lo tanto, decidimos realizarlo.

3.5 Métricas de Evaluación de Desempeño

En este capítulo, abordamos las métricas que empleamos para evaluar el desempeño de nuestros modelos de reconocimiento de eventos acústicos urbanos. Estas métricas son esenciales para comprender cómo nuestros modelos se desempeñan en la tarea de clasificar eventos acústicos urbanos y su complejidad computacional.

3.5.1 Almacenamiento

Una consideración importante en nuestro análisis es la gestión del almacenamiento, particularmente en dispositivos de borde con recursos limitados. Aquí evaluamos dos aspectos:

Número de Parámetros: Esta métrica revela la complejidad del modelo. Un alto número de parámetros indica una mayor complejidad, lo que a menudo conlleva una mayor necesidad de almacenamiento y recursos computacionales. En esencia, representa el tamaño del modelo y, por ende, su demanda en términos de espacio y potencia de procesamiento.

Peso del Modelo [Bytes]: Esta métrica cuantifica la cantidad exacta de memoria necesaria para almacenar el modelo completo. Es una medida precisa y directa del espacio de almacenamiento que el modelo ocupará en un dispositivo específico. Comprender el peso del modelo es esencial para garantizar su implementación eficiente en dispositivos con limitaciones de almacenamiento.

3.5.2 Precisión de Modelo

Para evaluar el rendimiento de los modelos de aprendizaje automático en el reconocimiento de eventos acústicos urbanos, es necesario métricas adecuadas que puedan medir con exactitud la capacidad del modelo para clasificar los diferentes eventos.

F1-Score [%]: Comúnmente la métrica de Accuracy es utilizada para la evaluación de los modelos, pero esta puede ser engañosa cuando el conjunto de datos tiene un desequilibrio entre las clases. Por lo tanto, también se deben considerar otras métricas, como la Precisión, la Exhaustividad (Recall) y la puntuación F1, que brindan una evaluación más completa del rendimiento del modelo. La puntuación F1 es una métrica crítica que equilibra la Precisión y la Exhaustividad, proporcionando una evaluación más precisa del rendimiento del modelo en el reconocimiento de eventos acústicos urbanos [116]. En este estudio, utilizamos la puntuación F1 para evaluar nuestros modelos de CNN en el reconocimiento de eventos acústicos urbanos, asegurando que los modelos sean efectivos en la identificación de eventos acústicos en entornos urbanos.

3.5.3 Complejidad Computacional

La complejidad computacional de nuestros modelos de reconocimiento de eventos acústicos urbanos es una métrica esencial para evaluar su eficiencia en términos de recursos de cómputo requeridos.

Operaciones de Punto Flotante [FLOPS]: Una métrica comúnmente utilizada para evaluar la complejidad computacional de las redes neuronales profundas son las operaciones de punto flotante, abreviadas como FLOPS (del inglés "Floating-Point Operations"). FLOPS estima la cantidad de operaciones de punto flotante necesarias para realizar la inferencia de la red. Esta métrica ayuda a comparar la eficiencia de diferentes

arquitecturas de redes neuronales y determinar los requisitos computacionales para la inferencia. También, independiente de la plataforma, refleja los recursos computacionales necesarios para ejecutar un modelo DNN [117].

Tiempo de Ejecución [s]: El tiempo de ejecución se refiere al período necesario para procesar una tarea específica. En el contexto de nuestro estudio, representa el tiempo necesario para que nuestros modelos de reconocimiento de eventos acústicos urbanos realicen la clasificación de eventos en datos de audio. La variación en el tiempo de ejecución se expresa mediante el delta o diferencial de tiempo, que proporciona una visión de cómo es el rendimiento del modelo en nuestro escenario de uso.

Utilización de Memoria en Ejecución [MB]: La utilización de memoria en ejecución, expresada en megabytes (MB), es una métrica para evaluar la eficiencia de nuestros modelos durante la fase de inferencia. Representa la cantidad de memoria RAM que nuestros modelos requieren mientras se ejecutan para llevar a cabo la tarea de clasificación de eventos acústicos urbanos. Esta métrica es esencial para asegurar que nuestros modelos sean compatibles con dispositivos de borde con limitaciones de memoria.

En conjunto, estas métricas nos permiten evaluar de manera exhaustiva y coherente el desempeño de nuestros modelos de reconocimiento de eventos acústicos urbanos, asegurando que sean eficientes y precisos, lo que es fundamental para su despliegue en dispositivos de borde con recursos limitados.

3.6 Diseño Experimental

Nuestra elección de realizar esta búsqueda, está fundamentada en la necesidad de encontrar una configuración que conduzca a modelos con un mayor desempeño en precisión para nuestras tareas de detección de eventos de interés.

Utilizamos la biblioteca Librosa de Python [118] para la extracción de características de audio mediante MFCCs. Esta biblioteca nos proporcionó las herramientas necesarias para extraer los MFCC de los archivos de audio y ajustar parámetros clave para adaptarlos a nuestra aplicación.

Nuestro experimento explora hiperparámetros vinculados a la extracción de los MFCC. A continuación presentamos los hiperparámetros incluidos en la exploración del espacio de diseño y los valores que pueden tomar.

- La longitud de la Transformada Rápida de Fourier (FFT) (Nfft) con valores de 256, 512, 1024, 2048 y 4096 muestras.
- El tamaño de la ventana (NwinL), en que será seccionado el audio y aplicado el proceso, con valores de 256, 512, 1024, 2048 y 4096 muestras.
- La longitud del paso de ventana entre muestras sucesivas (NhopL), el contrario de solapamiento entre ventanas (100 % - NhopL), con variaciones del 25 %, 50 %, 75 % y 100 %.

- El número de coeficientes Cepstrales de Frecuencia de Mel (Nmfcc) en un rango de 3 a 45 con un incremento de 3.
- Además, exploramos un hiperparámetro clave del modelo de Red Neuronal Convolucional (CNN), el tamaño del kernel o filtro utilizado en las capas de convolución (Ksize), con valores de 2, 3, 5 y 7.

La tasa de muestreo de audio (sr) equivalente a la utilizada en el conjunto de datos de audio, de 22,05 KHz. Utilizamos la ventana de Hanning como el tipo de ventana predeterminado, ya que en [119] se reporta como la ventana con mejor desempeño de manera general, sin estar vinculada a una aplicación específica.

Esta combinación de cinco hiperparámetros resultó en un total de 3600 posibles modelos, excluyendo las configuraciones carentes de sentido donde el tamaño de la FFT es mayor que el tamaño de la ventana.

Para cada uno de estos modelos se utilizó un 80 % de los datos para entrenamiento y un 20 % para evaluación. Utilizamos la función de pérdida de entropía cruzada categórica (categorical crossentropy) y el optimizador Adam. El entrenamiento se realizó durante un máximo de 80 épocas, con un mecanismo de parada temprana (early stopper) configurado para prevenir el sobre ajuste, permitiendo una paciencia de 10 épocas. Para reducir el impacto de la inicialización aleatoria de los pesos del modelo de CNN, repetimos este proceso de entrenamiento y evaluación 5 veces para cada modelo y calculamos el promedio aritmético de los resultados en F1-Score.

En resumen, llevamos a cabo un entrenamiento y evaluación de un total de 54,000 modelos, abordando tres eventos de sonido, con 3,600 modelos por evento y cinco ejecuciones por modelo. Este enfoque nos permitió identificar la configuración óptima de hiperparámetros para la extracción de características MFCC en nuestro contexto específico de reconocimiento de eventos acústicos urbanos.

3.7 Resultados

En esta sección, presentamos los resultados obtenidos durante nuestros experimentos de búsqueda de hiperparámetros para la extracción de características MFCC en nuestros modelos de reconocimiento de eventos acústicos urbanos. Los resultados son presentados en las subsecciones 3.7.1, 3.7.2 y 3.7.3, para cada evento de interés Disparos, Sirenas y Gritos, respectivamente.

Para esto se presentan los gráficos de Pareto. Este gráfico incluye todos los resultados de las diferentes configuraciones experimentales de hiperparámetros MFCC realizadas para cada tipo de sonido, identificados como puntos azules. En el eje X se representa el número de FLOPS requeridos para la inferencia de cada modelo en orden ascendente, mientras que en el eje Y se muestra la métrica F1-Score para cada experimento. Para una mejor visualización de la frontera de Pareto, la métrica F1-Score se presenta en orden inverso ($100 - F1Score$). Destacamos en rojo los puntos que pertenecen a la frontera de Pareto en los gráficos.

En las Tablas 3.2, 3.3, y 3.4 mostramos los resultados de la frontera de Pareto obtenidos para cada modelo entrenado. Que corresponden a esas configuraciones en las cuales se logra maximizar el desempeño del modelo en F1-Score o se logra minimizar la métrica de FLOPS del modelo.

Finalmente, evaluamos la correlación entre los hiperparámetros y el rendimiento del modelo al controlar los efectos de otras variables. En este caso, el coeficiente de correlación tradicional de Pearson no es adecuado, ya que podría no ser capaz de detectar relaciones no lineales o controlar los efectos de variables de confusión. En su lugar, se optó por utilizar el coeficiente de correlación parcial de Spearman, que mide la relación entre dos variables mientras controla los efectos de otras variables [120]. Este enfoque posibilita una evaluación más precisa y detallada de la influencia de los hiperparámetros en el rendimiento del modelo, lo cual permite una comprensión completa de su comportamiento en el contexto del reconocimiento de los eventos acústicos urbanos de interés.

3.7.1 Disparos

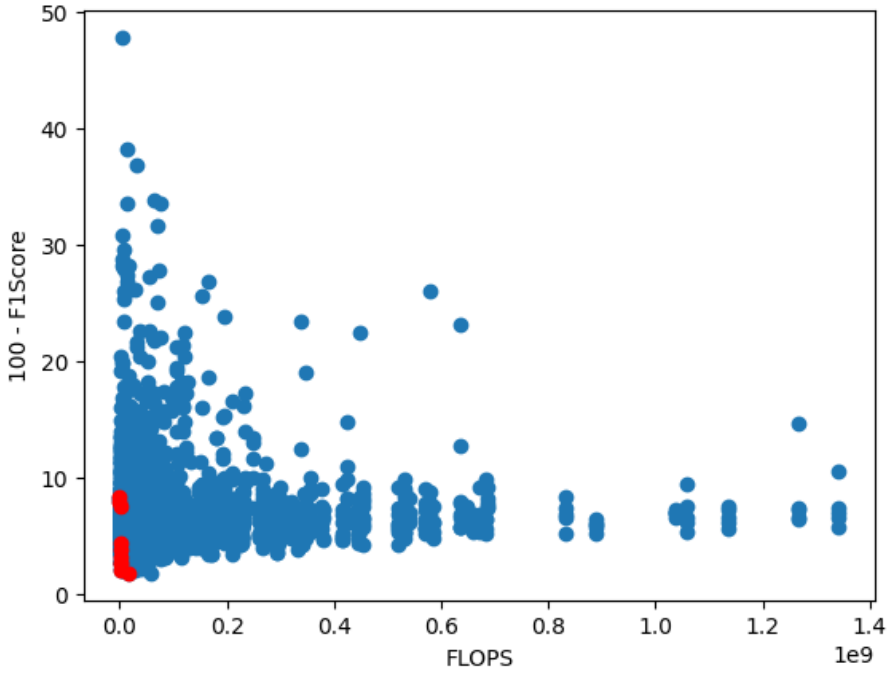

Figura 3.3: Frontera de Pareto en Rojo para las Configuraciones de Hiperparámetros MFCC para el modelo de Reconocimiento de Sonido de Disparos

La Tabla 3.2 muestra las configuraciones óptimas de Pareto para el reconocimiento de eventos de disparos. Una característica común en todas estas configuraciones óptimas es que la longitud de la FFT y de la ventana se encuentra en el valor máximo de 4096 muestras. Esto indica que un espectrograma de mayor resolución y una mínima distorsión de la ventana mejoran el rendimiento del modelo. En el espaciado entre muestras sucesivas se observa que mejoran el desempeño valores máximos de 75 % y 100 % de espaciado. No pudimos identificar un patrón relacionado con el número de coeficientes MFCC y el tamaño del kernel. Sin embargo, obtuvimos un buen rendimiento de hasta un 96.2 % de F1-Score con 3 coeficientes MFCC y tamaños de kernel de 2 y 3 sin aumentar significativamente el número de FLOPS. El máximo F1-Score alcanzado fue del 98.2 % con 27 coeficientes MFCC, implicando la mayor cantidad de

Tabla 3.2: Resumen Frontera de Pareto para Búsqueda de Hiperparámetros MFCC en el modelo de Reconocimiento de Sonido de Disparos

Nfft	NwinL	NhopL	Nmfcc	Ksize	F1-Score	FLOPS
4096	4096	100%	3	2	91,6	200374
4096	4096	75%	3	2	92	268190
2048	2048	100%	3	2	92,4	404574
4096	4096	100%	3	3	95,6	446294
4096	4096	75%	3	3	96,2	597710
4096	4096	100%	18	3	96,8	1743798
4096	4096	75%	15	3	97,2	1809998
4096	4096	75%	18	3	97,4	2337606
4096	2048	100%	15	3	98	2736910
4096	4096	75%	27	7	98,2	17950926

cómputo requerida con 17,9 $MFLOPS$.

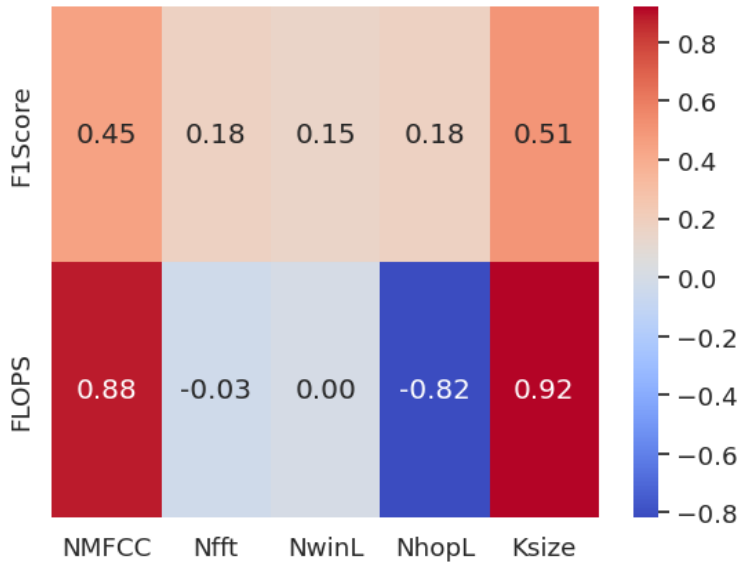

Figura 3.4: Correlación Parcial de Spearman para las Configuraciones de Hiperparámetros MFCC para el modelo de Reconocimiento de Sonido de Disparos

La Figura 3.4 muestra que todos los coeficientes de correlación parciales de Spearman para cada hiperparámetro con respecto al F1-Score son positivos. Esto significa que valores más altos de estos parámetros contribuyen a un mejor rendimiento del modelo. En orden de importancia, el tamaño del kernel de CNN y el número de coeficientes de MFCC mostraron la relación más fuerte con el F1-Score. Los otros tres hiperparámetros (Nfft, NwinL, NhopL) mostraron correlaciones similares individualmente, oscilando entre 0,15 y 0,18.

Con respecto a la relación entre cada hiperparámetro y el número de FLOPS. Observamos que la complejidad computacional del modelo depende principalmente del

tamaño del kernel de CNN y del tamaño de la imagen de entrada. Las diferentes combinaciones de hiperparámetros utilizadas para extraer los MFCC determinan el tamaño de la imagen. Por lo tanto, es predecible que los diferentes hiperparámetros tuvieran un impacto idéntico en el recuento de FLOPS. Los principales contribuyentes al número de FLOPS, en orden de importancia, son el tamaño del kernel de CNN, el número de MFCC y el solapamiento entre ventanas (NhopL). Los resultados muestran una correlación muy baja entre los hiperparámetros Nfft y NwinL con respecto al número de FLOPS. Los FLOPS se calcularon para la inferencia del modelo sin tener en cuenta la computación necesaria para obtener la imagen, lo que podría explicar la falta de correlación para estos hiperparámetros.

3.7.2 Sirenas

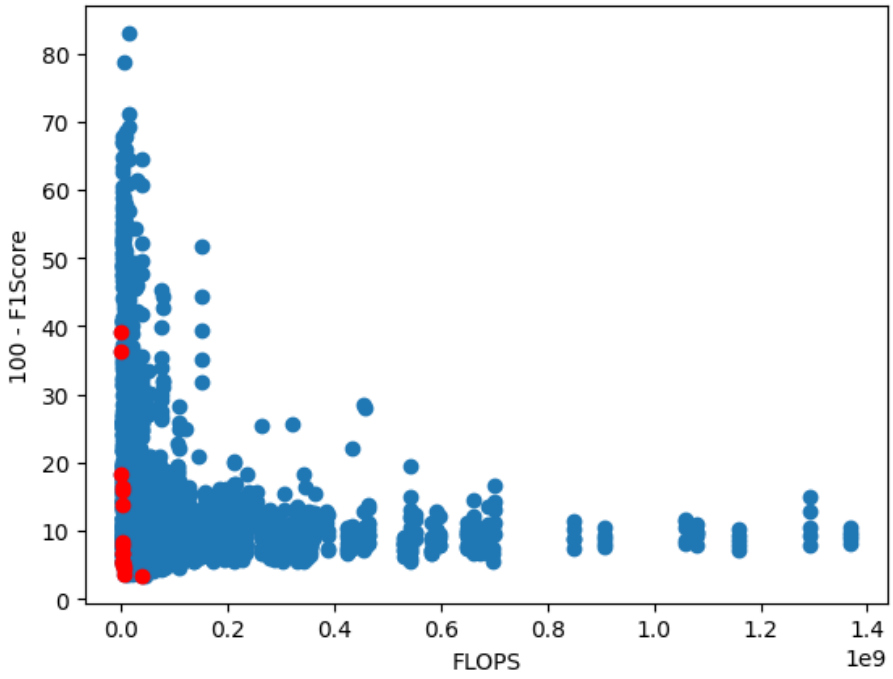

Figura 3.5: Frontera de Pareto en Rojo para las Configuraciones de Hiperparámetros MFCC para el modelo de Reconocimiento de Sonido de Sirenas

En cuanto a los modelos entrenados para el reconocimiento acústico de eventos de sirena, la Tabla 3.3 muestra cómo mejoró el rendimiento del modelo con combinaciones de Nfft y NwinL de 4096 muestras. Un NhopL del 100 % mejora en gran medida el rendimiento del modelo. Nuevamente, no existe un patrón claro con respecto al número de coeficientes MFCC y el tamaño del kernel en la frontera de Pareto. Sin embargo, obtuvimos el mejor rendimiento en F1-Score con valores de MFCC entre 15 y 30 y el tamaño de kernel más grande, lo que aumenta la cantidad de FLOPS necesarios para la ejecución del modelo. Logramos el máximo F1-Score del 96.6 % con un 75 % de NhopL, 30 coeficientes MFCC y un tamaño de kernel de 7, pero esta configuración también requiere el número máximo de FLOPS, con un valor de 38,8 $MFLOPS$.

Tabla 3.3: Resumen Frontera de Pareto para Búsqueda de Hiperparámetros MFCC en el modelo de Reconocimiento de Sonido de Sirenas

Nfft	NwinL	NhopL	Nmfcc	Ksize	F1-Score	FLOPS
4096	4096	100%	3	2	60,8	200374
4096	4096	75%	3	2	63,8	268406
4096	4096	100%	6	2	81,8	277910
4096	4096	100%	9	2	83,6	477382
4096	4096	100%	12	2	84	505190
4096	4096	100%	15	2	86,2	605782
4096	4096	100%	18	2	91,6	782838
4096	4096	100%	21	2	92,2	883430
4096	4096	100%	12	3	93,2	1125350
4096	4096	100%	18	3	93,4	1743798
4096	4096	100%	21	3	94,4	1967430
4096	4096	100%	27	3	94,6	2474070
4096	4096	75%	21	3	94,8	2641926
4096	4096	100%	30	3	95	2646486
4096	4096	100%	18	5	95,2	4818870
4096	4096	100%	21	5	95,6	5436230
4096	4096	100%	24	5	96,4	5604486
4096	2048	75%	30	7	96,6	38848966

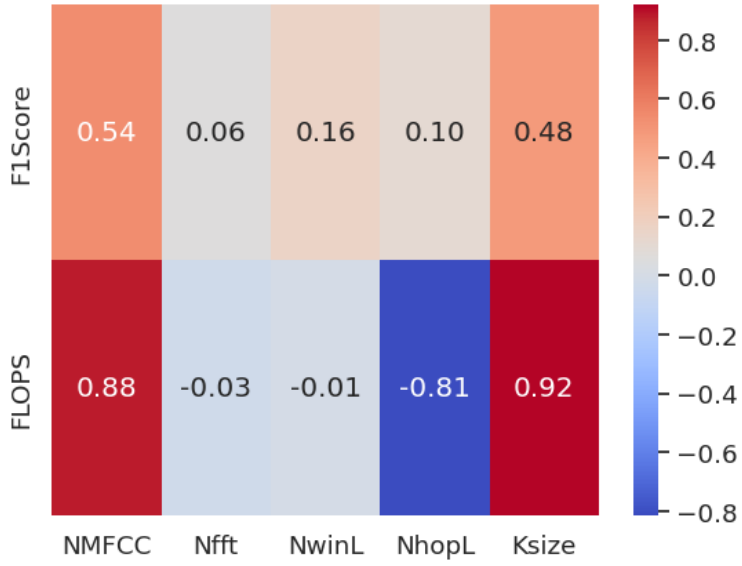

Figura 3.6: Correlación Parcial de Spearman para las Configuraciones de Hiperparámetros MFCC para el modelo de Reconocimiento de Sonido de Sirenas

En comparación con el evento de disparos en los resultados de correlación parcial de Spearman, la Figura 3.6 muestra un comportamiento similar para los eventos de sirena. El coeficiente de correlación parcial máximo en relación con el F1-Score del modelo se obtuvo para el número de hiperparámetros de MFCC, seguido por el tamaño del kernel de CNN. Con mucha menor intensidad, los hiperparámetros NwinL y Nhopl mostraron coeficientes de correlación de 0,16 y 0,10, respectivamente. La longitud de FFT (Nfft) prácticamente no tuvo correlación con el F1-Score del modelo.
Observamos un comportamiento similar al evento de disparos, en los resultados obtenidos con respecto a la relación entre cada hiperparámetro y el número de FLOPS. Observamos que la complejidad computacional del modelo depende principalmente del tamaño del kernel de CNN y del tamaño de la imagen de entrada.

3.7.3 Gritos

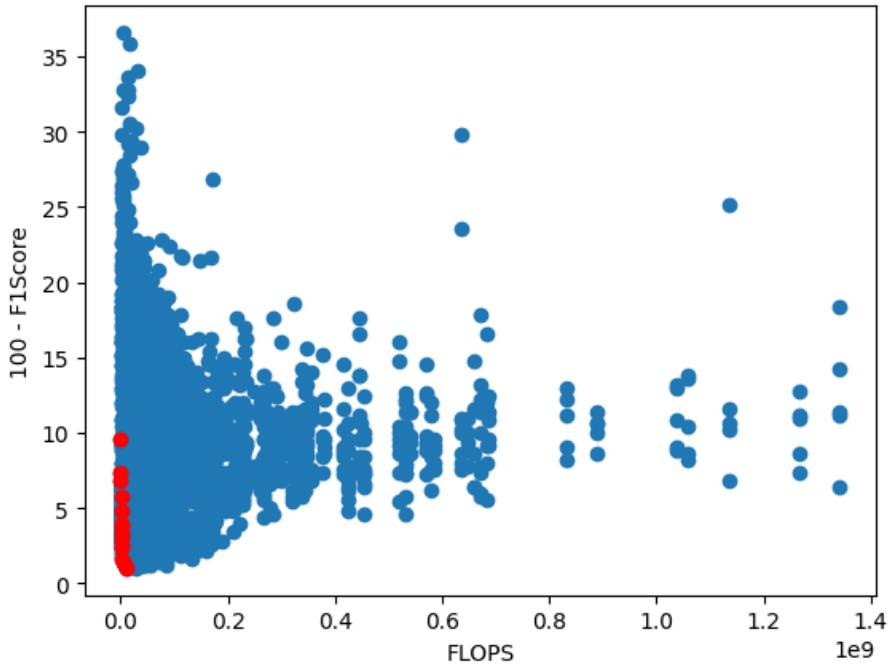

Figura 3.7: Frontera de Pareto en Rojo para las Configuraciones de Hiperparámetros MFCC para el modelo de Reconocimiento de Sonido de Gritos

La Tabla 3.4 presenta los resultados para el reconocimiento acústico de sonidos de gritos. El mejor modelo obtuvo un F1-Score del 99.0 % en los datos de prueba con 42 coeficientes Nmfcc, un 75 % de NhopL, un tamaño FFT y de NwinL de 2048 y un tamaño de kernel de 3. Similar a los sonidos anteriores, el mejor rendimiento en F1-Score fue obtenido por el modelo que requería la mayor cantidad de FLOPS en la frontera de Pareto, con un valor de 9,9 $MFLOPS$. Al igual que en los modelos anteriores, encontramos consistencia en los hiperparámetros de Nfft y NwinL, que maximizan el desempeño del modelo con los valores máximos de muestras evaluados de 2048 y 4096. En el tamaño de kernel y el número de coeficientes Nmfcc en la frontera de Pareto no se encontraron coincidencias destacables.

Tabla 3.4: Resumen Frontera de Pareto para Búsqueda de Hiperparámetros MFCC en el modelo de Reconocimiento de Sonido de Gritos

Nfft	NwinL	NhopL	Nmfcc	Ksize	F1-Score	FLOPS
4096	4096	100%	3	2	90,4	200374
4096	4096	75%	3	2	92,6	268190
4096	4096	100%	6	2	93,2	277910
4096	4096	75%	6	2	94,2	372198
4096	4096	100%	12	2	95,2	505190
4096	4096	75%	9	2	96	639486
4096	4096	75%	12	2	96,2	677190
4096	4096	75%	6	3	96,6	828998
4096	2048	100%	6	3	96,8	1251718
4096	4096	100%	36	2	97	1416166
4096	4096	75%	12	3	97,4	1508870
4096	4096	100%	6	5	97,6	1709270
4096	4096	75%	45	2	98,4	2405950
4096	4096	75%	36	3	98,6	4230406
2048	2048	75%	12	5	98,8	8191238
2048	2048	75%	42	3	99	9919318

La Figura 3.8 revela algunas diferencias con respecto a los eventos de gritos en comparación con los eventos de disparos o sirenas en las correlaciones parciales de Spearman. En contraste con los eventos de disparos o sirenas, el número de hiperparámetros de MFCC prácticamente no tuvo correlación con el F1-Score del modelo para el evento de grito. El tamaño del kernel de CNN mostró el coeficiente de correlación más robusto, con 0,59, seguido por los hiperparámetros NhopL y NwinL con coeficientes de 0,32 y 0,21, respectivamente. Al igual que en el caso de las sirenas, la longitud de FFT (Nfft) prácticamente no tuvo correlación con el F1-Score del modelo.
Observamos un comportamiento similar en los tres gráficos de correlaciones parciales de Spearman con respecto a la relación entre cada hiperparámetro y el número de FLOPS, para los eventos de disparos en la Figura 3.4, sirenas en la Figura 3.6 y gritos en la Figura 3.8.

En base a los resultados obtenidos, no existe una configuración dominante para los tres eventos acústicos estudiados. Por tanto de forma individual se recomienda utilizar diferentes configuraciones óptimas de hiperparámetros en la extracción de características de MFCC para cada uno de los tres eventos, dependiendo de los requisitos específicos de cada tarea. El contexto de la aplicación debe ayudar a seleccionar la configuración óptima entre los modelos en la frontera de Pareto.

En general, los modelos que utilizan valores más altos de parámetros (como coeficientes de MFCC, tamaño de ventana, intervalos más pequeños, etc.) tienden a tener un mejor rendimiento en el reconocimiento de eventos acústicos. Sin embargo, estos modelos son computacionalmente más costosos y pueden tardar más en entrenarse y

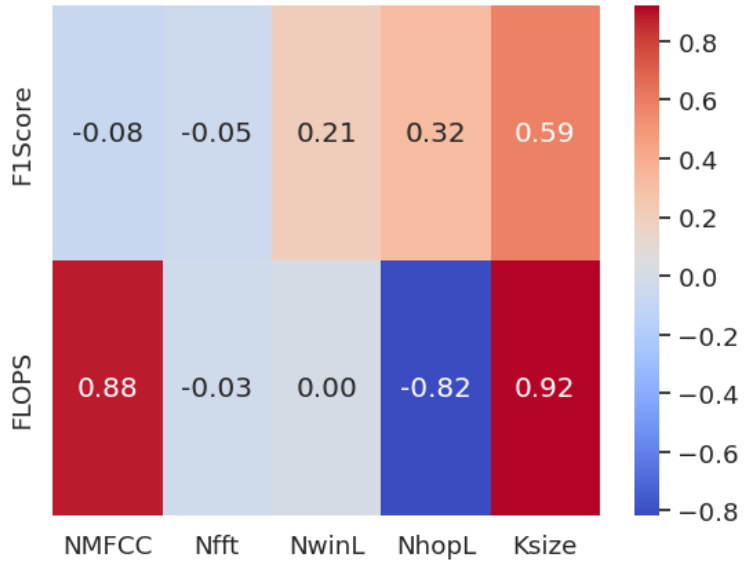

Figura 3.8: Correlación Parcial de Spearman para las Configuraciones de Hiperparámetros MFCC para el modelo de Reconocimiento de Sonido de Gritos

evaluar. Por lo tanto, existe un equilibrio entre el número de parámetros y el rendimiento del modelo que debe considerarse en función del hardware disponible y las limitaciones de tiempo.

3.8 Selección de Configuraciones

Para facilitar el logro de nuestro objetivo de realizar fusión de redes neuronales de tipo un conjunto de características múltiples tareas, debemos seleccionar una misma configuración de extracción de características para los 3 eventos acústicos seleccionados que posteriormente sus modelos serán fusionados. Para esto, realizamos un promedio aritmético de las métricas evaluadas de F1-Score y FLOPS para cada una de las configuraciones de hiperparámetros de los tres eventos acústicos. Los resultados son presentados mediante la gráfica de Pareto en 3.9 y la Tabla 3.5 resumen de las configuraciones Pareto óptimas.

Finalmente, realizamos la selección de dos configuraciones de las pertenecientes a la frontera de Pareto. Subrayadas en amarillo en la Tabla 3.5, tomamos para la construcción de los modelos este par de configuraciones de hiperparámetros para la extracción de características MFCC para el conjunto de datos de entrada. Por un lado tomamos la configuración que nos permite tener un mejor desempeño de los modelos con menor complejidad computacional medida en FLOPS. De aquí en adelante la nombraremos **Configuración 1**. Por otro lado, tomamos como **Configuración 2** aquella que nos entregó un mayor desempeño en F1-Score de los modelos, que para nuestro caso representa la mayor complejidad computacional en FLOPS de la frontera de Pareto.

Configuración 1: (Menor complejidad)

- Nfft con valor de 4096 muestras.

Tabla 3.5: Resumen Frontera de Pareto para Búsqueda de Hiperparámetros MFCC Resultado Promedio de las Métricas para los Tres Eventos Estudiados

Nfft	NwinL	Overl	Nmfcc	Ksize	F1-Score	FLOPS
4096	4096	0%	3	2	80,9	200374
4096	4096	25%	3	2	82,8	268262
4096	4096	0%	6	2	88,9	277910
4096	4096	0%	9	2	89,7	477382
4096	4096	0%	12	2	90,5	505190
4096	4096	0%	15	2	91,6	605782
4096	4096	0%	18	2	94,1	782838
4096	4096	0%	21	2	94,5	883430
4096	4096	0%	12	3	94,7	1125350
4096	4096	25%	12	3	94,8	1509478
4096	4096	0%	18	3	95,3	1743798
4096	4096	25%	15	3	95,5	1810758
4096	4096	0%	24	3	95,9	2028678
4096	4096	0%	27	3	96,1	2474070
4096	4096	25%	21	3	96,2	2639798
4096	4096	0%	30	3	96,3	2646486
4096	4096	0%	33	3	96,3	3092518
4096	4096	25%	27	3	96,5	3319974
4096	4096	0%	42	3	96,7	3771574
4096	4096	25%	42	3	96,8	5060630
4096	2048	0%	21	5	97	11024310
4096	4096	0%	45	5	97,3	11039814

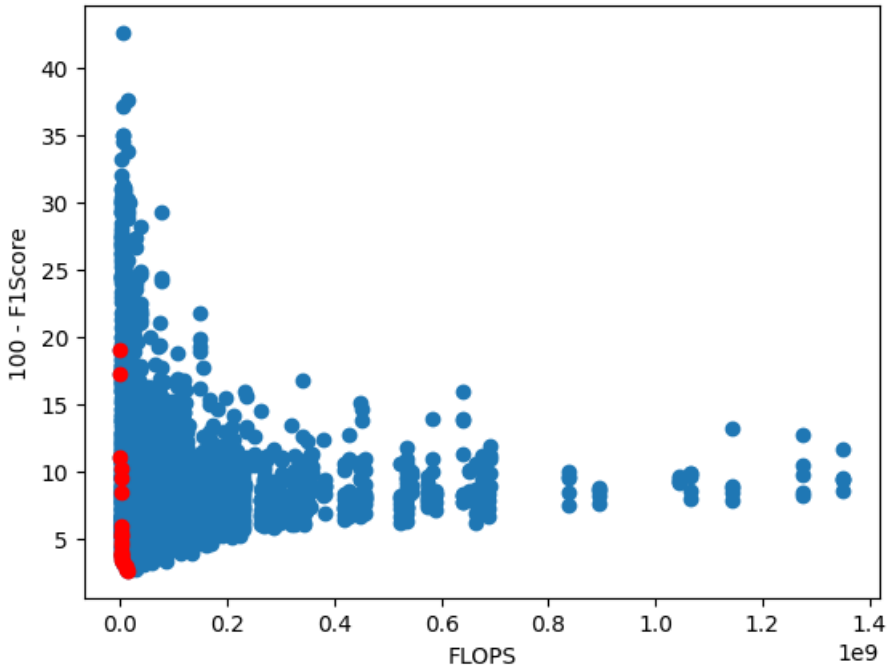

Figura 3.9: Frontera de Pareto en Rojo para las Configuraciones de Hiperparámetros MFCC para el Resultado Promedio de las Métricas en los Tres Eventos Estudiados.

- NwinL con valor de 4096 muestras.
- NhopL con valor del 100 %.
- Nmfcc con valor de 3.
- Ksize con valor de 2.

Configuración 2: (Mayor desempeño)

- Nfft con valor de 4096 muestras.
- NwinL con valor de 4096 muestras.
- NhopL con valor del 100 %.
- Nmfcc con valor de 45.
- Ksize con valor de 5.

Capítulo 4

Fusión de Modelos CNN para AER

En este capítulo presentamos inicialmente los modelos fundamentales empleados como origen en los experimentos de fusión de redes neuronales para el reconocimiento de eventos acústicos urbanos. Seguidamente presentamos las métricas de similitud entre modelos, los modelos de referencia comparativa y las estrategias de fusión propuestas en este trabajo de investigación. Finalmente, presentamos los experimentos realizados y los resultados que obtuvimos.

4.1 Modelos Individuales

En primer lugar, se entrenaron los modelos individuales para cada tipo de evento acústico desde cero en las 2 configuraciones seleccionadas. Con el fin de obtener únicos modelos iniciales para aplicarles las estrategias de fusión. En miras de evaluar si se podía mejorar el F1-Score de los modelos, se realizó el entrenamiento durante 20 épocas más respecto a las realizadas durante la exploración de espacio de diseño presentada en la Sección 3.6 del capitulo anterior. Cada modelo fue entrenado utilizando las bases de datos descritas en la Sección 3.2. El entrenamiento se llevó a cabo durante un máximo de 100 épocas, con las mismas configuraciones de compilación del modelo presentadas en la Sección 3.6.

Los resultados obtenidos medidos en desempeño F1-Score para cada evento y para las dos configuraciones de extracción de características seleccionadas se presentan en la Tabla 4.1.

Tabla 4.1: Modelos Entrenados Individuales para la Clasificación de Eventos de Interés

Evento	**F1-Score**	
	Configuración 1	**Configuración 2**
Disparos	91%	97%
Sirenas	72%	96%
Gritos	94%	96%
Promedio	86%	96%

Los resultados de la Tabla 4.1, nos muestran que algunos de los modelos bajo ciertas configuraciones si pudieron obtener mejores desempeño al aumentarse el número de épocas de entrenamiento. Al comparar el promedio aritmético del F1-Score con los resultados de la Tabla 3.5 vemos que para la configuración 1 aumento en 5 % el desempeño, pero para la configuración 2 disminuyó en 1 %.

4.2 Similitud entre Redes Neuronales Convolucionales

En esta sección dedicada a la evaluación de similitud entre pesos de los filtros de los modelos, llevamos a cabo una exploración para determinar la compatibilidad y diferencias entre los modelos a fusionar. Inicialmente, examinamos la similitud entre los pesos de los filtros de forma pareada. Para este propósito, propusimos una métrica basada en la diferencia euclidiana y la norma de los filtros, permitiéndonos evaluar tanto la convergencia como las disparidades en los modelos. Esta métrica se aplicó de dos maneras distintas: primero, sobre los filtros sin normalizar y, luego, sobre los filtros normalizados mediante la técnica min-max.

Este análisis nos permite profundizar en la similitud de los valores de los pesos intercapas, comparando cada filtro de una capa convolucional de un modelo A con todos los filtros de la misma capa convolucional en el otro modelo B.

4.2.1 Métrica de Similitud entre Filtros

La métrica propuesta para nuestra evaluación de similitud es la siguiente:

$$Similitud = 1 - \frac{||K_A - K_B||}{average(||K_A||, ||K_B||)} \tag{4.1}$$

donde K_A representa un filtro del modelo A, en todos los canales; K_B representa un filtro del modelo B, en todos los canales, y el operador $||K_B||$ representa la norma matricial del filtro cuadrado K_B. Esta métrica nos entregará un valor entre 0 y 1, que nos informa un valor cercano a 1 cuando ambos filtros tienen valores de pesos muy similares y 0 si no encuentra similitud entre ellos. En la Figura 4.1 se muestra una representación de la aplicación de la métrica.

4.2.2 Diseño Experimental

Realizamos la comparación de los modelos para cada una de las 4 capas convolucionales de los modelos. Para esto comparamos todos los filtros de cada capa de un modelo A versus todos los filtros del mismo índice de capa correspondiente al modelo B. Además, evaluamos diversas transformaciones que podrían revelar patrones de similitud entre los pesos de los filtros. Investigamos las rotaciones de los pesos de los filtros en ángulos de 90^o, 180^o y 270^o, así como las reflexiones horizontales y verticales de los filtros. En estas transformaciones, uno de los filtros se mantuvo estático mientras

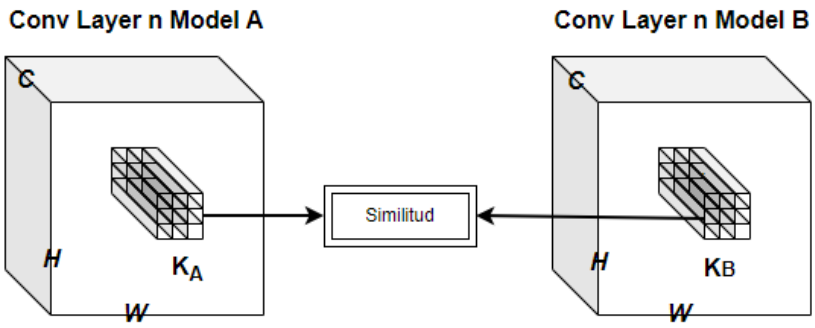

Figura 4.1: Representación sobre aplicación métrica de similitud de pesos de filtros

que se aplicaron variaciones al filtro del otro modelo. También, se calculó el promedio aritmético de los valores de similitud para cada grupo de transformaciones por capa. Estas transformaciones permitieron extraer valores métricos que profundizaron nuestra comprensión sobre las relaciones de similitud entre los modelos, arrojando luz sobre las complejidades de la fusión de modelos en nuestro estudio.
Se aplicó la métrica de similitud tanto a los filtros sin normalizar, como a los filtros después de normalizarlos mediante la técnica min-max.

4.2.3 Resultados

A continuación presentamos los resultados de aplicar la métrica de similitud sobre la comparativa entre los modelos del evento de disparos y el evento de gritos para la configuración 1 de características. En las Tablas 4.2 y 4.3 se encuentran los resultados del promedio de similitud después de aplicar la métrica sobre todas las comparativas entre los pesos de los filtros de ambos modelos. Separamos el promedio de similitud por cada una de las transformaciones aplicadas, ya sea rotación de un filtro o reflexión. En la Tabla 4.2 se encuentran los resultados correspondientes al comparativo entre los modelos entrenados desde cero de disparos y gritos. En la Tabla 4.3 se presentan los resultados de los mismos modelos respectivos a la tabla anterior, pero en este caso la métrica de similitud fue aplicada una vez los filtros pasaron previamente por la normalización min-max.

Se observa en las Tablas 4.2 y 4.3 la diferencia entre las métricas de similitud cuando es aplicada sobre los valores de los filtros normalizados mediante min-max y cuando no son normalizados. Incrementa mucho la similitud al ser normalizados pues así tiene en cuenta los casos de escalabilidad que pueden existir entre los filtros. Este tipo de normalización min-max busca que los filtros que son escalares entre sí se tengan en cuenta como similares. Al momento de realizar una fusión de modelos teniendo como base este tipo de similitud se debería tener en cuenta este factor de escala que podría existir entre los filtros y esto significaría un aumento en la complejidad computacional.

También, observamos allí que no se pueden obtener similitudes mayormente signi-

Tabla 4.2: Resultados Métrica de Similitud Modelos Entrenados desde cero Comparativa Disparo VS Gritos

Capa	Solo Filtros	Rotaciones				Reflexiones		
		90^o	180^o	270^o	Avg	Vertical	Horizontal	Avg
L0	0,3013	0,31281	0,31415	0,29713	0,30803	0,30503	0,30806	0,30655
L1	0,29587	0,29338	0,29575	0,29783	0,29565	0,29465	0,29677	0,29571
L2	0,29544	0,29377	0,29282	0,29412	0,29357	0,29324	0,29468	0,29396
L3	0,29369	0,29368	0,29337	0,29345	0,2935	0,29313	0,29401	0,29357

Tabla 4.3: Resultados Métrica de Similitud Modelos Entrenados desde cero Comparativa Disparo VS Gritos con Filtros Normalizados

Capa	Solo Filtros	Rotaciones				Reflexiones		
		90^o	180^o	270^o	Avg	Vertical	Horizontal	Avg
L0	0,64406	0,64737	0,6467	0,64132	0,64513	0,64503	0,64502	0,64502
L1	0,7147	0,71377	0,71456	0,71546	0,7146	0,71428	0,71503	0,71465
L2	0,73796	0,73738	0,73703	0,73748	0,7373	0,73718	0,73769	0,73743
L3	0,79385	0,79385	0,79376	0,79378	0,79379	0,79369	0,79394	0,79381

ficativas al realizar transformaciones en los pesos de los filtros. Mantenían su rango de similitud en los mismos valores de los filtros al ser comparados uno a uno sin transformar. La máxima similitud promedio fue obtenida con la métrica al ser aplicada sobre los filtros después de la normalización min-max, entre las cuartas capas convolucionales de los modelos, comparando solo los filtros, sin transformaciones. Con un valor máximo de 0.79385.

Con el objetivo de encontrar mayores similitudes entre los pesos de los filtros de los modelos, que permitan unos mejores resultados en miras a la fusión de los modelos. Probamos aplicar la métrica de similitud sobre modelos entrenados a partir de Transfer Learning, en los cuales la inicialización de los pesos puede aportar en ese sentido.

4.3 Modelos por Transfer Learning

En esta sección se adoptó un enfoque particular, al entrenar previamente un modelo base durante un mayor número de épocas para clasificar todas las clases dentro de los conjuntos de datos utilizados. Para con este aplicar la estrategia de *Transfer Learning* con el propósito de aprovechar el conocimiento previamente adquirido en un contexto más amplio de clases, hacia los modelos específicos para los eventos de interés.

4.3.1 Modelo Base

Este modelo base fue entrenado utilizando las 10 clases del conjunto de datos urbansound8k, tales como aire acondicionado, bocina de automóvil, juego de niños, taladro, ladrido de perro, motor, disparo, martillo neumático, sirena y música callejera. Posteriormente, se añadieron las clases de sonido de explosiones y sonido de gritos, ampliando así el conjunto de datos a un total de 12 clases.

Se realizó el entrenamiento para las dos configuraciones de extracción de características seleccionadas previamente y se sometió a un máximo de 1000 épocas de entrenamiento. Para prevenir el sobreajuste del modelo, se implementó un mecanismo de detención temprana con una paciencia configurada en 100 épocas. En términos de configuraciones de compilación del modelo y división del conjunto de datos, se aplicaron las mismas de los modelos anteriores. Los resultados obtenidos por este modelo base se presentan en la Tabla 4.4.

Tabla 4.4: Modelo Base para Transfer Learning

Modelo	**F1-Score**	
	Configuración 1	**Configuración 2**
Modelo Base	66%	90%

Los resultados de la Tabla 4.4 nos muestran un decremento significativo en el desempeño del modelo respecto a los obtenidos para estas mismas configuraciones en lo modelos entrenados anteriormente. Esto nos permite comprobar que aunque fue entrenado por un número mayor de épocas, el incremento del número de clases hace que sea aún más difícil la clasificación de las mismas. Evidenciando que el tamaño del modelo no es adecuado para la múltiple clasificación de diversas clases, sino más efectivo en la clasificación de una cantidad reducida de clases.

4.3.2 Modelos Específicos

Una vez entrenado el modelo base, realizamos transferencia de conocimiento del modelo base de todas las capas convolucionales hacia el modelo especifico de cada evento. Una vez transferido el conocimiento, realizamos un re-entrenamiento de todas

las capas del modelo durante un máximo de 50 épocas con el *early stopper* configurado a 10 épocas de paciencia. Esta vez el re-entrenamiento se realiza sobre el conjunto de datos especifico para cada evento individual. En la Tabla 4.8 recopilamos los diferentes desempeños de cada modelo en las dos configuraciones experimentadas.

Tabla 4.5: Modelos Entrenados por Transfer Learning para los Eventos de Interés

Evento	F1-Score	
	Configuración 1	Configuración 2
Disparos	95%	99%
Sirenas	82%	97%
Gritos	99%	99%

Con los desempeños de la Tabla 4.5 se superaron los desempeños obtenidos para los modelos de cada evento en cada configuración obtenidos al entrenarse desde cero recopilados anteriormente en la Tabla 4.1. Comprobando que la transferencia de conocimiento puede aportar al incremento del desempeño de los modelos. Para nuestros modelos obtuvimos mayormente desempeños superiores al 95% en F1-Score, a excepción del modelo de eventos de sirenas en la configuración 1 para el cual se obtuvo un desempeño del 82%, pero aún así, este último obtuvo un aumento de 10 puntos porcentuales respecto al entrenamiento desde cero.

Ahora, presentamos en la Subsección 4.3.3 los resultados de aplicar las métricas de similitud sobre los modelos específicos obtenidos por transfer learning, para validar nuestra hipótesis del aumento en similitud entre modelos al tener una inicialización común en los pesos.

4.3.3 Resultados de Similitud

Presentamos los resultados de aplicar la métrica de similitud sobre la comparativa entre los modelos del evento de disparos y el evento de gritos para la configuración 1 de características. Bajo el mismo diseño experimental expuesto en la Subsección 4.2.2. En la Tabla 4.6 se encuentran los resultados correspondientes al comparativo entre los modelos entrenados por transfer learning para disparos y gritos. En la Tabla 4.7 se presentan los resultados de los mismos modelos respectivos a la tabla anterior, pero en este caso la métrica de similitud fue aplicada una vez los filtros pasaron previamente por la normalización min-max.

Los hallazgos encontrados en las Tablas 4.6 y 4.7 nos permitieron confirmar los encontrados previamente en la Subsección 4.2.3 en cuanto al mínimo aporte de similitud encontrado al realizar transformaciones de rotación y reflexión entre los pesos de los filtros. Por tanto, a partir de allí descartamos su uso, ya que representaban un gasto computacional mayor y no aportaban a la similitud para su fusión. Esto, teniendo como base la comparativa solo los filtros, representados en la primera columna de las

Tabla 4.6: Resultados Métrica de Similitud Modelos Entrenados por Transfer Learning Comparativa Disparo VS Gritos

Capa	Solo Filtros	Rotaciones				Reflexiones		
		90^o	180^o	270^o	Avg	Vertical	Horizontal	Avg
L0	0,31383	0,26928	0,26825	0,27003	0,26919	0,27958	0,27465	0,27712
L1	0,32015	0,29538	0,28747	0,29577	0,29287	0,29128	0,30086	0,29607
L2	0,32099	0,29426	0,27705	0,29407	0,28846	0,27795	0,31117	0,29456
L3	0,3265	0,30149	0,28698	0,30132	0,2966	0,28736	0,31597	0,30167

Tabla 4.7: Resultados Métrica de Similitud Modelos Entrenados por Transfer Learning Comparativa Disparo VS Gritos con Filtros Normalizados

Capa	Solo Filtros	Rotaciones				Reflexiones		
		90^o	180^o	270^o	Avg	Vertical	Horizontal	Avg
L0	0,71674	0,70001	0,70044	0,70038	0,70028	0,70729	0,70219	0,70474
L1	0,90607	0,90288	0,90181	0,90295	0,90254	0,90236	0,90361	0,90299
L2	0,93135	0,92863	0,92672	0,92865	0,928	0,9268	0,93058	0,92869
L3	0,93669	0,93434	0,93294	0,93434	0,93388	0,93298	0,93576	0,93437

tablas, que para cada una de las capas representaba en promedio la mayor similitud en los modelos evaluados.

Como segunda observación a resaltar, encontramos un incremento en el valor de la similitud promedio por capa entre los modelos, al comparar los modelos entrenados desde cero y los modelos entrenados mediante Transfer Learning. Se observa un incremento de la similitud significativo en los modelos que partieron del entrenamiento con Transfer Learning, un resultado que esperábamos, puesto que la inicialización de los pesos en este caso parten de la misma base y tienden a tener patrones similares para sus valores. Conclusión que también fue encontrada por Singh y Jaggen en [121]. Por tanto para el tipo de fusión que realizamos en este trabajo de investigación utilizaremos de aquí en adelante los modelos entrenados a partir de Transfer Learning por sus mayores similitudes entre pesos.

4.4 Modelos de Referencia Biclase

En esta sección del capítulo, exploramos el entrenamiento de modelos mediante la técnica de Transfer Learning, siguiendo un enfoque ligeramente diferente al presentado anteriormente en la Sección 4.3. En este caso, decidimos re-entrenar modelos específicos para la clasificación de dos clases seleccionadas de interés. Esta elección estratégica se hizo con el objetivo de obtener una perspectiva comparativa adicional para comparar en el proceso de fusión posterior.

Para obtener estos modelos, partimos del mismo modelo base 4.4, transfiriendo el conocimiento de todas las capas convolucionales al modelo específico destinado a cada par de clases de interés. Estos nuevos modelos fueron sometidos a un proceso de re-entrenamiento sobre el conjunto de datos relevante, limitado a un máximo de 50 épocas. Para mitigar el riesgo de sobreajuste, implementamos el mecanismo de detención temprana (*early stopper*) configurado con una paciencia de 10 épocas. Los resultados de desempeño de estos modelos se presentan en la Tabla 4.8.

Tabla 4.8: Modelos Biclase Entrenados por Transfer Learning para Pares de Eventos de Interés

Eventos	**F1-Score**	
	Configuración 1	**Configuración 2**
Disparos - Gritos	95%	99%
Disparos - Sirenas	82%	97%
Gritos - Sirenas	84%	97%

Los desempeños en F1-Score de la Tabla 4.8 para los modelos biclase, nos permitirán tener una perspectiva comparativa de tener un modelo entrenado para la clasificación de dos eventos de interés versus al ser fusionados los eventos a partir de los modelos individuales. Los resultados obtenidos nos muestran que, frente a los modelos individuales para cada evento por tranfer learning, presentados en la Tabla 4.8 en la mayoría de los casos se mantuvo o redujo el desempeño en F1-Score, pero no hubo una mejora.

4.5 Estrategias de Fusión

En la subsección 2.5.3 presentamos el tipo de fusión que exploramos en este trabajo de investigación; un conjunto de características múltiples tareas. En este sentido, se presentó allí también el trabajo realizado por Mallouh et al. en [46] como referencia a los tipos de fusión que se pueden trabajar sobre los modelos individuales para obtener uno fusionado. Aunque allí realizaron únicamente la fusión a nivel de la última capa profunda de los modelos realizando la suma elemento a elemento, y obtuvieron buenos resultados en la clasificación de las tareas. Existen algunas otras técnicas a diferentes niveles de fusión que también han sido exploradas y presentamos a continuación.

En [122] Amin et al. presentaron en su trabajo la fusión de redes neuronales convolucionales poco profundas y profundas utilizando un autoencoder. Estas CNN fueron preentrenadas y posteriormente, eliminaron las capas softmax y densas de las CNN y se fusionaron concatenando las características extraídas por las capas de pooling. Este estudio propuso ese modelo de fusión de CNN de múltiples capas para la clasificación de la Interfaz Cerebro-Computadora (MI, por sus siglas en inglés) a partir de datos electroencefalográficos (EEG). En los resultados demostraron que el preentrenamiento ayudó a mejorar la precisión y a evitar el sobreajuste del modelo.
Por parte de Singh y Jaggen en [121] diseñaron un algoritmo de fusión de modelos capa por capa para redes neuronales que utiliza el transporte óptimo para alinear las neuronas entre los modelos antes de promediar sus parámetros asociados. Demostrando que esto puede lograr una transferencia de conocimiento "de un solo disparo"(es decir, sin requerir ningún nuevo entrenamiento) entre redes neuronales entrenadas en datos no i.i.d heterogéneos. Siguiendo la idea de promedio de parámetros entre modelos para su fusión, Grimberg et al. en [123] muestran su estudio enfocado en aprendizaje federado. En el estudio exploraron enfoques para entrenar modelos neuronales personalizados para nodos individuales considerando diferencias en datos y tareas. Allí demostraron que el promedio aritmético entre un modelo local y uno global, bajo ciertas suposiciones mínimas sobre las distribuciones, reduce el error cuadrático esperado en comparación con el uso único del modelo local entrenado en datos locales. Este enfoque les proporcionó una mejora significativa en la calidad del desempeño en comparación con el uso exclusivo de datos locales de entrenamiento del modelo del nodo.
A partir de esto, presentamos en las Subsecciones 4.5.1, 4.5.2, 4.5.3 y 4.5.4, las estrategias que proponemos y exploramos en este trabajo y los resultados que obtuvimos.

4.5.1 Fusión por Capas

Como primer acercamiento a la fusión de los modelos planteamos como estrategia la fusión de los modelos por capas. En la cual, se realiza la evaluación de desempeño de cada una de las capas convoluciones completas en ambos sentidos, evaluando el desempeño en el conjunto de datos de test respectivos para cada tarea. Esto, con el intercambio de cada una de las capas del modelo A en el modelo B y viceversa. Así, evaluamos de manera progresiva capa a capa, y eligiendo aquella que mayor desempeño en F1-Score en promedio aritmético para ambas de las tareas demostraba. En la Figura 4.2 mostramos una ilustración del tipo de fusión realizado. Así también en el Algoritmo 1 presentamos una visión general del algoritmo utilizado en esta estrategia.

4.5.2 Fusión por Filtros

Una vez se exploró la fusión a nivel de capa, optamos por evaluar la fusión de los modelos partiendo desde una estrategia a nivel de filtros. Evaluamos y planteamos diferentes estrategias de fusión a nivel de filtro de las capas convolucionales presentadas en las subsecciones 4.5.2.1 y 4.5.2.2.

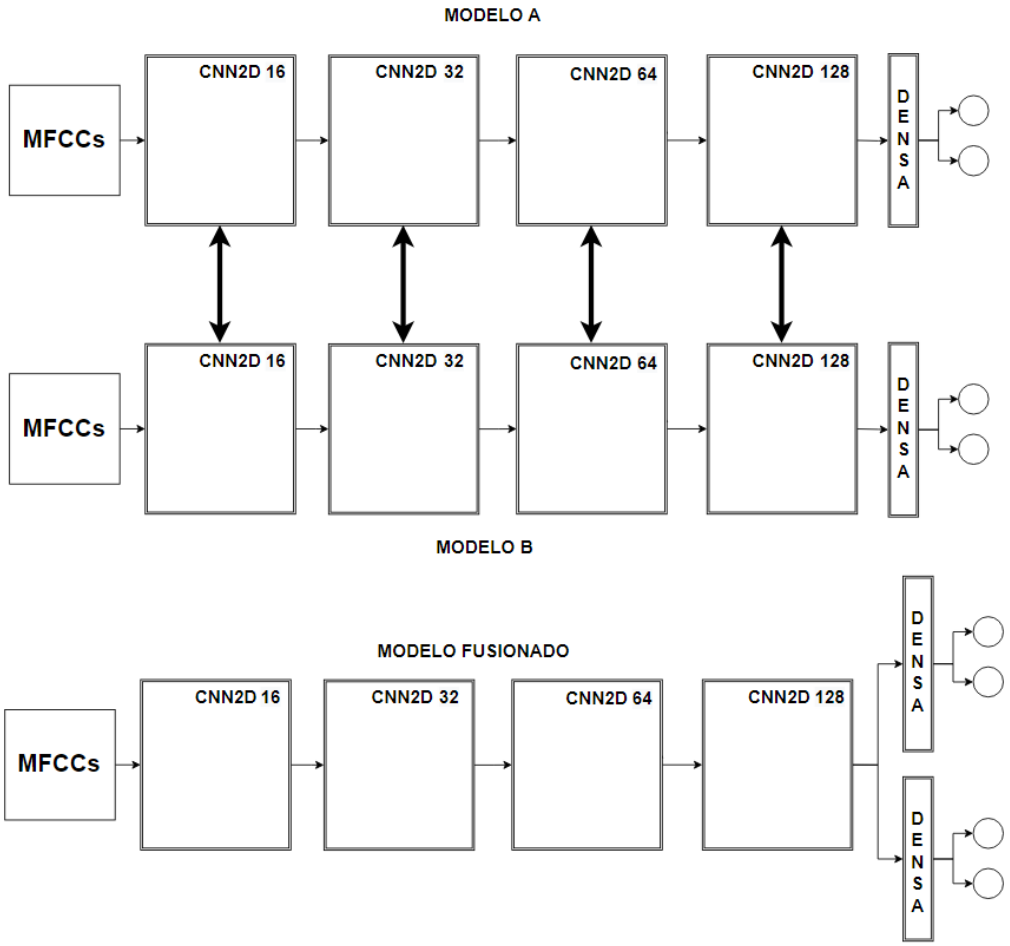

Figura 4.2: Estrategia de Fusión por Capa para dos Modelos Convolucionales A y B, evaluando desempeño y seleccionando aquella que maximiza el F1-Score promedio para ambas tareas

Algorithm 1: Fusión por capas de modelos convolucionales A y B.

Input: Modelo convolucional A, Modelo convolucional B
Output: Modelo fusionado F

```
1  Function FusionPorCapas(Modelo A, Modelo B):
2      Inicializar Modelo fusionado F con mismo número de capas convolucionales
         que A y B;
3      for cada capa convolucional en A y B do
4          Intercambiar la capa convolucional entre A y B;
5          Evaluar el desempeño en el conjunto de datos de test para A y B;
6          Calcular el promedio del desempeño F1-Score;
7          Seleccionar la capa con mejor desempeño promedio en ambas tareas;
8          Asignar la capa seleccionada a F;
9      Mantener las partes densas de los modelos A y B como clasificadores en F;
10     return Modelo fusionado F;
```

4.5.2.1. Sin Evaluar Desempeño

En esta estrategia planteamos realizar la fusión de los modelos mediante la técnica de operar los pesos filtro a filtro de cada una de las capas convolucionales. Por los resultados encontrados en los estudios presentados en [121] y [123] respecto a fusionar modelos basado en el promedio aritmético, decidimos operar mediante el promedio de los valores de los pesos de cada filtro convolucional de las capas de los modelos. Lo realizamos pareando los filtros según su índice por capa, respectivamente para cada modelo A y B. Para contrastar esta operación de promedio, aplicamos además las estrategias de fusión basada en la elección del valor máximo y otra mediante el valor mínimo de los pesos de los filtros. En la Figura 4.3 presentamos una ilustración de la estrategia de fusión que se lleva a cabo, en este caso para un par de modelos A y B en una capa convolucional n. La operación de fusión filtro a filtro será, o por máximo valor de pesos, mínimo valor de pesos o promedio de los pesos entre los filtros.

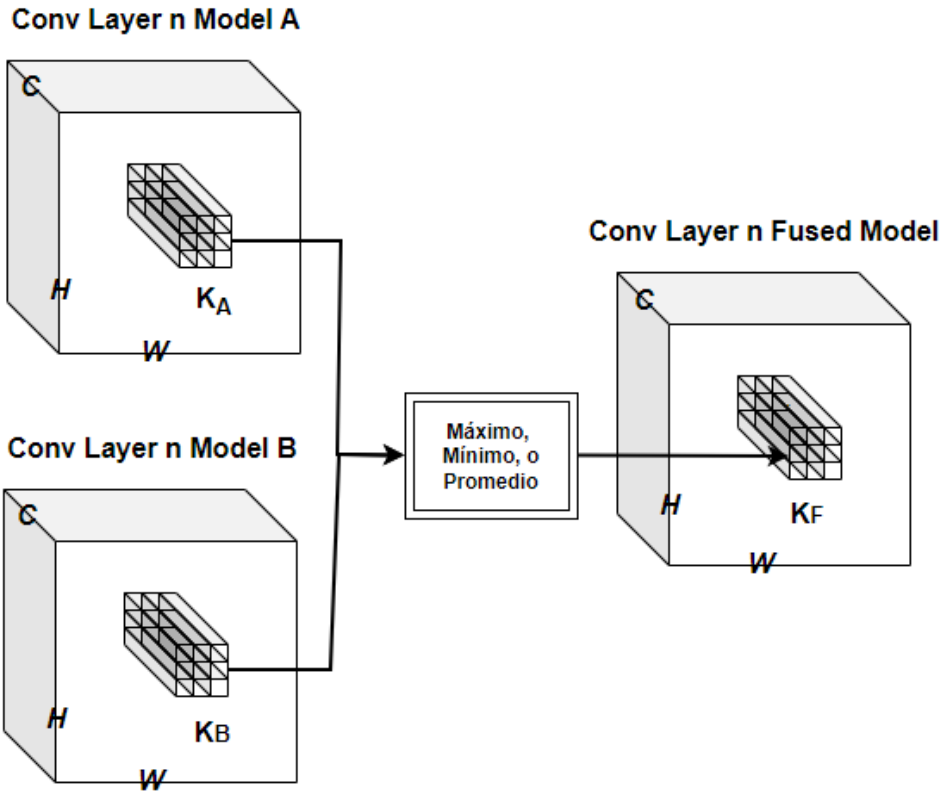

Figura 4.3: Fusión por Filtros Sin Evaluar Desempeño, Estrategia de Máximo, Mínimo y Promedio

4.5.2.2. Evaluando Desempeño

En base a la estrategia de fusión por capas de la sección 4.5.1 al realizar fusión por capas evaluando y seleccionando la capa que mejor desempeño obtenía en el modelo fusionado. Decidimos aplicar está técnica ahora a nivel de filtro. Planteamos estrategias basadas en evaluar el desempeño sobre los filtros teniendo en cuenta las métricas de similitud, normalizando y sin normalizar los filtros. Además, lo evaluamos sin las

métricas de similitud. Una descripción más detallada de los procedimientos son presentadas en las Subsecciones 4.5.2.3 y 4.5.2.4.

4.5.2.3. Evaluando Desempeño Sin Métricas de Similitud

En esta estrategia planteamos intercambiar cada uno de los filtros por cada capa convolucional del modelo A en el modelo B y viceversa. Esto siendo pareado en el mismo índice del filtro en cada capa. Allí, evaluamos el desempeño de cada uno de los modelos, en el conjunto de datos de test respectivos para cada tarea. Seleccionamos el filtro para el modelo fusionado en el mismo índice y capa mediante la evaluación del desempeño promedio aritmético de ambas tareas. El criterio de selección se basa en el filtro que logra el mayor F1-Score promedio. En caso de empate, en el desempeño al evaluar los filtros, probamos 3 técnicas diferentes de desempate; optimizar la selección hacia el filtro del modelo A, optimizar la selección hacia el filtro del modelo B u optimizar la selección realizando el promedio entre los pesos de ambos filtros.

4.5.2.4. Evaluando Desempeño Con Métricas de Similitud

En esta última estrategia de fusión de modelos, utilizamos los resultados de similitud de los pesos entre pares de filtros extraídos en la Sección 4.2. Aquí planteamos evaluar de manera similar a lo presentado en la Subsección anterior 4.5.2.3, el desempeño del modelo al intercambiar filtros entre ambos modelos A y B. En este caso, solo realizamos el intercambio de los filtros que presentaron mayores valores de similitud entre ellos. Para esto utilizamos los resultados de las métricas de similitud entre los filtros aplicadas a los filtros normalizados min-max y a los filtros sin normalizar. Tanto para realizar la selección de aquellos filtros que no cumplen con un umbral de similitud propuesto, como para los que al evaluar el desempeño tienen un empate en desempeño. Se realiza la selección del filtro para el modelo fusionado utilizando las mismas optimizaciones propuestas en la Subsección anterior 4.5.2.3, se seleccionan basados en optimizar la selección hacia el filtro del modelo A, optimizar la selección hacia el filtro del modelo B u optimizar la selección realizando el promedio entre los pesos de ambos filtros.

En el Algoritmo 2 presentamos el procedimiento, de manera general, para llevar a cabo la estrategia de fusión por filtros basada en la métrica de similitud.

4.5.3 Diseño Experimental

Definidas les estrategias de fusión que proponemos, procedemos a realizar la experimentación de fusión para los modelos individuales entrenados en la Subsección 4.3.2. Probamos la fusión de las combinaciones por pares de eventos de interés. Las combinaciones serán, modelos de disparos con modelos de gritos, modelo de disparos con modelo de sirenas, y modelo de gritos con modelo de sirenas. Para cada una de las combinaciones se aplican la fusión para las dos configuraciones seleccionadas de características de entrada. Las diferentes estrategias de fusión aplicadas son listadas como

Algorithm 2: Algoritmo de Fusión por Filtros con Métrica de Similitud

```
Input: Modelo A, Modelo B, Resultados de Similitud, Umbral de Similitud
Output: Modelo Fusionado
Function FusionFiltros(ModeloA, ModeloB, MetricaSimilitud,
  UmbralSimilitud):
    foreach Capa Convolucional en ModeloA y ModeloB do
        if Optimización es para ModeloA then
            Inicializar Filtros del Modelo Fusionado con Filtros de ModeloA;
        else if Optimización es para ModeloB then
            Inicializar Filtros del Modelo Fusionado con Filtros de ModeloB;
        else
            Inicializar Filtros del Modelo Fusionado con Promedio de Filtros de
              ModeloA y ModeloB;
        Se ordenan descendente los valores de similitud entre filtros;
        foreach Valor de Similitud en Métrica de Similitud do
            if Valor de Similitud > UmbralSimilitud then
                if Par de Filtros no ha sido Intercambiado previamente then
                    Intercambiar Filtros Bidireccionalmente entre ModeloA y
                      ModeloB;
                    Evaluar Desempeño en Datos de Test;
                    Calcular F1-Score Promedio;
                    Seleccionar Mejor Filtro según F1-Score y Regla de
                      Optimización en Caso de Empate;
                    Asignar Mejor Filtro al Modelo Fusionado;
    Mantener las partes densas de los modelos A y B como clasificadores en el
      modelo Fusionado;
    return Modelo Fusionado;
```

resumen, a continuación.

- Fusión por capas 4.5.1.
- Fusión por filtros sin evaluar desempeño, con promedio, máximo y mínimo valor de pesos 4.5.2.1.
- Fusión por filtros evaluando desempeño sin métrica de similitud con optimización de desempate seleccionando optimización hacia modelo A, modelo B o promedio 4.5.2.3.
- Fusión por filtros evaluando desempeño con métrica de similitud con optimización de desempate seleccionando optimización hacia modelo A, modelo B o promedio. Evaluando valores de umbral de similitud: 100 %, 95 %, 90 %, 80 %, 70 %, 60 %, 50 %, 40 % y 20 % 4.5.2.3.

4.5.4 Resultados

En esta subsección presentamos los resultados de aplicar las anteriores estrategias de fusión para las combinaciones de los eventos de interés que estamos explorando en esta investigación. Para todas se muestran los resultados del desempeño promedio aritmético en F1-Score del modelo fusionado para ambas de las tareas, el promedio aritmético del desempeño en F1-Score de los modelos individuales entrenados por transferencia de conocimiento y el desempeño del modelo biclase entrenado por transferencia de conocimiento para la clasificación de ambas clases para tenerlos de comparativa. Se presentan tanto para la configuración 1 como para la configuración 2 de características seleccionadas en este estudio.

4.5.4.1. Fusión por capas

En las Figuras 4.4, 4.5 y 4.6 se muestran los resultados para la fusión de los modelos de disparos y gritos, disparos y sirenas, y gritos y sirenas, respectivamente.

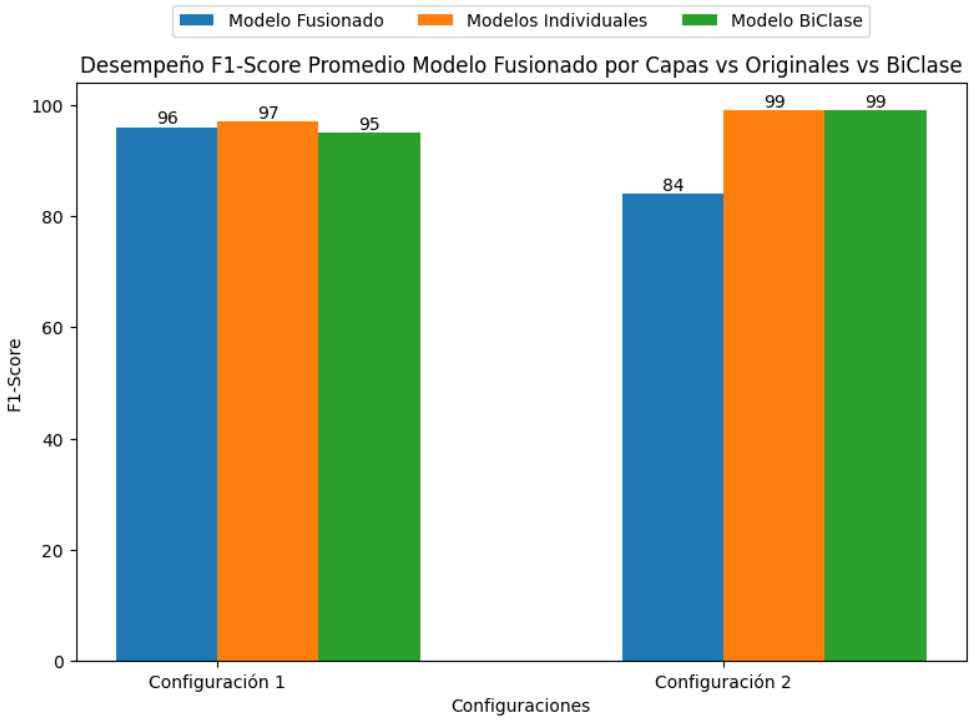

Figura 4.4: Fusión por Capa para Modelos de Disparos y Gritos

Con estos resultados, vemos un panorama que alienta al tipo de fusión planteado en este trabajo de investigación. Hubo una degradación del desempeño del F1-Score promedio en el modelo fusionado para la clasificación de ambas de las tareas, pero a valores aceptables de desempeño. En general para los tres modelos fusionados hubo un mejor desempeño en la configuración 1 de características. La diferencia porcentual máxima encontrada en el desempeño del modelo fusionado versus los modelos individuales fue de 23 % en la configuración 2 al fusionar el modelo de disparos con el de

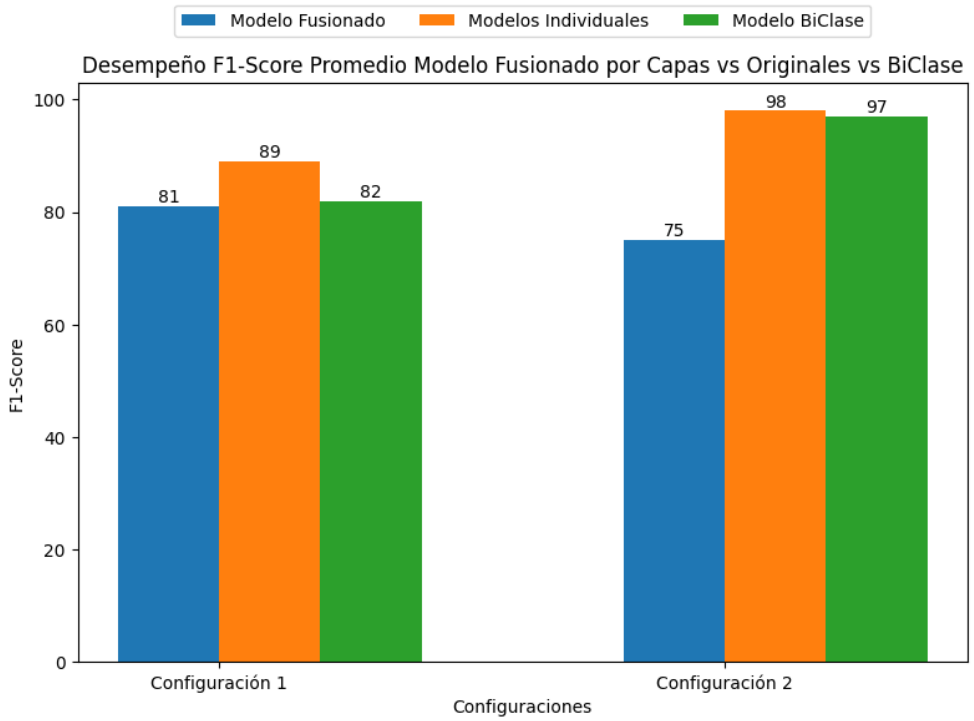

Figura 4.5: Fusión por Capa para Modelos de Disparos y Sirenas

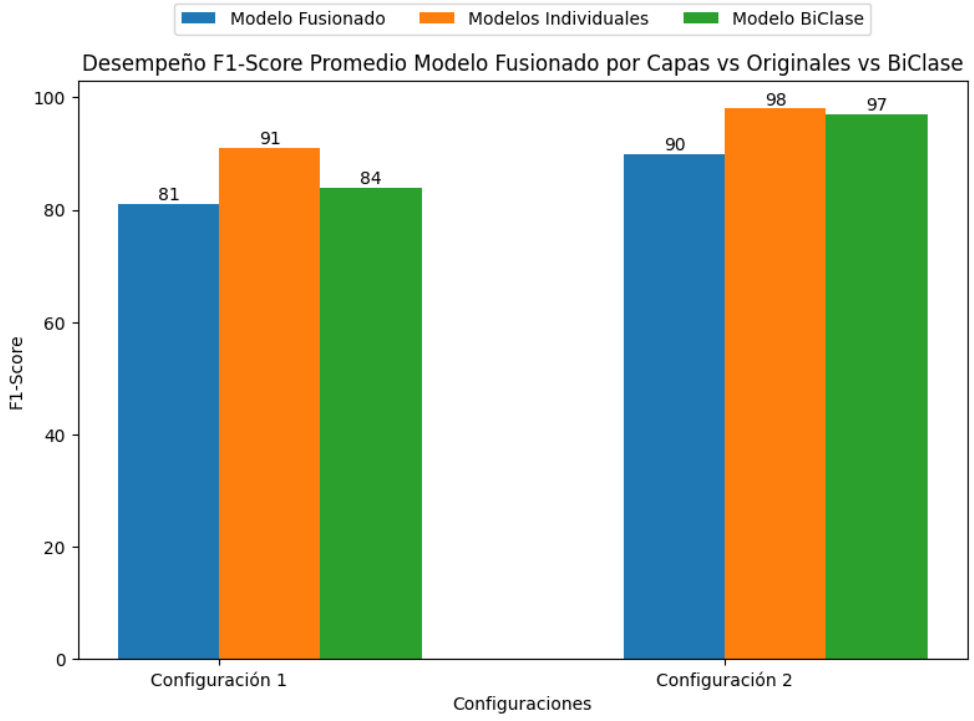

Figura 4.6: Fusión por Capa para Modelos de Gritos y Sirenas

sirenas.

4.5.4.2. Fusión por filtros sin evaluar desempeño

Como resultados de aplicar la estrategia de fusión por filtros, realizando operaciones de máximo, mínimo o promedio aritmético entre los valores de pesos de filtros de las capas convolucionales de los modelos. Se presentan las Figuras 4.7, 4.8 y 4.9, donde se muestran los resultados para la fusión de los modelos de disparos y gritos, disparos y sirenas, y gritos y sirenas; respectivamente.

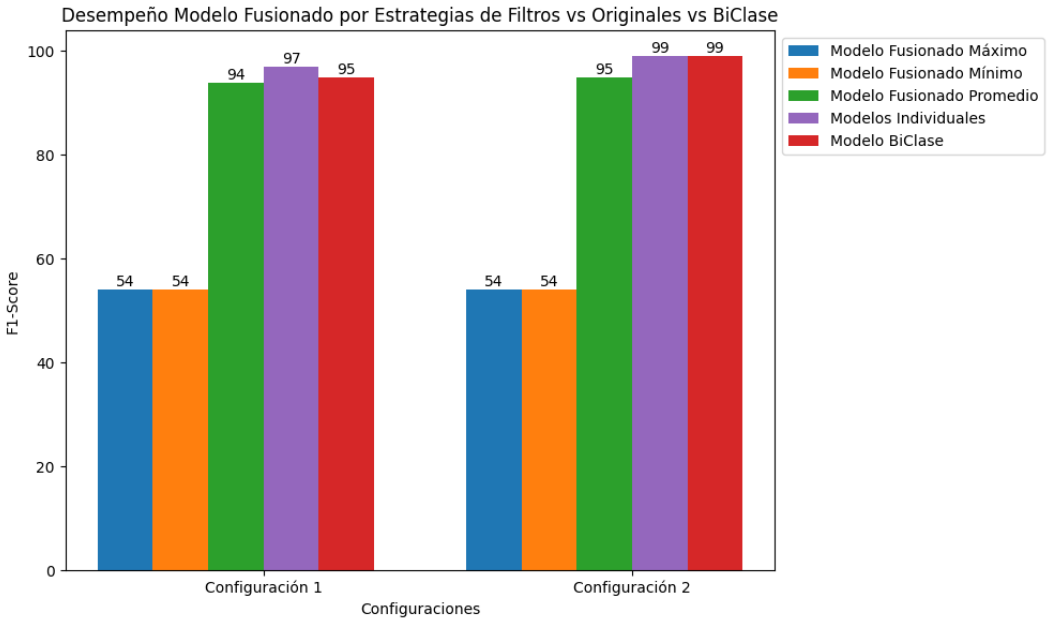

Figura 4.7: Fusión por Filtros Estrategias de Máximo, Mínimo y Promedio para Modelos de Disparos y Gritos

Los resultados presentan una mejor estrategia de fusión que la planteada en la Subsección 4.5.1. Ya que, a través de la estrategia por fusión de promedio aritmético entre los filtros en el mismo índice por capa, presentó mucho mejor desempeño a lo largo de los tres modelos fusionados evaluados. Esto tanto para la configuración 1 como para la configuración 2. Los otros dos tipos de fusión por filtros evaluados, a través de las operaciones de máximo y mínimo valor de los pesos de los filtros, arrojaron una degradación del modelo, hasta un punto prácticamente de ser aleatoria para la clasificación de las dos tareas del modelo fusionado.

4.5.4.3. Fusión por filtros evaluando desempeño sin métrica de similitud

Los resultados para la comparativa de esta estrategia de fusión descrita en la Subsección 4.5.2.3, son presentados en las Figuras 4.10, 4.11 y 4.12.

Con está estrategia de fusión, pudimos obtener un desempeño del modelo en F1-Score promedio en algunos casos mayor que los modelos individuales o el modelo

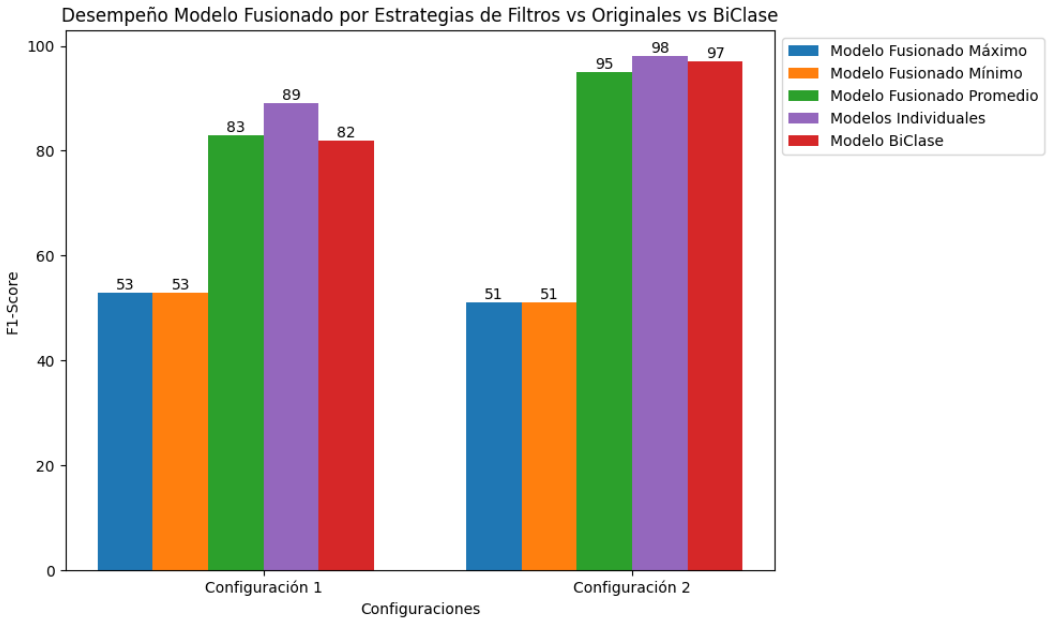

Figura 4.8: Fusión por Filtros Estrategias de Máximo, Mínimo y Promedio para Modelos de Disparos y Sirenas

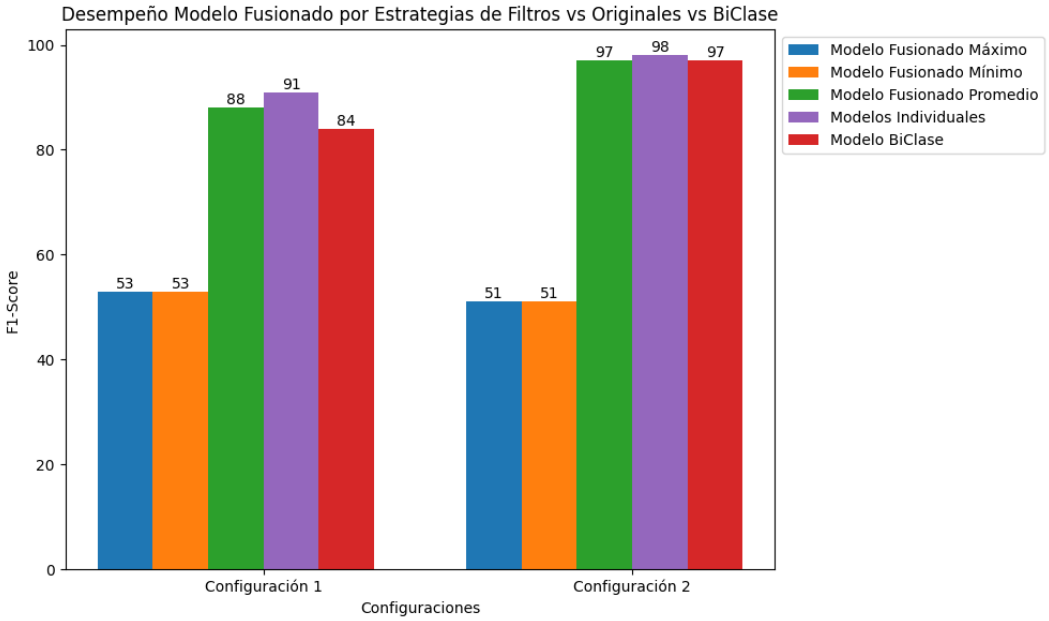

Figura 4.9: Fusión por Filtros Estrategias de Máximo, Mínimo y Promedio para Modelos de Gritos y Sirenas

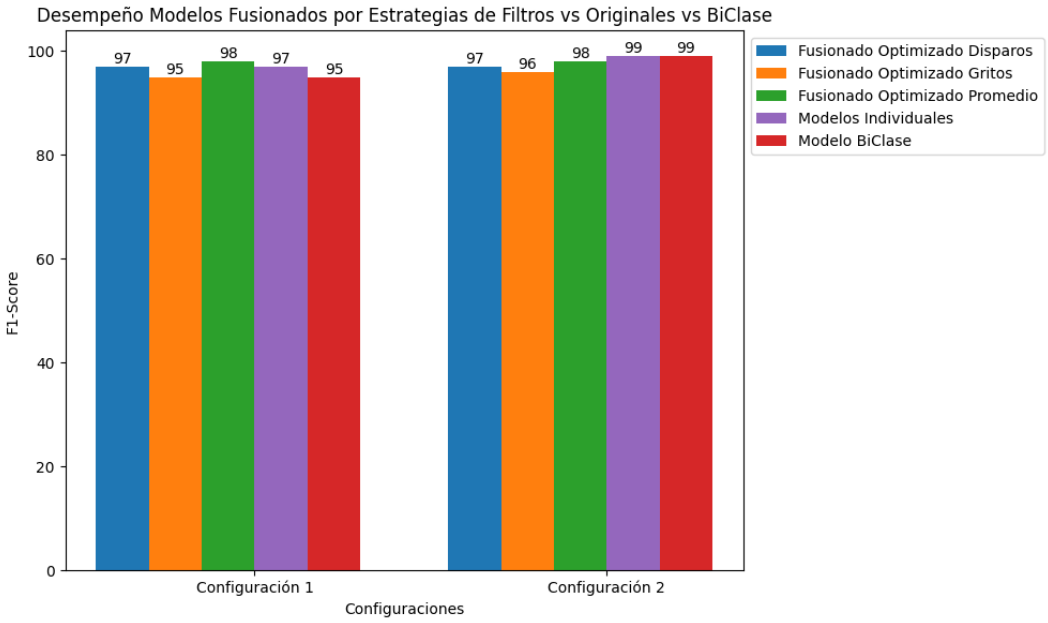

Figura 4.10: Fusión por Filtros Estrategias Optimización Modelo de Disparos, Optimización Modelo de Gritos y Optimización Promedio para Modelos de Disparos y Gritos

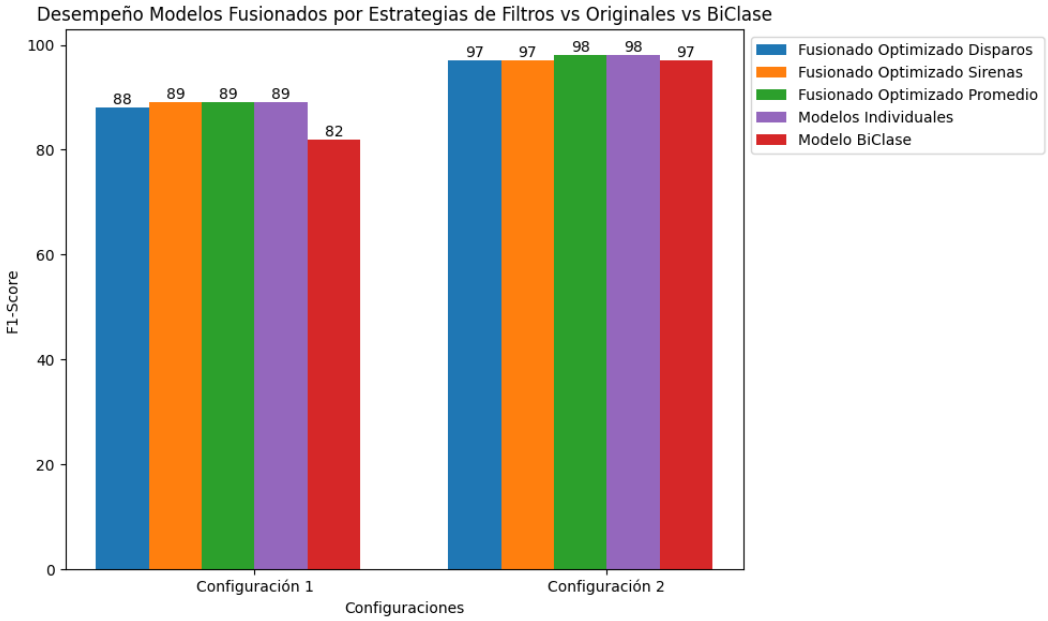

Figura 4.11: Fusión por Filtros Estrategias Optimización Modelo de Disparos, Optimización Modelo de Sirenas y Optimización Promedio para Modelos de Disparos y Sirenas

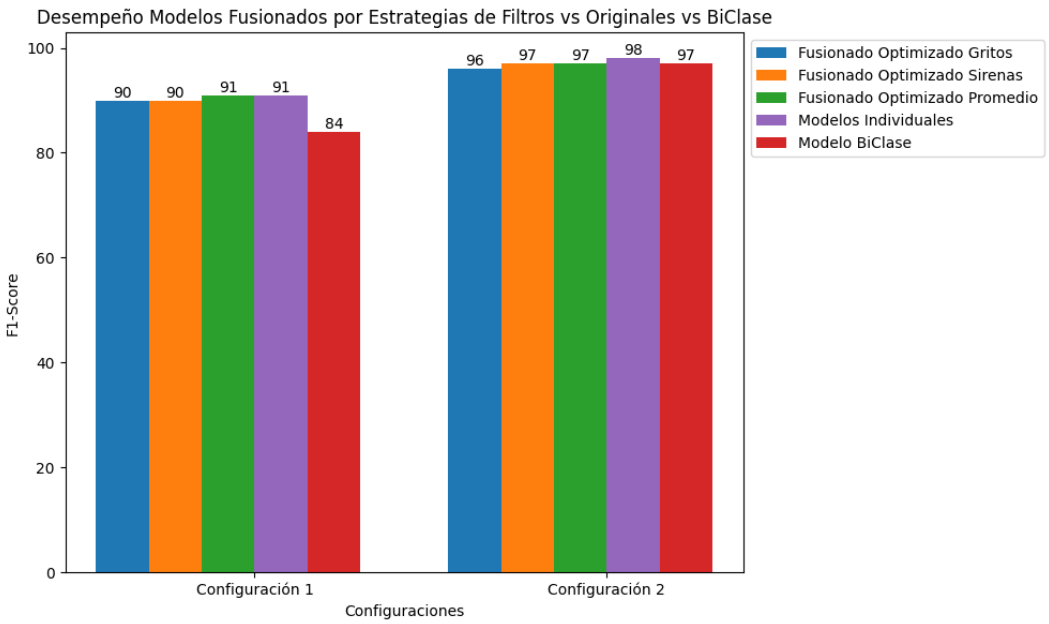

Figura 4.12: Fusión por Filtros Estrategias Optimización Modelo de Gritos, Optimización Modelo de Sirenas y Optimización Promedio para Modelos de Gritos y Sirenas

biclase de comparación. Resultados que demuestran que la fusión y la operación entre modelos individuales específicos pueden mejorar el desempeño de modelos entrenados para la clasificación de múltiples eventos. De manera general, para ambas configuraciones de características y los tres modelos fusionados, la estrategia de fusión por filtros evaluando el desempeño optimizado hacia el promedio entre los filtros fue la que mejor desempeño demostró versus las otras dos estrategias optimizadas hacia la selección del filtro del modelo A o la selección optimizada hacia el filtro del modelo B.

4.5.4.4. Fusión por filtros evaluando desempeño con métrica de similitud

Los resultados son presentados a continuación para cada par de eventos de interés fusionados. En lo cuales se gráfica los desempeños en F1-Score promedio aritmético del modelo fusionado para cada optimización de la estrategia de fusión, también se realiza la distinción para la métrica de similitud aplicada sea sobre los filtros normalizados min-max o sin normalización. En el caso de la fusión de los eventos de disparos y gritos se presentan en las Figuras 4.13 y 4.14 para las 2 configuraciones de extracción de características evaluadas. En las Figuras 4.15 y 4.16 para los eventos de disparos y sirenas. Por último en las Figuras 4.17 y 4.18 para los eventos de gritos y sirenas.

Tanto para la estrategia de fusión evaluando el desempeño de los filtros utilizando las métricas de similitud con los filtros normalizados como sin normalizar, encontramos un patrón en el desempeño según el umbral de similitud en los filtros que se está fusionando. Empiezan con una tendencia creciente en el desempeño F1-Score promedio según va disminuyendo el umbral de similitud, hasta llegar a un punto de estabilización de desempeño máximo o de decaimiento nuevamente del desempeño. Este patrón de estabilización de desempeño se encontró más presente en la estrategia de fusión con op-

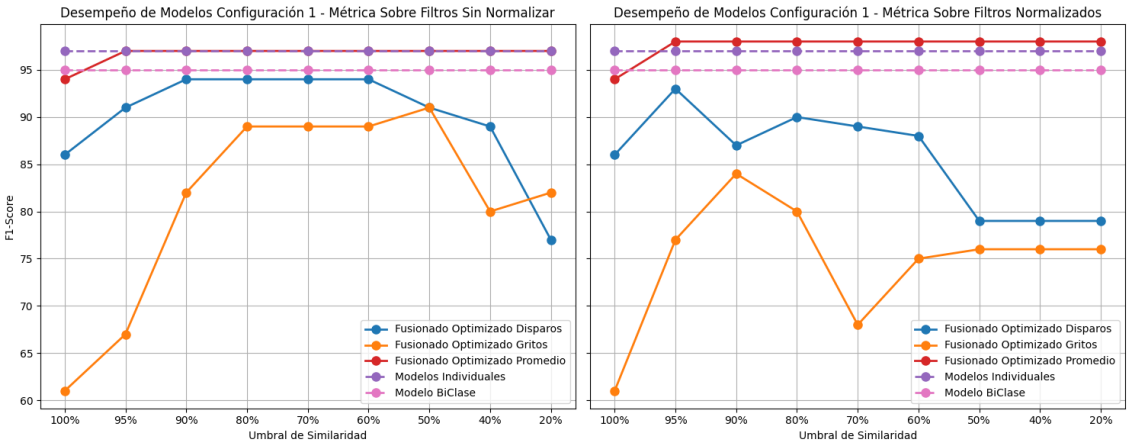

Figura 4.13: Fusión por Filtros Estrategias Optimización Modelo de Disparos, Optimización Modelo de Gritos y Optimización Promedio para Modelos de Disparos y Gritos con Métrica de Similitud para la Configuración de Características 1

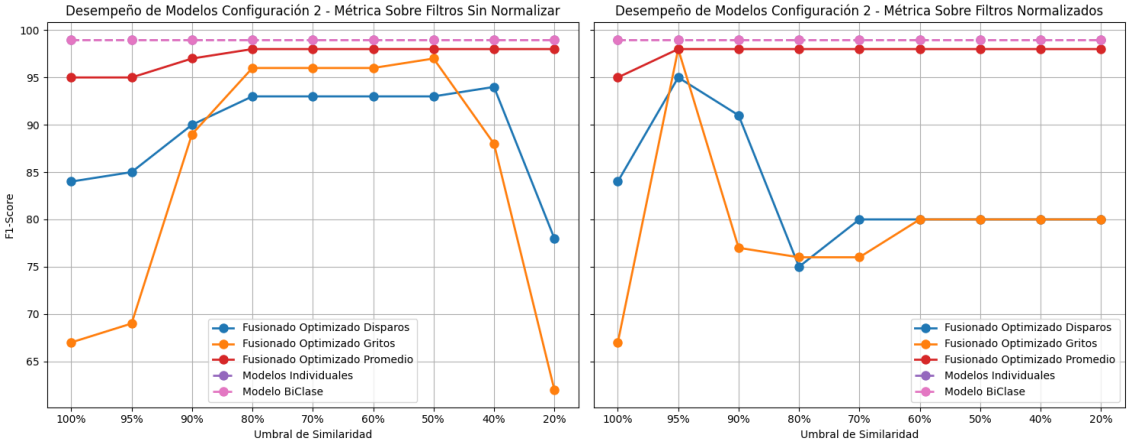

Figura 4.14: Fusión por Filtros Estrategias Optimización Modelo de Disparos, Optimización Modelo de Gritos y Optimización Promedio para Modelos de Disparos y Gritos con Métrica de Similitud para la Configuración de Características 2

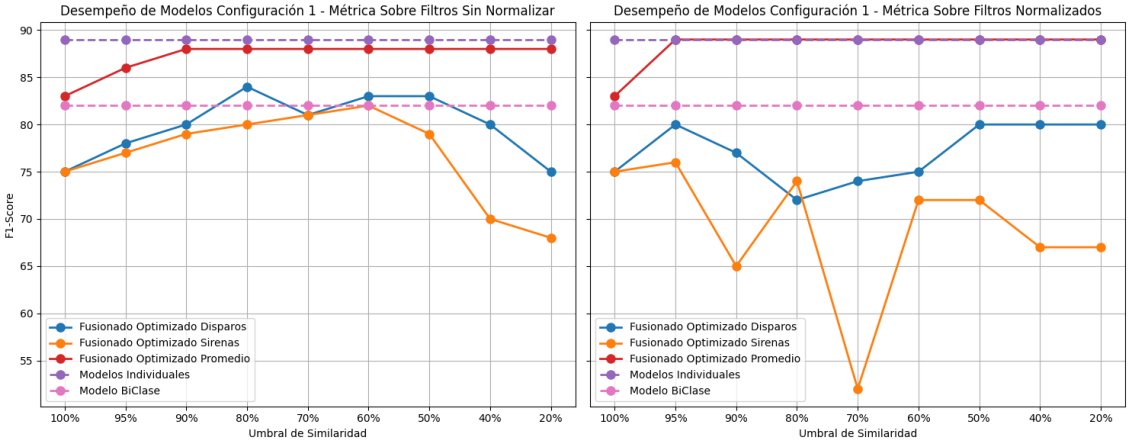

Figura 4.15: Fusión por Filtros Estrategias Optimización Modelo de Disparos, Optimización Modelo de Sirenas y Optimización Promedio para Modelos de Disparos y Sirenas con Métrica de Similitud para la Configuración de Características 1

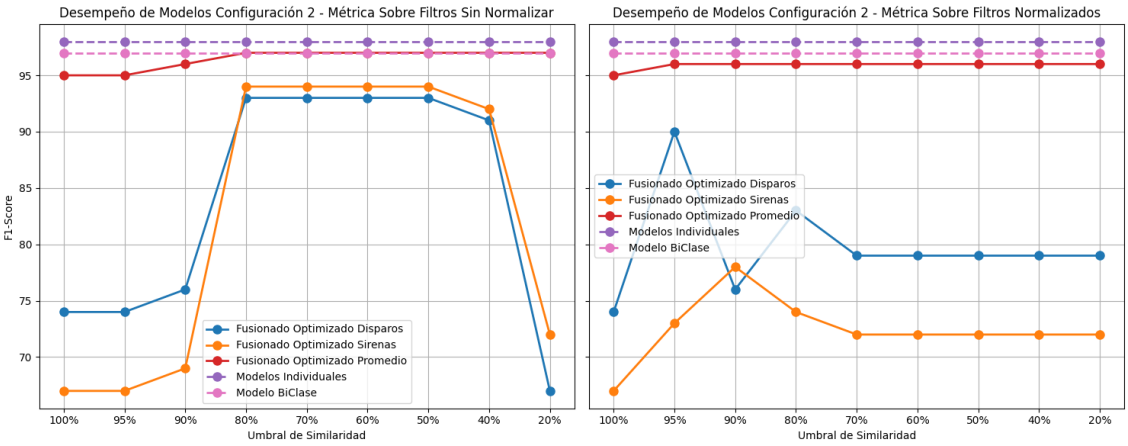

Figura 4.16: Fusión por Filtros Estrategias Optimización Modelo de Disparos, Optimización Modelo de Sirenas y Optimización Promedio para Modelos de Disparos y Sirenas con Métrica de Similitud para la Configuración de Características 2

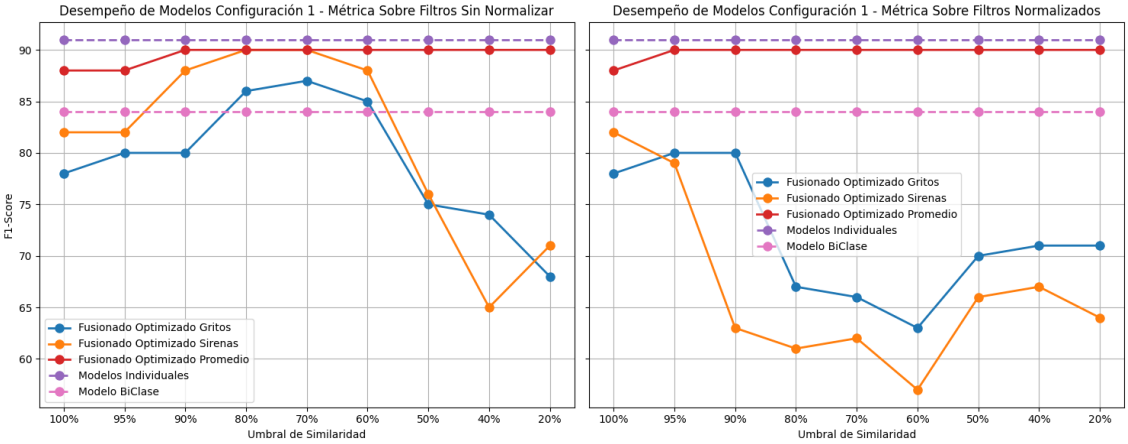

Figura 4.17: Fusión por Filtros Estrategias Optimización Modelo de Gritos, Optimización Modelo de Sirenas y Optimización Promedio para Modelos de Gritos y Sirenas con Métrica de Similitud para la Configuración de Características 1

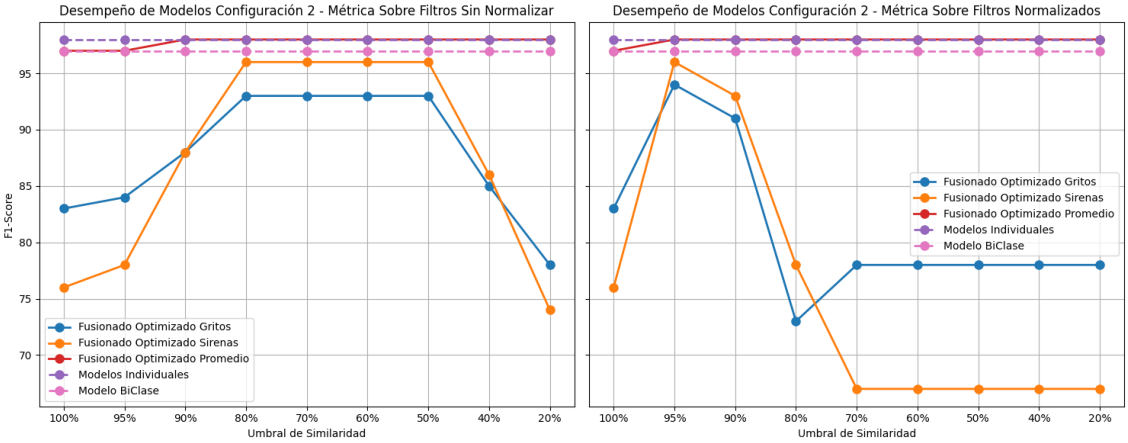

Figura 4.18: Fusión por Filtros Estrategias Optimización Modelo de Gritos, Optimización Modelo de Sirenas y Optimización Promedio para Modelos de Gritos y Sirenas con Métrica de Similitud para la Configuración de Características 2

timización al promedio entre los filtros como desempate e inicialización. Así también, esta estrategia fue la que en todos los casos presentó mejor desempeño en F1-Score versus las otras estrategias. Logrando en algunos casos igualar y superar el desempeño de los modelos individuales y el modelo biclase. Se encontró que para todos los casos explorados, esta estrategia de fusión alcanzó en el 80 % de similitud entre los filtros o antes, el mayor desempeño del modelo fusionado.

Capítulo 5

Evaluación de Desempeño en Ejecución

En este capítulo comparamos los resultados de la evaluación de desempeño en ejecución en un dispositivo de borde de recursos limitados de los modelos fusionados frente a la ejecución simultanea de los modelos individuales y la ejecución del modelo biclase de referencia.
En todos los casos de las estrategias de fusión propuestas y aplicadas, se utilizó la arquitectura de modelo convolucional fusionado presentado en la Figura 4.2. Para estás estrategias de fusión se pasó de tener dos modelos individuales A y B con misma arquitectura convolucional, a un solo modelo fusionado que realiza la clasificación para ambas de las tareas a partir de la misma configuración de datos de entrada. Esto, nos permite generalizar la evaluación del desempeño de los modelos fusionados, a un modelo representativo de ellos, pues tendrán todos la misma arquitectura y complejidad computacional para su ejecución. Para la evaluación utilizamos el modelo fusionado de diparos y gritos con la estrategia de fusión por filtros evaluando desempeño con métrica de similitud, con desempate promedio y un umbral del 80 % de similitud.
En la Sección 5.1 se presentan los resultados de las métricas de evaluación de los modelos que son inherentes a la ejecución, en ellas se encuentran el número de parámetros de los modelos y el número de FLOPS. Para las demás métricas de evaluación de desempeño que dependen de la ejecución, se planteó un diseño experimental en la Sección 5.2 donde se presenta las configuraciones, el dispositivo y las herramientas de medición utilizadas para obtener los resultados. Los resultados de estas métricas son presentados en las Secciones 5.3, 5.4, 5.5 y 5.6. Finalmente, en la Sección 5.7 se presenta los resultados de medición de tiempo de ejecución para una implementación real de la solución, teniendo en cuenta la adquisición y procesamiento de los datos.

5.1 Resultados de Evaluación en Métricas de Número de Parámetros y FLOPS

En la Figura 5.1 comparamos los resultados de evaluar las métrica de número de parámetros en los modelos y en la Figura 5.2 el número de operaciones flotantes requeridas para llevar a cabo una inferencia del modelo. En ellas podemos observar que, para la configuración 1, el número de parámetros del modelo fusionado solo supera por 0.3 % al modelo biclase. Con respecto al modelo individual, el modelo fusionado tiene

50 % menos número de parámetros requerido para las inferencias. Para la configuración 2 se observa de manera similar que el modelo fusionado requiere 0.04 % más de parámetros comparado con el modelo biclase y 50 % menos comparado con los requerimientos de los modelos individuales. Teniendo presente que las estrategias planteadas no requieren el conjunto de datos de entrenamiento para la fusión, estos valores de incremento no representan mayor complejidad para los resultados. Esto al compararse con el modelo biclase, que desde un inicio es planteado y entrenado con ambos conjuntos de datos. En el caso de los modelos individuales, es un ahorro mayor al 50 % en parámetros.
En el caso del número de operaciones flotantes requeridas para la ejecución de una inferencia en los modelos, se pueden extrapolar las conclusiones anteriores. Para esta métrica los valores porcentuales del modelo fusionado para la configuración 1 fueron de un incremento del 0.06 % versus el modelo biclase y de un decremento 50.06 % menos versus los modelos individuales. Para la configuración 2 fueron un incremento de 0.001 % versus el modelo biclase y un decremento del 50 % versus los modelos individuales.

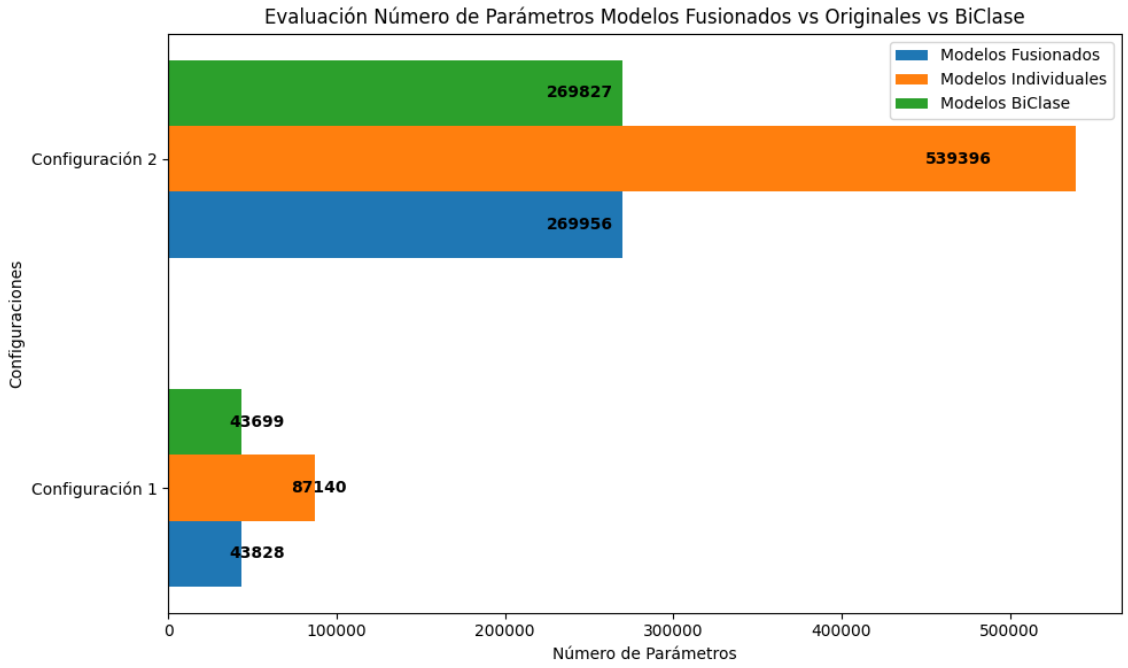

Figura 5.1: Evaluación de Número de Parámetros de Modelo Fusionado Comparado con Modelos Individuales y Modelo BiClase para Configuración de Características 1 y 2.

5.2 Diseño Experimental

En esta sección presentamos las configuraciones, plataforma y herramienta de medición utilizada para obtener las métricas de evaluación de desempeño de los modelos durante su ejecución. Para la experimentación se utilizó como dispositivo de borde, una tarjeta VIM3 PRO de Khadas. Esta tarjeta cuenta con un chipset Amlogic A311D que tiene un procesador Quad-core Cortex-A73 a 2.2GHz y Dual-core Cortex-A53 a

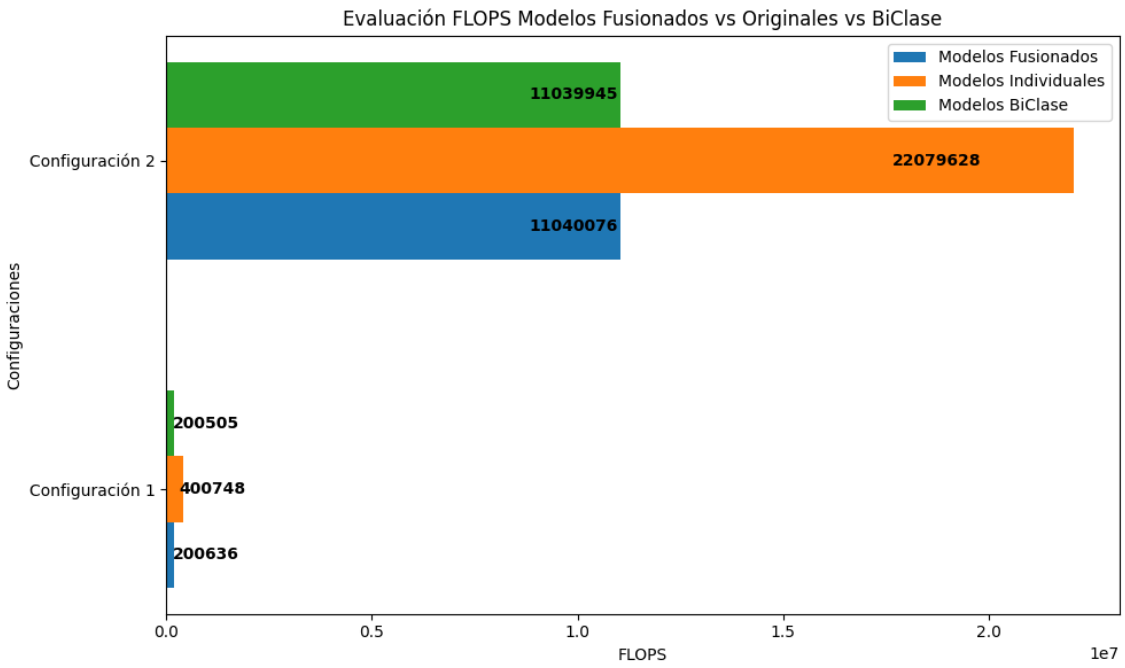

Figura 5.2: Evaluación de FLOPS de Modelo Fusionado Comparado con Modelos Individuales y Modelo BiClase para Configuración de Características 1 y 2.

1.8GHz, con GPU ARM G52 MP4 de hasta 800MHz. Incluye 4GB de RAM y memoria EMMC 5.1 de 32GB.

Para aprovechar las bondades de ejecución de modelos de redes neuronales en dispositivos de borde, utilizamos la conversión de los modelos a uno reducido dentro del mismo framework. TensorFlow ofrece la herramienta de realizar optimizaciones del modelo llevándolo a TensorFlowLite con tipos de datos de menor tamaño, mediante cuantización. Para este trabajo evaluamos 3 tipos de datos diferentes para la representación del modelo. Listados a continuación.

- Punto Flotante de 32 bits (Por defecto).
- Punto Flotante de 16 bits.
- Enteros de 8 bits.

Como herramienta de medición para las métricas, utilizamos el benchmark[124] disponibilizado por el mismo framework TensorFlow, con el cual evaluamos el peso de los modelos, la huella en memoria y el tiempo de ejecución de los modelos. El Benchmark es un script que evalúa las diferentes métricas en la ejecución de un modelo en la arquitectura de hardware específica. Lo realiza ejecutando 50 veces el modelo y obteniendo un valor promedio aritmético para cada métrica, en relación a las 50 ejecuciones.

5.3 Resultados de Evaluación: Métrica F1-Score Post Cuantizar

Esta sección condensa los resultados obtenidos para la métrica de desempeño F1-Score después de evaluar los modelos en el conjunto de test, para cada una de las representaciones en los tipos de datos experimentados. Con ello buscamos encontrar cómo se ve afectado el desempeño de los modelos al ser representados en diferentes tipos de datos para reducir su complejidad de ejecución en dispositivos de recursos limitados.
Para esta evaluación, se hizo la diferencia entre eventos acústicos fusionados, puesto que, el desempeño del modelo es una métrica atada a la aplicación específica. Para las demás secciones de este capitulo no se realiza esa distinción, ya que son métricas que no están atadas a la aplicación especifica, sino a la arquitectura y complejidad del modelo.
Se tomó como modelo fusionado de referencia para esta evaluación de desempeño, el modelo para el cual se obtuvo mejores desempeños de las estrategias de fusión aplicadas. Este corresponde a la estrategia de fusión por filtros evaluando desempeño, con métrica de similitud en umbral de 80 % y optimización de desempate promedio aritmético.

Así, en la Figura 5.3 se presentan los resultados para el modelo fusionado de disparos y gritos comparado con los modelos individuales y el modelo biclase. De manera similar en la Figura 5.4 para los eventos de disparos y sirenas, y finalmente en la Figura 5.5 para los eventos de gritos y sirenas.

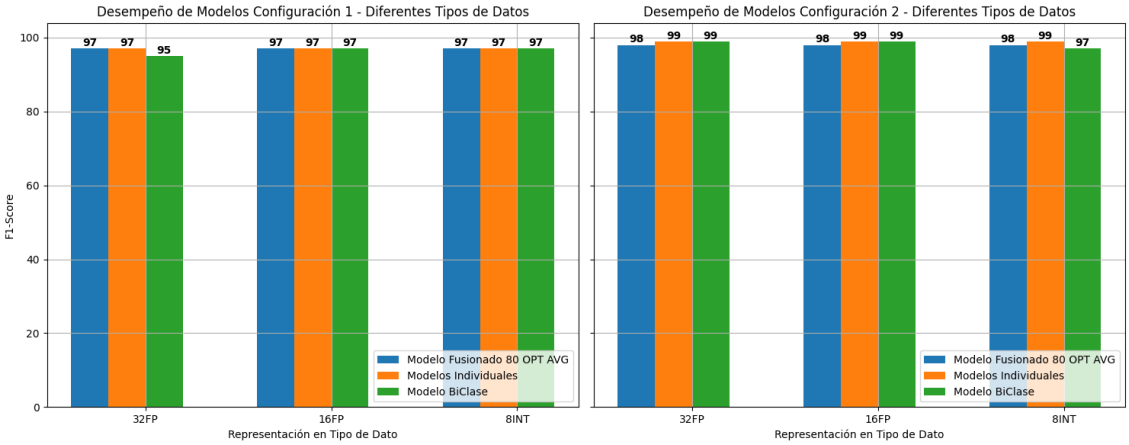

Figura 5.3: Desempeño en F1-Score Promedio Bajo Diferentes Tipos de Datos para Modelos de Disparos y Gritos

En las Figuras 5.3, 5.4 y 5.5 encontramos que para los modelos fusionado e individuales, al reducir el tipo de dato utilizado en la representación de los modelos, no se perdía significativamente el desempeño en F1-Score del modelo. Máximo para estos casos hubo una degradación de 1 % en el desempeño al reducir el tipo de dato a Punto Flotante de 16 bits y a enteros de 8 bits. Por parte del modelo biclase, presentó una mayor degradación en el desempeño en F1-Score, al reducir el tamaño del tipo de dato. Específicamente al pasar el modelo de Punto Flotante de 32 bits a Flotante de 16 bits, para la configuración 1. Con reducciones en desempeño del 6 % en el modelo biclase

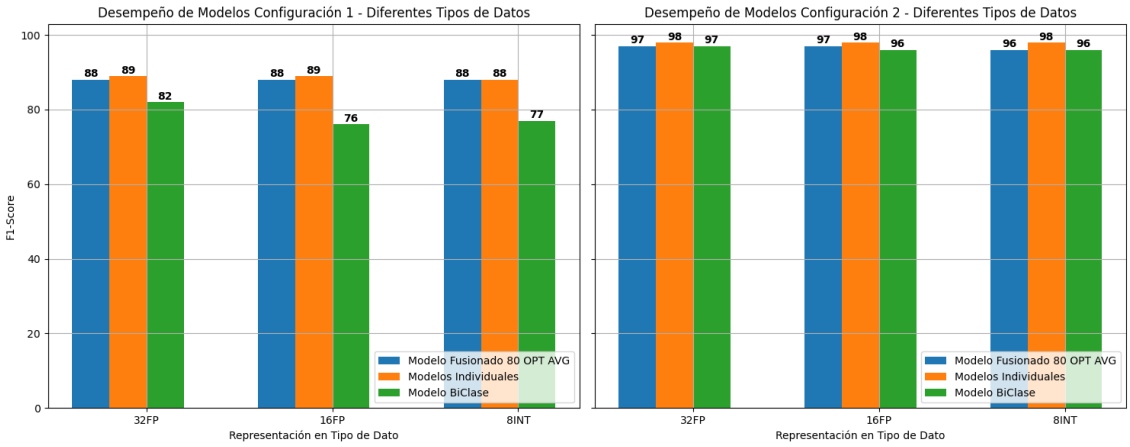

Figura 5.4: Desempeño en F1-Score Promedio Bajo Diferentes Tipos de Datos para Modelos de Disparos y Sirenas

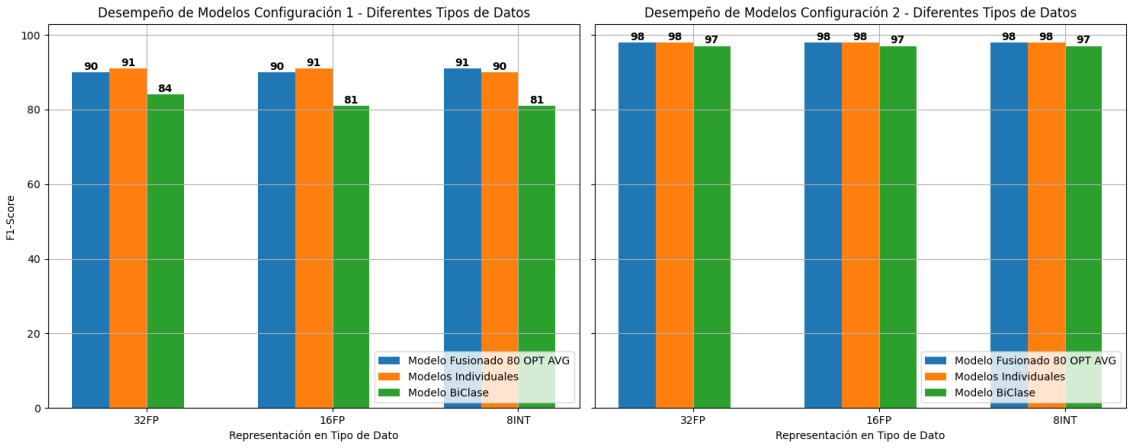

Figura 5.5: Desempeño en F1-Score Promedio Bajo Diferentes Tipos de Datos para Modelos de Gritos y Sirenas

de disparos y sirenas, y 3 % en el de gritos y sirenas.

5.4 Resultados de Evaluación: Métrica de Peso del Modelo

En la Figura 5.6, se presenta el resultado posterior a aplicar el Benchmark, sobre la arquitectura de los modelos fusionado, individuales y biclase. Extrayendo la métrica del peso en Megabytes del peso de los modelos para las dos configuraciones evaluadas y los 3 tipos de datos de representación. Para los modelos individuales se tiene en cuenta el peso de los dos modelos que se deben almacenar para la ejecución de ambas de las tareas.

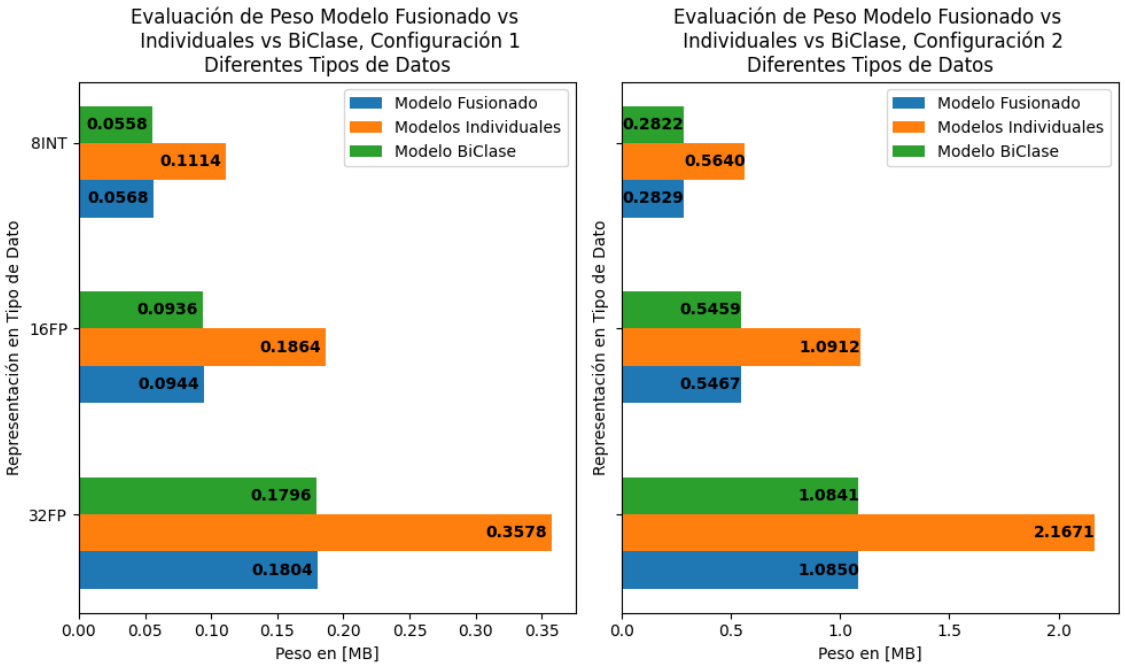

Figura 5.6: Evaluación de Peso del Modelo Fusionado vs Individuales vs BiClase, Bajo Diferentes Tipos de Datos para Configuración de Características 1 y 2, en tarjeta VIM3 PRO

Observando los resultados encontramos que para la configuración 1 al pasar de punto flotante de 32 bits a punto flotante de 16 bits se reduce en 1.99x el tamaño del modelo, y al pasar de punto flotante de 16 bits a entero de 8 bits se reduce en 1.93x el tamaño del modelo. Por tanto al pasar de punto flotante de 32 a entero de 8 bits se reduce en 3.84x el tamaño del modelo. Esto aplicable tanto para el modelo fusionado, como para los modelos individuales y el modelo biclase.
Hallazgos similares presenta la configuración 2, donde al pasar de punto flotante de 32 bits a punto flotante de 16 bits se reduce en 1.92x el tamaño del modelo, y al pasar de punto flotante de 16 bits a entero de 8 bits se reduce en 1.67x el tamaño del modelo. Por tanto al al pasar de punto flotante de 32 a entero de 8 bits se reduce en 3.2x el tamaño del modelo.

Respecto a la relación entre los pesos de los modelos de la configuración 1 y la configuración 2, aproximadamente aplicable para todos los tipo de datos, la configuración

2 es 5x más pesada en MegaBytes que la configuración 1. Por último, el peso de los modelos individuales es 2x más pesado que los modelos fusionado y biclase, estos últimos son dimensional-mente similares en peso.

5.5 Resultados de Evaluación: Métrica de Tiempo de Ejecución

Los resultado posterior a aplicar el Benchmark, sobre la arquitectura de los modelos fusionado, individuales y biclase, para la métrica de Tiempo de Ejecución de una inferencia del modelo, son presentados en la Figura 5.7. Para los modelos individuales se tiene en cuenta el tiempo de ejecución de los dos modelos que se deben ejecutar en simultáneo para el reconocimiento de ambas de las tareas.

Al analizar los resultados, encontramos que tanto para la configuración 1 como para la configuración 2, la complejidad de computo al ejecutar una inferencia del modelo medida en tiempo de ejecución. Es igual para la tarjeta VIM3 PRO sea que el modelo se encuentre en tipo de datos de punto flotante de 32 bits o punto flotante de 16 bits. Por parte de la cuantización a enteros de 8 bits, si se encontró una reducción de 1.6x en el tiempo de ejecución para la configuración 1, y una reducción de 1.4x para la configuración 2. Comparando el tiempo de ejecución de manera general de la configuración 1 con la configuración 2, para todos los modelos y tipos de datos. La configuración 2 es entre 36x y 42x más lenta que la configuración 1 en realizar una ejecución de inferencia de los modelos. Para los modelos fusionados y biclase presentan igual complejidad en tiempo de ejecución, entre sus respectivas configuraciones. Al comparar estos con la ejecución de los modelos individuales son 2x más rápidos, debido al requerimiento de la ejecución de los dos modelos individuales para realizar las tareas de los anteriores mencionados.

5.6 Resultados de Evaluación: Métrica de Huella en Memoria

En esta última sección presentamos en la Figura 5.8, la utilización de memoria en MegaBytes medida por el benchmark para una ejecución de inferencia de los modelos. Encontramos que en cuanto a uso de memoria, el tipo de dato de punto flotante de 16 bits presentó mayores requerimientos frente a los otros formatos en ambas configuraciones. Aproximadamente utilizó 1.05x y 1.14x más memoria los modelos de punto flotante de 16 bits que los modelos de punto flotante de 32 bits, en la tarjeta VIM3 PRO. Al comparar la cuantización del modelo de punto flotante de 32 bits a el modelo en enteros de 8 bits, para la configuración 1 es euivalente el uso de memoria, pero para la configuración 2 se logra un reducción de uso aproximada de 1.25x.
Respecto a los modelos fusionados y biclase presentan equivalencia en utilización de memoria en ejecución, entre sus respectivas configuraciones. Al comparar estos con la ejecución de los modelos individuales requieren 2x menos memoria, debido al requerimiento de la ejecución de los dos modelos individuales para realizar ambas de las

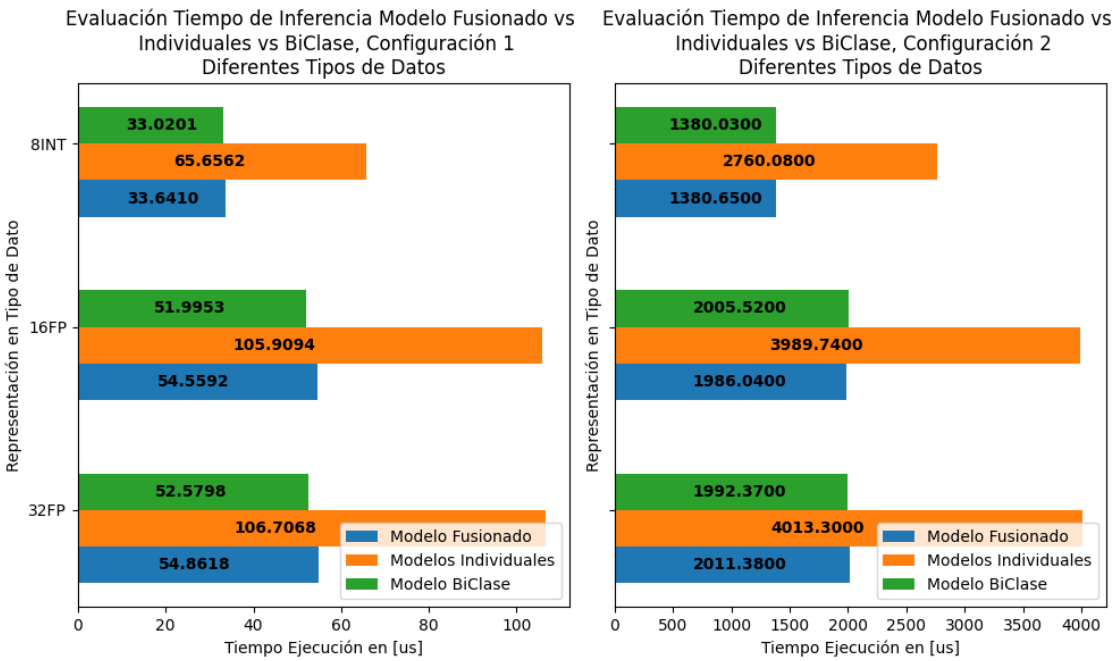

Figura 5.7: Evaluación de Tiempo de Ejecución de Modelo Fusionado Comparado con Modelos Individuales y Modelo BiClase para Configuración de Características 1 y 2, en tarjeta VIM3 PRO

tareas.

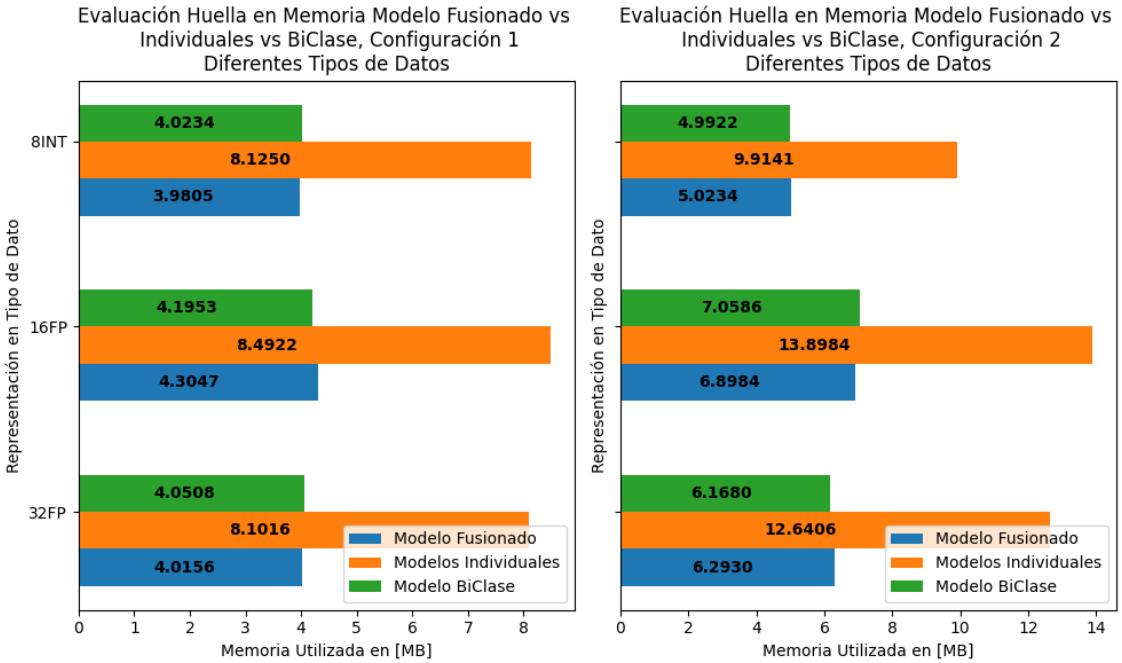

Figura 5.8: Evaluación de Huella de Memoria en la Ejecución de Modelo Fusionado Comparado con Modelos Individuales y Modelo BiClase para Configuración de Características 1 y 2, en tarjeta VIM3 PRO

5.7 Resultados Tiempo de Ejecución en Implementación Real

Con el propósito de complementar los resultados obtenidos en las secciones anteriores, donde se compararon los modelos a través de métricas específicas, se llevó a cabo una evaluación del tiempo de ejecución de la solución implementada. Esta evaluación consideró todos los pasos involucrados en el proceso, desde la adquisición de datos hasta la predicción del modelo.Esto implica, la grabación de audio, la extracción de las características MFCC y la ejecución de los modelos. Para esta evaluación, se utilizó como referencia el modelo fusionado que mostró el mejor desempeño en las estrategias de fusión evaluadas. Específicamente, se empleó la estrategia de fusión por filtros con una métrica de similitud de umbral del 80 % y una optimización del desempate promedio aritmético.

El experimento de implementación real se desarrolló mediante un script en Python, el cual permitió medir los tiempos de ejecución en cada etapa del proceso: grabación de audio, preprocesamiento de la señal y predicción del modelo. Cada audio grabado tuvo una duración de 4 segundos, y se repitió el experimento para cada modelo en cinco ocasiones. Los tiempos de ejecución se promediaron aritméticamente y se presentan en la Tabla 5.1.

Tabla 5.1: Tiempos de Ejecución Implementación Real

Configuración 1			
Modelos	**Grabación [ms]**	**Preprocesamiento [ms]**	**Predicción [ms]**
Fusionado	4060,1529	2582,3012	0,1799
Biclase	4063,1688	2571,2996	0,1715
Individuales	4061,5569	2585,2255	0,2821
Configuración 2			
Modelos	**Grabación [ms]**	**Preprocesamiento [ms]**	**Predicción [ms]**
Fusionado	4063,8667	2583,8482	2,2952
Biclase	4061,6578	2593,8457	2,3050
Individuales	4060,9773	2595,4971	4,5008

Con los resultados de los tiempos de ejecución obtenidos en la implementación real, tal como se muestra en la Tabla 5.1, se puede apreciar una consistencia notable independientemente de la configuración utilizada. Específicamente, se destaca que el tiempo requerido para el preprocesamiento de extracción de los MFCC se mantiene prácticamente constante. Este hallazgo sugiere que la variación en los tiempos totales

de ejecución está más influenciada por otros factores, como los requisitos de precisión de los modelos y la capacidad de almacenamiento.

Al comparar el tiempo de predicción entre las dos configuraciones, se observa que la Configuración 1 es significativamente más rápida que la Configuración 2, siendo entre $12\times$ y $16\times$ más rápida. Este contraste se atribuye principalmente al número de FLOPS requeridos por cada configuración, siendo considerablemente mayor en la Configuración 2, como se detalla en la Tabla 3.5.

En cuanto al tiempo de predicción, se encuentra una similitud con las mediciones presentadas en la Sección 5.5 de los resultados de evaluación con la métrica de tiempo de ejecución. Tanto los modelos fusionados como los biclase muestran una complejidad de ejecución similar en sus respectivas configuraciones. Además, al comparar los modelos fusionados con los individuales, se observa que los primeros son aproximadamente $2\times$ más rápidos, lo cual se atribuye a la necesidad de ejecutar únicamente un único modelo en lugar de dos.

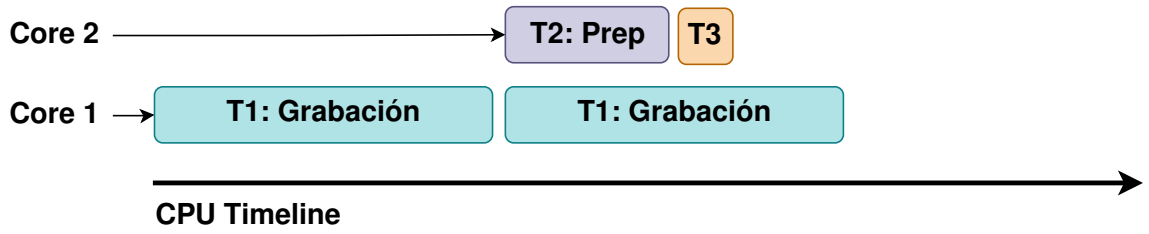

Figura 5.9: Implementación aplicación en dos hilos, donde T1: Grabación, T2: Preprocesamiento, T3: Predicción.

Basándonos en los resultados de tiempo medidos, esta implementación real de la solución demuestra ser altamente adaptable para un entorno de monitoreo continuo. Al dividir las cargas de trabajo en dos hilos diferentes, como se muestra en la Figura 5.9, se logra una optimización eficiente del tiempo de ejecución. Mientras un hilo se encarga de la gestión del proceso de grabación de 4 segundos, el otro se dedica al preprocesamiento y la predicción del modelo. Para ambas configuraciones se cumple que la duración del tiempo de grabación es mayor que la suma del tiempo dedicado al preprocesamiento y a la predicción.

Capítulo 6

Conclusiones y Trabajo Futuro

6.1 Conclusiones

En este trabajo de investigación planteamos y exploramos diferentes estrategias de fusión de redes neuronales convolucionales, con las cuales encontramos que es posible construir un modelo fusionado a partir de dos modelos individuales en el que se iguale o mejore en algunos casos, el desempeño en F1-Score promedio en la clasificación de las tareas individuales. Esto, sin comprometer el desempeño en métricas de huella en memoria, peso del modelo y tiempo de ejecución en un contexto de dispositivo de borde.

Las estrategias de fusión más eficaces se identificaron como la fusión por filtros, evaluando el rendimiento sin métricas de similitud con optimización promedio, y la fusión por filtros con métricas de similitud con optimización promedio y utilizando un umbral inferior al 80 %. Ambas estrategias superaron el rendimiento promedio de los modelos individuales y del modelo biclase de referencia. Entre estas dos alternativas, la fusión por filtros sin métricas de similitud se destaca como la opción menos costosa computacionalmente. Esto se debe a que la estrategia que incorpora métricas de similitud implica un costo computacional elevado al extraer estas métricas de todos los filtros entre los modelos.

El uso de la estrategia de fusión en costos computacionales, como huella en memoria, peso del modelo y tiempo de ejecución, beneficia en 2x la ejecución del modelo fusionado en comparación con los modelos individuales. Se observó que el rendimiento del modelo fusionado es comparable en todas las métricas con la ejecución de un único modelo, cumpliendo con las tareas de clasificación correspondientes a dos modelos individuales. A diferencia del modelo biclase, el modelo fusionado tiene el beneficio de superar el desempeño promedio en F1-Score.

La estrategia de fusión demuestra ser beneficiosa en entornos donde no se dispone de conjuntos de entrenamiento para el modelo biclase. En lugar de depender de conjuntos de entrenamiento, la fusión se puede implementar utilizando un conjunto de datos de prueba con dos modelos individuales previamente entrenados. Esta estrategia resulta ventajosa en el despliegue de modelos dinámicos en sistemas con recursos limitados, permitiendo la selección y fusión de modelos específicos entrenados para tareas

particulares en un sistema más amplio. Extrapolando este enfoque a un sistema de seguridad ciudadana, donde diferentes zonas de vigilancia pueden tener distintos requisitos en términos de eventos acústicos de interés, las técnicas de fusión de CNN podrían ser aplicadas para obtener un modelo fusionado adaptado a los eventos de interés de cada zona y horario específico. Como evidencian los resultados, este enfoque requiere menos recursos de almacenamiento en comparación con el entrenamiento de múltiples modelos multiclase.

La cuantización de los modelos resultó en notables reducciones en el tamaño, logrando una disminución de hasta 3.84x al transicionar de punto flotante de 32 bits a enteros de 8 bits. Además, se observó una reducción significativa en el tiempo de ejecución. La configuración 2 demostró ser entre 36x y 42x más lenta en términos de tiempo de ejecución en comparación con la configuración 1, independientemente del tipo de dato del modelo. Se identificó una equivalencia en el uso de memoria en ejecución entre modelos fusionados y biclase en sus configuraciones respectivas, mientras que los modelos individuales requirieron aproximadamente el doble de memoria debido a la ejecución simultánea de dos modelos separados para cumplir con sus tareas asignadas.

De manera precedente a las estrategias de fusión, se presentó la exploración del espacio de diseño en los hiperparámetros de la extracción de características del audio mediante MFCC. En donde, de manera general encontramos que, los modelos que utilizan valores de parámetros más altos (como coeficientes MFCC, tamaño de ventana, pasos más pequeños, etc.) tienden a tener un mejor rendimiento en el reconocimiento de eventos acústicos. Sin embargo, estos modelos son más costosos computacionalmente y pueden tardar más tiempo en entrenarse y evaluar. Por lo tanto, existe un equilibrio entre el número de parámetros y el rendimiento del modelo que debe considerarse en función del hardware disponible y las limitaciones de tiempo. Una forma de seleccionar ese punto óptimo de Pareto para cada modelo, es construir una función de coste que combine ambas métricas. Según el contexto de la aplicación, el diseñador del sistema debe priorizar la precisión del modelo o el coste computacional.

6.2 Trabajo Futuro

Como trabajo futuro, a partir de las estrategias de fusión planteadas, es evaluar y generalizar la estrategia de fusión para más de dos modelos individuales. En este trabajo lo probamos únicamente con la fusión de dos modelos.

Con la métrica de similitud aplicada sobre los filtros normalizados, encontramos que existían mayores valores que con la métrica aplicada sobre los filtros sin normalización min-max. Esto, lo explicamos a través de que al estar normalizados los filtros, se contemplaba además la similitud de filtros escalares entre ellos, que con un valor escalar se podría representar un filtro en termino de otro. En este caso, se podría abrir una línea de trabajo futuro en la cual se exploten estas capacidades de similitud entre modelos y pueda hacerse optimizaciones aún mayores, ya sea a nivel de arquitectura o almacenamiento al momento de realizar fusión entre modelos.

Bibliografía

[1] "Grupo de Convivencia y Seguridad Ciudadana." [Online]. Available: https://2022.dnp.gov.co/programas/justicia-seguridad-y-gobierno/grupo-de-convivencia-y-seguridad-ciudadana 1

[2] "Paz y justicia - Desarrollo Sostenible. Desarrollo Sostenible." [Online]. Available: https://www.un.org/sustainabledevelopment/es/peace-justice/ 1

[3] "Encuesta de Percepción Ciudadana de Medellín, 2023." [Online]. Available: https://www.medellincomovamos.org/ 1

[4] T. D. Räty, "Survey on contemporary remote surveillance systems for public safety," *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, vol. 40, no. 5, pp. 493–515, 2010. 1

[5] W. Huang, T. K. Chiew, H. Li, T. S. Kok, and J. Biswas, "Scream detection for home applications," in *2010 5th IEEE Conference on Industrial Electronics and Applications*. IEEE, 2010, pp. 2115–2120. 1, 2.1

[6] H. Nanjo, T. Nishiura, and H. Kawano, "Acoustic-based security system: Towards robust understanding of emergency shout," in *2009 Fifth International Conference on Information Assurance and Security*, vol. 1. IEEE, 2009, pp. 725–728. 1, 2.1

[7] J.-S. Park and S.-H. Kim, "Sound learning–based event detection for acoustic surveillance sensors," *Multimedia Tools and Applications*, vol. 79, no. 23-24, pp. 16 127–16 139, 2020. 1, 2.1

[8] Y. Arslan, "A new approach to real time impulsive sound detection for surveillance applications," *arXiv preprint arXiv:1906.06586*, 2019. 1, 2.1

[9] S. Ntalampiras, I. Potamitis, and N. Fakotakis, "On acoustic surveillance of hazardous situations," in *2009 IEEE International Conference on Acoustics, Speech and Signal Processing*. IEEE, 2009, pp. 165–168. 1, 2.1

[10] A. Morehead, L. Ogden, G. Magee, R. Hosler, B. White, and G. Mohler, "Low cost gunshot detection using deep learning on the raspberry pi," in *2019 IEEE International Conference on Big Data (Big Data)*. IEEE, 2019, pp. 3038–3044. 1, 2.1, 2.3.1

[11] S. U. Rahman, A. Khan, S. Abbas, F. Alam, and N. Rashid, "Hybrid system for automatic detection of gunshots in indoor environment," *Multimedia Tools and Applications*, vol. 80, pp. 4143–4153, 2021. 1, 2.1

[12] I. L. Freire and J. A. Apolinário Jr, "Gunshot detection in noisy environments," in *Proceeding of the 7th International Telecommunications Symposium, Manaus, Brazil*, vol. 1, no. 4, 2010. 1, 2.1

[13] M. Sigmund and M. Hrabina, "Efficient feature set developed for acoustic gunshot detection in open space," *Elektronika ir Elektrotechnika*, vol. 27, no. 4, pp. 62–68, 2021. 1, 2.1

[14] R. Torres, D. Battaglino, and L. Lepauloux, "Baby cry sound detection: A comparison of hand crafted features and deep learning approach," in *Engineering Applications of Neural Networks: 18th International Conference, EANN 2017, Athens, Greece, August 25–27, 2017, Proceedings*. Springer, 2017, pp. 168–179. 1

[15] Y. Lavner, R. Cohen, D. Ruinskiy, and H. IJzerman, "Baby cry detection in domestic environment using deep learning," in *2016 IEEE international conference on the science of electrical engineering (ICSEE)*. IEEE, 2016, pp. 1–5. 1

[16] Y. Li, X. Li, Y. Zhang, M. Liu, and W. Wang, "Anomalous sound detection using deep audio representation and a blstm network for audio surveillance of roads," *Ieee Access*, vol. 6, pp. 58 043–58 055, 2018. 1, 2.1

[17] N. Almaadeed, M. Asim, S. Al-Maadeed, A. Bouridane, and A. Beghdadi, "Automatic detection and classification of audio events for road surveillance applications," *Sensors*, vol. 18, no. 6, p. 1858, 2018. 1, 2.1

[18] S. Jamil, Fawad, M. Rahman, A. Ullah, S. Badnava, M. Forsat, and S. S. Mirjavadi, "Malicious uav detection using integrated audio and visual features for public safety applications," *Sensors*, vol. 20, no. 14, p. 3923, 2020. 1, 2.1

[19] M. Z. Anwar, Z. Kaleem, and A. Jamalipour, "Machine learning inspired sound-based amateur drone detection for public safety applications," *IEEE Transactions on Vehicular Technology*, vol. 68, no. 3, pp. 2526–2534, 2019. 1, 2.1

[20] M. Ferroudj, A. Truskinger, M. Towsey, L. Zhang, J. Zhang, and P. Roe, "Detection of rain in acoustic recordings of the environment," in *PRICAI 2014: Trends in Artificial Intelligence: 13th Pacific Rim International Conference on Artificial Intelligence, Gold Coast, QLD, Australia, December 1-5, 2014. Proceedings 13*. Springer, 2014, pp. 104–116. 1

[21] B. B. Ma, J. A. Nystuen, and R.-C. Lien, "Prediction of underwater sound levels from rain and wind," *The journal of the acoustical society of America*, vol. 117, no. 6, pp. 3555–3565, 2005. 1

[22] B. Kim and B. Pardo, "Sound event detection using point-labeled data," in *2019 IEEE Workshop on Applications of Signal Processing to Audio and Acoustics (WASPAA)*. IEEE, 2019, pp. 1–5. 1

[23] D. Hoiem, Y. Ke, and R. Sukthankar, "Solar: Sound object localization and retrieval in complex audio environments," in *Proceedings.(ICASSP'05). IEEE*

International Conference on Acoustics, Speech, and Signal Processing, 2005., vol. 5. IEEE, 2005, pp. v–429. 1

[24] D. Stowell, M. D. Wood, H. Pamuła, Y. Stylianou, and H. Glotin, "Automatic acoustic detection of birds through deep learning: the first bird audio detection challenge," *Methods in Ecology and Evolution*, vol. 10, no. 3, pp. 368–380, 2019. 1

[25] Y. Komori, K. Ohno, T. Fujieda, T. Suzuki, and S. Tadokoro, "Detection of continuous barking actions from search and rescue dogs' activities data," in *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2015, pp. 630–635. 1

[26] M.-A. Carbonneau, N. Lezzoum, J. Voix, and G. Gagnon, "Detection of alarms and warning signals on an digital in-ear device," *International Journal of Industrial Ergonomics*, vol. 43, no. 6, pp. 503–511, 2013. 1

[27] L. Marchegiani and P. Newman, "Listening for sirens: Locating and classifying acoustic alarms in city scenes," *IEEE transactions on intelligent transportation systems*, vol. 23, no. 10, pp. 17 087–17 096, 2022. 1

[28] P. Arce, D. Salvo, G. Piñero, and A. Gonzalez, "Fiware based low-cost wireless acoustic sensor network for monitoring and classification of urban soundscape," *Computer Networks*, vol. 196, p. 108199, 2021. 1, 2.1

[29] D. Pramanick, H. Ansar, H. Kumar, S. Pranav, R. Tengshe, and B. Fatimah, "Deep learning based urban sound classification and ambulance siren detector using spectrogram," in *2021 12th International Conference on Computing Communication and Networking Technologies (ICCCNT)*. IEEE, 2021, pp. 1–6. 1, 2.1

[30] S. Agarwal, K. Khatter, and D. Relan, "Security threat sounds classification using neural network," in *2021 8th International Conference on Computing for Sustainable Global Development (INDIACom)*. IEEE, 2021, pp. 690–694. 1, 2.1

[31] E. Vozáriková, M. Pleva, J. Juhár, and A. Cizmár, "Surveillance system based on the acoustic events detection," *Journal of Electrical and Electronics Engineering*, vol. 4, no. 1, p. 255, 2011. 1, 2.1

[32] A. M. Ghosh and K. Grolinger, "Edge-cloud computing for internet of things data analytics: Embedding intelligence in the edge with deep learning," *IEEE Transactions on Industrial Informatics*, vol. 17, no. 3, pp. 2191–2200, 2020. 1.1

[33] Q. Kong, I. Sobieraj, W. Wang, and M. Plumbley, "Deep neural network baseline for dcase challenge 2016," *Proceedings of DCASE 2016*, 2016. 1.1, 2.1, 3

[34] S.-Y. Chou, J.-S. R. Jang, and Y.-H. Yang, "Framecnn: A weakly-supervised learning framework for frame-wise acoustic event detection and classification," *Recall*, vol. 14, pp. 55–64, 2017. 1.1, 2.1, 3

[35] E. Cakir, S. Adavanne, G. Parascandolo, K. Drossos, and T. Virtanen, "Convolutional recurrent neural networks for bird audio detection," in *2017 25th European signal processing conference (EUSIPCO)*. IEEE, 2017, pp. 1744–1748. 1.1, 2.1, 3

[36] E. Tsalera, A. Papadakis, and M. Samarakou, "Comparison of pre-trained cnns for audio classification using transfer learning," *Journal of Sensor and Actuator Networks*, vol. 10, no. 4, p. 72, 2021. 1.1, 2.1, 3.3

[37] S. J. Johnston, P. J. Basford, C. S. Perkins, H. Herry, F. P. Tso, D. Pezaros, R. D. Mullins, E. Yoneki, S. J. Cox, and J. Singer, "Commodity single board computer clusters and their applications," *Future Generation Computer Systems*, vol. 89, pp. 201–212, 2018. 1.1

[38] G. E. Hinton and D. Van Camp, "Keeping the neural networks simple by minimizing the description length of the weights," in *Proceedings of the sixth annual conference on Computational learning theory*, 1993, pp. 5–13. 1.1, 2.4

[39] W. Zhe, J. Lin, V. Chandrasekhar, and B. Girod, "Optimizing the bit allocation for compression of weights and activations of deep neural networks," in *2019 IEEE International Conference on Image Processing (ICIP)*. IEEE, 2019, pp. 3826–3830. 1.1

[40] M. Courbariaux, Y. Bengio, and J.-P. David, "Binaryconnect: Training deep neural networks with binary weights during propagations," *Advances in neural information processing systems*, vol. 28, 2015. 1.1, 2.4, 2.4.1

[41] I. Hubara, M. Courbariaux, D. Soudry, R. El-Yaniv, and Y. Bengio, "Quantized neural networks: Training neural networks with low precision weights and activations," *The Journal of Machine Learning Research*, vol. 18, no. 1, pp. 6869–6898, 2017. 1.1, 2.4.1

[42] Y. LeCun, J. Denker, and S. Solla, "Optimal brain damage," *Advances in neural information processing systems*, vol. 2, 1989. 1.1, 2.4, 2.4.1

[43] T.-J. Yang, Y.-H. Chen, and V. Sze, "Designing energy-efficient convolutional neural networks using energy-aware pruning," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 5687–5695. 1.1, 2.4.1

[44] N.-M. Ho and W.-F. Wong, "Exploiting half precision arithmetic in nvidia gpus," in *2017 IEEE High Performance Extreme Computing Conference (HPEC)*. IEEE, 2017, pp. 1–7. 1.1, 2.4, 2.4.2

[45] N. Burgess, J. Milanovic, N. Stephens, K. Monachopoulos, and D. Mansell, "Bfloat16 processing for neural networks," in *2019 IEEE 26th Symposium on Computer Arithmetic (ARITH)*. IEEE, 2019, pp. 88–91. 1.1, 2.4, 2.4.2

[46] A. A. Mallouh, Z. Qawaqneh, and B. D. Barkana, "Combining two different dnn architectures for classifying speakers age and gender," in *International Conference on Bio-inspired Systems and Signal Processing*, vol. 5. SCITEPRESS, 2017, pp. 112–117. 1.1, 2.5.3, 4.5

[47] P. Giammatteo, F. V. Fiordigigli, L. Pomante, T. Di Mascio, and F. Caruso, "Age & gender classifier for edge computing," in *2019 8th Mediterranean Conference on Embedded Computing (MECO)*. IEEE, 2019, pp. 1–4. 1.1, 2.5.2

[48] C. Nakkach, A. Zrelli, and T. Ezzeddine, "Deep learning algorithms enabling event detection: A review," in *2nd International Conference on Industry 4.0 and Artificial Intelligence (ICIAI 2021)*. Atlantis Press, 2022, pp. 170–175. 2.1

[49] X. Xia, R. Togneri, F. Sohel, and D. Huang, "Random forest classification based acoustic event detection utilizing contextual-information and bottleneck features," *Pattern Recognition*, vol. 81, pp. 1–13, 2018. 2.1

[50] H. Phan, M. Maaß, R. Mazur, and A. Mertins, "Random regression forests for acoustic event detection and classification," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 23, no. 1, pp. 20–31, 2014. 2.1

[51] A. Diment, T. Heittola, and T. Virtanen, "Sound event detection for office live and office synthetic aasp challenge," *Proc. IEEE AASP Challenge on Detection Classif. Acoust. Scenes Events (WASPAA)*, 2013. 2.1

[52] J. Schröder, N. Moritz, M. R. Schädler, B. Cauchi, K. Adiloglu, J. Anemüller, S. Doclo, B. Kollmeier, and S. Goetze, "On the use of spectro-temporal features for the ieee aasp challenge detection and classification of acoustic scenes and events," in *2013 IEEE Workshop on Applications of Signal Processing to Audio and Acoustics*. IEEE, 2013, pp. 1–4. 2.1

[53] W. Nogueira, G. Roma, and P. Herrera, "Automatic event classification using front end single channel noise reduction, mfcc features and a support vector machine classifier," *IEEE AASP Challenge on Detection and Classification of Acoustic Scenes and Events*, pp. 1–2, 2013. 2.1

[54] Y. Wei, Z. Gong, S. Yang, K. Ye, and Y. Wen, "Edgecrnn: an edge-computing oriented model of acoustic feature enhancement for keyword spotting," *Journal of Ambient Intelligence and Humanized Computing*, pp. 1–11, 2022. 2.1

[55] S. Zhilibayev, "Real-time speech emotion recognition (rser) in wireless multimedia sensor networks," 2021. 2.1

[56] M. U. Liaquat, H. S. Munawar, A. Rahman, Z. Qadir, A. Z. Kouzani, and M. P. Mahmud, "Localization of sound sources: A systematic review," *Energies*, vol. 14, no. 13, p. 3910, 2021. 2.1

[57] J. A. Belloch, J. M. Badia, F. D. Igual, M. Cobos, and E. S. Quintana-Ortí, "On the use of a gpu-accelerated mobile device processor for sound source localization," *Procedia Computer Science*, vol. 108, pp. 586–595, 2017. 2.1

[58] J. Kotus, K. Łopatka, and A. Czyzewski, "Detection and localization of selected acoustic events in 3d acoustic field for smart surveillance applications," in *Multimedia Communications, Services and Security: 4th International Conference, MCSS 2011, Krakow, Poland, June 2-3, 2011. Proceedings 4*. Springer, 2011, pp. 55–63. 2.1

[59] J. H. Park, W. Cho, and S.-C. Kim, "Improving acoustic localization accuracy by applying interaural level difference and support vector machine for aoa outlier removal," in *2021 International Conference on Electronics, Information, and Communication (ICEIC)*. IEEE, 2021, pp. 1–4. 2.1

[60] C. Zieger, A. Brutti, and P. Svaizer, "Acoustic based surveillance system for intrusion detection," in *2009 Sixth IEEE International Conference on Advanced Video and Signal Based Surveillance*. IEEE, 2009, pp. 314–319. 2.1

[61] F. Rosenblatt, "The perceptron: a probabilistic model for information storage and organization in the brain." *Psychological review*, vol. 65, no. 6, p. 386, 1958. 2.3

[62] D. E. Rumelhart, G. E. Hinton, R. J. Williams *et al.*, "Learning internal representations by error propagation," 1985. 2.3

[63] E. Kavlakoglu, "Ai vs. machine learning vs. deep learning vs. neural networks: whats the difference?: Ibm; 2020." [Online]. Available: https://www.ibm.com/blog/ai-vs-machine-learning-vs-deep-learning-vs-neural-networks/ 2.3

[64] A. Intelligence and M. L. as tools to augment Human Intelligence, "Modeling threshold logic neural networks: McCulloch-Pitts Model and Rosenblatts perceptrons," Dec 2020. [Online]. Available: https://dominicm73.blogspot.com/2020/08/modeling-threshold-logic-neurons-and.html 2.3

[65] M. Gurucharan, "Basic cnn architecture: Explaining 5 layers of convolutional neural network," *URL: https://www. upgrad. com/blog/basic-cnn-architecture*, 2020. 2.3.1

[66] A. laddad, "Basic understanding of lstm," Mar 2019. [Online]. Available: https://blog.goodaudience.com/basic-understanding-of-lstm-539f3b013f1e 2.3.2

[67] W. Y. N. Naing, Z. Z. Htike, and A. A. Shafie, "Real time end-to-end glass break detection system using lstm deep recurrent neural network," 2019. 2.3.2

[68] M. Verhelst and B. Moons, "Embedded deep neural network processing: Algorithmic and processor techniques bring deep learning to iot and edge devices," *IEEE Solid-State Circuits Magazine*, vol. 9, no. 4, pp. 55–65, 2017. 2.4

[69] T. Chen, Z. Du, N. Sun, J. Wang, C. Wu, Y. Chen, and O. Temam, "Diannao: A small-footprint high-throughput accelerator for ubiquitous machine-learning," *ACM SIGARCH Computer Architecture News*, vol. 42, no. 1, pp. 269–284, 2014. 2.4

[70] S. Chakradhar, M. Sankaradas, V. Jakkula, and S. Cadambi, "A dynamically configurable coprocessor for convolutional neural networks," in *Proceedings of the 37th annual international symposium on Computer architecture*, 2010, pp. 247–257. 2.4

[71] S. Srinivas and R. V. Babu, "Data-free parameter pruning for deep neural networks," *arXiv preprint arXiv:1507.06149*, 2015. 2.4.1

[72] Z. Mariet and S. Sra, "Diversity networks: Neural network compression using determinantal point processes," *arXiv preprint arXiv:1511.05077*, 2015. 2.4.1

[73] H. Li, A. Kadav, I. Durdanovic, H. Samet, and H. P. Graf, "Pruning filters for efficient convnets," *arXiv preprint arXiv:1608.08710*, 2016. 2.4.1

[74] Y. He, X. Zhang, and J. Sun, "Channel pruning for accelerating very deep neural networks," in *Proceedings of the IEEE international conference on computer vision*, 2017, pp. 1389–1397. 2.4.1

[75] X. Gao, Y. Zhao, Ł. Dudziak, R. Mullins, and C.-z. Xu, "Dynamic channel pruning: Feature boosting and suppression," *arXiv preprint arXiv:1810.05331*, 2018. 2.4.1

[76] C. Bucilu, R. Caruana, and A. Niculescu-Mizil, "Model compression," in *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, 2006, pp. 535–541. 2.4.1

[77] G. Chen, W. Choi, X. Yu, T. Han, and M. Chandraker, "Learning efficient object detection models with knowledge distillation," *Advances in neural information processing systems*, vol. 30, 2017. 2.4.1

[78] A. Agrawal, S. M. Mueller, B. M. Fleischer, X. Sun, N. Wang, J. Choi, and K. Gopalakrishnan, "Dlfloat: A 16-b floating point format designed for deep learning training and inference," in *2019 IEEE 26th Symposium on Computer Arithmetic (ARITH)*. IEEE, 2019, pp. 92–95. 2.4.2

[79] Y.-H. Chen, J. Emer, and V. Sze, "Eyeriss: A spatial architecture for energy-efficient dataflow for convolutional neural networks," *ACM SIGARCH computer architecture news*, vol. 44, no. 3, pp. 367–379, 2016. 2.4.3

[80] P. Panda, A. Sengupta, and K. Roy, "Conditional deep learning for energy-efficient and enhanced pattern recognition," in *2016 Design, Automation & Test in Europe Conference & Exhibition (DATE)*. IEEE, 2016, pp. 475–480. 2.4.3, 2.5.1

[81] S. Teerapittayanon, B. McDanel, and H.-T. Kung, "Branchynet: Fast inference via early exiting from deep neural networks," in *2016 23rd international conference on pattern recognition (ICPR)*. IEEE, 2016, pp. 2464–2469. 2.4.3, 2.5.1

[82] ——, "Distributed deep neural networks over the cloud, the edge and end devices," in *2017 IEEE 37th international conference on distributed computing systems (ICDCS)*. IEEE, 2017, pp. 328–339. 2.4.3, 2.5.1

[83] J. Lee, B. Varghese, R. Woods, and H. Vandierendonck, "Tod: Transprecise object detection to maximise real-time accuracy on the edge," in *2021 IEEE 5th International Conference on Fog and Edge Computing (ICFEC)*. IEEE, 2021, pp. 53–60. 2.4.3, 2.5.1

[84] W. Lou, L. Xun, A. Sabet, J. Bi, J. Hare, and G. V. Merrett, "Dynamic-ofa: Runtime dnn architecture switching for performance scaling on heterogeneous embedded platforms," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021, pp. 3110–3118. 2.4.3, 2.5.1

[85] J.-H. Choi and J.-S. Lee, "Embracenet: A robust deep learning architecture for multimodal classification," *Information Fusion*, vol. 51, pp. 259–270, 2019. 2.5.2, 2.5.2.3

[86] R. F. Rachmadi, I. K. E. Purnama, S. M. S. Nugroho, and Y. K. Suprapto, "Image-based kinship verification using fusion convolutional neural network," in *2019 IEEE 11th International Workshop on Computational Intelligence and Applications (IWCIA)*. IEEE, 2019, pp. 59–65. 2.5.2, 2.5.2.3

[87] J. Williams, R. Comanescu, O. Radu, and L. Tian, "Dnn multimodal fusion techniques for predicting video sentiment," in *Proceedings of grand challenge and workshop on human multimodal language (Challenge-HML)*, 2018, pp. 64–72. 2.5.2, 2.5.2.1, 2.5.2.2, 2.5.2.3, 2.5.2.3

[88] J. Ngiam, A. Khosla, M. Kim, J. Nam, H. Lee, and A. Y. Ng, "Multimodal deep learning," in *Proceedings of the 28th international conference on machine learning (ICML-11)*, 2011, pp. 689–696. 2.5.2

[89] Y. Wang, W. Huang, F. Sun, T. Xu, Y. Rong, and J. Huang, "Deep multimodal fusion by channel exchanging," *Advances in neural information processing systems*, vol. 33, pp. 4835–4845, 2020. 2.5.2

[90] Y. Zhang, D. Sidibé, O. Morel, and F. Mériaudeau, "Deep multimodal fusion for semantic image segmentation: A survey," *Image and Vision Computing*, vol. 105, p. 104042, 2021. 2.5.2

[91] P. K. Atrey, M. A. Hossain, A. El Saddik, and M. S. Kankanhalli, "Multimodal fusion for multimedia analysis: a survey," *Multimedia systems*, vol. 16, pp. 345–379, 2010. 2.5.2

[92] Y. Liu, J. Li, Q. Yan, X. Yuan, C. Zhao, I. Reid, and C. Cadena, "3d gated recurrent fusion for semantic scene completion," *arXiv preprint arXiv:2002.07269*, 2020. 2.5.2.3, 2.5.2.3

[93] Y. Zhang and Q. Yang, "A survey on multi-task learning," *IEEE Transactions on Knowledge and Data Engineering*, vol. 34, no. 12, pp. 5586–5609, 2021. 2.5.3

[94] S. J. Pan and Q. Yang, "A survey on transfer learning," *IEEE Transactions on knowledge and data engineering*, vol. 22, no. 10, pp. 1345–1359, 2009. 2.5.3

[95] M. S. Sorower, "A literature survey on algorithms for multi-label learning," *Oregon State University, Corvallis*, vol. 18, no. 1, p. 25, 2010. 2.5.3

[96] H. Borchani, G. Varando, C. Bielza, and P. Larranaga, "A survey on multi-output regression," *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, vol. 5, no. 5, pp. 216–233, 2015. 2.5.3

[97] K. Ahmad and N. Conci, "How deep features have improved event recognition in multimedia: A survey," *ACM Transactions on Multimedia Computing, Communications, and Applications (TOMM)*, vol. 15, no. 2, pp. 1–27, 2019. 3

[98] F. Jiang, H. Li, Z. Zhang, and X. Zhang, "An event recognition method for fiber distributed acoustic sensing systems based on the combination of mfcc and cnn," in *2017 International Conference on Optical Instruments and Technology: Advanced Optical Sensors and Applications*, vol. 10618. SPIE, 2018, pp. 15–21. 3, 3.4

[99] J. Cao, M. Cao, J. Wang, C. Yin, D. Wang, and P.-P. Vidal, "Urban noise recognition with convolutional neural network," *Multimedia Tools and Applications*, vol. 78, pp. 29 021–29 041, 2019. 3

[100] A. Jain and O. Sharma, "Evaluation of mfcc for speaker verification on various windows," in *International conference on recent advances and innovations in engineering (ICRAIE-2014)*. IEEE, 2014, pp. 1–6. 3, 3.4

[101] M. Sadeghi and H. Marvi, "Optimal mfcc features extraction by differential evolution algorithm for speaker recognition," in *2017 3rd Iranian Conference on Intelligent Systems and Signal Processing (ICSPIS)*. IEEE, 2017, pp. 169–173. 3, 3.4

[102] J. Salamon, C. Jacoby, and J. P. Bello, "A dataset and taxonomy for urban sound research," in *Proceedings of the 22nd ACM international conference on Multimedia*, 2014, pp. 1041–1044. 3.2, 3.1

[103] T. Spadini, "Sound events for surveillance applications," Oct 2019. [Online]. Available: https://zenodo.org/record/3519845 3.2, 3.1

[104] S. A. Arusha, "Scream dataset," May 2022. [Online]. Available: https://www.kaggle.com/datasets/sanzidaakterarusha/scream-dataset?datasetId=1903423&sortBy=dateRun&tab=collaboration 3.2, 3.1

[105] Y. Chen, Y. Zhang, and Z. Duan, "Dcase2017 sound event detection using convolutional neural network," *Detection and Classification of Acoustic Scenes and Events*, 2017. 3.3

[106] S. Doshi, T. Patidar, S. Gautam, and R. Kumar, "Acoustic scene analysis and classification using densenet convolutional neural network," EasyChair, Tech. Rep., 2022. 3.3

[107] M. Massoudi, S. Verma, and R. Jain, "Urban sound classification using cnn," in *2021 6th International Conference on Inventive Computation Technologies (ICICT)*. IEEE, 2021, pp. 583–589. 3.3

[108] D. Stowell, D. Giannoulis, E. Benetos, M. Lagrange, and M. D. Plumbley, "Detection and classification of acoustic scenes and events," *IEEE Transactions on Multimedia*, vol. 17, no. 10, pp. 1733–1746, 2015. 3.4

[109] Dcase, "Detection and classification of acoustic scenes and events." [Online]. Available: https://dcase.community/ 3.4

[110] A. Mesaros, T. Heittola, A. Diment, B. Elizalde, A. Shah, E. Vincent, B. Raj, and T. Virtanen, "Dcase 2017 challenge setup: Tasks, datasets and baseline system," in *DCASE 2017-Workshop on Detection and Classification of Acoustic Scenes and Events*, 2017. 3.4

[111] S. Okazaki, Q. Kong, and T. Yoshinaga, "Ldslvision submissions to dcase21: A multi-modal fusion approach for audio-visual scene classification enhanced by clip variants," *DCASE2021 Challenge, Tech. Rep, Tech. Rep.*, 2021. 3.4

[112] V. Ghodasara, S. Waldekar, D. Paul, and G. Saha, "Acoustic scene classification using block-based mfcc features," in *Proceedings of the Detection and Classification of Acoustic Scenes and Events 2016 Workshop (DCASE2016)*, 2016. 3.4

[113] Z. Xu, F. Yu, C. Liu, and X. Chen, "Hasp: A high-performance adaptive mobile security enhancement against malicious speech recognition," *arXiv preprint arXiv:1809.01697*, 2018. 3.4

[114] A. Khodista Syaka, N. Akhmad Setiawan, and O. Wahyunggoro, "Comparison on classification of the holy quran verses using mfcc and rqa," in *Proceedings of the 2021 International Conference on Computer, Control, Informatics and Its Applications*, 2021, pp. 49–54. 3.4

[115] A. Benba, A. Jilbab, A. Hammouch, and S. Sandabad, "Voiceprints analysis using mfcc and svm for detecting patients with parkinson's disease," in *2015 International conference on electrical and information technologies (ICEIT)*. IEEE, 2015, pp. 300–304. 3.4

[116] A. Tharwat, "Classification assessment methods," *Applied computing and informatics*, vol. 17, no. 1, pp. 168–192, 2020. 3.5.2

[117] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778. 3.5.3

[118] B. McFee, C. Raffel, D. Liang, D. P. Ellis, M. McVicar, E. Battenberg, and O. Nieto, "librosa: Audio and music signal analysis in python," in *Proceedings of the 14th python in science conference*, vol. 8, 2015, pp. 18–25. 3.6

[119] S. Gupta, J. Jaafar, W. W. Ahmad, and A. Bansal, "Feature extraction using mfcc," *Signal & Image Processing: An International Journal*, vol. 4, no. 4, pp. 101–108, 2013. 3.6

[120] C. Spearman, "The proof and measurement of association between two things." 1961. 3.7

[121] S. P. Singh and M. Jaggi, "Model fusion via optimal transport," *Advances in Neural Information Processing Systems*, vol. 33, pp. 22 045–22 055, 2020. 4.3.3, 4.5, 4.5.2.1

[122] S. U. Amin, G. Muhammad, W. Abdul, M. Bencherif, and M. Alsulaiman, "Multi-cnn feature fusion for efficient eeg classification," in *2020 IEEE International Conference on Multimedia & Expo Workshops (ICMEW)*. IEEE, 2020, pp. 1–6. 4.5

[123] F. Grimberg, M.-A. Hartley, S. P. Karimireddy, and M. Jaggi, "Optimal model averaging: Towards personalized collaborative learning," *arXiv preprint arXiv:2110.12946*, 2021. 4.5, 4.5.2.1

[124] TensorFlow, "Performance measurement: Tensorflowlite." [Online]. Available: https://www.tensorflow.org/lite/performance/measurement 5.2