Compresión y fusión de redes convolucionales para el análisis de imágenes en dispositivos de borde

Fabian Stiven Duque Duque

Tesis de maestría presentada para optar al título de Magíster en Ingeniería

Director
Fredy Alexander Rivera Vélez, Doctor (PhD)
Codirector
Ricardo Andrés Velásquez Vélez, Doctor (PhD)

Universidad de Antioquia
Facultad de Ingeniería
Maestría en Ingeniería
Medellín, Antioquia, Colombia
2024

Cita	Duque Duque[1]
Referencia Estilo IEEE (2020)	[1] F. Duque Duque, “Compresión y fusión de redes convolucionales para el análisis de imágenes en dispositivos de borde”, Tesis de maestría, Maestría en Ingeniería, Universidad de Antioquia, Medellín, Antioquia, Colombia, 2024.

Maestría en Ingeniería, Cohorte XXXVI.

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

UNIVERSIDAD
DE ANTIOQUIA
1 8 0 3

Compresión y fusión de redes convolucionales para el análisis de imágenes en dispositivos de borde

Trabajo de investigación presentado como requisito para optar al título de:
Magíster en Ingeniería

Estudiante:

Fabian Stiven Duque Duque

Asesores:

Fredy Alexander Rivera Vélez, PhD.
Ricardo Andrés Velásquez Vélez, PhD.

Universidad de Antioquia
Facultad de Ingeniería
2023

Resumen

La seguridad ciudadana ha adquirido una relevancia central en la garantía del bienestar de los habitantes de diversas ciudades en todo el mundo. La integración de herramientas tecnológicas, como las cámaras de vídeo, ha demostrado ser efectiva para mitigar situaciones de riesgo a través de estrategias de monitoreo y control. Las imágenes captadas por estas cámaras se han convertido en una fuente invaluable de información para identificar eventos potencialmente peligrosos, generando alertas que son canalizadas a las autoridades correspondientes. En el ámbito del procesamiento de información visual, las redes neuronales profundas son comúnmente empleadas debido a su capacidad precisa para identificar situaciones de riesgo. La computación en el borde, como alternativa a la computación en la nube, emerge como una tendencia destinada a optimizar la eficiencia de los sistemas de monitoreo. La convergencia de modelos de aprendizaje profundo y la computación en el borde plantea desafíos para la ejecución eficiente de modelos en dispositivos con limitaciones en capacidad de cómputo y memoria. Para mejorar la eficiencia en la ejecución de modelos neuronales en el borde se emplea la compresión como estrategia para disminuir el tamaño y la complejidad del modelo. La fusión de modelos de redes neuronales se destaca también como una estrategia que permite la combinación de dos o más modelos en uno solo. Este nuevo modelo fusionado conserva la capacidad de realizar inferencias de los modelos originales, pero con una reducción en los requisitos de cómputo y memoria. El presente trabajo emplea modelos de aprendizaje profundo para identificar, en imágenes de vídeo, objetos que pueden comprometer la seguridad de los ciudadanos. Los modelos individuales, que identifica cada tipo de objeto en particular, son comprimidos y luego fusionados para permitir su ejecución en dispositivos de borde. Esta aproximación es evaluada para garantizar una identificación precisa y oportuna de los objetos y así alertar sobre situaciones de riesgo para las personas. Los resultados obtenidos son alentadores, mostrando que la nuestra tiene un enorme potencial para desplegar modelos de aprendizaje profundo complejos sobre platafomas con limitaciones computacionales y de memoria.

Índice general

Índice de figuras

Índice de tablas

Agradecimientos

Quiero expresar mi profundo agradecimiento a las personas e instituciones que hicieron posible la culminación de este trabajo de investigación.
A mi familia, les agradezco de corazón por su apoyo constante y amor inquebrantable. Siempre estuvieron presentes, brindándome fuerza y ánimo en cada etapa de este proceso. Sus palabras de aliento fueron un faro en los momentos de dificultad.
A Adriana, mi fuente de inspiración y apoyo inquebrantable, te agradezco por tu cariño y aliento. Tus palabras siempre elevaron mi espíritu y me impulsaron a seguir adelante.
Agradezco a los profesores Fredy y Ricardo por su orientación, paciencia y sabiduría durante todo el desarrollo de esta investigación. Sus conocimientos y guía fueron invaluables.
No puedo dejar de mencionar al grupo de investigación Sistemic, que abrió sus puertas y me brindó un espacio para aprender y colaborar en un entorno de conocimiento y creatividad.
Finalmente, mi gratitud se extiende a la Universidad de Antioquia por brindarme la oportunidad de regresar y recorrer sus inigualables espacios. Este trabajo no habría sido posible sin su apoyo y recursos.
A todos ustedes, mi más sincero agradecimiento. Este logro es también suyo, y estoy agradecido por cada paso que dimos juntos en este viaje.

Capítulo 1

Introducción

Se prevé que para el 2045 la población urbana con respecto al 2014 aumentará aproximadamente en 2000 millones de habitantes y para el 2050 se incrementará en un 66 %, con más de 6000 millones de habitantes viviendo en los territorios urbanos [1]. Este crecimiento urbano transformará las condiciones de vida de los habitantes en las ciudades en términos medioambientales, sociales, económicos y de seguridad. La problemática de la seguridad es uno de los grandes retos a nivel nacional e internacional en términos de desarrollo y transformación de los territorios, ya que tiene un papel fundamental en el bienestar de las personas.

A nivel internacional, la Organización de las Naciones Unidas (ONU) en la Agenda 2030, entre sus Objetivos de Desarrollo Sostenible (ODS), incluye el objetivo 16: "Paz, Justicia e Instituciones Sólidas", donde su propósito es "reducir significativamente todas las formas de violencia y las correspondientes tasas de mortalidad en todo el mundo" y "fortalecer las instituciones nacionales pertinentes, incluso mediante la cooperación internacional, para crear a todos los niveles, particularmente en los países en desarrollo, la capacidad de prevenir la violencia y combatir el terrorismo y la delincuencia" [2]. La Organización para la Cooperación y el Desarrollo Económicos (OCDE) establece que la seguridad ciudadana es un factor determinante para el bienestar [3]. Teniendo en cuenta esta perspectiva, el riesgo de que las personas sean víctimas de un asalto u otro tipo de delito, representa un alto impacto para su bienestar, generando sensaciones de vulnerabilidad. A nivel nacional, el Departamento Nacional de Planeación (DNP) a través del Observatorio de Sistemas de Ciudades, establece que la seguridad es crucial en una ciudad moderna, que "ofrece protección efectiva de los derechos de sus ciudadanos a la vida, la propiedad

y la integridad" [4]. La seguridad se puede medir a partir de factores como las tasas de víctimas de secuestro, de lesiones entre personas, de homicidios, y de hurtos comunes por cada 100000 habitantes. En el Valle de Aburrá, dentro del estudio de Seguridad Ciudadana desde la Gobernanza Metropolitana, en el análisis de riesgo del capítulo 5 se enuncia la "necesidad de contar con un sistema metropolitano de gestión de la información, que centralice los datos de criminalidad y contravenciones generados por las instituciones encargadas de la seguridad ciudadana y la administración de justicia, tanto en los municipios como en el Área Metropolitana" [5].

Las ciudades disponen cada vez de más aplicaciones basadas en las Tecnologías de la Información y la Comunicación (TIC), con el ánimo de convertirse en ciudades inteligentes [6]. En este contexto, la computación de borde ha cobrado protagonismo considerando el tipo de aplicaciones que pueden ser útiles en una ciudad inteligente. Este enfoque consiste en migrar algunos recursos de la nube (procesamiento y almacenamiento) a una infraestructura cercana al origen de los datos, para proporcionar servicios inteligentes que puedan satisfacer las necesidades críticas en aplicaciones de tiempo real, optimización de datos, inteligencia de aplicaciones, privacidad de la información y seguridad, que cumplan con los requisitos de baja latencia y ancho de banda en la red. Los dispositivos de borde en una ciudad inteligente se están usado en diferentes áreas, como los hogares inteligentes, los vehículos autónomos, las cámaras de vigilancia, los robots de producción industrial, entre otros [7]. En 2016 existían 17100 millones de dispositivos conectados a internet y en el año 2019 el tráfico de información asociado a estos dispositivos en los centros de datos globales alcanzó los 10,4 Zettabytes (1 ZB = 10^{21} Bytes), donde el 45 % de los datos fueron almacenados, procesados y analizados cerca a la fuente de información [8].

En el entorno urbano, la videovigilancia se ha hecho presente en lugares como centros comerciales, hospitales, edificios gubernamentales y en las calles. En algunos casos estos sistemas de vigilancia se instalan con la función de persuasión ante comportamientos inaceptables, grabando y registrando eventos como material probatorio. Con las TICs, los sistemas de videovigilancia se han hecho inteligentes para ocuparse de la supervisión en tiempo real de objetos fijos y transitorios dentro de un entorno específico. Las principales funciones de estos son ofrecer una interpretación automática de las escenas, comprender y predecir las acciones e interacciones de los objetos observados, basados en la información captada por las cámaras [9]. Con ayuda de la computación

de borde y el aprendizaje profundo se pueden analizar patrones de comportamiento de las personas y generar alertas tempranas que apoyen la seguridad ciudadana. La popularidad del aprendizaje profundo ha crecido en la comunidad académica e industrial, convirtiéndose en una tendencia para el desarrollo de aplicaciones. El aprendizaje profundo simula la estructura jerárquica del cerebro humano, procesando datos de un nivel inferior a un nivel superior [10], utilizando múltiples capas de procesamiento para descubrir patrones y estructuras en grandes conjuntos. Cada capa aprende un concepto de los datos sobre los que se apoyan las capas posteriores. Cuanto más alto es el nivel, más abstractos son los conceptos que se aprenden [11].

1.1. Planteamiento del problema

Actualmente, los modelos de aprendizaje profundo han demostrado su robustez en una amplia gama de aplicaciones, pero a medida que evolucionan y se vuelven más sofisticados, se hace evidente un desafío creciente en términos de complejidad computacional y consumo de recursos. Este fenómeno se torna aún más crítico cuando se intenta implementar estos modelos bajo el paradigma de computación en el borde, donde se presentan restricciones significativas debido al tipo de dispositivos utilizados.

La implementación de sistemas de videovigilancia inteligentes requiere la detección de múltiples eventos y para lograrlo se suelen entrenar diferentes modelos de redes neuronales con conjuntos de datos particulares, de modo que tengan un buen rendimiento en entornos específicos. Sin embargo, en las aplicaciones prácticas de la inteligencia artificial es habitual tratar múltiples tareas simultáneamente, lo que conlleva una gran demanda de recursos computacionales, tanto en la fase de entrenamiento como en la de inferencia [12]. Por esta razón, es necesario realizar una integración compacta y eficaz de redes neuronales pre-entrenadas, que permita su ejecución en dispositivos con hardware limitado. Es aquí donde surgen las técnicas de compresión para reducir el número de parámetros y cálculos de cada capa en las redes neuronales [13] y las técnicas de fusión de dichas redes con el objetivo de producir un modelo compacto que pueda ejecutar las tareas originales simultáneamente [12]. Mediante la aplicación de estas técnicas, el cómputo, almacenamiento y energía necesarios para ejecutar las redes neuronales se reduce. Hasta el momento, según la información consultada en el estado del arte, no se han reportado trabajos que se ocupen de la combinación de estas dos estrategias para reducir el costo computacional cuando se desea detectar múltiples eventos en un dispositivo de borde.

En el contexto de la computación de borde se destacan los computadores de una sola tarjeta (SBC - Single-Board Computer), que se han convertido en una elección popular gracias a su bajo costo y eficiencia energética. Sin embargo, los SBCs imponen limitaciones notables en términos de memoria RAM, velocidad de los núcleos y capacidad de procesamiento, con capacidades que oscilan entre 512MB y 8GB de RAM, velocidades de núcleos que varían de 700MHz a 2GHz y un máximo de 8 núcleos por procesador [14]. El almacenamiento nativo en estos dispositivos es también restringido, aunque se puede expandir mediante memorias SD, módulos M2 y discos externos. Estas restricciones, junto con la necesidad de implementar modelos de aprendizaje profundo, plantean desafíos significativos.

Por consiguiente, en esta tesis de maestría se propone implementar la combinación entre compresión y fusión de redes neuronales pre-entrenadas, que optimicen el uso de los recursos computacionales, para detectar múltiples eventos asociados a la seguridad ciudadana en dispositivos de borde.

1.2. Objetivos

1.2.1. Objetivo general

Explorar el uso simultáneo de técnicas de compresión y fusión de redes neuronales convolucionales que reduzcan el costo computacional y la huella de memoria al momento de implementar sistemas de vídeo vigilancia en dispositivos de borde para la detección de eventos asociados a la seguridad ciudadana.

1.2.2. Objetivos específicos

1. Analizar el desempeño de diferentes redes neuronales convolucionales y seleccionar al menos dos de ellas, que permitan detectar eventos asociados a la seguridad ciudadana, considerando precisión, tiempo de ejecución y recursos computacionales requeridos.

2. Identificar y aplicar técnicas de compresión de redes neuronales que reduzcan el tamaño de la red y optimicen el uso de los recursos computacionales en un dispositivo de borde.

3. Estudiar y seleccionar técnicas de fusión para redes convolucionales que permitan la optimización de recursos de cómputo y memoria, y adicionalmente posibiliten su integración con las técnicas de compresión identificadas.

4. Evaluar la combinación de las técnicas de compresión y fusión utilizando como referencia los modelos originales, empleando métricas de precisión, costo computacional y huella de memoria.

1.3. Estado del arte

En esta sección se analizan diferentes trabajos e ideas encontradas en el estado del arte. Estos trabajos cubren temas relacionados con la videovigilancia inteligente y la compresión y fusión de redes neuronales.

1.3.1. Detección automática de eventos anómalos en un vídeo

La detección de anomalías es un problema muy relevante en los sistemas de vigilancia inteligente. Para este propósito se despliega un gran número de

cámaras en diferentes espacios públicos como parques, estaciones de metro, aeropuertos y calles. Algunas de las situaciones que se ha intentado identificar de manera automática con sistemas basados en inteligencia artificial son [15]:

1. Accidentes de trafico
2. Robos
3. Actividades ilegales
4. Peleas
5. Cruces imprudentes de calles
6. Violencia en multitudes
7. Incendios provocados

Sultani *et al.* [16] proponen un sistema de detección de anomalías a partir de vídeos que presentan situaciones anómalas y otros que son considerados normales. En lugar de etiquetar los segmentos (clips) de los vídeos en los que hay situaciones anómalas, lo cual tomaría mucho tiempo, los autores etiquetan los vídeos mismos como anómalos o normales, y luego los segmentos con anomalías son tratados como instancias en un entorno de aprendizaje de múltiples instancias que se encarga de predecir grandes valores de anomalía para los segmentos anómalos en un vídeo, utilizando vídeos de entrenamiento débilmente etiquetados. Es decir, sólo se conoce las etiquetas a nivel de vídeo, donde esta describe si el vídeo contiene o no una anomalía en alguna parte, pero no sabe en qué parte del vídeo se encuentra la anomalía. Por otro lado, Landi *et al.* [17] proponen aprovechar la localización inherente de las anomalías y utilizar la información espacio-temporal realizando recortes y redimensionamiento de los fotogramas de entrada para identificar si hay cambios en las coordenadas un evento u objeto de un fotograma a otro. En contraste, Xu *et al.* [18] localizan de forma automática todas las zonas de interés en las que podrían producirse acciones de pelea, extrayendo varias zonas de activación a partir de un mapa de activación del movimiento que mide la actividad de cada posición. Por otra parte, Hassner *et al.* [19] proponen un método para la detección de comportamientos violentos en las multitudes, donde tienen en cuenta estadísticas de cómo cambia el flujo de las personas en el tiempo, para luego clasificar si existe un acto violento o no apoyándose de las máquinas de vectores de soporte lineales (SVM). De manera similar Datta *et al.* [20] identifican los actos de

violencia entre personas en vídeos, basándose en información de trayectoria de movimiento e información relacionada a las extremidades de una persona.

Estos trabajos representan un primer acercamiento a la detección de eventos anómalos, sin embargo, su limitación principal es que se desarrollaron en el contexto de la computación en la nube.

Detección de armas en imágenes y vídeos

En esta sección se exploran diferentes estudios que han explorado diferentes arquitecturas de redes neuronales (del ingles Neural Networks, NNs) avanzadas y metodologías innovadoras sobre los retos que plantea el reconocimiento de armas en imágenes y vídeos.

Olmos *et al.* [21] propusieron un método basado en redes neuronales convolucionales (del ingles Convolutional Neural Network, CNN) para identificar armas de fuego. En su formulación, dividieron este desafío en dos clases: arma y fondo. Al etiquetar manualmente fotografías tomadas de Internet y utilizando un modelo de detección construido a partir de la Faster Region-based CNN (R-CNN) VGG-16 [22], generaron una base de datos de entrenamiento. Como resultado, cuando se entrenó en el conjunto de datos de 6000 imágenes, este modelo logró una precisión del 91.43 %. Sin embargo, es importante destacar que el conjunto de datos utilizado en este estudio puede tener limitaciones; en particular, las imágenes que representan armas de fuego no reflejan con precisión situaciones de la vida real. En la misma línea, Reddy *et al.* [23] crearon y compararon dos modelos para un detector de armas de fuego. Específicamente, con el propósito de detectar armas ocultas, emplearon un modelo Multi-Channel CNN (MC-CNN) y un modelo Faster R-CNN. A través de su investigación, descubrieron que el modelo Faster R-CNN superó al MC-CNN debido a su arquitectura más compleja. Además, Verma *et al.* [24] desarrollaron un sistema de reconocimiento automático de armas basado en CNN para escenarios congestionados. Utilizaron un modelo más avanzado de faster R-CNN, incorporando transferencia de aprendizaje (Transfer Learning). Para llevar a cabo sus estudios utilizaron un conjunto de datos de una sección de la Internet Movie Firearm Database (IMFDb) [25]. Para reconocer y clasificar tres tipos diferentes de armas: escopetas, revólveres y pistolas, emplearon SVM y lograron una precisión del 89.9 %. IMFDb es una base de datos completa de armas de fuego utilizadas en películas. Sin embargo, las imágenes de armas de fuego disponibles en esta base de datos no representan situaciones del mundo real. En su lugar, generalmente se presentan en un contexto idealizado con un fondo blanco y el arma posicionada centralmente en la imagen.

En un enfoque diferente, Kaya *et al.* [26] introdujeron un nuevo modelo basado en la arquitectura VGG, entrenado en un conjunto de datos con siete clases distintas de armas. Estas clases incluyeron rifles de asalto, bazucas, granadas, rifles de caza, cuchillos, pistolas y revólveres. Para comparar y evaluar los resultados de sus modelos, compararon los mejores resultados de clasificación con los de VGG-16, ResNet-101 [27] y ResNet-50 [28] en la nube. Su modelo sugerido superó notablemente a ResNet-50 con una precisión del 93.7 %, VGG-16 con 89.75 % y ResNet-101 con 83.33 %. Sin embargo, es importante mencionar que la limitación del estudio fue el uso principalmente de imágenes no reales en el conjunto de datos, lo que limitó la generalización del modelo en escenarios en tiempo real. Ahmed *et al.* [29] introdujeron un sistema de detección de armas utilizando Scaled-YOLOv4. Para mejorar el rendimiento del modelo utilizaron TensorRT con un énfasis específico en la cuantización. La aplicación de este sistema se ejecutó en una placa Jetson Nano. A pesar de que lograron mejorar la eficiencia, esto llevó a una pequeña reducción en la puntuación de precisión promedio. De manera similar, Berardini *et al.* [30] propusieron una estrategia innovadora que emplea un detector de personas para centrar la búsqueda de armas en individuos. Este enfoque, que implica la detección de personas seguida de la detección de armas, se implementó en la plataforma NVIDIA Jetson Nano Developer. El método logró una precisión promedio del 79.30 %, destacando la efectividad de la integración de la detección de personas como un paso preliminar en la detección de armas. En esta instancia resulta en una mayor precisión y una utilización más eficiente de los recursos computacionales. Sin embargo, es importante tener en cuenta que la eficacia de esta estrategia podría estar limitada en escenarios donde fluye un gran número de personas, lo que podría plantear desafíos al intentar detectar armas en entornos concurridos.

1.3.2. Compresión de redes neuronales

En esta sección, exploraremos una selección de trabajos relevantes que se centran en la compresión de NNs, una disciplina esencial en el campo del aprendizaje profundo. La creciente complejidad y el tamaño de las NNs han impulsado la necesidad de investigar métodos y técnicas que permitan reducir la carga computacional y el consumo de recursos, sin comprometer significativamente el rendimiento. La compresión de NNs se ha convertido en una vía fundamental para abordar este desafío.

Liu *et al.* [31] se inspiraron en la optimización de la tasa de distorsión en la codificación de imágenes y vídeo, usando un método de optimización del rendimiento computacional (CPO) para eliminar los filtros convolucionales redundantes en una CNN con restricciones de rendimiento. Para demostrar la efectividad del método, aplicaron la misma caída en la relación señal-ruido pico y la precisión para la evaluación del rendimiento de la tarea. Realizaron experimentos con VGG-19, Resnet-32 y Mobilenet-22 en los que se pudo eliminar un gran número de parámetros y computación de punto flotante sin una caída significativa en la precisión. Por otro lado, Gope *et al.* [32] proponen una arquitectura de red híbrida para una aplicación de localización de palabras clave capaz de ofrecer niveles de precisión de última generación, al tiempo que requiere una fracción de los parámetros del modelo y un número considerablemente menor de operaciones para la inferencia. Además, utiliza StrassenNets [33] para reducir el tamaño del modelo. La arquitectura híbrida se hace posible aprovechando unas pocas capas de las redes neuronales para extraer características de entrada y alimentando un árbol de decisión poco profundo para realizar la clasificación. Utilizando este modelo híbrido logran reducir un 11.1 % el número de cálculos, un 52.2 % el tamaño del modelo y un 30.6 % la huella de memoria global con respecto a una NN. Por el contrario, Shi *et al.* [34] mencionan que su principal propósito es acelerar la inferencia multi-modelo en dispositivos heterogéneos en procesadores multi-core. Su objetivo es desplegar diversos modelos de detección de un solo objeto en lugar de un modelo pesado de detección de varias clases, debido a que en la mayoría de los casos solo se necesita detectar uno o dos objetos en un escenario. Sus resultados muestran que el rendimiento paralelo de la inferencia multi-modelo es notable, mientras que su flexibilidad y la mejora de la funcionalidad son mucho mejores en comparación con el método original de inferencia de un modelo. Por último, Wu *et al.* [35] hacen referencia a que las diferentes capas de una NN pueden tener diferentes requisitos de poda, por lo cual propone un método de poda de pesos evolutivo diferencial por capas. En primer lugar, analizan la sensibilidad a la poda de cada capa y, posteriormente, comprimen la red mediante la iteración del proceso de poda de pesos. La evolución diferencial es un método eficaz basado en la optimización de poblaciones que puede utilizarse para abordar esta tarea. Además, adoptan una estrategia para recuperar algunas de las conexiones eliminadas para aumentar la capacidad del modelo podado durante la fase de ajuste fino. La eficacia de su método se ha probado en estudios experimentales, logrando comprimir el número de parámetros de peso en LeNet, AlexNet y VGG-16 en 24×, 29× y 12×, respectivamente.

1.3.3. Fusión de redes neuronales

En esta sección presentamos trabajos relacionados con la fusión de redes neuronales. La fusión es una técnica en la que varios modelos se combinan para compartir una parte de la arquitectura del modelo final implementado, es decir, una porción de los parámetros; de ahí el término "fusión".

Choi *et al.* [36] proponen una arquitectura de fusión multimodal basada en aprendizaje profundo para tareas de clasificación, que garantiza la compatibilidad con cualquier tipo de modelo de aprendizaje, tratando la información intermodal y evitando la degradación del rendimiento debido a la ausencia parcial de datos. Utilizaron dos conjuntos de datos para tareas de clasificación multimodal, construyeron modelos basados en su arquitectura y en otras configuraciones de red, y analizaron su rendimiento en varios escenarios. Los resultados muestran que la arquitectura supera a otras de fusión multimodal cuando algunas partes de los datos no están disponibles. Por otro lado, Rachmadi *et al.* [37] analizan el rendimiento de fusión de CNNs aplicando fusión temprana y tardía para verificación de parentesco. Las ventajas de la configuración de fusión temprana son que no hay grandes cambios en la arquitectura del clasificador y que sólo la primera capa se tiene un tamaño de filtro diferente. La configuración de fusión tardía se forma creando una red CNN doble para extraer las características profundas de cada imagen y clasificar la relación de parentesco utilizando dos capas totalmente conectadas. De manera similar, Williams *et al.* [38] proponen una fusión de características de nivel intermedio, uniendo los pesos para cada modalidad con entradas de audio, vídeo y texto durante el entrenamiento e incluyendo un entrenamiento adicional posterior en el análisis de sentimientos unimodal. Obtienen resultados de precisión del 72.4 % para nivel temprano, 74 % para el intermedio y 72.5 % para el tardío, comparadas con la clasificación unimodal. Por último, Mallouh *et al.* [39] proponen un sistema denominado AGender-Tuning DNN para clasificar la edad y el género de los hablantes mediante la combinación de dos arquitecturas DNN (Deep Neural Network): Age-DNN para clasificar cuatro grupos de edad, y Gender-DNN para clasificar el género. Proponen una tercera capa de salida para combinar las capas de salida de las Age y Gender DNNs utilizando la suma de características a nivel intermedio, obteniendo como resultado, en términos de precisión, un 55.16 % para el modelo fusionado comparado con los modelos originales que obtienen un 47.89 % y 43.8 %, respectivamente. Chou *et al.* [40] proponen un método llamado NeuralMerger con el fin de fusionar redes neuronales para producir un modelo compacto que ejecute las tareas originales simultáneamente. El método consiste en alinear las capas de la red y

codificar los pesos representativos de las redes, conservado las capas que no son similares entre las redes.

En la **Tabla 1.1** se presenta un resumen de los trabajos citados previamente que tienen un enfoque referido a la detección de eventos de seguridad, compresión y fusión de redes neuronales.

Tabla 1.1: Resumen de trabajos relacionados

Referencia	Detección de eventos de seguridad	Compresión	Fusión
Olmos *et al.* [21]	✓	×	×
Reddy *et al.* [23]	✓	×	×
Verma *et al.* [24]	✓	×	×
Kaya *et al.* [26]	✓	×	×
Ahmed *et al.* [29]	✓	×	×
Berardini *et al.* [30]	✓	×	×
Liu *et al.* [31]	×	✓	×
Gope *et al.* [32]	×	✓	×
Shi *et al.* [34]	×	✓	×
Wu *et al.* [35]	×	✓	×
Choi *et al.* [36]	×	×	✓
Rachmadi *et al.* [37]	×	×	✓
Williams *et al.* [38]	×	×	✓
Mallouh *et al.* [39]	×	×	✓
Chou *et al.* [40]	×	×	✓

Como se puede observar en la **Tabla 1.1**, a la fecha del 31 de octubre del 2023, en el estado del arte no existe un trabajo que pretenda detectar eventos asociados a la seguridad ciudadana empleando aprendizaje profundo que recurra a la compresión y fusión de modelos para posibilitar su ejecución en dispositivos con recursos restringidos. Nuestro enfoque es novedoso por su integralidad, y ofrece una visión más amplia que la que hasta ahora hemos podido identificar.

1.4. Contribución del trabajo de investigación

En este trabajo de maestría se implementaron y evaluaron diversas estrategias de compresión y fusión de redes neuronales convolucionales. Nuestro principal enfoque consistió en la creación de un modelo combinado a partir de dos modelos individuales, logrando, en algunos casos, igualar o incluso mejorar el desempeño en cuanto a la métrica de precisión (accuracy) promedio en

el reconocimiento de objetos en comparación con las obtenidas por las tareas individuales. Este resultado fue logrado reduciendo el tamaño del modelo y la cantidad de memoria necesaria para su ejecución. Un aspecto interesante es que a su vez el modelo combinado exhibió una reducción significativa significativa el tiempo de ejecución cuando se le compara con la ejecución independiente de los modelos individuales. Estos aspectos demuestran la factibilidad de nuestra aproximación para la ejecución de modelos de aprendizaje profundo bajo el paradigma de la computación en el borde.

Con los resultados de este proyecto se presentó un artículo titulado "Optimizing Convolutional Neural Networks for Efficient Weapon Detection on Edge Devices" a IEEE ChileCon 2023, el cual fue aceptado para su presentación y publicación. IEEE ChileCon es una conferencia internacional que cuenta con Comité Científico en la que los artículos sometidos pasan por un proceso de tres revisiones anónimas. La versión 2023 se llevó a cabo en Valdivia, Región de los Ríos (Chile) del 5 al 7 de diciembre de 2023.

1.5. Estructura del trabajo de investigación

La estructura de este trabajo se organiza en capítulos, cada uno abordando aspectos fundamentales del proyecto. En el capítulo 2, se presentan los fundamentos teóricos con un enfoque en las ciudades inteligentes, la inteligencia artificial, los paradigmas de computación y la compresión y fusión de redes neuronales. El capítulo 3 presenta el derrotero metodológico empleado en el desarrollo del proyecto, incluyendo una descripción del entorno experimental hardware/software que fue empleado para la evaluación de las propuestas. En el capítulo 4 se describe el proceso de identificación, evaluación y selección de las redes convolucionales útiles para la detección de eventos asociados con la seguridad de personas. En este mismo capítulo se incluye la creación de una base de datos necesaria para llevar a cabo el proyecto debido a que no existía una que ajustara a sus requerimientos. Los capítulos 5 y 6 están enfocados en la implementación y evaluación de técnicas de compresión y fusión, respectivamente, de redes neuronales. Por su parte el capítulo 7 presenta la implementación y evaluación de la combinación de la compresión y la fusión de redes neuronales. Finalmente, en el capítulo 8, se presentan las conclusiones de la investigación y se esbozan posibles direcciones para el trabajo futuro, cerrando así el análisis exhaustivo de este proyecto.

Capítulo 2

Marco teórico

En capítulo, se introducen los conceptos de ciudades inteligentes y seguridad ciudadana, junto con los sistemas de vigilancia inteligente. A continuación se exploran los principios fundamentales de la inteligencia artificial (IA), haciendo especial hincapié en las técnicas relacionadas con el aprendizaje profundo, así como la compresión y fusión de redes neuronales, que son aspectos de interés en el contexto de este proyecto.

2.1. Ciudades inteligentes

En los últimos años, el término "ciudad inteligente" o "smart city" ha ganado considerable atención tanto en el ámbito académico como en el industrial. Este enfoque se ha centrado en las disciplinas de la informática y la ingeniería, con el propósito de aplicar las TICs en entornos urbanos. Aunque no existe una definición universalmente aceptada para "ciudades inteligentes", existe un consenso general en cuanto a la importancia de aprovechar las TICs para transformar una ciudad en una entidad inteligente [41]. Esto abre oportunidades para brindar soluciones eficaces a los desafíos que plantea el crecimiento urbano, abordando necesidades económicas, sociales, ambientales y de seguridad en las áreas urbanas.

La construcción de una ciudad inteligente va más allá de la implementación de infraestructuras tecnológicas basadas en hardware y software avanzados. También implica la participación activa de individuos con creatividad, diversidad y educación, así como instituciones que fomenten una gobernanza inteligente y políticas adecuadas. Estos elementos se reflejan en un modelo conceptual [42], que se asemeja al que se muestra en la **Figura 2.1**.

La tecnología es fundamental para construir una ciudad inteligente. Song

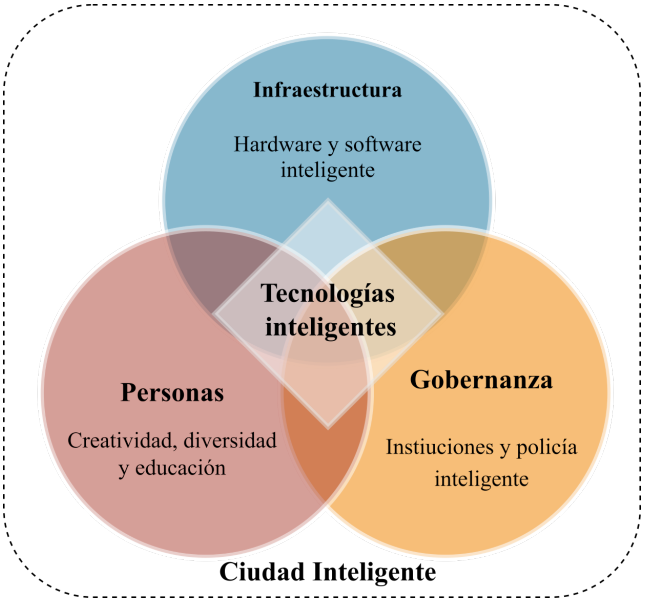


Figura 2.1: Modelo conceptual de ciudad inteligente

et al. [43] nos dicen que el uso de las TIC en servicios de infraestructura como la administración municipal, la educación, la sanidad, la seguridad pública, el sector inmobiliario, el transporte y los servicios públicos es esencial para que estén interconectados y sean más eficientes. De manera similar, Deloitte US [44] afirma que una ciudad inteligente estará impulsada hacía el éxito de la innovación, teniendo en cuenta seis aspectos importantes como la economía, el medio ambiente y la energía, el gobierno y educación, vida y salud, movilidad y, por último, seguridad y protección. Sin embargo, Deakin [45] establece que es necesaria una mayor interacción de la comunidad para lograr una ciudad inteligente, donde la ciudad inteligente no solo se limita a incorporar la tecnología de las TICs, sino que también logra desarrollar tecnología para generar impactos positivos para la comunidad de los territorios.

En resumen, una ciudad inteligente abarca dimensiones como el ambiente, la educación, la movilidad, la seguridad, entre otras, que son cruciales para ofrecer beneficios y mejores condiciones de vida. Desarrollando tecnologías innovadoras se pueden satisfacer las necesidades de los ciudadanos. En el caso de la seguridad pública, esta es un factor muy importante para las ciudades porque genera un impacto positivo en el bienestar, disminuyendo las sensaciones de vulnerabilidad de las personas.

2.2. Seguridad ciudadana y vigilancia inteligente

La seguridad ciudadana se encarga principalmente de mantener los entornos de la sociedad estables y seguros. El principal objetivo es proteger a las personas, el medio ambiente y los bienes públicos y privados, y evitar actos que atenten contra ellos. Con el uso de las TICs se ha logrado una mejor gestión de situaciones de emergencia y catástrofes [46]. Por ejemplo, con la recolección oportuna de información de datos de voz e imágenes se han reducido los tiempos de respuesta de las entidades encargadas durante una emergencia. Para garantizar la seguridad en las ciudades se han incorporado cámaras para monitorear en tiempo real las actividades en un área objetivo y producir grabaciones de vídeo, que se pueden usar para la gestión de tráfico, respuesta a una emergencia y seguridad [42]. Actualmente, las cámaras inteligentes tienen capacidades integradas de procesamiento y comunicación además de la función tradicional de captura de imágenes [43]. Estas cámaras tienen la habilidad de analizar directamente los datos capturados para aplicaciones como la supervisión de las actividades humanas, movimientos de vehículos, congestiones de tráfico, etc. Por tanto, en lugar de limitarse a transmitir imágenes en bruto, las cámaras inteligentes se pueden configurar para que transmitan sólo los datos útiles o señales de control, con lo que se considera la privacidad de la información, el ancho de banda de los datos y otros aspectos anexos. Zhu *et al.* [47] mencionan que la detección automática de anomalías a través del análisis de visión por computador juega un papel fundamental que no solo aumenta significativamente la eficiencia del monitoreo, sino que también reduce la carga de la vigilancia en directo por personas especializadas. Las anomalías en los vídeos se definen ampliamente como eventos o actividades que son inusuales y significan un comportamiento irregular. Las cámaras inteligentes distribuidas son sistemas embebidos que implementan la visión por computador utilizando múltiples cámaras. Este enfoque ha surgido gracias a los avances simultáneos en tres disciplinas clave: los sensores de imagen, los sistemas embebidos y las redes de sensores [48]. En el mercado, se encuentran disponibles múltiples dispositivos con capacidades integradas de análisis de imágenes, tales como detección y seguimiento de movimiento y rostros. Entre estos dispositivos se incluyen modelos con funciones de giro, inclinación y zoom (PTZ), que permiten un control direccional y de zoom, tanto manual como automático. Estas características permiten enfocarse en detalles para un análisis preciso, así como proporcionar una cobertura más amplia en comparación con los dispositivos de visión fija tradicionales [49], [50]. Además, las cámaras inteligentes pueden

ser gestionadas de forma remota a través de Internet, lo que facilita configuraciones y actualizaciones de software para agregar o mejorar características, siendo esto frecuentemente necesario en los sistemas de videovigilancia de una ciudad inteligente [51], [52].

En conclusión, los sistemas de videovigilancia inteligente son una herramienta que facilita tener entornos seguros en las ciudades, ya que no solo se limitan a enviar la información cruda, sino que también pueden enviar la información precisa de alguna anomalía detectada y generar alertas tempranas a los entes encargados para tomar acciones frente a estas situaciones.

2.3. Inteligencia artificial

La IA es una área de la computación que se encarga de desarrollar máquinas inteligentes que pueden llevar a cabo tareas que normalmente requerirían habilidad humana. La base de la IA es la idea de que las máquinas pueden imitar y aplicar el razonamiento, el aprendizaje y la toma de decisiones humanas para resolver problemas complejos [53].
El objetivo principal de la IA es crear sistemas capaces de llevar a cabo tareas con precisión, eficiencia y autonomía que normalmente requerirían intervención humana. A medida que la IA continúa desarrollándose, se están investigando nuevas aplicaciones en diversos campos, que incluyen medicina, industria, seguridad, automatización de procesos y asistencia personal, entre otros.
Dentro de la IA, el aprendizaje de máquinas (del ingles Machine Learning, ML) emerge como una rama fundamental. El ML se basa en la capacidad de las máquinas para aprender y mejorar a partir de la experiencia, sin necesidad de una programación explícita. Dentro del ML, el aprendizaje profundo (del ingles Deep Learning, DL) representa un avance significativo, utilizando redes neuronales profundas para modelar datos complejos y aprender representaciones jerárquicas. En resumen, el DL es una subcategoría del ML, y ambos son componentes esenciales de la IA como se muestra en la **Figura 2.2**, permitiendo a las máquinas tomar decisiones y realizar tareas complejas de manera autónoma y cada vez más precisa.

2.4. Machine learning

El ML es una rama fundamental del la IA que se centra en el estudio de algoritmos y modelos estadísticos que permiten a los sistemas informáticos. Implica analizar y aprender patrones en los datos y luego utilizar esos patro-

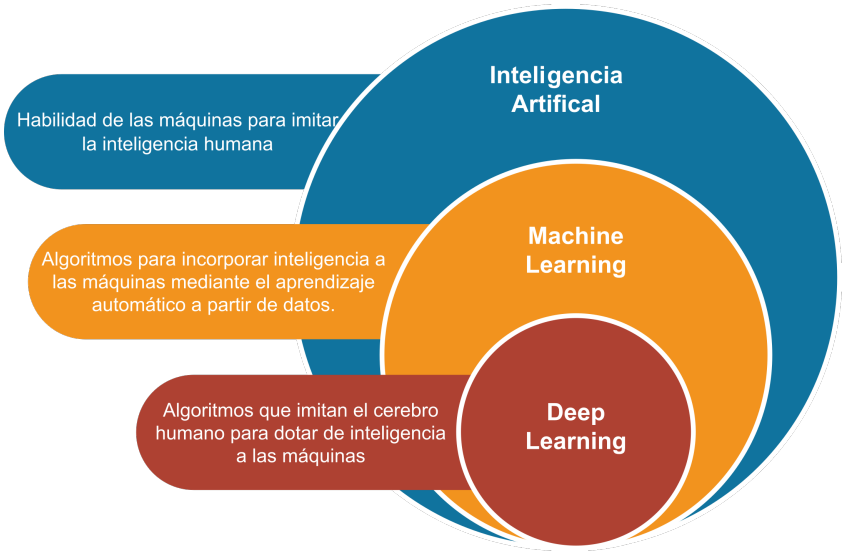


Figura 2.2: Campos interconectados: IA, ML y DL - La base de los sistemas inteligentes modernos

nes para hacer predicciones o tomar decisiones sobre nuevos datos [54].
Durante la fase de entrenamiento de un modelo de aprendizaje automático, se analizan datos y se realizan múltiples iteraciones para minimizar el error y mejorar la precisión de las predicciones. Aquí están los conceptos clave relacionados con el tipo de entrenamiento en ML:

- **Aprendizaje supervisado (Supervised Learning):** En este método, se entrena el modelo utilizando un conjunto de datos etiquetado. Las etiquetas funcionan como respuestas conocidas, y el modelo aprende a hacer predicciones que se asemejen a esas etiquetas. Es una técnica de aprendizaje automático ampliamente utilizada en herramientas con las que interactuamos a diario, como detectores de correo basura, detectores de imágenes en captchas y otras aplicaciones. En la **Figura 2.3**, se representa el proceso de aprendizaje supervisado. Los datos de entrada, que pueden ser imágenes o cualquier otro tipo de información, están debidamente etiquetados, lo que significa que cada entrada tiene una correspondiente etiqueta que indica la salida deseada. Estos datos etiquetados conforman el conjunto de entrenamiento.

  Los datos de entrada y sus etiquetas asociadas se introducen en un algoritmo de aprendizaje automático, que podría ser un modelo de regresión lineal, una NN u otra técnica. Durante el entrenamiento, el algoritmo

ajusta sus parámetros para aprender patrones y relaciones entre las entradas y las etiquetas, con el objetivo de realizar predicciones precisas.

Una vez que el modelo ha sido entrenado con éxito, se convierte en un modelo predictivo. Este modelo puede recibir nuevos datos, es decir, datos no vistos durante el entrenamiento, y tratar de identificar a qué etiqueta pertenecen en función de lo que ha aprendido anteriormente.

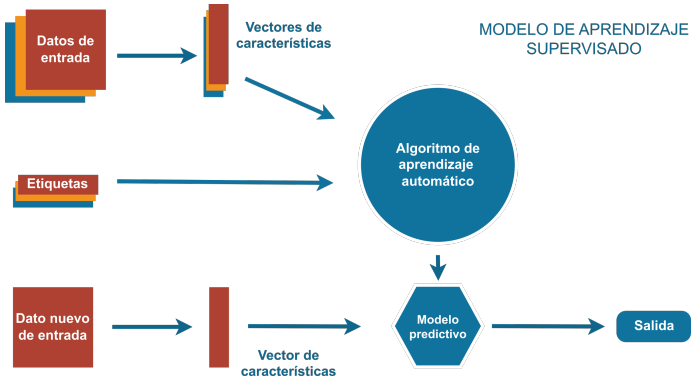


Figura 2.3: Diagrama de flujo del aprendizaje supervisado

- **Aprendizaje no supervisado (Unsupervised Learning):** Por otro lado, el aprendizaje no supervisado se lleva a cabo sin etiquetas. El modelo agrupa o segmenta automáticamente los datos buscando patrones o estructuras. En la **Figura 2.4**, se ilustra el proceso de arendizaje no supervisado. A diferencia del aprendizaje supervisado, en este escenario, los datos de entrada no están etiquetados. Este conjunto de datos no supervisado se utiliza para explorar patrones y estructuras inherentes sin tener etiquetas predefinidas.

  Los datos no etiquetados se introducen en un algoritmo de aprendizaje no supervisado, como puede ser un algoritmo de clustering (agrupamiento) o reducción de dimensionalidad. Durante el proceso de entrenamiento, el algoritmo busca identificar patrones subyacentes o relaciones intrínsecas entre las instancias de datos sin depender de etiquetas externas.

  Una vez completado el entrenamiento, el modelo resultante puede revelar estructuras ocultas o agrupar datos similares sin la necesidad de información de etiquetas. Este proceso puede ser fundamental para explorar la naturaleza de los datos y descubrir patrones emergentes.

- **Aprendizaje reforzado (Reinforcement Learning):** Este método permite al modelo adquirir conocimiento interactuando con su ambien-

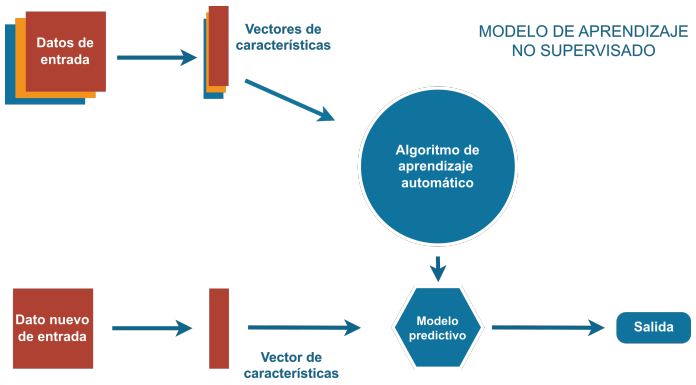


Figura 2.4: Diagrama de flujo del aprendizaje no supervisado

te, aprendiendo a través de la experiencia y la retroalimentación en forma de recompensas, donde las acciones correctas generan valores positivos y las acciones incorrectas generan valores negativos. Para visualizar claramente este proceso, la **Figura 2.5** presenta un diagrama ilustrativo del aprendizaje por refuerzo. Donde los componentes principales son

- **Agente:** Es la entidad que toma decisiones y realiza acciones en el entorno. El agente aprende a través de la retroalimentación que recibe del entorno en forma de recompensas o penalizaciones.
- **Ambiente o entorno:** Es el contexto en el cual el agente opera. Puede ser real o simulado y proporciona las condiciones en las que el agente toma decisiones y experimenta consecuencias.
- **Acciones:** Son las decisiones que el agente puede tomar en un momento dado. La variedad
- **Recompensa:** Es una señal numérica que el entorno proporciona al agente como retroalimentación sobre la calidad de sus acciones. El objetivo del agente es aprender a tomar decisiones que maximicen la recompensa acumulada a lo largo del tiempo.

2.5. Aprendizaje profundo

En esta sección, exploraremos el aprendizaje profundo, una subdisciplina del ML que ha revolucionado la IA en las últimas décadas. El aprendizaje profundo es una rama del ML que se inspira en la estructura y el funcionamiento

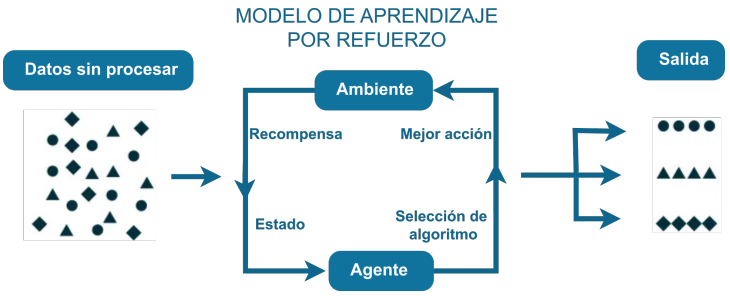


Figura 2.5: Diagrama de flujo aprendizaje por refuerzo

del cerebro humano para procesar información y tomar decisiones, emulando el funcionamiento de las NNs biológicas. A diferencia de los modelos de ML tradicionales, que utilizan características seleccionadas manualmente, el DL permite que el modelo aprenda representaciones de datos de manera automática, lo que lo hace especialmente poderoso en tareas complejas como el procesamiento de imágenes, el procesamiento de lenguaje natural y la visión por computadora [55]. El DL consta de un gran número de unidades llamadas neuronas artificiales, que se conectan para transmitir información desde la capa de entrada y fluye por toda la red neuronal. El flujo de información dentro de la red se somete a distintas operaciones no lineales hasta obtener los valores de salida en la última capa [56].

La **Ecuación 2.1** y la **Figura 2.6** resumen el proceso en el cual las arquitecturas de redes neuronales generalmente consisten en una combinación de módulos simples, conocidos como capas de abstracción. Cada capa se compone de neuronas que reciben un conjunto de valores en la entrada. Donde el vector $\boldsymbol{x}$ contiene las características del conjunto de entrenamiento, y cada neurona tiene su propio conjunto de parámetros denominados $\boldsymbol{w}$ (vector de pesos) y b (sesgo) que cambian durante el proceso de aprendizaje. Por último, el resultado de este cálculo se hace pasar por una función de activación no lineal denominada ϕ.

$$y = \phi(w_1x_1 + w_2x_2 + w_3x_3 + \ldots + w_nx_n + b) = \phi(\boldsymbol{w}^\top \boldsymbol{x} + b) \tag{2.1}$$

En una NN, cada capa transforma la entrada y aumenta el nivel de abstracción del modelo, lo que ayuda a reducir posibles sesgos de la salida debido a factores externos.

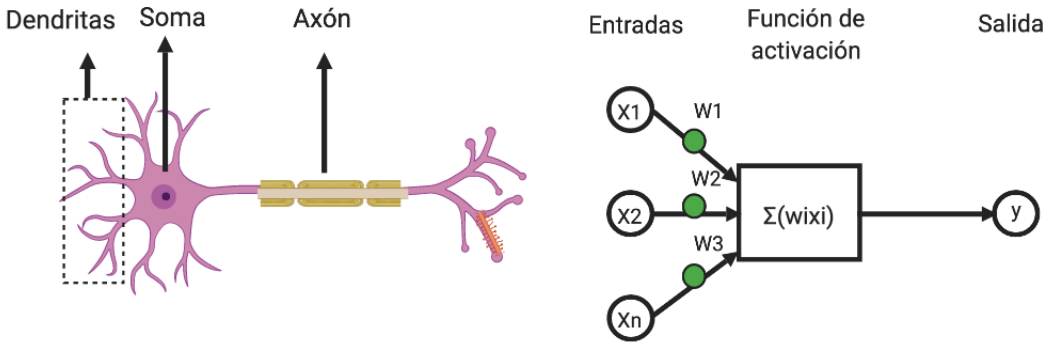


Figura 2.6: Neurona biológica vs. Neurona artificial[1]

2.5.1. Redes neuronales profundas

Las DNNs están compuestas por dos o más capas ocultas con una variedad de características lineales y no lineales. Las funciones lineales son el producto de los pesos con el valor de entrada más el sesgo: $f(\boldsymbol{x}) = \boldsymbol{w}^\top \boldsymbol{x} + b$, mientras que $\phi(\boldsymbol{x})$ son las funciones no lineales (funciones de activación), como sigmoide, tanh, softmax, entre otras. Las activaciones no lineales permiten al sistema "aprender" operaciones o funciones más complejas [57]. La salida de una DNN puede expresarse según la **Ecuación 2.2**, donde j es el número de capas que indica la profundidad del modelo, y o representa la capa de salida. El esquema general de una DNN con j capas de abstracción se muestra en la **Figura 2.7**. Cuando se desarrolla una DNN extremadamente compleja es común que se produzca un sobreajuste, es decir, que el algoritmo se adapta completamente a los datos de entrenamiento y pierde la capacidad de generalizar al problema real.

$$y = \phi_o(\phi_j(f_j(\ldots(\phi_2(f_2(\phi_1(f_1(\boldsymbol{x}))))))))) \tag{2.2}$$

2.5.2. Redes neuronales convolucionales

Las CNNs integran el algoritmo de clasificación de patrones y las etapas de extracción y selección de características en una sola arquitectura [58]. Una CNN esta diseñada específicamente para procesar datos compuestos por múltiples matrices, como las imágenes en color que constan de tres canales RGB (Rojo-Verde-Azul). Estas redes neuronales se componen de tres tipos de capas

[1]Imagen tomada de: `https://futurelab.mx/images/blog/1neuralBneuralC.png`

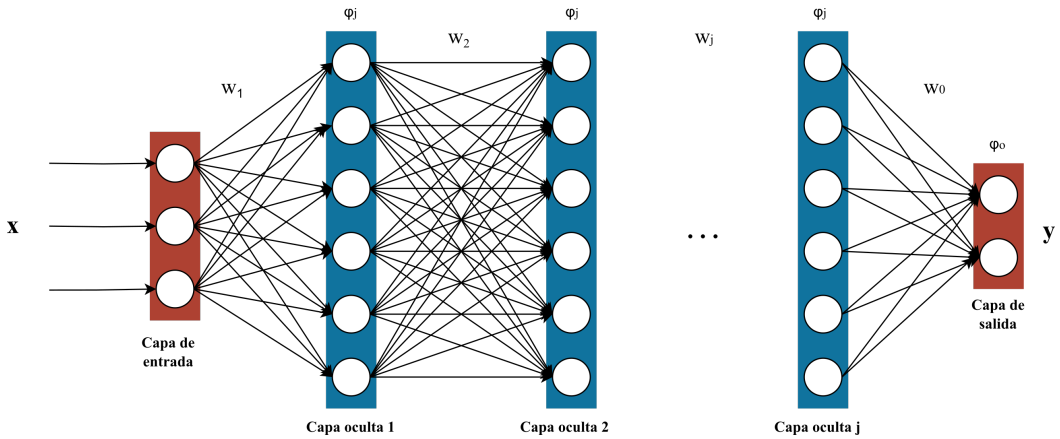


Figura 2.7: Esquema general de una DNN

o componentes principales: capas de convolución, capas de agrupación (también conocidas como pooling) y capas totalmente conectadas. Las capas de convolución y agrupación se encargan de extraer características fundamentales de los datos, mientras que las capas totalmente conectadas realizan la tarea de mapear estas características extraídas hacia la salida final, que generalmente se utiliza para la clasificación. Este proceso se puede apreciar de manera visual en la **Figura 2.8**.

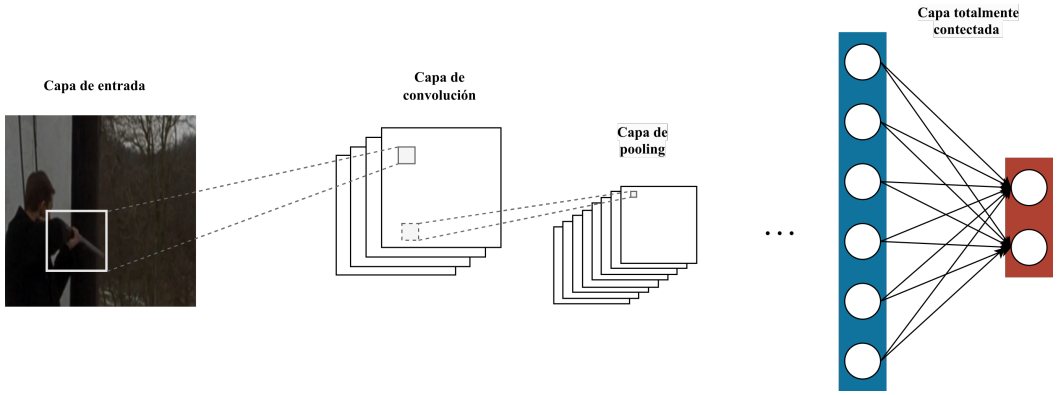


Figura 2.8: Estructura tipica de una CNN

Capa convolucional

La entrada de una CNN puede ser una matriz o un tensor $\boldsymbol{X} \in \mathbb{R}^{v \times h \times c}$, donde v, h y c representan, por ejemplo, la cantidad de canales verticales y

horizontales, así como el valor de una imagen en formato RGB. Un tensor de pesos $\boldsymbol{W} \in \mathbb{R}^{m \times m \times d}$, también conocido como kernel, se le aplica a la entrada de cada capa convolucional de acuerdo con la **Ecuación 2.3**, lo que resulta en una representación oculta $\boldsymbol{H} \in \mathbb{R}^{(v-m+1)\times(h-m+1)\times d}$ de las características extraídas, donde m es el orden del filtro convolucional y d es el número de unidades ocultas en la capa, conocidas como mapas de características.

$$\boldsymbol{H}(i,j,d) = \text{conv}(\boldsymbol{X}, \boldsymbol{W}_d)(i,j) = \sum_{l=1}^{m}\sum_{n=1}^{m} \boldsymbol{X}(i+l, j+n)\boldsymbol{W}_d(l,n) \tag{2.3}$$

Capa de agrupamiento (Pooling)

Esta capa generalmente se encuentra después de la capa convolucional. Su principal utilidad es reducir las dimensiones espaciales de la capa de entrada a partir de un resumen estadístico de las salidas más cercanas en la capa. La operación realizada por esta capa también se llama submuestreo, ya que la reducción de tamaño resulta en pérdida de información. Sin embargo, este tipo de pérdida puede ser beneficiosa para la red por dos razones: (1) la disminución del tamaño conduce a un menor costo computacional, (2) reduce el sobreajuste. Una de las operaciones más comunes en esta capa se llama Max pooling (ver **Figura 2.9**), un método que informa el valor máximo de salida de sus vecinos más cercanos en una matriz rectangular.

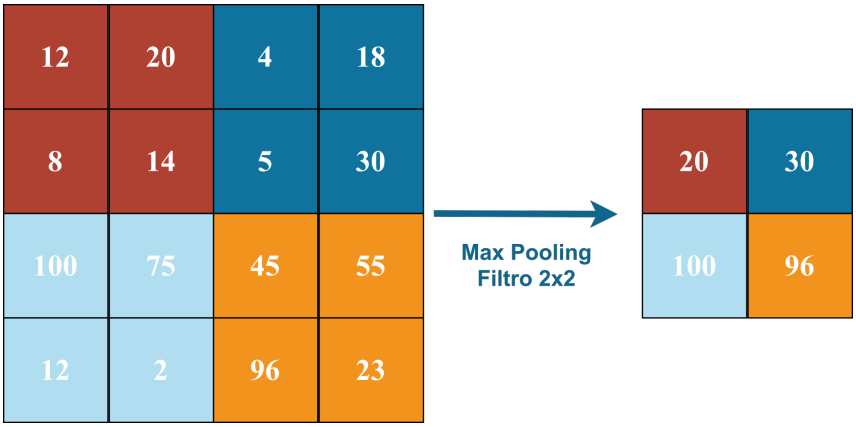


Figura 2.9: Método de Max Pooling

Capa totalmente conectada (Fully connected)

Las CNN suelen utilizar capas completamente conectadas al final de las capas de convolución y agrupación, donde cada píxel se considera una neurona separada, como en una red neuronal normal. Para predecir las clases, esta capa final de clasificación deberá tener un número de neuronas igual al número de clases que se desean predecir.

2.5.3. Entrenamiento

Un modelo en aprendizaje automático comienza sin conocimiento específico sobre el problema. Al ser expuesto a un conjunto de datos de entrenamiento que incluye ejemplos etiquetados, el modelo ajusta sus parámetros mediante un proceso iterativo. Primero, realiza predicciones iniciales y calcula la pérdida, que mide la discrepancia entre las predicciones y las etiquetas reales. Luego, mediante la retroalimentación de la pérdida, se ajustan gradualmente los pesos y sesgos del modelo utilizando técnicas como la propagación hacia atrás. Este proceso se repite varias veces para optimizar el modelo, haciendo que se adapte y mejore su capacidad para realizar predicciones precisas en nuevos datos no vistos durante el entrenamiento.

Función de costo

La función de costo calcula la diferencia entre el valor estimado por la red neuronal y la predicción real que contiene etiquetas. Minimizando esta diferencia, se ajustarán los parámetros de la red para que sean los mejores posible. Una función de costo habitual es la entropía cruzada (Cross-entropy), en la que cada probabilidad estimada por la red neuronal se contrasta con la etiqueta real. Luego, se calcula un puntaje que penaliza la probabilidad en función de la distancia respecto al valor esperado. La **Ecuación 2.4** presenta esta función de costo, donde H denota el tamaño del conjunto de datos de entrenamiento, $\boldsymbol{y_i}$ representa el vector de etiquetas reales y $\boldsymbol{p_i}$ corresponde a las predicciones de la red neuronal, generalmente interpretadas como valores de probabilidad. Cuando se trata de problemas de clasificación, la entropía cruzada se utiliza con frecuencia porque las predicciones convergen rápidamente y con mayor fuerza [59].

$$J(\boldsymbol{w}, b) = \frac{1}{H} \sum_{i=1}^{H} [\boldsymbol{y_i} \log(\boldsymbol{p_i}) + (1 - \boldsymbol{y_i}) \log(1 - \boldsymbol{p_i})] \tag{2.4}$$

Descenso de gradiente estocástico (Stochastic Gradient Descent, SGD)

El SGD es una técnica común para optimizar los parámetros de una DNN. Permite minimizar o maximizar alguna función $f(x)$ utilizando su gradiente respecto a x. Finalmente, la técnica termina en un mínimo local o global. La idea clave detrás del SGD es que el gradiente es una expectativa. Esta expectativa se puede estimar aproximadamente utilizando un pequeño conjunto de ejemplos. Específicamente, en cada iteración, se puede tomar una mini-batch de ejemplos $\mathbb{R} = \{x^{(1)}, x^{(2)}, \ldots, x^{(m')}\}$ seleccionados de manera uniforme del conjunto de entrenamiento. El tamaño del mini-batch, m', suele ser un número relativamente pequeño de ejemplos, que puede variar desde uno hasta varios cientos [56]. La estimación del gradiente se expresa mediante la **Ecuación 2.5**.

$$\boldsymbol{g} = \frac{1}{m'} \nabla_\theta \sum_{i=1}^{m'} J(\boldsymbol{\theta}) \tag{2.5}$$

En esta ecuación, J representa una función de pérdida, que podría ser cualquier función, como la entropía cruzada. Por otro lado, $\boldsymbol{\theta}$ se refiere a los parámetros de la red neuronal. La idea fundamental del algoritmo SGD es seguir el gradiente estimado para minimizar la función de pérdida y mejorar el rendimiento de la red neuronal, tal como se muestra en la **Ecuación 2.6**:

$$\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} - \eta \boldsymbol{g} \tag{2.6}$$

donde $\boldsymbol{g}$ es el gradiente y η representa la tasa de aprendizaje, reflejando el tamaño del paso en cada iteración.

Algoritmo de retropropagación (Back-propagation)

La retropropagación es el proceso por el cual una red neuronal modifica sus parámetros para "aprender" una representación interna de la información. En otras palabras, es el algoritmo que determina la dirección de máxima variación de cada capa y luego actualiza los pesos a través del gradiente descendente.

Comencemos con la **Ecuación 2.7**, donde la suma de los pesos multiplicados por la entrada y un sesgo se representa como $\boldsymbol{z}$. Esta información se pasa a través de una función de activación ϕ y finalmente llega a la función de costo J. Además, el número de capas de la red se representa con la letra L.

$$J[\phi(\boldsymbol{z}_L)] = J[\phi(\boldsymbol{w}_L \boldsymbol{x} + b_L)] \tag{2.7}$$

Para encontrar la dirección de variación máxima para la capa L, se utiliza la regla de la cadena para obtener la derivación de la ecuación anterior con respecto a $\boldsymbol{w}_L$.

$$\frac{\partial J}{\partial \boldsymbol{w}_L} = \frac{\partial J}{\partial \phi_L} \cdot \frac{\partial \phi_L}{\partial \boldsymbol{z}_L} \cdot \frac{\partial \boldsymbol{z}_L}{\partial \boldsymbol{w}_L} \tag{2.8}$$

En la **Ecuación 2.8**, los dos primeros términos de la derecha de la igualdad se refieren al error en la función de costo cuando hay un cambio en la suma de las neuronas. Esta definición se conoce como el error atribuido de las neuronas y se denota como $\boldsymbol{\delta}_L$. Mientras que el último término representa cómo cambia $\boldsymbol{z}_L$ con respecto a los pesos. Teniendo en cuenta que las entradas a esta capa son las salidas de la capa anterior, denotamos como ϕ_{L-1} la salida de la capa $(L-1)$. Por lo tanto, el cambio de la función de costo con respecto a los pesos se puede escribir como se muestra en la **Ecuación 2.9**:

$$\frac{\partial J}{\partial \boldsymbol{w}_L} = \boldsymbol{\delta}_L \cdot \phi_{L-1} \tag{2.9}$$

Ahora, continuando con el algoritmo y realizando el mismo análisis para las capas anteriores, se obtiene la **Ecuación 2.10**:

$$\frac{\partial J}{\partial \boldsymbol{w}_{L-1}} = \boldsymbol{\delta}_L \cdot \boldsymbol{w}_L \cdot \frac{\partial \phi_{L-1}}{\partial \boldsymbol{z}_{L-1}} \cdot \phi_{L-2} \tag{2.10}$$

En resumen, el algoritmo de retropropagación funciona siguiendo tres pasos deducidos de la explicación anterior, comenzando desde la última capa y realizando el mismo proceso secuencialmente hasta llegar a la primera capa:

1. Calcular el error de la última capa (L):

$$\boldsymbol{\delta}_L = \frac{\partial J}{\partial \phi_L} \cdot \frac{\partial \phi_L}{\boldsymbol{z}_L} \tag{2.11}$$

2. Propagar el error a la capa anterior $(L-1)$:

$$\boldsymbol{\delta}_{L-1} = \boldsymbol{w}_L \cdot \boldsymbol{\delta}_L \cdot \frac{\partial \phi_{L-1}}{\partial \boldsymbol{z}_{L-1}} \tag{2.12}$$

3. Calcule las derivadas de la capa utilizando el error::

$$\frac{\partial J}{\partial \boldsymbol{w}_{L-1}} = \boldsymbol{\delta}_{L-1} \cdot \boldsymbol{\delta}_{L-2} \tag{2.13}$$

2.5.4. Aprendizaje por transferencia

Los algoritmos tradicionales de ML y DL han sido diseñados tradicionalmente para trabajar de manera aislada. Estos algoritmos se entrenan para resolver tareas específicas y los modelos deben construirse desde cero una vez que cambia la distribución del espacio de características. El aprendizaje por transferencia (del ingles Transfer Learning, TL) propone superar el paradigma de aprendizaje aislado utilizando el conocimiento adquirido en una tarea para resolver otras relacionadas (ver **Figura 2.10**). El TL puede aprovechar el conocimiento (características, pesos y otros) de modelos previamente creados para entrenar nuevos, abordar problemas con pequeñas cantidades de datos y reducir el tiempo de entrenamiento [56].

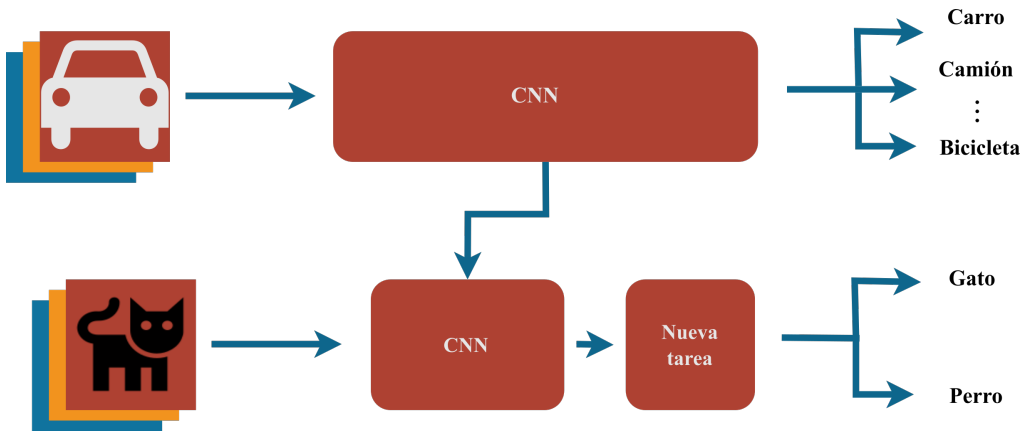


Figura 2.10: Proceso del aprendizaje por transferencia

Modelos pre-entrenados como extractores de características

Los sistemas y modelos de aprendizaje profundo son arquitecturas en capas que aprenden diferentes características en diferentes niveles (representaciones jerárquicas de características en capas). Estas capas suelen estar conectadas al final mediante una capa completamente conectada, en el caso de la clasificación, para obtener la salida final. Este tipo de arquitectura permite utilizar un modelo pre-entrenado sin su capa final como un extractor de características fijas para otras tareas. La idea principal de esta estrategia de transferencia de aprendizaje es aprovechar las capas de caracterización de un modelo previamente entrenado y luego utilizar estas características como una incrustación para clasificar el nuevo fenómeno.

Ajuste fino (Fine-tuning)

A diferencia de la estrategia anterior, el ajuste fino no solo reemplaza la capa final (para clasificación/regresión), sino que también modifica selectivamente algunas capas anteriores. En las NNs profundas, las capas iniciales capturan características genéricas, mientras que las capas posteriores se centran más en la tarea específica. Por lo tanto, es posible congelar ciertas capas mientras se vuelven a entrenar o ajustar el resto de ellas para que se adapten a nuestras necesidades. En este caso, se utiliza el conocimiento en términos de la arquitectura general de la red y se utilizan sus estados como punto de partida para nuestro paso de re-entrenamiento. Además, esto ayuda a lograr un mejor rendimiento con menos tiempo de entrenamiento.

2.6. Computación en la Nube, la Niebla y el Borde

La evolución tecnológica ha dado lugar a un panorama diverso en la forma en que las organizaciones y sistemas gestionan sus recursos computacionales y datos. En este contexto, tres paradigmas han surgido como pilares fundamentales para abordar las demandas de un mundo altamente conectado y digital: la computación en la nube, la computación en la niebla y la computación en el borde.

2.6.1. Computación en la nube (Cloud computing)

En este modelo los servicios, aplicaciones y datos se almacenan y ejecutan en servidores remotos y se hacen disponibles a través de Internet. Esto permite a las empresas acceder a una amplia gama de recursos computacionales de manera escalable y rentable, sin necesidad de invertir en infraestructura local costosa. La computación en la nube se divide en varios modelos de servicio como Infraestructura como Servicio (IaaS), Plataforma como Servicio (PaaS) y Software como Servicio (SaaS) que ofrecen diferentes niveles de control y gestión [60].

2.6.2. Computación en la Niebla (Fog Computing)

La computación en la niebla es una extensión de la computación en la nube que se enfoca en llevar el procesamiento y almacenamiento de datos más cerca de la fuente de datos, es decir, cercano al borde de la red, en lugar de depender

de centros de datos remotos. Esto se hace para reducir la latencia y mejorar la eficiencia en aplicaciones que requieren respuestas rápidas en tiempo real. Los dispositivos y nodos de niebla se ubican en puntos estratégicos de la red, más cerca de los sensores y dispositivos IoT (Internet de las cosas). La computación en la niebla es especialmente relevante en aplicaciones como la automatización industrial, la salud, los vehículos autónomos y las ciudades inteligentes [61].

2.6.3. Computación en el borde (Edge Computing)

La computación en el borde es una evolución de la computación de niebla y se refiere a la realización de cálculos y procesamiento de datos en dispositivos locales o en el "borde"de la red, lo más cerca posible de la fuente de datos. Esto se hace para minimizar la latencia, reducir la carga en la red y mejorar la privacidad de los datos al procesarlos de manera local. Los dispositivos de borde incluyen servidores locales, gateways IoT y dispositivos de usuario final. La computación en el borde es esencial en aplicaciones donde se requiere un procesamiento rápido y en tiempo real, como el análisis de vídeo, la automatización industrial, la telemedicina y los sistemas de vehículos autónomos [62].

En resumen, estos tres enfoques de computación, nube, niebla y borde, representan diferentes estrategias para gestionar datos y aplicaciones en un mundo cada vez más conectado y dependiente de la tecnología (ver **Figura 2.11**). Cada uno tiene sus propias ventajas y se adapta a diferentes casos de uso y requisitos de rendimiento.

2.7. CNNs en la computación en el borde

En el contexto de la detección de eventos relacionados con la seguridad ciudadana, las CNNs han emergido como una herramienta clave. Estas redes ofrecen una capacidad creciente para analizar vídeos o imágenes, logrando un alto nivel de precisión en la clasificación y detección de objetos. Sin embargo, esta precisión conlleva un alto costo en términos de recursos computacionales y de memoria. Teniendo en cuenta el ámbito de la computación en el borde, donde se busca realizar cálculos cercanos a la fuente de datos para mejorar los tiempos de respuesta y evitar la transferencia de datos a la nube, se han investigado diversas soluciones de hardware. Estas soluciones incluyen sistemas heterogéneos, sistemas en un chip (SoC), matrices de compuertas programables (FPGAs), unidades de procesamiento gráfico (GPUs) y CPUs de bajo

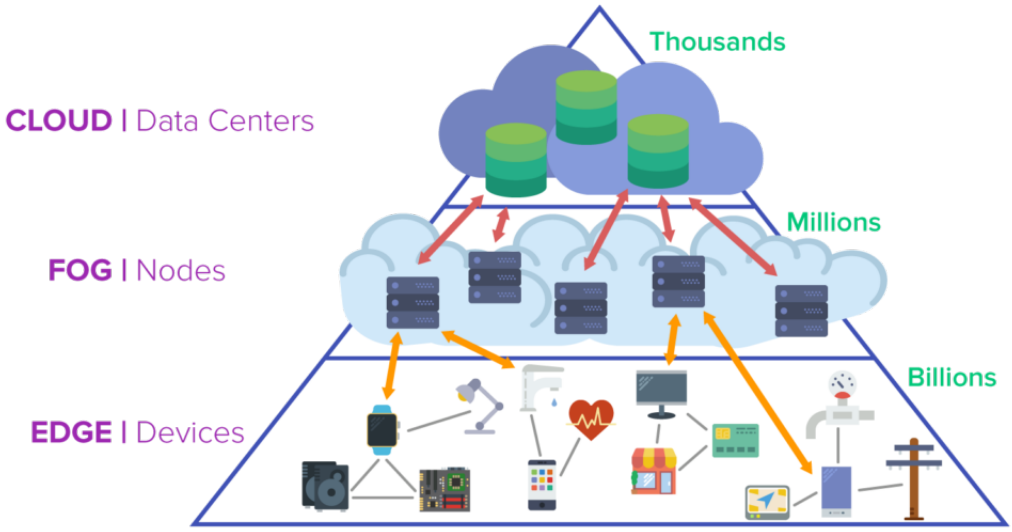


Figura 2.11: Estrategias para gestión datos y aplicaciones[2]

consumo. A pesar de los avances en el hardware, persisten desafíos significativos para mantener la precisión y el rendimiento de las CNNs en sistemas embebidos con restricciones de memoria y potencia [63]. Para hacer viable la ejecución de CNNs sobre dispositivos de borde es necesario aligerar un tanto los modelos. Es aquí donde cobran protagonismo las técnicas de compresión y fusión de redes neuronales. Estas técnicas buscan reducir la carga computacional y los requisitos de memoria de las CNNs sin sacrificar su eficacia en la detección de eventos de seguridad.

2.8. Compresión de redes neuronales

El objetivo principal de la compresión es reducir el tamaño de una NN, ya sea en número de parámetros o de bytes en disco; para de esta manera posibilitar la reducción en la latencia y el consumo de memoria al momento de realizar la inferencia.

[2]Imagen tomada de: `https://www.pubnub.com/blog/moving-the-cloud-to-the-edge-computing/`

2.8.1. Poda de NNs (Pruning)

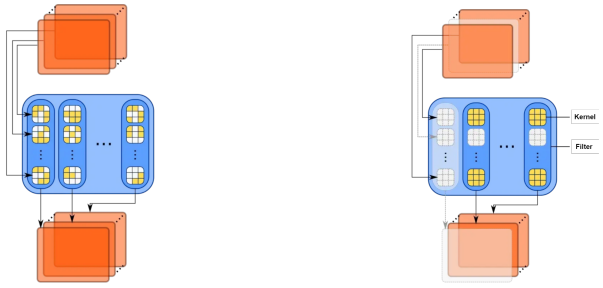
La poda de NNs es un método clásico en la compresión de modelos cuyos primeros trabajos datan del final del siglo XX [64], [65]. La poda implica la eliminación estratégica de componentes de la red que tienen poca importancia, son redundantes o no aportan información significativa. Esto se logra utilizando un umbral de decisión para determinar qué elementos deben conservarse o podarse. Uno de los métodos de poda es el método de expansión de Taylor, propuesto inicialmente por Molchanov *et al.* [66]. Esta técnica aproxima el impacto de eliminar parámetros específicos midiendo el cambio en la función de costo. Se centra en podar parámetros con gradientes de la función de costo cercanos a cero con respecto a sus mapas de características correspondientes. Es importante destacar que la implementación exitosa de este criterio requiere la acumulación tanto del gradiente de la función de costo como de su función de activación asociada durante el proceso de retropropagación. Además de la poda basada en el criterio de expansión de Taylor, otro enfoque es la poda aleatoria, donde no se tiene en cuenta la importancia de los pesos, sino que se elige al azar cualquier parámetro y se elimina. Este enfoque ofrece una alternativa que no depende de la importancia de los pesos y puede ser útil en ciertos casos. Aquí se describen dos enfoques comunes de poda en las redes neuronales:

- **Poda a nivel de peso:** elimina parámetros o canales innecesarios entre capas en una red neuronal.
- **Poda a nivel de unidad:** elimina todas las conexiones hacia una neurona o filtro y elimina todas las filas de la unidad en lugar de simplemente podar las conexiones.

La Figura 2.12 muestra la diferencia entre la poda a nivel de peso y la poda a nivel de unidad.

2.8.2. Cuantización de parámetros (Quantization)

Otro enfoque de compresión es la cuantización de parámetros. Las redes profundas suelen utilizar una precisión de punto flotante de 32 bits (FP32), ya sea para entrenamiento o inferencia, lo que conlleva una exigencia significativa en los requisitos computacionales y de memoria necesarios para llevarlos a cabo. La cuantización de parámetros consiste en utilizar una representación de menor precisión para almacenar los pesos, activaciones y gradientes de la red, con el objetivo de reducir la memoria requerida para almacenar la red,



(a) Poda a nivel de peso

(b) Poda a nivel de unidad

Figura 2.12: Diferencia entre (a) la poda a nivel de peso y (b) la poda a nivel de unidad[3]

a la vez que se aligeran los requerimientos de capacidad computacional para ejecutar el modelo [67].

2.9. Fusión de redes neuronales

El término fusión de NNs no se utiliza de manera amplia en la literatura de redes neuronales. Aunque algunos trabajos mencionan la fusión multimodal, consideramos que este es simplemente un caso específico de la fusión de redes neuronales. En general, la fusión de redes neuronales implica combinar varios modelos en uno solo que sea capaz de hacer lo que de manera aislada hacían los modelos individuales. De acuerdo con el conjunto de características de entrada y la cantidad de tareas que la red neuronal fusionada debe realizar es posible implementar una variedad de estrategias de fusión en este contexto. Estas estrategias pueden incluir un conjunto de características para una tarea, múltiples conjuntos de características para una tarea o un conjunto de características para múltiples tareas.

2.9.1. Un conjunto de características para una tarea

En esta estrategia, un conjunto de características se utiliza para alimentar múltiples modelos, todos ellos desempeñando la misma tarea. Su objetivo principal es adaptar el modelo predictivo a cambios dinámicos en los datos

[3]Imagen tomada de: `https://towardsdatascience.com/neural-network-pruning-101-af816aaea61`

de entrada, con la capacidad de cómputo disponible o con el tiempo disponible para realizar la tarea. Enfoques comunes para llevar a cabo esta fusión de redes neuronales incluyen estrategias como la "salida temprana"[68]-[70] y el "intercambio de modelos"[71], [72]. En la estrategia de salida temprana, si bien se considera como un solo modelo, cada salida requiere la inclusión de una capa softmax, lo que implica en que cada salida se convierte en un modelo predictivo independiente de los demás (ver **Figura 2.13**). Cada una de estas salidas está asociada con una porción de los parámetros del modelo global.

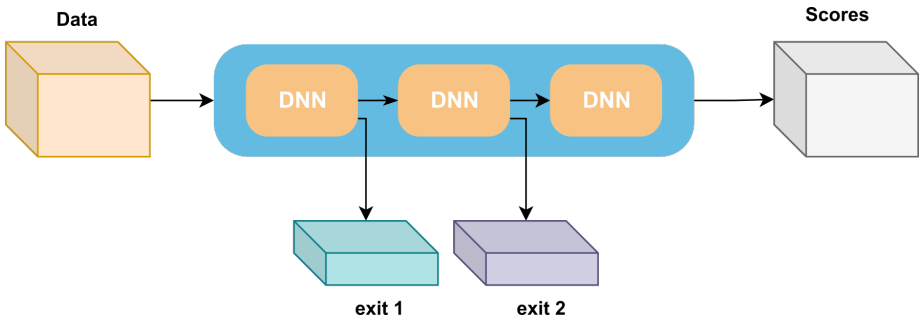


Figura 2.13: Diagrama de un conjunto de características para una tarea

2.9.2. Múltiples conjuntos de características para una tarea

Este método de fusión combina las características de varios sensores o fuentes de información complementarias para mejorar la capacidad predictiva en una sola tarea. La técnica de fusión de redes neuronales más frecuentemente mencionada en la literatura es la fusión multimodal, que se ha documentado en una gran cantidad de estudios [38], [73]-[77]. La fusión multimodal ha sido utilizada en tareas como el reconocimiento de emociones y análisis del sentimiento, usando características de entradas de tipo imagen, vídeo, audio y texto, tal como se puede ver representado en la **Figura 2.14**.

2.9.3. Un conjunto de características para múltiples tareas

Es común encontrar fuentes de información que funcionan como entradas simultáneas para múltiples modelos neuronales. Por ejemplo, una cámara de vídeo en una calle puede usarse para detectar peatones, identificar placas de automóviles, estimar la congestión de vehículos y reconocer personas según su

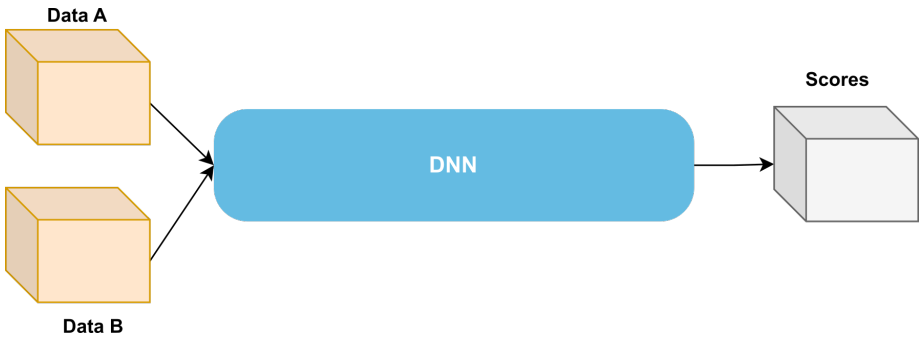


Figura 2.14: Diagrama de múltiples conjuntos de características para una tarea

forma de caminar, entre otras aplicaciones. En este contexto podemos definir un tercer método de fusión de redes neuronales. Este método utiliza una serie de características de entrada y las proporciona a múltiples modelos neuronales. En este caso la táctica principal es compartir parámetros de las capas de las NNs que están diseñadas para identificar características generales de la entrada. Luego, el modelo se especializa para realizar las tareas específicas (ver **Figura 2.15**).

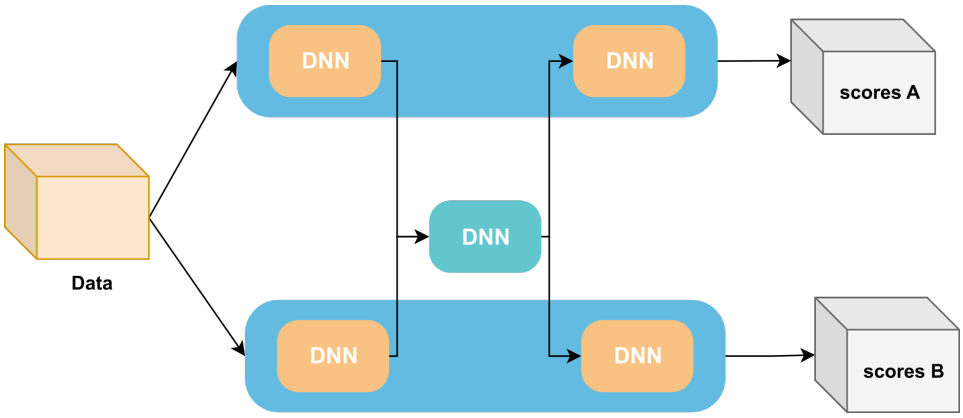


Figura 2.15: Diagrama un conjunto de características para múltiples tareas

Capítulo 3

Metodología

Esta sección se enfoca en describir los pasos seguidos en el desarrollo de este trabajo, con el propósito de detallar los procesos aplicados para la detección de eventos relacionados a la seguridad ciudadana utilizando técnicas de compresión, fusión y su combinación en modelos de redes neuronales. El enfoque principal de esta metodología es la detección de objetos que pueda ayudar a generar alertas cuando la seguridad de las personas se ve comprometida por su presencia.

El proyecto se divide en cuatro etapas interconectadas con el fin de posibilitar la detección de eventos de seguridad, optimizar recursos computacionales y aprovechar al máximo las redes neuronales convolucionales. En la primera etapa se realiza la selección de redes neuronales convolucionales para la detección de eventos relacionados con la seguridad ciudadana. Esto implica una revisión bibliográfica, la implementación de redes pre-entrenadas y la evaluación de su desempeño, seleccionando al menos dos redes que sean pertinentes para el proyecto.

En la segunda etapa, se centra en la compresión de redes neuronales para reducir su tamaño y mejorar la eficiencia computacional. Se identifican las técnicas de compresión, se implementan en las redes seleccionadas y se mide la reducción de tamaño y la eficiencia.

La tercera etapa se enfoca en la fusión de redes neuronales para optimizar aún más el uso de los recursos computacionales y de almacenamiento. Se seleccionan y aplican las técnicas de fusión en las redes, evaluando cómo mejoran el rendimiento y la eficiencia.

Por último, en la fase de evaluación de la combinación de técnicas, se aplica un enfoque en el que la compresión de las NNs se lleva a cabo con la consideración de una fusión posterior de modelos enfocados en el reconocimiento de un solo

tipo de objetos.
En este documento cada etapa se aborda en un capítulo independiente, buscando presentar de manera clara los aspectos claves para su implementación y evaluación. Para la obtención y el análisis de los resultados experimentales se contó con el siguiente entorno hardware/software: un servidor equipado con duna CPU AMD EPYC7453 corriendo a 2.75 GHz y dos tarjetas gráficas Nvidia A30; una plataforma Khadas VIM3 con arquitectura big-little A311D que incluye cuatro núcleos Cortex-A73 a 2.2 GHz y dos núcleos Cortex-A53 a 1.8 GHz, acompañados de 4 GB de RAM; y una plataforma Raspberry Pi 4 equipada con un procesador Quad-core Cortex-A7 a 1.8 GHz y 8 GB de RAM. Para el desarrollo y despliegue de las redes neuronales, se utilizaron los frameworks PyTorch en su versión 2.0.1 y ONNX (Open Neural Network Exchange) en la versión 1.14.0. PyTorch es un framework de código abierto que facilita la creación y entrenamiento de modelos de aprendizaje profundo, mientras que ONNX [78] es un formato que permite la interoperabilidad entre diferentes frameworks de aprendizaje automático.

Capítulo 4

Identificación de redes neuronales para la detección de eventos relacionados a la seguridad ciudadana

Identificar NNs especializadas en la detección de eventos relacionados con la seguridad ciudadana es un aspecto fundamental para esta investigación. Este proceso implica la selección de arquitecturas de NNs que muestren buen rendimiento en la tarea de detección de objetos que atenten contra la seguridad de las personas.

4.1. Selección de redes neuronales convolucionales

Cuando llevamos a cabo la búsqueda de CNNs relacionadas con la seguridad ciudadana, nos encontramos con la limitación de que no había redes disponibles de uso libre específicamente diseñadas para este propósito. Ante esta situación, realizamos una búsqueda exhaustiva de redes pre-entrenadas que contuvieran etiquetas relacionadas con la seguridad ciudadana.

A partir de esta exploración se identifican diversas redes, entre las cuales se encuentra AlexNet [79], caracterizada por 5 capas convolucionales y 3 capas totalmente conectadas, logrando una precisión del 84.6 % TOP-5[1]. Asimismo,

[1] *TOP-5 (tasa de error 5): Es porcentaje de imágenes donde la etiqueta correcta no es una de las cinco etiquetas más probables del modelo*

hemos destacado VGG-16, compuesta por 13 capas convolucionales y 3 capas totalmente conectadas, con una precisión del 92.7 % en la misma prueba. Finalmente, ResNet [27] con una precisión del 96.4 % en la prueba TOP-5, gracias a su arquitectura de 152 capas y la implementación de conexiones de salto, conocidas como bloques residuales, para contrarrestar el desvanecimiento del gradiente. En la **Tabla 4.1** se muestran los resultados de precisión obtenidos en prueba TOP-5 para las arquitecturas AlexNet, ResNet y VGG-16. Los valores de precisión se presentan como porcentaje, mientras que el número de capas indica la profundidad de cada red.

Tabla 4.1: Comparación la precisión y la cantidad de capas entre diferentes arquitecturas de CNNs

Comparación entre CNNs		
CNN	**# de Capas**	**Precisión en TOP-5**
AlexNet	8	84.6 %
ResNet	152	96.4 %
VGG-16	16	92.7 %

Elegimos para nuestro caso de estudio la CNN VGG-16, ya que esta red cuenta con múltiples capas convolucionales y tiene mejor precisión que AlexNet. Se descartó ResNet ya que introduce bloques residuales, lo que implica más conexiones a tener en cuenta al momento de aplicar los método de poda. Después de la elección de la red VGG-16, nos encontramos con un desafío relacionado con las base de datos para realizar el re-entrenamiento de la red. La mayoría de las bases de datos reportadas presentan imágenes en condiciones ideales que no reflejan adecuadamente las situaciones reales que buscamos abordar en nuestro trabajo. Reconociendo la importancia de contar con imágenes que representen escenarios más cercanos a la realidad, tomamos la decisión de emprender la creación de una base de datos propia. Los detalles y procesos específicos de construcción de esta base de datos se abordan de manera detallada en la sección 4.3.

4.1.1. VGG-16

La arquitectura VGG es una CNN introducida por el Grupo de Geometría Visual de la Universidad de Oxford. Simonyan y otros [22] propusieron varias arquitecturas y configuraciones de CNN profundas, una de las cuales se presentó en el ImageNet Large Scale Visual Recognition Challenge 2013 (ILSVRC-2013). La **Figura 4.1** muestra la configuración de VGG-16. La principal diferencia entre VGG-16 y sus predecesores es el uso de una serie de capas

convolucionales con campos receptivos pequeños en las capas iniciales en lugar de unas pocas capas con campos receptivos grandes. Esto resulta en menos parámetros y más no linealidades entre ellos, lo que hace que la función de decisión sea más discriminatoria y que el modelo sea más fácil de entrenar.

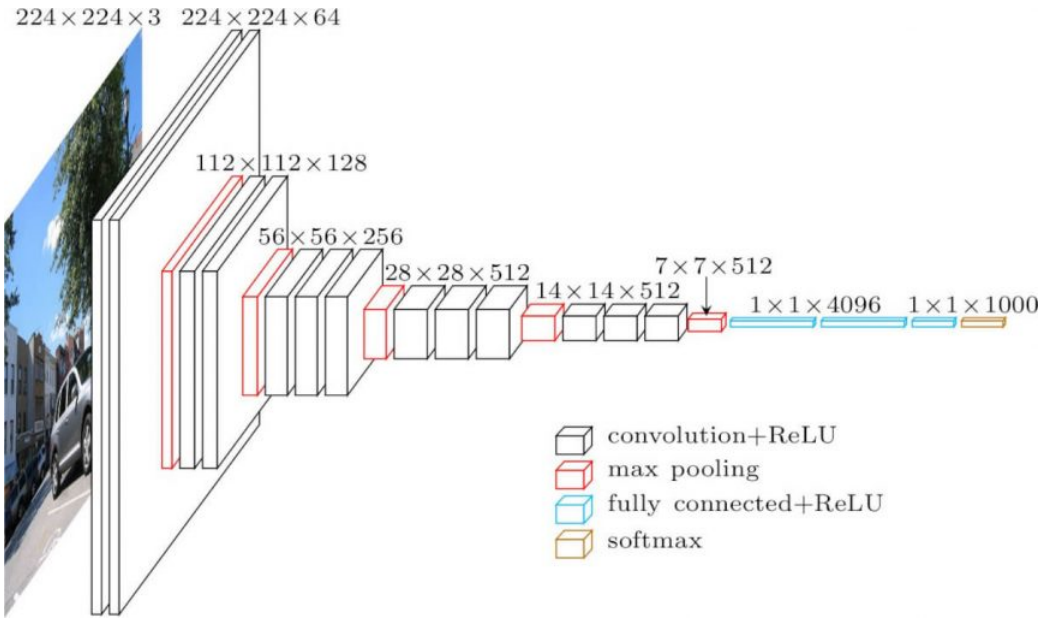


Figura 4.1: Arquitectura de la CNN VGG-16[2]

4.2. Métricas de evaluación de desempeño

En esta sección exploramos las métricas utilizadas para evaluar el rendimiento de nuestros modelos para la detección objetos en imágenes. Estas métricas son fundamentales para comprender el desempeño de los modelos en la clasificación de eventos en imágenes y su complejidad computacional.

4.2.1. Precisión de modelo

Es crucial evaluar de manera exhaustiva el rendimiento de un modelo, especialmente en el contexto de clasificación. Para lograr una evaluación precisa, es esencial emplear métricas adecuadas que midan con exactitud la capacidad del modelo para realizar clasificaciones precisas.

[2]Imagen tomada de: [80]

Precisión (accuracy)

Esta métrica proporciona una medida general de la capacidad del modelo para clasificar correctamente las clases. Se calcula dividiendo el número de predicciones correctas entre el total de predicciones realizadas. La precisión, al ofrecer una visión global del rendimiento, es particularmente útil cuando todas las clases son igualmente importantes y no hay desbalance significativo en la distribución de las clases. La precisión se puede calcular como se muestra en la **Ecuación 4.1** [81].

$$\text{Precisión} = \frac{\text{VP} + \text{VN}}{\text{VP} + \text{VN} + \text{FP} + \text{FN}} \tag{4.1}$$

Donde:

- **VP:** Verdaderos positivos
- **VN:** Verdaderos negativos
- **FP:** Falsos positivos
- **FN:** Falsos negativos

4.2.2. Complejidad computacional

La complejidad computacional de nuestros modelos de clasificación de objetos que comprometan la seguridad ciudadana es una métrica esencial para evaluar su eficiencia en términos de recursos de cómputo requeridos.

Tiempo de ejecución

El tiempo de ejecución se define como el lapso necesario para llevar a cabo una tarea específica. En el contexto de nuestro estudio, este lapso representa el intervalo requerido para que nuestros modelos de clasificación de armas realicen la categorización de eventos en imágenes. La variación en el tiempo de ejecución se expresa a través del delta o diferencial de tiempo, ofreciendo una perspectiva sobre el rendimiento del modelo en el escenario de aplicación.

Uso de memoria en ejecución

La utilización de memoria en ejecución, medida en bytes (B), constituye una métrica fundamental para evaluar la eficiencia de nuestros modelos durante

la fase de inferencia. Indica la cantidad de memoria RAM que se necesitan mientras se encuentran en ejecución para llevar a cabo la tarea de clasificación de eventos que pueden poner en riesgo la seguridad de las personas. Esta métrica resulta esencial para garantizar la compatibilidad de los modelos con dispositivos de borde que presentan limitaciones en su capacidad de memoria.

En conjunto, estas métricas permiten evaluar de manera exhaustiva y coherente el desempeño en la clasificación asegurando que sean eficientes y precisos, lo que es fundamental para su despliegue en dispositivos de borde con recursos limitados.

4.3. Base de Datos

Dentro del ámbito de la seguridad ciudadana, es crucial disponer de bases de datos con imágenes precisas y representativas. En particular, es fundamental que estas bases de datos contengan imágenes que representen situaciones de riesgo en escenarios del mundo real, donde las situaciones de riesgo son solo una parte de la escena completa. Sin embargo, al comenzar a buscar las bases de datos apropiadas para nuestro proyecto, encontramos con un problema importante: la mayoría de las bases de datos existentes no lograban capturar la complejidad y autenticidad de dichas situaciones en entornos reales.

Exploramos diversas bases de datos que contenían situaciones de riesgo registradas en vídeos, abarcando incidentes como accidentes de tráfico, peleas, robos y otros eventos de interés en el contexto de la seguridad ciudadana, como se detalla en la **Tabla 4.2**. Por el tipo de situaciones que incluye (accidentes, robos y peleas) UFC Crime surgió como una base de datos relevante. Sin embargo, al profundizar en su análisis, identificamos ciertas limitaciones que llamaron nuestra atención. En ocasiones, los vídeos carecían de la calidad necesaria y las etiquetas asignadas a los eventos no siempre coincidían de manera precisa con la situación de riesgo que se estaba desarrollando. Las otras bases de datos no resultaron de interés para nuestro estudio, por el tipo de situaciones en las que estaban centradas (correr por la calle, circulación de ciclistas o automóviles, por ejemplo).

Ante estos desafíos decidimos explorar otra alternativa: la base de datos IMFDb [25]. A diferencia de las bases de datos previamente consideradas, la IMFDb se centra exclusivamente en imágenes de armas de fuego, extrayendo su contenido de diversas películas. Aunque este recurso tenía un valor potencial para el ámbito de la seguridad ciudadana, encontramos con una restricción significativa: las imágenes de armas estaban perfectamente orquestadas, mostrando únicamente el arma en un entorno con fondo blanco. A pesar de que

Tabla 4.2: Datasets y ejemplos de anomalías

Datasets				
Dataset	**# de videos**	**Frames promedio**	**Ejemplo de anomalías**	**Referencia**
Avenue	37	839	Correr, arrojar, nuevo objeto	[82]
UMN	5	1290	Correr	[83]
DAD	1730	100	Accidentes de tráfico	[84]
ShangaiTech	437	726	Ciclistas, automóviles	[85]
UFC Crime	1900	7247	Incendio provocado, accidente, robo, pelea	[86]
StreetScene	81	2509	Cruzar la calle, estacionarse ilegalmente	[87]

esta representación es útil para otros propósitos, no capturaba la autenticidad de situaciones reales que buscábamos para el proyecto.

En el contexto de las ciudades inteligentes donde la eficiencia y la gestión de datos son fundamentales, la capacidad para identificar eventos de riesgo, como el uso de armas y cuchillos, se convierte en un pilar esencial para garantizar la seguridad de las personas.

La presencia de armas y cuchillos en entornos urbanos puede desencadenar situaciones críticas que requieren respuestas inmediatas. La detección temprana de estos eventos permite una acción preventiva, contribuyendo así a la prevención de incidentes graves y a la gestión eficiente de situaciones de riesgo. Dado que la tecnología desempeña un papel central en la mejora de la calidad de vida de los ciudadanos, la capacidad de identificar y abordar amenazas potenciales fortalece directamente la seguridad y el bienestar de la comunidad, alineándose con los principios fundamentales de una ciudad inteligente y segura.

Como respuesta a esta limitación creamos un nuevo conjunto de datos. Este dataset se compone de imágenes extraídas de películas en las que se capturaron diversas escenas que reflejaban situaciones de riesgo similares a las que ocurren en el mundo real. Nuestro conjunto de datos recibe el nombre de "Weapons in Movies" y se diseñó con la intención de representar situaciones realistas donde se muestra una escena completa, resultando fundamental para nuestro enfoque en la detección de situaciones de riesgo en entornos auténticos.

4.3.1. Metodología de creación de la base de datos

La creación de nuestra base de datos se llevó a cabo en varias etapas meticulosamente planificadas. Comenzamos recopilando material visual de películas que contenían escenas con elementos de interés, como armas de fuego, cuchillos y otros objetos relacionados. En total, se recopilaron 2180 imagenes de escenas que contenían estos elementos.

Una vez que obtuvimos este conjunto inicial de imágenes, el siguiente paso fue el proceso de etiquetado. Para lograr esto, utilizamos la herramienta LabelStudio [88], que permitió generar cuadros de delimitación (bounding boxes) alrededor de los objetos de interés presentes en cada imagen. Fueron seleccionados varios elementos relacionados con tres categorías principales: armas de fuego, armas blancas y objetos no relacionados con armas. Además, este conjunto de datos abarca subcategorías específicas, que se detallan a continuación:

Armas de fuego

- Pistola
- Revólver
- Fusil
- Escopeta
- Rifle de francotirador

Armas blancas

- Cuchillo
- Machete
- Navaja

Objetos

- Teléfonos celulares
- Tarjetas
- Billetera
- Libros
- Bolsas
- Botella

Con el objetivo de aumentar la diversidad de nuestra base de datos y garantizar la uniformidad en el procesamiento, desarrollamos un algoritmo para estandarizar todas las imágenes a un tamaño de 1024 por 1024 píxeles. Esto aseguró que todas las imágenes tuvieran la misma resolución, lo que simplificó el procesamiento posterior.

Luego, implementamos un proceso de división de las imágenes en subimágenes de 256 por 256 píxeles. Para este paso, consideramos un factor crítico: el solapamiento entre las subimágenes y la precisión. La precisión se obtiene entrenando la red VGG-16 con las subimágenes obtenidas a partir de los diferentes niveles de solapamiento. Para el uso de estas subimágenes en un entorno de clasificación, nos enfocamos en etiquetar cada subimagen que superara un umbral de solapamiento del 10 % con respecto al bounding box. Esto implicaba que si una subimagen contenía una porción significativa de un objeto de interés, se etiquetaba con la misma etiqueta del bounding box correspondiente. Este enfoque aseguraba que las subimágenes seleccionadas contuvieran una porción del objeto de interés en su interior. La **Tabla 4.3** presenta los resultados de la exploración del porcentaje de solapamiento entre las subimágenes y su respectiva presición. Se evaluaron porcentajes desde 20 % hasta 80 % en relación con las imágenes de armas. Observamos que el porcentaje de solapamiento óptimo que proporcionó la mejor precisión fue del 60 %. Esta exploración desempeñó un papel crucial en la determinación del valor de solapamiento adecuado para nuestras subimágenes, y en consecuencia, decidimos crear todas las imágenes

de nuestro conjunto de datos con un porcentaje de solapamiento del 60 %.

Tabla 4.3: Se muestra la relación entre el porcentaje de solapamiento y la precisión de la clasificación en el conjunto de datos.

Porcentaje de Solapamiento vs. Precisión	
Porcentaje de Solapamiento	**Precisión**
20 %	74 %
40 %	74 %
60 %	82 %
80 %	77 %

Sin embargo, para garantizar que las subimágenes etiquetadas fueran verdaderamente relevantes y proporcionaran información útil para la clasificación de objetos, llevamos a cabo un proceso adicional de filtrado. Esto implicó la aplicación de medidas como la entropía y la complejidad de la imagen. Las subimágenes que aportaban mayor información a partir de estas métricas se consideraron valiosas y se incluyeron en el conjunto de datos final, ya que se consideraba que contenían información relevante para la clasificación de objetos.

Gracias a esta metodología, logramos un aumento de 8,7× en la cantidad de imágenes en nuestro conjunto de datos. Específicamente, obtuvimos un total de 17700 imágenes que pertenecen a la categoría de armas de fuego (BD-A)y 1300 imágenes para la categoría de cuchillos (BD-B). Es importante destacar que la cantidad de imágenes para la categoría de objetos se mantuvo igual en cada caso, asegurando así que nuestras bases de datos estén balanceadas y proporcionen una representación equitativa de las diferentes categorías. La **Figura 4.2** muestra ejemplos de las imágenes que forman parte de nuestro conjunto de datos.

4.4. Construcción de los modelos de armas de fuego y cuchillos

Después de seleccionar la red base y lograr una base de datos equilibrada, procedimos con el proceso de TL, utilizando el optimizador SGD. Optamos por llevar a cabo este proceso a lo largo de 20 épocas de entrenamiento, ya que después de este punto se empieza a observar sobre-entrenamiento. Reentrenamos la red VGG-16 con los conjuntos de datos BD-A y BD-B donde el 75 % fue para entrenamiento y el 25 % para prueba. En el caso de la red especializada en

(a) Dataset para la detección de armas de fuego (BD-A)

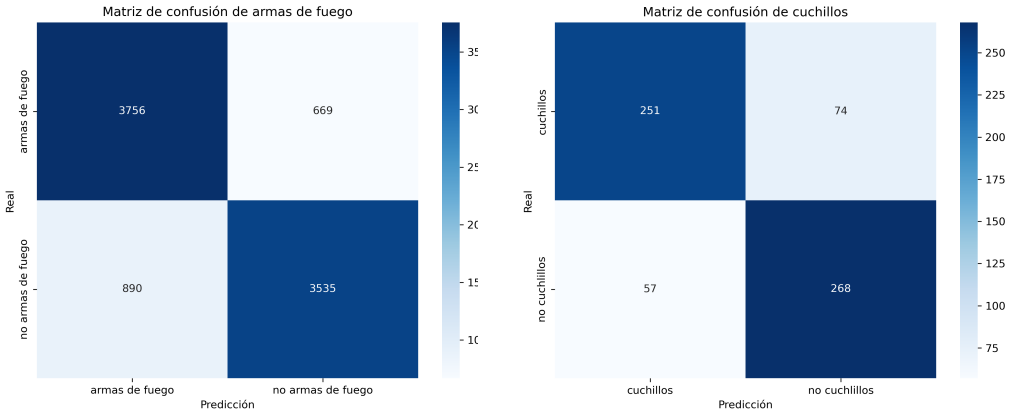
(b) Dataset para la detección de cuchillos (BD-B)

Figura 4.2: Dataset "Weapons in Movies"que contiene (a) armas de fuego y (b) cuchillos

la detección de cuchillos, logramos alcanzar una precisión del 80 %, mientras que para la red enfocada en la detección de armas de fuego, obtuvimos una precisión del 82 %. En la **Figura 4.3** se presenta las matrices de confusión de los modelos para identificar armas de fuego y cuchillos en imágenes. Cada matriz muestra la distribución de las predicciones del modelo en comparación con las clases verdaderas.

Estos resultados son altamente alentadores, ya que demuestran la eficacia de nuestras redes adaptadas en la identificación de eventos de seguridad relacionados con cuchillos y armas de fuego en entornos urbanos.

Con lo expuesto en este capítulo se da cumplimiento del objetivo 1 al lograr obtener dos redes que detectan eventos críticos relacionados con la seguridad ciudadana.



(a) Matriz de confusión modelo de armas de fuego

(b) Matriz de confusión modelo de cuchillos

Figura 4.3: Matrices de confusión para los modelos (a) armas de fuego y (b) cuchillos

Capítulo 5

Compresión de redes neuronales

El objetivo principal de la compresión en redes neuronales es reducir su tamaño, para que ocupe menos espacio en memoria y se haga un poco más ligera su ejecución. Hay una redundancia significativa en las NNs debido principalmente a la sobreparametrización en la red [89]. Los modelos sobreparametrizados requieren más memoria y capacidad de procesamiento, lo que dificulta su implementación en dispositivos con recursos limitados. Por esta razón es necesario comprimir las NNs para reducir sus requisitos computacionales y de almacenamiento sin comprometer significativamente su precisión. Este proceso involucra, en primera instancia, llevar a cabo una poda a nivel de unidades y posteriormente aplicando cuantización.

5.1. Poda iterativa

En este sección exploramos en detalle el proceso de poda iterativo aplicado a la red, inicialmente entrenada para la detección de armas y que presentaba una precisión del 82 %. Este método de poda se realiza de manera iterativa a nivel de unidad, seleccionando los filtros para su eliminación según los criterios de Taylor y aleatorio. A diferencia de la poda a nivel de peso, este enfoque permite la eliminación de todas las conexiones que rodean la unidad en cuestión, en lugar de solo centrarse únicamente en el peso individual de la red.

Es importante destacar que este proceso iterativo se configura cuidadosamente para definir la cantidad de filtros a eliminar en cada iteración, en este caso, se mantuvo constante en 512 filtros por iteración. Además, se realizó un barrido para la poda, abarcando una eliminación de filtros que oscila entre el 20 % y el 95 % del total de filtros del modelo. Esta configuración permite una adaptabilidad y control sobre el grado de poda aplicado, lo cual es esencial pa-

ra encontrar un equilibrio óptimo entre la eficiencia y la precisión del modelo resultante.

5.1.1. Criterio de Taylor

El criterio de Taylor para la poda de NNs se basa en la derivada parcial de la pérdida respecto a los pesos de la red. En este contexto, la idea es identificar y retener los filtros que tienen un impacto más significativo en la función de pérdida durante el entrenamiento. Matemáticamente, la derivada parcial de la pérdida $\boldsymbol{L}$ con respecto a un peso $\boldsymbol{w}_i$ se calcula como:

$$\frac{\partial \boldsymbol{L}}{\partial \boldsymbol{w}_i} \tag{5.1}$$

Cuando se aplica el criterio de Taylor para la poda, se seleccionan los pesos con los valores absolutos más bajos de sus derivadas parciales. La intuición detrás de esto es que los pesos con derivadas pequeñas tienen menos impacto en la función de pérdida y, por lo tanto, pueden ser eliminados sin afectar significativamente el rendimiento del modelo. En el **Algoritmo 1** presentamos una visión general del algoritmo utilizado en esta estrategia.

El proceso implica la identificación y eliminación gradual de los pesos menos importantes, lo que conduce a una red más compacta y eficiente.

5.1.2. Criterio aleatorio

El criterio aleatorio para la poda de NNs implica la selección de conexiones o pesos de la red de manera aleatoria para su eliminación durante el proceso de poda. A diferencia de enfoques más deterministas, como el criterio de Taylor, la selección aleatoria no se basa en ninguna medida específica de la importancia de los pesos en términos de la pérdida o el rendimiento del modelo.

En este método se eligen filtros al azar y se eliminan durante cada iteración de poda. En el **Algoritmo 2** presentamos una visión general del algoritmo utilizado en esta estrategia. Este enfoque busca explorar diferentes configuraciones de la red.

5.2. Evaluación de la compresión

Después de realizar la poda con los criterios de Taylor y aleatorio procedemos a evaluar el rendimiento de los modelos comprimidos en términos de

Algoritmo 1: Algoritmo de poda con criterio de Taylor

Input: Red preentrenada, umbral de poda, conjunto de datos de validación

Output: Modelo podado

```
while No se cumplan criterios de parada do
    // Calcular derivadas parciales de la pérdida con respecto a los pesos
    Calcular ∂L/∂w_i para cada peso w_i en la red;
    // Seleccionar filtros con derivadas parciales más bajas
    for Cada capa de la red do
        foreach Conexión en la capa do
            if El valor absoluto de ∂L/∂w_i es menor que el umbral de poda then
                Eliminar conexión;
    // Evaluar el rendimiento del modelo podado
    Calcular precisión en el conjunto de datos de validación;
    // Actualizar criterios de parada si es necesario
    if Criterios de parada alcanzados then
        Detener el proceso de poda;
```

Algoritmo 2: Algoritmo de poda con selección aleatoria

Input: Red preentrenada, umbral de poda, conjunto de datos de validación
Output: Modelo podado

```
1 while No se cumplan criterios de parada do
      // Seleccionar aleatoriamente filtros para eliminar
2     for Cada capa de la red do
3         foreach Conexión en la capa do
4             if Generar número aleatorio ≤ umbral de poda then
5                 Eliminar conexión;
      // Evaluar el rendimiento del modelo podado
6     Calcular precisión en el conjunto de datos de validación;
      // Actualizar criterios de parada si es necesario
7     if Criterios de parada alcanzados then
8         Detener el proceso de poda;
```

su precisión, tiempo de ejecución y consumo de memoria. Para una evaluación exhaustiva empleamos los frameworks PyTorch y ONNX.

La elección de utilizar ONNX en nuestra evaluación se fundamenta en la necesidad de tener un segundo framework a nivel de ejecución para medir el impacto en el tiempo de ejecución y el consumo de memoria después de aplicar el proceso de poda a nuestro modelo implementado en PyTorch. ONNX facilita la portabilidad de modelos al permitir su conversión entre diferentes frameworks de machine learning como PyTorch. En este contexto, la ejecución de modelos ONNX se simplifica gracias a ONNX Runtime [78] donde usamos la version 1.15.1, una biblioteca diseñada para operar modelos ONNX de manera eficiente en diferentes plataformas.

En resumen, los pasos seguidos fueron los siguientes (ver **Figura 5.1**): inicialmente, seleccionamos la red VGG-16 ajustada para la detección de armas de fuego. A continuación, aplicamos el proceso de poda utilizando el un criterio para optimizar la estructura de la red. Posteriormente, convertimos el modelo resultante al formato ONNX. Como paso adicional, aplicamos una técnica de cuantización para mejorar aún más la eficiencia del modelo. Finalmente, implementamos el modelo optimizado en dispositivos de borde.

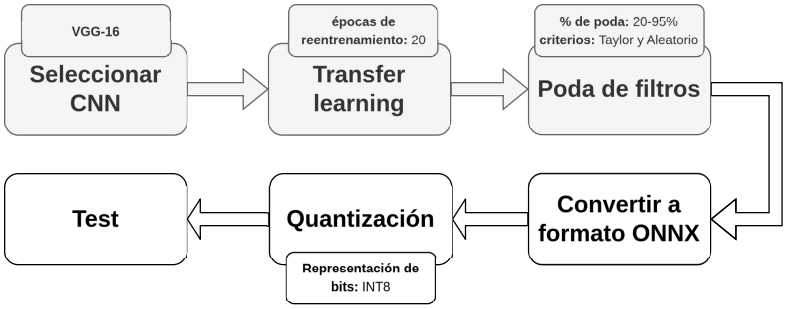


Figura 5.1: Pasos para realizar la optimización de la red

5.2.1. Resultados de la compresión

Presentamos nuestros resultados comparando la precisión de la red VGG-16 podada y reentrenada para la detección de armas de fuego utilizando nuestro conjunto de datos, considerando varios porcentajes de poda y criterios. Para medir con precisión el tiempo de inferencia, realizamos 1000 ejecuciones de inferencia y registramos el tiempo que llevó el proceso. Este enfoque elimina cualquier tiempo relacionado con la carga de bibliotecas y la carga de imágenes, lo que nos permite obtener una medida más precisa. Además, evaluamos el grado de poda en las capas de la red y analizamos el tiempo de ejecución y el consumo de memoria en diferentes plataformas.

Precisión vs. Porcentaje de poda

La **Tabla 5.1** resalta la eficacia de nuestro enfoque de poda, con la evaluación de precisión realizada sobre el servidor GPU. Los resultados muestran que el uso del criterio de Taylor para la poda produce una precisión superior en comparación con la poda aleatoria. Esta tendencia se alinea perfectamente con el concepto fundamental de la expansión de Taylor, que enfatiza la importancia de eliminar parámetros con una influencia mínima en la función de costo para mejorar la precisión del modelo podado.

Análisis de tiempo de los modelos

La **Figura 5.2** muestra los resultados del tiempo de inferencia de los modelos original, podado y podado y cuantizado. Para el modelo podado se eligió una reducción del 60 % de los filtros, y luego para el modelo podado y cuantizado se empleó una precisión de 8 bits (INT8), en contraste con los 32 bits del modelo original y el podado. Optamos por seleccionar el modelo comprimido al 60 % de su tamaño original, utilizando el criterio de Taylor durante el

Tabla 5.1: Porcentaje de filtros podados vs. Precisión para los criterios aleatorio y de Taylor en el servidor GPU

Aleatorio vs. Criterio de Taylor			
% de Poda	**# de filtros**	**Acc. Random**	**Acc. Taylor**
Antes de poda	4224	82 %	82 %
20 % de filtros podados	3380	84 %	85 %
40 % de filtros podados	2535	82 %	85 %
60 % de filtros podados	1690	76 %	82 %
80 % de filtros podados	845	74 %	78 %
90 % de filtros podados	423	61 %	75 %
95 % de filtros podados	212	50 %	74 %

proceso de poda. Esta elección se basa en la observación de que, a pesar de la reducción significativa en el número de filtros, el modelo podado y comprimido logra mantener la misma precisión que el modelo original. Es importante destacar que el modelo cuantizado mostró mejoras de velocidad notables, con un impresionante aumento de 8.23× en la plataforma VIM3 y una aceleración significativa de 2.51× en la Raspberry Pi 4 usando ONNX, en comparación con el modelo original de PyTorch ejecutándose sobre la plataforma correspondiente. Estas mejoras evidentes subrayan aún más las eficiencias operativas derivadas de los procesos integrados de poda y cuantización.

Además, al comparar los tiempos de ejecución entre PyTorch y ONNX arroja una observación interesante. La diferencia evidente sugiere que PyTorch destaca como una herramienta efectiva para el entrenamiento de modelos, ya que utiliza su propio formato optimizado para almacenar los parámetros del modelo, facilitando así un rendimiento eficiente durante la fase de entrenamiento. Por otro lado, ONNX se presenta como una herramienta valiosa para la conversión de modelos, adoptando un enfoque más genérico al representar los modelos en un formato intermedio. Este formato intermedio de ONNX permite la interpretación por diversos frameworks, brindando así una mayor portabilidad entre plataformas durante la implementación y ejecución de modelos entrenados. Esta dualidad ofrece un enfoque de optimización de dos etapas, donde PyTorch sobresale en el entrenamiento y ONNX destaca en la posterior conversión y despliegue de modelos en diferentes entornos. Es esencial destacar que, a pesar de esta diferencia en el tiempo de ejecución, la conversión de modelos a través de ONNX no comprometió la precisión las simulaciones arrojaron una precisión uniforme, alrededor del 82 %.

Cabe señalar que al ejecutar el modelo podado y cuantizado con PyTorch

se observan incrementos en el tiempo de ejecución. Estos incrementos pueden ser atribuibles a las complejidades inherentes a la cuantización, que afectan la eficiencia computacional en el entorno específico de PyTorch. Además, es posible que parte de esta variación en el rendimiento se deba a la generación de código para el SBC el compilador no es muy hábil para paralelizar el procesamiento de valores expresados como INT8 sobre una arquitectura de 32 bits que es la que poseen los SBCs. Esta interacción entre la cuantización y las limitaciones de la arquitectura del SBC puede contribuir a los incrementos observados en el tiempo de ejecución durante la implementación del modelo cuantizado en dicho entorno.

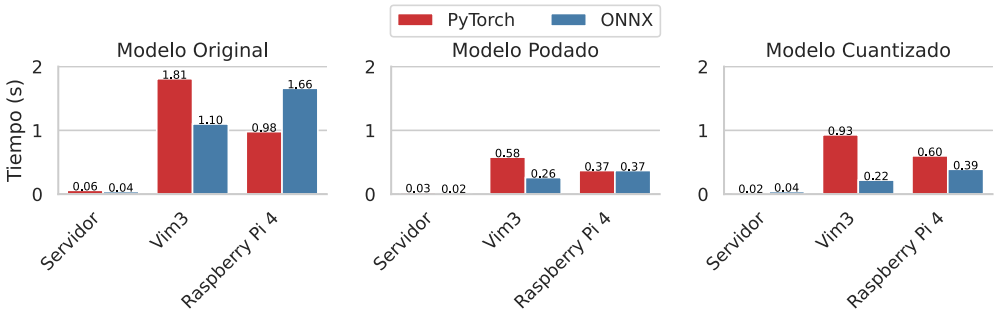


Figura 5.2: Tiempo de ejecución para inferencia en diferentes plataformas

Análisis de consumo de memoria de los modelos

En la **Figura 5.3** y la **Tabla 5.2** se ofrece una visión integral del consumo de memoria en las tres plataformas, registrando la utilización en KB. Un aspecto particularmente notable se revela durante la ejecución en Vim3, donde el modelo podado y cuantizado muestra un consumo de memoria a partir de 138.7 KB. Este hecho resalta la marcada reducción en el uso de memoria lograda mediante la poda y la cuantización, representando una mejora significativa en la eficiencia de recursos durante la implementación del modelo. Se observa una reducción en un factor de hasta 7× del modelo podado y cuantizado en comparación con el modelo original ejecutados en la Vim3. Es crucial subrayar que este menor consumo de memoria resulta particularmente favorable para dispositivos de borde, donde la restricción de recursos de memoria es una consideración crítica. Sin embargo, es interesante notar que al emplear nuevamente PyTorch para la ejecución del modelo, se observa que, en el caso de la ejecución del modelo podado y cuantizado, el consumo de memoria in-

cluso supera al registrado cuando se ejecuta sin cuantizar. Estas observaciones resaltan la complejidad y las variaciones en la eficiencia de los frameworks, enfatizando la importancia de considerar cuidadosamente las características específicas de cada entorno al implementar modelos optimizados. En el caso de la ejecución en SBC, la limitada capacidad del compilador para paralelizar el procesamiento de valores INT8 sobre una arquitectura de 32 bits, común en los SBCs, agrega desafíos adicionales. Esto subraya la necesidad de comprender a fondo las interacciones entre la cuantización, la arquitectura de hardware y el entorno de ejecución para lograr implementaciones eficientes y optimizadas en sistemas particulares.

Tabla 5.2: Consumo de memoria en kilobytes (KB) en diferentes plataformas para distintas variantes de modelos (original, podado y cuantizado) en PyTorch y ONNX.

Consumo de memoria en diferentes plataformas en KB						
Plataforma	**Pytorch (Original)**	**ONNX (Original)**	**Pytorch (Podado)**	**ONNX (Podado)**	**Pytorch (Cuantizado)**	**ONNX (Cuantizado)**
Servidor	808.4	1076.5	519.6	366.9	345.4	148.9
Vim3	979.9	983.3	659.8	368.8	749.4	138.7
RPI4	1259.7	983.5	573.2	373.1	733.1	359.4

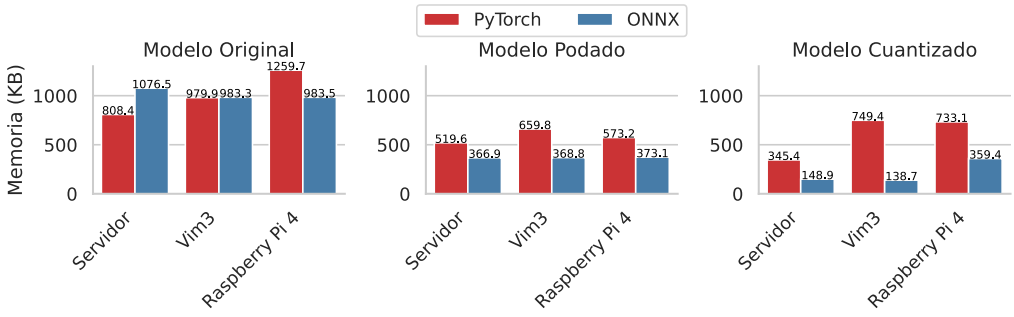


Figura 5.3: Consumo de memoria en diferentes plataformas

5.2.2. Conclusiones

Nuestra exploración de la precisión en función de los porcentajes de poda subraya la superioridad del criterio de Taylor sobre la poda aleatoria como se puede observar en la **Tabla 5.1**. Es importante destacar que la poda aleatoria puede llevar a una pérdida de precisión irreparable debido a que no considera la importancia de los filtros dentro de la red. Observamos mejoras significativas

de rendimiento a través de nuestros esfuerzos de optimización. Centrándonos en el modelo podado al 60 %, nuestra aplicación de la cuantización resultó en ganancias sustanciales de velocidad, logrando una aceleración notable de 8.23× en la plataforma Vim3 y un aumento de 2.51× en la Raspberry Pi 4, en contraste con el modelo original de PyTorch sobre la misma plataforma. Esto muestra los beneficios de eficiencia derivados de la poda y la cuantización.

Además, nuestra evaluación del tiempo de ejecución entre PyTorch y ONNX reveló una diferencia notable, donde es buena idea usar PyTorch para el entrenamiento y ONNX para conversión de modelos e incluso su ejecución. Este enfoque de dos etapas destaca la practicidad de aprovechar ambos frameworks para fases distintas del proceso de optimización.

En cuanto a la utilización de memoria, nuestro análisis mostró el bajo consumo de memoria del modelo podado y cuantizado, con un consumo de memoria a partir de 138.7 KB durante la ejecución en los SBCs. Esto significa una reducción notable de hasta un factor de 7× en comparación con el modelo original.

Capítulo 6

Fusión de NNs

La fusión de redes neuronales implica la combinación de varios modelos neuronales, compartiendo parte de su estructura y algunos parámetros en el para obtener un único modelo que da cuenta de los modelos individuales fusionados. Dados los resultados previos, donde la plataforma Vim3 exhibió un rendimiento superior en cuanto a tiempo y consumo de memoria, hemos tomado la decisión de llevar a cabo los experimentos subsiguientes exclusivamente en el servidor GPU y la Vim3, utilizando el framework ONNX.

6.1. Modelos individuales

En esta sección nos enfocamos en los modelos obtenidos en la sección 4.4, donde desarrollamos y evaluamos modelos dedicados a la detección de cuchillos y armas de fuego. El modelo diseñado para la detección de cuchillos logró alcanzar una precisión del 80 %, mientras que el modelo orientado a la detección de armas alcanzó un destacado 82 % de precisión.

6.2. Modelo fusionado

En el proceso de fusión integramos los modelos dedicados a la detección de armas y cuchillos. Para ambos modelos se conserva toda la etapa de extracción de características, manteniendo la base común que se reentrenó con base en la arquitectura VGG-16. En este enfoque, congelamos la etapa de extracción de características y reentrenamos exclusivamente las capas relacionadas con la clasificación en cada red.

Esta estrategia nos permite fusionar de manera efectiva toda la etapa de

extracción de características, preservando simultáneamente las capas de clasificación específicas de cada modelo. En consecuencia, obtenemos un modelo fusionado con múltiples salidas, aprovechando así las capacidades individuales de detección de eventos relacionados con armas de fuego y cuchillos en un único marco integral. Este enfoque de fusión proporciona una representación conjunta poderosa al combinar las características distintivas de ambos modelos especializados.

6.3. Modelo multiclase

Implementamos un modelo capaz de detectar la presencia de armas de fuego, cuchillos o ninguna de las anteriores en una escena dada. Al igual que en los modelos individuales, nos basamos en la arquitectura VGG-16 como referencia. A través del proceso de TL, reentrenamos el modelo durante 40 épocas utilizando el optimizador SGD, logrando así una precisión del 83 %. Este enfoque de entrenamiento permitió que la red aprendiera patrones representativos de cada clase, resultando en un modelo multiclase efectivo para la detección de eventos relacionados con armas de fuego y cuchillos, con la capacidad adicional de discernir cuando ninguna de estas situaciones está presente. Este modelo se convierte en un punto de referencia fundamental para la comparación con el modelo fusionado, destacando su desempeño en la identificación de clases específicas y su utilidad en situaciones de clasificación multiclase.

6.4. Evaluación de los modelos individuales

La medición del tiempo de ejecución y el consumo de memoria para los modelos individuales se realizó de manera secuencial. Es esencial destacar que esta secuencialidad se implementó con el objetivo de garantizar condiciones comparables y resultados confiables en la evaluación de cada modelo.

La **Figura 6.1** ilustra de manera concisa el proceso de fusión de dos modelos. En el primer paso, se seleccionan los dos modelos destinados a la fusión. Luego, se lleva a cabo un proceso de fusión, identificando y consolidando las partes que son idénticas en ambos modelos. La etapa siguiente implica la conversión posterior del modelo fusionado al formato ONNX, un paso esencial para el despliegue eficiente. Finalmente, se implementa la cuantización para optimizar aún más el modelo. Este flujo de trabajo permite una fusión efectiva y una preparación óptima para la implementación en entornos de recursos limitados.

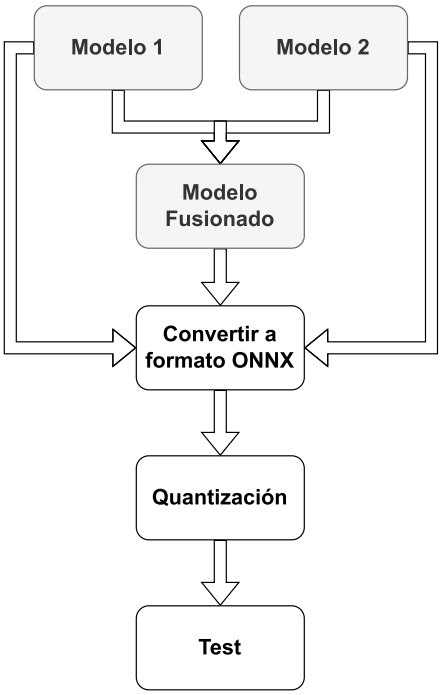


Figura 6.1: Pasos para realizar la fusión de las redes

6.4.1. Resultados de la fusión

Análisis de tiempo de los modelos individuales y fusionado

La **Figura 6.2** proporciona información sobre el tiempo de ejecución de diferentes modelos, incluyendo modelos individuales de armas de fuego y cuchillos, así como un modelo fusionado y el modelo multiclase. Además, se presenta una comparación entre modelos cuantizados y no cuantizados.

Al ejecutar el modelo fusionado (sin cuantizar) en la plataforma Vim3 mediante ONNX se logra un notable tiempo de ejecución de 1.17 segundos. Este rendimiento es significativamente más eficiente en comparación con la ejecución secuencial de los modelos individuales, que requieren un tiempo total de 2.21 segundos para realizar inferencias. Es crucial destacar que el tiempo de ejecución se optimiza aún más al considerar el rendimiento de la red multiclase, donde su tiempo de ejecución es de 1.09 segundos. Este resultado revela que el modelo multiclase supera al modelo fusionado únicamente por 0.08 segundos. El modelo fusionado demuestra ser la opción más eficiente en términos de tiempo de ejecución comparado con los originales, pero no con respecto al modelo multiclase.

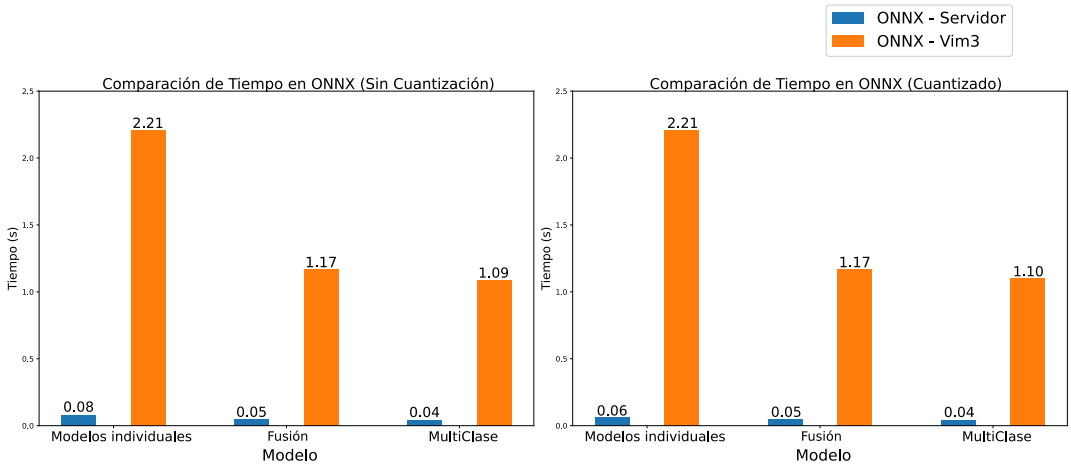


Figura 6.2: Tiempo de ejecución del modelo fusionado

Cuando se realiza la ejecución de los diferentes modelos cuantizados no se logra evidenciar una mejora significativa con respecto al tiempo de ejecución.

Análisis de consumo de memoria de los modelos originales y fusionado

La **Figura 6.3** presenta los resultados del consumo de memoria en nuestros modelos, considerando tanto la ejecución individual de modelos como la ejecución de modelos fusionados y el modelo multiclase. Este análisis se realiza en las plataformas servidor GPU y VIM3 con el framework ONNX.

Cuando evaluamos el modelo fusionado (sin cuantizar) en la plataforma Vim3 observamos un consumo de memoria de 1927.92 KB. Esto contrastado con el consumo de memoria total de los modelos individuales de armas de fuego y cuchillos que es de 1951.4 KB sobre la misma plataforma.

En el caso del modelo fusionado cuantizado, el consumo de memoria es de 1570.9 KB en la plataforma Vim3. Esto es favorable en comparación con el consumo total de memoria de los modelos individuales cuantizados de armas de fuego y cuchillos, que asciende a 1980.5 KB. Una vez más, el modelo fusionado cuantizado demuestra ser más eficiente en cuanto a consumo de memoria en comparación con ejecutar modelos separados. Por otro lado, es importante destacar que en este caso el modelo multiclase se muestra más eficiente en

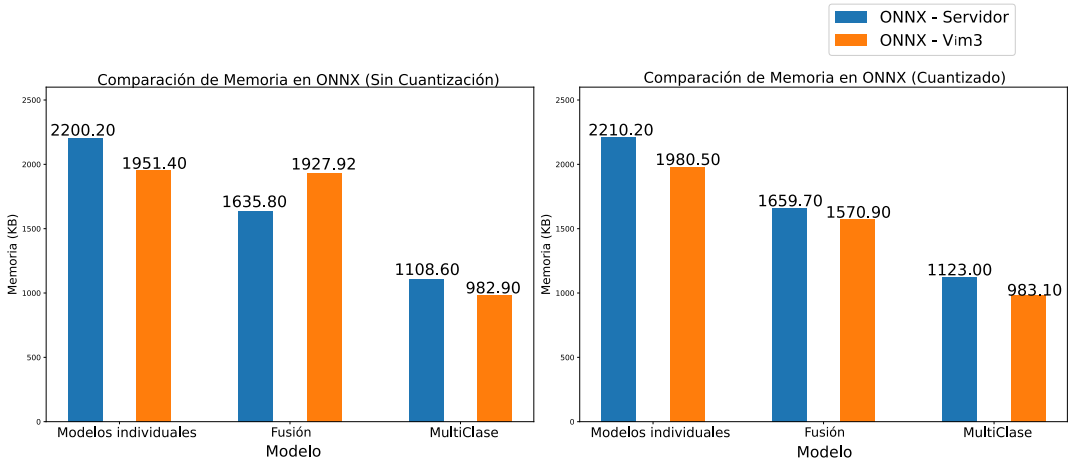


Figura 6.3: Consumo de memoria del modelo fusionado

términos de consumo de memoria. Para el modelo sin cuantizar, se obtiene un consumo de memoria de 982.9 KB. Este valor es similar para el modelo cuantizado sobre la misma plataforma. Cuando se realiza la ejecución de los modelos cuantizados y sin cuantizar se logra evidenciar que no hay una mejora significativa excepto para el caso del modelo fusionado para la Vim3. Esto puede deberse a que durante el proceso de conversión a ONNX es necesario realizar algunos ajustes internos para representar las diferentes salidas del modelo fusionado. Sin embargo, cuando se lleva a cabo el proceso de cuantización, estos ajustes se ven reducidos y, en consecuencia, la diferencia de consumo de memoria disminuye.

6.4.2. Observaciones sobre la fusión

El análisis de los tiempos de ejecución y el consumo de memoria en nuestros modelos de detección de armas de fuego y cuchillos, así como en el modelo fusionado, arroja resultados significativos y valiosos. Estos resultados son fundamentales para comprender la eficiencia y la optimización en la implementación de estos modelos en diferentes plataformas y frameworks.

En primer lugar, nuestros hallazgos destacan la importancia de la fusión de modelos para aplicaciones de detección de objetos múltiples, como armas de fuego y cuchillos. El modelo fusionado, que combina ambas categorías, demues-

tra un notable beneficio en términos de tiempo de ejecución en comparación con la ejecución de modelos individuales. En la plataforma Vim3, por ejemplo, el modelo fusionado logra un tiempo de ejecución de 1.17 segundos, en contraste con los 2.21 segundos requeridos para ejecutar los modelos individuales. Sin embargo, es importante señalar que el modelo multiclase aún supera al modelo fusionado por un margen reducido, con un tiempo de ejecución de 0.08 segundos en la plataforma Vim3.

En cuanto al consumo de memoria, observamos que el modelo fusionado consume igual memoria en comparación con el modelo individual. Sin embargo, esta inversión en memoria se traduce en una mejora significativa en el rendimiento de tiempo de ejecución. Es crucial destacar que la elección de framework también desempeña un papel vital en el consumo de memoria y el rendimiento.

Estos resultados respaldan la viabilidad y la eficacia de la fusión de modelos en aplicaciones de detección de objetos. Es importante destacar que estos resultados no afectaron la precisión de cada modelo individual. El modelo multiclase mantiene su precisión en un 83 %, mientras que el modelo fusionado logra una precisión promedio del 81 %, calculada como el promedio de las precisiones de cada una de sus salidas, ya que estas no se vieron afectadas cuando se realizó la fusión.

Capítulo 7

Combinación de técnicas de compresión y fusión de NNs

En capítulo analizaremos en detalle cómo la compresión de redes neuronales se convierte en un habilitador fundamental para la implementación exitosa de modelos fusionados. La compresión se presenta como un paso crucial en la ruta hacia modelos de detección de objetos múltiples más ágiles y eficientes mediante la optimización de recursos computacionales y la reducción de parámetros.

7.1. Aplicación combinada de la compresión y la fusión

Con el objetivo de evaluar la combinación de técnicas de compresión y fusión de modelos, se llevó a cabo inicialmente un proceso de compresión de CNNs. Esta compresión se orientó hacia la futura fusión de modelos, centrándose en identificar los filtros que serían seleccionados para su eliminación según el criterio de Taylor. El proceso de compresión se ejecutó de la siguiente manera:

- **Selección de filtros para eliminación:** inicialmente se aplicó el criterio de Taylor sobre las CNNs destinadas a la identificación de armas de fuego y cuchillos con el objeto de determinar qué filtros deberían ser seleccionados para su eliminación. Solo se consideraron para la eliminación aquellos filtros que fueron seleccionados por el criterio de Taylor en ambas CNNs.
- **Proceso de poda iterativo:** se realizó un proceso de poda iterativo donde se eliminaron los filtros que cumplían con los criterios mencionados

en el paso anterior. Este proceso se repitió hasta alcanzar el porcentaje de poda deseado o hasta que ya no se encontraron más filtros idénticos en ambas redes.

Una vez completado el proceso de compresión, se procedió a la fase de fusión de modelos, que se llevó a cabo utilizando la técnica de fusión a través de filtros (ver **Figura 7.1**). En este proceso de fusión, se combinaron los filtros seleccionados de ambas redes para crear un modelo fusionado. Los filtros se fusionaron siguiendo las estrategias previamente descritas, conservando los pesos todos los valores de los filtros de uno de los modelos o conservando el valor máximo, mínimo o promedio de cada filtro.

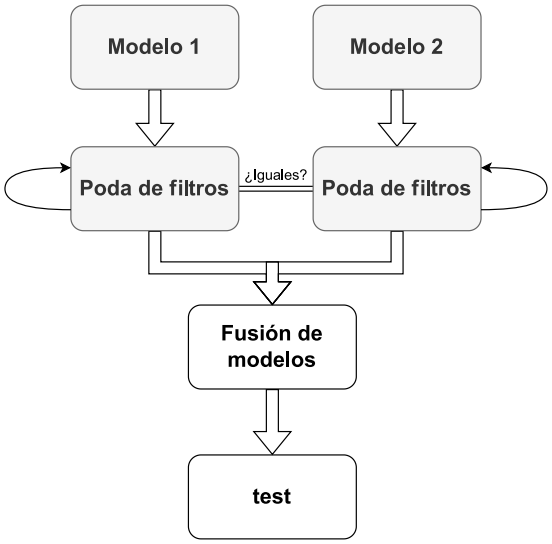


Figura 7.1: Pasos para realizar la fusión de la red teniendo en cuenta la poda

7.2. Modelos individuales

En la **Tabla 7.1** se presenta una comparación del porcentaje de filtros conservados en los modelos de detección de armas y cuchillos, junto con las precisiones correspondientes en cada caso. El propósito de esta tabla es determinar en qué punto de poda ambos modelos alcanzan su mejor desempeño en términos de precisión, para posteriormente aplicar las técnicas de fusión descritas.

Se destaca que al conservar el 44 % de los filtros en ambas redes, el modelo de cuchillos solo experimenta una degradación mínima del 1 % con respecto

Tabla 7.1: Porcentaje de filtros conservados vs. Precisión para los modelos de armas y cuchillos

Porcentaje de filtros conservados vs. Precisión para los dos modelos					
Porcentaje de filtros conservados	**100 %**	**87.7 %**	**66.4 %**	**44.0 %**	**22.7 %**
Acc. Modelo armas	82 %	87 %	88 %	86 %	69 %
Acc. Modelo cuchillos	80 %	75 %	72 %	79 %	69 %

a su versión original, que tenía una precisión del 80 %. Por otro lado, el modelo de armas muestra un aumento del 4 % en su precisión con respecto al modelo original, que tenía un 82 % de precisión. Estos resultados indican que el nivel de poda del 44 % es el punto óptimo para ambos modelos, ya que permite una reducción significativa en la complejidad del modelo sin comprometer sustancialmente su rendimiento en la tarea de detección. Estos modelos sirvieron como punto de partida para la aplicación de diversas técnicas de fusión. Además, se observa un comportamiento fluctuante en el rendimiento del modelo de cuchillos en comparación con el modelo de armas. Esta variabilidad podría estar relacionada con la disparidad en el tamaño de las bases de datos utilizadas para entrenar ambos modelos, siendo la base de datos del modelo de armas significativamente más extensa que la correspondiente al modelo de cuchillos. Esta diferencia en el volumen de datos podría afectar la precisión del modelo de cuchillos, especialmente durante el proceso de poda.

7.3. Fusión de modelos a través de filtros

En esta sección se presentan diversos métodos que se centran en la fusión de filtros en CNNs, siguiendo la aproximación de un conjunto de características para múltiples tareas. Este enfoque ofrece una visión integral de cómo la fusión de filtros puede potenciar la capacidad de las CNNs.

7.3.1. Conservar todos los pesos de un modelo

En este enfoque, se conservan todos los pesos de uno de los modelos individuales, mientras que los filtros de los otros modelos se descartan. Esto implica que el modelo resultante tendrá la misma arquitectura que el modelo conservado, lo que puede ser beneficioso si ese modelo tiene un rendimiento particularmente fuerte en ciertas categorías de objetos.

7.3.2. Conservar los pesos con el valor máximo de cada filtro

En este caso, para cada filtro, se conserva el peso con el valor máximo entre los modelos individuales. Esto significa que el modelo resultante enfatizará las características detectadas con mayor confianza por los modelos originales. Puede ayudar a mejorar la precisión en la detección de objetos prominentes.

7.3.3. Conservar los pesos con el valor mínimo de cada filtro

Al contrario del enfoque anterior, aquí se conserva el peso con el valor mínimo de cada filtro entre los modelos fusionados. Este enfoque puede ser útil cuando se desean mantener las características menos evidentes pero aún relevantes detectadas por los modelos individuales.

7.3.4. Obtener el promedio de cada filtro

En este enfoque, se calcula el valor promedio de los pesos de cada filtro entre los modelos individuales. Esto resulta en un modelo que busca un equilibrio entre las características detectadas por todos los modelos originales. Puede ser beneficioso cuando se busca un rendimiento generalizado y robusto en la detección de objetos.

La **Tabla 7.2** presenta una comparación de métricas de precisión para diferentes enfoques de fusión en los modelos de armas de fuego y cuchillos. En la fila "Fusión", se describen los enfoques de fusión explorados, que incluyen el "promedio", donde se calcula el promedio de los filtros correspondientes de ambos modelos; "máximos", que conserva los valores máximos de cada par de filtros correspondientes de ambos modelos; "mínimos", que conserva los valores mínimos de cada par de filtros correspondientes de ambos modelos; "Filtros armas", que conserva todos los filtros del modelo de armas de fuego; y "Filtros cuchillos", que conserva todos los filtros del modelo de cuchillos. Las filas "Acc. armas" y "Acc. cuchillos" muestran la precisión del modelo cuando se evalúa con respecto al dataset de prueba para cada modelo. Estos valores se expresan como porcentajes y representan la precisión de los modelos en la detección de armas y cuchillos, respectivamente. Se observa que el modelo fusionado, que conserva todos los filtros del modelo de armas de fuego, logra la mejor precisión promedio en la detección de cuchillos y armas de fuego en comparación con los

Tabla 7.2: Comparación de métricas de precisión para diferentes enfoques de fusión en los modelos de armas y cuchillos.

Porcentaje de filtros conservados vs. Precisión para los dos modelos					
Fusión	**Promedio**	**Máximos**	**Mínimos**	**Filtros cuchillos**	**Filtros armas**
Acc. armas	72 %	51 %	51 %	55 %	86 %
Acc. cuchillos	75 %	56 %	56 %	79 %	64 %

otros enfoques de fusión. La precisión promedio se calcula como el promedio de las precisiones individuales, proporcionando una medida de la capacidad del modelo para abordar múltiples clases de objetos. Cuando se mide la salida del modelo fusionado con respecto al dataset de prueba para cuchillos, se obtiene que el modelo logra una precisión del 64 %. Esto indica que después de realizar la compresión y la fusión del modelo la precisión se ha degradado en un 16 % en comparación con el modelo individual de cuchillos. Sin embargo, al evaluar la salida de este mismo modelo fusionado con respecto al dataset de prueba para armas de fuego, se logra conservar la precisión de 86 %. Dado este rendimiento superior en términos de precisión, se selecciona este enfoque como el candidato principal para las siguientes mediciones de tiempo de ejecución y consumo de memoria en las plataformas servidor GPU y Vim3.

7.4. Análisis de tiempo y consumo de memoria de los modelos individuales, fusionado y multiclase

En las **Figuras 7.2 y 7.3**, se puede apreciar que no hay un cambio significativo en el tiempo de ejecución y el consumo de memoria cuando el modelo se ejecuta en diferentes plataformas y si este es cuantizado o no. Sin embargo, se destaca una diferencia cuando se ejecuta el modelo comprimido-fusionado en comparación con los modelos individuales. Específicamente, al ejecutar el modelo comprimido-fusionado (sin cuantizar) en la Vim3 se logra un tiempo de ejecución de 0.32 segundos y un consumo de memoria de 348.19 KB. En contraste, al ejecutar los dos modelos originales de manera simultanea en las mismas condiciones se requieren 0.62 segundos y un consumo de memoria de 1131.71 KB. Estos resultados resaltan la eficiencia del modelo comprimido y fusionado en términos de tiempo de ejecución y uso de memoria en comparación con la ejecución secuencial de los modelos individuales.

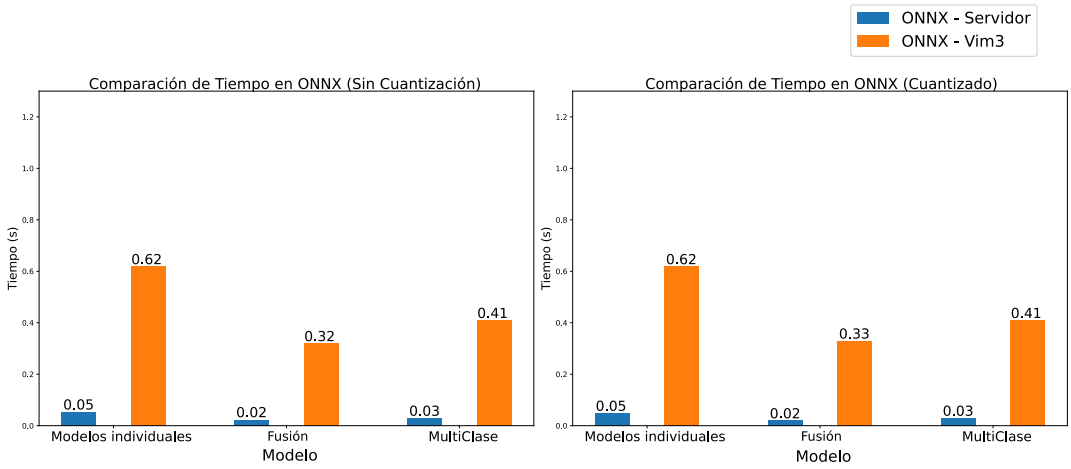


Figura 7.2: Tiempo de ejecución del modelo comprimido - fusionado

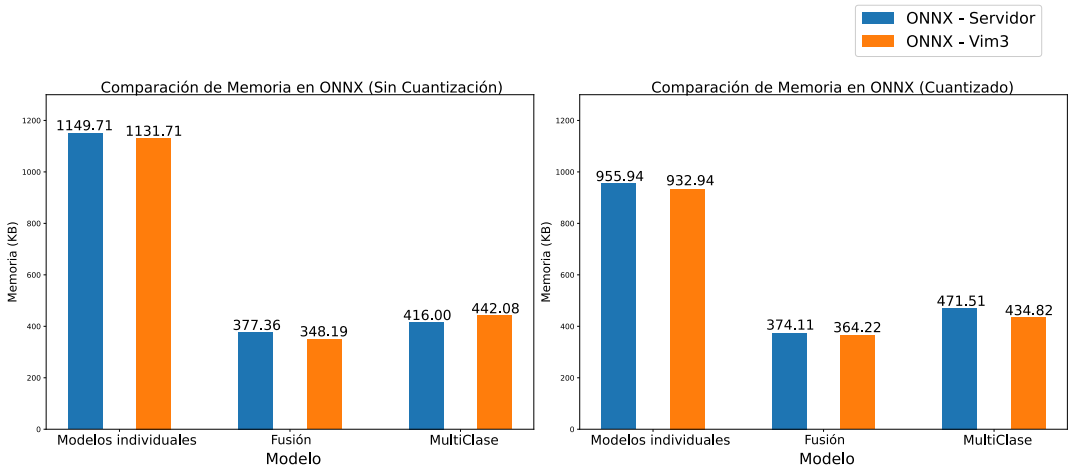


Figura 7.3: Consumo de memoria del modelo comprimido - fusionado

7.5. Observaciones sobre la combinación de la compresión y la fusión

Los resultados derivados de la comparación de precisión frente al porcentaje de filtros conservados en los modelos de detección de armas de fuego y

cuchillos indican que conservar el 44 % de los filtros en ambas redes es un punto óptimo. Este nivel de conservación proporciona una reducción sustancial en la complejidad del modelo sin sacrificar significativamente su rendimiento en la tarea de detección. Es importante resaltar que la fusión de filtros, especialmente al conservar todos los filtros del modelo de armas, conduce a una mejora notable en la precisión para la detección de objetos en comparación con otros enfoques de fusión.

A pesar de la ligera disminución en la precisión, la implementación de modelos comprimidos y fusionados exhibe una eficiencia destacada en términos de tiempo de ejecución y consumo de memoria, especialmente al ejecutarse en la plataforma Vim3. Además, al comparar el modelo fusionado con el modelo multiclase, se observa que el modelo fusionado supera al multiclase en términos de precisión. Este resultado resalta la eficacia del modelo fusionado en comparación con los modelos individuales y sugiere que se acerca a la precisión del modelo multiclase.

Estos hallazgos evidencian que el modelo fusionado no solo obtiene mejores resultados en comparación con los modelos individuales, sino que también se aproxima a la precisión del modelo multiclase.

Capítulo 8

Conclusiones y trabajo futuro

Con los resultados de esta investigación se demuestra que la implementación de técnicas de optimización de CNNs es una estrategia clave para la construcción de modelos cuando se tienen recursos limitados. La poda de modelos utilizando el criterio de Taylor demostró ser una técnica de compresión eficiente. Eliminar hasta el 60 % de los filtros de la red resultó ser el punto óptimo para reducir la complejidad del modelo sin sacrificar la precisión en la detección de armas de fuego. Por otro lado, cuando se realiza la poda conjunta pensando en la fusión futura se evidencia que conservar el 44 % de los filtros de cada red es un punto óptimo en el que no se afecta de forma significativa la la precisión de las redes. Otra estrategia clave fue la cuantización de modelos, que implicó la representación de parámetros en una cantidad inferior de bits. Esta técnica proporcionó mejoras notables en la velocidad de inferencia, con una aceleración de hasta 8.23× en diferentes plataformas, como la Vim3 y la Raspberry Pi 4. Además, se comparó el rendimiento entre los frameworks PyTorch y ONNX, destacando que ONNX resultó ser especialmente eficiente para la ejecución de modelos ya cuantizados.

La fusión de modelos fue una fase crucial en este estudio. Se exploraron varios enfoques de fusión y se descubrió que conservar todos los filtros del modelo de armas ofrecía la mejor precisión en la detección de estos dos eventos relacionados a la seguridad ciudadana. Esta fusión permitió la detección simultánea de armas de fuego y cuchillos con un solo modelo, simplificando considerablemente la implementación.

En términos de eficiencia y rendimiento, se realizaron mediciones de tiempo de ejecución y consumo de memoria en diferentes plataformas y frameworks. Los resultados demostraron que la implementación de modelos comprimidos y fusionados ofrecía una eficiencia notable, especialmente cuando se ejecutaban

en la plataforma Vim3 con ONNX como framework. Aunque hubo una ligera disminución en la precisión debido a la compresión y la fusión, esta degradación se consideró razonable dada la mejora en la eficiencia.

Esta investigación proporciona una base para el desarrollo redes adaptadas dependiendo la necesidad para la detección de eventos relacionado a la seguridad ciudadana basados en la IA, destacando la importancia de estrategias de adaptación, poda, cuantización y fusión de modelos para lograr un equilibrio efectivo entre eficiencia y precisión en la detección de eventos de seguridad en entornos urbanos, cuando los recursos computacionales y de memoria sean restringidos.

Una de las áreas clave para futuras investigaciones es la mejora de la eficiencia en la fusión de modelos. Actualmente nuestro enfoque se basa en métodos relativamente simples, como el promedio de valores o la selección de máximos y mínimos para combinar información de modelos. No obstante, existe un potencial significativo para lograr una optimización más eficaz de esta fusión mediante la implementación de criterios de similitud entre los filtros. Otra dirección prometedora implicaría la aplicación de aprendizaje por refuerzo, que sería una estrategia valiosa para determinar de forma automática parámetros de fusión óptimas. Al proporcionar recompensas basadas en el rendimiento de detección, nuestro sistema podría aprender a tomar decisiones sobre qué modelo o fuente de información priorizar según el contexto y los datos de entrada, lo que podría conducir a una adaptación dinámica de la fusión de modelos en entornos variables.

Si bien nuestro enfoque actual se ha centrado en la detección de armas de fuego y cuchillos, existe un amplio espectro de eventos relacionados con la seguridad que podrían ser igualmente críticos para la seguridad pública y la protección de la personas. Ampliar nuestro trabajo a la detección de estos eventos presenta oportunidades interesantes. Esta expansión nos permitiría no solo abordar una variedad más amplia de amenazas potenciales, como accidentes de transito, incendios o dispositivos sospechosos, sino también lograr un sistema más dinámico y adaptable. Esto es esencial ya que diferentes entornos y situaciones pueden requerir enfoques de seguridad específicos. Imaginemos una situación en la que se necesita una mayor atención a la detección de incendios en un evento público masivo. En este escenario, el sistema podría cambiar automáticamente su enfoque hacia la detección de incendios, priorizando la información y los modelos relacionados con esta amenaza específica.

Referencias

[1] R. Nambiar, R. Shroff y S. Handy, «Smart cities: Challenges and opportunities», en *2018 10th international conference on communication systems & networks (COMSNETS)*, IEEE, 2018, págs. 243-250.

[2] ONU, *Objetivos y metas de desarrollo sostenible - Desarrollo Sostenible*, 2015. dirección: `https://www.un.org/sustainabledevelopment/es/sustainable-development-goals/`.

[3] OECD, *OECD Better Life Index*, 2020. dirección: `https://www.oecdbetterlifeindex.org/topics/safety/`.

[4] OSC, *Índice de ciudades modernas*, 2021. dirección: `https://osc.dnp.gov.co/media/com_inicio/img/presentacion-ICM-2021.pdf`.

[5] A. M. Ardila Felacio, M. J. Quiñones Corrales et al., «La convivencia en las políticas públicas: análisis de los instrumentos de planeación pública en el departamento de Antioquia, el área Metropolitana del Valle de Aburrá y el Distrito de Ciencia, Tecnología e Innovación de Medellín 2016-2022», Tesis doct., Universidad EAFIT, 2023.

[6] C. X. Mavromoustakis, G. Mastorakis y C. Dobre, *Advances in mobile cloud computing and big data in the 5G era*. Springer, 2017.

[7] K. Cao, Y. Liu, G. Meng y Q. Sun, «An overview on edge computing research», *IEEE access*, vol. 8, págs. 85 714-85 728, 2020.

[8] D. Evans, «The internet of things», *How the Next Evolution of the Internet is Changing Everything, Whitepaper, Cisco Internet Business Solutions Group (IBSG)*, vol. 1, págs. 1-12, 2011.

[9] 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, n.º 5, págs. 493-515, 2010.

[10] K. Yu, L. Jia, Y. Chen, W. Xu et al., «: yesterday, today, and tomorrow», *Journal of computer Research and Development*, vol. 50, n.º 9, págs. 1799-1804, 2013.

[11] N. Rusk, *Nature Methods*, vol. 13, n.º 1, págs. 35-35, 2016.

[12] Y.-M. Chou, Y.-M. Chan, J.-H. Lee, C.-Y. Chiu y C.-S. Chen, «Unifying and merging well-trained deep neural networks for inference stage», *arXiv preprint arXiv:1805.04980*, 2018.

[13] Y. He, J. Lin, Z. Liu, H. Wang, L.-J. Li y S. Han, «Amc: Automl for model compression and acceleration on mobile devices», en *Proceedings of the European conference on computer vision (ECCV)*, 2018, págs. 784-800.

[14] S. J. Johnston, P. J. Basford, C. S. Perkins et al., «Commodity single board computer clusters and their applications», *Future Generation Computer Systems*, vol. 89, págs. 201-212, 2018.

[15] N. Nasaruddin, K. Muchtar, A. Afdhal y A. P. J. Dwiyantoro, «Deep anomaly detection through visual attention in surveillance videos», *Journal of Big Data*, vol. 7, n.º 1, págs. 1-17, 2020.

[16] W. Sultani, C. Chen y M. Shah, «Real-world anomaly detection in surveillance videos», en *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, págs. 6479-6488.

[17] F. Landi, C. G. Snoek y R. Cucchiara, «Anomaly locality in video surveillance», *arXiv preprint arXiv:1901.10364*, 2019.

[18] Q. Xu, J. See y W. Lin, «Localization guided fight action detection in surveillance videos», en *2019 IEEE International Conference on Multimedia and Expo (ICME)*, IEEE, 2019, págs. 568-573.

[19] T. Hassner, Y. Itcher y O. Kliper-Gross, «Violent flows: Real-time detection of violent crowd behavior», en *2012 IEEE computer society conference on computer vision and pattern recognition workshops*, IEEE, 2012, págs. 1-6.

[20] A. Datta, M. Shah y N. D. V. Lobo, «Person-on-person violence detection in video data», en *2002 International Conference on Pattern Recognition*, IEEE, vol. 1, 2002, págs. 433-438.

[21] R. Olmos, S. Tabik y F. Herrera, «Automatic handgun detection alarm in videos using deep learning», *Neurocomputing*, vol. 275, págs. 66-72, 2018.

[22] K. Simonyan y A. Zisserman, «Very deep convolutional networks for large-scale image recognition», *arXiv preprint arXiv:1409.1556*, 2014.

[23] R. Reddy, K. G. Vallabh y S. Sharan, «Multiclass weapon detection using multi contrast convolutional neural networks and faster region-based convolutional neural networks», en *2021 2nd International Conference for Emerging Technology (INCET)*, IEEE, 2021, págs. 1-8.

[24] G. K. Verma y A. Dhillon, «A handheld gun detection using faster r-cnn deep learning», en *Proceedings of the 7th international conference on computer and communication technology*, 2017, págs. 84-88.

[25] *Internet Movie Firearms Database - Guns in Movies, TV and Video Games*. dirección: `https://www.imfdb.org`.

[26] V. Kaya, S. Tuncer y A. Baran, «Detection and classification of different weapon types using deep learning», *Applied Sciences*, vol. 11, n.º 16, pág. 7535, 2021.

[27] K. He, X. Zhang, S. Ren y J. Sun, *Deep Residual Learning for Image Recognition*, 2015. arXiv: `1512.03385 [cs.CV]`.

[28] B. Koonce y B. Koonce, «ResNet 50», *Convolutional Neural Networks with Swift for Tensorflow: Image Recognition and Dataset Categorization*, págs. 63-72, 2021.

[29] S. Ahmed, M. T. Bhatti, M. G. Khan, B. Lövström y M. Shahid, «Development and optimization of deep learning models for weapon detection in surveillance videos», *Applied Sciences*, vol. 12, n.º 12, pág. 5772, 2022.

[30] D. Berardini, L. Migliorelli, A. Galdelli, E. Frontoni, A. Mancini y S. Moccia, «A deep-learning framework running on edge devices for handgun and knife detection from indoor video-surveillance cameras», *Multimedia Tools and Applications*, págs. 1-19, 2023.

[31] C.-T. Liu, T.-W. Lin, Y.-H. Wu et al., «Computation-performance optimization of convolutional neural networks with redundant filter removal», *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 66, n.º 5, págs. 1908-1921, 2018.

[32] D. Gope, G. Dasika y M. Mattina, «Ternary hybrid neural-tree networks for highly constrained iot applications», *Proceedings of Machine Learning and Systems*, vol. 1, págs. 190-200, 2019.

[33] M. Tschannen, A. Khanna y A. Anandkumar, «StrassenNets: Deep learning with a multiplication budget», en *International Conference on Machine Learning*, PMLR, 2018, págs. 4985-4994.

[34] P. Shi, F. Gao, S. Liang y S. Yu, «Multi-model inference acceleration on embedded multi-core processors», en *2020 International Conference on Intelligent Computing and Human-Computer Interaction (ICHCI)*, IEEE, 2020, págs. 400-403.

[35] T. Wu, X. Li, D. Zhou, N. Li y J. Shi, «Differential evolution based layer-wise weight pruning for compressing deep neural networks», *Sensors*, vol. 21, n.º 3, pág. 880, 2021.

[36] J.-H. Choi y J.-S. Lee, «EmbraceNet: A robust deep learning architecture for multimodal classification», *Information Fusion*, vol. 51, págs. 259-270, 2019.

[37] R. F. Rachmadi, I. K. E. Purnama, S. M. S. Nugroho e Y. K. Suprapto, «Image-based Kinship verification using fusion convolutional neural network», en *2019 IEEE 11th International Workshop on Computational Intelligence and Applications (IWCIA)*, IEEE, 2019, págs. 59-65.

[38] J. Williams, R. Comanescu, O. Radu y L. Tian, «Dnn multimodal fusion techniques for predicting video sentiment», en *Proceedings of grand challenge and workshop on human multimodal language (Challenge-HML)*, 2018, págs. 64-72.

[39] A. A. Mallouh, Z. Qawaqneh y B. D. Barkana, «Combining two different DNN architectures for classifying speaker's age and gender», en *International Conference on Bio-inspired Systems and Signal Processing*, SCITEPRESS, vol. 5, 2017, págs. 112-117.

[40] Y.-M. Chou, Y.-M. Chan, J.-H. Lee, C.-Y. Chiu y C.-S. Chen, «Unifying and merging well-trained deep neural networks for inference stage», *arXiv preprint arXiv:1805.04980*, 2018.

[41] R. K. R. Kummitha, «Smart cities and entrepreneurship: An agenda for future research», *Technological Forecasting and Social Change*, vol. 149, pág. 119 763, 2019.

[42] T. Song, J. Cai, T. Chahine y L. Li, «Towards smart cities by Internet of Things (IoT)—a silent revolution in China», *Journal of the Knowledge Economy*, vol. 12, págs. 1-17, 2021.

[43] D. Washburn y U. Sindhu, *Making Leaders Successful Every Day Helping CIOs Understand "Smart City" Initiatives*, 2010.

[44] D. US, *Define Your Smart City Strategy.* dirección: `https://www2.deloitte.com/us/en/pages/consulting/solutions/smart-cities-strategies.html`.

[45] M. Deakin, *Smart cities: governing, modelling and analysing the transition*. Routledge, 2013.

[46] X. Li, D. Guo, H. Yin y G. Wei, «The public safety wireless broadband network with airdropped sensors», en *2015 IEEE China Summit and International Conference on Signal and Information Processing (ChinaSIP)*, IEEE, 2015, págs. 443-447.

[47] S. Zhu, C. Chen y W. Sultani, «Video anomaly detection for smart surveillance», en *Computer Vision: A Reference Guide*, Springer, 2020, págs. 1-8.

[48] B. Rinner y W. Wolf, «An introduction to distributed smart cameras», *Proceedings of the IEEE*, vol. 96, n.º 10, págs. 1565-1575, 2008.

[49] S. N. Sinha, «Pan-Tilt-Zoom (PTZ) Camera», en *Computer Vision: A Reference Guide*, Springer, 2021, págs. 941-947.

[50] T. Marques, L. Lukic y J. Gaspar, «Observation functions in an information theoretic approach for scheduling pan-tilt-zoom cameras in multitarget tracking applications», en *Robot 2015: Second Iberian Robotics Conference: Advances in Robotics, Volume 2*, Springer, 2016, págs. 503-515.

[51] T. Zhang, A. Chowdhery, P. Bahl, K. Jamieson y S. Banerjee, «The design and implementation of a wireless video surveillance system», en *Proceedings of the 21st Annual International Conference on Mobile Computing and Networking*, 2015, págs. 426-438.

[52] J. Fernández, L. Calavia, C. Baladrón et al., «An intelligent surveillance platform for large metropolitan areas with dense sensor deployment», *Sensors*, vol. 13, n.º 6, págs. 7414-7442, 2013.

[53] IBM. «What is Artificial Intelligence (AI)?» (2023), dirección: `https://www.ibm.com/topics/artificial-intelligence`.

[54] S. Brown. «Machine learning, explained». (abr. de 2021), dirección: `https://mitsloan.mit.edu/ideas-made-to-matter/machine-learning-explained`.

[55] L. Deng, D. Yu et al., «: methods and applications», *Foundations and trends® in signal processing*, vol. 7, n.º 3–4, págs. 197-387, 2014.

[56] I. Goodfellow, Y. Bengio y A. Courville, *Deep learning*. MIT press, 2016.

[57] C. M. Bishop, *Neural networks for pattern recognition*. Oxford university press, 1995.

[58] J. Zhong y W. Li, «Predicting customer churn in the telecommunication industry by analyzing phone call transcripts with convolutional neural networks», en *Proceedings of the 2019 3rd International Conference on Innovation in Artificial Intelligence*, 2019, págs. 55-59.

[59] R. Reed y R. J. MarksII, *Neural smithing: supervised learning in feedforward artificial neural networks*. Mit Press, 1999.

[60] T. Dillon, C. Wu y E. Chang, «Cloud computing: issues and challenges», en *2010 24th IEEE international conference on advanced information networking and applications*, Ieee, 2010, págs. 27-33.

[61] S. Yi, C. Li y Q. Li, «A survey of fog computing: concepts, applications and issues», en *Proceedings of the 2015 workshop on mobile big data*, 2015, págs. 37-42.

[62] W. Z. Khan, E. Ahmed, S. Hakak, I. Yaqoob y A. Ahmed, «Edge computing: A survey», *Future Generation Computer Systems*, vol. 97, págs. 219-235, 2019.

[63] C. Alippi, S. Disabato y M. Roveri, «Moving convolutional neural networks to embedded systems: the alexnet and VGG-16 case», en *2018 17th ACM/IEEE International Conference on Information Processing in Sensor Networks (IPSN)*, IEEE, 2018, págs. 212-223.

[64] J. S. Denker, «Optimal brain damage», *Advances in neural information processing systems*, vol. 2, págs. 598-605, 1989.

[65] B. Hassibi, «D. Stork," Second order derivatives for network pruning: optimal brain surgeon», *Advances in Neural Informa*, 1993.

[66] P. Molchanov, S. Tyree, T. Karras, T. Aila y J. Kautz, «Pruning convolutional neural networks for resource efficient inference», *arXiv preprint arXiv:1611.06440*, 2016.

[67] Y. Guo, «A survey on methods and theories of quantized neural networks», *arXiv preprint arXiv:1808.04752*, 2018.

[68] P. Panda, A. Sengupta y K. Roy, «Conditional deep learning for energy-efficient and enhanced pattern recognition», en *2016 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, IEEE, 2016, págs. 475-480.

[69] S. Teerapittayanon, B. McDanel y H.-T. Kung, «Branchynet: Fast inference via early exiting from deep neural networks», en *2016 23rd international conference on pattern recognition (ICPR)*, IEEE, 2016, págs. 2464-2469.

[70] S. Teerapittayanon, B. McDanel y H.-T. Kung, «Distributed deep neural networks over the cloud, the edge and end devices», en *2017 IEEE 37th international conference on distributed computing systems (ICDCS)*, IEEE, 2017, págs. 328-339.

[71] J. Lee, B. Varghese, R. Woods y H. Vandierendonck, «Tod: Transprecise object detection to maximise real-time accuracy on the edge», en *2021 IEEE 5th International Conference on Fog and Edge Computing (ICFEC)*, IEEE, 2021, págs. 53-60.

[72] W. Lou, L. Xun, A. Sabet, J. Bi, J. Hare y G. V. Merrett, «Dynamic-OFA: Runtime DNN architecture switching for performance scaling on heterogeneous embedded platforms», en *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021, págs. 3110-3118.

[73] J.-H. Choi y J.-S. Lee, «EmbraceNet: A robust deep learning architecture for multimodal classification», *Information Fusion*, vol. 51, págs. 259-270, 2019.

[74] R. F. Rachmadi, I. K. E. Purnama, S. M. S. Nugroho e Y. K. Suprapto, «Image-based Kinship verification using fusion convolutional neural network», en *2019 IEEE 11th International Workshop on Computational Intelligence and Applications (IWCIA)*, IEEE, 2019, págs. 59-65.

[75] J. Ngiam, A. Khosla, M. Kim, J. Nam, H. Lee y A. Y. Ng, «Multimodal deep learning», en *Proceedings of the 28th international conference on machine learning (ICML-11)*, 2011, págs. 689-696.

[76] Y. Wang, W. Huang, F. Sun, T. Xu, Y. Rong y J. Huang, «Deep multimodal fusion by channel exchanging», *Advances in neural information processing systems*, vol. 33, págs. 4835-4845, 2020.

[77] Y. Zhang, D. Sidibé, O. Morel y F. Mériaudeau, «Deep multimodal fusion for semantic image segmentation: A survey», *Image and Vision Computing*, vol. 105, pág. 104 042, 2021.

[78] D. Stefani, S. Peroni, L. Turchet et al., «A comparison of deep learning inference engines for embedded real-time audio classification», en *Proceedings of the International Conference on Digital Audio Effects, DAFx*, MDPI (Multidisciplinary Digital Publishing Institute), vol. 3, 2022, págs. 256-263.

[79] A. Krizhevsky, I. Sutskever y G. E. Hinton, «Imagenet classification with deep convolutional neural networks», *Communications of the ACM*, vol. 60, n.º 6, págs. 84-90, 2017.

[80] M. Hassan, *VGG16 – Convolutional Network for Classification and Detection*, "https://neurohive.io/en/popular-networks/vgg16/", 2018.

[81] F. Pedregosa, G. Varoquaux, A. Gramfort et al., «Scikit-learn: Machine Learning in Python», *Journal of Machine Learning Research*, vol. 12, págs. 2825-2830, 2011.

[82] C. Lu, J. Shi y J. Jia, «Abnormal Event Detection at 150 fps in Matlab», en *IEEE International Conference on Computer Vision (ICCV)*, 2013.

[83] U. of Minnesota. «Unusual Crowd Activity Dataset». ().

[84] F.-H. Chan, Y.-T. Chen, Y. Xiang y M. Sun, «Anticipating Accidents in Dashcam Videos», en *Asian Conference on Computer Vision*, Springer, 2016, págs. 136-153.

[85] W. Liu, D. Lian, W. Luo y S. Gao, «Future Frame Prediction for Anomaly Detection – A New Baseline», en *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018.

[86] W. Sultani, C. Chen y M. Shah, «Real-World Anomaly Detection in Surveillance Videos», en *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, págs. 6479-6488.

[87] B. Ramachandra y M. Jones, «Street Scene: A New Dataset and Evaluation Protocol for Video Anomaly Detection», en *The IEEE Winter Conference on Applications of Computer Vision*, 2020, págs. 2569-2578.

[88] H. Inc, *Open Source Data Labeling — Label Studio.* dirección: `https://labelstud.io/`.

[89] M. Denil, B. Shakibi, L. Dinh, M. Ranzato y N. De Freitas, «Predicting parameters in deep learning», *Advances in neural information processing systems*, vol. 26, 2013.