UNIVERSIDAD DE SALAMANCA

Facultad de Ciencias
Departamento de Informática y Automática

VNiVERSiDAD
D SALAMANCA

TESIS DOCTORAL

Arquitectura Inteligente para el Procesamiento Acelerado de Datos de los Dispositivos IoT en la Computación de Borde

Autora:
Vanessa Estefania Alvear Puertas

Dirigida por los Doctores:
Vivian Félix López Batista
Paul David Rosero Montalvo

Julio 2024

UNIVERSIDAD DE SALAMANCA

Departamento de Informática y Automática
Facultad de Ciencias
Grupo de investigación:
MIDAS

AUTORA:

Vanessa Estefania Alvear Puertas

DIRECTORES:

Vivian Félix López Batista, PhD.
Paul David Rosero Montalvo, PhD.

Declaración de Autoría

La Dra. Vivian Félix López Batista, Profesora Titular del Departamento de Informática y Automática de la Universidad de Salamanca y el Dr. Paul David Rosero Montalvo.

HACEN CONSTAR:

Que, la doctoranda Vanessa Estefania Alvear Puertas ha desarrollado el trabajo titulado: Arquitectura Inteligente para el Procesamiento Acelerado de Datos de los Dispositivos IoT en la Computación de Borde, bajo su supervisión, y por ello autorizan su presentación para la obtención del título de Doctora.

Vivian Félix López Batista
Firmado digitalmente por Vivian Félix López Batista
Fecha: 2024.07.12 18:34:39 +02'00'

Fdo. Vivian Félix López Batista

Paul David Rosero Montalvo
Firmado digitalmente por Paul David Rosero Montalvo
Fecha: 2024.07.12 17:40:08 -05'00'

Fdo. Paul David Rosero Montalvo

Salamanca, 12 de julio de 2024.

DEDICATORIA

Para mi amada familia.

AGRADECIMIENTOS

La inteligencia es la capacidad para adaptarse al cambio.
Stephen Hawking

La presente tesis doctoral es el fruto de mucho trabajo, constancia y dedicación. Muchas personas han sido parte del camino, pero quisiera tomarme un tiempo para agradecer a las que más han estado presentes:

A Isaac y Sebastian por ser mi más grande motivación e inspiración. A mi familia, de manera especial a mi esposo por ser mi apoyo y ejemplo de superación. A mi amada madre, hermanas y hermanos.

A mi tutora la Dra. Vivian López Batista, que con su importante trayectoria en docencia y su profundo conocimiento en investigación, me ha guiado de la manera más acertada durante este proceso.

A mi codirector el Dr. Paul Rosero Montalvo, que gracias a su conocimiento técnico supo darme directrices para la ejecución adecuada de cada uno de los experimentos que dieron como resultado esta tesis doctoral.

RESUMEN

Durante la última década, se ha observado cómo la tecnología de Internet de las Cosas (IoT) se ha posicionado dentro de la vida cotidiana de las personas, ya sea inmersa en el uso de artefactos domésticos conectados al internet o en procesos de manufactura a nivel industrial, conducción de vehículos autónomos, sistemas de seguridad, entre muchas otras aplicaciones. El acelarado desarrollo de esta tecnología la ha llevado también a estar presente dentro del campo de la investigación, donde día a día se establecen nuevos hitos para su desarrollo y aprovechamiento en todos sus ámbitos de aplicación.

Con su importante presencia y uso cotidiano, IoT se ha convertido en una de las tecnologías con mayor generación de datos. Esto ha derivado en que la adquisición de datos desde los dispositivos IoT generen cargas masivas, que demandan altas tasas de procesamiento con tiempos de latencia mínimos, por ello requieren de sistemas robustos, capaces de analizar y procesar toda la información recolectada de manera eficiente. Por lo expuesto, la tendencia actual requiere que los sistemas encargados de procesar datos de dispositivos IoT combinen las prestaciones de *hardware* de Unidades Centrales de Procesamiento (CPUs), Unidades de Procesamiento Gráfico (GPUs), Unidades de Procesamiento de Tensores (TPUs) y Matrices de Puertas Programables en Campo (FPGAs), al emplear este tipo de *hardware* es posible aplicar técnicas de aprendizaje automático que faciliten el procesamiento, la toma de decisiones con los datos adquiridos y reduzcan el tiempo de respuesta de los sistemas.

No obstante, con la diversidad de aplicaciones y dispositivos IoT, se ha convertido un reto el poder determinar las capacidades computacionales y requerimientos a nivel de *software* y *hardware* siendo una problemática aún abierta. Como una solución a esta necesidad, la computación de borde supone cubrir las demandas de estos sistemas. La ventaja de trabajar con la Computación de Borde es que representa menor tiempo de latencia, además proporciona privacidad y seguridad de los datos (al procesarse dentro de la red local) y también cuenta con la posibilidad de enviar datos a la nube cuando se requiera, brindando flexibilidad y escalabilidad al sistema. Los dispositivos que conforman la Computación de Borde son más robustos que los dispositivos IoT por lo que implementar algoritmos de aprendizaje automático resulta una tarea menos compleja en términos de recursos computacionales.

La presente tesis doctoral, propone una arquitectura de tres bloques, que abarca la fase de ingeniería de datos, gestión de modelos y despliegue de modelos. Para ello, se ha realizado un análisis de las herramientas de preprocesamiento de los datos más adecuadas, para luego probar distintas redes neuronales convolucionales pre-entrenadas (Transfer Learning) y pasar a la etapa de despliegue donde se probó la arquitectura en dispositivos de borde centrándose en dos casos de estudio. El principal objetivo es determinar, la correcta combinación de técnicas de preprocesamiento más el uso de algoritmos de aprendizaje automático y técnicas de optimización, para brindar al usuario un marco de trabajo que facilite la implementación de soluciones en dispositivos

de borde en aplicaciones de IoT.

ABSTRACT

During the last decade, it has been observed how the Internet of Things (IoT) technology has positioned itself within people's daily lives. People are either immersed in using domestic devices connected to the Internet or in manufacturing processes at the industrial level, driving autonomous vehicles and security systems, among many other applications. The accelerated development of this technology has also led it to be present in the field of research, where new milestones are set for its development and use in all its fields of application every day.

With its significant presence and daily use, IoT has become one of the technologies with the highest data generation. This has resulted in the acquisition of data from IoT devices generating massive loads, which demand high processing rates with minimum latency times, thus requiring robust systems capable of analyzing and processing all the information collected efficiently. Therefore, the current trend requires that the systems in charge of processing data from IoT devices combine the hardware performance of Central Processing Units (CPUs), Graphics Processing Units (GPUs), Sensor Processing Units (TPUs) and Field Programmable Gate Arrays (FPGAs), by using this type of hardware it is possible to apply machine learning techniques that facilitate processing, decision making with the acquired data and reduce the response time of the systems.

However, with the variety of IoT applications and devices, it has become a challenge to determine the computational capabilities and requirements at the software and hardware level, which still needs to be solved. As a solution to this requirement, edge computing is supposed to satisfy the demands of these systems. The advantage of working with edge computing is that it represents lower latency time, provides privacy and security of data (when processed within the local network) and can send data to the cloud when required, providing flexibility and scalability to the system. Edge computing devices are more robust than IoT devices, so implementing machine learning algorithms is a less complex task in terms of computational resources.

This doctoral thesis proposes a three-block architecture covering the data engineering, model management and model deployment phases. For this purpose, the most appropriate data preprocessing tools have been analyzed to test different pre-trained convolutional neural networks (Transfer Learning) and drive on to the deployment stage, where the architecture was tested on edge devices focusing on two case studies. The main objective is to determine the right combination of preprocessing techniques, machine learning algorithms, and optimization techniques to provide the user with a framework that facilitates the implementation of solutions on edge devices in IoT applications.

ÍNDICE GENERAL

ÍNDICE DE FIGURAS

ÍNDICE DE CUADROS

Parte I

Memoria de la tesis doctoral

CAPÍTULO 1: INTRODUCCIÓN

Este primer capítulo se presenta como una introducción del trabajo doctoral. En él se plantea la problemática que motivó al desarrollo de la investigación, se formula la hipótesis, se definen los objetivos y se describe la metodología de investigación que se empleó para el desarrollo de la tesis doctoral.

1.1. Introducción

El Internet de las Cosas (Internet of Things, IoT) es una tecnología emergente que ha tenido un acelerado desarrollo en los últimos años. Con millones de dispositivos conectados alrededor del mundo, el objetivo del IoT es brindar comodidad y bienestar a sus usuarios proporcionando información confiable para la toma de decisiones [1, 2]. Por lo tanto, los dispositivos IoT deben ser adaptables y flexibles para ser empleados en diferentes aplicaciones como hogares inteligentes, cuidado y monitoreo de indicadores de salud, monitoreo ambiental, transporte inteligente y seguridad [3–7]. Sin embargo, al tratarse de una tecnología en desarrollo, IoT también se enfrenta a varios retos en su implementación, entre los principales, podemos hacer referencia a la capacidad computacional requerida por los dispositivos IoT, su capacidad de almacenamiento y selección del protocolo de comunicación mediante el cual se transmitirán los datos relevantes hacia la Computación de Borde. Este proceso de discriminación de datos es inteligente al aplicar algoritmos de aprendizaje automático [8].

Al abordar los desafíos del IoT, es importante destacar que los dispositivos utilizados en esta tecnología están diseñados para cumplir con requisitos técnicos específicos según su entorno de trabajo. Sin embargo, debido a la diversidad de aplicaciones, arquitecturas y entornos de desarrollo e implementación, existen múltiples definiciones para estos dispositivos. En el contexto de esta investigación, se considera que un dispositivo IoT es aquel que se encarga de la adquisición de datos y de la comunicación con servidores de Computación de Borde. Además, se espera que estos dispositivos cumplan con criterios de seguridad, escalabilidad y disponibilidad. Técnicas como TinyML han buscado realizar la inferencia de datos directamente en el dispositivo IoT [9, 10]; no obstante, dependiendo de la aplicación, esto no siempre es viable.

Otro de los retos mencionados para los dispositivos IoT se refiere a la selección del protocolo de comunicación. Algunos de los protocolos de comunicación que pueden emplearse en dispositivos IoT son Bluetooth, WiFi, *Message Queuing Telemetry Transport* (MQTT), Zig-Bee, LoRaWAN, *Long Term Evolution* (LTE), entre los principales [1, 11, 12]. Antes de seleccionar el protocolo de comunicación es importante realizar un análisis minucioso, que permita determinar la capacidad del canal de transmisión, tamaño de las tramas a transmitir y la seguridad del canal, además dependiendo de la naturaleza de la aplicación existirán protocolos que se adapten de mejor manera, esto en términos de cobertura. Una vez definidos los requerimientos funcionales de los dispositivos

IoT, se conoce que tienen la capacidad de adquirir datos a través de sensores, cámaras, micrófonos, etc. Este proceso conlleva a una carga masiva de datos, que dependiendo de la aplicación podrán ser procesados en el borde (Edge Computing) o en la nube (Cloud Computing) [13], lo cual supone un elevado coste computacional. Para el manejo de estos datos se han desarrollado innumerables herramientas de aprendizaje automático (Machine Learning, ML) y aprendizaje profundo (Deep Learning, DL) [14–16]. Dependiendo de la naturaleza de los datos, ya sean estructurados o no estructurados, existirá un algoritmo con mejor rendimiento para su manejo [17]. Si bien existen innumerables trabajos que se realizan para el manejo de datos estructurados, los datos como imágenes y sonidos, debido a su complejidad, no han sido ampliamente estudiados y tratados en un entorno IoT.

Las cargas masivas de datos han impulsado el desarrollo de herramientas de aprendizaje automático, un ejemplo de ello son las redes neuronales. Recientes estudios sobre redes neuronales han demostrado que estos algoritmos muestran mejoras en el tratamiento de datos en comparación a los algoritmos tradicionales de aprendizaje automático. Entre las redes neuronales más empleadas, vale destacar las Redes Neuronales Convolucionales (Convolutional Neural Networks, CNN), que han presentado alta precisión en tareas de clasificación de imágenes y detección de objetos [14–17], también han sido empleadas para el tratamiento de sonidos presentando un alto rendimiento. Sin embargo, la complejidad de cálculo que demandan estos algoritmos ha limitado su aplicación, ejecutándose únicamente en escenarios donde se dispone de procesadores de altas prestaciones, con suficiente memoria y donde la latencia no es crítica para la aplicación. La capacidad computacional requerida dependerá del algoritmo que se esté ejecutando y de las características del conjunto de datos con el que se trabaje.

La tendencia actual del manejo de grandes volúmenes de datos y la ejecución de algoritmos complejos, ha impulsado a que la computación tradicional evolucione a fin de adaptarse a las nuevas demandas [6]. En la actualidad la computación en la nube se ha presentado como una solución flexible para usuarios finales, desarrolladores de aplicaciones e ingenieros y diseñadores de tecnologías de la información. El acceso a los recursos de forma remota que supone el uso de la computación en la nube, representa tiempos de latencia en los sistemas IoT. Por ello como una solución se introduce la Computación de Borde la cual busca reducir el tiempo que toma transmitir datos de extremo a extremo. Al no realizar consultas a los servidores de la nube y procesar los datos de forma local, representan una importante disminución en los tiempos de respuesta y carga en la red [18, 19]. En contraste a los dispositivos IoT, en el caso de la Computación de Borde, se espera que sus dispositivos sean capaces de ejecutar algoritmos de aprendizaje automático.

A pesar de que las prestaciones de los servidores de la Computación de Borde y la computación en la nube han sido pensadas para altas demandas de coste computacional, el acelerado crecimiento en el número de dispositivos IoT y con ello el aumento en el volumen de datos, ha obligado a que la computación tradicional evolucione hacia una computación heterogénea, capaz de realizar complejas tareas de procesamiento y con mínimos tiempos de respuesta. La computación heterogénea consiste en combinar el uso de CPUs, GPUs y FPGAs aprovechando el rendimiento de cada uno estos elementos para distintas tareas de procesamiento [20], prestando las condiciones idóneas para la ejecución de redes neuronales complejas [14].

Si bien existe un amplio estudio de los dispositivos IoT, sus datos adquiridos y sus miles de aplicaciones presentes en la vida cotidiana, aún quedan varias brechas de conocimiento que requieren ser investigadas y mejoradas. Los requerimientos de los dispositivos IoT, el uso de algoritmos

de aprendizaje automático en los dispositivos de borde y el uso de computación heterogénea son algunos de los temas que generan interés y motivan la presentación de esta tesis doctoral. Por ello, este trabajo plantea diseñar una arquitectura de procesamiento acelerado de datos para entornos IoT, que permita generar un conjunto de datos depurado capaz de brindar información relevante para modelos de aprendizaje automático y que pueda ser ejecutado eficientemente en dispositivos de borde.

1.2. Hipótesis

Respecto a las aplicaciones IoT, el aprendizaje automático y los dispositivos de Computación de Borde mencionados en el apartado anterior, se ha realizado una revisión sistemática del estado del arte. Esto ha permitido conocer los diferentes trabajos que se están desarrollando en estas áreas. Sin embargo, aún queda un amplio campo de investigación en cuanto a la integración de estas tecnologías en una sola arquitectura que optimice el procesamiento de datos y su ejecución en dispositivos de borde, donde los recursos computacionales pueden ser limitados. Por lo tanto, se formula la siguiente hipótesis:

> *Es posible definir una arquitectura de procesamiento acelerado de datos para entornos IoT, capaz de generar un conjunto de datos depurado que proporcione información relevante para modelos de aprendizaje automático y que pueda ser ejecutado eficientemente en dispositivos de borde, considerando sus configuraciones específicas de hardware y software.*

Con la finalidad de validar la hipótesis planteada, esta tesis doctoral pretende abordar una serie de objetivos los cuales se presentan en la siguiente sección.

1.3. Objetivos

Para poder validar la hipótesis planteada, es necesario establecer un enfoque que permita abordar y afrontar la problemática que supone. Por ello, a través de una combinación de investigación teórica y aplicada, se persiguen los siguientes objetivos:

1.3.1. Objetivo general

Diseñar una arquitectura de procesamiento acelerado de datos para entornos IoT, que permita generar un conjunto de datos depurado capaz de brindar información relevante para modelos de aprendizaje automático y que pueda ser ejecutado eficientemente en dispositivos de borde.

1.3.2. Objetivos específicos

- Realizar un análisis comparativo de diferentes arquitecturas que integren IoT y Computación de Borde para evaluar sus ventajas y desventajas, y contrastarlas con la arquitectura propuesta.
- Analizar las técnicas y herramientas de preprocesamiento de imágenes que permitan obtener conjuntos de datos refinados, proporcionando información relevante para el clasificador.
- Optimizar los modelos de aprendizaje automático para reducir su tiempo de procesamiento y tamaño para que puedan ser compilados en los dispositivos de borde sin reducir su rendimiento.

- Validar la arquitectura propuesta mediante su implementación en casos de estudio de aplicaciones IoT.

La consecución del objetivo general y objetivos específicos planteados permitirá diseñar la arquitectura y evaluar los resultados obtenidos que a su vez servirán para contrastar la hipótesis establecida para esta tesis doctoral.

1.4. Metodología de investigación

Para el desarrollo de la presente investigación se han empleado principalmente dos metodologías que permitieron encaminar de manera adecuada la hipótesis llevándola a la consecución de los objetivos de investigación a través de distintas actividades. Las principales metodologías empleadas fueron: Investigación Acción (Action Research) y Estudio de Caso (Study Case).

Se seleccionó Action Research, al tratarse de una metodología que permite mantener un ciclo continuo de planeamiento, acción, análisis y conclusión. Así fue que se empleó para la definición de la problemática, ejecución de actividades como el diseño de la arquitectura, definición de métricas de evaluación y las pruebas de los algoritmos de aprendizaje automático en sus diferentes escenarios. Posteriormente, se procedió a la fase de análisis de los resultados y, finalmente la etapa de conclusiones.

Por otra parte, se empleó la metodología de Estudio de Caso. Una vez establecida la arquitectura, algoritmos de aprendizaje automático y métricas de evaluación, se procedió a testearla mediante su aplicación en casos de estudio. De esta manera se pudo analizar en profundidad y extraer resultados que permitieron identificar desafíos y determinar mejores prácticas.

Con el enfoque en las metodologías antes mencionadas, esta investigación se desarrolló mediante la ejecución de etapas específicas. En primer lugar, se realizó una revisión del estado del arte en aprendizaje automático, arquitecturas de IoT y Computación de Borde, poniendo especial énfasis en comparar distintas arquitecturas que integran IoT y Computación de Borde. Esto permitió identificar ciertas ventajas y desventajas, que se contrastaron con los resultados de la arquitectura propuesta. En segundo lugar, se diseñó una arquitectura de procesamiento acelerado de datos, compuesta por tres etapas: Ingeniería de Datos, Gestión de Modelos y Despliegue de Modelos. En tercer lugar, se validó la arquitectura propuesta mediante su implementación en dos casos de estudio de aplicaciones de IoT en escenarios reales.

Como parte del proceso del trabajo doctoral y con el objetivo de fortalecer conocimientos y contribuir a la educación científica, se ha asistido y participado en diversas conferencias, cursos, seminarios y congresos internacionales. Además, se presentaron publicaciones sobre los avances realizados en la investigación. Así también, la ejecución de una estancia de investigación en un centro de investigación de alto nivel fortaleció la transferencia de conocimiento y fomentó la investigación colaborativa.

1.5. Organización de la memoria

Atendiendo a la hipótesis anteriormente formulada y con el propósito de satisfacer los objetivos establecidos como punto de partida del trabajo, la memoria se estructura de la siguiente manera:

Esta memoria está dividida en cinco capítulos, siendo éste el primero de ellos.

El Capítulo 2 presenta una revisión sistemática del estado del arte, donde se definen los principales conceptos respecto a aprendizaje automático, sus algoritmos y técnicas de preprocesamiento necesarias para su implementación. También aborda, arquitecturas y conceptos de IoT y la Computación de Borde. Finalmente, presenta una sección donde se muestran trabajos relacionados que sirven para avalar la presente investigación.

Una vez analizadas las principales investigaciones en relación con la hipótesis de partida, y tras haber detectado las carencias y necesidades de los sistemas existentes, en el Capítulo 3 se desarrolla la propuesta de este trabajo. Concretamente, se plantea la arquitectura y se detallan las tres fases propuestas.

En el Capítulo 4 se describen los resultados alcanzados con la propuesta diseñada. Además se presentan dos casos de estudio que se han llevado a cabo en entornos reales con el objetivo de validar la arquitectura propuesta. El primer caso de estudio, corresponde a la clasificación de imágenes satelitales. El segundo caso de estudio, corresponde a la clasificación de imágenes para su uso en sistemas de alerta temprana en invernaderos.

El Capítulo 5 recoge las principales contribuciones del trabajo, se presentan las conclusiones inferidas a lo largo del proceso de investigación y se discute el valor de la propuesta en relación con el cumplimiento de los objetivos inicialmente planteados. Adicionalmente, este capítulo esboza algunas líneas de trabajo futuro que podrán ser llevadas a cabo partiendo de esta investigación.

CAPÍTULO 2: CONTEXTO Y ESTADO DEL ARTE

Una vez que se ha definido la problemática y antes de desarrollar la propuesta, el presente capítulo ofrece una descripción de conceptos fundamentales relacionados con el aprendizaje automático, técnicas de preprocesamiento, Redes Neuronales Convolucionales, Aprendizaje por Transferencia y algunas técnicas de optimización. Además se abordan las arquitecturas, dispositivos y tendencias del Internet de las Cosas. Se introducen los conceptos de la Computación de Borde, se analizan sus dispositivos y su integración con la Inteligencia Artificial. Finalmente, se revisan algunos de los trabajos relacionados más relevantes, analizando sus ventajas y desventajas para contrastar con la propuesta de este trabajo.

2.1. Aprendizaje automático

El aprendizaje automático, es una rama de la Inteligencia Artificial (IA) cuyo principal objetivo es que los sistemas computacionales puedan aprender a realizar tareas basándose en experiencias previas. Aunque, los conceptos de aprendizaje automático se remontan a finales de la década de 1950, cuando A.L. Samuel introdujo los primeros estudios de técnicas de aprendizaje automático aplicadas al juego de damas [21], hoy en día encontramos al aprendizaje automático como una pieza fundamental en todas las áreas del conocimiento.

Los algoritmos empleados en el aprendizaje automático requieren grandes cantidades de datos para mejorar su proceso de aprendizaje y su rendimiento. Para ello, se emplean técnicas que permiten identificar patrones en los conjuntos de datos y así ser capaces de crear modelos predictivos [22]. Por lo tanto, la selección de los conjuntos de datos para la fase de entrenamiento de un modelo de aprendizaje automático será determinante para su rendimiento.

Dependiendo de los mecanismos empleados, el campo de aprendizaje automático, se subdivide en dos grandes categorías: 1. Aprendizaje Supervisado, donde se emplean conjuntos de datos etiquetados, para generar una función que mapea las entradas y salidas [23]. En este enfoque, se dispone de clases previamente determinadas. 2. Aprendizaje No Supervisado, que emplea conjuntos de datos no etiquetados y tiene como objetivo descubrir automáticamente patrones y estructuras en los datos, para crear sus propias etiquetas [23, 24]. Además de estos dos grandes grupos, también se han definido los mecanismos de aprendizaje semi supervisado, donde se emplean datos etiquetados y no etiquetados, y el aprendizaje por refuerzo, en el que el algoritmo aprende una política sobre cómo actuar en relación con los datos que recibe. Las decisiones tomadas recibirán una retroalimentación que guía al algoritmo, indicándole si su decisión fue acertada o no.

En el campo del IoT, el aprendizaje automático ha tomado gran relevancia al potenciar las funcionalidades de sus dispositivos. Hoy en día, es posible procesar grandes volúmenes de datos

y tomar decisiones en tiempo real, lo que beneficia de forma directa a los entornos IoT presentes en diversas áreas como la industria, la medicina, el comercio, la seguridad y la investigación. A continuación, en esta sección se presentan los principales aspectos de interés que complementan el aprendizaje automático y el IoT.

2.1.1. Preprocesamiento de datos

Dentro de un entorno de IoT, varios sensores pueden ser usados para la adquisición de datos que se encuentran instalados en diferentes condiciones ambientales que puede afectar la calidad de la información. En consecuencia, los datos pueden verse afectados por ruido que deriva en la generación de modelos de aprendizaje automático complejos. Además, en ocasiones, se ha considerado que los conjuntos de datos de fuentes externas actúan como cajas negras porque se desconoce los escenarios reales donde fue tomada la información. Muchos de estos conjuntos de datos fueron realizados en entornos de laboratorio, lejos de las condiciones reales. Por tal motivo, el preprocesamiento es una etapa previa al entrenamiento de los algoritmos de aprendizaje automático que permite la limpieza de los datos crudos y hace más evidente las características específicas del fenómeno de estudio. Con esto, existen ciertas técnicas que no solo eliminan el ruido de la información, también este criterio puede ser usado para la detección y manejo de valores atípicos, la estandarización de la información y el ajuste de ellos para que puedan ser la entrada de los algoritmos de aprendizaje automático que usualmente cuentan con configuraciones estándar.

Por otro lado, a pesar que los dispositivos IoT son capaces de recolectar grandes volúmenes de datos, al ser preprocesados, solo unas cuantas muestras pueden ser válidas o representar información relevante para el algoritmo de aprendizaje automático. Esto conlleva a un procesamiento iterativo costoso de toma de datos, donde se debe anotarlos adecuadamente y validar su información. Por tal motivo, cuando se conoce que solo existe un número limitado de datos válidos para el modelo, se emplean técnicas de aumentación de datos que buscan proveer diferentes puntos de vista a los datos para enriquecer sus características y simular entornos reales de funcionamiento.

Bajo este escenario, algunas de las técnicas más empleadas según la literatura han sido tomadas en cuenta, las mismas que se presentan a continuación.

2.1.1.1. Aumentación de datos

La aumentación de datos es una técnica utilizada en el ámbito del aprendizaje automático y la visión por computadora que consiste en generar nuevas muestras de datos a partir de las muestras originales mediante la aplicación de transformaciones controladas. Los métodos de aumentación de datos más empleados incluyen rotación, traslación, volteo horizontal o vertical, cambio de escala, y otros tipos de transformaciones que ayudan a aumentar la variabilidad en el conjunto de datos y a mejorar la generalización del modelo [25–27].

2.1.1.2. Selección de características

La selección de características, también conocida como selección de atributos o variables, es un proceso crucial en el análisis de datos y el aprendizaje automático, consiste en identificar y elegir las características más relevantes e informativas de un conjunto de datos. El objetivo principal de la selección de características es reducir la dimensionalidad del conjunto de datos, eliminando aquellas características que son redundantes, irrelevantes o que no contribuyen significativamente a la capacidad predictiva del modelo. Esto no solo simplifica el modelo y reduce el tiempo de entrenamiento, sino que también puede mejorar su rendimiento al reducir el riesgo de sobreajuste

y mejorar la interpretabilidad de los resultados. La selección de características puede realizarse de manera manual, mediante el conocimiento experto del dominio, o de manera automática, utilizando algoritmos y técnicas específicas diseñadas para este propósito, como métodos de filtrado, métodos de envoltura y métodos integrados. Independientemente del enfoque utilizado, la selección de características es un paso crítico en el proceso de análisis de datos que puede influir significativamente en la calidad y la eficacia de los modelos de aprendizaje automático.

2.1.1.3. Reducción de dimensionalidad

El aprendizaje supervisado, en su búsqueda de obtener información más precisa y significativa a partir de conjuntos de datos más grandes, puede enfrentar desafíos al considerar el conocimiento intrínseco de los datos en el diseño de interfaces de usuario. Dado que dichas interfaces se basan en la percepción humana, particularmente en la visión, la cual actúa como un puente entre la información observada desde un ordenador y el cerebro humano, la complejidad de representar datos en tres dimensiones puede limitar la presentación de resultados por parte de algoritmos de aprendizaje automático. Los algoritmos de Reducción de Dimensionalidad (RD) buscan abordar este desafío al simplificar la estructura de los datos y, en consonancia con el rendimiento de respuesta de sistemas embebidos, reducir el número de variables a un conjunto más manejable. Esto tiene como objetivo facilitar la interpretación visual del proceso realizado por el algoritmo y disminuir el tamaño de la información almacenada para su procesamiento posterior. Los métodos utilizados para la reducción de dimensionalidad abarcan enfoques espectrales, basados en disimilitudes, en divergencias y heurísticos. De acuerdo a Van der Maaten [28], los métodos espectrales como Locally Linear Embedding (LLE) y LaPlacian Eigenmaps (LE), así como el método lineal de Análisis de Componentes Pricipales (PCA) y métodos estocásticos basados en divergencias como Stochastic Neighbor Embedding (SNE) y t-Distributed Stochastic Neighbor Embedding (t-SNE), constituyen los principales enfoques y la base de todos los métodos existentes de RD. A continuación, se describen a detalle los algoritmos mencionados:

- Principal Component Analysis (PCA): es un método que busca conseguir un nuevo conjunto de variables o componentes principales que están incorrelacionadas entre sí, a través de transformaciones ortogonales realizadas al conjunto de variables originales. Todo esto se realiza con el objetivo de reducir la dimensionalidad de estos. Una característica de esta técnica es que logra trabajar fácilmente con gran cantidad de datos, consiguiendo así evitar una alta carga computacional [28].
- Laplacian Eigenmaps (LE): es un algoritmo que utiliza técnicas espectrales, es decir, se basa en la suposición de que los datos se encuentran en un forma de baja dimensión en un espacio de alta dimensión. Donde cada dato se utiliza como un nodo y la conectividad entre ellos se rige por la proximidad de los puntos vecinos [29].
- Locally Linear Embedding (LLE): busca hallar la variedad de baja dimensión dentro del conjunto de datos que son de alta dimensión. Además, pretende conservar la propiedad de que si dos puntos estaban próximos en el espacio origen, lo seguirán estando en el nuevo espacio de baja dimensión. Para ello, se realiza una búsqueda de vecinos más cercanos para construir la matriz de pesos y descomponer parcialmente los valores propios [30].
- Stochastic Neighbor Embedding (SNE): utiliza una distribución gaussiana para cada valor de entrada de los datos de alta dimensión con el fin de usar su densidad. De esta manera, establece una distribución de probabilidad de todos los vecinos. Luego, aproxima esta distribución de probabilidad tanto como sea posible repitiendo la estructura de parentesco en un espacio de menor dimensión [30].

- t-distributed Stochastic Neighbor Embedding (t-SNE): es un algoritmo que hace uso de una distribución que mide la similitud entre pares de objetos de entradas y una medida de distribución entre parejas similares de los correspondientes puntos de análisis, representando así un conjunto con una menor dimensión. Este método comienza por encontrar patrones en los datos mediante la identificación de grupos que han sido observados según la similitud de los puntos de datos con varias características. Realiza la reducción de dimensionalidad debido a que asigna los datos multidimensionales a un espacio de menor dimensión [28].
- Autovectores (*eigenvectores*): Los vectores propios, autovectores o *eigenvectores* son vectores asociados a una transformación lineal que no cambia su dirección cuando la transformación lineal es aplicada a ellos. Cada autovector tiene su correspondiente autovalor, que son escalares empleados para representar la magnitud de la transformación que se aplica a los autovectores. En el ámbito del aprendizaje automático son aplicados en técnicas de reducción de dimensionalidad y extracción de características [31–33]

A modo de resumen se presenta el Cuadro 2.1.

Algoritmo de RD	**Objetivo del algoritmo**	**Linealidad**	**Limitaciones del algoritmo**
PCA	Maximiza la varianza	Lineal	Limitado a proyecciones lineales
LE	Preserva distancias locales	No lineal	Puede presentar problemas al generar gráficos de datos vecinos
LLE	Preserva propiedades locales	No lineal	Alto consumo de memoria
SNE	Preserva la estructura local	No lineal	Agrupación excesiva
t-SNE	Preserva la estructura local	No lineal	Provee solo 2 o 3 características
Autovectores	Preserva la estructura local	Lineal	No captura estructuras subyacentes complejas

Cuadro 2.1: Características de algoritmos de reducción de dimensionalidad

2.1.1.4. Selección de prototipos

Las técnicas de selección de prototipos (SP) se fundamentan en la premisa de que no todos los datos aportan información relevante para el clasificador. Por consiguiente, su principal objetivo radica en reducir la base de datos de entrenamiento, al tiempo que se incrementa o mantiene el rendimiento de la clasificación. En este contexto, se aplican tres criterios distintos: (i) *Condensación*, que abarca técnicas destinadas a conservar los puntos más próximos a los límites de decisión, conocidos como puntos de borde. Esta estrategia se sustenta en la observación de que los puntos internos tienen un impacto menor en los límites de decisión en comparación con los puntos fronterizos, y por ende, pueden ser eliminados con un efecto relativamente insignificante en la clasificación. (ii) *Edición*, cuyo propósito es eliminar los puntos de borde al considerarlos como datos ruidosos o discordantes con respecto a sus vecinos, lo que contribuye a suavizar los límites de decisión. Por último, (iii) *Híbrido*, que busca identificar el subconjunto más reducido, denominado S, que mantiene o incluso mejora la precisión de la generalización en la evaluación con datos de prueba. Con este fin, se permite la eliminación tanto de puntos internos como de puntos de borde, siguiendo los criterios adoptados por las estrategias anteriores [34, 35].

A continuación, se describen dos algoritmos de selección de prototipos que son ampliamente utilizados en el campo del aprendizaje automático:

- Condensed Nearest Neighbor (c-NN): Es una técnica de selección de prototipos que se encarga de seleccionar un subcojunto de instancias de entrenamiento, capaz de clasificar correctamente los datos del conjunto original, de esta manera se podrán remover aquellos puntos redundantes, sin perjudicar el rendimiento del modelo. c-NN utiliza el algoritmo de los vecinos más cercanos *kNearest Neighbors* (kNN), es decir, utiliza el criterio de proximidad para clasificar o predecir una clase [27, 36].
- All-k*Nearest Neighbors* (All-kNN): Esta técnica se considera como una extensión del algoritmo kNN. Su funcionamiento se basa en aplicar el criterio de kNN a todos los puntos del conjunto de datos para determinar los datos atípicos. Una vez aplicado kNN, se procede a identificar y eliminar las instancias que son inconsistentes con los valores de sus k vecinos más cercanos [37].

2.1.1.5. Detección de anomalías

La detección de datos anómalos es fundamental en el aprendizaje automático, su objetivo es identificar patrones que tengan características diferentes de las instancias normales [38]. Para detectar anomalías existen diversos mecanismos, algunos se basan en métodos estadísticos, otros toman la distancia como métrica, también existen algoritmos de aprendizaje supervisado y algunos modelos de aprendizaje profundo. A continuación se describen dos técnicas de detección de anomalías basadas en aprendizaje no supervisado:

- *Isolation Forest*: Se trata de un algoritmo de detección de anomalías que basa su funcionamiento en construir árboles de aislamiento, denominados *iTrees*. Fue propuesto por Liu et al. (2008), y toma como punto de partida el concepto de que las anomalías son pocas y diferentes y bajo ese concepto las aísla. Este algoritmo ha probado su eficiencia en grandes conjuntos de datos [38, 39] .
- *One-Class Support Vector Machine (One-Class SVM)*: Es un algoritmo de detección de anomalías que identifica datos anómalos partiendo del concepto de definir una clase normal y establecer un área donde estarán los datos de esta clase. Así, cuando detecte un dato atípico éste se encontrará fuera de la región conocida y podrá identificarlo fácilmente como anómalo.

2.1.1.6. Unicidad de los datos

La unicidad de los datos se refiere a la propiedad de los datos de ser únicos y distintos entre sí. En otras palabras, se espera que cada dato presente en un conjunto de datos sea único y no se repita o se solape con otros datos en el mismo conjunto. La garantía de la unicidad de los datos es crucial para asegurar la integridad y la precisión de los análisis y las decisiones basadas en ellos. En muchos casos, la presencia de datos duplicados o redundantes puede distorsionar los resultados de los análisis y llevar a conclusiones erróneas. Por lo tanto, la detección y eliminación de datos duplicados es una tarea importante en la preparación y limpieza de datos antes de su análisis.

2.1.2. Aprendizaje profundo

El aprendizaje profundo, es una subárea del aprendizaje automático, que ha tomado gran relevancia debido a su capacidad para manejar grandes volúmenes de datos y ejecutar tareas complejas de procesamiento de imágenes, video y audio [3]. Según LeCun et al. (2015), el aprendizaje profundo busca crear modelos capaces de aprender representaciones de datos en distintos niveles de abstracción, empleando para ello el algoritmo de Retropropagación (*Backpropagation*). Este algoritmo ajusta los pesos de las neuronas y minimiza el error de predicción basado en el descenso del gradiente [40].

Dentro del ámbito del aprendizaje profundo, se destacan dos grandes grupos de redes neuronales: las Redes Neuronales Convolucionales (CNNs) y las Redes Neuronales Recurrentes (RNNs). Por su parte, las CNNs son empleadas principalmente para el manejo de imágenes, por su capacidad de captar patrones y estructuras espacciales en los datos. Las RNNs son más efectivas en el manejo de datos secuenciales y series de tiempo. Dado el enfoque de esta tesis doctoral en el procesamiento de imágenes, en esta sección se abordan los principales temas relacionados con las CNNs.

2.1.2.1. Procesamiento de imágenes

El procesamiento de imágenes se ha convertido en una de las técnicas más empleadas en las ciencias de la computación, estando presente en aplicaciones industriales, comerciales, médicas, educativas, de investigación, etc. Esta técnica busca extraer características de determinados objetos con el propósito de realizar tareas de detección, clasificación o reconocimiento de patrones [41, 42].

Existen numerosas técnicas y algoritmos empleados para el procesamiento digital de imágenes, los cuales deben seguir pasos fundamentales para su ejecución. El uso de aprendizaje profundo se ha convertido en una de las principales herramientas para el procesamiento de imágenes, principalmente las CNN, las cuales se mencionan a continuación.

2.1.2.2. Redes Neuronales Convolucionales

Las CNN representan un avance innovador en el ámbito del aprendizaje profundo, especialmente en lo que concierne a tareas vinculadas con la visión por computador. A diferencia de las redes neuronales tradicionales, que enfrentan dificultades para capturar las relaciones espaciales en los datos de imágenes, las CNN han sido diseñadas específicamente para abordar de manera eficaz la estructura en forma de cuadrícula de los datos, lo que las convierte en herramientas altamente efectivas para actividades como el reconocimiento de imágenes, la detección de objetos y la segmentación de imágenes [40].

La innovación arquitectónica central de las CNN radica en la incorporación de capas convolucionales. Estas capas emplean operaciones convolucionales para explorar los datos de entrada utilizando filtros que se ajustan durante el proceso de entrenamiento, capturando así características jerárquicas de regiones locales. Al compartir pesos y sesgos a lo largo de todo el espacio de entrada, las CNN pueden identificar patrones, bordes y texturas en imágenes de forma eficiente, lo que les permite aprender representaciones jerárquicas de la información visual de manera automática [43].

La arquitectura típica de una CNN comprende múltiples capas convolucionales, seguidas de capas de agrupación diseñadas para reducir las dimensiones espaciales y, por consiguiente, disminuir la carga computacional. Las capas completamente conectadas son a menudo añadidas al final del modelo para llevar a cabo tareas de clasificación basadas en las características extraídas previamente [43, 44]. En el Cuadro 2.2 se presenta una descripción de las capas que conforman una CNN:

Tradicionalmente, una capa en redes convolucionales tiene los siguientes pasos:

- Redimensionamiento: El *re-scaling* se refiere al proceso de ajustar los valores de los píxeles de una imagen para que estén dentro de un rango específico. Comúnmente, las imágenes se escalan de modo que los valores de los píxeles estén en el rango [0, 1] o [-1, 1]. Esto facilita el procesamiento numérico y ayuda a evitar problemas de divergencia durante el entrenamiento.

Nombre de la capa	**Descripción**
Capa de entrada	Ingresa los datos a la red, puede tener diferente tamaño o dimensión dependendiendo de la naturaleza de los datos
Capa Convolucional	Aplica operaciones convolucionales a los datos de entrada usando filtros para detercar patrones y características. Cada filtro será responsable de capturar diferentes aspectos de los datos de entrada para después emplearlos en la tarea de clasificación/detección/segmentación [44]
Capa de Activación	Introduce no-linealidad aplicando funciones de activación como ReLU (Unidad de lineal de rectificación), sigmoide o tangente hiperbólica [45]
Capa de agrupación (*pooling*)	Reduce las dimensiones espaciales de los datos de entrada aplicando operaciones de agrupación, al final se obtiene un tamaño de datos reducidos manteniendo las características [45]
Capa de Normalización	Normaliza los datos de capas anteriores mejorando la convergencia y generalización de la red [45]
Capa Totalmente Conectada	Encargada de tomar las características resultantes de las capas anteriores y transformarlas en un vector unidimensional, además de conectar las neuronas con las anteriores y posteriores capas, generalmente empleada al final de la red [44]
Capa de salida	Realiza las predicciones finales de la red, aquí se empleará una función de activación dependiendo de la tarea. La capa de salida produce las predicciones finales de la red, utilizando una función de activación adecuada para clasificación/detección/segmentación [45]

Cuadro 2.2: Capas de una CNN

- Normalización: Es un paso importante para asegurar que los datos de entrada estén en una escala comparable y para ayudar al proceso de entrenamiento. En el contexto de las CNNs, la normalización puede referirse a la normalización de lotes (Batch Normalization) o a la normalización de los datos de entrada de la red.
- Normalización de lotes: Se aplica después de la operación de convolución o de cualquier capa completamente conectada. Ayuda a estabilizar y acelerar el entrenamiento normalizando la activación de cada neurona en una capa a través de los datos de entrenamiento en minilotes [46]. Esto ayuda a mitigar el problema del desvanecimiento o explosión del gradiente y puede permitir el uso de tasas de aprendizaje más altas.
- Relleno de ceros: El relleno de ceros se aplica a menudo antes de realizar operaciones de convolución en imágenes. Agrega ceros alrededor del borde de la imagen de entrada para mantener el tamaño de la salida igual al tamaño de la entrada después de aplicar la convolución. Esto es útil para preservar la información espacial y evitar que la imagen de salida se reduzca demasiado rápido.
- Convolución 2D: La convolución 2D es la operación fundamental en una CNN. Consiste en aplicar un filtro (también conocido como *kernel*) a una imagen de entrada para producir una característica de salida. Esta operación ayuda a extraer características relevantes de la imagen, como bordes, texturas o patrones.
- Activación: Después de aplicar una operación de convolución, se aplica una función de activación a la salida para introducir no linealidad en la red. Algunas de las funciones de activación más comunes son ReLU (Rectified Linear Unit), que es la más utilizada debido a su eficiencia

y efectividad en el entrenamiento, pero también hay otras como Leaky ReLU, ELU (Exponential Linear Unit), etc. Estas funciones ayudan a la red a aprender relaciones no lineales en los datos de entrada, lo que es crucial para el aprendizaje efectivo de características.

2.1.2.3. Arquitecturas de Redes Neuronales Convolucionales

En la actualidad, existen arquitecturas de Redes Neuronales Convolucionales muy estudiadas y analizadas en diferentes contextos donde han demostrado su alto rendimiento, a continuación se describen algunas de las más importantes:

- **VGG16:** Es una arquitectura de red convolucional profunda, propuesta por el *Visual Geometry Group* (VGG) de la Universidad de Oxford. Fue presentada originalmente en el *ImageNet Challenge* del año 2014, donde logró un desempeño destacado, obteniendo el primer lugar en las tareas de localización y clasificación de imágenes [47].

  La principal innovación de VGG16 radica en el uso de filtros de convolución muy pequeños, de 3 x 3, demostrando que una mayor profundidad en la red, con capas convolucionales más pequeñas, mejora significativamente el rendimiento en tareas de visión artificial [47, 48]. Esta arquitectura consta de 16 capas, de las cuales 13 son convolucionales y 3 son totalmente conectadas, con un tamaño de entrada de 224 x 224 píxeles. Cada capa convolucional utiliza la función de activación *Rectified Linear Unit* (ReLU), que permite introducir no linealidades esenciales, esto permite que la red sea más eficiente y mitiga el problema del gradiente descendiente. Así, la red es capaz de aprender representaciones complejas y profundas de las imágenes, mejorando su capacidad de clasificación y reconocimiento [49]. Además, VGG16 utiliza capas de agrupamiento máximo después de cada conjunto de capas convolucionales, esto disminuye la dimensionalidad espacial de las características extraídas, mejorando el rendimiento y reduciendo el riesgo de sobreajuste.

  El funcionamiento de VGG16 puede describirse en algunos pasos. Primero, la red toma los datos de entrada con un tamaño de 224 x 224 píxeles, con tres canales de color (Red Green Blue, RGB). Después, las capas convolucionales se organizan en bloques; los bloques contienen 2 o 3 capas convolucionales seguidas de una capa de agrupamiento. Las capas convolucionales extraen características de los datos. A continuación, se aplica la función de activación ReLU. El siguiente paso es añadir una capa de agrupamiento máximo con un tamaño de filtro de 2 x 2, aquí se reducen el número de parámetros. Al final de la red se añaden 3 capas totalmente conectadas que funcionan como clasificador. Para la capa de salida se utiliza una función de activación *Softmax* [49].

- **ResNet50:** Se trata de una arquitectura de red neuronal profunda, propuesta por el equipo de investigación de Microsoft, liderado por He et al. en 2015.

  ResNet50 aborda el problema de la degradación del gradiente introduciendo un *framework* de aprendizaje residual profundo [50]. Esta red consta de 50 capas de produnfidad, distribuidas en 48 capas convolucionales, una capa de agrupación promedio y una capa de máxima agrupación, con un tamaño de entrada de 224 x 224 píxeles. Los tamaños de los filtros de ResNet50 son de 1 x 1, 3 x 3, y 7 x 7. Los filtros de 1 x 1 se encargan de reducir la dimensionalidad. Por su parte, los filtros de 3 x 3 y 7 x 7 se usan para la extracción de características complejas. Esta es una de las primeras arquitecturas en adoptar la normalización por lotes, que se realiza después de cada convolución y antes de cada activación [51].

La estructura de ResNet50 se basa en bloques residuales. La entrada de cada bloque se sumará a su salida, con lo que busca mitigar el gradiente descendiente. Además, cada bloque residual está formado por capas convolucionales y se emplea la función de activación ReLU. La implementación de ResNet50 ha presentado importantes avances en tareas de visión artificial, específicamente en clasificación y segmentación de imágenes y detección de objetos [52].

- **Inception-V3:** Esta arquitectura fue propuesta por Szegedy et al. en el año 2015, siendo parte del equipo de investigación de Google, específicamente del proyecto GoogLeNet [53].

  El funcionamiento de Inception-V3 puede describirse en algunos pasos. Primero, la red toma los datos de entrada con un tamaño de 299 x 299 píxeles, con tres canales de color (RGB). Después, se utilizan capas convolucionales y capas de agrupamiento, con el objetivo de reducir el tamaño espacial de los datos de entrada. A continuación, se aplican los bloques Inception, éstos permiten la extracción de características de manera profunda. Los módulos Inception están estructurados por convoluciones de 1 x 1, 3 x 3 y 5 x 5 y capas de agrupamiento y máximo agrupamiento. El siguiente paso es la factorización de convoluciones, que busca disminuir la cantidad de parámetros y operaciones en la red. Después, se aplica la normalización por lotes para acelerar el entraniemiento de la red. Al final de la red, se añaden capas totalmente conectadas que funcionan como clasificador. Para la capa de salida se utiliza una función de activación *Softmax* [51].

  Inception-V3 es una arquitectura compleja pero eficiente. Utiliza diversas técnicas avanzadas para mejorar el rendimiento y la eficiencia computacional. Su diseño modular, el uso de convoluciones asimétricas y la normalización por lotes la hacen adecuada para tareas de visión artificial complejas, como clasificación de imágenes y detección de objetos.

- **MobileNetV2:** Esta es una arquitectura de red neuronal, especialmente diseñada para aplicaciones móviles y entornos donde los recursos son limitados. Fue presentada por el equipo de investigación de Google, liderado por Sandler et al. en el año 2018.

  Su funcionamiento se basa en una estructura residual invertida donde las conexiones están entre las capas delgadas de cuello de botella [54]. La principal característica de MobileNetV2 es su convolución separable en profundidad, ésta a su vez puede dividirse en convoluciones profundas y convoluciones puntuales, con ello es capaz de mejorar la dimensionalidad para extraer características y luego reducirla.

  El funcionamiento de MobileNetV2 es el siguiente: Primero, la red toma los datos de entrada con un tamaño de 224 x 224 píxeles, con tres canales de color (RGB). Después se inicia con una capa convolucional de 3 x 3 y 32 filtros, en este paso se extraen algunas características básicas. A continuación, se utilizan los bloques invertidos residuales, estos bloques se forman de tres capas: Convolución Expansiva, Convolución en Profundida y Convolución Lineal Proyectiva. MobileNetV2 emplea la función de activación ReLU6. La capa final emplea la función de activación *Softmax* [55].

- **EfficientNet:** Se trata de una familia de modelos que usa arquitecturas neuronales como línea base para construir redes que alcanzan mejor precisión y eficiencia. Fue presentada en el año 2020, como parte del trabajo de investigación de Google, liderado por Tan et al. [56].

La arquitectura propuesta denominada EfficienNet-B7 es 8.4 veces más pequeña y 6.1 veces más rápida que redes convolucionales anteriores. Los modelos de EfficientNet tienen como objetivo principal optimizar la precisión y el número de operaciones de coma flotante por segundo (Floating Point Operations per second, FLOPs). En esta red, se mide la cantidad de FLOPs necesarios para realizar una inferencia. Además, EfficientNet se destaca por su diseño eficiente y su enfoque de escalado balanceado, que permite tener un alto rendimiento empleando un menor número de FLOPs en comparación con otras redes neuronales. Esto genera, modelos más rápidos y eficientes en términos de consumo de energía.

La arquitectura se forma principalmente a partir de bloques residuales de cuello de botella invertido móvil de MobileNetV2 y bloques de compresión y excitación. EfficientNet-B0 tiene el menor costo y complejidad con 237 capas, mientras que EfficientNet-B7 funciona en el otro lado del espectro con 813 capas [56, 57].

A continuación, en el Cuadro 2.3 se presenta una comparativa de las arquitecturas CNN mencionadas.

Arquitectura	**Propuesta por**	**Año**	**Número de capas**	**Tamaño de entrada**	**Principales características**	**Ventajas**	**Desventajas**
VGG16	Simonyan y Zisserman, Universidad de Oxford [47]	2014	16	224 x 224	Capas convolucionales con filtros de 3 x 3	Diseño sencillo y fácil despliegue	Alta carga computacional
ResNet50	He et al., Microsoft [50]	2015	50	224 x 224	Aplica aprendizaje residual para solventar el problema de la degradación del gradiente	Alta precisión	Mayor uso de memoria
Inception-V3	Szegedy et al., Google [53]	2015	48	299 x 299	Bloques de convoluciones paralelas y factorizadas	Eficiente uso de los recursos computacionales	Diseño de los bloques de Inception aportan mayor complejidad
MobileNetV2	Sandler et al, Google [54]	2018	Variable	224 x 224	Convolución separable en profundidad. Mejora la extracción de características	Bajo consumo de energía	Menor precisión en comparación con arquitecturas más robustas
EfficientNet	Tan y Le, Google [56]	2019	Variable	224 x 224	Método de escalado, emplea bloques residuales y bloques de compresión y excitación	Alta eficiencia computacional	Mayor complejidad en diseño e implementación

Cuadro 2.3: Arquitecturas CNN

2.1.2.4. Aprendizaje por Transferencia

El Aprendizaje por Transferencia (Transfer Learning, TL), es una técnica empleada en el aprendizaje automático, que permite la transferencia del conocimiento de un dominio a otro. Esto significa que se pueden usar modelos desarrollados para una tarea específica como base para un modelo en una nueva tarea. Esto permite, por una parte, desarrollar modelos más eficientes y precisos, y por otra parte reduce el tiempo de entrenamiento de los modelos de aprendizaje automático [58].

El principio aplicado en la transferencia de conocimiento es que existen representaciones generales aprendidas en modelos de redes neuronales profundas que contienen características generales aplicables a otros modelos. Es el caso de las CNNs, entrenadas con grandes conjuntos de imágenes pueden convertirse en un excelente extractor de características. Este enfoque, beneficia principalmente a nuevos conjuntos de datos donde el número de imágenes es limitado, ya que aprovechará las características aprendidas previamente por el modelo [59].

Una vez entrenada la arquitectura CNN, los pesos de las neuronas se congelan y se comparten en un repositorio para futuras descargas. A partir de esta arquitectura, se pueden agregar nuevas capas para crear un clasificador preciso, entrenado con algunas muestras del nuevo dominio; como resultado, el modelo se ajusta al nuevo dominio. La técnica de Aprendizaje por Transferencia ha mostrado un gran potencial en dispositivos de borde, facilitando el desarrollo de modelos más precisos y eficientes mientras minimiza los datos de entrenamiento y los recursos computacionales necesarios [60]. Sin embargo, puede ser necesario optimizar estos modelos para que se ajusten a los requisitos de memoria y coincidan con las arquitecturas de *hardware* de los dispositivos de borde modernos, algunos de los cuales cuentan con aceleradores de *hardware* para mejorar el proceso de inferencia [61]. En este escenario, técnicas de optimización, como la cuantización, la poda de modelos y la destilación del conocimiento pueden reducir la complejidad y el uso de memoria de los modelos de aprendizaje automático, manteniendo al mismo tiempo una precisión adecuada.

2.1.3. Optimización de modelos

Dentro del aprendizaje automático, las técnicas de optimización de modelos son de especial relevancia debido a su capacidad para mejorar la eficiencia y rendimiento de los modelos. En entornos de IoT y dispositivos de borde, estas técnicas adquieren mayor importancia debido a los recursos limitados propios de estos ambientes. A continuación, se describen algunas de las técnicas más importantes para la optimización de modelos.

2.1.3.1. Cuantización

Las técnicas de optimización se centran en reducir la cantidad de los números utilizados para representar los pesos y las activaciones de la red, lo que puede llevar a una menor demanda de memoria y un procesamiento más rápido, sin sacrificar significativamente la precisión del modelo [62]. Sin embargo, los modelos de aprendizaje automático entrenados con la técnica de Aprendizaje por Transferencia tienen su propia arquitectura y complejidad, lo que afecta su capacidad para adaptarse a diferentes arquitecturas de *hardware* que pueden influir en su rendimiento, adaptabilidad y consumo de energía. Además, estos modelos de aprendizaje automático se entrenaron con conjuntos de datos limpios, mientras que en las aplicaciones de IoT, los datos entrantes para inferir en el dispositivo pueden presentar ruido, lo que afecta la precisión del modelo [60].

Incluso si los modelos de aprendizaje automático pudieran exportarse a los dispositivos de borde sin ninguna optimización, son computacionalmente costosos y consumen mucha energía.

Por lo tanto, los modelos deben optimizarse para ajustarse a las restricciones de *hardware* en cuanto a procesamiento, memoria, consumo de energía, uso de red y espacio de almacenamiento. Las técnicas de cuantización podrían acelerar la activación y el peso de las neuronas en diferentes variables. Una de las técnicas de cuantización más utilizadas es la Cuantización Uniforme, que a su vez, cuenta con dos técnicas que son la Cuantización de Enteros y la Cuantización de Punto Fijo. Respecto a la Cuantización de Enteros, se trata de una técnica donde los valores flotantes de los pesos y las activaciones se mapean a valores enteros. Por su parte, la Cuantización de Punto Fijo usa un rango fijo y mapea los valores a números enteros dentro de ese rango [63].

2.1.3.2. Poda de modelos

La poda de modelos es una técnica de optimización que consiste en comprimir los modelos de aprendizaje profundo eliminando algunos de los parámetros menos relevantes. Según Zhang et al. (2022), la poda es una de las técnicas más efectivas debido a que se enfoca en eliminar pesos innecesarios en modelos sobreparametrizados [64]. En [65] se mencionan dos categorías principales de poda: poda de pesos y poda de filtros.

La poda de pesos es un método que borra o comprime los pesos y puede eliminar ciertas conexiones innecesarias; sin embargo, su desventaja es que, al no mantener una estructura organizada, puede dejar datos dispersos. Uno de los tipos de poda por pesos más comunes es la basada en magnitud. Esta técnica elimina los pesos con las magnitudes más pequeñas, inicialmente se ordenan los pesos por su magnitud y luego se elimina un porcentaje de los pesos más pequeños.

Por otro lado, la poda por filtros es una técnica estructurada que elimina filtros completos, y en algunos casos también mapas de características. Esta técnica conserva la estructura regular del modelo y ofrece un ahorro significativo en el consumo de recursos computacionales [66]. Los modelos de poda podrían omitir los ceros durante la inferencia para mejorar la latencia [62].

2.1.3.3. Destilación del conocimiento

La destilación de conocimiento es un método de optimización que ayuda el proceso de entrenamiento de una red reducida denominada *estudiante*, bajo la supervisión de una red más grande denominada *maestro*, con el objetivo de reducir el tamaño del modelo, pero manteniendo la capacidad del *estudiante* de realizar tareas similares a las del *maestro* con costos computacionales menores. Las aplicaciones actuales de destilación de conocimiento son implementadas en dispositivos móviles y sistemas embebidos [67, 68].

El proceso de destilación del conocimiento comprende algunas etapas, las cuales se resumen a continuación: 1. Entrenamiento del modelo *maestro*, se busca alcanzar un alto rendimiento en la tarea para la cual ha sido creado. 2. Generación de salidas blandas, con el modelo *maestro* ya creado, se generan salidas en un conjunto de datos de entrenamiento, se busca crear conocimiento sólido para el aprendizaje del modelo *estudiante*. 3. Entrenamiento del modelo *estudiante*, con el conocimiento adquirido de las salidas blandas más el conocimiento adquirido del conjunto de datos original se aplica una función que permita afinar la capacidad en la tarea que aprendió del *maestro*.

El principal beneficio de la destilación de conocimiento es la eficiencia computacional que se puede alcanzar debido al tamaño reducido del modelo *estudiante*, además debido a su rápida ejecución hace ideal su implementación en entornos IoT y dispositivos con recursos limitados [69]. Otro de los beneficios es la preservación del conocimiento, debido a que permite que el modelo

maestro transfiera su conocimiento sin pérdidas representativas en el rendimiento [70].

2.1.4. Métricas de evaluación en modelos de aprendizaje automático

En el contexto de aprendizaje automático y especialmente en aprendizaje profundo, es fundamental establecer métricas de evaluación que permitan medir el rendimiento de los modelos. Estas métricas no solo facilitan la comparación entre modelos para determinar el más adecuado, sino que también se deben seleccionar en función de la tarea específica y del entorno donde se implementarán. A continuación, se detallan las métricas de evaluación que se consideran más relevantes. Además, se incluyen tres métricas enfocadas a dispositivos de borde: tamaño del modelo, tiempo de ejecución y consumo de potencia.

2.1.4.1. Exactitud

La exactitud, también conocida por su nombre en inglés *Accuracy*, es la métrica que determina la proporción de predicciones correctas entre el número total de predicciones realizadas. Se trata de una de las métricas más empleadas debido a su facilidad de cálculo y comprensión para evaluar el rendimiento de un modelo. Para calcular la exactitud se emplea la siguiente ecuación 2.1.1:

$$\text{Exactitud} = \frac{\text{VP+VN}}{\text{VP+VN+FP+FN}} \tag{2.1.1}$$

Donde:

- VP (Verdaderos Positivos) es el número de muestras correctamente clasificadas como positivos
- VN (Verdaderos Negativos) es el número de muestras correctamente clasificadas como negativos
- FP (Falsos Positivos) es el número de muestras incorrectamente clasificadas como positivos
- FN (Falsos Negativos) es el número de muestras incorrectamente clasificadas como negativos

2.1.4.2. Tasa de Error

Otra métrica empleada en modelos de aprendizaje automático, especialmente en tareas de clasificación, es la Tasa de Error, conocida en inglés como *Error Rate*. Esta métrica permite medir la proporción de predicciones incorrectas en relación al total de predicciones, en otras palabras es el inverso de la exactitud. Para su cálculo se utiliza la siguiente fórmula 2.1.2:

$$\text{Tasa de Error} = \frac{\text{Número de predicciones incorrectas}}{\text{Número total de predicciones}} \tag{2.1.2}$$

2.1.4.3. Exhaustividad *(Recall)*

La exhaustividad o mejor conocida por su término en inglés como *Recall*, es la proporción entre verdaderos positivos sobre el total de positivos reales. Para calcular el *recall* se emplea la siguiente ecuación 2.1.3:

$$\textit{Recall} = \frac{\text{VP}}{\text{VP + FN}} \tag{2.1.3}$$

Donde:

- VP (Verdaderos Positivos) es el número de muestras correctamente clasificadas como positivos

- FN (Falsos Negativos) es el número de muestras positivas que fueron clasificadas incorrectamente como negativos

2.1.4.4. F1-*Score*

Este parámetro de evaluación corresponde a la media armónica entre la exactitud y el *recall*, se utiliza cuando se busca un balance entre los dos parámetros. Para calcularlo se emplea la siguiente ecuación 2.1.4:

$$\text{F1-}Score = 2\text{x}\frac{\text{Exactitud x } Recall}{\text{Exactitud + } Recall} \tag{2.1.4}$$

2.1.4.5. Tamaño del modelo

En entornos de IoT y especialmente cuando se trabaja con dispositivos de borde, una métrica de evaluación fundamental es el tamaño del modelo de aprendizaje automático. Un modelo más pequeño demanda menos recursos computacionales en términos de memoria y tiempo de procesamiento, lo cual es muy beneficioso cuando se trabaja con recursos limitados. Además, presenta ventajas en cuanto a despliegue y consumo de energía. El tamaño del modelo puede ser medido en términos de: número de parámetros, tamaño del archivo (KB, MB, GB), memoria en uso, número de FLOPS.

2.1.4.6. Tiempo de ejecución

La métrica de tiempo de ejecución de un modelo de aprendizaje automático, está principalmente relacionada a la aplicación que se ejecuta. En aplicaciones en tiempo real, se espera que las predicciones se realicen con mínima latencia. En aplicaciones móviles y web, se espera que la respuesta sea transparente para el usuario. Además, un menor tiempo de ejecución representa un menor consumo de energía, lo cual es una ventaja significativa en la fase de implementación de dispositivos de borde.

2.1.4.7. Consumo de potencia

El consumo de potencia es una métrica fundamental cuando se ejecutan aplicaciones en dispositivos de borde. Para su medición se pueden emplear opciones de *software* integradas en los dispositivos, o a su vez empleando *hardware* adicional para medir el consumo. La medida de consumo de potencia estará en vatios (W) o en sus múltiplos. Mantener un bajo consumo permite alargar la vida útil de la batería lo que mejora la eficiencia operativa.

2.2. Arquitectura del Internet de las Cosas

El IoT es un paradigma tecnológico que se refiere a la interconexión de varios dispositivos físicos con características computacionales diferentes, que envían datos entre sí y hacia fuera de su red con el objetivo de describir un fenómeno de forma digital. Generalmente, son unidades computacionales compactadas que cuentan con la integración de una gran variedad de sensores que traducen magnitudes físicas en eléctricas para su posterior interpretación por sistemas de procesamiento de datos. En consecuencia, estos dispostivos son instalados generalmente cerca del evento a medir y que a la vez, se encuentra distante de una arquitectura de datos cableada y de un suministro de energía estable. Esto conlleva a que sean dispositivos de una gran portabilidad y flexibilidad de instalación y que al mismo tiempo sacrifica poder computacional al ser alimentado por baterías. Además, ellos cuentan con una gran capacidad de adquisición de datos con frecuencia de muestreo elevada, rondando usualmente varios KHz. Esto produce que varios de estos dispositivos puedan generar una

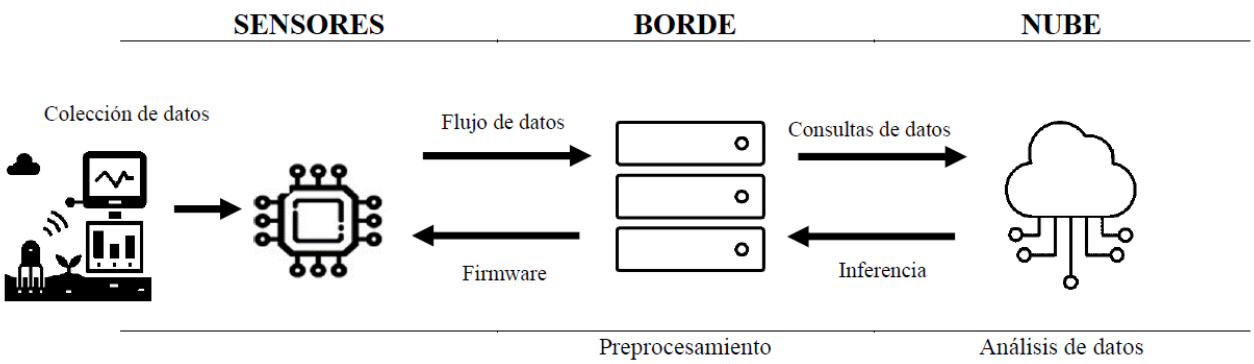


Figura 2.1: Arquitectura en entornos IoT

gran cantidad de datos que deben ser procesados en su mayoría en tiempo real, para contar con una respuesta acorde al tiempo de reacción, en la medida que el fenómeno evoluciona con el tiempo.

Como resultado, se han presentado algunos desafíos relacionados con el balance entre el poder computacional necesario para el procesamiento de datos, la frecuencia de muestreo y envío a un nodo central o servidores fuera de la red local y el consumo de batería. Estos dispositivos encargados de la orquestación de los datos son llamados dispositivos de borde, los cuales debido a sus configuraciones de *hardware* pueden ser parte del entrenamiento de los modelos de aprendizaje automático manteniendo la confidencialidad de la información al procesar los datos de forma local, sin riesgos de exponerlos al enviarlos a la computación en la nube. Finalmente, a pesar que una arquitectura IoT busca descentralizar los sistemas de datos, la computación en la nube juega un papel importante en el entrenamiento y evaluación de los algoritmos de aprendizaje automático. Por esta razón, las primeras etapas de exploración de modelos e inspección de un conjunto reducido de datos, son llevadas a cabo en la nube [71].

En la Figura 2.1 se presenta un ejemplo de una arquitectura común en entornos IoT. En primer lugar, se dispone de una capa de sensores/actuadores para la adquisición de datos. La siguiente capa corresponde al borde, donde se encuentran dispositivos más robustos, capaces de realizar tareas de preprocesamiento y ejecutar modelos de aprendizaje automático. La capa final corresponde a la nube, donde están ubicados los servidores de altas prestaciones que analizan cargas masivas de datos. El intercambio de información puede darse de forma bidireccional, retroalimentando al modelo para mejorar su rendimiento.

2.2.1. Protocolos de comunicación en Internet de las Cosas

Los protocolos de comunicación en IoT son fundamentales para facilitar la interoperabilidad y la transferencia de datos entre los dispositivos conectados [72]. Entre los protocolos más comunes se encuentran *Message Queuing Telemetry Transport* (MQTT), *Constrained Application Protocol* (CoAP), *Hypertext Transfer Protocol* (HTTP) y *Message Queuing Telemetry Transport for Sensor Networks* (MQTT-SN). Cada uno de estos protocolos tiene sus propias características y ventajas, lo que los hace adecuados para diferentes escenarios y requisitos de aplicación dentro del ecosistema del IoT [11]. No obstante, MQTT ha ganado mayor atención debido a la eficacia en su comunicación al ser ligero en el consumo de recursos de red y batería, con una baja latencia de comunicación en un modelo de publicación/subcripción, que permite la comunicación asíncrona de baja latencia entre dispositivos y servidores. Además es muy fiable al ofrecer mecanismos de calidad de servicio que garantiza la entrega confiable de mensajes, incluso en condiciones de comunicación inestable o con

pérdidas temporales de conexión. Finalmente, MQTT es un protocolo de comunicación estándar y abierto, lo que permite ser compatible con una amplia gama de plataformas y dispositivos [73].

2.2.2. Aplicaciones y tendencias en Internet de las Cosas

Con la proliferación de dispositivos inteligentes en todo el mundo, permanece abierta la discusión sobre definir qué dispositivos pueden considerarse IoT y cuáles no. Por lo tanto, es necesario definir qué criterios deben cumplir los dispositivos IoT en cuanto a requisitos tanto de *hardware* como de *software*. En cuanto al *hardware* se espera que los dispositivos sean de tamaño reducido pero a la vez cuenten con suficiente memoria de almacenamiento, capacidad de procesamiento, conectividad y suficiente batería. En cuanto al *software*, se espera que sean capaces de detectar parámetros específicos, ejecutar algoritmos de aprendizaje automático, encriptación y autogestión de la batería, por mencionar algunos.

Una de las principales características que se busca mejorar en los dispositivos IoT se basa en los aspectos de seguridad, la cual en la actualidad es una limitante para un mayor despliegue en aplicaciones a gran escala [74]. Actualmente, los sistemas utilizan algoritmos de cifrado básicos que pueden corromperse fácilmente, lo que implica un riesgo para la seguridad de los sistemas. La correcta gestión de sistemas criptográficos robustos en dispositivos con bajos recursos computacionales es uno de los retos pendientes y tendencia en desarrollo en la actualidad. Por otra parte, respecto a los requisitos de *software* para dispositivos IoT, se espera que éstos sean capaces de ejecutar algoritmos de aprendizaje automático; sin embargo, la complejidad y tamaño de los modelos implica limitaciones a la hora de exportar el modelo a un microcontrolador. Por ello, el desarrollo de técnicas de cuantización y optimización de los modelos de aprendizaje automático para dispositivos IoT es otro reto abierto. Es fundamental definir cuánto es posible cuantificar u optimizar un modelo sin comprometer su precisión.

Sin lugar a dudas, existe una masiva presencia de aplicaciones y dispositivos IoT en la vida cotidiana. En la Figura 2.2 se condensan algunas de las aplicaciones que son tendencia en la actualidad. En el sector de transporte y movilidad, las aplicaciones de IoT se utilizan para controlar el tráfico, y ha habido un aumento en el número de vehículos autónomos no tripulados, así como en aplicaciones de monitoreo y rastreo. En sistemas de seguridad, las tecnologías IoT se aplican en la detección de rostros, como en los controles de seguridad de los aeropuertos. Por otra parte, en la agricultura, los drones y robots automatizan procesos de riego y detección de enfermedades. Finalmente, la presencia de IoT en la industria ha marcado un hito, siendo esencial en procesos de predicción de mantenimiento, automatización y control.

2.2.3. Dispositivos del Internet de las Cosas

Debido a su gran variedad de aplicaciones e integración de sensores, los dispostivos IoT se pueden clasificar según la forma en que adquieren datos. En consecuencia, los dipositivos IoT puede ser de teledetección (con su acrónimo en inglés *remote sensing*) y en el lugar (con su acrónimo en inglés *in situ*). Los sensores de teledetección se encuentran alejados del evento para proteger sus características físicas o tener un mayor campo de visión sobre el objeto a sensar. Generalmente, estos dispositivos son robustos ya que integran cámaras en la mayoría de los casos y esto demanda un mayor coste computacional en la toma de datos con protocolos de comunicación de datos de mediano y largo alcance. Por su parte, los dispositivos IoT de sensado en el sitio, son sistemas que emplean sensores ubicados en la proximidad inmediata del área que se desea monitorear o controlar. Estos sensores están diseñados para capturar datos específicos dentro de un entorno físico particular, sin necesidad de enviar los datos a una ubicación centralizada para su procesamiento

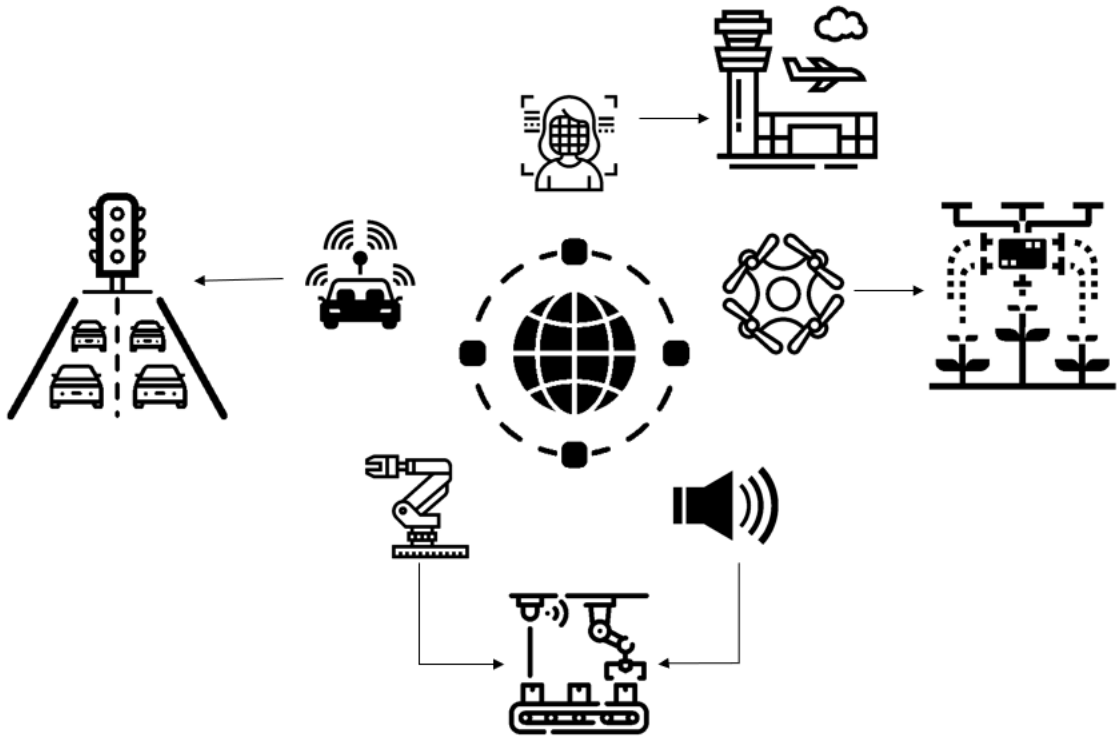


Figura 2.2: Escenarios y tendencias de aplicaciones de IoT

y análisis. Por tal motivo, en estos escenarios, se enfocan en ser sensores de tamaño reducido con un mínimmo consumo de energía para prolongar su tiempo de vida y que generalmente están expuestos a condiciones atmosféricas extremas [75].

2.2.4. Computación de Borde

Debido a que los dispositivos IoT no cuentan con la capacidad de procesar los datos generados por ellos mismos, requieren de una arquitectura estándar de comunicación que cuente con un nodo central encargado de la orquestación del sistema y del envío y recepción de datos. Por lo tanto, estos dispositivos llamados de borde, pertenecen a la red local de comunicación, y ofrecen una mayor potencia computacional que los dispositivos IoT. En la actualidad, los dispositivos de borde disponen de diferentes configuraciones de *hardware* para acelerar el procesamiento de datos. Una de sus principales ventajas es que, al ser implementados dentro de la red local y no enviar los datos directamente a la nube, la latencia en la transmisión, es menor. En consecuencia, se convierte en un sistema que permite el ahorro de energía en los dispositivos IoT al encargarse de las tareas de análisis de datos y del envío a la nube si es necesario. Además, los dispositivos de borde suponen un ahorro en el consumo de ancho de banda al enviar los datos en la red local, siendo además una forma confiable, flexible y adaptable al poder contar con una rápida reconfiguración del sistema [5,18,19].

La Computación de Borde ha impulsado un cambio en el uso tradicional de sistemas centralizados y computación en la nube, abriendo el paradigma del uso de dispositivos en el borde de cada aplicación IoT y, por lo tanto, más cerca de los dispositivos finales. Esta tecnología podría reducir los tiempos de latencia, y ha modificado significativamente el tráfico de red. También se ha acoplado al uso de diversos modelos de aprendizaje automático, introduciendo el concepto de Inteligencia Artificial en el borde (Edge AI), que es una de las propuestas más adaptables a las demandas globales, especialmente en entornos IoT [76]. Sin embargo, los modelos de aprendizaje

automático necesitan ser optimizados para ejecutarse en dispositivos pequeños. Esta arquitectura enfrenta diferentes retos, como elegir la técnica de optimización adecuada, el modelo de Aprendizaje por Transferencia a utilizar, definir el *hardware* adecuado en el borde para el entrenamiento con pocas épocas, y las restricciones de *hardware* que pueda tener.

2.2.4.1. Dispositivos de borde

Los dispositivos de borde deben contar con varias configuraciones de *hardware*, procesadores, memoria, controladores de periféricos y circuitos de entrada/salida en un solo chip o módulo compacto que se adapten a las condiciones específicas de cada aplicación. Por tal razón, los sistemas en un chip *(System on a Chip, SoC)* y sistemas en un módulo *(System on a Module, SoM)* son opciones muy adecuadas para el despliegue de soluciones en el borde. Actualmente, los SoM han ganado gran atención debido a tener una integración en microcontroladores robustos que pueden ser activados para un consumo muy bajo de potencia y que a su vez puede activar varios procesadores para tareas intensas de cómputo. Además, con la miniaturización de semiconductores, los aceleradores de *hardware* como las TPUs, GPUs, FPGAs y las unidades de procesamiento de neuronas *(Neural Processing Units, NPUs*), son dispositivos diseñados para realizar tareas computacionales específicas de manera más eficiente que los procesadores tradicionales [77]. Las TPUs están optimizadas para operaciones de aprendizaje automático y procesamiento de grandes volúmenes de datos en paralelo, mientras que las GPUs son ideales para tareas intensivas en cálculos vectoriales, como gráficos en 3D y procesamiento de imágenes. Por otro lado, las FPGAs ofrecen flexibilidad y capacidad de reconfiguración, lo que las hace adecuadas para aplicaciones que requieren una alta personalización y baja latencia, como la criptografía y el procesamiento de señales [78]. Las NPUs, por su parte, están diseñadas específicamente para acelerar operaciones relacionadas con la Inteligencia Artificial y el procesamiento de redes neuronales, ofreciendo un rendimiento optimizado para cargas de trabajo de inferencia y entrenamiento de modelos de aprendizaje automático. Estos aceleradores de*hardware* juegan un papel crucial en la mejora del rendimiento y la eficiencia de los sistemas informáticos modernos, permitiendo la ejecución de aplicaciones cada vez más complejas y exigentes en términos de recursos computacionales. Sin embargo, para que todas sus capacidades computacionales sean utilizadas eficientemente, se deben considerar instrucciones personalizadas en cada configuración.

2.2.4.2. Inteligencia Artificial en el Borde (Edge AI)

Como se mencionó en las secciones anteriores, los dispositivos actuales cuentan con la capacidad de integrar aceleradores de *hardware* en los dispositivos de borde, debido a ello, una parte del desarrollo del modelo de aprendizaje automático puede ser integrado dentro de esos dispositivos. La capacidad de tomar sus propias decisiones, evaluar su rendimiento y re-entrenar modelos se denomina Inteligencia Artificial en el borde (Edge Artificial Intelligence, Edge IA). Es una propuesta que combina la Inteligencia Artificial con los beneficios de mantener la computación en el borde. Los dispositivos de Edge AI generalmente disponen de sensores y procesadores capaces de soportar distintos protocolos de comunicación para transferir los datos recopilados. Algunos ejemplos de dispositivos de Edge AI son teléfonos inteligentes, relojes inteligentes, cámaras de vigilancia, dispositivos de detección biométrica, GPS, por nombrar algunos ejemplos. Por su capacidad de procesamiento estos dispositivos han implementado técnicas de aprendizaje automático, haciéndolos capaces de tomar decisiones y ejecutar tareas de forma autónoma. Edge AI también permite utilizar dispositivos periféricos y servidores locales, lo que facilita el procesamiento de datos siendo una estrategia de trabajo colaborativo y eficiente. El principal objetivo de Edge AI es minimizar los tiempos de respuesta en los sistemas, es decir, reducir la latencia, lo que también implica una

reducción del consumo de energía [71].

El acelerado despliegue de Edge AI le ha permitido posicionarse como una de las tecnologías con mayor número de aplicaciones en desarrollo. En [79] se analizan algunas de las aplicaciones más relevantes de esta tecnología, como son: parqueo inteligente, vehículos energéticamente inteligentes, drones para IoT industrial, monitoreos de salud, realidad aumentada, hogares inteligentes, *marketing* digital, entre otros. Dependiendo de la aplicación se emplearán distintos algoritmos de aprendizaje automático y aprendizaje profundo, permitiendo a los dispositivos la toma de decisiones, en algunos casos en tiempo real. El futuro de Edge AI promete estar presente en la vida cotidiana de las personas, y más activamente en los sectores industriales y de comercio.

2.2.5. Computación en la Nube

El objetivo de la Computación en la Nube (Cloud Computing) es brindar diferentes servicios bajo demanda, y con ello reducir los costos de implementación brindando a cada usuario los recursos que requiere. Bajo este concepto, Cloud Computing ofrece tres modelos de servicios que son: Iaas (Infraestructure as a Service), enfocado en atender la demanda de administradores de tecnologías de la información e ingenieros de redes; PaaS (Platform as a Service), pensado en el uso para desarrolladores de aplicaciones; y SaaS (Software as a Service) donde se encuentran millones de usuarios finales que acceden a aplicaciones tan cotidianas como el correo electrónico. Para el despliegue de estos tres modelos, la computación en la nube emplea diferentes herramientas, principalmente la virtualización, computación distribuida y los servicios de software. Por su naturaleza, el Cloud Computing requiere de infraestructura tecnológica robusta a la cual se accederá de forma remota, con lo cual la latencia empieza a hacerse presente. [19]

Además, la computación en la nube permite un acceso a recursos informáticos, como servidores o almacenamiento de datos por medio de internet. Por esta razón, se tiene una ventaja sobre servidores locales debido a que su escalibilidad es sencilla sin incurrir en altos costes. Además, el acceso a sus servicios no depende de una ubicación fija, ya que mediante credenciales y accesos seguros, la información o servicios pueden ser vistos desde varios puntos de forma simultánea. Con respecto al desarrollo de algoritmos de aprendizaje automático, los servidores alojados en la nube, usualmente tienen robustos aceleradores de *hardware*, los cuales permiten entrenar modelos a una mayor velocidad ya que cuentan con sistemas electrónicos dedicados. Sin embargo, a pesar que dentro del envío de datos hacia la nube, se cuenta con una gran cantidad de herramientas de seguridad, no está completamente excento de que usuarios mal intensionados encuentren la forma de acceder a los repositorios y copiar/modificar los datos. Además, a pesar que los servicios de la nube garantizan la confidencialidad de la información, muchos proveedores usan *software* o *hardware* de terceros, convirtiéndose en una puerta trasera de acceso.

2.3. Trabajos Relacionados

Una vez realizado un exhaustivo análisis del estado del arte, a continuación se recogen los trabajos relacionados que se consideraron más innovadores, en los cuales se abordan distintos enfoques y soluciones para la computación en el borde y los algoritmos de aprendizaje profundo que pueden ejecutarse en dichos dispositivos. Estos trabajos brindan perspectivas que han permitido contrastar con la arquitectura propuesta más adelante en esta tesis doctoral.

2.3.1. Arquitecturas propuestas

Las arquitecturas de Internet de las Cosas, generalmente se diseñan según los requisitos técnicos de aplicaciones específicas, lo que ha dado lugar a numerosas propuestas en ámbitos comerciales, industriales y de investigación. El número de capas de la arquitectura y sus funcionalidades dependerá del contexto y aplicación para la cual se ha establecido. Por otro lado, la Computación de Borde, ha emergido como una tecnología clave para entornos IoT, y es así que su arquitectura de referencia se encuentra en constante evolución. Por ejemplo, Sun et al. (2016) presentaron uno de los primeros trabajos que analizaron los desafíos de arquitecturas tradicionales donde los datos son procesados en la nube, y para contrarrestar estas limitaciones propuso una arquitectura para IoT que se ejecute en el borde. El principal enfoque de esta arquitectura fue proveer flexibilidad y privacidad en los servicios IoT. Como resultado, su arquitectura fue capaz de reducir la carga de tráfico en la red central y mejorar la latencia entre los dispositivos [80].

Otro trabajo importante que propone una solución para la Computación de Borde es el presentado por Ren et al. (2017), donde se describe una arquitectura escalable para el IoT. El principal enfoque es trasladar la provisión de servicios desde la nube hacia el borde, lo cual reduce el tiempo de latencia y mejora la eficiencia energética. La arquitectura propone varias capas: una capa de usuario final, una capa de red de borde, una capa de red central, una capa de servicios y almacenamiento y una capa de gestión [81].

Por su parte, Plastiras et al. (2019) presentaron una arquitectura para dispositivos en el borde, diseñada para aplicaciones de detección de objetos que requería baja latencia y la toma de decisiones en tiempo real, para ello emplearon Redes Neuronales Convolucionales. Propusieron un marco de trabajo capaz de detectar objetos en fotogramas de video, que prometía mejoras en cuanto a tiempos de procesamiento en dispositivos de recursos limitados. El caso de estudio presentado, se trató de la detección de peatones con un vehículo autónomo no tripulado (conocido por sus siglas en inglés como UAV). Como resultado, presentaron métricas de sensibilidad, tiempo de procesamiento y consumo de energía. De acuerdo a los autores, demostraron que al implementar un mecanismo inteligente de reducción de datos contribuyó a mejorar la precisión del modelo y permitió centrar el cálculo en las regiones importantes de las imágenes [82].

Más adelante, Qiu et al. (2020) diseñaron una arquitectura para la Computación de Borde en aplicaciones industriales de IoT (IIoT). La arquitectura de referencia consta de tres capas: Capa de Dispositivos, Capa de Borde y Capa de Nube, a su vez la capa de borde se subdivide en borde cercano, borde medio y borde lejano. El trabajo además abarca un análisis respecto a protocolos de comunicación, enrutamiento, análisis de datos y seguridad [83].

Además, encontramos el trabajo de Chang et al. (2021), el cual ofrece una detallada revisión de los avances recientes de la Computación de Borde impulsada por la IA. En su investigación, proponen una arquitectura orquestada entre el borde y la nube para sistemas IoT con Inteligencia Artificial. Esta arquitectura se compone de tres capas principales: Capa final, donde millones de sensores y actuadores interconectados son desplegados para la toma de datos, y se espera de ellos la ejecución de tareas sencillas y preprocesamiento de los datos. Capa de borde: en esta capa se dispone de capacidad de cómputo y almacenamiento, con la posibilidad de enviarlo a la nube si la complejidad de cómputo requerida supera a la disponible. Capa de nube: es capaz de coordinar la capa final y la capa de borde, en esta capa se entrenarán los modelos de Inteligencia Artificial y se trabajará con las cargas de datos masivas [5].

Para una mejor organización de la información, se presenta el Cuadro 2.4 donde se resumen algunos de los trabajos revisados en la búsqueda del estado del arte.

Trabajos revisados	**Principales contribuciones**	**Ventajas**	**Desventajas**
[80]	Propone la arquitectura *EdgeIoT* que busca mejorar el procesamiento de IoT empleando la Computación de Borde	Reducción de latencia y tráfico en la red	Criterios de seguridad y privacidad por definir
[81]	Propone una arquitectura basada en la computación transparente, trasladando la provisión de servicios desde la nube hacia el borde	Reducción de latencia y eficiencia energética	Escalabilidad
[84]	Propone *Edge of Things*, a través de la integración de IoT, edge computing y cloud computing	Optimización de recursos mediante una computación distribuida	Complejidad en la orquestación de los recursos en la red
[82]	Presenta *EdgeNet* una arquitectura para mejorar la detección de imágenes mediante CNN	Disminución del consumo de energía en dispositivos de borde	Restricciones de rendimiento dependiendo de las prestaciones del dispositivo de borde
[83]	Propone una arquitectura para Industrial IoT (IIoT)	Reducción de latencia y tráfico en la red	Criticidad en el manejo de datos y seguridad
[5]	Integra la Computación de Borde con la IA en una arquitectura de tres capas	Se propone para aplicaciones en tiempo real	Heterogeneidad de dispositivos e interoperabilidad
[85]	Presenta un protocolo y arquitectura que integra IA, edge computing y blockchain para dispositivos IoT	Alto rendimiento y seguridad avanzada	Aumento del consumo de energía
[86]	Presenta un estudio de la integración de IoT con tecnologías como el aprendizaje federado y la IA explicable, destinado a la detección de intrusos	Aporta un enfoque de seguridad multicapa empleando la IA	Alto consumo de energía
[87]	Presenta una propuesta para asignación eficiente de recursos en sistemas colaborativos de la Computación de Borde y computación en la nube	Aporta un esquema colaborativo para optimizar recursos computacionales	Alta complejidad debido a la diversidad de dispositivos y sus características

Cuadro 2.4: Comparativa de arquitecturas que integran IoT y Computación de Borde

2.3.2. Modelos de aprendizaje automático para dispositivos de borde

Los sistemas que ejecutan algoritmos de aprendizaje automático adoptan diferentes enfoques para generar inferencias; lo que afecta la precisión, rendimiento, consumo de energía y calidad del modelo. Como resultado, existen numerosas formas de combinar hiperparámetros, métricas, herramientas y modelos de inferencia. Debido a esta diversidad, evaluar los modelos y definir la mejor receta para implementar en una solución específica puede resultar un desafío. Por lo tanto, los puntos de referencia (conocidos por su nombre en inglés como *benchmark*) buscan guiar el proceso de implementación del aprendizaje automático en varios escenarios.

En los últimos años, los *benchmarks* de aprendizaje de automático han adquirido especial relevancia debido al enfoque que pueden proporcionar respecto de las numerosas arquitecturas existentes. Trabajos como [88–91] han investigado diferentes aspectos y aplicaciones de los algoritmos de aprendizaje automático para establecer métricas de comparación que permitan tomar las mejores decisiones a la hora de implementar soluciones.

En 2018, MLCommons, una asociación formada por líderes en Inteligencia Artificial en el ámbito académico, industrial, investigadores y laboratorios, desarrolló MLPerf™ para mantenerse a la vanguardia de las tendencias de la industria. Desde ese año, MLPerf ha seguido evolucionando hasta convertirse en un estándar para la industria, y continúan realizando pruebas a intervalos regulares y agregando nuevas cargas de trabajo y dispositivos de acuerdo con las demandas de la industria. Los principales estudios y métricas se centran en las siguientes áreas: MLPerf Training, MLPerf Training HPC, MLPerf Inference: Datacenter, MLPerf Inference: Edge, MLPerf Inference: Mobile, MLPerf Inference: Tiny y MLPerf Storage.

De las áreas mencionadas, MLPerf Inference es el más relacionado con la presente tesis doctoral, [92] proporciona un extenso trabajo que establece un método de evaluación comparativa para sistemas de aprendizaje automático y aporta reglas específicas para su aplicación en la industria. En el mencionado trabajo se analizan diferentes modelos versus casos de uso. En este sentido, es fundamental encontrar un equilibrio entre la tarea objetivo y el modelo que se va a utilizar. También es fundamental analizar el escenario y los recursos disponibles para cada tarea. Otro análisis crítico dentro de este trabajo es la comparación entre latencia versus rendimiento; hay casos en los que un sistema puede ofrecer un rendimiento excelente a pesar de tener tiempos de latencia elevados; en este caso, se resume en el principio de que todo dependerá de la tarea objetivo para la que se implemente el sistema. Finalmente, MLPerf Inference define un marco para medir el rendimiento de inferencia de los algoritmos de aprendizaje automático en diferentes escenarios.

En [58] presentan una comparación de arquitecturas de CNN ejecutándose en dispositivos de la Computación de Borde, algunos de ellos con aceleradores de *hardware*. La investigación se conduce empleando conjuntos de datos y aplicando técnicas de Aprendizaje por Transferencia, también presentan técnicas de optimización de los modelos de aprendizaje automático con el objetivo de mejorar la precisión y rendimiento de los modelos. Como resultado, han demostrado el mejor rendimiento con EfficientNet, incluso con dispositivos de borde con limitaciones de *hardware*.

Si bien existen trabajos relacionados con arquitecturas de IoT, algunos se centran en los protocolos de comunicación, mientras que otros analizan modelos de aprendizaje automático y otros evalúan el rendimiento basados en métricas como el consumo de energía, tamaño de modelos, entre otros. Sin embargo, aún queda abierta la posibilidad de proponer una arquitectura que compare diversas técnicas de reducción de dimensionalidad, ponga a prueba diversos modelos de aprendizaje automático y evalúe su implementación mediante casos de estudio en conjuntos de datos reales. Por tanto, esta tesis doctoral busca proponer una arquitectura que aborde todos estos aspectos y proporcione a los usuarios un marco de trabajo capaz de ejecutarse en dispositivos de borde. En los siguientes capítulos se desarrollará la arquitectura propuesta y se presentarán los resultados obtenidos

CAPÍTULO 3: ARQUITECTURA PROPUESTA

El presente capítulo detalla la arquitectura de aprendizaje automático propuesta para conseguir un conjunto de datos depurado, que presente información relevante al modelo de ML con una arquitectura CNN conocida y que permita que el modelo pueda ser ejecutado en los dispositivos de borde acorde con sus configuraciones de hardware y software. Para ello, el modelo debe ser optimizado, siendo capaz no solo de realizar inferencias, sino también reentrenar el modelo cuando el rendimiento de éste disminuya. Esto se realiza en base a un enfoque donde el dispositivo de borde es un nodo central que recopila información de dispositivos IoT de forma independiente, pudiendo tener limitaciones en sus condiciones de trabajo, que exijan reducir su coste computacional sin decaer en su rendimiento.

3.1. Arquitectura propuesta

Para enfrentar el desafío de los entornos de IoT y de manera especial de los dispositivos de borde, que deben analizar y procesar cargas masivas de datos, se propone una arquitectura que facilite el procesamiento acelerado de datos. Aunque estas cargas comprenden datos heterogéneos como videos, audio e imágenes, en la actualidad es posible darles un tratamiento común al transformarlos a un mismo formato: imágenes. Es así, que los datos de videos pueden ser descompuestos en fotogramas y procesados como imágenes estáticas. Por otro lado, los datos de audio pueden interpretarse visualmente mediante espectogramas de frecuencia y amplitud en el tiempo. Por esta razón, se propone una arquitectura diseñada específicamente para el procesamiento de imágenes, considerando que su flexibilidad le permitirá adaptarse a otros tipos de datos.

La arquitectura propuesta comprende tres bloques principales. El primer bloque es la ingeniería de datos, donde se describen las bases de datos utilizadas en esta investigación, seleccionadas por su relevancia en diversas aplicaciones de IoT. En esta etapa, se consideran varias técnicas de preprocesamiento para reducir el tamaño de los conjuntos de datos según criterios categóricos (seleccionando datos que aporten mayor relevancia al modelo de aprendizaje automático) o numéricos (reduciendo el número de instancias para maximizar la capacidad de decisión del modelo). Para determinar el criterio más adecuado, se entrenaron y ajustaron varias arquitecturas de CNNs para entrenar varios modelos de ML con estos subconjuntos de datos. Como resultado, se evaluó el rendimiento de clasificación de cada arquitectura CNN en un escenario sin restricciones computacionales, para que luego se puedan comparar en entornos IoT con limitaciones de cómputo.

Una vez seleccionado el criterio de preprocesamiento y evaluado el rendimiento de los modelos de ML, se continua con el segundo bloque, donde se optimizarán utilizando dos técnicas. Primero, se empleará la cuantización para realizar inferencias con modelos de resolución de 32 y 8 *bits*. Segundo, se aplicará la destilación de conocimiento para entrenar modelos derivados con menos

capas CNN, reduciendo su complejidad y tamaño mientras se maximiza su rendimiento. Esta etapa, siendo computacionalmente intensiva, se realizará en un servidor de altas prestaciones, sin restricciones de tiempo de procesamiento. Tras validar los modelos derivados con diferentes métricas de rendimiento, se exportarán a varios dispositivos de borde para verificar su desempeño en infraestructuras de *hardware* limitadas.

Finalmente, el tercer bloque corresponde al despliegue de modelos. En este bloque los modelos de aprendizaje automático seleccionados se reentrenarán en los dispositivos de borde para aplicaciones IoT específicas y con el conjunto de datos depurado con el objetivo de probar la arquitectura propuesta sin comprometer su capacidad de decisión. La Figura 3.1 muestra el flujo de trabajo propuesto en cada etapa, el cual se describirá en detalle en las siguientes secciones de este capítulo.

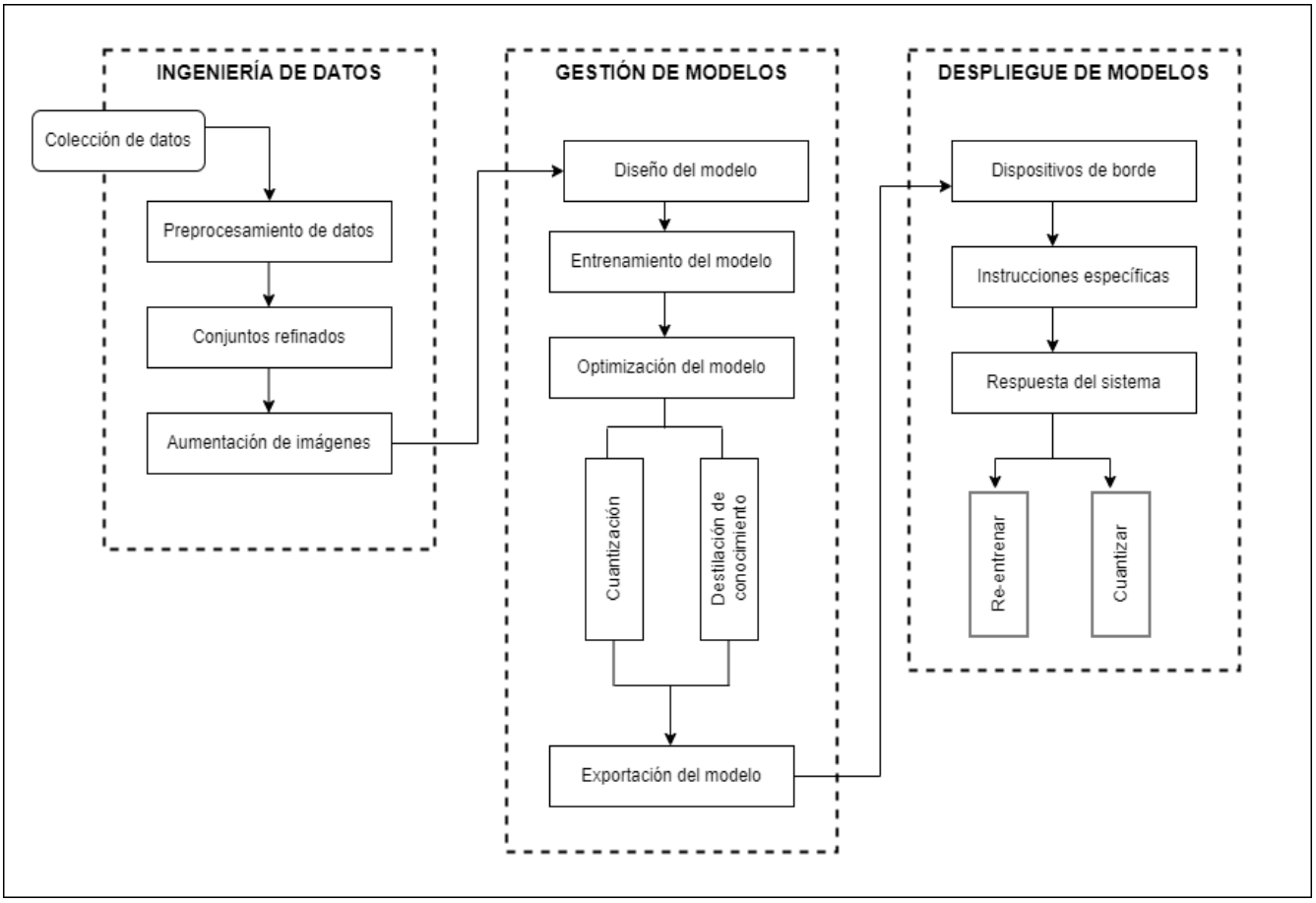


Figura 3.1: Arquitectura propuesta para la optimización de modelos de ML en dispositivos de borde

3.1.1. Fase 1: Ingeniería de datos

Esta sección se centra en la recolección de datos de diferentes aplicaciones IoT y su preprocesamiento para disminuir su tamaño mientras que no se afecte la información intrínseca que contiene. La primera fase inicia con la descripción de los conjuntos de datos tomados en consideración en esta investigación, estos conjuntos de datos se han empleado debido a que ejemplifican las aplicaciones de IoT más conocidas. Luego, ya que uno de los enfoques es estudiar la relevancia en el tamaño del conjunto de entrenamiento en modelos pre-entrenados, se emplean dos criterios. El primero, es reducir el número de instancias de los conjuntos de datos de forma aleatoria y observar en porcentaje, cuál es el valor mínimo que los modelos de aprendizaje automático necesitan para

generar un buen rendimiento. Segundo, empleamos técnicas de extracción de características para valorar la relevancia de cada imagen y su aporte hacia el clasificador para decidir conservarla o eliminarla. En consecuencia, para emplear estas técnicas, la reducción de dimensionalidad es necesaria. Estos algoritmos serán explicados en el Capítulo 4. Además, para realizar la reducción de dimensionalidad, también se ha considerado emplear redes neuronales para la extración de los autovectores.

Finalmente, empleando los dos criterios previamente explicados, se obtienen conjuntos refinados que serán utilizados para entrenar varios modelos de aprendizaje automático con diferentes arquitecturas CNN para determinar su rendimiento. Además, estos conjuntos pueden emplear aumentación de datos para brindar características adicionales al clasificador para mejorar su rendimiento. Como resultado, se espera definir los criterios de preprocesamiento que más se ajusten a los modelos de aprendizaje automático aplicados en entornos IoT. La Figura 3.2 muestra gráficamente el proceso de ingeniería de datos presentado en esta investigación.

Donde:

- P, corresponde al conjunto de datos de prueba.
- E, corresponde al conjunto de datos de entrenamiento.
- V, corresponde al conjunto de datos de validación.
- RD, corresponde a las técnicas de reducción de dimensionalidad.
- CD, corresponde de forma general a los conjuntos de datos.

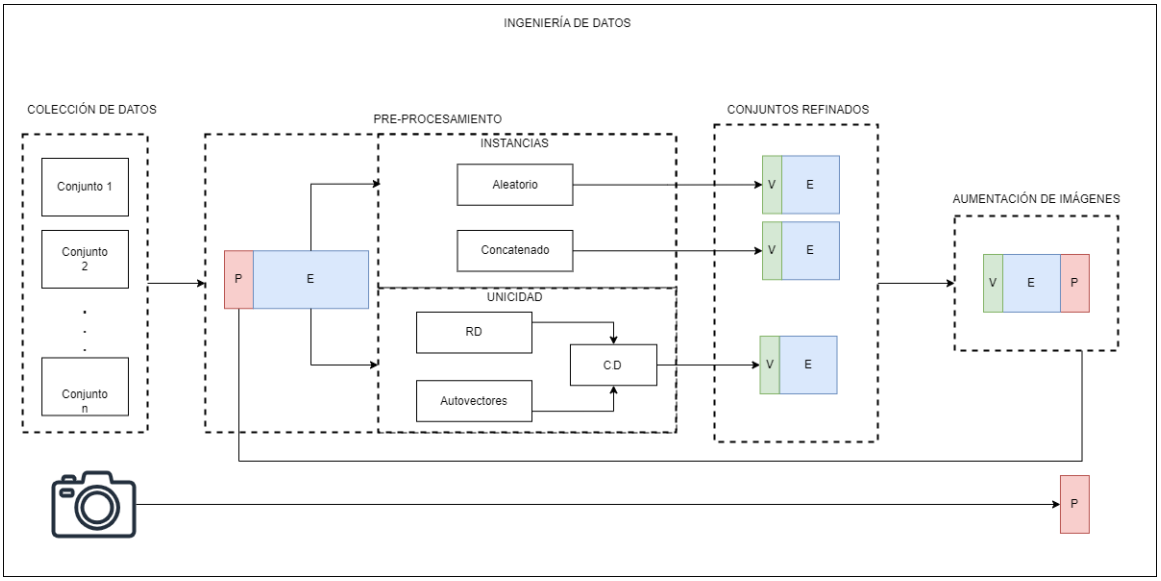


Figura 3.2: Flujo de trabajo en la etapa de Ingeniería de Datos

3.1.1.1. Colección de datos

Los dispositivos de borde se utilizan ampliamente en aplicaciones en diferentes contextos, como agricultura inteligente, detección remota, edificios/hogares inteligentes, detección de vida silvestre y operaciones industriales [76]. Se encontraron conjuntos de datos representativos para ser

entrenados por modelos de aprendizaje automático basados en estos dominios de aplicación. Los criterios de búsqueda fueron enfocados en que sean datos de libre acceso o repositorios de datos públicos, con diferente número de etiquetas y condiciones de toma de datos. Una parte de ellos son diseñados en entornos de laboratorio, mientras que los restantes han sido desarrollados en condiciones reales. En consecuencia, se emplearon cinco conjuntos de datos para el entrenamiento de los modelos de aprendizaje automático, en el Cuadro 3.1 se describen brevemente cada uno de ellos.

Nombre	**Aplicación de IoT**	**Descripción**	**Características**		
			Número de imágenes	Número de etiquetas	Dimensión
Detección de enfermedades en hojas	Agricultura Inteligente	Conjunto de datos empleado para diagnosticar enfermedades en diferentes cultivos. Para el propósito de esta investigación se han considerado las imágenes de plantas de tomate	11000	10 (Diferentes eneferme-dades en plantas de tomate)	256 x 256 x 3
Clasificación de residuos	Edificios Inteligentes	Conjunto de datos empleado para la clasificación automática de residuos en contenedores de basura, separándolos en orgánica o inorgánica según el caso	25000	2 (Orgánico e inorgánico)	256 x 256 x 3
Detección de aves	Detección de vida silvestre	Conjunto de datos del Dansk Ornitologisk Forening (Asociación Danesa de Ornitología) donde se cuenta con información actualizada de las especies de aves más representativas que han sido observadas en ese país	80000	18 (Diferentes tipos de aves)	224 x 224 x 3
Detección de grietas en paneles solares	Operaciones industriales	Conjunto de datos que busca detectar posibles imperfecciones en paneles solares provocadas durante los procesos de manufactura, transporte u operación. Una célula solar con grietas no contribuirá a la producción de energía por lo que su detección es importante para el mantenimiento de estos sistemas	2624	2 (paneles funcionales y dañados)	300 x 300 x 3
Imágenes satelitales	Sensores remotos	Conjunto de datos de la superficie terrestre, cuyo objetivo es determinar la presencia de hielo en zonas de difícil acceso	3500	7 (Imágenes con fondo vengetales, cultivos, agua, nieve y ciudades)	256 x 256 x 3

Cuadro 3.1: Descripción de los conjuntos de datos usados

En las Figuras 3.3, 3.4, 3.5,3.6 y 3.7 se presentan algunos ejemplos de las imágenes disponibles en los conjuntos de datos.

(a) Hojas secas (b) Hojas sanas

Figura 3.3: Conjunto de datos de detección de enfermedades en hojas

(a) Residuos inorgánicos (b) Residuos orgánicos

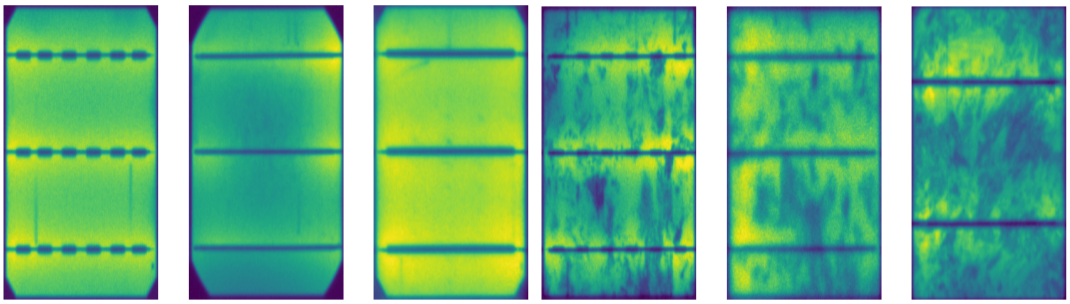
Figura 3.4: Conjunto de datos de clasificación de residuos

(a) Downy Woodpecker (b) Eared Pita

Figura 3.5: Conjunto de datos de aves danesas

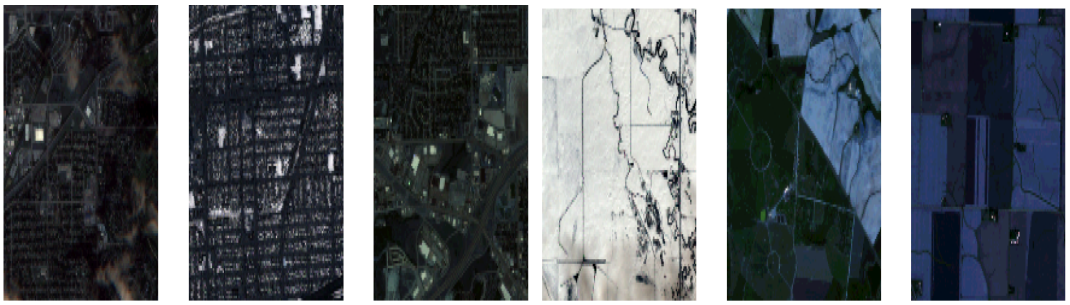
3.1.1.2. Preprocesamiento de datos

Una vez definidos los conjuntos de datos para esta investigación, el principal criterio de la etapa de preprocesamiento de datos es brindar al modelo de aprendizaje automático toda la información posible para que puedan ser entrenados con el objetivo de realizar predicciones y la clasificación. Teniendo la mayor cantidad de aciertos en la toma de decisiones con el menor número de datos posible en su entrenamiento. Como parte del preprocesamiento se emplearán técnicas de reducción de dimensionalidad, selección de prototipos y detección de datos atípicos.



(a) Buen estado (b) Mal estado

Figura 3.6: Conjunto de datos de paneles solares



(a) Fondo artificial (b) Vegetación

Figura 3.7: Conjunto de datos de imágenes satelitales

3.1.1.3. Conjuntos refinados

El preprocesamiento de datos se enfoca en observar las características principales de cada imagen y determinar su relevancia o aporte al modelo de aprendizaje automático. Es decir, datos repetidos o que no varíen en demasía, genera redundancia en la información, ocasionando que si el conjunto de datos no está balanceado (muy general en casos reales) exista un sobre entrenamiento sobre la etiqueta que tenga mayor número de instancias. Por lo tanto, si se observa la calidad de los datos, se puede reducir su tamaño sin disminuir el conocimiento intrínseco que contienen. A pesar que este criterio es muy utilizado en datos de series temporales o estáticos, en el procesamiento de imágenes sigue siendo un desafío abierto ya que al ser datos multidimensionales, determinar la relevancia de cada instancia puede generar sesgos en la información obtenida. Ya que para emplear los criterios de selección de características, es necesario reducir la dimenisonalidad de las imágenes. En consecuencia, se emplea la reducción de dimensionalidad con la utilización de los algortitmos PCA y t-SNE a los conjuntos de datos para luego aplicar las técnicas de selección de características. Éstas pueden enfocarse en eliminar los datos atípicos o seleccionar lo más adecuados para el modelo de ML. Una vez que el conjunto de datos ha sido optimizado, se puede aplicar aumentación sintética de datos para simular los entornos reales a los que el dispositivo de borde se enfrentará. Algunos conjuntos de datos, al ser creados en laboratorio, carecen de ciertas variaciones de exposición de luz o vibraciones que son muy relevantes para contar con un modelo de ML adaptado a su entorno.

3.1.1.4. Aumentación de datos

Como se mencionó en la sección 2.1.1.1, la aumentación de datos permite generar nuevas muestras sintéticas de datos a partir de las muestras originales. Por ello, una vez que los conjuntos de datos han sido preprocesados, se procede a aplicar técnicas de aumentación para asegurar que el

conjunto de datos contenga las variaciones existentes en los entornos IoT. En este escenario, cada conjunto de datos puede ser aumentado utilizando algunas transformaciones. Esto enriquece al conjunto de datos debido a que provee imágenes al modelo de aprendizaje automático con diferentes ángulos, exposición de luz, entre otros. Con el objetivo de encontrar equilibrio en los resultados de clasificación, se usa el criterio de validación cruzada, el cuál se enfoca en elegir aleatoriamente el 80 % de los datos para el conjunto de entrenamiento y validación y el restante 20 % para el de prueba. El Cuadro 3.2 muestra los métodos de aumentación de imágenes generalmente empleados.

Método de aumentación	**Parámetro configurado**	**Descripción del método**
Rotación	rotation_range=40	Permite rotar las imágenes en un rango de -40 a 40 grados, dando mayor variabilidad al conjunto de datos
Zoom	zoom_range=0.1	Permite ampliar o alejar las imágenes en un rango de 90 % al 110 %, esto ayudará a que el modelo sea capaz de reconocer imágenes en diferentes escalas
Desplazar ancho	width_shift_range=0.1	Permite desplazar las imágenes horizontalmente en un rango que va desde el -0.1 a 0.1 veces del ancho original de la imagen
Desplazar altura	height_shift_range=0.1	Permite desplazar las imágenes verticalmente en un rango del 10 % de la altura original de la imagen
Ajuste aleatorio del brillo	brightness_range=[0.4, 1.5]	Permite ajustar el brillo de las imágenes aleatoriamente dentro de un rango que va desde 0.4 hasta 1.5 veces el brillo original de la imagen
Girar horizontalmente	horizontal_flip=True	Permite que las imágenes puedan ser giradas horizontalmente de forma aleatoria
Girar verticalmente	vertical_flip=True	Permite que las imágenes puedan ser giradas verticalmente de forma aleatoria

Cuadro 3.2: Métodos de aumentación de datos

3.1.2. Fase 2: Gestión de modelos

Esta fase se enfoca en mostrar las diferentes formas de emplear modelos pre-entrenados para afinarlos a una nueva aplicación y cómo se pueden optimizar para realizar las inferencias del modelo o re-entrenarlos en los dispositivos de borde. Primero, se establecen las diferentes configuraciones con las que el modelo de aprendizaje automático será entrenado, éstas serán explicadas más adelante en la sección 3.1.2.2, tomando en cuenta sus hiperparámetros de configuración. Posteriormente, se describirá el entorno donde los modelos han sido entrenados y los resultados parciales obtenidos al aplicar la ingeniería de datos presentada en la sección anterior. Una vez entrenados los modelos, se emplean dos formas para optimizarlos, la primera es la cuantización, que se encarga de reducir la resolución de los modelos de 32 a 8 *bits* para que puedan ser ejecutados en arquitecturas computacionales con capacidades limitadas. Posteriormente, se emplea el aprendizaje por destilación del conocimiento, que busca entrenar submodelos (usualmente modelos hijos) de un modelo complejo. Con ello, se reduce su complejidad pero mantiene la arquitectura en cuanto a la extracción de características y clasificación. Luego, se observará el comportamiento de los submodelos cuantizados en dispositivos de borde. La Figura 3.8 muestra gráficamente cada paso previamente descrito.

3.1.2.1. Diseño del modelo

Los modelos de aprendizaje automático obtenidos mediante Aprendizaje por Transferencia contienen las arquitecturas CNN de ResNet50, Inception-V3, EfficientNetB0, MobileNet y VGG16. Estas arquitecturas han sido empleadas debido a que han sido ampliamente estudiadas para la cla-

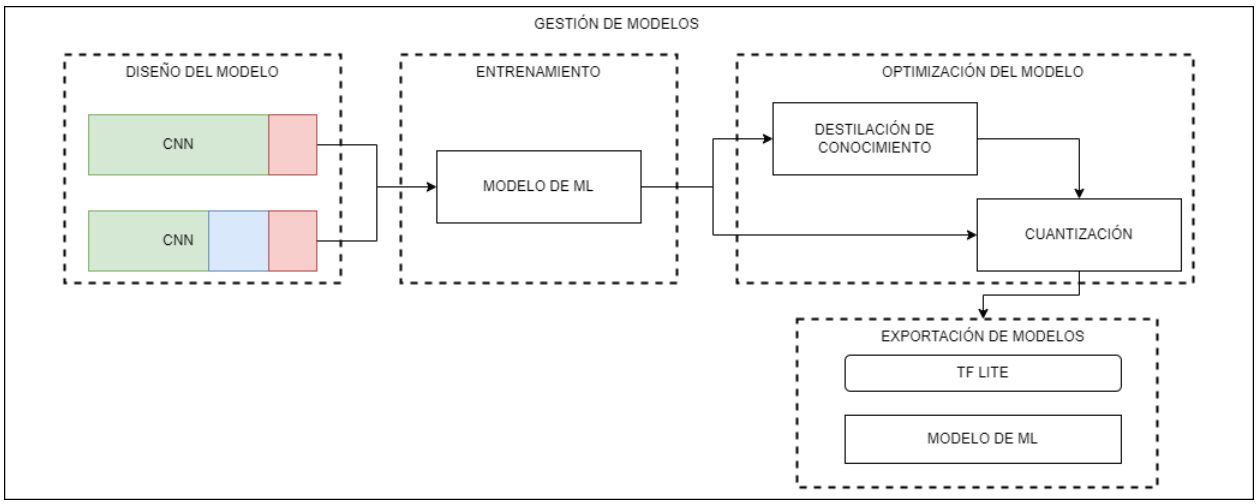


Figura 3.8: Flujo de trabajo en la etapa de gestión de modelos

sificación de imágenes y también debido a que su tamaño es considerado mediano o liviano. Cada una de ellas cuentan con parámetros específicos como la dimensión de la imagen y las diferentes formas de normalizarlas. Por ello, cada modelo de aprendizaje automático debe conservar sus propias características para ser afinado al nuevo conjunto de datos y no tenga errores de compilación. Como primer paso, las imágenes de entrada que se emplearán para re-entrenar los modelos de ML, son organizadas en lotes de 32 imágenes. Estos lotes seleccionan aleatoriamente imágenes de cada etiqueta. Como segundo paso, se establece que cada modelo será entrenado durante un máximo de 25 épocas, con una interrupción en el entrenamiento cuando se detecte que el rendmiento de clasificación no mejore durante las siguientes 5 épocas. Cuando esto suceda, el modelo deja de ser entrenado y se guarda automáticamente aún sin haber terminado el número máximo de épocas establecidas. En cada época, todo el conjunto de datos pasa a través de la red neuronal (conocido en inglés como *steps per epoch*) o iteración. Como entorno de programación, se ha seleccionado *TensorFlow* debido a su robustez en librerías y su capacidad para optimizar los modelos de ML. Para evaluar el rendimiento de cada modelo, se establecen las siguientes métricas:

- Exactitud
- Tasa de Error
- *Recall*
- *F1-Score*

Como métricas relacionadas a los dispositivos de borde:

- Tamaño del modelo
- Tiempo de ejecución
- Consumo de potencia

3.1.2.2. Entrenamiento

Para el entrenamiento de cada modelo de aprendizaje automático, se han usado sus modelos pre-entrenados (Aprendizaje por Transferencia) y se han realizado las siguientes configuraciones:

- Configuración 1: A cada uno de los modelos se ha removido el clasificador añadido por defecto y se ha integrado una capa totalmente conectada *(Fully Connected Layer)* con el número de neuronas correspondiente al número de etiquetas de cada conjunto de datos.
- Configuración 2: El modelo es descongelado x capas para afinarlo al conjunto de datos del entrenamiento. Luego se añade una capa conectada *(Connected Layer)* con el número de neuronas correspondiente al número de etiquetas de cada conjunto de datos.
- Configuración 3: A la Configuración 2, se añade la aumentación de datos para ampliar el conjunto de datos que alimenta a la red neuronal.

Los modelos son entrenados en un servidor de altas prestaciones con varios nodos computacionales con las siguientes características: 2x Intel(R) Xeon(R) Gold 6242 CPU @ 2.80GHz Cores: 64, RAM: 384 GiB y una GPU 6x Tesla V100 PCIe 32 GiB.

3.1.2.3. Optimización del modelo

Los algoritmos de aprendizaje automático una vez entrenados deben ser exportados en un paquete de compilación para que sea el encargado de hacer inferencias en los dispostivos de borde, esto se debe a que su arquitectura puede procesar generalmente datos entre 16 u 8 *bits*. Además, bajo la premisa que los modelos que utilizan Aprendizaje por Transferencia emplean arquitecturas CNN muy complejas y con el objetivo de re-entrenar los modelos en los dipostivos de borde, reducir su complejidad para ajustarlos al fenómeno estudiado puede acelerar su proceso de entrenamiento, reduciendo su coste computacional. Como se ha mencionado anteriormente, la cuantización permite reducir la resolución del modelo entrenado con variables flotantes, de 32 a 8 *bits*. No obstante, cabe recalcar que este proceso se puede llevar a cabo solo para hacer inferencias del modelo. Por su parte, el aprendizaje por destilación permite entrenar modelos subyacentes de menor tamaño pero con la misma arquitectura. Esto permite tener un modelo liviano que se puede ajustar a las capacidades computacionales del dipostivo de borde.

3.1.2.4. Evaluación

Con el objetivo de determinar la mejora del rendimiento de los modelos de aprendizaje automático al aplicar la arquitectura propuesta, cada modelo fue entrenado en su forma estándar. Es decir, se importó el modelo mediante Aprendizaje por Transferencia y empleó el conjunto de datos original obtenido desde su repositorio inicial. De esta manera, se estableció una línea base del rendimiento general del modelo, que se convierte en nuestra métrica a superar al aplicar la nueva arquitectura.

El rendimiento del modelo se evaluó mediante las métricas de exactitud, tasa de error, *recall* y F1-*Score*. Especialmente se analizó su capacidad para asignar correctamente etiquetas a nuevas instancias, utilizando el conjunto de prueba. La métrica de *recall* del clasificador, se enfoca en presentar cuán equilibradas son sus decisiones. La métrica *F1-Score*, que representa la media armónica de la exactitud y el *recall*, indica si existe un sesgo hacia las variables con un mayor número de muestras en conjuntos de datos no balanceados, como es nuestro caso. Todas estas métricas se obtienen de la matriz de confusión.

3.1.3. Fase 3: Despliegue de Modelos

Una vez definidos los modelos de aprendizaje automático adecuados para ser desplegados en los dispositivos de borde, su implementación debe cumplir con los requisitos de funcionamiento específicos. Para ello, se describen los dispositivos de borde más comunes y sus aceleradores de

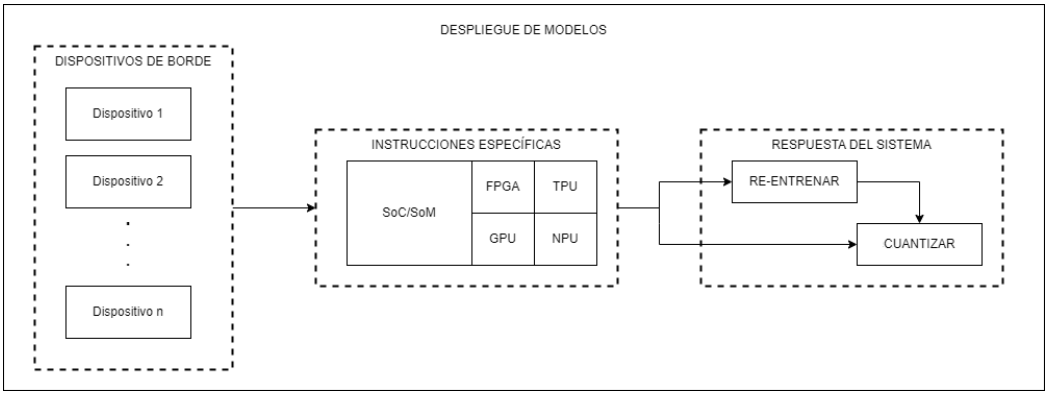


Figura 3.9: Flujo de trabajo en la etapa de despliegue de modelos

hardware, que permiten reducir el tiempo de procesamiento durante el proceso de inferencias o el re-entrenamiento del modelo. Finalmente, mediante métricas como la exactitud, la tasa de error, el *recall*, el *F1-Score*, el tamaño del modelo, el tiempo de ejecución y el consumo de potencia, se determinan los modelos de ML y las formas óptimas de desplegar estos algoritmos en los dispositivos de borde. La Figura 3.9 muestra cada uno de los pasos descritos previamente.

3.1.3.1. Dispositivos de borde

Los modelos entrenados, con su cuantización a 8 *bits* son exportados al dispositivo de borde para que luego de usar instrucciones de *hardware* específicas, se reduzca el tiempo de ejecución al inferir nuevas imágenes con su etiqueta. Cabe recalcar que los aceleradores de *hardware*, usualmente emplean arquitecturas de 8 *bits*.

3.1.3.2. Instrucciones específicas

Como se indicó anteriormente, al ejecutar los modelos entrenados en dispositivos de borde que cuentan con una unidad de procesamiento gráfica (GPU), es necesario hacer uso de bibliotecas específicas para este propósito. De manera puntual, se debe trabajar con TensorRT, que es una biblioteca de inferencia desarrollada por NVIDIA, cuyo objetivo es optimizar redes neuronales para que sean compatibles con sus GPUs [93]. Para la optimización del modelo, TensorRT permite convertir los modelos de 32 *bits* a 16 y 8 *bits*. La reducción del tamaño del modelo afecta directamente el uso de memoria y por lo tanto el consumo de energía, sin afectar significativamente la precisión. Además, empleando TensorRT es posible acelerar la inferencia de los modelos, reduciendo la latencia y aumentando el rendimiento hasta 10 veces en comparación al modelo de tamaño de 32 *bits*.

Para el caso de dispositivos con unidades centrales de procesamiento (CPU) y unidades de procesamiento de tensores (TPU), la biblioteca a emplear es *TensorFlow Lite*, por lo que una vez entrenados los modelos, para el despliegue en dispositivos de borde deberán contar con ese formato. Para ejecutar la inferencia en una TPU, Google ha diseñado ciertas instrucciones por *software* para compilar el modelo. Esta información se encuentra disponible en la paǵina oficial del producto Coral.ai.

3.1.3.3. Respuesta del sistema

Esta etapa proporciona información sobre cómo afecta perder la resolución del modelo que ha sido entrenado con variables de tamaño de 32 *bits* y al procesar imágenes en condiciones reales, donde el modelo de aprendizaje automático ahora las procesa a 8 *bits*. Los pasos son los siguientes:

- Re-entrenamiento del modelo: Debido a que los dispositivos de borde cuentan con CPUs de varios núcleos, no solo están diseñados para inferir sobre modelos ya entrenados en el servidor, los modelos obtenidos por la desitilación de conocimiento pueden ser exportados al dispositivo en un formato que permite re-entrenarlos. Este proceso no se podrá realizar en el acelerador de *hardware* ya que no están diseñados para esta tarea.
- Cuantización: Este proceso dentro del dispositivo de borde tiene dos funcionalidades. La primera, es recibir desde el servidor el modelo cuantizado y enviarlo al acelerador de *hardware* para su ejecución cuando el dispositivo tome nuevas imágenes. La segunda, cuando el modelo haya sido re-entreado en el propio dispositivo, se debe cuantizar el modelo para emplearlo solo para realizar inferencias y disminuir el coste computacional requerido para realizar inferencias con todo el modelo de ML.

CAPÍTULO 4: RESULTADOS

Una vez que se ha completado el diseño de la arquitectura y se han definido sus etapas, es necesario validarla en aplicaciones y escenarios reales. Aunque existen diversas aplicaciones de IoT, en este capítulo se presentan dos casos de estudio que contribuyen a la validación de la hipótesis. El primer caso de estudio se centra en la clasificación de imágenes satelitales; mientras que el segundo aborda la clasificación de imágenes en invernaderos. En ambos casos se han analizado los tiempos de respuesta del sistema con el objetivo de determinar qué dispositivo de borde se ajusta mejor a los requerimientos específicos de cada caso.

4.1. Introducción

Una vez presentada la arquitectura de datos acelerada en dispositivos de borde, el análisis de datos realizado con los conjuntos de datos en entornos IoT, descritos en la sección 3.1.1.1 es el siguiente:

- Todas las bases de datos fueron empleadas para entrenar los modelos de ML con las configuraciones mencionadas. Este proceso se lleva a cabo para conocer cuál de estas configuraciones es la adecuada para afinar dichos modelos hacia las características de cada conjunto de datos. Para determinar la mejor configuración, las métricas de mayor relevancia son: el tiempo de entrenamiento, el tamaño del modelo y el rendimiento de clasificación.
- Una vez entrenados los modelos y seleccionada su configuración de entrenamiento, son optimizados por medio de la cuantización. Para ello, cada modelo de ML es llevado a su versión de *TensorFlow Lite* que solo puede inferir la etiqueta de los datos considerados como prueba y que no puede ser re-entrenado. Como resultado, los modelos de ML fueron optimizados a 16 y 8 *bits* para conocer su rendimiento de clasificación. Se pudo comprobar que la versión de 16 *bits* no representa una reducción significativa en el tamaño del modelo y que con posterioridad, este modelo no puede ser desplegado en los dispositivos de borde por sus limitaciones computacionales. Cabe recalcar, que estos resultados parciales son la referencia para superar o igualar durante la aplicación de la arquitectura propuesta.
- Los conjuntos de datos de entornos IoT se optimizan considerando dos criterios principales: el número de instancias y las características, con el objetivo de reducir su tamaño. En cuanto al criterio basado en número de instancias, se selecciona aleatoriamente imágenes de cada etiqueta sin emplear ningún algoritmo adicional. Por otro lado, la reducción del número de imágenes basada en la selección de sus características más relevantes, emplea la selección de prototipos y la detección de datos atípicos con sus algoritmos más empleados y que han demostrado ser los más adecuados para este tipo de datos, los cuales son: *One-Class SVM* e

Isolation Forest (detección de datos atípicos) y All-kNN y c-NN (selección de prototipos). Es importante destacar que para aplicar estos algoritmos, como paso previo, es necesario reducir la dimensionalidad. Para esto, se emplea PCA, t-SNE y autovectores.

- Una vez que los conjuntos de datos IoT han sido optimizados, los diferentes modelos de ML se entrenan con la configuración establecida para seleccionar los algoritmos de optimización de datos que más se ajusta a la arquitectura propuesta.
- Los modelos de ML han sido entrenados con todo el conjunto de datos y sus versiones optimizadas. Con esto, se puede analizar qué arquitectura CNN puede ser empleada en la arquitectura propuesta, sin que decaiga su rendimiento, aunque se emplee un conjunto de datos liviano.
- Con las arquitecturas CNN definidas, se analizan en profundidad para generar modelos livianos que mantiene la misma funcionalidad de capas convolucionales pero aplicando un número menor de filtros y neuronas (destilación de conocimiento). Estos modelos pueden ser entrenados desde cero o desde la versión ya entrenada con la configuración establecida en la arquitectura propuesta. Se comparan los resultados de clasificación de estos modelos livianos con respecto a sus modelos originales para comprobar sus ventajas. Luego, estos modelos livianos son re-entrenados con los conjuntos de datos livianos obtenidos en los puntos anteriores.
- Establecida la configuración de entrenamiento de los modelos de ML, la forma de optimizarlos al aplicar el conocimiento por destilación y su forma cuantizada para realizar solo inferencias, se aplica a los casos de estudio, con el conjunto de datos reducido en dos aplicaciones IoT reales con requerimientos computacionales definidos.

4.2. Configuraciones de entrenamiento de los modelos de aprendizaje automático

Los conjuntos de datos han sido entrenados con las 3 configuraciones establecidas en el capítulo anterior: 1. Solo añade una capa de clasifación sobre un modelo pre-entrenado, 2. Se descongelan n capas en los modelos para afinarlo a cada conjunto de datos y 3. Se emplea la aumentación de datos en conjunto con la Configuración 2. Se evaluó el rendimiento de cada arquitectura CNN y el tiempo de ejecución para entrenarse con cada configuración.

Todos los modelos fueron entrenados en un servidor de alto rendimiento, que tiene varios nodos informáticos descritos a continuación: 2 CPU Intel(R) Xeon(R) Gold 6242 a 2,80 GHz Núcleos: 64, RAM: 384 GiB y acelerador de *hardware*: GPU 6x Tesla V100 PCIe 32 GiB. Respecto Configuración 1, cada modelo se ha entrenado de forma muy rápida en comparación con las dos configuraciones restantes. Sin embargo, ningún modelo de aprendizaje automático ha alcanzado un promedio de rendimiento superior al 90 %. MobileNet tiene un rendimiento promedio del 70 % y el resto de modelos cerca del 80 %. Por su parte, en la Configuración 2, el rendimiento de los modelos de EfficientNet e Inception-V3 cuentan con rendimientos sobre el 90 %. Por su parte, MobileNet disminuye significativamente su rendimineto por debajo, en algunos casos, del 60 %. Resnet50 y VGG16 tienen altos rendimientos de clasificación, pero con tiempos de entrenamiento muy altos respecto al resto de modelos, especialmente VGG16 donde puede sextuplicar el tiempo de entrenamiento que los demás modelos entrenados.

En relación a la Configuración 3, todos los modelos de aprendizaje automático experimentaron un aumento del 35 % su tiempo de entrenamiento, mientras que su rendimiento disminuyó en un

promedio del 8 %. Esto se debe a que el aumento de datos resulta significativo cuando los conjuntos de datos se han recolectado en condiciones reales, en contraste con entornos controlados, como es el caso de la mayoría de estos conjuntos. Esto demuestra la disparidad existente entre las bases de datos y las condiciones reales. Por otro lado, si las condiciones ambientales no son muy variantes, esta técnica puede no ser tomada en cuenta debido a la robustez del modelo de aprendizaje automático. La Figura 4.1 muestra los resultados obtenidos en cada configuración y con los 5 conjuntos de datos, en el eje vertical izquierdo se muestran los valores del rendimiento de clasificación y en el eje vertical derecho se presenta el tiempo de ejecución.

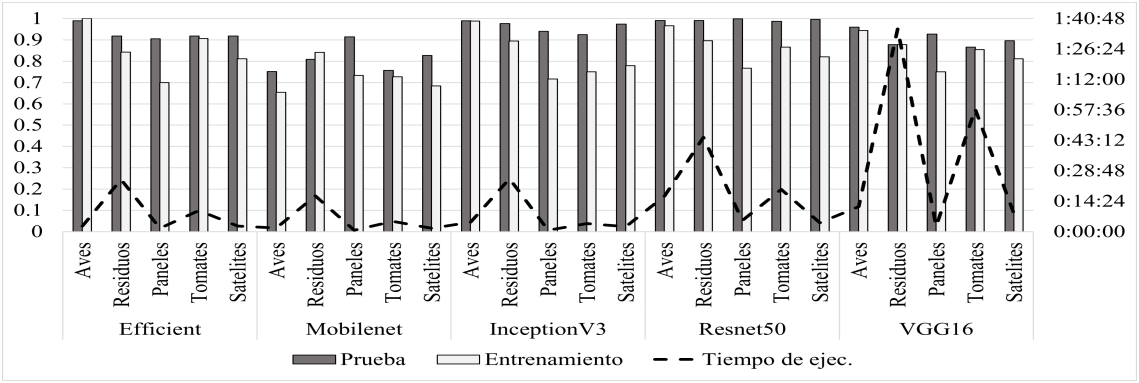
Finalmente, se ha optado por continuar con la Configuración 2, que implica el descongelamiento de capas sin emplear aumentación de datos. Esta configuración es la que mejores resultados ha obtenido en términos de rendimiento y tiempo de ejecución. Se llevaron a cabo varias pruebas para determinar el número óptimo de capas a descongelar, y los resultados sugieren que solo las últimas 20 capas proporcionan un equilibrio entre un alto rendimiento y un tiempo de entrenamiento aceptable.

4.3. Gestión de Modelos

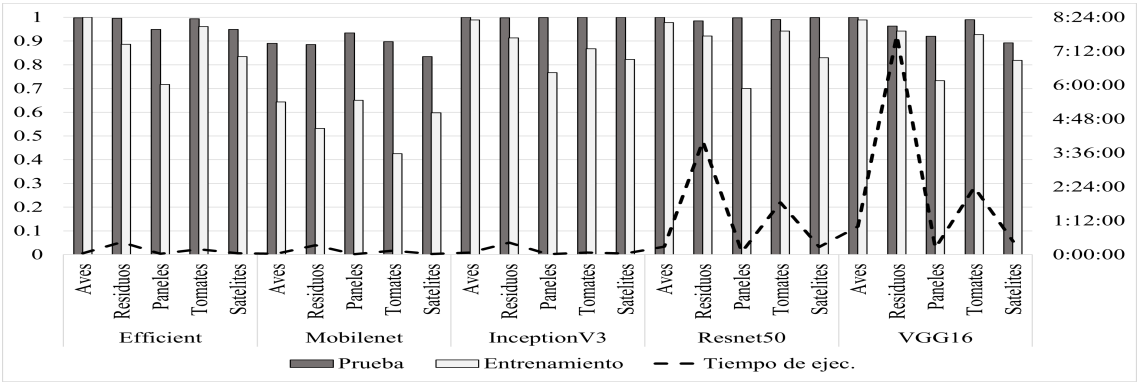
Esta sección, se enfoca en la selección de modelos de aprendizaje automático que se adapten mejor a la naturaleza de los datos en entornos IoT y en cómo pueden ser exportados a dispositivos de borde. Como primer paso, cada modelo entrenado con la Configuración 2 se exporta en su versión cuantizada a 8 *bits*. Esta versión solo puede ejecutar inferencias sobre nuevas imágenes sin la necesidad de ser re-entrenado el modelo. Usualmente, este archivo se guarda en formato *.hdf5*, pero debido a que se utiliza el *framework TensorFlow*, también puede guardarse como un archivo de extensión *.keras* para ser utilizado en diferentes dispositivos. Se utilizan las librerías provenientes de *TensorFlow Lite* para este propósito. Con esto, se pueden obtener los resultados generales de cada modelo con respecto a los conjuntos de datos optimizados previamente descritos.

Por un lado, los modelos guardados en la versión de *TensorFlow Lite* pueden ser posteriormente optimizados con diferentes técnicas; una de ellas es la cuantización, que decrementa la resolución del modelo al reducir el número de *bits* de cada variable para disminuir el tamaño del archivo. Como se observó en el Capítulo 2, la optimización sobre el tamaño de cada variable, que normalmente es de 32 *bits*, puede reducirse a 16 u 8 *bits*, o una combinación de ambos. Por lo tanto, dado que las variables son de menor tamaño, el modelo puede ser compilado para arquitecturas computacionales de menor capacidad, con procesadores de 16 u 8 *bits*. Una vez exportado el modelo, se analiza su rendimiento con el mismo conjunto de prueba que se utilizó para evaluar el modelo de alta resolución para conocer su pérdida de rendimiento. Esto proporcionará información relevante para la selección de modelos de aprendizaje automático y cómo esta optimización los afecta al ser llevados a dispositivos de borde.

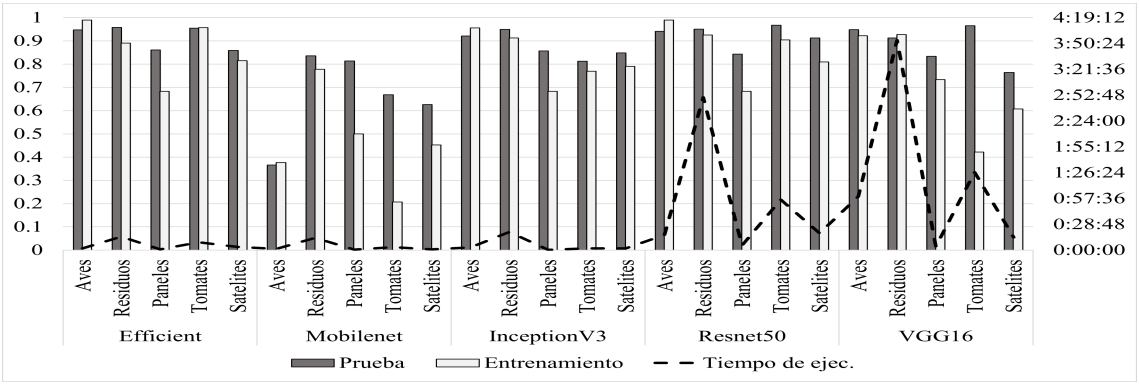
Por otro lado, aunque estos modelos sean optimizados en su resolución y, por ende, el peso del archivo se disminuya, la complejidad del modelo se mantiene debido a que cuenta con la arquitectura CNN con el mismo número de filtros, funciones de activación, entre otros. Para reducir esta complejidad, el método de destilación del conocimiento permite entrenar modelos de menor tamaño, llamados "hijos", desde el modelo obtenido por transferencia de conocimiento. Esto permite seleccionar las capas convolucionales que contienen los filtros que mejor se ajustan a cada conjunto de datos y eliminar el resto. No obstante, para validar que el modelo reducido tenga un mejor rendimiento al ser entrenado desde el modelo por transferencia de conocimiento en lugar de



(a) Configuración 1 de entrenamiento de los modelos de ML



(b) Configuración 2 de entrenamiento de los modelos de ML



(c) Configuración 3 de entrenamiento de los modelos de ML

Figura 4.1: Configuraciones de los algoritmos de ML para los conjuntos de datos IoT

entrenarlo desde cero, se realizaron comparaciones de estas dos formas de entrenamiento para elegir la más adecuada. Finalmente, se presentan dos casos de estudio donde se aplicará la metodología propuesta para identificar sus fortalezas y debilidades.

4.3.1. Optimización de modelos

En esta sección, se utilizan los modelos de aprendizaje automático entrenados con la Configuración 2. Luego, se optimizaron a 16 y 8 bits, donde solo son capaces de realizar inferencias. Para seleccionar adecuadamente el modelo de aprendizaje automático, se compara su rendimiento de clasificación con respecto a los 5 conjuntos de datos mencionados anteriormente. Posteriormente, se realiza una comparación del tamaño reducido para determinar los requisitos computacionales

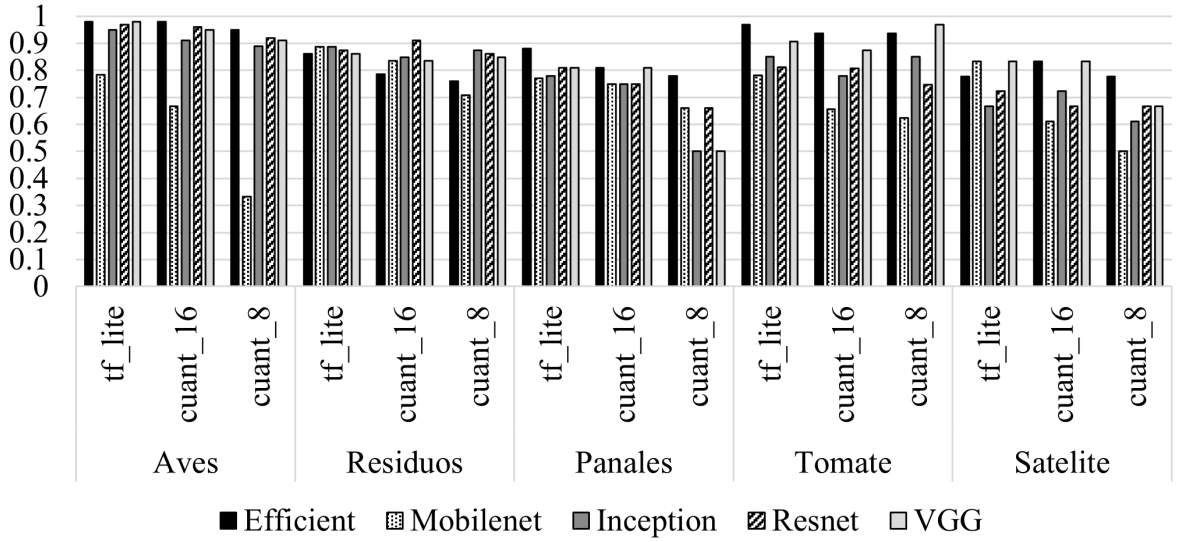


Figura 4.2: Análisis de rendimiento de clasificación de las versiones optimizadas de los modelos de ML

que deben cumplir los dispositivos de borde.

En la Figura 4.2, se muestra el rendimiento de clasificación de cada modelo de aprendizaje automático con una resolución de 32 *bits* (tf_{lite}), *16 bits* (cuant_{16}) y 8 *bits* (cuant_{8}). Se puede apreciar que solo los modelos obtenidos por MobileNet, tienen un rendimiento de clasificación significativamente menor en versiones cuantizadas. Muchos de estos modelos se encuentran por debajo del 50 % de rendimiento, convirtiéndose en modelos no aptos para ser desplegados en dispostivos de borde. Por su parte, los modelos cuantizados a 16 bits con respecto a los de 8 *bits*, no representan una significativa mejora y en promedio tienen entre 3 % y 7 % de pérdida de rendimiento. Además, empleando modelos que usan 16 *bits* de resolución, algunos dispositivos de borde puede tener limitaciones al procesarlos o su acelerador de *hardware* no va a poder compilarlos. Por esta razón, no son viables para ser empleados en las siguientes etapas de la arquitectura propuesta. EfficientNet e Inception-V3 tienen el mejor rendimiento de clasificación, donde sus versiones cuantizadas no distan de los modelos sin optimizar. Esto se puede observar en la Figura 4.2.

Con el objetivo de evaluar la reducción del tamaño de cada modelo, la Figura 4.3 muestra el rendimiento promedio de cada modelo de aprendizaje automático en relación con la resolución y el tamaño en disco necesario. En la Figura 4.3, el eje vertical izquierdo representa la exactitud alcanzada por el modelo, mientras que el eje vertical derecho muestra el tamaño del modelo medido en MB. Como se puede observar, Inception-V3 y ResNet50 requieren al menos 80 MB para ser almacenados, VGG16 necesita 60 MB, EfficientNet alrededor de 20MB y MobileNet 9 MB. Sus versiones cuantizadas a 8 *bits* necesitan 23, 26, 14, 5 y 3 MB respectivamente. Esto demuestra que únicamente MobileNet reduce su tamaño en 3 veces, mientras que el resto de modelos lo hace 4 veces. Además, los modelos cuantizados tiene un rendimiento de clasificación del 80 %, mientras que las versiones de 32 *bits* 85 %.

4.4. Ingeniería de datos

En esta sección se muestra el proceso llevado a cabo para lograr conjuntos de datos refinados a partir de los conjuntos originales de aplicaciones de IoT, detallados en el Cuadro 3.1.

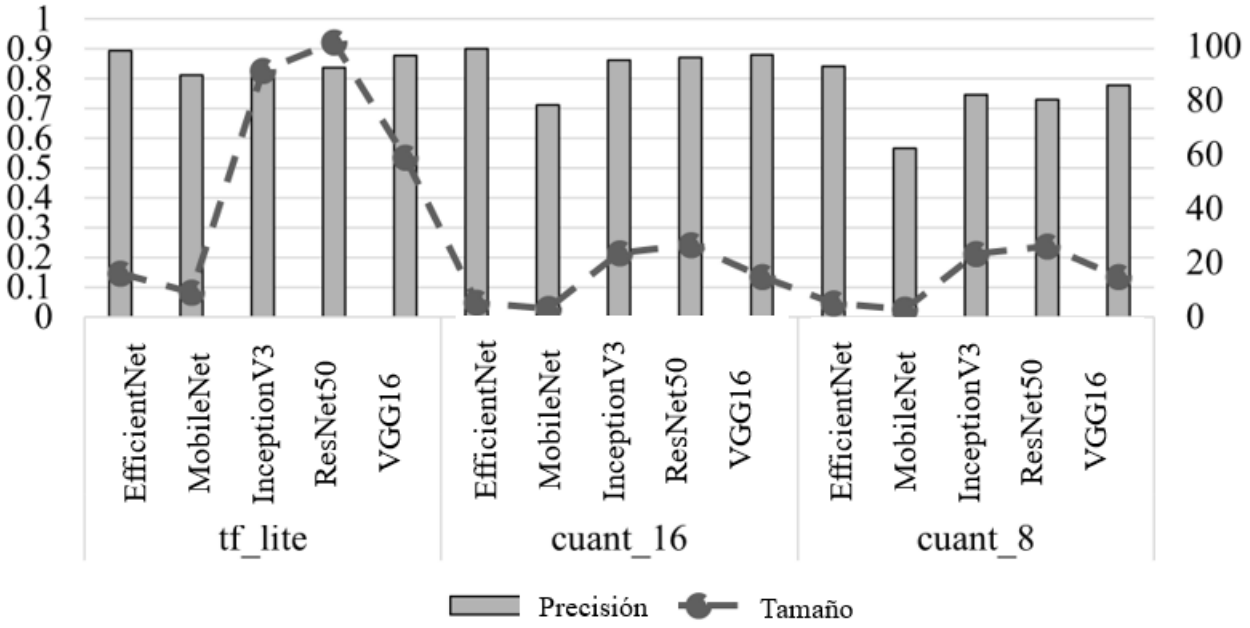


Figura 4.3: Comparación de rendimiento de clasificación vs. tamaño del modelo

4.4.1. Instancias

Para determinar la cantidad óptima de datos necesarios para re-entrenar los modelos de ML y ajustarlos adecuadamente en las configuraciones de entrenamiento descritas en el Capítulo 3, se han extraído subconjuntos de datos de las bases de datos de las aplicaciones IoT. Se han considerado porcentajes que van desde el 1 % (opción extrema), 10 %, 25 %, 50 % y 75 %. Para cada porcentaje, se han creado subconjuntos de datos de dos formas: en la primera, cada conjunto de datos se crea de forma aleatoria; en la segunda, cada subconjunto de datos alimenta al conjunto de datos de mayor tamaño de forma consecutiva. Esto se realiza para observar la calidad de los datos considerados en cada subconjunto. El Cuadro 4.1 muestra el tamaño de entrenamiento de los subconjuntos generados según los porcentajes establecidos

Conjunto de Datos	Subconjuntos					
	1 %	10 %	25 %	50 %	75 %	100 %
Detección de enfermedades	1.81MB	18MB	45.25MB	90.5MB	135.75MB	181MB
Clasificación de residuos	2.33MB	23.3MB	58.25MB	116.5MB	174.75MB	233MB
Detección de aves	6.5kB	6.5MB	16.25MB	32.5MB	48.75MB	65MB
Paneles solares	2.09MB	10.45MB	20.9MB	104.5MB	156.75MB	200MB
Imágenes satelitales	260KB	2.6MB	6.5MB	13MB	19.5MB	26 MB

Cuadro 4.1: Tamaño en MB de los subconjuntos de datos de los conjuntos de aplicaciones de IoT.

Una vez establecidos los subconjuntos y empleando la Configuración 2 de entrenamiento de modelos de aprendizaje automático, se observó que al seleccionar entre el 1 % y el 10 % del tamaño total de los conjuntos de datos resulta insuficiente para lograr un rendimiento de clasificación aceptable. No obstante, al seleccionar el 25 % de los datos, el rendimiento es considerablemente alto. Sin embargo, en algunos conjuntos de datos aún resulta insuficiente. Por lo tanto, al utilizar el 50 % de cada conjunto de datos, los modelos EfficientNet e Inception V3 demostraron ser los más adecuados, con rendimientos superiores al 90 % en la mayoría de los casos. Además, se observó que no había impacto significativo al adquirir los subconjuntos de datos de forma aleatoria o concate-

nada. En consecuencia, en futuras pruebas, se seleccionarán aleatoriamente.

Finalmente, se observó que al utilizar entre el 50 % y 75 % del conjunto total, el rendimiento de clasificación era similar al de utilizar todo el conjunto de datos. Esto sugiere, por un lado, la robustez de las arquitecturas CNN y, por otro, que los datos no varían considerablemente entre sí. La Figura 4.4 muestra los resultados obtenidos al generar subconjuntos de datos y entrenarlos con diferentes arquitecturas CNN. Cada subfigura muestra el rendimiento de clasificación por cada conjunto da datos que ha sido reducido de forma porcentual entre 1 % hasta 75 % de forma aleatoria y de forma concatenada. Finalmente, cada subfigura también hace referencia al tiempo de procesamiento que cada modelo necesitó para ser entrenado.

4.4.2. Características

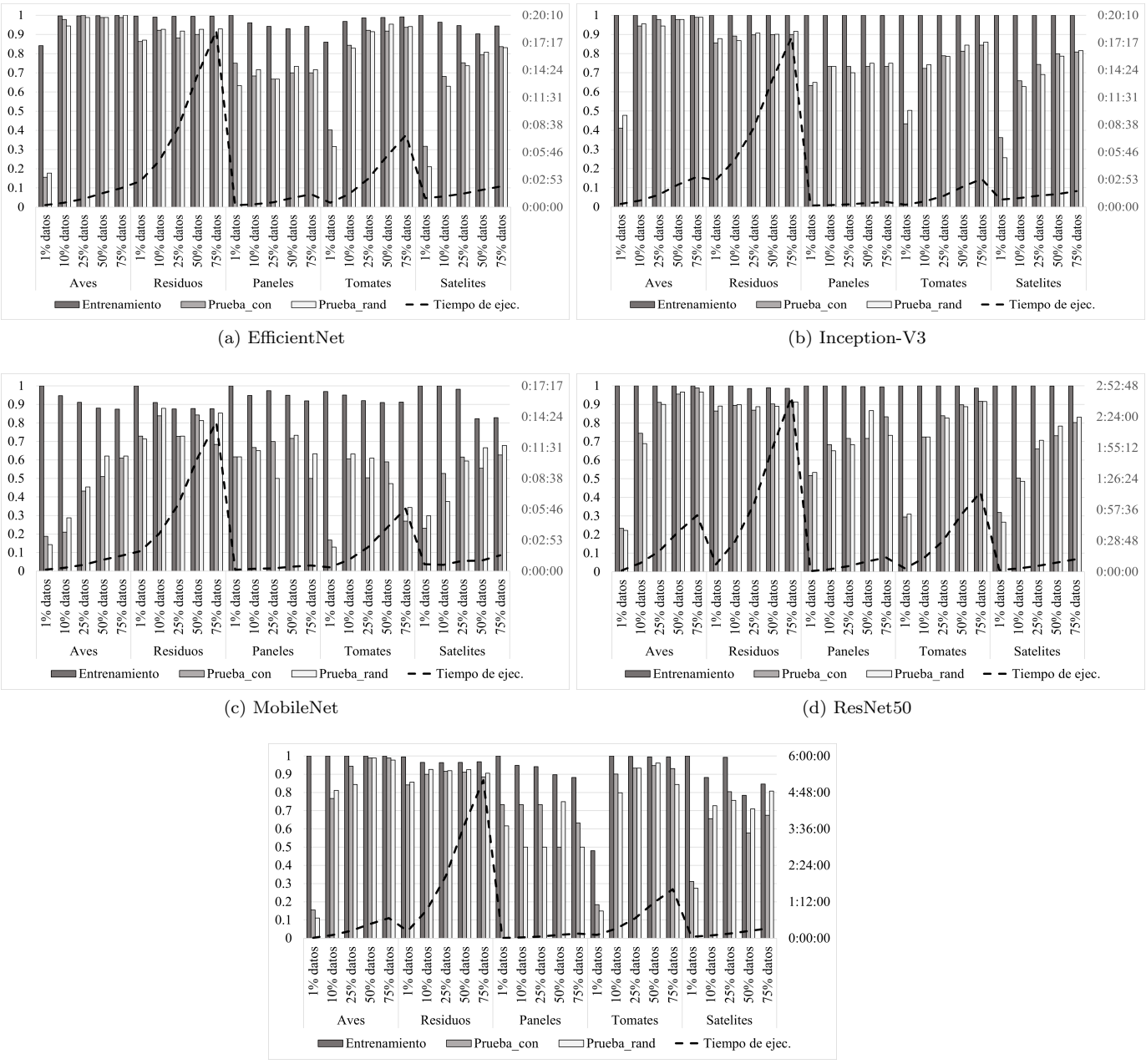
En este enfoque, el objetivo es encontrar un subconjunto de datos refinado que, por un lado, elimine los datos atípicos, es decir, aquellos con características diferentes al resto de los datos pertenecientes a la misma etiqueta. Por otro lado, se busca que el subconjunto de datos esté conformado por datos que aporten información relevante al clasificador, conocido como selección de prototipos. Esto implica seleccionar los datos más cercanos al centroide de cada etiqueta, lo que hace que los bordes de decisión sean más evidentes.

Para lograr esto, las imágenes de cada conjunto de datos deben pasar por algunas etapas previas. Primero, es necesario reducir la dimensionalidad del conjunto de datos, debido a que los algoritmos utilizados para este propósito no trabajan con matrices multidimensionales, como son las imágenes, sino con vectores. Dado que la mayoría de los conjuntos de datos contienen imágenes RGB (*Red, Green, Blue*), que son matrices tridimensionales donde cada una contiene 254 x 254 píxeles, se convierte cada imagen a escala de grises para obtener una sola matriz de 254 x 254. Luego, esta matriz se convierte en un vector concatenando secuencialmente cada fila de la matriz de escala de grises.

Sin embargo, dado que este conjunto de datos puede ser muy extenso y podría afectar computacionalmente su despliegue en entornos IoT, y con el objetivo de tener representaciones visuales para observar cómo se seleccionan los datos, se propone reducir la dimensión a un vector de dos dimensiones. Estos vectores pueden ser utilizados para aplicar algoritmos como PCA y t-SNE. Como resultado, se obtendrán las componentes principales de cada imagen en un vector con dos columnas, las mismas que se pueden emplear para realizar un gráfico y observar la distribución de los datos.

Por otro lado, el uso de autovectores emplea una arquitectura CNN para extraer las características en un vector especificado por el usuario. Esta técnica es diferente y no requiere los pasos previamente mencionados, ya que utiliza las imágenes de cada conjunto de datos y las representa en una dimensión menor. En nuestro caso, se emplea un autovector de 2 posiciones que también se puede representar en un gráfico. Es importante destacar que este proceso implica cierta pérdida de información al eliminar una gran cantidad de detalles. Las Figuras 4.5 y, 4.6 ejemplifican el proceso comentado, en donde cada color representa una etiqueta. Como se puede apreciar gráficamente, PCA intenta comprimir a todas las etiquetas en una sola ubicación, lo cuál hace que sea un algoritmo que reduce las diferencias de cada etiqueta y que ocasionará confusiones en el clasificador. Motivo por el cual, no es tomado en cuenta en los siguientes pasos de la arquitectura propuesta.

Por su parte, el algoritmo t-SNE busca mantener las características representativas de cada



(a) EfficientNet

(b) Inception-V3

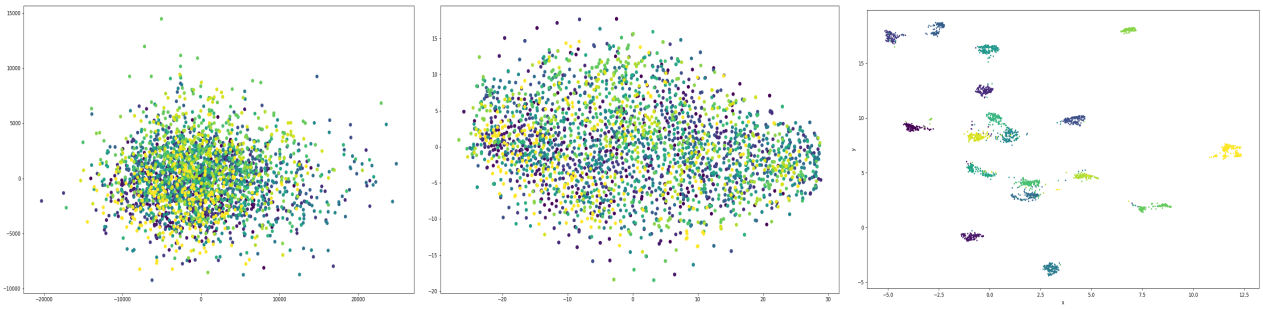
(c) MobileNet

(d) ResNet50

(e) VGG16

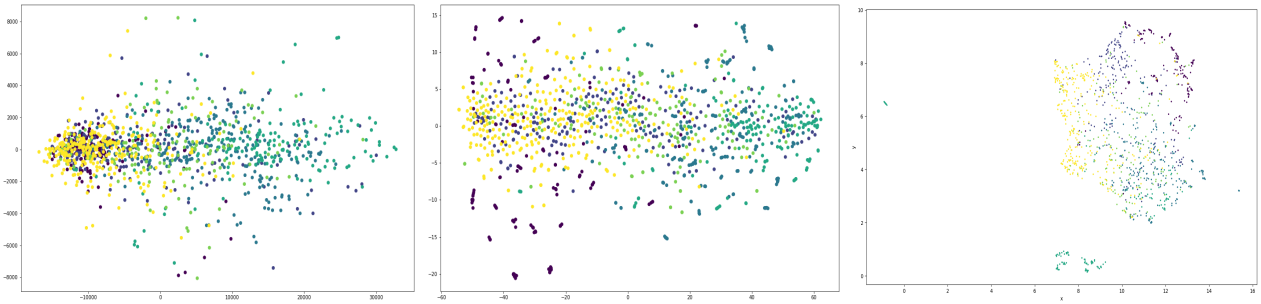
Figura 4.4: Rendimiento de clasificación de subconjuntos de datos (desde 1 % hasta 75 %) con diferentes modelos de ML. Eje vertical derecho se refiere al tiempo de entrenamiento.

etiqueta y las presenta en diferentes ubicaciones. Esto se puede apreciar claramente en la Figura 4.7, donde a pesar de que el conjunto de datos tiene 10 etiquetas, se observa cómo cada etiqueta está distante de las demás. Sin embargo, en algunas bases de datos, las etiquetas pueden confundirse en un plano bidimensional, aunque siguen siendo más claras que con PCA. Finalmente, los autovectores, al no tener que procesar los datos con diferentes técnicas y utilizar directamente las imágenes, tienen un mejor rendimiento al mantener las características de cada etiqueta. En las Figuras 4.5, 4.6 y 4.7 se aprecia cómo cada etiqueta ocupa una ubicación clara en el espacio bidimensional. Esto facilita que los algoritmos de selección de características puedan eliminar datos atípicos o seleccionar los datos más relevantes.



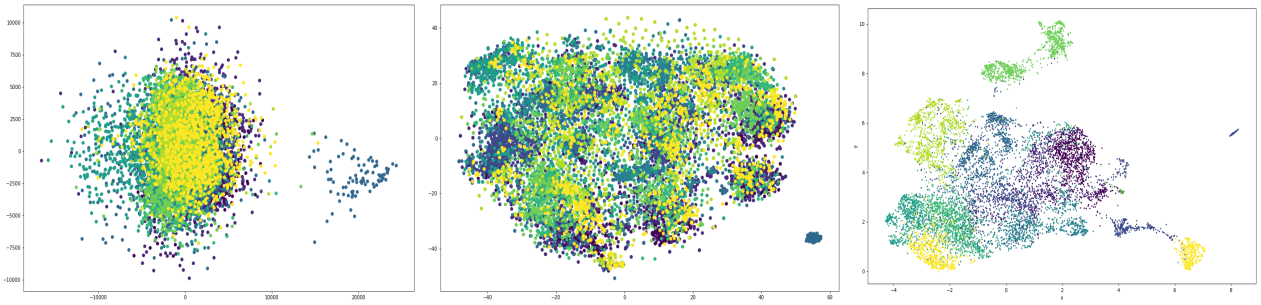
(a) PCA. (b) t-SNE. (c) AUTOVECTORES.

Figura 4.5: Conjunto de datos de aves danesas representadas en un espacio bidimensional



(a) PCA. (b) t-SNE. (c) AUTOVECTORES.

Figura 4.6: Conjunto de datos de imágenes satelitales representadas en un espacio bidimensional



(a) PCA. (b) t-SNE. (c) AUTOVECTORES.

Figura 4.7: Conjunto de datos de las diferentes enfermedades en hojas de tomate reducido su tamaño usando el criterio de selección de características

Los conjuntos obtenidos mediante t-SNE y autovectores fueron sometidos a diferentes técnicas de eliminación de datos atípicos y selección de prototipos para su depuración. En la Figura 4.8, se muestra cómo los algoritmos de detección de datos atípicos eliminan instancias en base a sus reglas específicas. Cabe mencionar que estas técnicas solo pueden aplicarse a imágenes cuando se han representado en menos dimensiones que una imagen RGB, lo cual implica que parte de sus características principales se han reducido o perdido. Se observa que el algoritmo *Isolation Forest* intenta mantener la distribución de los datos, conservando las características de cada etiqueta. Este algoritmo, aplicado al conjunto de datos para la detección de enfermedades en plantas de tomate,

logra una reducción del 20 %.

Por su parte, el algoritmo *One-Class SVM* es más estricto en la eliminación lineal de datos que no cumplen con sus reglas, lo que se refleja en su alta capacidad de eliminación de instancias, conservando solo el 15 % del conjunto total.

Posteriormente, al conjunto de datos de dos dimensiones obtenido mediante t-SNE, se aplicaron los algoritmos c-NN y All-kNN para reducir los datos que no presentan información relevante para el clasificador. Se observa que el algoritmo c-NN reduce un gran número de instancias, intentando mantener los datos con la misma etiqueta lo más juntos posible, al punto de reducir alrededor del 75 % del conjunto de datos. Por esta razón, también se considera inadecuado para las etapas futuras de la metodología propuesta en esta tesis doctoral.

El algoritmo All-kNN demuestra ser más flexible en sus reglas, debido a que analiza al vecino más cercano de cada instancia y, si esta es similar, intenta mantener el dato. Este proceso se realiza iterativamente hasta eliminar los datos del borde de cada etiqueta. Esta técnica permite suavizar los bordes de decisión de cada etiqueta, preservando las características de los datos en la mayor medida posible. Como resultado, el algoritmo All-kNN reduce en promedio el 48 % de los datos.

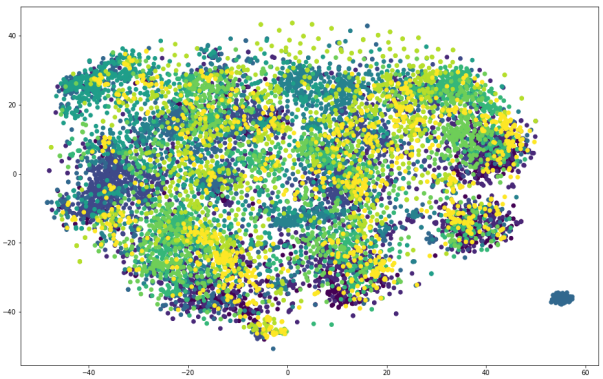
Relacionado con la reducción de dimensionalidad mediante autovectores, el algoritmo *Isolation Forest* mantiene el criterio de suavizar los bordes de decisión y busca conservar la mayor cantidad de datos posible. Por tal motivo, mantiene promedios similares de eliminación de instancias, con una reducción del 20 % del conjunto de datos. Sin embargo, *One-Class SVM* mantiene su estricta política de reducción de instancias y, en este caso, elimina el 30 % de los datos. En esta ocasión, debido a que los datos se mantienen unificados según su etiqueta, esto permite a los algoritmos reconocer de mejor forma los datos atípicos que probablemente afectarían al clasificador.

Por su parte, c-NN sigue siendo un algoritmo con reglas estrictas, reduciendo casi el 87 % de los datos. Esta importante reducción lo convierte en un algoritmo que no es viable para futuras etapas, debido a que como se observó en pruebas anteriores, el conjunto de datos resultante no es suficiente para entrenar arquitecturas CNN. Finalmente, All-kNN vuelve a presentar un buen rendimiento siendo un algoritmo muy recomendable, gráficamente se puede observar cómo busca mantener cada etiqueta lo más alejada posible del resto. Esto se aprecia en la Figura 4.9 (e).

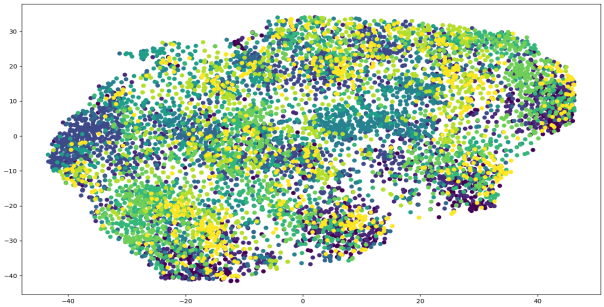
4.4.3. Conclusiones de la Ingeniería de datos

En resumen, todos los conjuntos de datos fueron expuestos a estos criterios de reducción de instancias para observar sus resultados y determinar cuál de ellos se presenta como una opción confiable para los siguientes pasos de la arquitecrura propuesta. En el Cuadro 4.2 se muestra la eliminación de instancias después de aplicar la reducción de dimensionalidad con t-SNE y autovectores, seguida de la selección de prototipos y la eliminación de datos atípicos.

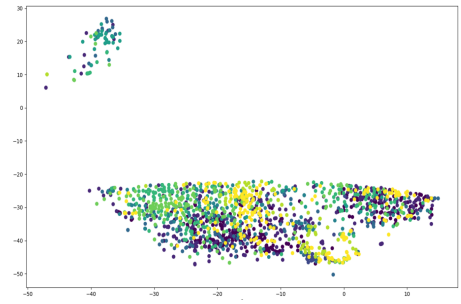
Con base en los resultados antes expuestos, se concluye que la combinación de la técnica de reducción de dimensionalidad mediante el algoritmo t-SNE y la aplicación del algoritmo *Isolation Forest* para la detección de datos atípicos muestra un rendimiento superior en comparación con el uso del algoritmo *One-Class SVM*. Como resultado, se considera el *Isolation Forest* es considerado para futuros análisis.



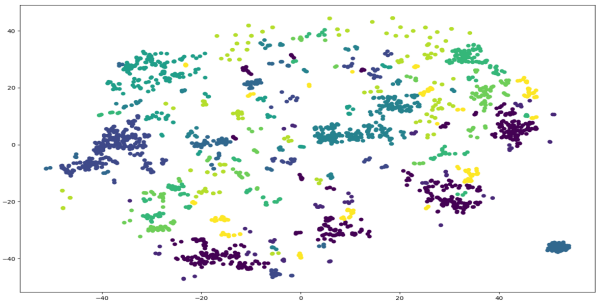
(a) t-SNE.



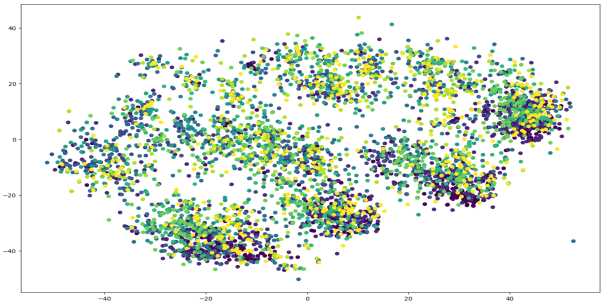
(b) t-SNE+ISO.



(c) t-SNE+One-Class SVM



(d) t-SNE+c-NN.



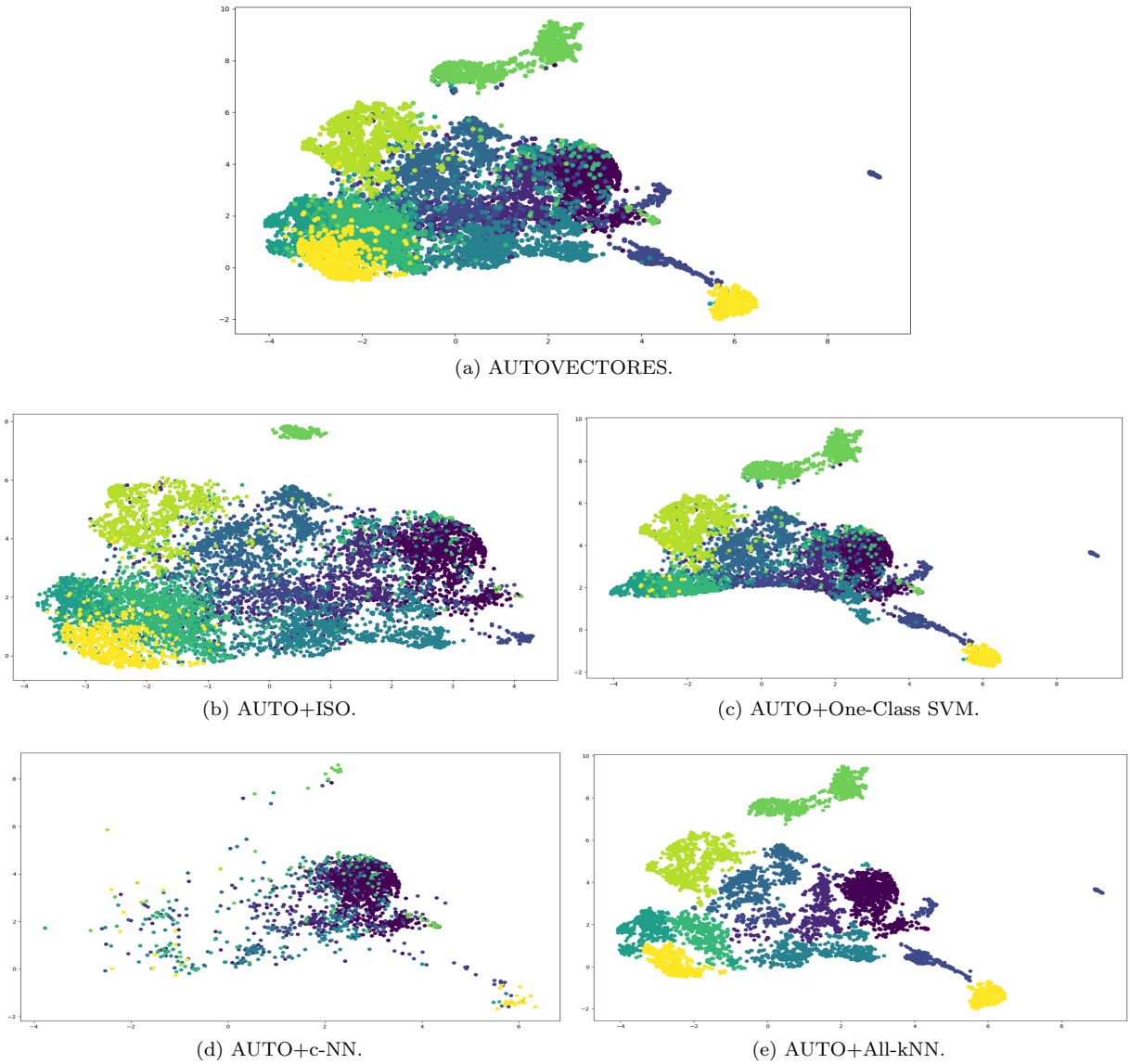
(e) t-SNE+All-kNN.

Figura 4.8: Análisis de ténicas de detección de datos atípicos y selección de prototipos al aplicar la reducción de dimensionalidad con T-SNE.

Conjunto de datos	t-SNE				Autovectores			
	ISO	One-Class SVM	c-NN	All-kNN	ISO	One-Class SVM	c-NN	All-kNN
Detección de enfermedades	19 %	85 %	75 %	48 %	23 %	29 %	85 %	42 %
Clasificación de residuos	18 %	70 %	40 %	35 %	21 %	37 %	40 %	30 %
Detección de aves	22 %	72 %	40 %	28 %	22 %	30 %	65 %	45 %
Paneles solares	19 %	65 %	55 %	25 %	23 %	28 %	40 %	31 %
Imágenes satelitales	24 %	62 %	54 %	59 %	19 %	27 %	54 %	48 %

Cuadro 4.2: Reducción de instancias mediante selección de características con conjuntos de datos reducidas a dos dimensiones

En cuanto a la selección de prototipos, se observó que el algoritmo c-NN es bastante estricto, lo que conlleva a una reducción significativa del número de instancias en el conjunto de datos y potencialmente a la pérdida de información valiosa para el clasificador. Por su parte, el algoritmo All-kNN demostró ser capaz de suavizar los bordes de decisión de cada etiqueta mientras mantiene



(a) AUTOVECTORES.

(b) AUTO+ISO.

(c) AUTO+One-Class SVM.

(d) AUTO+c-NN.

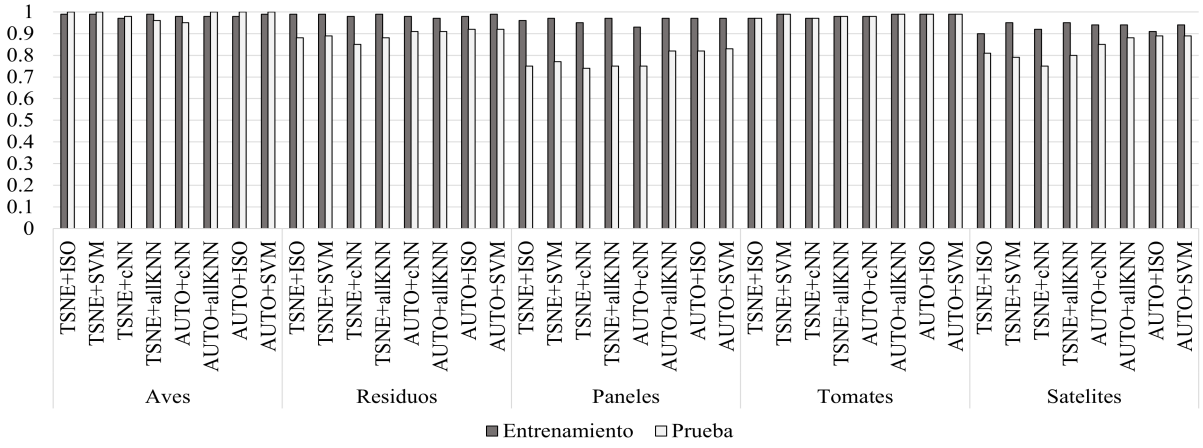
(e) AUTO+All-kNN.

Figura 4.9: Análisis de técnicas de detección de datos atípicos y selección de prototipos al aplicar la reducción de dimensionalidad con autovectores.

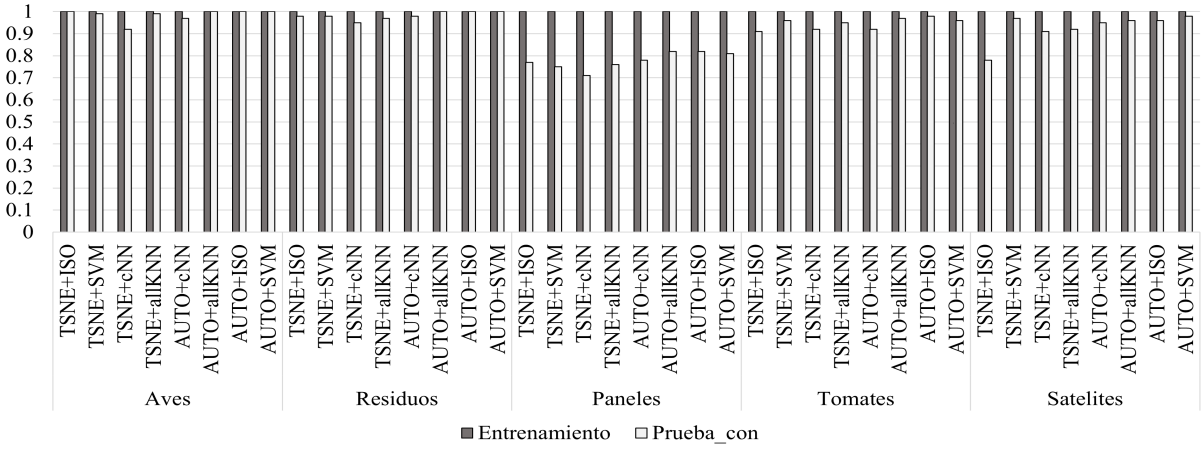
un número adecuado de instancias por etiqueta. En consecuencia, este último se presenta como la opción adecuada, independientemente de que se aplique la reducción de dimensionalidad mediante autovectores o t-SNE.

Finalmente, se concluye que se analizarán los algoritmos Isolation Forest y All-kNN para las respectivas técnicas de eliminación de datos atípicos y selección de prototipos. Esto se realizará utilizando las bases de datos reducidas a dos dimensiones mediante autovectores y t-SNE, con el objetivo de determinar su impacto y aplicabilidad en el entrenamiento de los modelos de aprendizaje automático. En la Figura 4.10 se presentan los resultados obtenidos al aplicar las distintas técnicas de detección de datos atípicos y selección de prototipos, cuando se entrenan los modelos con las arquitecturas EfficientNet e Inception-V3.

Finalmente, una vez realizado los análisis sobre la selección de características y los algoritmos de aprendizaje automático en conjunto con sus optimizaciones se definen los siguientes resultados para ser empleados en las siguientes etapas:



(a) EfficientNet



(b) Inception-V3

Figura 4.10: Comparativa de rendimiento de clasificación con subconjuntos de datos aplicando detección de datos atípicos y selección de prototipos.

- Con respecto a la reducción de instancias de los conjuntos de datos, al reducir de forma aleatoria hasta el 50 % se pudo mantener el mismo rendimiento de clasificación que al usar el 100 %.
- Mediante el análisis de características con algoritmos clásicos para seleccionar los datos con mayor información relevante, la reducción de dimensionalidad es esencial. Entre los métodos más efectivos encontramos t-SNE y autovectores, que representan adecuadamente el conjunto de datos. Además, técnicas como *Isolation Forest* (ISO) y All-kNN eliminan los datos atípicos de manera efectiva y mantienen alto rendimiento en la clasificación.
- Los modelos de aprendizaje automático EfficientNet e Inception-V3 son los que tienen el mejor rendimiento de clasificación en todos los conjuntos de datos, y sus versiones optimizadas son igualmente las más adecuadas.
- Todos estos algoritmos son implementados en las siguientes aplicaciones de Internet de las Cosas.

4.5. Destilación de conocimiento

Se establece esta sección en este punto para evitar el entrenamiento adicional de más modelos y destinar esta optimización al despliegue de modelos de aprendizaje automático en dispositivos de borde. Cabe mencionar que para lograr modelos optimizados, se requiere un profundo conocimiento de cada arquitectura, debido a que se debe mantener su integridad mientras se reducen capas o se ajustan parámetros de configuración.

4.5.1. EfficientNet

La principal idea detrás de EfficientNet es encontrar un equilibrio óptimo entre tres componentes clave de una CNN: la profundidad, el ancho y la resolución de la entrada de la imagen. Estos componentes están interrelacionados y afectan tanto el rendimiento como la eficiencia computacional de la red [56].

- Profundidad: Se refiere a cuántas capas tiene la red. Una red más profunda tiene mayor capacidad para aprender características complejas, pero también es más costosa computacionalmente.
- Ancho: Se refiere al número de canales en cada capa. Más ancho significa más parámetros y más capacidad de aprendizaje.
- Resolución de entrada: Es el tamaño de la imagen de entrada. Una imagen de mayor resolución captura más detalles, pero requiere más recursos computacionales.

EfficientNet utiliza un enfoque llamado escalado compuesto para equilibrar estos tres componentes de manera eficiente. El escalado compuesto aumenta simultáneamente la profundidad, el ancho y la resolución de la entrada de la imagen de manera uniforme [57]. Por ello, EfficientNet tiene 5 bloques que funcionan de la siguiente manera:

- Módulo de Escalado de Profundidad (Depthwise Convolution): Este módulo utiliza convoluciones separables en profundidad, que dividen una convolución en dos etapas: una convolución en cada canal de entrada por separado, seguida de una convolución 1x1 que mezcla los resultados. Esto reduce significativamente el costo computacional, ya que se aplican menos operaciones de convolución. Además, permite una mayor paralelización y, por lo tanto, una mayor eficiencia en los dispositivos con recursos limitados.
- Módulo de Escalado de Ancho (Width Scaling): Este módulo escala el ancho de la red, es decir, el número de canales en cada capa. Aumentar el ancho de la red mejora su capacidad de representación y su capacidad para aprender características más complejas. Sin embargo, esto también aumenta el costo computacional. EfficientNet utiliza un coeficiente de escala para controlar el ancho de la red de manera proporcional al aumento de profundidad y resolución.
- Módulo de Escalado de Resolución (Resolution Scaling): Este módulo aumenta la resolución de las imágenes de entrada. Las imágenes de mayor resolución capturan más detalles y características finas, lo que puede mejorar el rendimiento de la red en tareas de visión artificial. Sin embargo, el aumento de la resolución también aumenta el costo computacional. EfficientNet utiliza un coeficiente de escala para controlar el aumento de resolución junto con el aumento de profundidad y ancho.
- Conexión de Ramificación (Squeeze-and-Excitation): Este módulo, también conocido como SE (Squeeze-and-Excitation), se agrega a cada bloque de convolución para mejorar la capacidad de atención de la red a características importantes. Funciona mediante la modelación de

las relaciones entre los canales de características mediante la adaptación de los pesos de los canales. Esto permite que la red se centre más en las características relevantes y suprima el ruido.

- Conexión de Capa Residual (Residual Connection): Este módulo se basa en la idea de las conexiones residuales introducidas por las redes residuales (ResNets). Las conexiones residuales permiten que la información fluya directamente a través de la red sin alteraciones, lo que facilita el entrenamiento de redes muy profundas. En EfficientNet, se utilizan conexiones residuales entre bloques de capas para ayudar a mitigar el problema del desvanecimiento del gradiente y mejorar el flujo de información a través de la red.

Una vez entendido el funcionamiento de EfficientNet, los modelos hijos pueden reducir la complejidad del modelo al ajustar el número de canales, los coeficientes de expansión, los saltos y los tamaños de los kernels. Con el objetivo de encontrar un buen balance en los modelos reducidos, solo se modificará el número de canales, debido a que estos definen las dimensiones de las características en cada capa de la red neuronal convolucional. En cada capa convolucional, se produce un conjunto de mapas de características, donde cada mapa es una matriz tridimensional. La profundidad de esta matriz corresponde al número de canales. Por lo tanto, los canales son importantes porque determinan la cantidad de información que una capa puede capturar y procesar. Cuantos más canales tenga una capa, más complejas y ricas serán las características que puede aprender. En una EfficientNetB0 tradicional, el vector de los canales tiene la forma: [32, 16, 24, 40, 80, 112, 192, 320, 1280]. Mediante pruebas para encontrar la arquitectura adecuada, nuestra versión reducida es: [32, 8, 12, 20, 40, 66, 64, 96, 320], lo que permite reducir al menos dos veces el tamaño de la red neuronal.

4.5.2. Inception-V3

La arquitectura de Inception-V3 se basa en el concepto de "módulos de Inception", que son bloques de construcción clave que combinan diferentes operaciones de convolución en paralelo para capturar una amplia gama de características en diferentes escalas espaciales [51,53]. A continuación, se detalla una descripción de los componentes principales de esta arquitectura:

- Convoluciones Inception: Los módulos de Inception incluyen varias convoluciones en paralelo, como convoluciones 1x1, 3x3 y 5x5, así como convoluciones de agrupación máxima (max-pooling). Estas convoluciones en paralelo se diseñan para capturar características en diferentes escalas y niveles de abstracción en una sola capa, lo que permite que la red aprenda representaciones más ricas y complejas.

- Factorización de Convoluciones: Para reducir el costo computacional y el número de parámetros, Inception-V3 utiliza convoluciones factorizadas. Esto implica dividir las convoluciones 3 x 3 y 5 x 5 en dos o más convoluciones más pequeñas, como convoluciones 1 x 3 y 3 x 1. Esta técnica reduce significativamente el número de operaciones requeridas y permite un entrenamiento más eficiente de la red.

- Reducción de Dimensión: Entre los módulos de Inception, se utilizan capas de reducción de dimensión para reducir la resolución espacial de los mapas de características y el número de canales. Esto ayuda a reducir el costo computacional y permite que la red aprenda representaciones más compactas y generalizables.

- Uso de Batch Normalization y Regularización: Inception-V3 utiliza Batch Normalization después de cada capa convolucional para estabilizar y acelerar el entrenamiento de la red. Además,

utiliza técnicas de regularización, como la regularización L2, para evitar el sobreajuste y mejorar la generalización del modelo.

- Uso de Funciones de Activación ReLU: Inception V3 utiliza la función de activación ReLU (Rectified Linear Unit) después de cada operación de convolución para introducir no linealidades en la red y permitir que la red aprenda representaciones más complejas y no lineales de los datos de entrada.

Bajo este criterio, los modelos hijos de Inception-V3 han sido reducidos el número de capas de cada convolución Inception, donde inicialmente cada una de ellas puede tener 96 capas en los primeros bloques, reduciendo a 32 en el segundo, 48 en el tercero y 24 en el cuarto. Bajo esta configuración, se busca reducir este número en los diferentes bloques para observar el impacto en el rendimiento de clasificación.

4.6. Casos de Estudio

Los casos de estudio demuestran la utilidad de la arquitectura propuesta aplicados en entornos reales donde los requerimientos de los dispositivos de borde son muy estrictos, especialmente su tiempo de procesamiento. Cada uno de ellos presentan una descripción del sistema, el objetivo de la aplicación, los escenarios de trabajo y los resultados obtenidos, comparándolos con modelos de Aprendizaje por Transferencia estándar.

4.6.1. Descripción de los experimentos

La arquitectura propuesta en esta investigación será evaluada en los siguientes experimentos:

- Los modelos pre-entrenados con las arquitecturas de Inception-V3 y EfficientNet obtenidos con la Configuración 2 proveniente de la gestión de modelos, son analizados mediante las métricas de rendimiento comentadas anteriormente.
- Los modelos pre-entreandos con las arquitecturas de Inception V3 y EfficientNet obtenidos con la Configuración 2 son cuantizados a 8 *bits* para comparar su rendimiento con el punto anterior.
- Los modelos obtenidos por la destilación de conocimiento obtenidos de las arquitecturas de Inception-V3 y EfficientNet son analizados con las métricas de rendimiento comentadas anteriormente.
- Los modelos obtenidos por la destilación de conocimiento obtenidos de las arquitecturas de Inception-V3 y EfficientNet son cuantizados a 8 *bits* y se compara su rendimiento con el punto anterior para determinar la arquitectura CNN adecuada.
- Los modelos obtenidos desde la etapa de destilación de conocimiento son re-entrenados con los subconjuntos de datos obtenidos al aplicar los algoritmos de *Isolation Forest* (eliminación de datos atípicos) y All-kNN (selección de prototipos) en conjunto con las técnicas de reducción de dimensionalidad T-SNE y autovectores para conocer si el rendimiento del clasificador decae.
- Finalmente, se establece la técnica de selección de características adecuada (*Isolation Forest* o *All-kNN*) en conjunto con la reducción de dimensionalidad (t-SNE o autovectores) para re-entrenar el modelo obtenido por la destilación de conocimiento en el dispotivo de borde y analizar sus resultados.

4.6.2. Aplicación de la arquitectura propuesta para la clasificación en imágenes satelitales

Los CubeSats, una clase revolucionaria de nanosatélites, han redefinido el panorama de la exploración espacial y la innovación tecnológica. Estos satélites estandarizados con forma de cubo, que normalmente miden 10 centímetros de cada lado, permiten a organizaciones e instituciones de diferentes escalas participar en iniciativas espaciales de vanguardia. Sin embargo, los CubeSats enfrentan varios desafíos, siendo uno de los más importantes la limitación del ancho de banda y los cortos intervalos de tiempo para la transmisión de mensajes con la estación terrestre. Además, las restricciones físicas, térmicas y de energía requieren altos estándares para la selección de los componentes electrónicos adecuados.

Hoy en día, los CubeSats permiten aplicaciones de teledetección, debido a que el satélite se considera un dispositivo IoT para enviar información valiosa y descubrir nuevos patrones en un dominio específico. Por lo tanto, comúnmente disponen de pequeñas cámaras. Estas cámaras pueden ser de tipo multiespectral, proporcionando una visión completa de las diferentes bandas del espectro radioeléctrico de las áreas objetivo. Algunas cámaras empleadas en estos dispositivos son: cámaras de campo amplio, que capturan escenas amplias para un barrido general de información; cámaras con teleobjetivo, que permiten varios enfoques; y cámaras infrarrojas, que revelan información más allá del espectro visible, usualmente destinadas al análisis de gases específicos o temperatura. Al unir estas capacidades permiten un enfoque multifacético para la adquisición y el análisis de datos. Además, una unidad de cómputo debe procesar las imágenes localmente (en el satélite) o enviarlas a la estación terrestre.

Uno de los principales desafíos es contar con una base de datos que tenga la misma resolución que las cámaras en estos satélites, ya que la mayoría de los conjuntos de datos disponibles provienen de satélites de grandes prestaciones. Por lo tanto, sus imágenes son muy largas en relación con la distancia a la que operan desde la Tierra y los lentes ópticos que utilizan sus cámaras. Por tal motivo, el conjunto de datos que se empleó para este caso de estudio ha sido previamente cuadriculado para obtener imágenes del tamaño que las arquitecturas de CNN pueden procesar. En este caso, las imágenes fueron obtenidas durante un año en el mismo lugar (una provincia de Estados Unidos), donde expertos anotaron los datos con etiquetas visiblemente clasificables. Las etiquetas son: Fondo artificial (empleado para carreteras e industrias), Cultivo, Agua, Naturaleza sin contacto humano, Vegetación con árboles y Nieve. La descripción de cada etiqueta se muestra en el Cuadro 4.3.

4.6.2.1. Objetivo de la aplicación

El principal objetivo en este conjunto de datos se enfoca en el reconocimiento de hielo para realizar un estudio sobre el impacto del cambio climático. Por lo tanto, el satélite tiene que ser capaz de tomar las imágenes con sus cámaras de campo amplio y teleobjetivo, y determinar si contiene nieve/hielo. Una vez que se hayan capturado las imágenes, deben ser enviadas a la estación terrestre para un futuro análisis. Este proceso debe llevarse a cabo de forma rápida debido a que el tiempo de paso del satélite sobre la estación terrestre es muy corto, y no es posible enviar todas las imágenes que puede tomar durante su tiempo de vuelo. La Figura 4.11 presenta ejemplos de cada etiqueta.

Número	Etiqueta	Descripción
0	Fondo Artificial	El fondo artificial comprende las mediciones de las ciudades que se encuentran rodeadas de carreteras y fábricas, generalmente son imágenes con varios puntos luminosos.
1	Cultivos	Son espacios amplios de cultivos generalmente organizados por parcelas con varios colores que representan diferentes cultivos.
2	Naturaleza sin contacto humano	Es un espacio de vegetación en tono montañoso sin demasiada vegatación a la vista.
3	Vegetación y cultivos	Son espacios que comparten vegetación y cultivos a la par.
4	Nieve	Ubicaciones geográficas que anteriormente pudieron tener otra etiqueta han sido modificadas al tener nieve en la estación de invierno.
5	Vegetación con árboles	Vegetación de tono verde representada mayormente de árboles
6	Agua	Generalemente lagos que usualmente en la noche son grandes espacios de color negro.

Cuadro 4.3: Descripción de etiquetas de la aplicación de reconocimiento de hielo mediante satélites

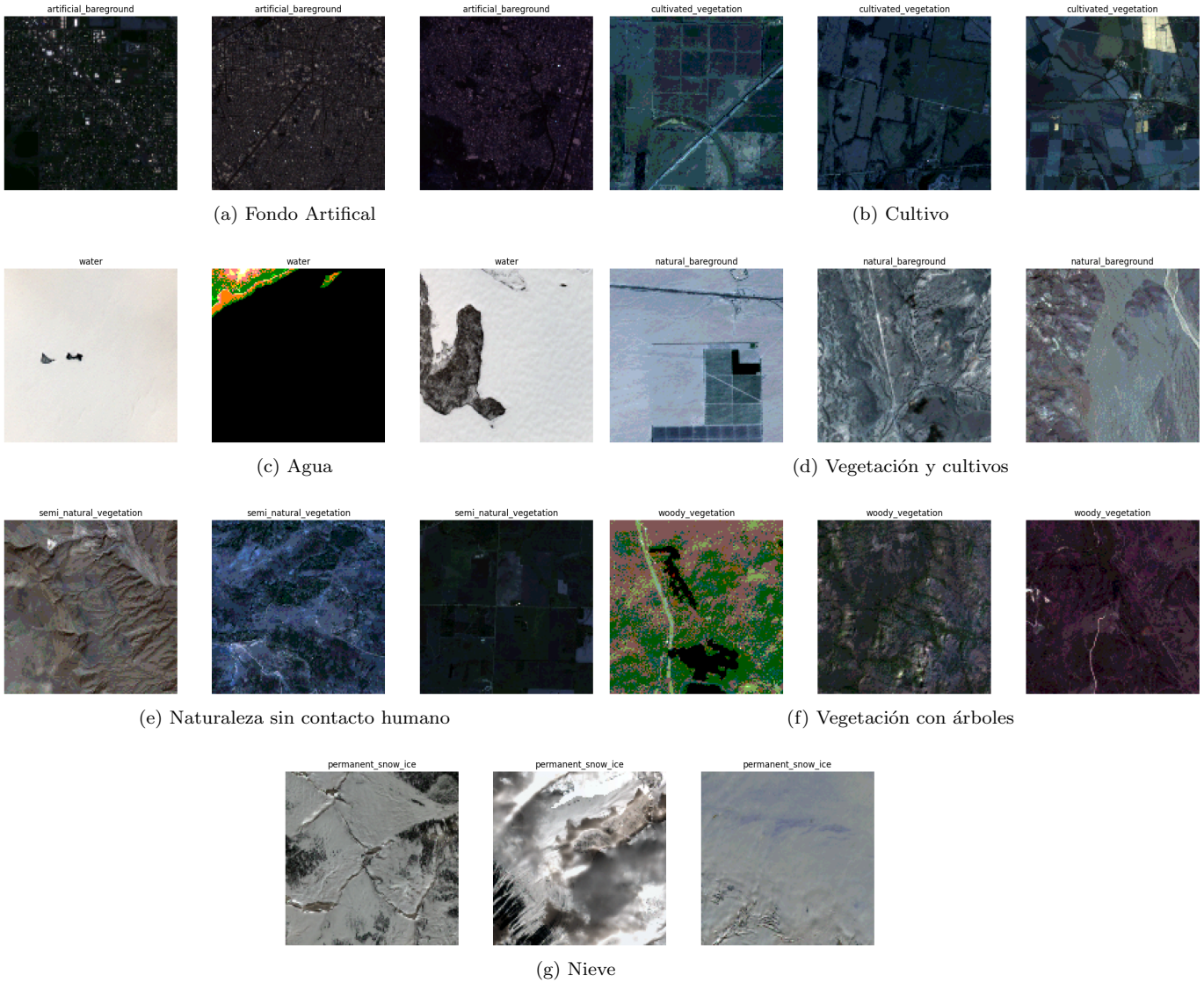
4.6.2.2. Descripción del sistema

Con el conjunto de datos descrito y la configuración de los modelos de aprendizaje automático establecida, es necesario optimizar su tiempo de respuesta debido a las limitadas ventanas de comunicación entre el satélite y la estación terrestre. En este contexto, se proponen dos métodos de optimización. El primero se basa en seleccionar el conjunto de datos óptimo para afinar el modelo al conjunto de datos propuesto, debido a que este operará en diferentes estaciones del año y será necesario actualizar el modelo periódicamente. El segundo método implica un análisis detallado de los filtros en cada arquitectura CNN que afectan el reconocimiento de patrones para inferir las etiquetas de las imágenes capturadas por la cámara. Posteriormente, se entrena un modelo de menor tamaño utilizando un enfoque de aprendizaje mediante transferencia de conocimiento, donde el modelo completo con todas sus capas actúa como maestro. Para verificar que el proceso de optimización es adecuado, los resultados iniciales se centran en el rendimiento de los modelos entrenados con el conjunto de datos completo. Luego, tanto el conjunto de datos como el modelo se optimizan para evaluar su rendimiento y realizar las debidas comparaciones con los modelos y datos de mayor volumen. Con el fin de reducir el tamaño del modelo de aprendizaje automático se aplica una cuantización a 8 *bits*, lo que permite realizar inferencias cuando el satélite esté en órbita. La Figura 4.12 muestra la configuración de las cámaras en un espacio de 10x10x10 cm.

4.6.2.3. Escenarios

La órbita planificada de un nano satélite suele ser de una órbita polar sincrónica con el sol a 500 kilómetros de altitud de la tierra. El sensor de la cámara está basado en un Alvium 1800 C-2040 con una resolución de imagen de 4512 x 4512 píxeles, un tamaño de píxel de 2.74 um y la distancia focal del objetivo es la máxima que se podría considerar para la misión. De inicio, se debe conocer la distancia de muestra del suelo (GSD por sus siglas en inglés *Ground Sample Distance*) que se refiere a la relación de lo que representa cada píxel con respecto a la tierra. Su forma básica para calcular esta distancia se presenta en la ecuación 4.6.1.

$$GSD = \frac{\text{Altura orbital} * \text{Tamaño de píxel}}{\text{Distancia focal}} = \frac{500km * 2{,}74um}{92mm} = 14{,}84 \tag{4.6.1}$$



(a) Fondo Artifical

(b) Cultivo

(c) Agua

(d) Vegetación y cultivos

(e) Naturaleza sin contacto humano

(f) Vegetación con árboles

(g) Nieve

Figura 4.11: Conjunto de datos para CubeSats para el reconicimiento de hielo permanente

Donde la altura orbital es de 500 km, el tamaño de píxel de 2.74 um y la distancia al lente es de 92 mm. Por lo tanto, el SGD es de 14.84. Ahora, suponiendo que existe una superposición del 50 % entre imágenes, es decir, capturando la misma superficie terrestre, esto proporciona un tiempo mínimo de vuelo (T_i) de 4,42 segundos entre imágenes consecutivas como se muestra en la ecuación 4.6.2.

$$T_i = \frac{GSD * H_i * \cap_i}{v} = \frac{14{,}84m/px * 4512px * 0{,}5}{7,585{,}16m/s} = 4{,}42s \tag{4.6.2}$$

Donde H_i es el número de píxeles, $\cap_i$ es la superposición de la imagen y v es la velocidad de vuelo que se considera una constante de 7585.16m/s. Con este valor, se plantea el estudio del cambio climático en Groenlandia que tiene una extensión de 2670000 km^2 y con el resto de variables previamente descritas, en la ecuación 4.6.3 se determina el número de imágenes (N_i) que se deben tomar para cubrir toda la superficie, que son 79.7 imágenes.

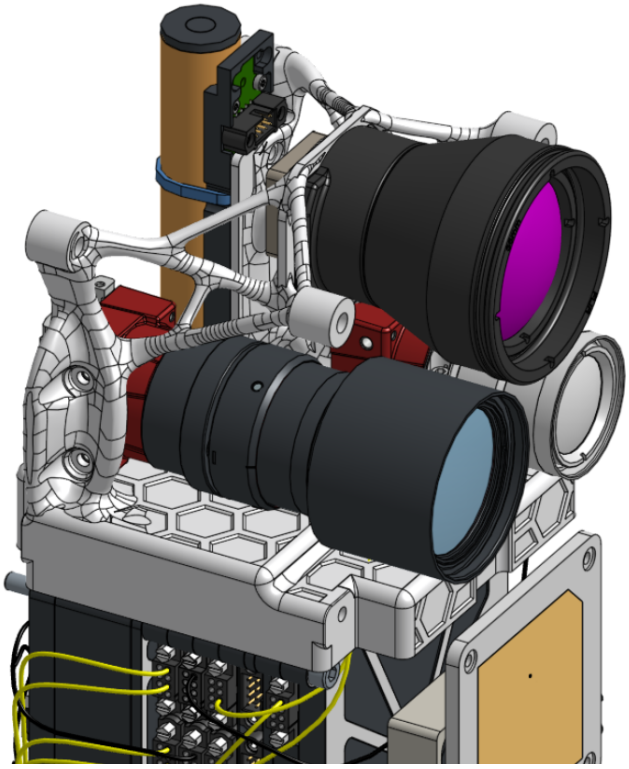


Figura 4.12: Imagen referencial de la configuración de cámaras en un nano satélite

$$N_i = \frac{H_G}{GSD * H_i * \cap_i} = \frac{2670000}{14{,}84m/px * 4512px * 0{,}5} = 79{,}7 \tag{4.6.3}$$

Además, es importante considerar que el tamaño N_i está relacionado con la resolución de las imágenes que puede capturar la cámara. Posteriormente, es necesario cuadricular cada imagen para que se ajuste a las medidas que el modelo de aprendizaje automático requiere. En este caso, si la cámara puede capturar imágenes de 4512x4512 píxeles y el modelo soporta imágenes de 224x224 píxeles, se obtienen 400 imágenes a partir de cada imagen capturada por el satélite, a cada una de las cuales se debe inferir su etiqueta correspondiente. Basándonos en la información presentada anteriormente, se plantean dos escenarios:

- **Escenario A - Procesamiento en tiempo real:** Se busca reconocer si cada imagen tiene hielo en la superficie de la tierra y si de ser positivo, almacenarla en un búfer de datos hasta que el satélite termine de capturar imágenes y enviar todas en conjunto cuando el satélite siga con su trayectoria alrededor de la tierra. Por lo tanto, el modelo de aprendizaje automático debe procesar en 4.42 segundos 400 imágenes hasta que reciba una nueva imagen de mayor resolución tomada por la cámara, por lo que el tiempo máximo de procesamiento por imagen debe ser de **11 milisegundos**.
- **Escenario B - Procesamiento de imagen en vuelo:** En esta configuración, el satélite tomará todas las imágenes de Groenlandia sin procesarlas y las almacenará en un búfer. El tiempo de vuelo de un nano satélite al dar una vuelta a la tierra en una trayectoria de órbita polar sincrónica con el sol es de 90 minutos. Primero, se debe restar el tiempo que el satélite volará por Groenlandia tomando fotos que será $T_i * N_i$ que es igual a $\approx 352 segundos$. Luego, el

satélite tendrá los 90 minutos (5739 segundos) del tiempo de vuelo, menos el tiempo de toma de imágenes (352 segundos) para procesar las imágenes que será de 5387 segundos para inferir la etiqueta de 80 imágenes de alta resolución. Luego, cada una de ellas se vuelve a convertir a 400 imágenes de menor resolución para que el modelo de aprendizaje automático pueda realizar la inferencia de las mismas. Por lo que $5387 segundos / 80 * 400 =$ **160 milisegundos**.

4.6.2.4. Análisis de Resultados

Con el fin de evaluar el rendimiento general de clasificación de los modelos de aprendizaje automático provenientes de las arquitecturas CNN de EfficientNet e Inception V3, el Cuadro 4.4 muestra las diferentes métricas de clasificación, incluyendo la exactitud, el *recall* y el *F1-Score* para EfficientNet. Además, se presentan los resultados cuando el modelo fue exportado para hacer inferencias sin ser optimizado (*TensorFlow Lite*) y cuando fue cuantizado a 8 *bits*. Se observa que todas las etiquetas presentan un alto rendimiento de clasificación. La etiqueta 4, correspondiente a hielo, muestra un 68 % de éxito en la asignación de etiqueta, lo cual es muy aceptable dadas las condiciones del entorno de trabajo. A pesar de la optimización del modelo a 8 *bits*, algunas etiquetas mantienen un alto rendimiento de clasificación. No obstante, se observa una disminución general en el rendimiento del modelo de 8 puntos.

Con respecto a su tamaño, se evidencia que el modelo al ser cuantizado reduce en 5 veces su tamaño, lo que permitirá rápidas inferencias cuando se tengan los lotes de 400 imágenes.

Etiqueta	**EfficientNet**					
	TensorFlow Lite			**Cuantización a 8 *bits***		
	Exactitud	Recall	F1-Score	Exactitud	Recall	F1-Score
0	0.99	1.00	0.99	1.00	0.75	0.86
1	0.72	0.90	0.80	0.80	1.00	0.89
2	0.67	0.97	0.79	0.72	0.84	0.78
3	1.00	0.27	0.42	1.00	0.25	0.40
4	0.68	0.38	0.46	0.60	0.38	0.43
5	1.00	0.96	0.98	0.50	1.00	0.67
6	0.77	0.94	0.84	0.71	1.00	0.83
Rendimiento	0.77			0.69		
Tamaño	25.4 MB			4.7 MB		

Cuadro 4.4: Rendimiento general de clasificación de la arquitectura CNN EfficientNet

En relación al modelo de aprendizaje automático con la arquitectura CNN de Inception-V3, su rendimiento de clasificación es menor al de EfficientNet en términos generales, pero la etiqueta relacionada al hielo tiene resultados muy similares. Además, cuando el modelo se cuantiza, su rendimiento general disminuye solo 4 puntos pero con una reducción en su tamaño de 4.44 veces. Cabe mencionar que el modelo cuantizado de Inception-V3 tiene un tamaño similar al modelo sin cuantizar de EfficientNet. Esta información se presenta en el Cuadro 4.5.

Con los modelos de aprendizaje automático entrenados mediante Aprendizaje por Transferencia, se diseñaron arquitecturas livianas de EfficientNet e Inception-V3, modificando sus filtros y parámetros de configuración para utilizar la técnica de conocimiento por destilación. El modelo generado por el Aprendizaje por Transferencia se encarga de enseñar al modelo ligero, el cual aprenderá y actualizará sus valores en cada capa para afinarse al conjunto de datos. De este modo, se puede evaluar la efectividad de la técnica de destilación de conocimiento.

Etiqueta	**Inception-V3**					
	TensorFlow Lite			**Cuantización a 8 *bits***		
	Exactitud	Recall	F1-Score	Exactitud	Recall	F1-Score
0	0.87	1.00	0.93	0.96	0.98	0.97
1	0.82	0.85	0.83	0.79	0.89	0.84
2	0.61	0.84	0.70	0.59	0.86	0.70
3	1.00	0.20	0.34	1.00	0.12	0.21
4	0.62	0.32	0.36	0.58	0.21	0.27
5	0.84	0.96	0.90	0.84	0.96	0.90
6	0.66	0.83	0.73	0.61	0.85	0.71
Rendimiento	0.73			0.69		
Tamaño	98.4 MB			22.12 MB		

Cuadro 4.5: Rendimiento general de clasificación de la arquitectura CNN Inception-V3

Etiqueta	**EfficienNet**					
	TensorFlow Lite			**Cuantización a 8 *bits***		
	Exactitud	Recall	F1-Score	Exactitud	Recall	F1-Score
0	0.86	0.77	0.81	0.74	0.77	0.75
1	0.39	0.51	0.44	0.34	0.42	0.38
2	0.53	0.91	0.67	0.57	0.88	0.69
3	0.65	0.56	0.60	0.63	0.57	0.60
4	0.52	0.22	0.26	0.50	0.21	0.24
5	0.93	0.29	0.44	1.00	0.16	0.28
6	0.73	0.53	0.62	0.66	0.58	0.61
Rendimiento	0.59			0.57		
Tamaño	2.7 MB			1.1 MB		

Cuadro 4.6: Rendimiento de clasificación del modelo obtenido por destilación de conocimiento con la arquitecura EfficientNet.

En el Cuadro 4.6 se puede observar que el modelo liviano tiene un rendimiento del 59 %, lejos del 77 % obtenido con el modelo proveniente de conocimiento por transferencia. También se puede apreciar que en la etiqueta 4, relacionada al hielo, el 52 % de las imágenes de prueba han sido clasificadas correctamente. Por su parte, en la versión cuantizada de 8 *bits*, su rendimiento de clasificación general es de 57 % ya que en todas las etiquetas se tiene una precisión inferior del 70 %. No obstante, su tamaño es significativamente menor, su versión sin cuantizar pesa 3.5MB y su versión cuantizada 1.1MB que representa una reducción de 7 veces su tamaño. Este modelo reducido fue entrenado desde cero para conocer su rendimiento de clasificación. Su resultado no fue satisfactorio ya que su rendimiento general es del 41 % y tiene un tamaño de 8.9 MB aún cuando se ha empleado la misma arquitectura CNN.

El modelo de aprendizaje automático basado en la arquitectura CNN de Inception pasó por el mismo proceso que el modelo de EfficientNet para encontrar un modelo hijo liviano. En este caso, se ajustó el número de capas en los diferentes bloques de Inception. Como resultado, tal como se presenta en el Cuadro 4.7, el modelo liviano tiene un rendimiento del 71 %, muy similar al modelo grande proveniente de la transferencia de conocimiento, pero con un tamaño de sólo 3.5 MB. Esto representa una reducción de 28 veces su tamaño inicial. Además, en su versión cuantizada, el modelo tiene un tamaño de sólo 941 KB, manteniendo un rendimiento del 69 %. La precisión en la detección de la etiqueta correspondiente a hielo es cercana al 60 % y la mayoría de las etiquetas restantes presentan rendimientos aceptables. Cuando el modelo liviano fue entrenado desde cero, su tamaño fue de 7.3 MB y no logró superar el 50 % en el rendimiento de clasificación.

Etiqueta	Inception-V3					
	TensorFlow Lite			Cuantización a 8 *bits*		
	Exactidud	Recall	F1-Score	Exactitud	Recall	F1-Score
0	0.91	0.96	0.93	0.91	0.96	0.93
1	0.68	0.77	0.72	0.64	0.77	0.70
2	0.61	0.91	0.73	0.61	0.92	0.73
3	0.80	0.59	0.68	0.79	0.58	0.67
4	0.58	0.19	0.25	0.58	0.18	0.24
5	0.95	0.76	0.84	0.97	0.76	0.85
6	0.77	0.72	0.74	0.78	0.71	0.74
Rendimiento	0.71			0.69		
Tamaño	3.5 MB			941KB		

Cuadro 4.7: Rendimiento de clasificación del modelo obtenido por destilación de conocimiento con la arquitecura Inception-V3.

Ya que Inception-V3 es seleccionada como la arquitectura CNN adecuada para el modelo de ML, este es re-entrenado con los conjuntos datos optimizados obtenidos al aplicar la reducción de dimensionalidad con t-SNE y autovectores y aplicando Isolation Forest y All-kNN para la selección de las imágenes más relevantes. Como se puede apreciar en el Cuadro 4.8, al aplicar All-kNN con autovectores, el conjunto de datos solo pesa alrededor de 15MB y mantiene el mismo rendimiento de los modelos que han sido entrenados con todo el conjunto. Por su parte, Isolation Forest mantiene el mismo rendimiento pero requiere un mayor número de imágenes en el conjunto de entrenamiento. Por esta razón, para la optimización del conjunto de datos se debe aplicar All-kNN en conjunto de autovectores.

Conjunto de datos	Tamaño MB	Rendimiento
ISO+t-NSE	22 MB	0.69
ISO+AUTO	22.8 MB	0.69
All-kNN+t-SNE	12 MB	0.65
All-kNN+AUTO	14.8 MB	0.69

Cuadro 4.8: Análisis de resultados de entrenamiento al emplear las bases de datos optimizados del conjunto de datos de imágenes satelitales

Finalmente, se concluye que para la aplicación propuesta y el futuro análisis en dispositivos de borde, el modelo liviano obtenido con la arquitectura de Inception-V3 es el más adecuado. Esto se debe a que su rendimiento de clasificación es similar al de los modelos obtenidos por transferencia de conocimiento, pero su tamaño ha sido reducido 28 veces. Por otro lado, aunque la arquitectura EfficientNet mostró el rendimiento de clasificación más alto, sus optimizaciones no lograron resultados prometedores. Por la parte de optimización del conjunto de datos All-kNN con autovectores permite eliminar las imágenes que no aportan información relevante al modelo de ML.

4.6.3. Aplicación de la arquitectura propuesta para la clasificación de imágenes en invernaderos

La producción agrícola es sin duda uno de los principales temas de interés en la agenda global. De acuerdo a la Unión Europea, la producción agrícola mundial debe duplicarse para el año 2050, esto debido a la proyección de crecimiento poblacional y la evolución en los hábitos alimentarios. Sin embargo, factores externos como el cambio climático, que afectan a la calidad del agua y del

suelo, están replanteando las técnicas y exigencias del mercado mundial y ponen en riesgo la capacidad de cubrir la demanda alimentaria. Por lo tanto, en países europeos actualmente existe una política agrícola que pretende ayudar a los agricultores a afrontar los desafíos y evolucionar a las nuevas exigencias de los consumidores. La calidad de los alimentos y su trazabilidad, son temas que han sido puestos sobre la mesa en la agenda internacional, buscando implementar técnicas sostenibles y respetuosas con el medio ambiente [94].

Por lo expuesto, la implementación de invernaderos es considerada una opción viable para mantener las condiciones ambientales, permitiendo mejorar los procesos de cultivos y cosecha. Sin embargo, a pesar de ser un ambiente controlado, aún existen variables que pueden afectar el entorno de los cultivos, por ello es necesario asegurar la estabilidad de la temperatura, humedad, concentración de gases, entre otros factores [95–97].

En respuesta a las demandas de la producción agrícola y como una alternativa para su constante mejora, se han introducido tecnologías avanzadas en el sector agrícola tales como IoT, satélites, drones, vehículos autónomos no tripuraldos (UAVs), redes de sensores inalámbricos y sistemas de gestión de agricultura mediante el uso de *Big Data.* Algunas de estas soluciones innovadoras reducen y optimizan el uso de fertilizantes y pesticidas, mejorando así la eficiencia de la producción agrícola [98]. Otras soluciones buscan mantener entornos más eficientes y manejables a través del monitoreo de diversas variables dentro de invernaderos [97, 99].

Este caso de estudio es una aplicación IoT centrada en desarrollar un modelo de aprendizaje automático que funciona con un robot autónomo, el cual recopila datos de cultivos en un invernadero de pequeña y mediana escala para proporcionar a los agricultores información relevante sobre cambios en las condiciones ambientales y apoyar la toma de decisiones. El robot emplea una cámara RGB para capturar imágenes de los cultivos y detectar cuáles presentan condiciones normales de crecimiento o muestran inconvenientes, reflejados en la forma de sus hojas.

El robot autónomo mencionado consta de dos subsistemas. El primer subsistema recopila datos de la plantación de tomate y asigna una etiqueta sobre la condición del cultivo. El segundo subsistema se encarga del diseño y control del robot para garantizar su correcto funcionamiento mientras sigue las rutas de cultivo planificadas. Para lograr esto, un controlador proporcional-integral-derivado (PID) regula los motores de corriente continua (CC) que controlan el movimiento del robot. El controlador PID está diseñado para que el robot pueda desplazarse sobre terrenos irregulares y en entornos hostiles, avanzando a baja velocidad por las líneas de cultivo sin perder el agarre proporcionado por el chasis y las orugas. Una imagen referencial de un robot adquiriendo datos en un invernadero fue tomada de [97] y puede observarse en la Figura 4.13.

4.6.3.1. Objetivo de la aplicación

La detección de posibles enfermedades en plantas de tomate es el objetivo de este caso de estudio. Para ello, se emplea un conjunto de datos que fue etiquetado por expertos. El sistema captura cuatro fotos consecutivas en vertical de cada planta y asigna la etiqueta correspondiente. Si esta etiqueta indica la presencia de una enfermedad, se enviará una alarma a un servidor central con la ubicación del robot para que se puedan tomar acciones correctivas en el futuro.

4.6.3.2. Descripción del sistema

Como se ha mencionado, el robot autónomo consta de dos subsistemas. El subsistema encargado del movimiento está fuera del alcance de esta aplicación, pero se conoce que el robot sigue

Figura 4.13: Vehículo autónomo tomando datos en un invernadero

rutas planificadas y se desplaza a una velocidad de 8 m/s. El segundo subsistema se encarga del reconocimiento de etiquetas entre hojas sanas, hojas con manchas, hojas amarillas (indicio de falta de humedad) y hojas secas. La descripción de cada etiqueta se muestra en el Cuadro 4.9

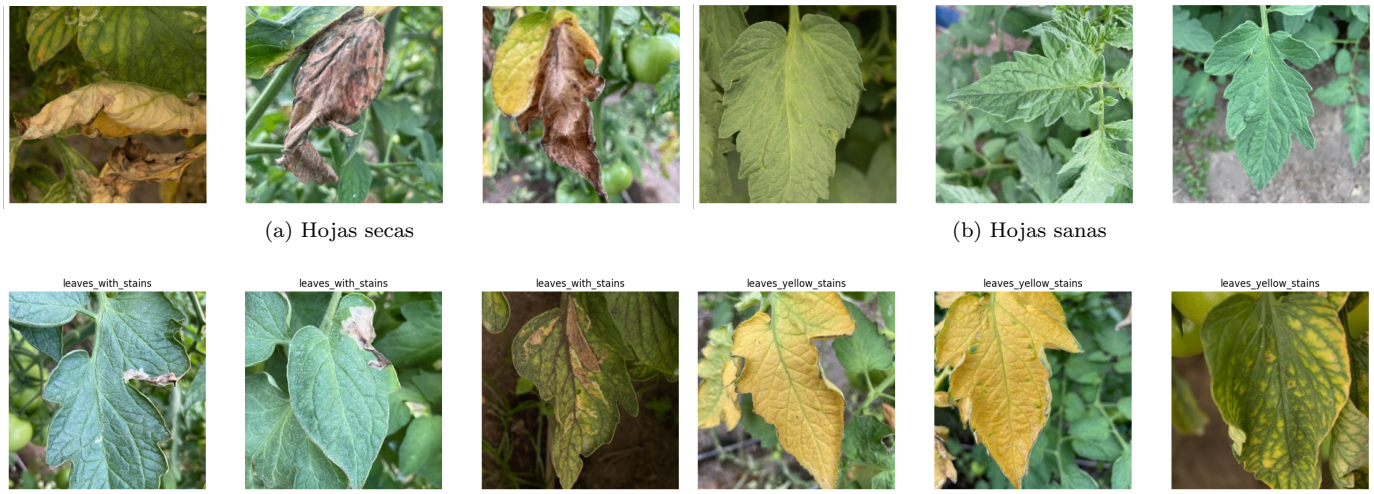
Número	Etiqueta	Descripción
0	Hojas secas	Sugiere la presencia de enfermedad en la planta acompañada de poca humedad.
1	Hojas sanas	EL cultivo sin precesencia de enfermedades
2	Hojas con manchas	Suele indicar la posible presencia de insectos y enfermedades que rápidamente pueden contagiar al resto.
3	Hojas amarillas	Muestra el inicio de falta de humedad y/o alta exposición a rayos UV.

Cuadro 4.9: Descripción de etiquetas de la aplicación de detección de posibles enfermedades en invernaderos

La cámara seleccionada es una ArduCam de 16 megapíxeles que captura imágenes de 4656x3496 píxeles. Esta cámara estará conectada al dispositivo de borde, el cual será responsable de ejecutar el algoritmo de aprendizaje automático. La Figura 4.14 muestra algunos ejemplos de hojas de tomate en relación con sus respectivas etiquetas.

4.6.3.3. Escenario

Dado que el sistema toma 4 fotos de cada planta con una resolución de 4656 x 3496 píxeles, es necesario cuadricular cada imagen para que cada segmento tenga el tamaño adecuado para el modelo de inferencia, que es de 224 x 224 píxeles. Por cada imagen tomada por la cámara, se generan aproximadamente 300 imágenes de menor tamaño para ejecutar las inferencias del modelo. Puesto que se toman 4 fotos para cubrir toda la altura de la planta, el dispositivo de borde debe procesar 1200 imágenes en 24 segundos, ya que el robot tiene planificado tomar fotografías cada 3 metros y su velocidad es de 8 m/s. Esto implica que el tiempo de procesamiento por cada inferencia debe ser de 20 milisegundos.



(a) Hojas secas (b) Hojas sanas

(c) Hojas con manchas (d) Hojas amarillas

Figura 4.14: Hojas de plantas de tomate con las diferentes etiquetas

4.6.3.4. Análisis de Resultados

El modelo de aprendizaje automático con la arquitectura EfficientNet tiene un alto rendimiento de clasificación que es del 89 % con un tamaño de 15.6 MB, así como su versión cuantizada pero un tamaño de 4.7 MB. Por su parte, el modelo con la arquitectura Inception-V3 alcanza un rendimiento del 84 % en la versión de *TensorFlow Lite* y su cuantización, con pesos de 85.8 MB y 21.9 MB respectivamente. Estos valores se pueden apreciar en el Cuadro 4.10.

Etiqueta	**Modelos**					
	EfficientNet			**Inception-V3**		
	Exactitud	Recall	F1-Score	Exactitud	Recall	F1-Score
0	0.90	0.97	0.94	0.79	0.99	0.87
1	1.0	0.74	0.85	0.94	0.51	0.66
2	0.72	1.0	0.84	0.64	1.0	0.78
3	1.00	0.89	0.94	0.89	0.60	0.78
Rendimiento	0.89			0.84		
Tamaño	15.6 MB / 4.7MB			85.8 MB / 21.9 MB		

Cuadro 4.10: Rendimiento general de clasificación de la arquitectura CNN EfficientNet e Inception V3

Cuando el modelo liviano se desarrolló utilizando el método de aprendizaje por destilación, se observó que el rendimiento de clasificación disminuyó significativamente hasta un 71 % en ambas versiones. Este modelo, entrenado desde cero, presenta un rendimiento del 41 %. A pesar de que la técnica de destilación de conocimiento ha demostrado ser adecuada para optimizar los modelos de aprendizaje automático, el rendimiento de EfficientNet decrece en más de un 10 %. En el Cuadro 4.11 se presentan las diferentes métricas de clasificación.

Por su parte, correspondiente con la arquitectura Inception-V3, la versión del modelo liviano mantiene el rendimiento de clasificación. Este modelo tiene un tamaño de 3.5 MB en la versión sin optimizar y 941 KB en la versión optimizada. Se mantienen los valores de las métricas de exactitud, *recall* y el *F1-Score*. Por tal motivo, vuelve a ser el modelo adecuado para ser llevado al análisis de rendimiento en dispositivos de borde. Finalmente, la versión liviana del modelo entrenada desde

Etiqueta	**EfficientNet**					
	TensorFlow Lite			**Cuantización a 8 *bits***		
	Exactitud	Recall	F1-Score	Exactitud	Recall	F1-Score
0	0.80	0.99	0.88	0.78	1.00	0.88
1	0.84	0.98	0.91	0.83	1.00	0.91
2	0.50	0.54	0.55	0.40	0.43	0.42
3	0.43	0.29	0.35	0.53	0.18	0.37
Rendimiento	0.71			0.70		
Tamaño	2.7 MB			1.1 MB		

Cuadro 4.11: Rendimiento de clasificación del modelo obtenido por destilación de conocimiento con la arquitecura EfficientNet - invernaderos

cero tiene un rendimiento inferior que no llega al 70 %. Por ende, no es tomado en cuenta para los siguientes análisis y no se muestran el resto de métricas. El Cuadro 4.12 muestra todas las métricas de clasificación de la versión obtenida por la destilación de conocimiento.

Etiqueta	**Inception-V3**					
	TensorFlow Lite			**Cuantización a 8 bits**		
	Exactitud	Recall	F1-Score	Exactitud	Recall	F1-Score
0	0.97	0.91	0.94	0.97	0.93	0.95
1	0.76	1.0	0.86	0.76	1.0	0.86
2	0.63	0.41	0.50	0.66	0.41	0.51
3	0.89	0.87	0.88	0.89	0.87	0.88
Rendimiento	0.83			0.83		
Tamaño	3.5 MB			941KB		

Cuadro 4.12: Rendimiento de clasificación del modelo obtenido por destilación de conocimiento con la arquitecura InceptionV3 - invernaderos

En el Cuadro 4.13, se puede apreciar que de la misma manera que en la aplicación anterior, usando Inception-V3 como arquitectura CNN y usando el algoritmos All-kNN en conjunto de autovectores para optimizar el conjunto de datos, el rendimiento de clasificación es el adecuado. No obstante, debido a que en esta aplicación IoT, no cuenta con demsiadas etiquetas, el algoritmo t-SNE demostró también ser útil a pesar que su eliminación de imágenes del conjunto de entrenamiento es muy agresiva.

Conjunto de datos	**Tamaño MB**	**Rendimiento**
ISO+t-NSE	10.3 MB	0.80
ISO+AUTO	10.4 MB	0.79
All-kNN+t-SNE	3.27 MB	0.80
All-kNN+AUTO	9.19 MB	0.81

Cuadro 4.13: Análisis de resultados de entrenamiento al emplear las bases de datos optimizados del conjunto de datos de invernadero

4.7. Análisis comparativo de los resultados obtenidos en los casos de estudio

Con las aplicaciones IoT establecidas y los requerimientos computacionales definidos, es esencial conocer el tiempo de procesamiento que cada dispositivo de borde debe cumplir. En este estudio,

se seleccionaron tres dispositivos de borde con diferentes características. El primer dispositivo es un Raspberry Pi 4, ampliamente conocido y empleado en diversas aplicaciones IoT. El segundo dispositivo incluye una GPU como acelerador de *hardware* de la marca NVIDIA. Este sistema en módulo cuenta con una GPU que interactúa con seis procesadores y dispone de 8 GB de RAM. El tercer dispositivo es el Coral TPU, que incorpora una unidad de procesamiento de tensores. Similar al dispositivo de NVIDIA, es un sistema en módulo que cuenta con cuatro procesadores y 1 GB de RAM. Se distingue por utilizar una versión muy ligera de Linux, que por defecto no permite configurar una interfaz de usuario avanzada. El resumen de sus principales características se presenta en el Cuadro 4.14.

	Dispositivos de borde		
Especificaciones	Raspberry Pi 4	Jetson Nano Orin	Coral Dev Board
CPU	Quad core Cortex-A72 64-bit SoC	6 ARM Cortex-A57	Quad Cortex-A53, Cortex-M4F
RAM	4GB SDRAM	8GB LPDDR4	1 GB LPDDR4
Almacenamiento	Micro-SD card (64 GB)	Micro-SD card (64 GB)	Micro-SD card (64 GB)
Wireless	2.4 GHz and 5.0 GHz IEEE 802.11ac wireless	2.4GHz 802.11 b/g/n wireless	802.11a/b/g/n/ac 2.4/5GHz
Conector de la cámara	2-lane MIPI CSI camera port	CSI-2 camera connector	MIPI-CSI2 camera input (4-lane)
Acelerador de *hardware*	Ninguna	128-core GPU y orin nano 40 TOPS	Google Edge TPU: 4 TOPS (int8)

Cuadro 4.14: Descripción de dispositivos de borde

En cada dispositivo de borde se compilaron los modelos obtenidos mediante la técnica Aprendizaje por Transferencia, denominados como modelos de gran volumen. Además, se compilaron los modelos livianos obtenidos al emplear la técnica de destilación de conocimiento, manteniendo la misma arquitectura que los modelos de gran volumen pero con un tamaño reducido. Las arquitecturas utilizadas fueron Inception-V3 y EfficientNetB0, debido a que demostraron tener el mejor rendimiento de clasificación. A estas arquitecturas también se les aplicaron técnicas de optimización para que pudieran ejecutar la inferencia del modelo a 32 y 8 *bits*. En la Figura 4.15 se muestran los resultados de compilar todos los modelos previamente mencionados en cada dispositivo. Todos ellos utilizaron *TensorFlow* y sus intérpretes para ejecutar los modelos.

Las métricas clave incluyeron el consumo de energía (muy relevante para la aplicación en nano satélites), el tiempo de procesamiento, el consumo de RAM y el porcentaje de uso de la CPU. Se observó que la versión sin cuantizar de Inception-V3 requiere 1.2 segundos para ejecutar el modelo de aprendizaje automático. Por su parte, las versiones livianas de Inception-V3 y EfficientNet cuantizadas a 8 *bits* necesitan 215 milisegundos y 320 milisegundos respectivamente cuando se ejecutan en el Raspberry Pi 4. En consecuencia, este dispositivo no cumple con los requisitos operativos para las aplicaciones IoT. En cuanto al consumo de memoria RAM, es aproximadamente 300 MB, y su consumo de energía no supera los 2 watts.

Por su parte, el dispositivo Coral TPU mostró un rendimiento similar al ejecutar los modelos de gran volumen no cuantizados. Sin embargo, la versión liviana y cuantizada de Inception-V3 puede procesar cada imagen en 121 milisegundos. Este resultado aún está lejos de los requisitos establecidos en las aplicaciones IoT, pero demuestra un avance significativo en la optimización de modelos, ya que el tiempo de ejecución por inferencia es 10 veces menor. Además, el modelo

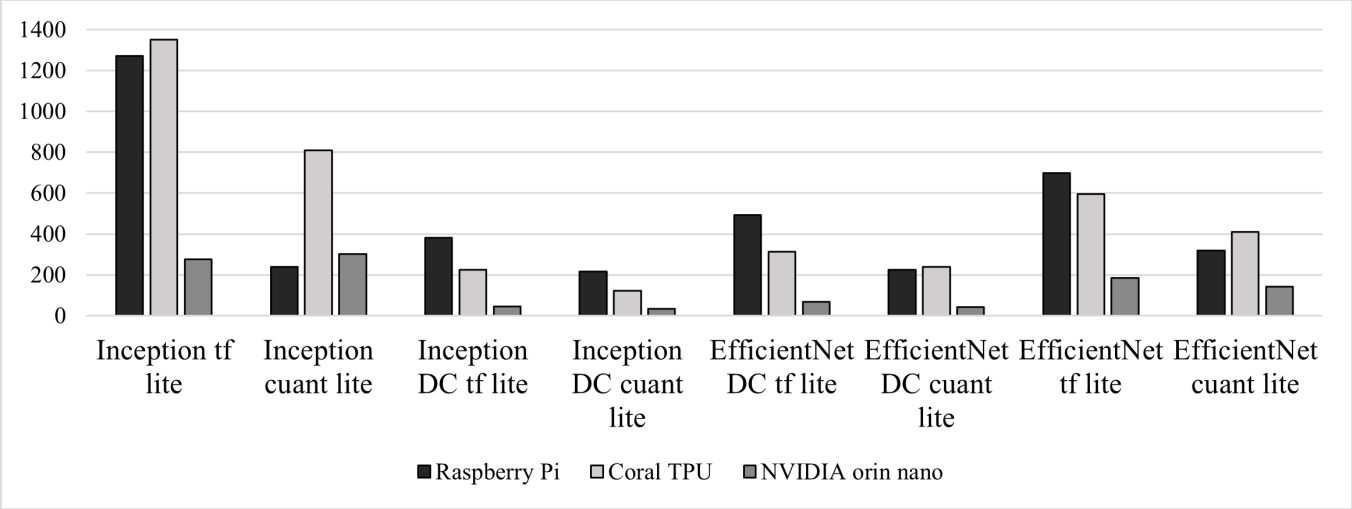


Figura 4.15: Tiempo de procesamiento de inferencias de los modelos de ML en los dispostivos de borde

EfficientNet presentó ciertos inconvenientes al interactuar con los procesadores del Coral TPU. Los desarrolladores de Coral TPU han señalado que, al ser un modelo complejo, EfficientNet requiere un mayor costo computacional. Por esta razón, no se pudo implementar este modelo en la TPU debido a la falta de soporte. En cambio, el modelo de gran volumen y el liviano de Inception-V3 en sus versiones de 8 *bits* fueron procesados en la TPU, con tiempos de ejecución por inferencia de 44.7 milisegundos y 2.65 milisegundos, respectivamente. En este caso, el modelo liviano es 20 veces más rápido y cumple con todos los requisitos computacionales de las aplicaciones IoT. Su funcionamiento se mantiene por debajo de los 1.7 watts.

El dispositivo NVIDIA también mostró resultados satisfactorios al ejecutar los diferentes modelos de aprendizaje automático. Las versiones no optimizadas de cada arquitectura no superan los 400 milisegundos. Las versiones livianas y optimizadas requieren 34 milisegundos para Inception-V3 y 42 milisegundos para EfficientNet. Al ejecutar estos modelos en la GPU, el modelo robusto de Inception-V3 a 32 *bits* toma 2.95 ms y el compilado a 8 *bits* 5.2 ms. Esto se debe a que la GPU tiene la capacidad de procesar modelos a 32 *bits*, pero necesita realizar cálculos adicionales al procesar datos de 8 *bits*. Sin embargo, el dispositivo NVIDIA consume más energía en comparación con el resto de los dispositivos, pudiendo llegar a los 6 Watts cuando la GPU está en funcionamiento. Estos resultados se pueden apreciar en la Figura 4.16.

Otro aspecto que es necesario considerar al elegir un dispositivo de borde fue el tiempo necesario para cargar el modelo en la RAM y su capacidad para interactuar con los procesadores y el acelerador gráfico. Específicamente, el Coral TPU requiere hasta 3 milisegundos para cargar el modelo en la RAM o en la TPU para realizar inferencias. En contraste, la GPU demostró ser más compleja y menos compatible con el *framework* de programación, ya que necesita entre 10 y 20 segundos para cargar el modelo. Durante este tiempo, el consumo de energía es elevado y, en ocasiones, con el modelo de aprendizaje automático basado en la arquitectura de EfficientNet, se saturó la memoria RAM de 8 GB. Estas razones llevaron a la selección del Coral TPU como dispositivo de borde, junto con el modelo liviano de Inception-V3. Finalmente, para visualizar gráficamente las optimizaciones realizadas en el modelo seleccionado, las Figuras 4.17 y 4.18 muestran el tamaño de cada arquitectura en relación con todas las capas empleadas.

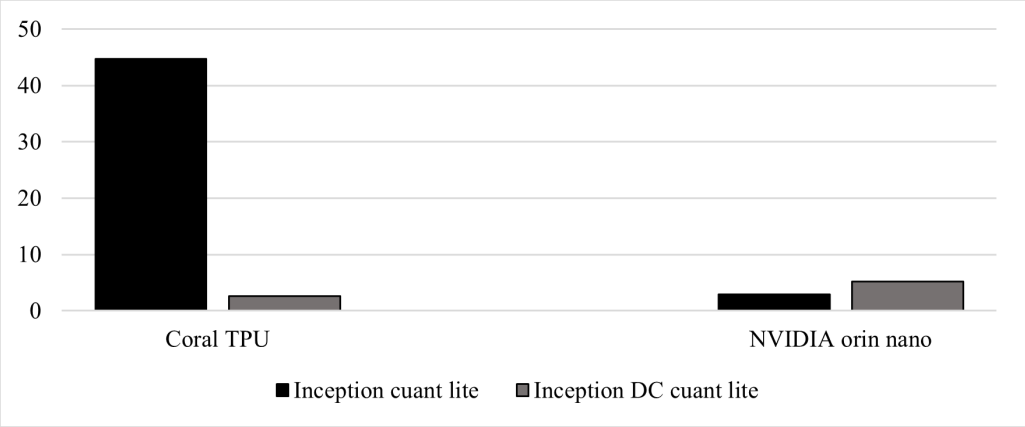


Figura 4.16: Tiempo de procesamiento de los modelo de ML usando aceleradores de *hardware*

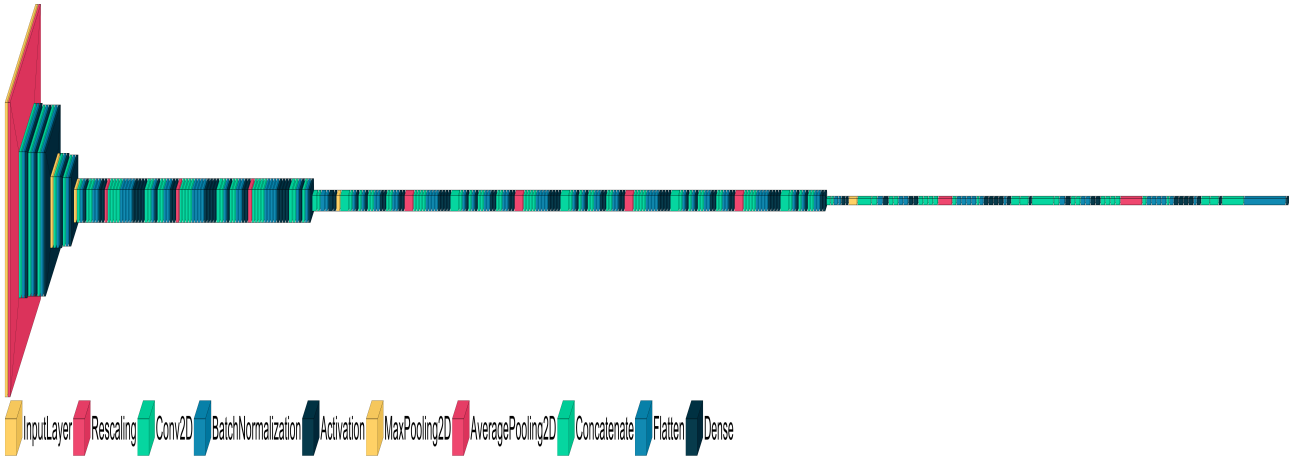


Figura 4.17: Tamaño del modelo con sus capas empleadas

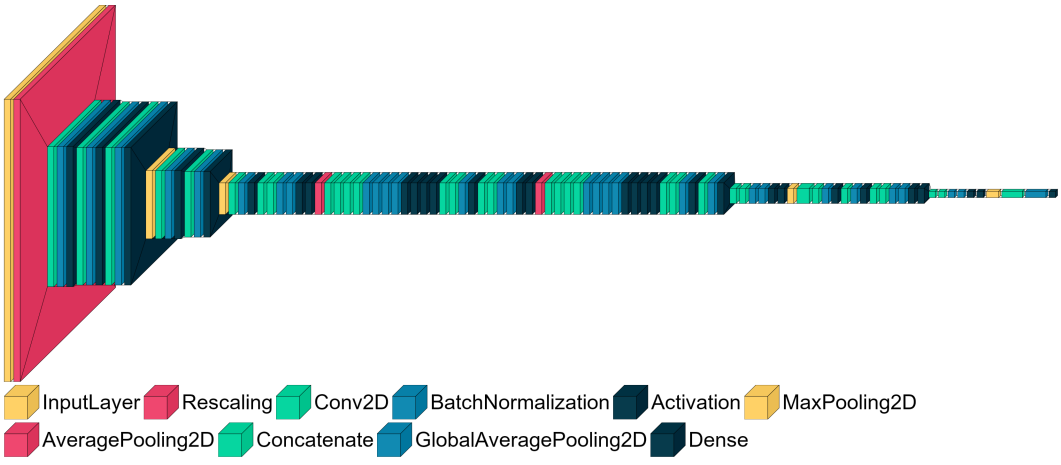


Figura 4.18: Tamaño del modelo optimizado con sus capas empleadas

4.7.1. Re-entrenamiento del modelo

Una vez definido el modelo y el dispositivo de borde, el re-entrenamiento es lo siguiente en la última fase de la propuesta planteada. En este caso, el modelo sólo actualizará la última capa del clasificador dentro del dispositivo con el conjunto de datos refinado, es decir, los que eliminaron

datos atípicos o datos que no son relevantes para el clasificador. El modelo debe ser re-entrenado a 32 *bits* ya que el *framework* de *TensorFlow* solo permite esta configuración.

Por lo tanto, el código de entrenamiento contiene los siguientes parámetros: los datos de entrada del modelo se dividen en batches de 32 imágenes de forma aleatoria con cada etiqueta donde el 20 % de ellos son empleados para validación y observar si el modelo se sobreajusta; se definen 15 épocas para entrenar el modelo. Posteriormente se empleará la función de *Callbacks* (función para monitorear el progreso del entrenamiento en determinados intervalos) con ello conoceremos si la red neuronal sigue aprendiendo con una espera de 5 épocas, caso contrario el entrenamiento se detiene. Luego, el modelo es congelado, excepto la última capa donde se encuentra el clasificador, éste es sustituido por una capa adicional que contienen las nuevas neuronas que permitirán la nueva clasificación. Su función de activación es *softmax* ya que es una clasificación multiclase. A continuación, un nuevo modelo es creado donde se adjunta el modelo congelado sin la última capa y la nueva capa de clasificación de forma secuencial. Para entrenar el modelo, se emplean los datos de entrenamiento y validación, y los pasos por épocas son el tamaño del conjunto de entrenamiento para evitar que el sistema se sobreajuste al modelo.

Este proceso de ha realizado en los dos casos de estudio, es decir con el conjunto de datos de imágenes satelitales y con el conjunto de datos para detección de enfermedades en invernaderos. Donde el rendimiento con el conjunto de datos de imágenes satelitales fue de 73 % y de cultivos en invernaderos del 82 %. Cumpliendo el objetivo de mantener un alto rendimiento de clasificación pero con modelos muy livianos. Cabe mencionar que el modelo proveniente de la arquitectura Inception-V3, pesa 3.7KB.

4.8. Discusión

Uno de los principales desafíos atravesados durante la implementación de los casos de estudio fue ajustar el tamaño de las imágenes para las arquitecturas de CNN. Esto se debe a que los modelos reciben imágenes de 224 x 224 píxeles, mientras que las cámaras comúnmente usadas en dispositivos IoT y de borde operan con resoluciones mayores, lo que implica un mayor consumo de tiempo y recursos para redimensionar las imágenes. Además, algunas cámaras solo capturan en blanco y negro, mientras que los modelos están diseñados para procesar imágenes en RGB. Estas consideraciones son cruciales en el diseño de soluciones IoT, aunque a menudo no se tienen en cuenta. Abordar estos desafíos es crucial para optimizar el rendimiento y la eficiencia de los sistemas implementados, garantizando un adecuado tiempo de procesamiento y buena precisión.

Por otra lado, al analizar los *benchmarks* disponibles, se evidencia que la mayoría de ellos llevan a cabo sus comparaciones y entrenan las diferentes arquitecturas empleando conjuntos de datos sintéticos. Si bien esta práctica es útil para entrenar modelos y proporcionar estimaciones de precisión y rendimiento, los resultados en escenarios reales distan en gran medida. Por lo tanto, la implementación de arquitecturas CNN en entornos reales tiende a tener un rendimiento menor al esperado según los *benchmarks*. Lograr mejores rendimientos en las aplicaciones de IoT requerirá no solo mejores recursos computacionales sino también mayor consumo de energía, lo cual en este contexto es muy limitado.

CAPÍTULO 5: CONCLUSIONES

Este capítulo expone las conclusiones derivadas del presente trabajo doctoral y analiza la validez de la hipótesis formulada. Mediante la implementación de la arquitectura en los casos de estudio, se valida la arquitectura propuesta y se presentan las principales contribuciones de la misma. Finalmente, se plantean algunas líneas para futuros trabajos.

Tal como se planteó en la hipótesis en el Capítulo 1, la presente investigación propone una arquitectura de procesamiento acelerado de datos para entornos IoT, capaz de generar un conjunto de datos depurado que proporcione información relevante para modelos de aprendizaje automático y que pueda ser ejecutada eficientemente en dispositivos de borde, considerando sus configuraciones específicas de *hardware* y *software*. Para contrastar esta hipótesis, se estableció un objetivo general y varios específicos, de los cuales se discute a continuación.

Como punto de partida, se realizó una exhaustiva revisión del estado del arte respecto a arquitecturas que integran IoT y Computación de Borde. El análisis comparativo de las diferentes arquitecturas presentadas en la comunidad científica, y cómo éstas interactúan con diferentes dispositivos electrónicos y algoritmos de aprendizaje automático permitió detectar ciertos desafíos que esta tesis los trata de la siguiente forma. Primero, existe una confusión al emplear los términos *dispositivo IoT* o *dispositivo de borde* debido a que no existe un parámetro específico que los diferencie. Por ello, en esta investigación se establece un factor diferenciador en función a sus capacidades de interactuar con los algoritmos de aprendizaje automático. En consecuencia, un dispositivo IoT puede ejecutar únicamente las inferencias del modelo ya que usualmente tiene un microprocesador o microcontrolador que no tiene las capacidades computacionales necesarias para re-entrenar modelos. En contraste, un dispositivo de borde cuenta con varios procesadores trabajando en conjunto con un sistema operativo administrando sus recursos, por ello lo hace viable para realizar el re-entrenamiento de modelos de aprendizaje automático.

La mayoría de las arquitecturas o plataformas para evaluar los modelos de aprendizaje automático en dispositivos de borde lo hacen usualmente con conjuntos de datos obtenidos en laboratorio. Es decir, sus características se acoplan a las necesidades de los modelos de aprendizaje automático y están muy lejos de las condiciones reales que los dispositivos de borde deben operar. Es así que esta tesis doctoral empieza con conjuntos de datos sintéticos o de laboratorio para tener un rendimiento general de la propuesta de procesamiento de datos acelerado para luego aplicarla en entornos IoT. Como resultado, se presenta información relevante no solo del lado de una comparativa teórica, sino también presenta una comparativa de aplicación.

En lo relativo al análisis de técnicas de preprocesamiento, se pusieron a prueba diversas herramientas, de las cuales podemos concluir que las técnicas de preprocesamiento de imágenes que usualmente se emplean para mejorar el rendimiento de los algoritmos de aprendizaje automático

se centran en la confianza de que cada imagen proporciona información relevante al clasificador. Sin embargo, desde el punto de vista de esta tesis doctoral, antes de emplear dichos criterios, es importante evaluar la relevancia de cada imagen y cómo cada una de ellas afecta en el tiempo de entrenamiento de un modelo con respecto a su rendimiento. Se implementaron criterios de reducción de dimensionalidad para aplicar algoritmos de selección de prototipos y detección de datos atípicos para reducir el tamaño del conjunto de entrenamiento sin sacrificar el rendimiento. Como resultado, al emplear algoritmos por transferencia de conocimiento, es posible reducir hasta en un 60 % aplicando las técnicas de *One-Class SVM* o *Isolation Forest* una vez que las imágenes se encuentren en 2 dimensiones aplicando autovectores. Este conjunto de datos refinado puede mejorar el rendimiento de clasificación. Esto evidencia la relevancia de este trabajo y presenta nuevas líneas de investigación sobre la correcta selección de datos para entrenar modelos de aprendizaje automático.

Con la arquitectura de datos propuesta y una vez que han sido optimizados los modelos de aprendizaje automático, fue posible comprobar que partiendo de modelos pre-entrenados por medio de transferencia de conocimiento, se evita realizar múltiples pruebas para definir una correcta arquitectura de capas convolucionales, empleando arquitecturas ampliamente estudiadas. Además, cada capa se encuentra con los valores óptimos para la extración de características, lo cuál evita un excesivo tiempo de entrenamiento.

Por otra parte, los criterios de optimización, como la destilación de conocimiento y la cuantización, jugaron un papel fundamental. Por un lado, permitieron reducir la complejidad del modelo al entrenar un modelo livino a partir de uno pre-entrenado. La cuantización, por su parte, permitió la reducción de la resolución del modelo a 8 *bits* y la utilización de aceleradores de *hardware* para mejorar su procesamiento. Después de la inferencia, si el modelo decae en su rendimiento, un dispositivo de borde tiene la capacidad de re-entrenar un modelo liviano con un conjunto refinado para mantener su rendimiento.

Una vez establecidos los criterios de optimización de los modelos de aprendizaje automático y refinados los datos, se plantearon dos aplicaciones IoT donde el tiempo de ejecución es el parámetro más relevante. Durante la implementación se evidenció que los aceleradores de *hardware* son múy útiles, aunque los modelos de Aprendizaje por Transferencia enfrentan desafíos en entornos reales. Por un lado, estos modelos suelen ser demasiado complejos para ser procesados en el tiempo establecido, y por otro, los modelos livianos no alcanzan el rendimiento requerido para una aplicación en condiciones reales.

5.1. Contribuciones de la investigación

El proceso de investigación realizado durante esta tesis doctoral en respuesta a la hipótesis establecida y para satisfacer los objetivos planteados en la introducción obtuvo como resultado una serie de aportaciones al estado del arte, de manera especial identificando las diferencias entre dispositivos IoT y de borde. Además con el estudio de las arquitecturas existentes fue posible definir un enfoque novedoso para una nueva arquitectura que abarque los modelos de CNN empleando técnicas de preprocesamiento y transferencia de conocimiento.

Para la fase de ingeniería de datos ha sido necesario un análisis exhaustivo de las técnicas de reducción de dimensionalidad y selección de prototipos para contar con conjuntos de datos refinados. Por otra parte, la flexibilidad de la arquitectura propuesta permite la inclusión de casos de

estudio con conjuntos de datos de aplicaciones IoT en escenarios reales.

De manerea concreta, las principales contribuciones de esta investigación se presentan a continuación:

- Se ha propuesto una etapa de afinamiento del conjunto de datos para reducir el tamaño del conjunto de entrenamiento manteniendo el rendimiento de clasificación. Para ello, reducir la dimensionalidad de las imágenes para aplicar algoritmos tradicionales de selección de prototipos y detección de datos atípicos tuvo un gran resultado y presenta una novedosa forma de emplear estos criterios en Redes Neuronales Convolucionales.
- En el análisis sobre los trabajos relacionados se pudo notar que existen pocas contribuciones sobre destilación de conocimiento aplicado como paso previo a la cuantización. Esto debido a que es un criterio que necesita un amplio conocimiento sobre las arquitecturas de redes neuronales y su funcionamiento. Por esta razón, la contribución se enfoca en la reducción de 10 veces el tamaño en modelos pre-entrenados, para aplicarlos a dispositivos de borde, su re-entrenamiento en entornos IoT y con un tiempo de ejecución debajo de los 10 milisegundos.

5.2. Líneas para trabajos futuros

Como parte final de esta tesis doctoral, en esta sección se proponen diversas líneas de ampliación de la investigación que permitirán la experimentación en otras aplicaciones IoT. La primera propuesta de ampliación y continuación de este trabajo está relacionada con las herramientas de optimización de modelos de aprendizaje automático. De manera especial se podría profundizar en mejorar las técnicas de destilación del conocimiento para reducir el tamaño de los modelos y que sean capaces de ejecutarse en dispositivos de *hardware* restringido.

Por otra parte, a nivel de *hardware*, se puede proponer un amplio análisis de cámaras disponibles en el mercado y que sean compatibles con aplicaciones de IoT, debido a que son la entrada para los conjuntos de datos que se ejecutarán con la arquitectura propuesta. Al respecto, se pueden analizar las siguientes características: resolución (compatibilidad con el tamaño de entrada de los modelos de CNN); número de frames por segundo (indispensable para calcular el tiempo de procesamiento en el sistema); compatibilidad de conexión (USB, Ethernet, WiFi, etc); consumo de energía; capacidad de procesamiento (algunas cámaras incluyen algoritmos de visión por computador integradas).

BIBLIOGRAFÍA

[1] R. Lohiya and A. Thakkar, "Application Domains, Evaluation Data Sets, and Research Challenges of IoT: A Systematic Review," *IEEE Internet of Things Journal*, vol. 8, no. 11, pp. 8774–8798, Jun. 2021. [Online]. Available: https://ieeexplore.ieee.org/document/9311636/

[2] M. N. Khan, A. Rao, and S. Camtepe, "Lightweight Cryptographic Protocols for IoT-Constrained Devices: A Survey," *IEEE Internet of Things Journal*, vol. 8, no. 6, pp. 4132–4156, Mar. 2021. [Online]. Available: https://ieeexplore.ieee.org/document/9205259/

[3] I. Kok, M. U. Simsek, and S. Ozdemir, "A deep learning model for air quality prediction in smart cities," in *2017 IEEE International Conference on Big Data (Big Data)*. Boston, MA, USA: IEEE, Dec. 2017, pp. 1983–1990. [Online]. Available: http://ieeexplore.ieee.org/document/8258144/

[4] M. Taştan and H. Gökozan, "Real-Time Monitoring of Indoor Air Quality with Internet of Things-Based E-Nose," *Applied Sciences*, vol. 9, no. 16, p. 3435, Aug. 2019. [Online]. Available: https://www.mdpi.com/2076-3417/9/16/3435

[5] Z. Chang, S. Liu, X. Xiong, Z. Cai, and G. Tu, "A Survey of Recent Advances in Edge-Computing-Powered Artificial Intelligence of Things," *IEEE Internet of Things Journal*, vol. 8, no. 18, pp. 13 849–13 875, Sep. 2021. [Online]. Available: https://ieeexplore.ieee.org/document/9453402/

[6] S. Yu, X. Chen, Z. Zhou, X. Gong, and D. Wu, "When Deep Reinforcement Learning Meets Federated Learning: Intelligent Multitimescale Resource Management for Multiaccess Edge Computing in 5G Ultradense Network," *IEEE Internet of Things Journal*, vol. 8, no. 4, pp. 2238–2251, Feb. 2021. [Online]. Available: https://ieeexplore.ieee.org/document/9205252/

[7] P. Chhikara, R. Tekchandani, N. Kumar, M. Guizani, and M. M. Hassan, "Federated Learning and Autonomous UAVs for Hazardous Zone Detection and AQI Prediction in IoT Environment," *IEEE Internet of Things Journal*, vol. 8, no. 20, pp. 15 456–15 467, Oct. 2021. [Online]. Available: https://ieeexplore.ieee.org/document/9409140/

[8] D. Zhang and S. S. Woo, "Real Time Localized Air Quality Monitoring and Prediction Through Mobile and Fixed IoT Sensing Network," *IEEE Access*, vol. 8, pp. 89 584–89 594, 2020. [Online]. Available: https://ieeexplore.ieee.org/document/9090830/

[9] L. Ravaglia, M. Rusci, D. Nadalini, A. Capotondi, F. Conti, and L. Benini, "A TinyML Platform for On-Device Continual Learning With Quantized Latent Replays," *IEEE Journal on Emerging and Selected Topics in Circuits and Systems*, vol. 11, no. 4, pp. 789–802, Dec. 2021. [Online]. Available: https://ieeexplore.ieee.org/document/9580920/

[10] C. R. Banbury, V. J. Reddi, M. Lam, W. Fu, A. Fazel, J. Holleman, X. Huang, R. Hurtado, D. Kanter, A. Lokhmotov, D. Patterson, D. Pau, J.-s. Seo, J. Sieracki,

U. Thakker, M. Verhelst, and P. Yadav, "Benchmarking TinyML Systems: Challenges and Direction," *arXiv:2003.04821 [cs]*, Jan. 2021, arXiv: 2003.04821. [Online]. Available: http://arxiv.org/abs/2003.04821

[11] A. Al-Fuqaha, M. Guizani, M. Mohammadi, M. Aledhari, and M. Ayyash, "Internet of Things: A Survey on Enabling Technologies, Protocols, and Applications," *IEEE Communications Surveys & Tutorials*, vol. 17, no. 4, pp. 2347–2376, 2015. [Online]. Available: https://ieeexplore.ieee.org/document/7123563/

[12] S. Ali, T. Glass, B. Parr, J. Potgieter, and F. Alam, "Low Cost Sensor With IoT LoRaWAN Connectivity and Machine Learning-Based Calibration for Air Pollution Monitoring," *IEEE Transactions on Instrumentation and Measurement*, vol. 70, pp. 1–11, 2021. [Online]. Available: https://ieeexplore.ieee.org/document/9241043/

[13] Z. Idrees, Z. Zou, and L. Zheng, "Edge Computing Based IoT Architecture for Low Cost Air Pollution Monitoring Systems: A Comprehensive System Analysis, Design Considerations & Development," *Sensors*, vol. 18, no. 9, p. 3021, Sep. 2018. [Online]. Available: http://www.mdpi.com/1424-8220/18/9/3021

[14] K. Guo, S. Zeng, J. Yu, Y. Wang, and H. Yang, "[DL] A Survey of FPGA-based Neural Network Inference Accelerators," *ACM Transactions on Reconfigurable Technology and Systems*, vol. 12, no. 1, pp. 1–26, Mar. 2019. [Online]. Available: https://dl.acm.org/doi/10.1145/3289185

[15] M. R. Azghadi, C. Lammie, J. K. Eshraghian, M. Payvand, E. Donati, B. Linares-Barranco, and G. Indiveri, "Hardware Implementation of Deep Network Accelerators Towards Healthcare and Biomedical Applications," *IEEE Transactions on Biomedical Circuits and Systems*, vol. 14, no. 6, pp. 1138–1159, Dec. 2020. [Online]. Available: https://ieeexplore.ieee.org/document/9250613/

[16] S. K. Lee, P. N. Whatmough, M. Donato, G. G. Ko, D. Brooks, and G.-Y. Wei, "SMIV: A 16-nm 25-mm^2 SoC for IoT With Arm Cortex-A53, eFPGA, and Coherent Accelerators," *IEEE Journal of Solid-State Circuits*, vol. 57, no. 2, pp. 639–650, Feb. 2022. [Online]. Available: https://ieeexplore.ieee.org/document/9566303/

[17] L. Du, Y. Du, Y. Li, J. Su, Y.-C. Kuan, C.-C. Liu, and M.-C. F. Chang, "A Reconfigurable Streaming Deep Convolutional Neural Network Accelerator for Internet of Things," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 65, no. 1, pp. 198–208, Jan. 2018. [Online]. Available: http://ieeexplore.ieee.org/document/8011462/

[18] N. Abbas, Y. Zhang, A. Taherkordi, and T. Skeie, "Mobile Edge Computing: A Survey," *IEEE Internet of Things Journal*, vol. 5, no. 1, pp. 450–465, Feb. 2018. [Online]. Available: http://ieeexplore.ieee.org/document/8030322/

[19] X. Liu, J. Yang, C. Zou, Q. Chen, X. Yan, Y. Chen, and C. Cai, "Collaborative Edge Computing With FPGA-Based CNN Accelerators for Energy-Efficient and Time-Aware Face Tracking System," *IEEE Transactions on Computational Social Systems*, vol. 9, no. 1, pp. 252–266, Feb. 2022. [Online]. Available: https://ieeexplore.ieee.org/document/9363321/

[20] D. Lee, S. Lee, S. Oh, and D. Park, "Energy-Efficient FPGA Accelerator With Fidelity-Controllable Sliding-Region Signal Processing Unit for Abnormal ECG Diagnosis on IoT Edge Devices," *IEEE Access*, vol. 9, pp. 122 789–122 800, 2021. [Online]. Available: https://ieeexplore.ieee.org/document/9528369/

[21] A. L. Samuel, "Some Studies in Machine Learning Using the Game of Checkers," 1959.

[22] T. M. Mitchell, *Machine Learning*, ser. McGraw-Hill series in computer science. New York: McGraw-Hill, 1997.

[23] V. Nasteski, "An overview of the supervised machine learning methods," *HORIZONS.B*, vol. 4, pp. 51–62, Dec. 2017. [Online]. Available: http://uklo.edu.mk/filemanager/HORIZONTI%202017/Serija%20B%20br.%204/6.An%20overview%20of%20the%20supervised.pdf

[24] T. K. Rodrigues, K. Suto, H. Nishiyama, J. Liu, and N. Kato, "Machine Learning Meets Computation and Communication Control in Evolving Edge and Cloud: Challenges and Future Perspective," *IEEE Communications Surveys & Tutorials*, vol. 22, no. 1, pp. 38–67, 2020. [Online]. Available: https://ieeexplore.ieee.org/document/8847416/

[25] A. Anaya-Isaza and L. Mera-Jimenez, "Data Augmentation and Transfer Learning for Brain Tumor Detection in Magnetic Resonance Imaging," *IEEE Access*, vol. 10, pp. 23 217–23 233, 2022. [Online]. Available: https://ieeexplore.ieee.org/document/9720964/

[26] Z. Wang, L. Du, J. Mao, B. Liu, and D. Yang, "SAR Target Detection Based on SSD With Data Augmentation and Transfer Learning," *IEEE Geoscience and Remote Sensing Letters*, vol. 16, no. 1, pp. 150–154, Jan. 2019. [Online]. Available: https://ieeexplore.ieee.org/document/8467523/

[27] P. D. Rosero-Montalvo, D. H. Peluffo-Ordóñez, V. F. López Batista, J. Serrano, and E. A. Rosero, "Intelligent System for Identification of Wheelchair User's Posture Using Machine Learning Techniques," *IEEE Sensors Journal*, vol. 19, no. 5, pp. 1936–1942, Mar. 2019. [Online]. Available: https://ieeexplore.ieee.org/document/8565996/

[28] L. van der Maaten, "Learning a parametric embedding by preserving local structure," in *Proceedings of the Twelth International Conference on Artificial Intelligence and Statistics*, ser. Proceedings of Machine Learning Research, D. van Dyk and M. Welling, Eds., vol. 5. Hilton Clearwater Beach Resort, Clearwater Beach, Florida USA: PMLR, 16–18 Apr 2009, pp. 384–391. [Online]. Available: https://proceedings.mlr.press/v5/maaten09a.html

[29] L. Xie, L. Zheng, Z. Liu, and Y. Zhang, "Laplacian Eigenmaps for Automatic Story Segmentation of Broadcast News," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 20, no. 1, pp. 276–289, Jan. 2012. [Online]. Available: http://ieeexplore.ieee.org/document/5934585/

[30] F. Anowar, S. Sadaoui, and B. Selim, "Conceptual and empirical comparison of dimensionality reduction algorithms (PCA, KPCA, LDA, MDS, SVD, LLE, ISOMAP, LE, ICA, t-SNE)," *Computer Science Review*, vol. 40, p. 100378, May 2021. [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/S1574013721000186

[31] G. Chao, S. Sun, and J. Bi, "A Survey on Multiview Clustering," *IEEE Transactions on Artificial Intelligence*, vol. 2, no. 2, pp. 146–168, Apr. 2021. [Online]. Available: https://ieeexplore.ieee.org/document/9395530/

[32] G. Pedretti, P. Mannocci, C. Li, Z. Sun, J. P. Strachan, and D. Ielmini, "Redundancy and Analog Slicing for Precise In-Memory Machine Learning—Part II: Applications and Benchmark," *IEEE Transactions on Electron Devices*, vol. 68, no. 9, pp. 4379–4383, Sep. 2021. [Online]. Available: https://ieeexplore.ieee.org/document/9495837/

[33] P. Liu, K. Jin, Y. Jiao, M. He, and S. Fei, "Prediction of Second Primary Lung Cancer Patient's Survivability Based on Improved Eigenvector Centrality-Based Feature Selection," *IEEE Access*, vol. 9, pp. 55 663–55 672, 2021. [Online]. Available: https://ieeexplore.ieee.org/document/9370095/

[34] P. D. Rosero-Montalvo, V. F. López-Batista, D. H. Peluffo-Ordóñez, V. C. Erazo-Chamorro, and R. P. Arciniega-Rocha, "Multivariate approach to alcohol detection in drivers by sensors and artificial vision," in *From Bioinspired Systems and Biomedical Applications to Machine Learning*, J. M. Ferrández Vicente, J. R. Álvarez-Sánchez, F. de la Paz López, J. Toledo Moreo, and H. Adeli, Eds. Cham: Springer International Publishing, 2019, pp. 234–243.

[35] P. Rosero-Montalvo, D. H. Peluffo-Ordonez, A. Umaquinga, A. Anaya, J. Serrano, E. Rosero, C. Vasquez, and L. Suarez, "Prototype reduction algorithms comparison in nearest neighbor classification for sensor data: Empirical study," in *2017 IEEE Second Ecuador Technical Chapters Meeting (ETCM)*. Salinas: IEEE, Oct. 2017, pp. 1–5. [Online]. Available: http://ieeexplore.ieee.org/document/8247530/

[36] P. Hart, "The condensed nearest neighbor rule (corresp.)," *IEEE Transactions on Information Theory*, vol. 14, no. 3, pp. 515–516, 1968.

[37] G. Chatzimilioudis, C. Costa, D. Zeinalipour-Yazti, W.-C. Lee, and E. Pitoura, "Distributed in-memory processing of all k nearest neighbor queries," *IEEE Transactions on Knowledge and Data Engineering*, vol. 28, no. 4, pp. 925–938, 2016.

[38] F. T. Liu, K. M. Ting, and Z.-H. Zhou, "Isolation Forest," in *2008 Eighth IEEE International Conference on Data Mining*. Pisa, Italy: IEEE, Dec. 2008, pp. 413–422. [Online]. Available: http://ieeexplore.ieee.org/document/4781136/

[39] S. Hariri, M. C. Kind, and R. J. Brunner, "Extended isolation forest," *IEEE Transactions on Knowledge and Data Engineering*, vol. 33, no. 4, pp. 1479–1489, 2021.

[40] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, pp. 436–444, May 2015. [Online]. Available: https://www.nature.com/articles/nature14539

[41] R. C. Gonzalez and R. E. Woods, *Digital image processing*. New York, NY: Pearson, 2018.

[42] V. Sze, Y.-H. Chen, T.-J. Yang, and J. S. Emer, "Efficient Processing of Deep Neural Networks: A Tutorial and Survey," *Proceedings of the IEEE*, vol. 105, no. 12, pp. 2295–2329, Dec. 2017. [Online]. Available: http://ieeexplore.ieee.org/document/8114708/

[43] K. H. Jin, M. T. McCann, E. Froustey, and M. Unser, "Deep Convolutional Neural Network for Inverse Problems in Imaging," *IEEE Transactions on Image Processing*, vol. 26, no. 9, pp. 4509–4522, Sep. 2017. [Online]. Available: http://ieeexplore.ieee.org/document/7949028/

[44] D. Bhatt, C. Patel, H. Talsania, J. Patel, R. Vaghela, S. Pandya, K. Modi, and H. Ghayvat, "CNN Variants for Computer Vision: History, Architecture, Application, Challenges and Future Scope," *Electronics*, vol. 10, no. 20, p. 2470, Oct. 2021. [Online]. Available: https://www.mdpi.com/2079-9292/10/20/2470

[45] L. Alzubaidi, J. Zhang, A. J. Humaidi, A. Al-Dujaili, Y. Duan, O. Al-Shamma, J. Santamaría, M. A. Fadhel, M. Al-Amidie, and L. Farhan, "Review of deep learning: concepts, CNN architectures, challenges, applications, future directions," *Journal of Big Data*, vol. 8, no. 1, p. 53, Mar. 2021. [Online]. Available: https://journalofbigdata.springeropen.com/articles/10.1186/s40537-021-00444-8

[46] X. Piao, D. Synn, J. Park, and J.-K. Kim, "Enabling Large Batch Size Training for DNN Models Beyond the Memory Limit While Maintaining Performance," *IEEE Access*, vol. 11, pp. 102 981–102 990, 2023. [Online]. Available: https://ieeexplore.ieee.org/document/10242106/

[47] K. Simonyan and A. Zisserman, "Very Deep Convolutional Networks for Large-Scale Image Recognition," Apr. 2015, arXiv:1409.1556 [cs]. [Online]. Available: http://arxiv.org/abs/1409.1556

[48] H. Yang, J. Ni, J. Gao, Z. Han, and T. Luan, "A novel method for peanut variety identification and classification by Improved VGG16," *Scientific Reports*, vol. 11, no. 1, p. 15756, Aug. 2021. [Online]. Available: https://www.nature.com/articles/s41598-021-95240-y

[49] Z. Qu, J. Mei, L. Liu, and D.-Y. Zhou, "Crack Detection of Concrete Pavement With Cross-Entropy Loss Function and Improved VGG16 Network Model," *IEEE Access*, vol. 8, pp. 54 564–54 573, 2020. [Online]. Available: https://ieeexplore.ieee.org/document/9040553/

[50] K. He, X. Zhang, S. Ren, and J. Sun, "Deep Residual Learning for Image Recognition," Dec. 2015, arXiv:1512.03385 [cs]. [Online]. Available: http://arxiv.org/abs/1512.03385

[51] A. Saber, M. Sakr, O. M. Abo-Seida, A. Keshk, and H. Chen, "A Novel Deep-Learning Model for Automatic Detection and Classification of Breast Cancer Using the Transfer-Learning Technique," *IEEE Access*, vol. 9, pp. 71 194–71 209, 2021. [Online]. Available: https://ieeexplore.ieee.org/document/9427477/

[52] S. K. Khare and V. Bajaj, "Time–Frequency Representation and Convolutional Neural Network-Based Emotion Recognition," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 32, no. 7, pp. 2901–2909, Jul. 2021. [Online]. Available: https://ieeexplore.ieee.org/document/9153955/

[53] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, "Rethinking the Inception Architecture for Computer Vision," in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. Las Vegas, NV, USA: IEEE, Jun. 2016, pp. 2818–2826. [Online]. Available: http://ieeexplore.ieee.org/document/7780677/

[54] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen, "MobileNetV2: Inverted Residuals and Linear Bottlenecks," Mar. 2019, arXiv:1801.04381 [cs]. [Online]. Available: http://arxiv.org/abs/1801.04381

[55] X. Chen, J. Chen, X. Han, C. Zhao, D. Zhang, K. Zhu, and Y. Su, "A Light-Weighted CNN Model for Wafer Structural Defect Detection," *IEEE Access*, vol. 8, pp. 24 006–24 018, 2020. [Online]. Available: https://ieeexplore.ieee.org/document/8976072/

[56] M. Tan and Q. V. Le, "EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks," Sep. 2020, arXiv:1905.11946 [cs, stat]. [Online]. Available: http://arxiv.org/abs/1905.11946

[57] ——, "EfficientNetV2: Smaller Models and Faster Training," Jun. 2021, arXiv:2104.00298 [cs]. [Online]. Available: http://arxiv.org/abs/2104.00298

[58] P. D. Rosero-Montalvo, P. Tözün, and W. Hernandez, "Optimized CNN Architectures Benchmarking in Hardware-Constrained Edge Devices in IoT Environments," *IEEE Internet of Things Journal*, pp. 1–1, 2024. [Online]. Available: https://ieeexplore.ieee.org/document/10444008/

[59] S. A. H. Minoofam, A. Bastanfard, and M. R. Keyvanpour, "TRCLA: A Transfer Learning Approach to Reduce Negative Transfer for Cellular Learning Automata," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 34, no. 5, pp. 2480–2489, May 2023. [Online]. Available: https://ieeexplore.ieee.org/document/9527390/

[60] Y. Tao, J. Qiu, and S. Lai, "A Hybrid Cloud and Edge Control Strategy for Demand Responses Using Deep Reinforcement Learning and Transfer Learning," *IEEE Transactions on Cloud Computing*, vol. 10, no. 1, pp. 56–71, Jan. 2022. [Online]. Available: https://ieeexplore.ieee.org/document/9557846/

[61] G. Peng, H. Wu, H. Wu, and K. Wolter, "Constrained multiobjective optimization for iot-enabled computation offloading in collaborative edge and cloud computing," *IEEE Internet of Things Journal*, vol. 8, no. 17, pp. 13 723–13 736, 2021.

[62] T. Dubhir, M. Mishra, and R. Singhal, "Benchmarking of quantization libraries in popular frameworks," in *2021 IEEE International Conference on Big Data (Big Data)*, 2021, pp. 3050–3055.

[63] G. Li, X. Ma, X. Wang, H. Yue, J. Li, L. Liu, X. Feng, and J. Xue, "Optimizing deep neural networks on intelligent edge accelerators via flexible-rate filter pruning," *Journal of Systems Architecture*, vol. 124, p. 102431, Mar. 2022. [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/S1383762122000303

[64] L. Zhang, C. Bao, and K. Ma, "Self-Distillation: Towards Efficient and Compact Neural Networks," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pp. 1–1, 2021. [Online]. Available: https://ieeexplore.ieee.org/document/9381661/

[65] G. Li, J. Wang, H.-W. Shen, K. Chen, G. Shan, and Z. Lu, "*CNN* Pruner: Pruning Convolutional Neural Networks with Visual Analytics," *IEEE Transactions on Visualization and Computer Graphics*, vol. 27, no. 2, pp. 1364–1373, Feb. 2021. [Online]. Available: https://ieeexplore.ieee.org/document/9222510/

[66] Y. He, P. Liu, Z. Wang, Z. Hu, and Y. Yang, "Filter Pruning via Geometric Median for Deep Convolutional Neural Networks Acceleration," in *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. Long Beach, CA, USA: IEEE, Jun. 2019, pp. 4335–4344. [Online]. Available: https://ieeexplore.ieee.org/document/8953212/

[67] L. Wang and K.-J. Yoon, "Knowledge Distillation and Student-Teacher Learning for Visual Intelligence: A Review and New Outlooks," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 44, no. 6, pp. 3048–3068, Jun. 2022. [Online]. Available: https://ieeexplore.ieee.org/document/9340578/

[68] Y. Liu, C. Shu, J. Wang, and C. Shen, "Structured Knowledge Distillation for Dense Prediction," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 45, no. 6, pp. 7035–7049, Jun. 2023. [Online]. Available: https://ieeexplore.ieee.org/document/9115859/

[69] D. Qin, J.-J. Bu, Z. Liu, X. Shen, S. Zhou, J.-J. Gu, Z.-H. Wang, L. Wu, and H.-F. Dai, "Efficient Medical Image Segmentation Based on Knowledge Distillation," *IEEE Transactions on Medical Imaging*, vol. 40, no. 12, pp. 3820–3831, Dec. 2021. [Online]. Available: https://ieeexplore.ieee.org/document/9491090/

[70] Q. Xu, Z. Chen, K. Wu, C. Wang, M. Wu, and X. Li, "KDnet-RUL: A Knowledge Distillation Framework to Compress Deep Neural Networks for Machine Remaining Useful

Life Prediction," *IEEE Transactions on Industrial Electronics*, vol. 69, no. 2, pp. 2022–2032, Feb. 2022. [Online]. Available: https://ieeexplore.ieee.org/document/9351733/

[71] V. Alvear-Puertas, P. D. Rosero-Montalvo, V. Félix-López, and D. H. Peluffo-Ordóñez, "Edge Artificial Intelligence for Internet of Things Devices: Open Challenges," in *New Trends in Disruptive Technologies, Tech Ethics and Artificial Intelligence*, D. H. De La Iglesia, J. F. De Paz Santana, and A. J. López Rivero, Eds. Cham: Springer Nature Switzerland, 2023, vol. 1452, pp. 312–319, series Title: Advances in Intelligent Systems and Computing. [Online]. Available: https://link.springer.com/10.1007/978-3-031-38344-1_30

[72] L. Chettri and R. Bera, "A Comprehensive Survey on Internet of Things (IoT) Toward 5G Wireless Systems," *IEEE Internet of Things Journal*, vol. 7, no. 1, pp. 16–32, Jan. 2020. [Online]. Available: https://ieeexplore.ieee.org/document/8879484/

[73] I. Butun, P. Österberg, and H. Song, "Security of the internet of things: Vulnerabilities, attacks, and countermeasures," *IEEE Communications Surveys Tutorials*, vol. 22, no. 1, pp. 616–644, 2020.

[74] M. G. Samaila, J. B. F. Sequeiros, T. Simoes, M. M. Freire, and P. R. M. Inacio, "IoT-HarPSecA: A Framework and Roadmap for Secure Design and Development of Devices and Applications in the IoT Space," *IEEE Access*, vol. 8, pp. 16 462–16 494, 2020. [Online]. Available: https://ieeexplore.ieee.org/document/8957116/

[75] P. D. Rosero-Montalvo, P. Tözün, and W. Hernandez, "Time-Series Forecasting to Fill Missing Data in IoT Sensor Data," *IEEE Sensors Letters*, vol. 7, no. 9, pp. 1–4, Sep. 2023. [Online]. Available: https://ieeexplore.ieee.org/document/10225693/

[76] Z. Zhou, X. Chen, E. Li, L. Zeng, K. Luo, and J. Zhang, "Edge Intelligence: Paving the Last Mile of Artificial Intelligence With Edge Computing," *Proceedings of the IEEE*, vol. 107, no. 8, pp. 1738–1762, Aug. 2019. [Online]. Available: https://ieeexplore.ieee.org/document/8736011/

[77] P. D. Rosero-Montalvo, Z. István, and W. Hernandez, "A Survey of Trusted Computing Solutions Using FPGAs," *IEEE Access*, vol. 11, pp. 31 583–31 593, 2023. [Online]. Available: https://ieeexplore.ieee.org/document/10080920/

[78] K. Seshadri, B. Akin, J. Laudon, R. Narayanaswami, and A. Yazdanbakhsh, "An Evaluation of Edge TPU Accelerators for Convolutional Neural Networks," Oct. 2022, arXiv:2102.10423 [cs]. [Online]. Available: http://arxiv.org/abs/2102.10423

[79] R. Singh and S. S. Gill, "Edge AI: A survey," *Internet of Things and Cyber-Physical Systems*, vol. 3, pp. 71–92, 2023. [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/S2667345223000196

[80] X. Sun and N. Ansari, "EdgeIoT: Mobile Edge Computing for the Internet of Things," *IEEE Communications Magazine*, vol. 54, no. 12, pp. 22–29, Dec. 2016. [Online]. Available: https://ieeexplore.ieee.org/document/7786106/

[81] J. Ren, H. Guo, C. Xu, and Y. Zhang, "Serving at the Edge: A Scalable IoT Architecture Based on Transparent Computing," *IEEE Network*, vol. 31, no. 5, pp. 96–105, 2017. [Online]. Available: http://ieeexplore.ieee.org/document/8000809/

[82] G. Plastiras, C. Kyrkou, and T. Theocharides, "EdgeNet: Balancing Accuracy and Performance for Edge-based Convolutional Neural Network Object Detectors," in *Proceedings of the 13th International Conference on Distributed Smart Cameras*, Sep. 2019, pp. 1–6, arXiv:1911.06091 [cs]. [Online]. Available: http://arxiv.org/abs/1911.06091

[83] T. Qiu, J. Chi, X. Zhou, Z. Ning, M. Atiquzzaman, and D. O. Wu, "Edge Computing in Industrial Internet of Things: Architecture, Advances and Challenges," *IEEE Communications Surveys & Tutorials*, vol. 22, no. 4, pp. 2462–2488, 2020. [Online]. Available: https://ieeexplore.ieee.org/document/9139976/

[84] H. El-Sayed, S. Sankar, M. Prasad, D. Puthal, A. Gupta, M. Mohanty, and C.-T. Lin, "Edge of Things: The Big Picture on the Integration of Edge, IoT and the Cloud in a Distributed Computing Environment," *H. El*, vol. 6, 2018.

[85] S. M. Alrubei, E. Ball, and J. M. Rigelsford, "A Secure Blockchain Platform for Supporting AI-Enabled IoT Applications at the Edge Layer," *IEEE Access*, vol. 10, pp. 18 583–18 595, 2022. [Online]. Available: https://ieeexplore.ieee.org/document/9713867/

[86] S. Arisdakessian, O. A. Wahab, A. Mourad, H. Otrok, and M. Guizani, "A Survey on IoT Intrusion Detection: Federated Learning, Game Theory, Social Psychology, and Explainable AI as Future Directions," *IEEE Internet of Things Journal*, vol. 10, no. 5, pp. 4059–4092, Mar. 2023. [Online]. Available: https://ieeexplore.ieee.org/document/9872110/

[87] W. Hua, P. Liu, and L. Huang, "Energy-Efficient Resource Allocation for Heterogeneous Edge–Cloud Computing," *IEEE Internet of Things Journal*, vol. 11, no. 2, pp. 2808–2818, Jan. 2024. [Online]. Available: https://ieeexplore.ieee.org/document/10175541/

[88] S. Farrell, M. Emani, J. Balma, L. Drescher, A. Drozd, A. Fink, G. Fox, D. Kanter, T. Kurth, P. Mattson, D. Mu, A. Ruhela, K. Sato, K. Shirahata, T. Tabaru, A. Tsaris, J. Balewski, B. Cumming, T. Danjo, J. Domke, T. Fukai, N. Fukumoto, T. Fukushi, B. Gerofi, T. Honda, T. Imamura, A. Kasagi, K. Kawakami, S. Kudo, A. Kuroda, M. Martinasso, S. Matsuoka, H. Mendonca, K. Minami, P. Ram, T. Sawada, M. Shankar, T. S. John, A. Tabuchi, V. Vishwanath, M. Wahib, M. Yamazaki, and J. Yin, "MLPerf™ HPC: A Holistic Benchmark Suite for Scientific Machine Learning on HPC Systems," in *2021 IEEE/ACM Workshop on Machine Learning in High Performance Computing Environments (MLHPC)*. St. Louis, MO, USA: IEEE, Nov. 2021, pp. 33–45. [Online]. Available: https://ieeexplore.ieee.org/document/9653178/

[89] C. Zhang, F. Zhang, X. Guo, B. He, X. Zhang, and X. Du, "iMLBench: A Machine Learning Benchmark Suite for CPU-GPU Integrated Architectures," *IEEE Transactions on Parallel and Distributed Systems*, vol. 32, no. 7, pp. 1740–1752, Jul. 2021. [Online]. Available: https://ieeexplore.ieee.org/document/9305972/

[90] P. Mattson, V. J. Reddi, C. Cheng, C. Coleman, G. Diamos, D. Kanter, P. Micikevicius, D. Patterson, G. Schmuelling, H. Tang, G.-Y. Wei, and C.-J. Wu, "MLPerf: An Industry Standard Benchmark Suite for Machine Learning Performance," *IEEE Micro*, vol. 40, no. 2, pp. 8–16, Mar. 2020. [Online]. Available: https://ieeexplore.ieee.org/document/9001257/

[91] A. Tousi and M. Lujan, "Comparative Analysis of Machine Learning Models for Performance Prediction of the SPEC Benchmarks," *IEEE Access*, vol. 10, pp. 11 994–12 011, 2022. [Online]. Available: https://ieeexplore.ieee.org/document/9676614/

[92] V. J. Reddi, C. Cheng, D. Kanter, P. Mattson, G. Schmuelling, C.-J. Wu, B. Anderson, M. Breughe, M. Charlebois, W. Chou, R. Chukka, C. Coleman, S. Davis, P. Deng, G. Diamos, J. Duke, D. Fick, J. S. Gardner, I. Hubara, S. Idgunji, T. B. Jablin, J. Jiao, T. S. John, P. Kanwar, D. Lee, J. Liao, A. Lokhmotov, F. Massa, P. Meng, P. Micikevicius, C. Osborne, G. Pekhimenko, A. T. R. Rajan, D. Sequeira, A. Sirasao, F. Sun, H. Tang, M. Thomson, F. Wei, E. Wu, L. Xu, K. Yamada, B. Yu, G. Yuan, A. Zhong, P. Zhang, and Y. Zhou,

"MLPerf Inference Benchmark," May 2020, arXiv:1911.02549 [cs, stat]. [Online]. Available: http://arxiv.org/abs/1911.02549

[93] N. Corporation, "NVIDIA TensorRT," Apr. 2024. [Online]. Available: https://docs.nvidia.com/deeplearning/tensorrt/archives/index.html

[94] C. Europea and D. G. de Comunicación, *Agricultura : una asociación entre Europa y los agricultores*. Oficina de Publicaciones, 2017.

[95] M. Ayaz, M. Ammad-Uddin, Z. Sharif, A. Mansour, and E.-H. M. Aggoune, "Internet-of-things (iot)-based smart agriculture: Toward making the fields talk," *IEEE Access*, vol. 7, pp. 129 551–129 583, 2019.

[96] V. Dhanya, A. Subeesh, N. Kushwaha, D. K. Vishwakarma, T. Nagesh Kumar, G. Ritika, and A. Singh, "Deep learning based computer vision approaches for smart agricultural applications," *Artificial Intelligence in Agriculture*, vol. 6, pp. 211–229, 2022. [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/S2589721722000174

[97] P. D. Rosero-Montalvo, C. A. Gordillo-Gordillo, and W. Hernandez, "Smart Farming Robot for Detecting Environmental Conditions in a Greenhouse," *IEEE Access*, vol. 11, pp. 57 843–57 853, 2023. [Online]. Available: https://ieeexplore.ieee.org/document/10146280/

[98] S. Yang, J. Ji, H. Cai, and H. Chen, "Modeling and force analysis of a harvesting robot for button mushrooms," *IEEE Access*, vol. 10, pp. 78 519–78 526, 2022.

[99] A. Z. M. Tahmidul Kabir, A. M. Mizan, N. Debnath, A. J. Ta-sin, N. Zinnurayen, and M. T. Haider, "Iot based low cost smart indoor farming management system using an assistant robot and mobile app," in *2020 10th Electrical Power, Electronics, Communications, Controls and Informatics Seminar (EECCIS)*, Aug 2020, pp. 155–158.