Diseño e implementación de un sistema portable para el registro de consumo de agua potable y energía eléctrica mediante procesamiento de imágenes

Cepeda Carrillo, Alex Luis y Toapanta Quilca, Cristopher Josue

Departamento de Eléctrica, Electrónica y Telecomunicaciones

Carrera de Ingeniería en Electrónica y Telecomunicaciones

Trabajo de titulación, previo a la obtención del título de Ingeniero en Electrónica y Telecomunicaciones

Ing. Román Alcides Lara Cueva, Ph.D

6 de febrero de 2023

Informe de originalidad

NOMBRE DEL CURSO
QuasiGraduados

NOMBRE DEL ALUMNO
ALEX LUIS CEPEDA CARRILLO

NOMBRE DEL ARCHIVO
ALEX LUIS CEPEDA CARRILLO - Tesis Revisión Originalidad

SE HA CREADO EL INFORME
18 ene 2023

Resumen

Fragmentos marcados	5	0,5 %
Fragmentos citados o entrecomillados	2	0,2 %

ROMAN ALCIDES LARA CUEVA
Firmado digitalmente por ROMAN ALCIDES LARA CUEVA
Fecha: 2023.01.18 11:39:08 -05'00'

Coincidencias de la Web

cloud.google.com	2	0,2 %
geoinnova.org	1	0,2 %
compuhoy.com	1	0,1 %
udelar.edu.uy	1	0,1 %
kolmite.com	1	0,1 %
1millionbot.com	1	0,1 %

1 de 7 fragmentos

Fragmento del alumno MARCADO

…en C + +. (Pang et al., 2019). **TensorFlow Lite** es la solución **para** ejecutar **modelos de** TensorFlow **en dispositivos móviles e integrados**. Se optimizó para precisión, baja latencia, tamaño de modelo…

Mejor coincidencia en la Web

TensorFlow Lite ofrece ya algunos modelos que han sido entrenados y optimizados **para** dispositivos móviles: MobileNet: **modelos** de visión capaces de distinguir entre 1000 clases de objetos diferentes,…

Google presenta Tensorflow Lite para facilitar la incorporación del … https://1millionbot.com/google-presenta-tensorflow-lite-para-moviles/

2 de 7 fragmentos

Fragmento del alumno CITADO

…2 matrices cero de los otros canales de color. **Además, los valores RGB se pueden manipular fácilmente con operaciones** de matrices **simples para aumentar o disminuir el brillo de la**

Mejor coincidencia en la Web

Además, los valores RGB se pueden manipular fácilmente con operaciones matriciales **simples para aumentar o disminuir el brillo** o contraste **de la** imagen.

Aprendizaje profundo para la extracción de edificios en ciudades
… https://www.colibri.udelar.edu.uy/jspui/bitstream/20.500.12008/34130/1/GON22.pdf

---

3 de 7 fragmentos

Fragmento del alumno MARCADO

…datos que son útiles en el entrenamiento del modelo. **Las anotaciones se guardan como archivos XML en formato PASCAL VOC, el formato utilizado por ImageNet**

Mejor coincidencia en la Web

Las anotaciones se guardan como archivos XML en formato PASCAL VOC, el formato utilizado por ImageNet. Además, también es compatible con los formatos YOLO y CreateML.

Explora - KolMitE https://kolmite.com/https-github-com-heartexlabs-labelimg/

---

4 de 7 fragmentos

Fragmento del alumno CITADO

Android es un sistema operativo móvil actualmente desarrollado por Google, **basado en** el **kernel de Linux y diseñado principalmente para dispositivos móviles con pantalla táctil, como teléfonos**…

Mejor coincidencia en la Web

Android es un sistema operativo móvil basado en una versión modificada del **kernel de Linux y** otro software de código abierto, **diseñado principalmente para dispositivos móviles con pantalla táctil,**…

¿Por qué Android está basado en Linux? - CompuHoy.com https://www.compuhoy.com/por-que-android-esta-basado-en-linux/

---

5 de 7 fragmentos

Fragmento del alumno MARCADO

…automáticamente cuando aumenta la demanda de datos del usuario. **En la Tabla** 5 **se** muestran **los precios del almacenamiento y de las operaciones correspondientes a cada ubicación de Firestore**.

Mejor coincidencia en la Web

En la siguiente **tabla se** indican **los precios del almacenamiento y de las operaciones** de lectura, escritura y eliminación **correspondientes a cada ubicación de Firestore**: ...

Precios | Firestore | Google Cloud https://cloud.google.com/firestore/pricing?hl=es

---

6 de 7 fragmentos

Fragmento del alumno MARCADO

Nota. **En la tabla se** indica **los precios del almacenamiento y de las operaciones de lectura, escritura y eliminación correspondientes a** Could **Firestore** ubicado en EE.UU.

Mejor coincidencia en la Web

En la siguiente **tabla se** indican **los precios del almacenamiento y de las operaciones de lectura, escritura y eliminación correspondientes a** cada ubicación de **Firestore**: ...

Precios | Firestore | Google Cloud https://cloud.google.com/firestore/pricing?hl=es

---

7 de 7 fragmentos

Fragmento del alumno MARCADO

Si la petición se realiza de forma correcta, el servidor responderá paquetes de archivos entre los que se encuentra código y ficheros que el **navegador organizará para** mostrar **el resultado en pantalla**.

Mejor coincidencia en la Web

Si la petición se realiza de forma correcta, el servidor nos **responderá** enviando **paquetes de archivos entre los que se encuentra código y ficheros que** nuestro **navegador** (cliente) **organizará para**...

HTML, CSS y JavaScript. Lenguajes para el desarrollo de páginas ... https://geoinnova.org/blog-territorio/html-css-y-javascript-lenguajes-para-el-desarrollo-de-paginas-web/

Departamento de Eléctrica, Electrónica y Telecomunicaciones

Carrera de Ingeniería en Electrónica y Telecomunicaciones

Certificación

Certifico que el trabajo de titulación: "**Diseño e implementación de un sistema portable para el registro de consumo de agua potable y energía eléctrica mediante procesamiento de imágenes**" fue realizado por los señores **Cepeda Carrillo, Alex Luis** y **Toapanta Quilca, Cristopher Josue;** el mismo que cumple con los requisitos legales, teóricos, científicos, técnicos y metodológicos establecidos por la Universidad de las Fuerzas Armadas ESPE, además fue revisado y analizado en su totalidad por la herramienta de prevención y/o verificación de similitud de contenidos; razón por la cual me permito acreditar y autorizar para que se lo sustente públicamente.

Sangolquí, 18 de enero de 2023

Firmado electrónicamente por:
ROMAN ALCIDES LARA CUEVA

.....

Ing. Román Lara Cueva, Ph.D

C.C.: 1713988218

Departamento de Eléctrica, Electrónica y Telecomunicaciones

Carrera de Ingeniería en Electrónica y Telecomunicaciones

Responsabilidad de Autoría

Nosotros, **Cepeda Carrillo, Alex Luis** y **Toapanta Quilca, Cristopher Josue**, con cédulas de ciudadanía n° 1725522138 y 1719252775 declaramos que el contenido, ideas y criterios del trabajo de titulación**: Diseño e implementación de un sistema portable para el registro de consumo de agua potable y energía eléctrica mediante procesamiento de imágenes** es de nuestra autoría y responsabilidad, cumpliendo con los requisitos legales, teóricos, científicos, técnicos, y metodológicos establecidos por la Universidad de las Fuerzas Armadas ESPE, respetando los derechos intelectuales de terceros y referenciando las citas bibliográficas.

Sangolquí, 6 de febrero de 2023

Cepeda Carrillo Alex Luis
Firmado digitalmente por Cepeda Carrillo Alex Luis
Fecha: 2023.02.06 19:35:49 -05'00'

.....

Cepeda Carrillo, Alex Luis

C.C.: 1725522138

Toapanta Quilca Cristopher Josue
Firmado digitalmente por Toapanta Quilca Cristopher Josue
Fecha: 2023.02.06 19:37:47 -05'00'

.....

Toapanta Quilca, Cristopher Josue

C.C.: 1719252775

Departamento de Eléctrica, Electrónica y Telecomunicaciones

Carrera de Ingeniería en Electrónica y Telecomunicaciones

Autorización de Publicación

Nosotros **Cepeda Carrillo, Alex Luis** y **Toapanta Quilca, Cristopher Josue**, con cédulas de ciudadanía n° 1725522138 y 1719252775, autorizamos a la Universidad de las Fuerzas Armadas ESPE publicar el trabajo de titulación: **Diseño e implementación de un sistema portable para el registro de consumo de agua potable y energía eléctrica mediante procesamiento de imágenes** en el Repositorio Institucional, cuyo contenido, ideas y criterios son de nuestra responsabilidad.

Sangolquí, 6 de febrero de 2023

Cepeda Carrillo Alex Luis
Firmado digitalmente por Cepeda Carrillo Alex Luis
Fecha: 2023.02.06 19:43:06 -05'00'

.....

Cepeda Carrillo, Alex Luis

C.C.: 1725522138

Toapanta Quilca Cristopher Josue
Firmado digitalmente por Toapanta Quilca Cristopher Josue
Fecha: 2023.02.06 19:43:45 -05'00'

.....

Toapanta Quilca, Cristopher Josue

C.C.: 1719252775

Dedicatoria

Dedico este trabajo de titulación a mi madre Olga, a mi padre Luis y a mi hermana Heidy, por su sacrificio y esfuerzo, por haber fomentado en mí, el deseo de superación y de triunfo en la vida y sobre todo por creer en mi capacidad de superación, lo que ha contribuido a la consecución de este logro.

Alex Luis Cepeda Carrillo

Este trabajo lo quiero dedicar a mis padres Miriam y Fabian por su amor, sacrificio y apoyo sobre toda circunstancia durante todos estos años, gracias por mantenerme en sus oraciones. Y cómo no, a mi hermana Gabriela quien gracias a su apoyo incondicional me ha permitido llegar a cumplir hoy este sueño. A ellos quiero dedicar este esfuerzo que ha significado esta trayectoria universitaria.

Cristopher Josue Toapanta Quilca

Agradecimiento

Mi agradecimiento se dirige a quien ha forjado mi camino y me ha dirigido por el sendero correcto, a Dios, quien me brindó a los mejores padres del mundo, gracias a la Universidad de las Fuerzas Armadas – ESPE por brindarme las herramientas necesarias que han sido fundamentales en mi trabajo.

A los grandes amigos que encontré durante esta gratificante experiencia, a los todos los profesores que compartieron sus conocimientos y experiencias durante cada clase impartida, en especial al Ing. Román Lara por ser el guía de este trabajo de titulación.

Y a Cristopher, mi amigo y compañero de grupo en muchos proyectos, por su esfuerzo y dedicación en cada uno de ellos.

Alex Luis Cepeda Carrillo

Agradezco a Dios por darme sabiduría, fortaleza y salud, por cuidarme de toda adversidad.

A mis padres Miriam y Fabian por su lucha constante diariamente por proveer de lo mejor a sus hijos, son claros ejemplos de esfuerzo y perseverancia. A mi hermana Gabriela por su amor y apoyo, quien dentro de su alcance ha sabido ayudarme y ser mi amiga. A mi tía Georgina quien siempre me mantuvo en sus oraciones para ser una persona ejemplar y un gran profesional. A mi abuelita María pilar de esta familia, con su cariño nos ha sabido guiar por el bueno camino.

A mis amigos con los que compartí experiencias, vivencias universitarias, de ellos me llevo una buena amistad y su compañía por este largo camino.

A Alex por su amistad y apoyo durante la vida universitaria y en la realización de este trabajo. Por su labor y esfuerzo efectuado en este trabajo.

A aquellos docentes de la Universidad de las Fuerzas Armadas – ESPE de quienes guardo gratos recuerdos, porque fueron grandes maestros con vocación por la enseñanza. Finalmente, al Ing. Román Lara por su guía y apoyo durante la elaboración y culminación de este trabajo.

Cristopher Josue Toapanta Quilca

Índice de Contenido

Índice de Tablas

Índice de Figuras

Resumen

Con el objetivo de optimizar procesos y ahorrar recursos es fundamental la implementación de nuevas tecnologías en los sectores estratégicos que mayor prioridad da el país, una de estas es la Inteligencia Artificial (AI, del inglés *Artificial Intelligence*), que permite a una máquina imitar funciones cognitivas humanas. La AI tiene varias especializaciones como la Visión Artificial (AV, del inglés *Artificial Vision*), esta tecnología proporciona a un sistema la capacidad de ver, analizar y actuar. Actualmente, existen varios proyectos que permiten optimizar los procesos de registro de lectura en los medidores de energía eléctrica o agua potable, como la implementación de la telemetría, sin embargo, esta tecnología es dependiente de redes LPWAN y energía eléctrica además que reduce plazas de empleo, es por ello que en el presente proyecto de investigación se desarrolló un sistema portable para el registro de consumo de agua potable y energía eléctrica mediante el procesamiento de imágenes, este sistema está compuesto por tres bloques: pre clasificador, clasificador y detector, considerando el *Transfer Learning* se usó el modelo de MobileNet V2 garantizando la compatibilidad con aplicaciones móviles. Estos bloques fueron programados en Google Colab Pro, una vez entrenados los modelos de ML fueron exportados como archivos de Tensorflow Lite e implementados en una aplicación Android, la cual permite tomar una fotografía y realizar el reconocimiento de un medidor para posteriormente extraer la lectura del consumo y enviarla a Cloud Firestore, una base de datos limitada en la nube, esta información se puede observar en una página web alojada en un servidor local. Las métricas de desempeño son menores al 5% en el caso de las pérdidas y mayor al 97% de exactitud para cada bloque.

Palabras claves: inteligencia artificial, visión artificial, aplicación móvil, página web.

Abstract

In order to optimize processes and save resources, it is essential to implement new technologies in the strategic sectors that the country gives the highest priority, one of these is Artificial Intelligence (AI), which allows a machine to mimic human cognitive functions. AI has several specializations such as Artificial Vision (AV), this technology provides a system the ability to see, analyze and act. Currently, there are several projects that allow to optimize the processes of reading registration in electricity or drinking water meters, such as the implementation of telemetry, however, this technology is dependent on LPWAN networks and electrical power in addition to reducing jobs, this is why in the present research project a portable system was developed for recording the consumption of drinking water and electrical energy through image processing, this system is composed of three blocks: pre-classifier, classifier and detector, Considering Transfer Learning, the MobileNet V2 model was used to ensure compatibility with mobile applications. These blocks were programmed in Google Colab Pro, once trained ML models were exported as Tensorflow Lite files and implemented in an Android application, which allows you to take a photograph and perform the recognition of a meter to then extract the reading of the consumption and send it to Cloud Firestore, a limited database in the cloud, this information can be seen on a web page hosted on a local server. Performance metrics are less than 5% for losses and more than 97% accuracy for each block.

Key words: Artificial intelligence, artificial vision, mobile application, web page.

Capítulo I

Introducción

En los últimos años, los teléfonos inteligentes tuvieron una gran evolución, se han incorporado cámaras de alta gama, conectividad a redes de última generación con autonomía de larga duración y otras tecnologías que resultan de gran apoyo en actividades que realizan las personas diariamente. En este contexto, las empresas se encuentran con la necesidad de adaptarse y aprovechar el desarrollo tecnológico a fin de reducir los costos de operación, además, facilitar la labor de sus trabajadores en sus respectivas actividades laborales.

Una de las tecnologías implementadas en los dispositivos inteligentes es la Visión Artificial (AV, del inglés *Artificial Vision*), la AV es una de las áreas correspondientes al campo de la Inteligencia Artificial (AI, del inglés *Artificial Intelligence*), la cual permite identificar información visual mediante la aplicación de técnicas de procesamiento, análisis y extracción de características de una imagen obtenida con una cámara (Alvarellos et al., 2017). Es así que, se han desarrollado aplicaciones en el área de la seguridad electrónica para la detección de movimiento e identificación de patrones, además, la AV también aparece en la industria para el control de calidad o clasificación de determinados productos, esta tecnología permitió desarrollar aplicaciones en peajes y estacionamientos inteligentes, ya que es capaz de identificar las placas de automóviles. Las aplicaciones son innumerables debido a que el sentido de la vista es uno de los más importantes para el ser humano, ya que, a través de éste se realizan varias tareas simultáneas que requieren precisión y habilidad (Garcia & Caranqui, 2015). Es por esto que durante varios años se han desarrollado y experimentado con diversos algoritmos de AV que puedan adaptarse a aplicaciones móviles, debido a que esta tecnología debe estar al alcance de cualquier persona para el uso en tareas cotidianas.

Es por ello que, el propósito de este trabajo es aprovechar los algoritmos y técnicas de AV, para obtener la información de consumo de energía eléctrica y agua potable, para lo cual se requieren herramientas que permitan el reconocimiento óptico de caracteres (OCR, del

inglés *Optical Character Recognition*), es decir, extraer los dígitos que tiene una imagen de un medidor y transformarlos a un dato procesable y así obtener el consumo de agua potable o energía eléctrica (Navarro, 2013).

Antecedentes

Actualmente, la Empresa Pública Metropolitana de Agua Potable y Saneamiento Quito (EPMAPS-Q) dispone de dos sistemas para realizar la lectura del consumo de agua potable, el sistema tradicional es aplicado en las parroquias rurales y el sistema de facturación inmediata es aplicado en la zona urbana. Para el primer caso los encargados son las Juntas Parroquiales, estas tienen obligaciones como la recolección de lecturas, detección de novedades, recaudación y transferencia a la EPMAPS-Q, mientras que para el segundo caso todo el proceso es automático y garantiza la seguridad de entrega de facturas al cliente, cobranza oportuna, reducción de tiempos de facturación y reducción de errores de lectura de medidores. La diferencia entre ambos métodos radica en el uso de un dispositivo móvil, el cual su única función es la toma de lectura de los medidores de agua potable.

Las empresas prestadoras de servicio de energía eléctrica tienen procedimientos similares a los ya mencionados, métodos convencionales para zonas urbanas y métodos más avanzados para zonas urbanas, esto por el costo que implica la dotación de equipos de recolección de lecturas al personal en zonas rurales, allí se recolectan los datos numéricos de los medidores de agua potable y energía eléctrica de una forma manual y rudimentaria, que es cada vez más difícil por el crecimiento demográfico.

Para suplir esta necesidad se desarrolló un sistema que permite el ingreso manual de las lecturas de consumo de agua potable a una aplicación con lo cual dotaron de una herramienta tecnológica a los recolectores, de esta forma se optimizó todo el proceso de lecturas (Cuenca & Ortega, 2017).

Una investigación más orientada al Internet de las cosas (IoT, del inglés *Internet of Things*) se realizó en (Hernández et al., 2020), ya que diseñó un sistema modular de telemetría

para hogares, con esto consiguió medir distintos parámetros como energía eléctrica, agua potable, presencia de gas propano, iluminación interior, temperatura y la energía generada por un panel fotovoltaico interconectado a la red eléctrica, el sistema permitía monitorizar al usuario los datos de manera remota y en tiempo real a través de una aplicación móvil y con base a su criterio, pudo ejercer acciones preventivas y correctivas para el aprovechamiento consciente de los recursos energéticos.

Con la evolución de la tecnología, los sistemas tienden a ser automáticos, muestra de ello es la aplicación de la telemetría que monitoriza diferentes variables físicas, sin embargo, a pesar de ser sistemas que dan las mejores prestaciones, no son viables, ya que tienen un costo de implementación alto, debido a que se cambia toda la estructura física de medición y se vuelve dependiente de la Internet y energía eléctrica.

Por ello, se empezó a dar soluciones al mismo problema con la utilización de diferentes tecnologías, como la que se presentó en (Alvares et al., 2020) con la implementación de un método basado en AV para automatizar el proceso de lectura de contadores de agua, contadores de gas y contadores de electricidad, con display analógico o digital, a través de una aplicación móvil para plataformas iOS y Android, en donde utilizaron una red neuronal convolucional para adquirir los dígitos que se muestran en la pantalla del medidor y almacenarlos en una base de datos para futuros análisis predictivos.

Se identificaron que los trabajos relacionados al tema propuesto son de interés para las empresas de agua potable y energía eléctrica, además, en el plan nacional de desarrollo del actual gobierno motiva a la generación, búsqueda e investigación que permitan mejoras sociales en términos de conectividad, comunicación y servicio. Desde esta perspectiva, es fundamental la introducción de nuevas tecnologías en los sectores estratégicos que mayor prioridad da el país, como el agua y la electricidad.

Justificación e importancia

El agua y la electricidad son recursos estratégicos del siglo XXI debido a la desmedida ambición que ciertos grupos económicos tienen por aumentar la explotación, control y administración de recursos naturales como el agua, principal actor en la generación de energía eléctrica en el Ecuador, es por ello que los prestadores del servicio deben implementar nuevas tecnologías para optimizar recursos y procesos.

La EPMAPS-Q tiene conciencia de la buena administración del agua potable de tal forma que en su plan general de negocios, expansión e inversión del 2021, menciona la intención institucional de optimizar la lectura de medidores de agua potable, con la ejecución de proyectos de innovación, además, detalla el impacto económico que conlleva la disminución de tiempo en la recolección de lecturas y la importancia de implementar nuevas estrategias tecnológicas en todos sus procesos (EPMAPS, 2021).

La telemetría es un conjunto de técnicas para la medida a distancia de magnitudes físicas (Real Academia Española, 2022), esta tecnología es la solución para empresas como la EPMAPS-Q, sin embargo, se requiere de una red LPWAN (del inglés, *Low Power Wide Area Network*) como Sigfox, que actualmente en el país no se encuentra implementado. Telcel una empresa líder en telecomunicaciones en México, menciona que la telemetría aplicada a la gestión del agua es una herramienta que sirve para mejorar la obtención, distribución y consumo, además, es posible monitorizar los niveles de los depósitos para evitar desperdicios con ello permitir ahorro y sustentabilidad (García, 2021).

En Ecuador se implementó la telemetría para la obtención de lecturas en medidores de energía eléctrica, sin embargo, el diario El Universo mencionó que se reportaron una cantidad masiva de reclamos relacionados con fallas o novedades en los equipos de medición de energía eléctrica en Guayaquil, esto debido a que la empresa a la cual se le entregó la adjudicación del contrato por parte de la Corporación Nacional de Electricidad (CNEL) para servicios de lectura, conexión y reconexión realizó cambios tanto en hardware como en

software de su sistema, añadieron servicios de notificaciones de clientes con impedimentos, entrega de facturas y lecturas en línea (Carrasco, 2021).

Actualmente, tanto en la lectura de medidores de agua potable como en los de energía eléctrica se ven muchas deficiencias como la demora en el proceso de lectura del medidor, errores de digitación y facturación, es por ello que, para facilitar dichos procesos, se propone diseñar e implementar un sistema portable para el registro de consumo de agua potable y energía eléctrica mediante el procesamiento de imágenes. El desarrollo del sistema tiene el objetivo de apoyar el proceso de toma de datos y mitigar los errores humanos, con ello optimizar recursos económicos y obtener mayores beneficios a menor costo y así brindar un servicio de calidad de acuerdo a las necesidades y expectativas del cliente, además, con la implementación del sistema no se eliminarán plazas de empleo, debido a que la telemetría implica la lectura y suspensión del servicio de forma automática, es decir, no se requiere de una persona en campo.

Alcance

En este proyecto de investigación se desarrolló un sistema portable para el registro del consumo de agua potable y energía eléctrica enfocado a facilitar la labor de los recolectores de lecturas de los medidores. Este sistema dispone de una aplicación móvil que integra dos modelos de AV entrenados con un *dataset* compuesto de 16500 imágenes, esto permite realizar correctamente la lectura de los medidores.

El sistema consta de 3 bloques principalmente, el primero es un modelo que permite identificar si en la imagen capturada existe o no un medidor de energía eléctrica o de agua potable, el segundo modelo se utiliza para diferenciar entre un medidor de agua potable y uno de energía eléctrica, este bloque es capaz de distinguir un medidor de otro gracias a las características como su forma, tamaño o estructura, el tercer modelo es un detector que permite identificar los dígitos en los medidores que marcan el consumo del servicio. Ambos modelos están desarrollados y entrenados en lenguaje Python con la utilización de librerías de

AI como TensorFlow y Keras. Para la visualización de los registros de cada lectura se tiene una página web que presenta la lectura realizada por cada mes. A fin de efectuar este proyecto de investigación se proponen los siguientes objetivos.

Objetivos

Objetivo general

- Diseñar e implementar un sistema portable para el registro de consumo de agua potable y energía eléctrica mediante el procesamiento de imágenes.

Objetivos específicos

- Desarrollar el estado del arte sobre Visión Artificial y Procesamiento Digital de Imágenes.
- Entrenar un modelo de reconocimiento de objetos con un dataset de imágenes de medidores de agua potable y de energía eléctrica.
- Aplicar técnicas de preprocesamiento de imágenes al dataset.
- Desarrollar un aplicativo móvil con un entorno intuitivo que permita al usuario el registro del consumo de agua potable.
- Desarrollar una página Web que permita visualizar la información recolectada por la aplicación móvil.
- Evaluar los parámetros de desempeño del sistema.

Trabajos relacionados

El avance tecnológico en el campo de la AV y sus algoritmos han sido perfeccionados durante los últimos años, de tal manera que AV resulta de utilidad en diversas aplicaciones donde es fundamental aplicar adecuadamente el sentido de la vista y automatizar procesos que beneficien a la sociedad.

En (Chang et al., 2015) se describe el desarrollo de un prototipo de calibración de medidores de agua mediante AV. Este trabajo consiste en utilizar técnicas y algoritmos de AV para realizar la lectura de las agujas del medidor analógico de agua y de un flotador de rotámetro, que sirve como referencia. De esta manera se pueden digitalizar los datos para automatizar la calibración del medidor de agua, acogiéndose a los requerimientos de la norma ISO 4064.

También se desarrolló e implementó un sistema para la detección de infracciones y matrículas en motocicletas en Valledupar, Colombia. Este sistema se basa en una aplicación móvil que hace uso de su cámara frontal para dichas detecciones. En su desarrollo se utiliza librerías de AV como son *OpenCV* para el procesamiento de imágenes y localización de la región de la matrícula y *Tesseract* para el reconocimiento óptico de caracteres de la matrícula de la motocicleta. El sistema desarrollado permite identificar si dos personas se movilizan en una motocicleta y si éstas no hacen uso del casco obligatorio de protección, mediante la transformada circular de Hough que permite analizar una imagen y localizar círculos, el algoritmo de AI YOLO, es utilizado para la detección de objetivos por medio de un entrenamiento (Valencia et al., 2020).

Por otro lado, los algoritmos de AV realizan diversos procesamientos de imagen que permiten adaptarla a las necesidades del sistema y de esta forma poder extraer características valiosas y esenciales de un determinado objeto de manera más sencilla. Este proceso se describe en (Alvarellos et al., 2017) donde se desarrolló un sistema para la medición de manómetros analógicos. En este trabajo, el algoritmo de procesamiento se basa en aplicar técnicas de AV para detectar características del instrumento. Inicialmente, se captura la imagen en blanco y negro, posteriormente se elimina el ruido de la imagen mediante un desenfoque Gaussiano, así pues, con la imagen suavizada se procede a detectar los bordes de la imagen del manómetro al hacer uso del algoritmo de Canny. Finalmente, se detectan los círculos y ángulos de las agujas para realizar las respectivas medidas del manómetro.

Capítulo II

Marco teórico

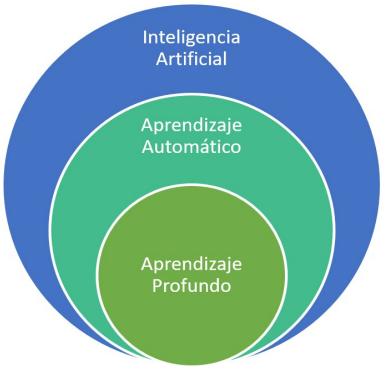
Inteligencia artificial

La AI es la tecnología que implementa algoritmos computacionales en máquinas para llevar a cabo labores que suelen realizar los seres humanos. Esta tecnología permite a las máquinas percibir una gran cantidad de información, aprender de estos y tomar decisiones con un bajo porcentaje de error como resultado de su aprendizaje, el objetivo principal es emular la inteligencia humana (Rouhiainen, 2018).

El campo de aplicación de la AI cada vez es más grande, de tal manera que puede ser implementado en la industria para la clasificación y etiquetado en las fábricas empacadoras o para dar mantenimiento predictivo, en el sector financiero con la finalidad de optimizar estrategias comerciales, en vehículos autónomos para la detección y clasificación de señales, en teléfonos inteligentes o computadoras con detección facial, asistentes de voz, en *chatbots* que permiten recolectar información y establecer patrones de interés para ofrecer respuestas eficientes a personas con respecto a algún problema.

Figura 1

La AI y sus disciplinas



Nota. La figura presenta las diferentes disciplinas que constituyen y dan cualidad a la AI.

La AI se puede implementar en todos los campos de estudio, es por ello que esta tecnología tiene diferentes disciplinas como el Aprendizaje Automático (ML, del inglés *Machine*

Learning) y el Aprendizaje Profundo (DL, del inglés *Deep Learning*), estas permiten solucionar tareas aún más complejas, como la clasificación de imágenes, reconocimiento de voz y procesamiento del lenguaje natural. En la Figura 1 se presenta como se encuentra conformado la AI y sus diferentes disciplinas.

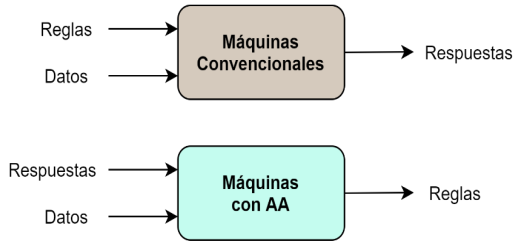
Aprendizaje automático

ML es una de las ramas más importantes de la AI, se basa en la capacidad que puede tener una máquina para aprender de una situación en particular de forma autónoma, mediante la aplicación de diversos algoritmos que permiten el aprendizaje de patrones que se encuentran en los datos de entrada. Comúnmente, las máquinas son programadas para realizar determinadas actividades o acciones, sin embargo, gracias al ML las máquinas pueden adquirir experiencia y nuevos conocimientos durante su utilización (Tatic, 2021).

La Figura 2 ilustra las diferencias entre las máquinas convencionales y las máquinas basadas en ML, respecto a las entradas y salidas. Un ejemplo claro de la aplicación del ML se encuentra en Meta con sus diferentes redes sociales, el cual genera resultados acordes a búsquedas previamente realizadas por un usuario.

Figura 2

Diferencia entre máquinas convencionales y basadas en el ML



Nota. La figura muestra gráficamente las principales características que diferencian a una máquina convencional como un PC de una máquina con ML.

Las máquinas convencionales a diferencia de las que disponen de ML están destinadas a obedecer reglas de manera lineal y constante acorde a determinados datos de entrada y así

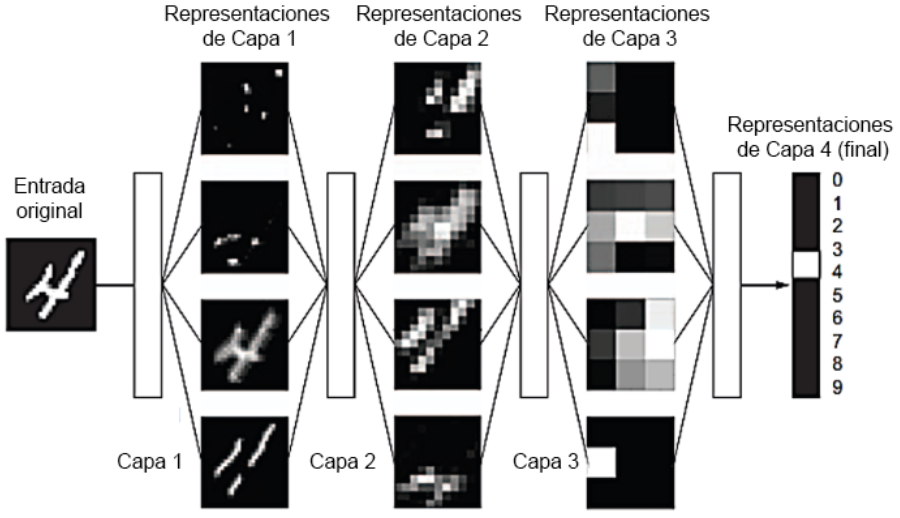
generar respuestas previamente definidas, por otro lado, las máquinas con ML son entrenadas con las respuestas deseadas y una gran cantidad de datos que las estimulan a fin generar reglas a la salida que permitan identificar patrones complejos en dichos datos (D'Arc, 2020).

Aprendizaje profundo

Dentro del ML se encuentra como subcampo el DL que es empleado en sistemas modernos y eficientes para la resolución de problemas complejos, como consecuencia se realiza el análisis de un gran volumen de datos, además de una robusta capacidad de procesamiento. El DL se basa en el uso de redes neuronales artificiales (ANN, del inglés *Artificial Neural Networks*), estas son organizadas en varias capas y simulan el comportamiento de neuronas del cerebro para identificar relaciones y patrones complejos que se encuentran en los datos y aprenderlos sucesivamente por cada una de las capas que componen la ANN, la organización de estas capas se presenta de manera gráfica en la Figura 3.

Figura 3

Red neuronal profunda para la clasificación de dígitos



Nota. La figura presenta cómo está constituida una red neuronal profunda para la clasificación de dígitos. Adaptado de (Planeta ChatBot, 2019).

El enfoque de una ANN es descomponer los datos de entrada de tal forma que el sistema sea eficiente y preciso, normalmente esto involucra que la ANN contenga decenas o

cientos de capas sucesivas con el objetivo de precisar y aprender dichos patrones complejos que pudiera haber en los datos (Planeta ChatBot, 2019).

La misión del DL es asistir de cierta manera al ML a través de las ANN para que mejore el aprendizaje por cada una de sus capas, las primeras capas aprenden relaciones o patrones simples, esta información fluye a la siguiente capa con la información aprendida y así con las demás capas.

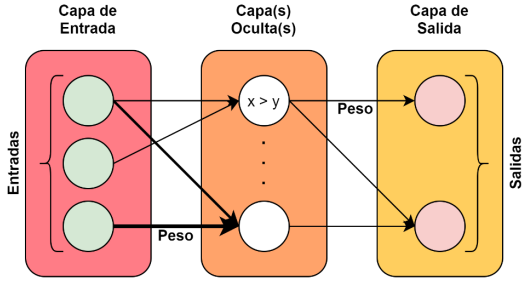
Las aplicaciones donde se implementa el DL son: el reconocimiento de voz, procesamiento del lenguaje, la AV, asistencia vehicular para el conductor, entre otras. Por ejemplo, como resultado del DL, la red social Facebook puede realizar alrededor de 4.500 millones de traducciones diariamente (Rouhiainen, 2018).

Redes neuronales artificiales

Las ANN están compuestas por varias capas y en estas se encuentran características representativas de los datos que deben ser aprendidas por el sistema. Una ANN bien entrenada puede llegar a reconocer texto, voz, caras de personas e incluso enfermedades como el Covid-19. La estructura básica de una ANN se constituye por la capa de entrada (CE), las capas ocultas (CO) y la capa de salida (CS), estas se muestran en la Figura 4. Las capas y neuronas se interconectan con una relación denominada peso, la cual indica la importancia que tiene una neurona para estimular o activar a la siguiente neurona a la cual está conectada.

Figura 4

Estructura de una ANN

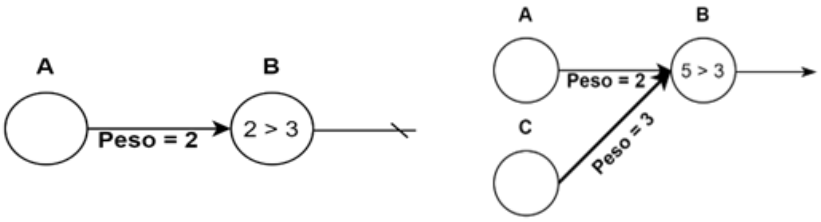


Nota. La figura presenta la estructura básica que tiene una ANN.

Por otro lado, se establece el umbral a las neuronas que se encuentran en las capas ocultas, esto es un valor determinado el cual define el peso mínimo de la relación entre las neuronas para que la última se active, es decir, si el peso de la relación que tiene una neurona A con una neurona B es de 2, pero el umbral de activación de la neurona B es de 3, dicha relación no permitirá excitar a la neurona B. Sin embargo, si la neurona B tiene una relación más con una neurona C con un peso de 3, entonces, el peso de la relación de la neurona A más el de la neurona C superarían el umbral de excitación de la neurona B. Esto proceso de relación entre neuronas se muestra en la Figura 5.

Figura 5

Relación entre neuronas en capas de una ANN



Nota. La figura presenta cómo es la relación entre diferentes neuronas de una ANN.

Ahora bien, esto se reproduce por decenas o cientos de neuronas que tiene una capa oculta de una ANN. Esta estructura permite tomar decisiones muy complejas y como se ha

especificado anteriormente, las primeras neuronas aprenderán cosas más sencillas y mientras se avanza por las capas se suscitarán relaciones y patrones más complejos que podrán ser solventados al ajustar el valor de los pesos y umbrales.

Redes neuronales convolucionales

Una red neuronal convolucional (CNN, del inglés *Convolutional Neural Networks*) es uno de los varios tipos de ANN y una de las más importantes por su desarrollo y aplicación para resolver problemas que implican un alto procesamiento de imágenes. El área donde más es promovida una CNN es la AV debido a su habilidad de procesar información o datos en matrices bidimensionales, estructura similar a la de los píxeles de una imagen.

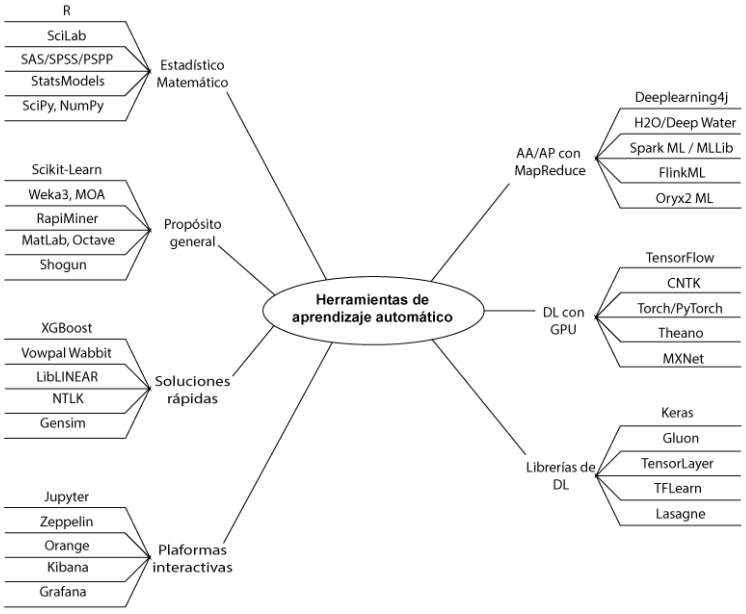
Una CNN está constituida por tres etapas: entrada, extracción de características y clasificación. La etapa de extracción de características contiene varias neuronas convolucionales que permiten extraer características representativas de una imagen como son líneas, curvas, ejes, trazos, formas complejas o siluetas particulares. La extracción de características que realiza la CNN se basa en el uso de operaciones matemáticas denominadas convoluciones, la cual permite aplicar varios filtros a una imagen o más específicamente a su matriz de píxeles, de tal manera que resulte más sencilla dicha extracción. Posteriormente, se detalla el procesamiento de imágenes con CNN.

Librerías de aprendizaje automático

Como se describió en secciones anteriores, existen una amplia gama de algoritmos de ML, de igual manera, las herramientas de software para DL que utilizan técnicas de ML son extensamente variadas ya que están diseñadas para diversos fines como: plataformas de análisis, sistemas predictivos, procesadores de imágenes, sonido o lenguaje. En la Figura 6 se muestran varias librerías utilizadas en aplicaciones de ML.

Figura 6

Resumen de las librerías de ML.



Nota. Se muestra una amplia gama de herramientas utilizadas para desarrollar aplicaciones con ML, adaptado de (Giang & Dlugolinsky, 2019).

TensorFlow

TensorFlow es una biblioteca de software libre y escalable para cálculos numéricos mediante gráficos de flujo de datos, desarrollada por el equipo de Google Brain para el entrenamiento y la implementación de modelos de DL (Yuan et al., 2017). Su arquitectura flexible permite que las operaciones se implementan mediante núcleos que se pueden ejecutar en distintos tipos de dispositivos como CPU, GPU o dispositivos móviles (Deng, 2019).

Esta librería es ampliamente utilizada para aplicaciones de ML, permite a los usuarios programar y entrenar eficientemente redes neuronales y otros modelos de ML implementarlos en producción. Los algoritmos centrales de TensorFlow están escritos en C++. (Pang et al., 2019). TensorFlow Lite es la solución para ejecutar modelos de TensorFlow en dispositivos móviles e integrados. Se optimizó para precisión, baja latencia, tamaño de modelo pequeño y

portabilidad para Android y otros dispositivos de IoT. Las herramientas de TensorFlow están disponibles en su página web oficial (Deng, 2019).

Keras

Keras es una interfaz de programación de aplicaciones (API, del inglés *Application Programming Interface*) de redes neuronales de alto nivel, escrita originalmente en Python y capaz de ejecutarse sobre Tensorflow, Theano y en el kit de herramientas cognitivas de Microsoft (CNTK, del inglés *Microsoft Cognitive Toolkit*).

Keras también se define como una biblioteca de redes neuronales altamente modular con funciones de activación, ordenamiento de capas, etapas de preprocesamiento, funciones objetivas y selección de métodos de optimización (Gevorkyan et al., 2019).

Procesamiento de imágenes

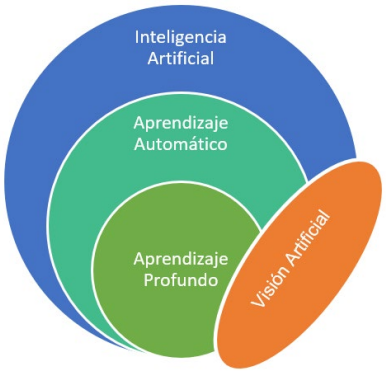
El procesamiento de imágenes es una de las principales áreas de aplicación del ML, gran parte de las aplicaciones desarrolladas tienen como objetivo reconocer personas o texto en una imagen, la base de estos desarrollos es la AV, es así que proyectos como Google Lens, permite identificar objetos en las fotos y muestra información detallada sobre ellas o Cooper I de Dahua, este tiene la funcionalidad de realizar un reconocimiento facial, es decir, se enfoca en la clasificación y reconocimiento de objetos o texto, más adelante se dedica una sección completa a este tema debido a que son pilares fundamentales dentro de este trabajo de titulación.

Visión artificial

La AV es una de las áreas correspondientes al campo de la AI que permite identificar información visual del entorno mediante la aplicación de métodos y técnicas de procesamiento, análisis, extracción de características y reconocimiento de patrones de una imagen obtenida con una cámara (Alvarellos et al., 2017). La AV resulta de gran utilidad en diversas aplicaciones prácticas, como un clasificador de imágenes, autonomía en vehículos inteligentes para la detección de objetos, etc.

Figura 7

La inteligencia artificial, sus disciplinas y su relación con AV



Nota. La figura presenta como la AV está basada en la inteligencia artificial y sus disciplinas.

Así mismo, la AV está basada en cada una de las disciplinas expuestas anteriormente: AI, ML, y DL, así pues, la AI se encuentra constituida con el componente de AV como se muestra en la Figura 7. En primer lugar, porque su fin es ofrecer la capacidad de analizar a las máquinas y así realizar labores que son llevadas a cabo por personas. Segundo, su entrenamiento se enfoca en las técnicas del aprendizaje supervisado. Finalmente, la AV hace uso de las CNN, ya que este tipo de red permite analizar, procesar y extraer características representativas de un determinado objeto presente una imagen.

Técnica transferencia de aprendizaje

En el apartado anterior se presentaron las capas que constituyen la arquitectura de una CNN, estas capas pueden estar distribuidas de distintas maneras, así como también pueden variar la cantidad de éstas, a fin de formar una arquitectura eficiente para la clasificación. Actualmente, la configuración de capas y neuronas son temas de investigación perteneciente al DL donde se desarrollan arquitecturas de CNN robustas que son entrenadas con miles de datos para ser capaces de predecir hasta mil objetos diferentes (Praveen et al., 2021).

La técnica de Transferencia de Aprendizaje (en inglés*, Transfer Learning*) radica en reutilizar arquitecturas de CNN robustas o modelos de clasificación, que ya han sido desarrollados, entrenados y validados, para crear sistemas capaces de solventar nuevos

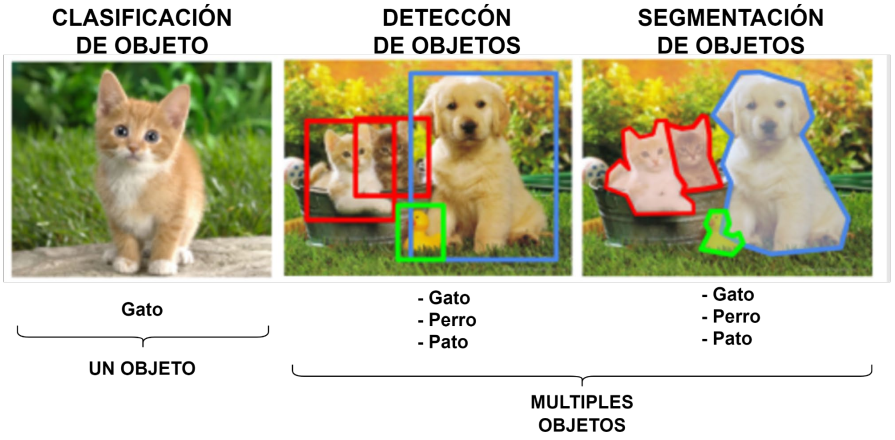
problemas. Con esta técnica lo que se pretende es modificar las últimas capas de clasificación adaptándolas a las necesidades de una aplicación en particular, que suele ser de clasificación de imágenes y así beneficiándose de la configuración de sus capas y los pesos de sus neuronas, también de su entrenamiento y rendimiento (Planche & Andres, 2019).

Aplicaciones de AV

La AV se despliega por tres principales campos de aplicación que tiene en común el tratamiento y análisis de imágenes, pero con diferentes técnicas, procedimientos y fines, la diferencia de estos campos se muestra la Figura 8. Estos campos son la clasificación, detección y segmentación de objetos que se encuentran en una imagen o incluso video, el desarrollo e implementación de estas aplicaciones resulta de gran beneficio, por ejemplo, en el área de la medicina para la detección de células, órganos, tumores e inclusive ciertas enfermedades (Lee & Street, 2003).

Figura 8

Campos de aplicación de AV



Nota. Campos de aplicación de la AV se basan en procesar una imagen, pero con diferentes resultados a su salida. Adaptado de (Programador Clic, n.d.)

Clasificación de objetos

Esta aplicación permite identificar si un determinado objeto está o no presente en una imagen. Un modelo de clasificación puede estar entrenado para identificar uno o varios objetos según la arquitectura de la CNN, este modelo toma como referencia de entrada una imagen, dando como resultado una predicción de un objeto, además de su respectiva probabilidad, pérdida y precisión (Sharma, 2021).

Detección de objetos

La clasificación resulta bastante útil al momento de identificar un objeto en una imagen, pero la tarea se complica cuando existen varios objetos diferentes en una sola imagen o video. Para solventar esto se aplica la detección de objetos que logra localizarlos y enmarcarlos con alguna figura geométrica distintiva como se muestra en la Figura 8, a partir de esta técnica se predice la ubicación de cada objeto en la imagen o video (Sharma, 2021).

Segmentación de objetos

Este campo de aplicación resulta una mejora o extensión de la detección de objetos ubicándolos en la imagen, pero con la diferencia de que, en este caso en lugar de delimitarlo con alguna figura, se aplica una máscara de píxeles con la forma de cada objeto o se dibuja su silueta, en la Figura 8 se presenta este proceso. De esta manera, se logra únicamente delimitar la región de interés del objeto para su posterior procesamiento y análisis, por lo cual se desecha toda la información innecesaria y se obtiene la información útil (Lee & Street, 2003).

Métricas de evaluación

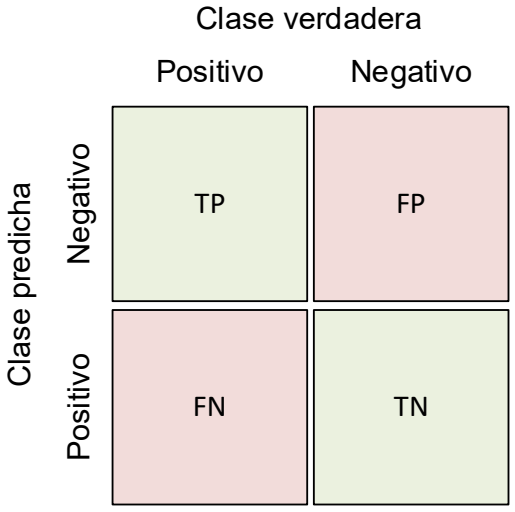
Después de aplicar los algoritmos de ML es necesario evaluar el modelo entrenado, es por ello que se requiere de métricas de evaluación, las más utilizadas para ML y AV son: matriz de confusión, exactitud, sensibilidad, especificidad y la tasa de error balanceada (BER, del inglés *Balance Error Rate*).

Matriz de confusión

Es una de las métricas más intuitivas y descriptivas, esta matriz compara la predicción del modelo con la realidad, se utiliza principalmente en algoritmos que tiene tienen como salida dos o más clases. En la Figura 9 se muestra como está compuesta una matriz de confusión con dos clases (Vakili et al., 2020).

Figura 9

Modelo básico de una matriz de confusión.



Nota. Verdadero positivo (TP, del inglés *True Positive*), falso positivo (FP, del inglés *False Positive*), falso negativo (FN, del inglés *False Negative*), verdadero negativo (TN, del inglés, *True Negative*).

Se puede extraer varias métricas a partir de la matriz de confusión, como:

Precisión

Es una métrica que menciona cuántos de los casos predichos correctamente resultaron ser positivos. La precisión es una métrica útil en caso de que los falsos positivos sean más preocupantes que los falsos negativos.

$$Precisión = \frac{TP}{TP + TN}$$

Exactitud

Se define como la relación entre los elementos de datos clasificados con precisión y el número total de observaciones, es decir, determina cuántas predicciones correctas realizó el modelo para el conjunto total de datos de prueba.

$$Exactitud = \frac{TP + TN}{TP + TN + FP + FN}$$

Sensibilidad

Muestra la cantidad de los casos positivos reales se pueden predecir correctamente con el modelo entrenado.

$$Sensibilidad = \frac{TP}{TP + FN}$$

Especificidad

Muestra la cantidad de los casos negativos reales que se pueden predecir correctamente con el modelo entrenado.

$$Especificidad = \frac{TN}{TN + FP}$$

BER

Es el promedio de error en cada clase positiva.

$$BER = 1 - \left(\frac{Sensibilidad + Especificidad}{2}\right)$$

Capítulo III

Materiales y métodos

Análisis del conjunto de datos para el clasificador

El conjunto de datos o también conocido como *dataset* es uno de los insumos más importantes para el correcto desarrollo e implementación de un clasificador de imágenes, pues su importancia radica en que dicho *dataset* contiene imágenes de diferentes medidores de agua potable y de energía eléctrica, por lo cual, está constituido por dos clases que son objetos de estudio de este proyecto.

El entrenamiento del modelo de una CNN para la clasificación de imágenes tiene como entrada una gran cantidad de datos, los suficientes para el adecuado aprendizaje de las diferentes clases, pues cuantas más imágenes compongan el *dataset* más eficientes serán las predicciones del clasificador, pues en caso contrario el modelo caerá en un estado de sobreajuste lo que indica que el modelo falla por falta de datos y como consecuencia realizará más predicciones erradas. Una regla general que debe cumplir el *dataset* es definir al menos 1000 imágenes por cada clase que lo componga (Mitsa, 2019) y en el caso de contar con poca cantidad de imágenes se puede optar por la técnica de *Data Augmentation*, la cual se explicará más adelante.

Obtención del conjunto de datos

Una de las primeras alternativas para obtener un conjunto de imágenes es a través de la Internet, ya que en los diferentes buscadores que existen en la actualidad se puede extraer gran cantidad de imágenes de manera sencilla, sin embargo, mediante este método no se obtiene variedad de datos. Por ello, se opta por recurrir a trabajos relacionados o entornos de la Internet, donde se comparte de manera libre el *dataset* de diferentes clases. Además, a fin de que las predicciones del clasificador se apeguen más a la realidad de la región donde se desarrolla este trabajo, se acude a empresas administradoras del consumo de agua potable y energía eléctrica para la adquisición de imágenes de medidores que son de su propiedad. A

continuación, se presentan las diferentes fuentes de datos utilizadas para la obtención de imágenes de medidores de agua potable y energía eléctrica.

- **Kaggle:** este *dataset* está formado por 1244 imágenes de diferentes medidores de agua con dimensiones superiores a 1000x1000 pixeles y en formato .jpg, en la Figura 10 se presenta una muestra. Este conjunto de imágenes se caracteriza porque ha sido utilizado para un fin similar al de este trabajo, detección de la región de los dígitos que marcar el consumo de agua.

Figura 10

Muestra de set de datos de Kaggle

Nota. Este *dataset* contiene más de 1200 imágenes de medidores de agua en diferentes posiciones y modelos, por su gran variedad de medidores y nitidez en las imágenes el set de imágenes resulta de ayuda para el entrenamiento. Tomado de (Roman, 2020).

- **UFPR-AMR:** Este *dataset* contiene en total 2000 imágenes de medidores de energía eléctrica de la región de Brasil con dimensiones superiores a 3024x4032 pixeles y en formato .jpg, en la Figura 11 se presenta una muestra, fue utilizado en el trabajo "*Convolutional Neural Networks for Automatic Meter Reading"*, con el mismo propósito de realizar la lectura de los medidores de energía eléctrica.

Figura 11

Muestra de set de datos de UFPR-AMR.

Nota. Set de datos con gran variedad de medidores de electricidad de la región de Brasil. Tomado de (Laroca et al., 2019).

- **EMELNORTE:** la Empresa Eléctrica Regional Norte S.A. con sede en Ibarra, Ecuador, proporcionó una *dataset* de medidores de energía eléctrica con 1553 imágenes con dimensiones variadas desde 240x240 pixeles tomadas por los trabajadores que realizan los recorridos para la toma de lectura del consumo de energía eléctrica, en la Figura 12 se presenta una muestra.

Figura 12

Muestra de set de datos proporcionado por EMELNORTE.

- **De propia autoría:** como parte del desarrollo de este trabajo se realizó la adquisición de 900 imágenes con dimensiones de 2992x4000 pixeles y en formato .jpg de medidores de agua potable y energía eléctrica de la región de Pichincha, Ecuador, en la Figura 13 se presenta una muestra.

Figura 13

Muestra de set de datos de propia autoría.

Data augmentation

Las CNN se han desempeñado notablemente bien en muchas tareas de AV, sin embargo, estas redes dependen en gran medida de la cantidad de datos de entrenamiento utilizados para evitar el sobreajuste, este fenómeno se produce cuando una red aprende una función con una varianza muy alta, como para modelar perfectamente los datos de entrenamiento. Por ello, se requiere incrementar el *dataset*, cuando la cantidad inicial de datos no es suficiente para garantizar buenos resultados (Shorten & Khoshgoftar, 2019).

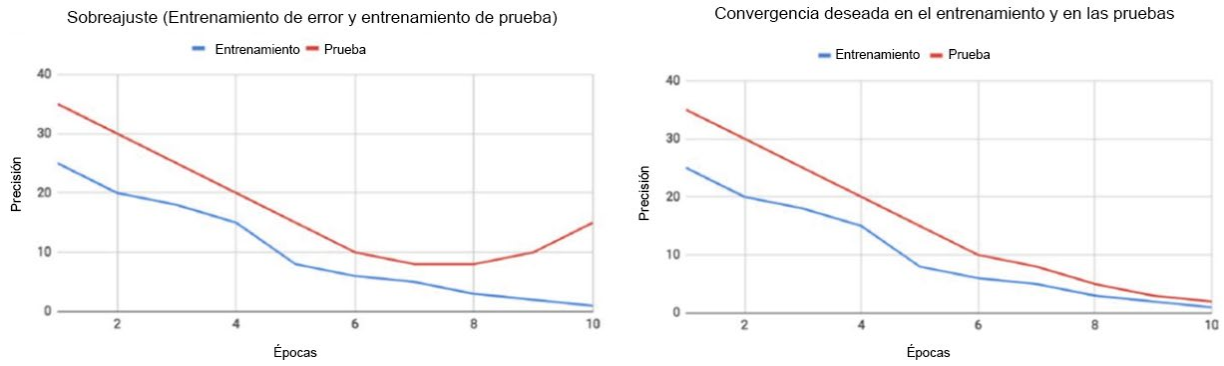
La técnica *Data Augmentation* es usada en modelos de ML que permite crear nuevos datos basados en los existentes sin necesidad de adquirirlos nuevamente.

La importancia de mitigar este fenómeno radica en la capacidad de generalización, esto se refiere a la diferencia de rendimiento de un modelo cuando se evalúa en datos de entrenamiento frente a datos de prueba (Wong et al., 2016).

Los modelos con poca capacidad de generalización han sobre ajustado los datos de entrenamiento. Una forma de descubrir el sobreajuste es trazar la precisión del entrenamiento y la validación en cada época durante el entrenamiento. La Figura 14 muestra el sobreajuste y se verifica que el aumento de épocas de entrenamiento provoca que el modelo se ajuste en exceso al *dataset* y tenga un rendimiento deficiente en el conjunto de prueba.

Figura 14

Comparación de la precisión entre datos de entrenamiento y datos de prueba para verificar el sobreajuste.



Nota. El gráfico de la izquierda muestra un punto de inflexión donde el error de validación comienza a aumentar a medida que la tasa de entrenamiento disminuye. En contraste, la gráfica de la derecha muestra un modelo con la relación deseada entre el entrenamiento y el error de prueba. Tomado de (Shorten & Khoshgoftar, 2019).

Para construir modelos eficientes de DL, el error de validación debe disminuir con el error de entrenamiento, el aumento de datos o *Data Augmentation* es un método muy poderoso para lograr esto. Los datos aumentados representan un conjunto más completo de posibles datos, esto permite reducir la distancia entre el conjunto de entrenamiento y validación, así como cualquier conjunto de prueba (Shorten & Khoshgoftar, 2019).

Técnicas de aumento de datos

A continuación, se presenta las técnicas de pre procesamiento de imágenes utilizadas para el aumento de datos utilizadas en el presente trabajo.

Inversión

La inversión del eje horizontal es mucho más común que la inversión del eje vertical. Este aumento es uno de los más fáciles de implementar y ha demostrado ser útil en conjuntos de datos como ImageNet. En conjuntos de datos que implican el reconocimiento de texto, esta no es una transformación que conserva la etiqueta.

Espacio de color

Los datos de imágenes digitales generalmente se codifican como un tensor de la dimensión (alto × ancho × canales de color). Realizar aumentos en el espacio de los canales de color es otra estrategia muy práctica de implementar. Los aumentos de color muy simples incluyen aislar un solo canal de color como rojo, verde o azul.

Una imagen se puede convertir rápidamente en su representación en un canal de color al aislar esa matriz y agregar 2 matrices cero de los otros canales de color. Además, los valores RGB se pueden manipular fácilmente con operaciones de matrices simples para aumentar o disminuir el brillo de la imagen (Shorten & Khoshgoftar, 2019).

Recorte

El recorte de imágenes se puede utilizar como un paso de procesamiento práctico para datos de imágenes con dimensiones mixtas de alto y ancho al recortar un parche central de cada imagen. Además, el recorte aleatorio también se puede utilizar para proporcionar un efecto muy similar a las traducciones.

El contraste entre el recorte aleatorio y las traducciones es que el recorte reduce el tamaño de la entrada, como de 256x256 pixeles a 224x224 pixeles, mientras que las traducciones conservan las dimensiones espaciales de la imagen (Wong et al., 2016).

Rotación

Los aumentos de rotación se realizan al girar la imagen hacia la derecha o hacia la izquierda en un eje entre 1° y 359°. La seguridad de los aumentos de rotación está fuertemente determinada por el parámetro del grado de rotación. Las rotaciones leves, como entre 1° y 20° o −1° a −20°, podrían ser útiles en tareas de reconocimiento de dígitos, pero como la rotación aumenta el grado, la etiqueta de los datos ya no se conserva después de la transformación (Shorten & Khoshgoftar, 2019).

Traslación

Desplazar las imágenes hacia la izquierda, hacia la derecha, hacia arriba o hacia abajo puede ser una transformación muy útil para evitar sesgo posicional en los datos. Por ejemplo, si todas las imágenes en un conjunto de datos están centradas, lo cual es común en los conjuntos de datos de reconocimiento facial, esto requiere que el modelo también se pruebe en imágenes perfectamente centradas. A medida que la imagen original se traslada en una dirección, el espacio restante se puede llenar con un valor constante, entre 0 y 255, o se puede llenar con ruido aleatorio o gaussiano.

Inyección de ruido

La inyección de ruido consiste en inyectar una matriz de valores aleatorios normalmente extraídos de una distribución gaussiana. Agregar ruido a las imágenes puede ayudar a las CNN a aprender funciones más sólidas (Wong et al., 2016).

Comparativa de modelos de redes neuronales convolucionales

En la actualidad, existen diversos modelos de CNN que se adaptan a las necesidades y requerimientos de determinadas aplicaciones de AV, estos modelos fácilmente pueden estar constituidos por miles de neuronas y parámetros entrenables, lo que les hacen mucho más robustos y efectivos en sus predicciones.

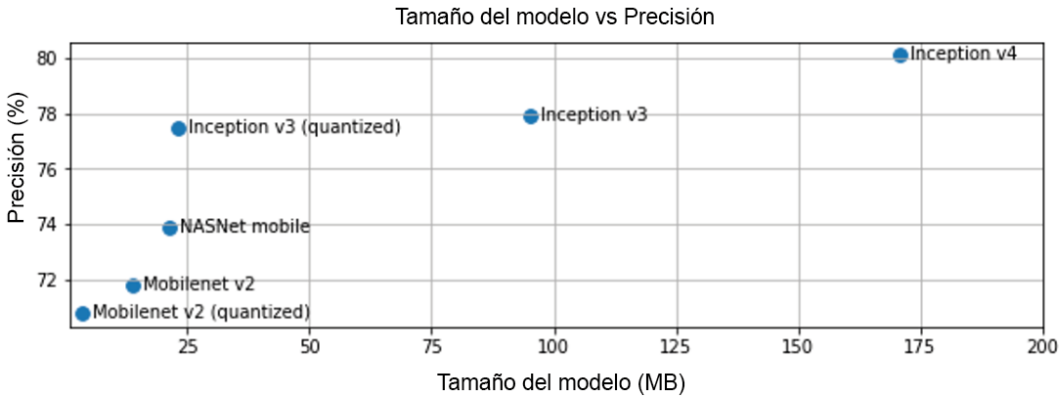
El sitio web oficial de TensorFlow presenta un apartado sobre las mejores prácticas de desempeño al momento de la elección de un modelo de clasificación, debido a que existen

varios modelos robustos con infinidad de neuronas que pueden ofrecer resultados notables y resulta sencillo decidirse implementar estos. Sin embargo, se debe considerar que cuanto más robusto sea el modelo se requiere de una mayor capacidad de procesamiento en el equipo donde se realice el entrenamiento y el dispositivo donde se vaya a implementar. Dado que el fin de este trabajo es incorporar un modelo de clasificación en una aplicación Android donde el dispositivo móvil dispone con recursos limitados, es necesario escoger un modelo que no requiera emplear grandes cantidades de tamaño de memoria y ni un uso excesivo del procesador del teléfono inteligente, pero con un porcentaje de predicción aceptable.

A continuación, en la Figura 15 se presentan diferentes modelos con el tamaño que requieren de memoria y el porcentaje de exactitud (del inglés, *accuracy*), existen 4 modelos que no pesan más de 25 MB de uso de memoria, sin embargo, los que menor memoria requieren se ven afectados por la precisión. Finalmente, el que destaca con una buena precisión y un tamaño alrededor de los 25 MB es el modelo Inception v3.

Figura 15

Comparativa Precisión vs Tamaño en modelos de DL.



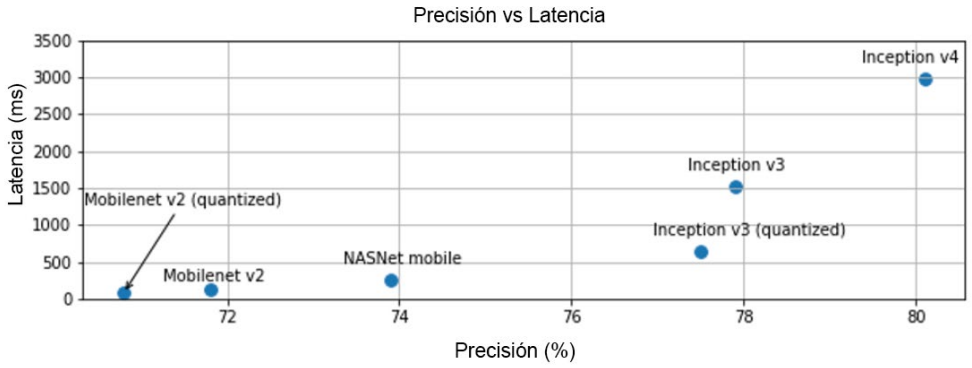
Nota. La figura presenta la comparación de la exactitud y el tamaño en MB de varios modelos de DL orientados para aplicaciones móviles. Tomado de (TensorFlow, 2021).

La Figura 16 presenta un parámetro importante en el desempeño del modelo, este parámetro es la latencia al momento de realizar una predicción, se considera que cuanto

menos tarde el modelo en arrojar el resultado de predicción será más eficaz, sin embargo, un valor bajo de latencia no le permite al modelo ser preciso en sus predicciones, por lo que su porcentaje de exactitud se ve disminuido.

Figura 16

Comparativa Latencia vs Precisión en modelos de DL.



Nota. La figura presenta la comparación de la latencia en milisegundos y la exactitud de varios modelos de DL orientados para aplicaciones móviles. Tomado de (TensorFlow, 2021).

Adicionalmente, Keras es una biblioteca para el uso de redes neuronales junto con TensorFlow, en su página web oficial presenta en un apartado de modelos de DL ya entrenados y disponibles para ser utilizados o modificados, además resultan útiles para el método de *Transfer Learning*. La Tabla 1 presenta modelos de CNN que contiene la biblioteca de Keras, en esta tabla se destacan dos parámetros en especial *Top*-1 *Accuracy* y *Top*-5 *Accuracy* (Ahn, 2021). El primero indica la exactitud convencional del modelo al predecir una clase, mientras que el segundo parámetro indica que las 5 mejores predicciones efectuadas coindicen con el resultado esperado. Con lo anteriormente definido, es que se debe escoger un modelo que tenga un valor alto de Top-1 *Accuracy*, pues se obtienen resultados más cercanos a la realidad. El modelo que destaca sobre los demás es el EfficientNetB1, dado que presenta una buena precisión sin un elevado uso de memoria ni demasiados parámetros que entrenar en comparación con los modelos de *Inception*, puesto que, si el modelo posee con una gran cantidad de parámetros, el entrenamiento de éste requiere de mucho más tiempo.

Tabla 1

Características de modelos de clasificación de imágenes.

Modelo	Tamaño (MB)	Top-1 *Accuracy*	Top-5 *Accuracy*	Parámetros
InceptionV3	92	77.9%	93.7%	23.9M
InceptionResNetV2	215	80.3%	95.3%	55.9M
MobileNet	16	70.4%	89.5%	4.3M
MobileNetV2	14	71.3%	90.1%	3.5M
DenseNet121	33	75.0%	92.3%	8.1M
DenseNet169	57	76.2%	93.2%	14.3M
DenseNet201	80	77.3%	93.6%	20.2M
NASNetMobile	23	74.4%	91.9%	5.3M
EfficientNetB0	29	77.1%	93.3%	5.3M
EfficientNetB1	31	79.1%	94.4%	7.9M

Nota. Los diferentes modelos presentan importantes características para su correcta elección acorde los requerimientos de la aplicación. Tomado de (Keras, n.d.).

Sin embargo, la cualidad que se considera sobre todos los modelos presentados, es el desempeño en aplicaciones móviles debido a que un dispositivo celular dispone con recursos limitados y no puede ejecutar todos los modelos de manera eficiente. Según (Howard et al., 2017; Praveen et al., 2021; Sandler et al., 2018) determinan que los modelos de *MobileNet* presentan una buena precisión y baja latencia en sus predicciones, además de ocupar un tamaño reducido de memoria, estas características hacen a estos modelos ideales para desempeñarse eficientemente en aplicaciones móviles de AV.

La elección de este modelo se lo realiza en base a las siguientes características: el tamaño en memoria que requiere, su Top-1 *Accuracy* y sus parámetros que constituyen la CNN del modelo. La familia de *MobileNet* ofrece varios modelos con buenas cualidades en aplicaciones móviles sobre otros modelos.

Modelo MobileNetV2

Para el entrenamiento del modelo de clasificación se utiliza la técnica de *Transfer Learning*, lo que significa reusar un modelo pre entrenado que ya ha sido desarrollado. La elección de este modelo se realizó en base a las siguientes características: el tamaño en memoria que requiere, su Top-1 *Accuracy* y sus parámetros que constituyen la CNN del modelo. La familia de *MobileNet* ofrece varios modelos con buenas cualidades en aplicaciones móviles sobre otros modelos.

El modelo utilizado para el desarrollo del clasificador de imágenes del presente trabajo es *MobilNetV2*, el cual presenta la arquitectura de su CNN en la Tabla 2. La modificación de este modelo se lo realiza en la etapa final *Softmax*, la que se encarga de clasificar, originalmente el modelo estuvo entrenado para realizar hasta predicciones de 1000 clases diferentes, para el presente trabajo se adaptó para clasificar dos objetos: medidor de agua y medidor de energía eléctrica. Esta adaptación consta en cambiar el tamaño de entrada de esta etapa a $1 \times 1 \times 2$, este formato define que se obtendrán 2 tipos de predicciones para los diferentes medidores.

Herramientas y entorno de entrenamiento del clasificador

El entrenamiento del modelo de clasificación bajo la técnica de *Transfer Learning* debe desarrollarse sobre *hardware* y *software* sofisticados, además de un lenguaje de programación robusto y librerías que permitan una ejecución correcta e ininterrumpida del entrenamiento con el fin de garantizar resultados convenientes. A continuación, se especifican las principales herramientas utilizadas y el entorno general de desarrollo del clasificador.

Tabla 2

Arquitectura modelo MobileNet.

Etapa	Tamaño	Tamaño de entrada
Conv / s2	$3 \times 3 \times 3 \times 32$	$224 \times 224 \times 3$
Conv dw / s1	$3 \times 3 \times 32\ dw$	$112 \times 112 \times 32$
Conv / s1	$1 \times 1 \times 32 \times 64$	$112 \times 112 \times 32$
Conv dw / s2	$3 \times 3 \times 64\ dw$	$112 \times 112 \times 64$
Conv / s1	$1 \times 1 \times 64 \times 128$	$56 \times 56 \times 64$
Conv dw / s1	$3 \times 3 \times 128\ dw$	$56 \times 56 \times 128$
Conv / s1	$1 \times 1 \times 128 \times 128$	$56 \times 56 \times 128$
Conv dw / s2	$3 \times 3 \times 128\ dw$	$56 \times 56 \times 128$
Conv / s1	$1 \times 1 \times 128 \times 256\ dw$	$28 \times 28 \times 128$
Conv dw / s1	$3 \times 3 \times 256\ dw$	$28 \times 28 \times 256$
Conv / s1	$1 \times 1 \times 256 \times 256$	$28 \times 28 \times 256$
Conv dw / s2	$3 \times 3 \times 256\ dw$	$28 \times 28 \times 256$
Conv / s1	$1 \times 1 \times 256 \times 512$	$14 \times 14 \times 256$
5×Conv dw / s1	$3 \times 3 \times 512\ dw$	$14 \times 14 \times 512$
5×Conv / s1	$1 \times 1 \times 512 \times 512$	$14 \times 14 \times 512$
Conv dw / s2	$3 \times 3 \times 512\ dw$	$14 \times 14 \times 512$
Conv / s1	$1 \times 1 \times 512 \times 1024$	$7 \times 7 \times 512$
Conv dw / s2	$3 \times 3 \times 1024\ dw$	$7 \times 7 \times 1024$
Conv / s1	$1 \times 1 \times 1024 \times 1024$	$7 \times 7 \times 1024$
Avg Pool / s1	$Pool\ 7 \times 7$	$7 \times 7 \times 1024$
FC / s1	1024×1000	$1 \times 1 \times 1024$
Softmax / s1	Clasificador	$1 \times 1 \times 1000$

Nota. La tabla presenta las diferentes etapas que constituyen el modelo *MobileNet*, el cual es utilizado para el presente trabajo.

Hardware

Con respecto al *hardware* se trabaja sobre dos equipos, una máquina virtual y una computadora portátil. La máquina virtual empleada se denomina *Colab*, se encuentra en la nube y es proporcionada por Google, está orientada a la ciencia de datos e investigación sobre IA, además que permite programar y ejecutar código en Python. Las características de estos equipos se presentan en la Tabla 3.

Tabla 3

Características de equipos utilizados para el entrenamiento del modelo de clasificación.

Descripción	Máquina virtual	Computadora portátil
Marca y Modelo	Google Colab	HP Notebook
CPU	AMD EPYC 7B12	Intel Core i5-6200U
GPU	T4 NVIDIA-SMI 460.32.03	-
RAM	12 GB	8 GB
Frecuencia del procesador	2.25 GHz	2.30 – 2.40 GHz
Sistema operativo	Linux	Windows 10

Nota. La tabla presenta las características técnicas de los equipos utilizados para llevar a cabo el entrenamiento del modelo de clasificación de imágenes con la aplicación de la técnica de Transfer Learning.

Al utilizar la versión gratuita de Google Colab se tiene un tiempo limitado de 12 horas de uso del entorno, es por ello que para la ejecución de tareas que requieren mucho tiempo se hace uso de la computadora portátil que, gracias a sus características que se presentan en la Tabla 3 no presenta interrupciones durante el entrenamiento y exportación del modelo de clasificación.

Metodología de entrenamiento y validación del modelo MobileNetV2

El desarrollo del entrenamiento y validación se lo realiza en base a diferentes pautas y fases que permiten al modelo de clasificación entrenado sea capaz de realizar predicciones con una exactitud cerca al 100% de validez. Con una exactitud alta en sus predicciones se considera al modelo apto para desempeñarse como clasificador en el entorno para el cual fue entrenado, en el caso contrario de obtener resultados no deseados o predicciones con baja exactitud se deben ajustar ciertos parámetros del modelo y entrenamiento. Este proceso es iterativo hasta obtener predicciones deseadas con alta exactitud.

Auxiliares de entrenamiento

Estos auxiliares permiten crear un entorno de entrenamiento más controlado y eficiente, a fin de garantizar buenos resultados al terminar el entrenamiento, además de realizarlo en un menor tiempo.

- **TensorBoard:** obtiene estadísticas durante el entrenamiento de métricas como la exactitud y la pérdida, de esta manera se determina visualmente el estado del entrenamiento durante todas las etapas.
- ***Checkpoint:*** guarda el modelo con cierta frecuencia durante el entrenamiento. Este auxiliar monitoriza métricas como la exactitud y la pérdida en el entrenamiento, así como la validación, si se tiene cierta mejora en una de estas métricas en alguna etapa del entrenamiento, se guarda el modelo hasta dicha etapa.
- ***Early Stopping:*** detiene el entrenamiento cuando una métrica como exactitud o pérdida haya dejado de mejorar, de esta manera se logra que el modelo no entre en un estado de sobreajuste.

Compilación de modelo

Con la elección del modelo y la adaptación respectiva realizada, se compila el modelo para que quede listo para usar y entrenar. En esta fase se define un optimizador y una función de pérdida.

- ***Optimizer:*** permite minimizar el error de predicción y asignar mejores pesos al modelo.
- ***Loss:*** la función de pérdida determina el rendimiento del modelo entrenado.
- ***Metrics:*** indica con qué métrica se juzga o se valora el rendimiento del modelo.

Parámetros de entrenamiento

Previo a iniciar el entrenamiento del modelo se requiere definir ciertas pautas con las que se rige dicho entrenamiento, éstas son:

- **Datos de entrenamiento:** define los datos de que se usan para el entrenamiento del modelo.
- **Datos de validación:** define los datos que se utilizan para validar al modelo durante el entrenamiento.
- ***Epoch*:** este valor indica las veces que los datos pasan por toda la CNN, con muchas épocas puede producirse un sobreajuste.
- ***Batch Size*:** este valor indica la cantidad de datos que se procesan por cada época. Es directamente proporcional a la capacidad de memoria que requiere de la computadora donde se realiza el entrenamiento y al tiempo de entrenamiento. Es decir, un valor alto requiere más memoria y mayor tiempo de entrenamiento, sin embargo, es más probable que el modelo aprenda más detalles de los datos de entrenamiento.

- ***Callbacks*:** en este campo se hace referencia a los auxiliares de entrenamiento TensorBoard, *Checkpoint, Early Stopping* para que sean utilizados durante el entrenamiento.

Exportación y conversión del modelo entrenado

Una vez finalizado el entrenamiento con las métricas de exactitud y pérdidas idóneas, es momento de guardar el modelo y convertirlo a una extensión capaz de ser integrado en una aplicación móvil Android.

Keras tiene la función *save* que permite guardar el modelo con extensión .h5 y para convertirlo a extensión compatible con Tensorflow Lite. A partir de esto, el modelo puede ser integrado en una aplicación móvil.

Herramientas y entorno de entrenamiento del detector de dígitos

En este apartado se expone los materiales y métodos utilizados para el desarrollo del detector de dígitos el cual se encarga de realizar la localización y predicción de los números en los medidores que indican el consumo.

Dataset

El conjunto de imágenes que se utiliza para el entrenamiento del modelo de detector de dígitos, es similar al conjunto de imágenes de medidores de agua y de energía eléctrica utilizado para el entrenamiento del modelo de clasificación. Para este caso se utilizó únicamente 1000 imágenes por cada clase, en total 2000 imágenes, la peculiaridad que diferencia este conjunto de imágenes al anterior es que se seleccionaron las imágenes que tenga mejor plano del medidor, es decir, que se pueda apreciar lo suficientemente bien los dígitos de los medidores para que posteriormente puedan ser etiquetados y localizados fácilmente, además que dichas imágenes presenten un plano centrado del medidor. Otra característica que deben tener las imágenes, es que su dimensión no sea menor a 320x320 pixeles, con el fin de que éstas tengan suficiente nitidez. Un ejemplo de este *dataset* de imágenes se presenta en la Figura 17.

Figura 17

Ejemplo del dataset para el entrenamiento del detector de dígitos

Nota. La figura presenta una porción de ejemplo del *dataset* utilizado para el etiquetado y posteriormente entrenamiento del detector de dígitos.

LabelImg

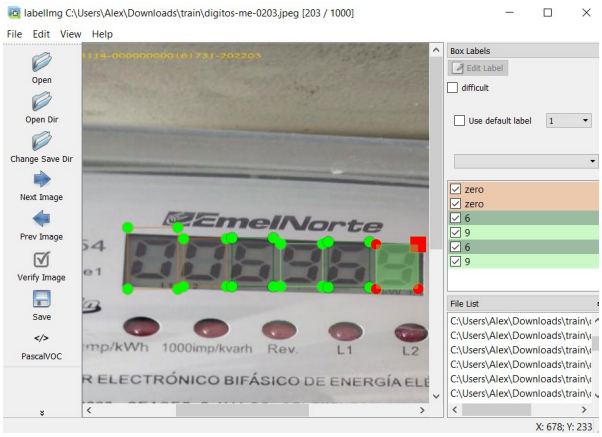
LabelImg es una herramienta sencilla y básica para etiquetar imágenes para crear un conjunto de datos que son útiles en el entrenamiento del modelo. Las anotaciones se guardan como archivos XML en formato PASCAL VOC, el formato utilizado por ImageNet. Además, también es compatible con los formatos YOLO y CreateML (GitHub & Tzutalin, 2022).

En el ML y DL el etiquetado de imágenes es la técnica de categorizar una imagen con la utilización de texto de anotación, herramientas de software o ambos para mostrar las características de datos que desea que el modelo identifique (Boesch, 2020).

Cuando se realizan anotaciones de imágenes, básicamente agrega metadatos a un conjunto de imágenes para especificar características propias del *dataset*. La creación de modelos de AV sigue la filosofía de dato que entra, dato que significa que es importante etiquetar con cuidado y precisión, es por ello que los conjuntos de datos de alta calidad son esenciales para la AV y la construcción de un modelo de alto rendimiento (Nelson, 2020). En la Figura 18 se muestra la interfaz gráfica de la aplicación labelImg que se utiliza para anotar las regiones de donde se encuentran los dígitos de un medidor de energía eléctrica.

Figura 18

Interfaz de la aplicación labelImg.



Nota. se muestra la aplicación labelImg con la selección etiquetada de las regiones en donde se encuentran los dígitos de un medidor de energía eléctrica, el resultado final de este etiquetado es un archivo plano *.txt* que se usa para el entrenamiento.

Archivos de entrenamiento

El entrenamiento de un modelo de detección de dígitos es diferente al de un clasificador, debido a que en la situación del detector se espera como resultado no solo la clasificación del dígito sino también la ubicación del dígito en la imagen. Para el etiquetado de la región donde se encuentran los dígitos se requiere un archivo *.pbtxt* que contiene las clases que se van a identificar, por ejemplo, para el número uno se tiene id:1 y Clase 1, tal como muestra el Código 1

Otro archivo que se necesita para el entrenamiento es el generado por la herramienta de etiquetado labelImg, este archivo tiene la extensión *.xml* donde se indica en que región de la imagen se encuentra determinada clase. Un ejemplo de este archivo se muestra en el Código 2.

Código 1

Ejemplo de archivo de entrenamiento .pbtxt con las clases a detectar.

```
item {
id: 1
name: 'Clase 1'
}

item {
id: 2
name: 'Clase 2'
}

…

item {
id: n
name: 'Clase n'
}
```

Nota. La figura muestra un ejemplo de cómo está formado el archivo *.pbtxt* que indica las diferentes clases que se requiere detectar después del entrenamiento.

Código 2

Ejemplo de archivo .xml generado por la herramienta labelImg.

```
<?xml version="1.0"?>
- <annotation>
        <folder>input</folder>
        <filename>Image1.jpg</filename>
        <path>C:\User\zizou\Destok\annotation-tool\input\Image1.jpg</path>
       -<source>
            <database>Unknown</Unknown>
        <source>
       +<size>
        <segmented>0</segmented>
       -<object>
            <name>with_mask</name>
            <pose>Unspecified</pose>
            <truncated>0</truncated>
            <difficult>0</difficult>
         -   <bnbox>
                <xmin>243</xmin>
                <ymin>124</ymin>
                <xmax>354</xmax>
                <ymax>204</ymax>
            </bnbox>
        </object>
    </annotation>
```

Hardware

El hardware que se utiliza para llevar a cabo el entrenamiento es la máquina virtual de Google Colab en su versión Pro. Este plan adquirido ofrece una GPU más rápida y más memoria RAM que la que se ofrece en su versión gratuita, además de mayores tiempos de ejecución sin cortes, lo cual, garantiza que el entrenamiento se efectúe sin interrupciones.

La Tabla 4 presenta las características del hardware que ofrece la versión Pro.

Tabla 4

Características de la máquina de Colab Pro.

Ítem	Versión
Marca y Modelo	Google Colab (*Pro version*)
GPU	P1000 NVIDIA-SMI 460.32.03
CUDA	11.2
RAM	27.3 GB

Nota. La tabla presenta las prestaciones mejoradas que se obtiene de la máquina de Google Colab en su versión Pro.

Entorno de desarrollo de aplicación móvil

Android Studio

Android es un sistema operativo móvil actualmente desarrollado por Google, basado en el *kernel* de Linux y diseñado principalmente para dispositivos móviles con pantalla táctil, como teléfonos inteligentes y tabletas (Aliferi, 2016).

Android Studio es el entorno de desarrollo integrado oficial (IDE, del inglés *Integrated Development Environment*) para el desarrollo de aplicaciones ejecutadas en el sistema operativo Android. El núcleo de este IDE es un potente editor de códigos y las herramientas para desarrolladores, además, se puede integrar con GitHub y plantillas de código para

compilar funciones de aplicaciones comunes y también importar código, es compatible con Google *Cloud Platform*, lo que facilita la integración con ML Kit de Firebase, factor indispensable para la ejecución del modelo de clasificación entrenado en Google Colab (Smyth, 2015).

Base de datos Could Firestore

Uno de los pilares fundamentales en aplicaciones web o móviles de alto rendimiento es una base de datos robusta. Una base de datos debe facilitar el almacenamiento de información de manera organizada, crear una vía para recuperar y administrar datos de forma *offline*.

Cloud Firestore es una base de datos de alto rendimiento que admite el escalado automático diseñada para el desarrollo de aplicaciones web y móviles gestionada por Firebase. Cloud Firestore mantiene los datos sincronizados entre cliente y servidor a través de objetos de escucha en tiempo real y ofrece soporte sin conexión, por lo que permite gestionar datos sin importar la latencia de la red ni la conectividad a la Internet (Google, 2022).

Cloud Firestore es una parte integral de la plataforma Google Firebase. Toma la forma de un servidor de base de datos NoSQL basado en la nube que se encarga del almacenamiento y sincronización de datos. Las aplicaciones pueden interactuar directamente con Firestore con el uso de SDK nativos. Además, es compatible con una amplia variedad de tecnologías como Java, C++, Unity, Go, Node.js SDK, REST y RPC API (Batschinski, 2021a).

Las ventajas de Firestore respecto a otras bases de datos es la sincronización fuera de línea, las aplicaciones web, Android e iOS utilizan la función sin conexión de Firestore. Le permite al usuario almacenar datos sin conexión y sincronizarlos con la base de datos inmediatamente cuando se restablece la conectividad.

La estructura de precios es una opción rentable para los desarrolladores. Al utilizar Firestore los precios dependen del número de documentos que se han escrito, leído y eliminado, también de la cantidad de almacenamiento que utiliza la base de datos, cantidad de ancho de banda utilizado, tanto el uso del almacenamiento como el del ancho de banda se

calculan en GB. Todos los cargos se acumulan diariamente. Firestore puede escalar automáticamente cuando aumenta la demanda de datos del usuario. En la Tabla 5 se muestra los precios del almacenamiento y de las operaciones correspondientes a cada ubicación de Firestore.

Tabla 5

Costos de Firestore.

Tipo de operación	Cuota gratuita al día	Precio tras superar la cuota gratuita	Unidad de precio
Operaciones de lectura de documentos	50000	0.06 USD	Por 1000000 documentos
Operaciones de escritura de documentos	20000	0.18 USD	Por 1000000
Operaciones de eliminación de documentos	20000	0.02 USD	Por 1000000
Datos almacenados	1 GB de almacenamiento	0.18 USD	GB al mes

Nota. En la tabla se indica los precios del almacenamiento y de las operaciones de lectura, escritura y eliminación correspondientes a Could Firestore ubicado en EE.UU.

Could Firestore comparte de la seguridad de Google, en esta se incluye la validación automática de datos, reglas no en cascada, protección de datos y características funcionales de recuperación ante desastres (Batschinski, 2021b).

Tecnologías para el diseño de páginas web

El lenguaje de marcado de hipertexto HTML (del inglés, HyperText Markup Language) es uno de los tres componentes principales de las páginas web modernas, junto con las hojas de estilo en cascada CSS (del inglés, *Cascading Style Sheets*) y JavaScript. HTML indica al

navegador qué elementos deben incluirse en la página web y en qué orden. CSS indica cómo se debe diseñar cada elemento. JavaScript proporciona un medio para que los autores de páginas web manipulen estos elementos mediante programación y en respuesta a las acciones del usuario final. A continuación, se describe las tecnologías utilizadas.

HTML

Proporciona la estructura, cuando escribimos una dirección de una página web en cualquier navegador web se realiza una petición de datos a un servidor donde se encuentra alojada la página, para ello, se utiliza un conjunto de protocolos como: TCP (del inglés, *Transmission Control Protocol*) y HTTP (de sus siglas en inglés, *Hypertext Transfer Protocol*). Si la petición se realiza de forma correcta, el servidor responderá paquetes de archivos entre los que se encuentra código y ficheros que el navegador organizará para mostrar el resultado en pantalla.

CSS

CSS proporciona estilo y apariencia, es decir, con CSS se puede asignar fuentes y color a textos o cajas, modificar tamaños, añadir imágenes de fondo, definir márgenes o incluso cambiar completamente la apariencia de un elemento HTML como una lista para convertirla en una barra o menú de navegación, además, permite que una página web se vea correctamente en otros dispositivos como móviles o tabletas. Es lo que se conoce como diseño web adaptativo o *responsive*.

Java Script

JavaScript es un lenguaje de programación de *backend* que permite implementar dinamismo y funcionalidad a una página web. Además del contenido estático, con JavaScript podremos mostrar actualizaciones de contenido, vincular eventos dinámicos a elementos HTML como: accesos a menús, filtros en formularios, botones. También permite almacenar datos en variables y con ellos usar funciones complementarias como gráficos o mapas mediante APIs de terceros.

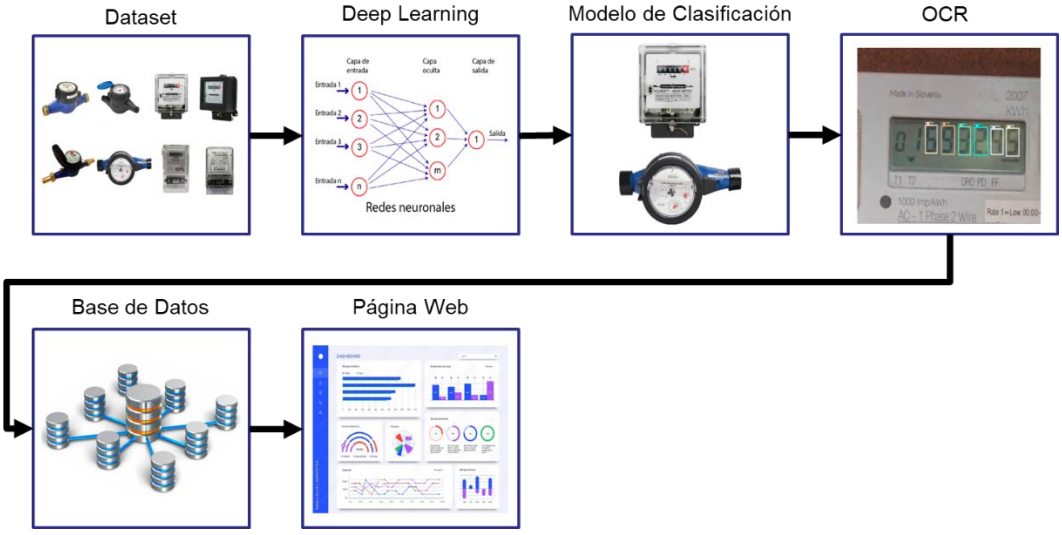
Capítulo IV

Diseño e implementación

El diagrama general del sistema de AV desarrollado y ejecutado en una aplicación móvil se presenta en la Figura 19. Se parte de una *dataset* que comprende las imágenes de medidores de agua y electricidad, con este se puede entrenar un modelo de DL para la clasificación de medidores. Una vez hecha la clasificación de medidores se realiza la lectura de los dígitos que marcan el consumo del servicio básico. Con esta lectura realizada e interpretada en forma de dato se la envía a una base de datos para posteriormente ser presentada en una página web.

Figura 19

Diagrama general del sistema.



A seguir, se va a describir las configuraciones necesarias para el entrenamiento de dos modelos de clasificación de medidores, un pre-clasificador para distinguir si existe un medidor o no en la fotografía y un clasificador para diferenciar entre un medidor de agua y un medidor de energía eléctrica, y modelo para la detección de dígitos (OCR). Finalmente, se encuentra la integración de los modelos obtenidos en una aplicación móvil Android y presentación en una página web.

Modelo de clasificación de medidores

Configuración del entorno de entrenamiento del modelo de clasificación

El desarrollo del entrenamiento del modelo de clasificación se realiza en un *script* codificado con el lenguaje de programación Python, ya que en este lenguaje están disponibles herramientas de ML y bibliotecas descritas en el capítulo anterior, las más importantes son TensorFlow y Keras. La plataforma donde se ejecuta este *script* posee de algunos requerimientos de *hardware*, como por ejemplo garantizar el uso de una GPU para que el entrenamiento se realice en un menor tiempo.

Colab

El *script* de entrenamiento se ejecuta en la plataforma de Colab en su versión gratuita. Al crear un *script* desde cero, permite ajustar los parámetros del entorno de ejecución y de esta manera usar un acelerador por *hardware*. Se puede escoger entre tres tipos: CPU, GPU y TPU.

Vincular Google Drive con Colab

La ejecución del entrenamiento requiere del uso de datos externos que no se encuentran por defecto en la plataforma virtual de Colab, para lo cual se realiza un vínculo entre Colab y Google Drive, y así utilizar los datos que están alojados en la unidad de Drive.

Código 3

Vincular el script con Google Drive.

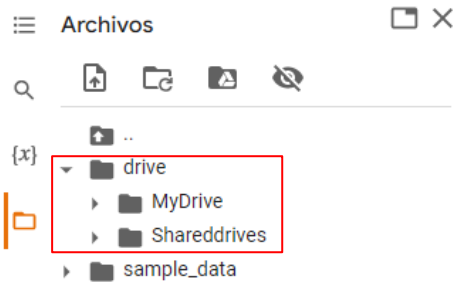
```
from google.colab import drive
drive.mount('/content/drive')
```

Nota. El código muestra cómo se enlaza el *script* realizado con Google Drive y de esta manera acceder a los archivos de la unidad correspondiente.

Al ejecutar las líneas presentadas en el Código 3, se solicita una serie de permisos y una cuenta de Gmail con la cual se desea ingresar, posteriormente se termina de ejecutarse y en la sección de archivos se muestra un directorio denominado `drive` el cual corresponde a la unidad de Google Drive que se acaba de vincular, como presenta la Figura 20.

Figura 20

Archivos disponibles para el script.



TensorBoard

Posterior, se implementa la herramienta TensorBoard para obtener estadísticas del entrenamiento y verificar el comportamiento del modelo y su aprendizaje durante todo el entrenamiento, tal como se muestra en la Figura 21.

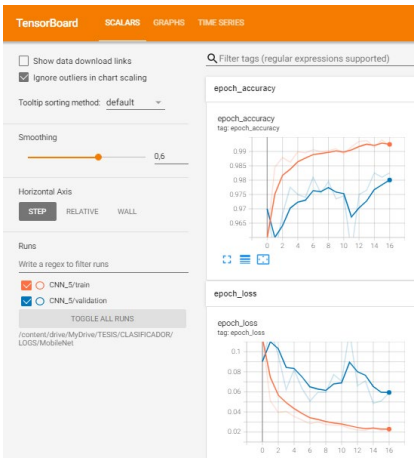
Código 4

Implementación de TensorBoard.

```
%load_ext tensorboard
%tensorboard—
logdir /content/drive/MyDrive/TESIS/CLASIFICADOR/LOGS/MobileNet
```

Figura 21

Gráficas de accuracy y loss que presenta TensorBoard.



A su vez también se define una variable de TensorBoard que indica donde se guardan los datos del entrenamiento, con la siguiente línea que muestra la Código 5.

Código 5

Guardar las estadísticas de TensorBoard.

```
tensorboard = TensorBoard(log_dir="G:/Mi
unidad/TESIS/CLASIFICADOR/LOGS/MobileNet/")
```

Nota. Se muestra cómo se establece el directorio donde se guardarán las estadísticas del entrenamiento, este directorio se encuentra en la unidad de Google Drive.

CheckPoint

Con el propósito de realizar respaldos del entrenamiento se establecieron puntos de control durante el entrenamiento que permitan guardar el modelo cuando tenga valores adecuados de *accuracy* y *loss*, esta implementación se muestra el Código 6. En este caso se utiliza como métrica de control `val_accuracy`, ésta es una métrica más real que indica el grado de exactitud del clasificador cuando se realiza una validación al terminar una época de entrenamiento.

Código 6

Implementación de auxiliar ModelCheckpoint.

```
checkpointCNN5 = ModelCheckpoint(
filepath='/content/drive/MyDrive/CLASIFICADOR/CHECKPOINTS/MobileNet/modelCNN
5_{epoch:02d}-{val_acc:.2f}-{val_loss:.2f}.h5',
                                 monitor='val_accuracy',
                                 verbose=1,
                                 save_best_only=True,
                                 save_weights_only=False,
                                 mode='auto', period=1)
```

Nota. Se muestra cómo se configura la función ModelCheckpoint, uno de los auxiliares para realizar respaldos del entrenamiento.

Adicionalmente, cuando se realice un respaldo del modelo se guarda con la descripción de la época en la que fue obtenido y el valor de *validation accuracy* y de *validation los*s. Todos los parámetros establecidos en esta función se presentan en la Tabla 6.

Early Stopping

Permite terminar el entrenamiento cuando ya no se obtuvieron mejoras en la métrica *validation accuracy*, se monitoriza esta métrica con una frecuencia de 20 épocas ya que con este valor se ha obtenido mejores resultado después de un proceso de pruebas, es decir, que si desde la última mejora han transcurrido 20 épocas se procede a terminar con el entrenamiento, porque ya no se obtendrán mejores resultados que los obtenidos, además de que esto ayuda a que el clasificador no tenga un sobre entrenamiento o un sobreajuste y como consecuencia se tiene un menor tiempo de entrenamiento y menor uso de recursos computacionales. La implementación de la función `EarlyStopping` en el *script* se presenta en el Código 7.

Tabla 6

Parámetros de la función ModelCheckpoint.

Parámetro	Valor	Descripción
Filepath	Directorio de preferencia	En este directorio se guardarán los modelos guardados
Monitor	val_accuracy	Métrica a monitorizar si mejora en el entrenamiento
Verbose	1	Indica mensaje cuando la función realiza una acción
Save_best_only	True	Guarda el modelo mejor entrenado y no sobrescribe en los demás
Sabe_weights_only	False	No se guarda únicamente los pesos, sino el modelo completo
Mode	auto	Se adapta automáticamente a la métrica monitorizada
Period	1	Monitoriza cada 1 periodo de entrenamiento

Nota. La tabla describe los diferentes parámetros utilizados para la implementación de la función `ModelCheckpoint`.

Código 7

Implementación de la función EarlyStopping.

```
stop = tf.keras.callbacks.EarlyStopping(monitor='val_accuracy',
                    min_delta=0, patience=20,
                    verbose=1,
                    mode='auto',
                    )
```

Esta función permite ajustar otros parámetros importantes para el desempeño del entrenamiento. La descripción de cada uno de éstos se presenta en la Tabla 7.

Tabla 7

Parámetros de la función EarlyStopping.

Parámetro	Valor	Descripción
Monitor	val_accuracy	Métrica a monitorizar si mejora en el entrenamiento
Min_delta	0	Diferencia mínima entre valor de métrica monitorizada
Patience	20	Cada 20 épocas se realiza una comparación de la métrica
Verbose	1	Indica mensaje cuando la función realiza una acción
Mode	auto	Se adapta automáticamente a la métrica monitoreada

Nota. La tabla describe los diferentes parámetros utilizados para la implementación de la función EarlyStopping.

Implementación del dataset de imágenes para entrenamiento del modelo de clasificación

Previo a realizar el entrenamiento se debe obtener el *dataset*, que en este caso son imágenes de medidores de consumo de agua potable y de energía eléctrica. A estas imágenes se aplican ciertos cambios como rotación, zoom y giros con la aplicación de la función `ImageDataGenerator` que se muestra en el Código 8.

Código 8

Implementación de la función ImageDataGenerator.

```
datagen = ImageDataGenerator(
    rescale=1./255,
    rotation_range=20,
    zoom_range=0.2,
    width_shift_range=0.1,
    height_shift_range=0.1,
    horizontal_flip=True,
    vertical_flip=False,
    preprocessing_function=preprocess_input
)
```

La descripción de los parámetros que constituyen la función ImageDataGenerator se muestra en la Tabla 8.

Tabla 8

Parámetros de la función ImageDataGenerator.

Parámetro	Valor	Descripción
Rotation_range	20	Se realizará rotaciones aleatorias de hasta 20 grados
Zoom_range	0.2	Se realizará zooms aleatorios de hasta un 80% (1 – 0.2 = 0.8) de la imagen
Width_shift_range	0.1	Efecto de desplazamiento horizontal de la imagen de hasta 10%
Height_shift_range	0.1	Efecto de desplazamiento vertical de la imagen de hasta 10%
Horizontal_flip	True	Efecto espejo horizontal en la imagen
Vertical_flip	False	Efecto espejo vertical en la imagen desactivado
Preprocessing_function	Preporcess_input	Las transformaciones anteriormente definidas se aplican a cada imagen

Una vez definidas las transformaciones que se realizan al *dataset* de entrada, es momento de importar esta información para el entrenamiento, como muestra el Código 9. La variable `data_train` contiene las imágenes de entrenamiento y la variable `data_validation` las imágenes de validaciones. Se trabaja con un total de 10000 imágenes (5000 imágenes de medidores de agua y 5000 de medidores de electricidad), el 80% de estas están destinadas para entrenamiento y el 20% restante para validación.

Código 9

Obtener el dataset de entrenamiento.

```
# DATOS DE ENTRENAMIENTO
data_train = datagen.flow_from_directory(directory="G:\Mi
unidad\TESIS\CLASIFICADOR\DATASET\TRAIN", target_size=(224,224),
                                                    batch_size=32,
class_mode='categorical')
data_validation = datagen.flow_from_directory(directory="G:\Mi
unidad\TESIS\CLASIFICADOR\DATASET\VALIDATION", target_size=(224,224),
                                                    batch_size=32,
class_mode='categorical')
```

Nota. La figura muestra cómo se implementan las imágenes para entrenamiento y validación.

La función que permite la importación de las imágenes es `flow_from_directory` y la descripción de sus parámetros se presentan en la Tabla 9.

Obtención del modelo de clasificación

Como se había especificado en el capítulo anterior, para el entrenamiento del clasificador se utiliza el método de *Transfer Learning*. El modelo escogido para el desarrollo del clasificador es MobileNetV2, éste se encuentra alojado en el sitio web oficial de TensorFlow Hub. Con la URL que proporciona el mismo sitio para su uso se logra importa el modelo para este *script* y posteriormente se guarda el modelo en la variable *mobilenetv2* en la cual se indica la URL del modelo y el tipo de entrada que tendrá, para este caso son imágenes de 224x224 pixeles y con 3 canales (RGB).En el Código 10 se muestra este proceso de creación del modelo y, además, se especifica que no se desea entrenar nuevamente el modelo y se harán validos los pesos con los que fue entrenado y validado (`mobilenetv2.trainable = False`).

Tabla 9

Parámetros de la función flow_from_directory.

Parámetro	Valor	Descripción
Directory	Directorio específico	Donde se encuentran las imágenes de entrenamiento y validación
Target_size	224,224	Las imágenes se redimensionan a 224x224 pixeles
Batch_size	32	Se procesan las imágenes de entrada en lotes de 32
Class_mode	categorical	Ideal para clasificaciones de varios objetos

Nota. La tabla describe los diferentes parámetros utilizados para la implementación de la función flow_from_directory.

Finalmente, se modifica el modelo importado, pero únicamente en su última capa denominada *softmax*. Con la función `Sequential` se crea el modelo deseado, con la unión del modelo importado y una capa de salida con activación *softmax*, para que el modelo sea capaz de realizar más de dos predicciones.

Código 10

Importar el modelo CNN para entrenamiento.

```
url = "https://tfhub.dev/google/tf2-preview/mobilenet_v2/feature_vector/4"

mobilenetv2 = hub.KerasLayer(url, input_shape=(224,224,3))

mobilenetv2.trainable = False

CNN_5 = tf.keras.Sequential([
    mobilenetv2, # Modelo importado
    tf.keras.layers.Dense(2, activation='softmax')# Capa de salida con dos p
redicciones
])
```

Nota. En el código se muestran la importación del modelo y la modificación de la última capa de la CNN.

Ya que se ha creado el modelo, el siguiente paso es compilarlo y que quede listo para el entrenamiento. Esto se lo realiza con la función `compile` como se muestra en el Código 11.

Código 11

Compilar el modelo creado.

```
model.compile(
    optimizer='adam',
    loss='categorical_crossentropy',
    metrics=['accuracy']
)
```

Nota. El código presenta manera de compilar el modelo con su optimizador y función de pérdida.

La descripción de los parámetros ajustados de esta función se muestra en la Tabla 10.

Tabla 10

Parámetros de la función Compile.

Parámetro	Valor	Descripción
Optimizer	Adam	Se utiliza el optimizador Adam el cual presenta menores pérdidas que otros.
Loss	categorical_crossentropy	Se utiliza la función Categorical Cross Entropy, ésta se adapta correctamente a aplicaciones de clasificación.
Metrics	accuracy	El rendimiento del modelo será en base a su exactitud

Nota. La tabla describe los diferentes parámetros utilizados para la implementación de la función Compile.

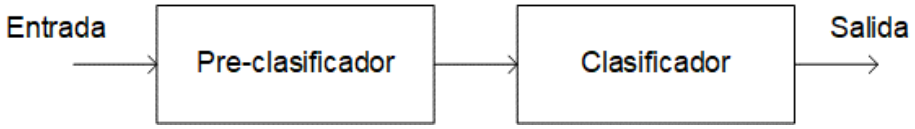
Entrenamiento del modelo de clasificación

El modelo de clasificación propuesto se muestra en la Figura 22, en donde se detalla un módulo previo denominado pre clasificador, su función es identificar si la imagen obtenida pertenece a un medidor de agua potable o energía eléctrica, en el caso de no identificar nada,

la salida será “No se ha detectado un medidor”, caso contrario pasa al siguiente módulo denominado clasificador.

Figura 22

Modelo propuesto para el clasificador



Nota. Es importante mencionar que el modelo del preclasificador y el modelo del clasificado tiene los mismos parámetros, únicamente cambia el *dataset* con el cual se realiza el entrenamiento, por ello, únicamente se detalla el código del clasificador.

Entonces, con el modelo compilado y todos los auxiliares definidos, es momento de efectuar el entrenamiento con las líneas de código que se muestran en el Código 12.

Código 12

Ejecutar el entrenamiento del modelo.

```
# ENTRENAMIENTO
train= model.fit(
    data_train,
    epochs=50, batch_size=32,
    validation_data=data_validation,
    callbacks=[tensorboardCNN5, checkpointCNN5, stop]
    )
```

Nota. El código indica cómo se implementan la función `fit` y la configuración de sus diferentes parámetros.

La función `fit` permite ejecutar el entrenamiento en base a ciertos parámetros, los cuales se definen en la Tabla 11.

Tabla 11

Parámetros de la función `fit`*.*

Parámetro	Valor	Descripción
Train_data	Data_train	Imágenes de entrenamiento definidas
Epochs	150	Número de épocas para el entrenamiento
Batch_size	32	Número imágenes que se procesaran por cada paso
Validation_data	Data_validation	Imágenes de validación definidas
Callbacks	tensorboardCNN5 checkpointCNN5 stop	Auxiliares del entrenamiento

Nota. La tabla describe los diferentes parámetros utilizados para la implementación de la función `fit`. Se establece 150 época después de un proceso de pruebas, se considera este un valor optimo para el entrenamiento.

Obtención de modelo de clasificación en aplicación móvil

Al terminar el entrenamiento de manera exitosa y con los resultados de *accuracy* y *loss* deseados se procede a guardar el modelo en extensión *.h5* y `tflite` para posteriormente realizar pruebas con el clasificador obtenido, este proceso se muestra en el Código 13.

Código 13

Exportar el modelo entrenado.

```
TF_LITE_MODEL_FILE_NAME = "G:/Mi
unidad/TESIS/CLASIFICADOR/MODELOS/MobileNet/modelX.tflite"
tf_lite_converter = tf.lite.TFLiteConverter.from_keras_model(model)
tflite_model = tf_lite_converter.convert()

open(TF_LITE_MODEL_FILE_NAME, "wb").write(tflite_model)
```

Nota. El código muestra las líneas ejecutadas para exportar el modelo entrenado en extensión .h5 y su posterior conversión a extensión. tflite.

Modelo de detección de dígitos en medidores

Etiquetado del dataset de imágenes para entrenamiento del detector

La aplicación LabelImg admite el etiuetado en formato de archivo de texto YOLO o VOC XML. Se recomienda que los usuarios utilicen el formato VOC XML predeterminado para producir etiquetas, debido a que es el formato de ImageNet, es un estándar universal que está asociado con la detección de objetos, mientras que las diferentes implementaciones de YOLO tienen diferentes formatos de línea de texto. Además, puede convertir fácilmente de VOC XML a cualquier otro formato con la ayuda de otras herramientas.

Configuración del entorno de entrenamiento para la detección de dígitos

Primero se recopilaron las imágenes de 1000 medidores de energía eléctrica y 1000 de medidores de agua potable, de este conjunto de imágenes se tomó el 80% para datos de prueba y el 20% para datos de entrenamiento.

Posteriormente, se realizan las anotaciones con el software labelImg que se describió en el capítulo anterior, para ello, es necesario modificar el archivo label_map.pbtxt en donde se configura el conjunto de datos a etiquetar, que para este trabajo serían los números: 0, 1, 2, 3, 4, 5, 6, 7, 8 y 9.

La configuración del entorno de entrenamiento para el detector de dígitos tiene las mismas características de software y hardware que el entorno de entrenamiento del modelo de clasificación, los primero es importar las librerías necesarias, estas se muestran en el Código 14.

Código 14

Instalar las librerías para el entrenamiento.

```
!pip install tensorflow==2.5
!pip install flatbuffers==1.12
import xml.etree.ElementTree as ET
import pandas as pd
import tensorflow as tf
print(tf.__version__)
```

Posterior, se crea carpetas en Google Drive vinculadas con Google Colab en donde se guardan los *checkpoints* durante el entrenamiento, esto se muestra en el Código 15.

Código 15

Vincular Notebook con los archivos de drive.

```
from google.colab import drive
drive.mount('/content/gdrive')
!ln -s /content/gdrive/My\ Drive/ /mydrive
!ls /mydrive
```

Nota. El código muestra cómo se enlaza el *script* realizado con Google Drive y de esta manera acceder a los archivos de la unidad correspondiente.

Después se genera los archivos para el entrenamiento, para lo cual primero se crea una carpeta *labels* para los correspondientes archivos XML etiquetados en formato PASCAL_VOC. El archivo XML contiene los datos de las 4 coordenadas del cuadro delimitador mencionado como: xmin, ymin, xmax, ymax, xmin-arriba a la izquierda, ymin-arriba a la izquierda, xmax-abajo a la derecha y ymax-abajo a la derecha. Esto se muestra en el Código 2.

También contiene el nombre del archivo de imagen de entrada y el nombre de la clase de objeto en la imagen. Si la imagen contiene varias clases de objetos, el archivo XML también tiene varias clases y varios valores para sus respectivas coordenadas escritas en ellos. A partir de este paso, se generó los archivos de entrenamiento y de prueba, se ajusta el 20% del *dataset* a prueba, esto se muestra en el Código 16.

Código 16

Generación de archivos .csv de train y test.

```
for label_path in ['train_labels', 'test_labels']:
  image_path = os.path.join(os.getcwd(), label_path)
  xml_df, classes = xml_to_csv(label_path)
  xml_df.to_csv(f'{label_path}.csv', index=None)
  print(f'Successfully converted {label_path} xml to csv.')
```

Posterior se generan los archivos *tfrecord* fundamentales para el entrenamiento, para esto se ejecuta el código que se muestra en el Código 17.

Código 17

Generación de los archivos tfrecord fundamentales.

```
#For train.record
!python /mydrive/customTF2/generate_tfrecord.py train_labels.csv  label_map.
pbtxt images/ train.record
```

A continuación, en el Código 18 se descarga el modelo MobileNet V2 para realizar el proceso de *Transfer Learning*.

Código 18

Descarga del modelo MobilNet V2.

```
!wget http://download.tensorflow.org/models/object_detection/tf2/20200711/ssd_mobilenet_v2_fpnlite_320x320_coco17_tpu-8.tar.gz
!tar -xzvf ssd_mobilenet_v2_fpnlite_320x320_coco17_tpu-8.tar.gz
```

Nota. Previo a ejecutar el siguiente paso, se debe modificar el archivo *.config* con las configuraciones necesarias para el entrenamiento.

Se modifica el archivo `pipeline.config`, en donde se establecen parámetros importantes para el entrenamiento como:

- número de clases: 10
- batch size: 16
- número de épocas: 75000
- modelo a utilizar: ssd_mobilenet_v2_fpn_keras
- directorio para el dataset de entrenamiento:

  /content/gdrive/MyDrive/customTF2/data/train.record
- directorio para el datase de prueba:

  /content/gdrive/MyDrive/customTF2/data/test.record

Cabe destacar que el número de clases, batch size, número de época se los definieron después de un proceso de pruebas y encontrando los valores más óptimos para el entrenamiento del modelo de detección de dígitos.

Finalmente, para la fase de entrenamiento primero se debe cargar el tensorboard con el modelo de entrenamiento seleccionado, dirigirse a la carpeta *object_detection* e instalar las librerías necesarias, el código para ejecutar lo anteriormente mencionado se muestra en el Código 19.

Código 19

Entrenamiento del modelo.

```
!python model_main_tf2.py—
pipeline_config_path=/content/gdrive/MyDrive/customTF2/data/ssd_mobilenet_v2
_fpnlite_640x640_coco17_tpu-8.config—model_dir=/mydrive/customTF2/training—
alsologtostderr
```

Nota. Se debe verificar que las versiones de las librerías sean compatibles con el modelo seleccionado.

Una vez verificado el modelo entrenado se exporta el detector en formato `.tflite`, necesario para cargarlo en la aplicación móvil, el código para esto se indica en el Código 20.

Código 20

Exportar el modelo entrenado en .tflite.

```
!tflite_convert—saved_model_dir=tflite/saved_model—
output_file=tflite/detect.tflite
```

Nota. Existe dos formas de exportar el archivo .tflite, el primero es un modelo completo, por tanto, es más pesado, otra forma de exportar este archivo en el mismo formato es más liviana, sin embargo, pierde la eficiencia.

Diseño de la aplicación móvil Android

La aplicación móvil desarrollada se constituye de varias interfaces a las cuales el usuario accede para realizar la lectura de consumo de medidores de agua potable y de energía eléctrica. La aplicación está diseñada y desarrollada en el IDE de Android Studio 4.2.2. con

versión mínima Android 6.0. Marshmallow. El diagrama de flujo del diseño de la aplicación móvil se presenta en la Figura 23 y Figura 24.

Para comprender con más claridad el funcionamiento de la aplicación móvil desarrollada, a continuación, se define las principales características y funcionalidades de la misma.

- La aplicación dispone de una primera interfaz de ingreso, en ésta se ingresan las credenciales registradas en la base de datos para acceder a las demás pestañas y funcionalidades de la aplicación.
- Una vez realizado el ingreso correctamente, se presenta 3 opciones: Principal, (+) Botón para realizar la lectura, e Información,
- En la opción Principal, se muestran los datos del recolector como su nombre, apellido y cédula.
- El botón con el símbolo (+), direcciona a otra interfaz para elegir el cliente, al que se desea realizar la lectura, y el método de obtención de la fotografía del medidor.
- En la opción de Información, se presenta detalles de la aplicación y de cómo usarla.
- La interfaz direccionada con el botón (+), consta de un campo donde se debe digitar el número de cédula del cliente y si está registrada en la base de datos, se despliegan dos opciones para elegir el método de ingreso de la fotografía. Estas dos opciones son: CAPTURAR y SELECCIONAR.
- La opción CAPTURAR muestra la interfaz de cámara del celular, en la cual se realiza la fotografía del medidor, ya sea de agua o de electricidad.

- La opción SELECCIONAR muestra el administrador de imágenes del dispositivo móvil, desde aquí se accede a los diferentes directorios del almacenamiento del dispositivo para seleccionar una fotografía.
- Una vez seleccionada la fotografía de un medidor, en una última interfaz se presenta la clasificación de la misma, es decir, si es un medidor de agua o de energía eléctrica, además de la lectura. Así mismo, se integra un botón de ENVIAR para remitir los resultados a la base de datos.

Figura 23

Diagrama de flujo del diseño de la aplicación móvil (Primera parte).

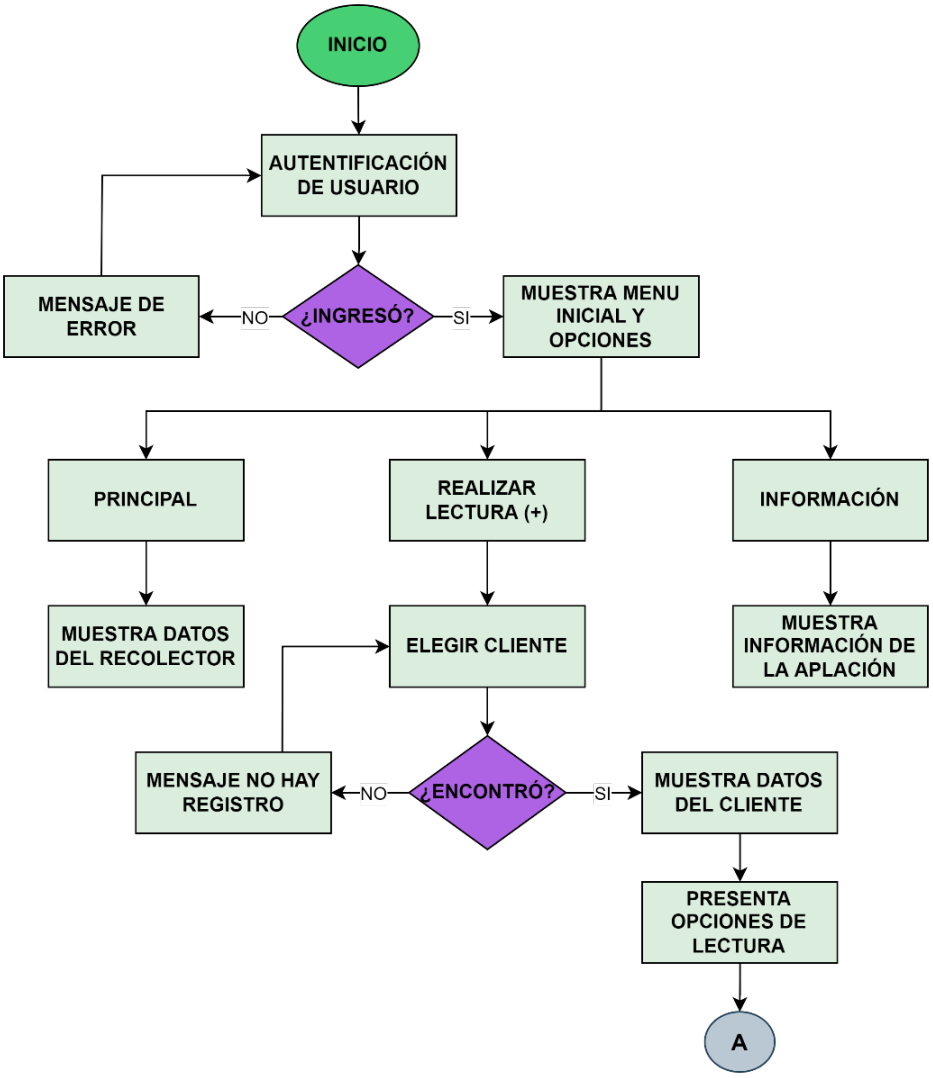
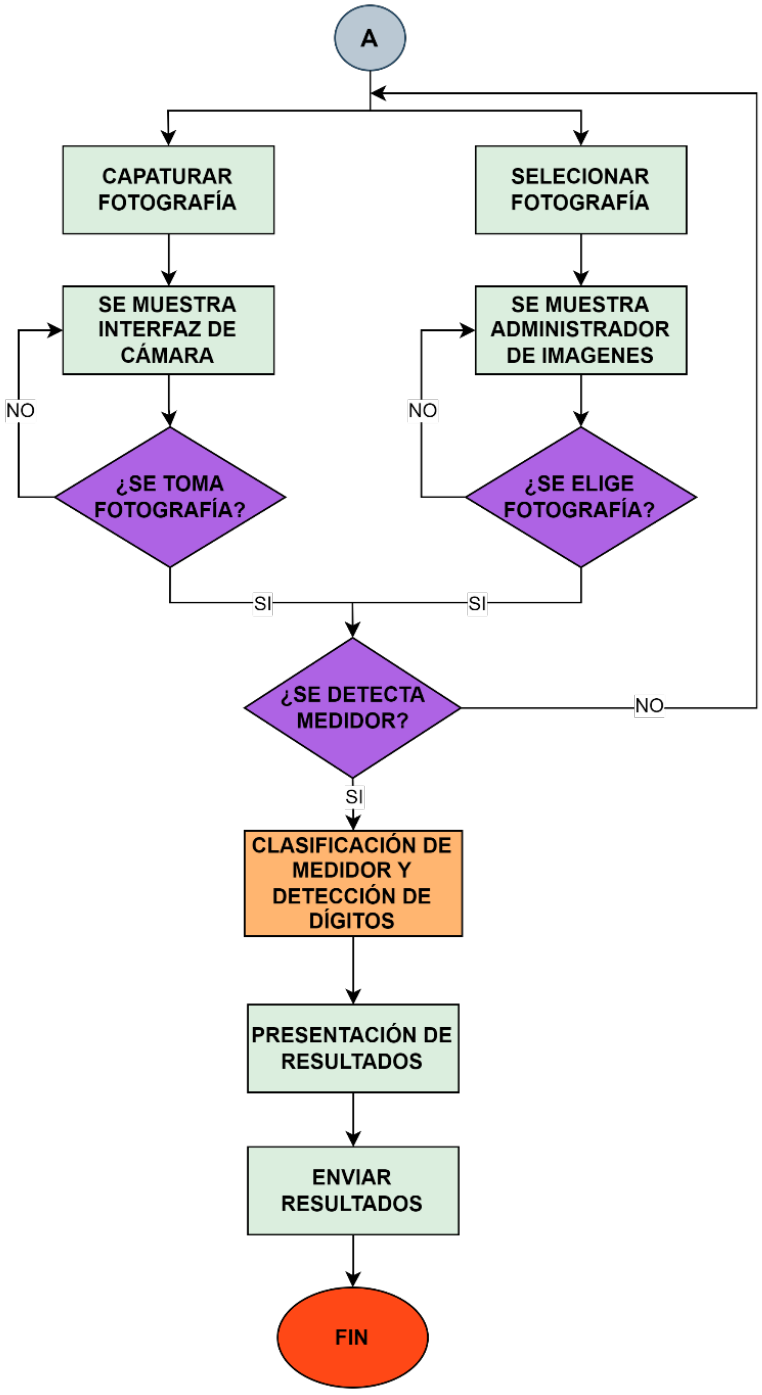


Figura 24

Diagrama de flujo del diseño de la aplicación móvil (Segunda parte).



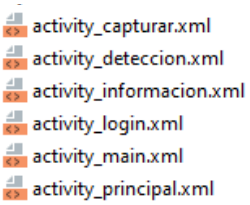
Nota. Las figuras presentan el diagrama de flujo del funcionamiento y diseño general de la aplicación móvil desarrollada.

Estructuración de la aplicación móvil

El diseño de las diferentes interfaces que constituyen la aplicación móvil, se desarrolla con *layouts* escritos en archivos xml, en estos se codifican la presentación y ubicación de los elementos como: botones, campos de texto, etc. Cada *layout* creado está vinculado a una clase tipo Java la cual le da funcionalidad. La aplicación móvil dispone de los *layouts* que se muestran en la Figura 25.

Figura 25

Vista de layouts creados para el diseño de la aplicación móvil.

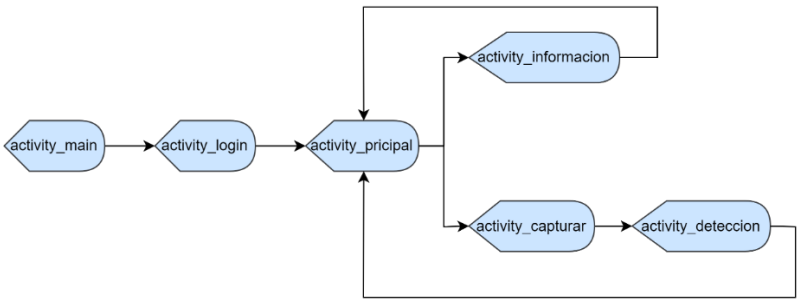


Nota. La figura presenta los diferentes *layouts* que se crearon para dar diseño a las diferentes interfaces que contiene la aplicación móvil.

La presentación de los *layouts* creados siguen un cierto orden, es decir, cierto *layout* se presenta antes que otros. La Figura 26 muestra el orden de presentación de los diferentes *layouts*.

Figura 26

Disposición de los layouts o interfaces de la aplicación móvil.



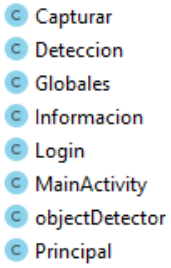
Nota. La figura presenta el orden en el que aparecen las interfaces de la aplicación móvil.

Diseño y programación de las interfaces de la aplicación móvil

Como se ha indicado anteriormente por cada *layout* creado se genera una clase Java para dar funcionalidad a la interfaz, además de realizar las transiciones por los demás *layouts*. Las clases que dan funcionalidad a la aplicación se presentan en la Figura 27.

Figura 27

Vista de las clases creadas para la aplicación móvil.



Nota. La figura muestra todas las clases creadas para dar funcionalidad a la aplicación, la mayoría de ellas corresponde a una sola interfaz creada.

Cabe destacar que existen dos clases adicionales como Globales y objectDetector, creadas para ofrecer ciertas funcionalidades extras a algunas clases.

Interfaz Capturar

La clase que le da la funcionalidad a esta interfaz se denomina Capturar, desde aquí se realiza la consulta de los datos del cliente, como su nombre, apellido y dirección. Si el cliente se encuentra en la base de datos se habilitan las dos opciones para obtener una fotografía del medidor. Este proceso se implementa en el método `buscarCliente` de la clase `Capturar`, como se muestra en el Código 21.

Código 21

Método BuscarCliente().

```
    cedula = binding.buscarCliente.getText().toString();
    if (cedula.isEmpty()) {
        Toast.makeText(Capturar.this, "¡Ingrese un número de cédula, por
favor!", Toast.LENGTH_SHORT).show();
    } else {

db.collection("Clientes").document(cedula).get().addOnSuccessListener(docume
ntSnapshot -> {
            if (documentSnapshot.exists()) {
                // DATOS
                String nombre = documentSnapshot.getString("nombre");
                String apellido = documentSnapshot.getString("apellido");
                String direccion = documentSnapshot.getString("direccion");

                binding.datosCliente.setText("Nombre: " + nombre + "\n" +
"Apellido: " + apellido + "\n" + "Dirección: " + direccion); // Presenta
datos en interfaz
```

Nota. El código muestra las líneas utilizadas para implementar el método BuscarCliente() que permite realizar una consulta a la base de datos Clould Firestore.

Después de haber elegido al cliente, como primer paso se pulsa del botón Capturar y se ejecuta el método `takePhoto()` que se presenta en el Código 22.

Con la ejecución de dicho método se habilita la interfaz de la cámara con la variable de tipo *Intent* que tiene como actividad `ACTION_IMAGE_CAPTURE`, tal como se muestra en la Figura 28.

Código 22

Método takePhoto().

```
    File photoFile = null;
    try {
        photoFile = createImageFile();
    } catch (IOException ex) {

    }
    if (photoFile != null) {
        Uri photoURI = FileProvider.getUriForFile(this,
                "com.example.android.appva.fileprovider",
                photoFile);
        takePictureIntent.putExtra(MediaStore.EXTRA_OUTPUT, photoURI);
        startActivityForResult(takePictureIntent, REQUEST_TAKE_PHOTO);
    }
```

Nota. El código muestra las líneas del método takePhoto() con el cual se realizará la captura de fotografías desde el celular.

Figura 28

Vista de la interfaz de cámara para capturar.

Nota. La figura presenta la interfaz de cámara del celular, desde la cual se realiza la captura de fotografías de los medidores.

Una vez tomada la fotografía, la misma interfaz permite aceptar o descartar la fotografía y volver a hacer otra, como muestra la Figura 29. Elegida la fotografía, ésta se guarda en cache para ser utilizada más adelante.

Figura 29

Vista de interfaz de cámara para seleccionar o descartar fotografía.

Nota. La figura muestra la interfaz desde la cual se aceptará o se descartará la fotografía captura.

Para el segundo caso si se pulsa el botón de Seleccionar, se ejecuta el método `SelecionarFoto()` el cual mediante una variable *Intent* abre la interfaz de administrador de imágenes del dispositivo móvil. Este método se presenta en el Código 23.

Código 23

Método `SeleccionarFoto().`

```
static final int SELECT_PICTURE=2;
private void SeleccionarFoto() {
    Intent i = new Intent();
    i.setType("image/*");
    i.setAction(Intent.ACTION_GET_CONTENT);

    startActivityForResult(i, SELECT_PICTURE);
}
```

Nota. El código muestra las líneas del método `SeleccionarFoto()`, el cual permite acceder al administrados de imágenes del celular para seleccionar una fotografía de un médidor.

Desde el administrador de imágenes se puede seleccionar una fotografía de un medidor que previamente se haya capturado, tal como se muestra en la Figura 30.

Figura 30

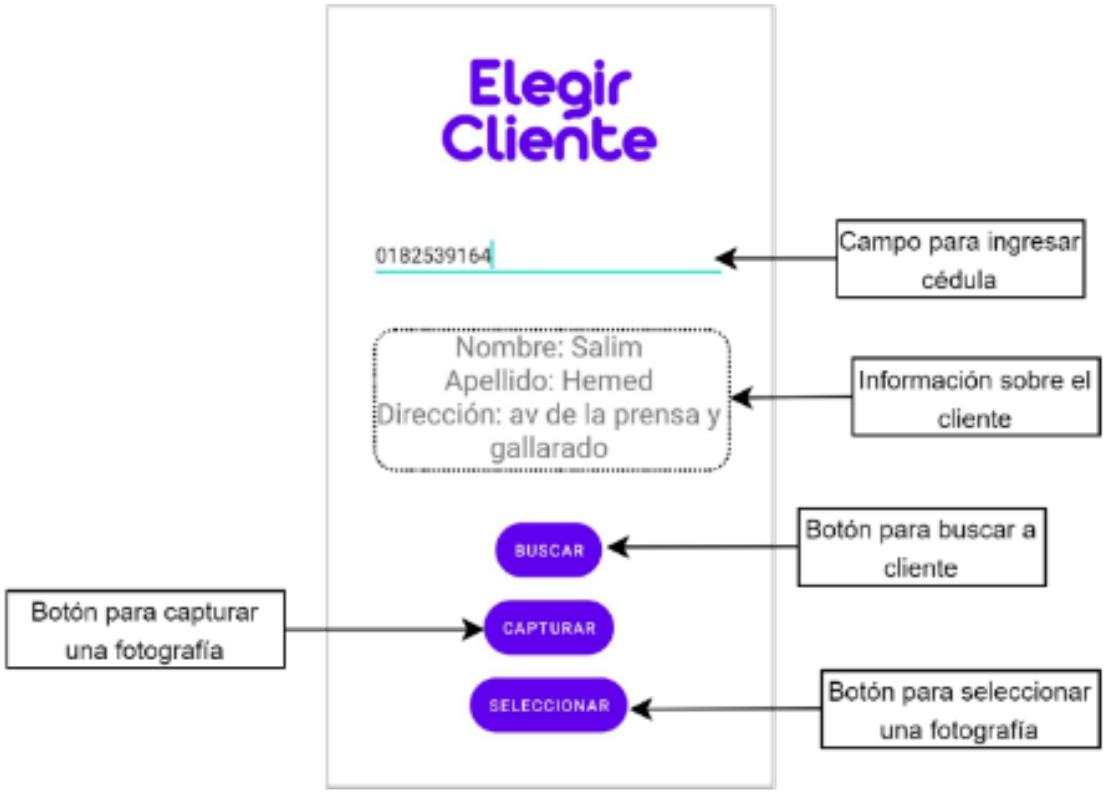
Vista de administrador de imágenes del celular.

Nota. La figura muestra el administrador de imágenes que se abre al ejecutarse el método `SelecionarFoto().`

La interfaz captura se presenta como la Figura 31 con todos sus elementos.

Figura 31

Vista de interfaz Capturar en aplicación móvil



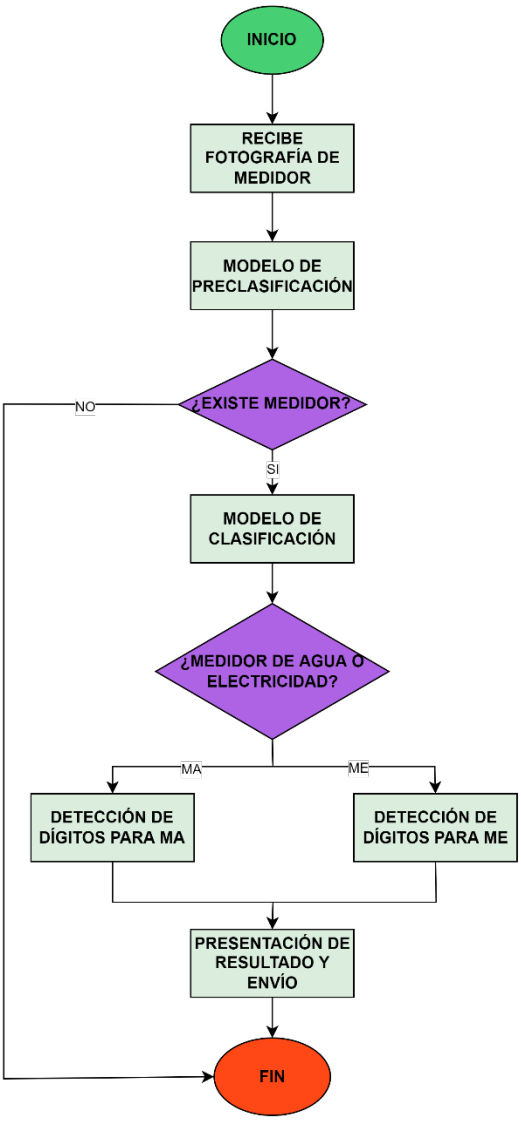
Nota. La figura presenta la interfaz Capturar con la aplicación móvil ejecutándose, además se presenta los elementos que la componen.

Interfaz Detección

El diagrama de flujo que representa la programación desarrollada para la clase de esta interfaz se muestra en la Figura 32.

Figura 32

Diagrama de flujo de clase Detección.



Nota. La figura muestra el diagrama de flujo que describe el funcionamiento de la clase Deteccion.

La clase que da funcionalidad de esta interfaz se denomina Deteccion, en esta clase se implementa el modelo de TensorFlow Lite de clasificación de medidores y el de detector de dígitos, además de otras funcionalidades. Cabe destacar que esta clase recibe la fotografía del medidor desde la anterior interfaz y la guarda en una variable de tipo Bitmap que permite

almacenar los valores de los pixeles en una matriz de bits. El Código 24 muestra la codificación correspondiente al diagrama de flujo presentado anteriormente.

Código 24

Clase Deteccion.

```
        preclasificador(imgBitmap);
        if (ExiteMedidor == true){
            clasificador(imgBitmap);
            if (clasificacion.equals("Medidor de Agua")) {
                modelPath = "detect_model_MA.tflite";
                binding.Unidades.setText("m3");
                aux_clasificacion = "1";
            } else if (clasificacion.equals("Medidor de
Electricidad")) {
                modelPath = "detect_model_ME.tflite";
                binding.Unidades.setText("kWh");
                aux_clasificacion = "2";
            }
        }
```

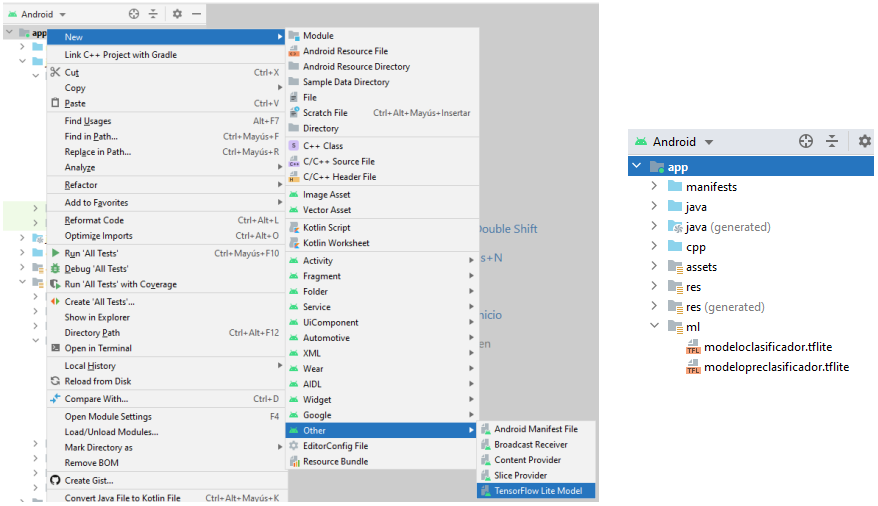
Nota. La figura muestra las líneas principales de la clase Deteccion, desde las cuales se ejecutan otros métodos.

La variable Bitmap que almacena la fotografía se envía al método `preclasificador()`, este método implementa el modelo de preclasificación, el mismo que se encarga de identificar que exista un medidor, ya sea de agua o de energía eléctrica. Para implementar y hacer uso del modelo en la aplicación móvil, previamente se debe obtener el modelo de clasificación en formato .tflite admitido por el entorno de desarrollo Android Studio.

El modelo se añade al directorio *app* y en el apartado *Other* se presentan varios tipos archivos y entre ellos está *TensorFlow Lite Model*, al añadir el archivo se guarda en un subdirectorio ml, este proceso se muestra en la Figura 33.

Figura 33

Implementar un modelo Tensorflow desde Android Studio



Nota. La figura presenta la manera cómo se implementa un modelo de clasificación en formato TensoFlow Lite sobre una aplicación desarrollada en Android Studio.

El método `preclasificador()` también realiza el procesamiento de la fotografía a fin de adaptarla para el modelo de preclasificación, tal como muestra el Código 25. Este procesamiento se basa en redimensionar la fotografía a 224x224 pixeles, además de normalizar cada uno de los valores de los pixeles a un rango de 0 a 1, la normalización se logra al dividir cada pixel por 255 (valor máximo de un pixel). La fotografía procesada se guarda en la variable byteBuffer y posteriormente se carga al modelo de preclasificación por medio de TensorBuffer. Como resultado de ejecutar el modelo de preclasificación se tiene dos resultados de predicciones en forma de porcentaje, seguidamente se determina cuál es el porcentaje más alto de predicción y según esto se determina a qué clase corresponde, para este caso de preclasificación existe dos clases como resultado: Existe medidor y No existe medidor.

Código 25

Método preclasificador()

```
        String[] classes = { "Se ha detectado medidor" ,"No se ha
detectado medidor"};

        if (maxPos == 0){   // Existe medidor
            ExiteMedidor = true;
            binding.btnEnviarLectura.setVisibility(View.VISIBLE);
        } else if (maxPos == 1){    // No existe medidor
            binding.rslClasificacion.setText(classes[maxPos]);
            binding.btnEnviarLectura.setVisibility(View.INVISIBLE);
            Toast.makeText(Deteccion.this, "Capture nuevamente, para
una nueva detección", Toast.LENGTH_LONG).show();
        }
```

Nota. El código muestra las líneas utilizadas en el método `preclasificador()`, con la ejecución de este método se verifica si existe un medidor en la fotografía.

Si existe medidor se envía la misma fotografía al método `clasificador()`, para que en esta ocasión se determine si se trata de un medidor de agua potable o de energía eléctrica. El método `clasificador()` sigue el mismo proceso que el de preclasifiación, con la diferencia en que ahora las clases a predecir son: Medidor de Agua y Medidor de Electricidad, tal como se presenta en el Código 26.

Código 26

Método clasificador()

```
        String[] classes = {"Medidor de Agua", "Medidor de
Electricidad"};
        clasificacion = classes[maxPos];
        binding.rslClasificacion.setText(classes[maxPos]);
        ExiteMedidor = false;
```

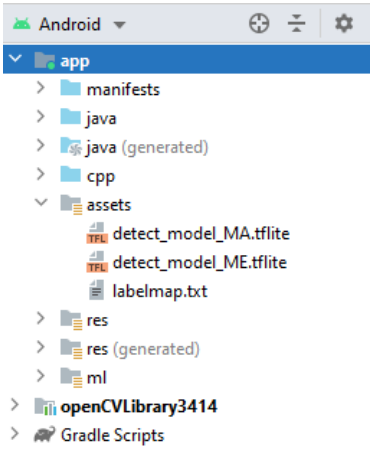
Nota. El código muestra las líneas utilizadas en el método clasificador, con la ejecución de este método se verifica qué tipo de medidor existe en la fotografía.

Una vez identificado de qué tipo de medidor se trata, se procede a cargar el modelo de detección de dígitos según corresponda. Se tiene dos modelos de detección, ya que los medidores de agua y de energía eléctrica tienen diferentes tipos de números, para el caso de medidores de energía eléctrica se tiene números digitales y para medidor de agua no son

digitales. Los dos modelos de detección se cargan en la carpeta *assets* del proyecto, tal como se muestra en la Figura 34.

Figura 34

Vista de los modelos de detección de dígitos



Nota. La figura muestra los modelos de detección de dígitos importados en el proyecto de la aplicación móvil.

Para extraer los dígitos del medidor de la fotografía, se envía al método `DigitDetection()` la misma fotografía que se utilizó para la clasificación, dicho método se presenta en el Código 27. En este método se instancia una clase denominada *objectDetector* donde se realiza el proceso para extraer lo dígitos y además retorna la imagen con los dígitos encerrados en rectángulos verdes.

Código 27

Método DigitDetection()

```
private void DigitDetection(Bitmap imageBitmap){
    Mat selected_image = new Mat(imageBitmap.getHeight(),
imageBitmap.getWidth(), CvType.CV_8UC4);
    Utils.bitmapToMat(imageBitmap, selected_image);
    selected_image = objectDetector.recognizePhoto(selected_image);

    Bitmap imgDetectBitmap =
Bitmap.createBitmap(selected_image.cols(), selected_image.rows(),
Bitmap.Config.RGB_565);      // Para imagen de salida Bitmap
    Utils.matToBitmap(selected_image, imgDetectBitmap);
    binding.imgConDeteccion.setImageBitmap(imgDetectBitmap);
```

La imagen que retorna la clase *objectDetector* se muestra en la Figura 35, en ésta se enmarca los dígitos hallados gracias al detector.

Figura 35

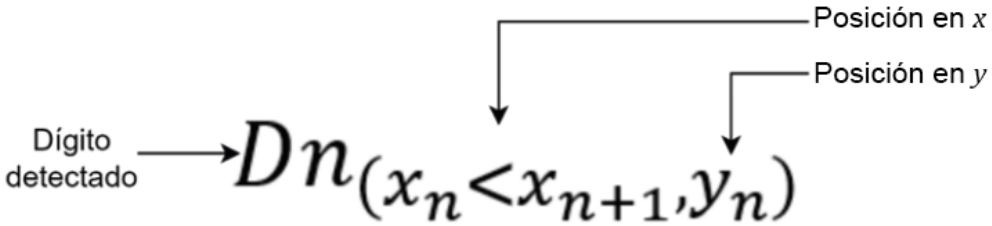
Detección de dígitos en medidor de agua

Nota. La figura muestra la detección de dígitos efectuada en una fotografía de un medidor de agua potable.

Los dígitos detectados se guardan en un arreglo según su localización en la imagen. Para comprender el orden en el que se guardan cada uno de los dígitos se debe considerar a la imagen como un plano cartesiano (x, y), entonces los dígitos que se localicen más a la derecha de la imagen ocupan las primeras posiciones del arreglo. La Figura 36 hace una representación al arreglo que almacena los dígitos.

Figura 36

Arreglo de dígitos de detectados



$$D1_{(x_1<x_2,y_1)} \mid D2_{(x_2<x_3,y_2)} \mid D3_{(x_3<x_4,y_3)} \mid D4_{(x_4<x_5,y_4)} \mid D5_{(x_5<x_6,y_5)} \mid D6_{(x_6<x_7,y_6)} \mid \dots$$

Los resultados de la clasificación y la lectura obtenida por medio de la detección de dígitos se envían a la base de datos conjuntamente con la hora, día y mes en el que se realiza la lectura, de esta manera se puede estructurar las lecturas realizadas de los medidores por tipo de medidor y fecha. El método que realiza este proceso se presenta en el Código 28.

Código 28

Método EnviarLectura()

```
    Map<String, Object> map = new HashMap<>();
    map.put("dia", day);
    map.put("hora", currentTime);
    map.put("lectura", lectura);

    db.collection("Clientes").document(cedula)
            .collection(aux_clasificacion).document(String.valueOf(year))

.collection(mes.substring(0,1).toUpperCase()+mes.substring(1)).document(mes)
            .set(map).addOnSuccessListener(new OnSuccessListener<Void>(
```

Nota. El código muestra las líneas que componen el método `EnviarLectura()`, este método permite enviar la información de clasificación y lectura realizada a la base de datos de Could Firestore.

Finalmente, con la clasificación del medidor obtenida y lectura de los dígitos, se presentan los resultados en la interfaz tal como muestra la Figura 37.

Figura 37

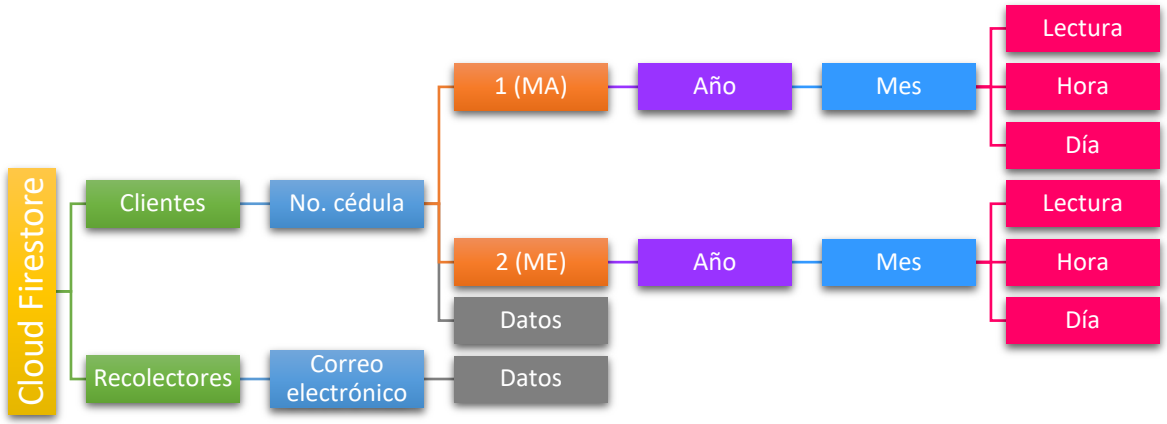
Vista de interfaz detección en aplicación móvil

Implementación de la base datos en Could Firestore

La base de datos en Could Firestore se estructura como muestra la Figura 38, desde la aplicación móvil y web se pueden publicar datos y hacer consultas. La base de datos tiene dos principales ramales o colecciones, uno para clientes donde se registran sus datos y las lecturas de sus consumos mensuales, y el segundo para recolectores quienes son los encargados de realizar las lecturas de los medidores.

Figura 38

Estructura de base de datos en Could Firestore



Nota. La figura presenta la estructura general implementada en la base de datos Could Firestore para almacenar los datos de clientes y sus lecturas realizadas mensualmente, así como información de los recolectores.

El despliegue de la rama de Clientes se forma de la siguiente manera:

- Los clientes se registran por su número de cédula, es decir, que en la rama de Clientes existen tantos números de cédulas como de clientes.
- Para cada cliente, o en este caso número de cédula se despliega dos ramas, nombradas 1 para Medidor de Agua y 2 para Medidor de Electricidad. Se sigue este tipo de identificación, ya que de esta manera resulta más sencilla realizar la consulta o publicación de datos.

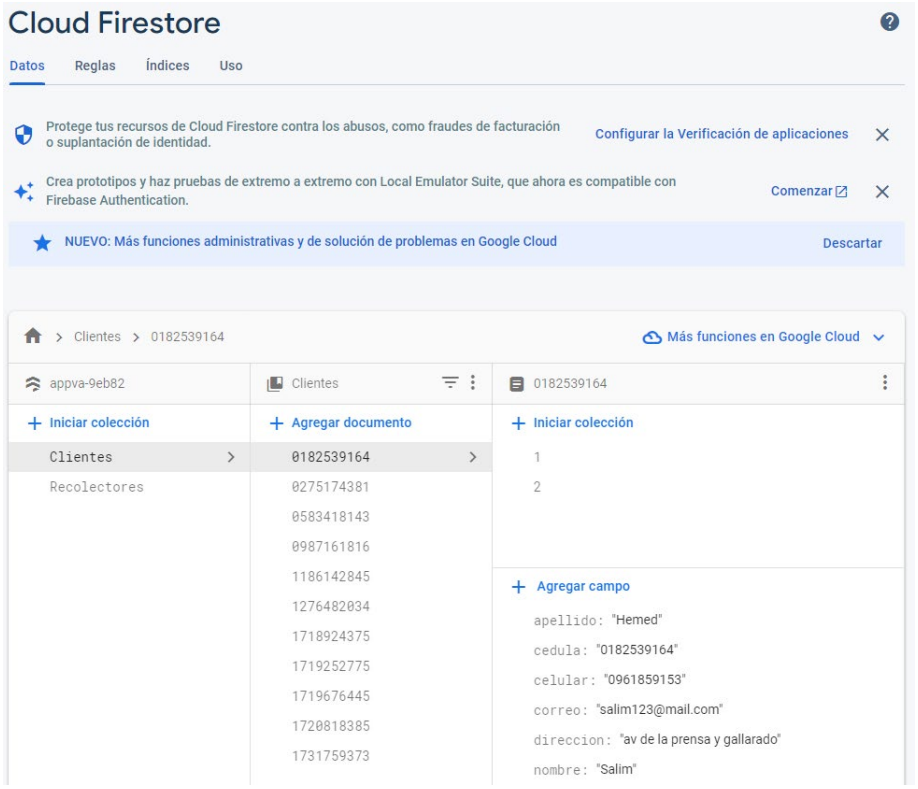
- Adicionalmente para cada número de cédula, se registran los datos del cliente como: nombre, apellido, cédula, celular, dirección y correo electrónico.

- Como se puede apreciar en la Figura 38, para cada ramal 1 (MA) y 2 (ME) se extiende otro ramal para el año y existen tanto ramales como lecturas registradas de anteriores años. Y de este ramal de año se expande por meses.
- Finalmente, por cada mes que se realice la lectura se registran en Could Firestore la lectura realizada, la hora y el día en que se la realizó.

En la consola de Firebase, la rama de Clientes implementada se aprecia como muestra la Figura 39.

Figura 39

Muestra de rama Cliente en Could Firestore



El despliegue de la rama de Recolectores, es mucho más sencilla y se forma de la siguiente manera:

- Los recolectores se registran por el correo electrónico y al igual que en el caso de clientes, existen tantos ramales de correos electrónicos como recolectores registrados.
- Finalmente, para cada recolector se registran sus datos como: nombre, apellido y número de cédula.

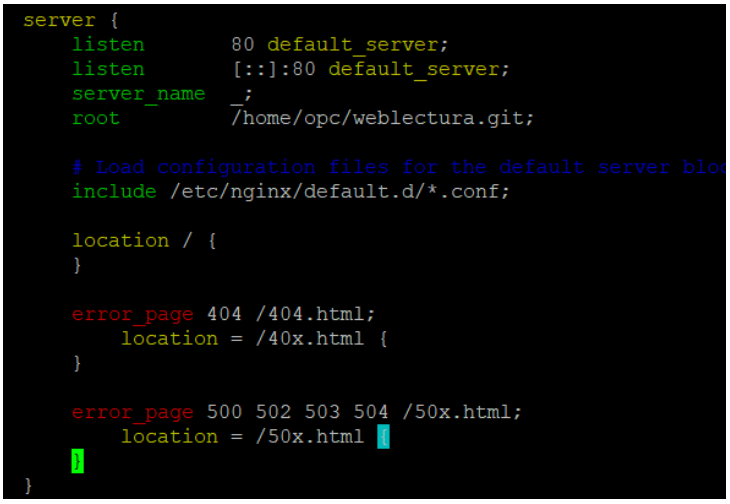
Implementación de la página web

Para la implementación de la página web se utiliza a Nginx, este es un servidor web y proxy inverso de código abierto y con alto rendimiento, que también incluye servicios de correo electrónico con acceso a (IMAP, del inglés *Internet Message Protocol*) y al servidor (POP, del inglés *Post Office Protocol*) y otros servicios (Acens, 2019). Nginx está diseñado para ofrecer un bajo uso de memoria y alta concurrencia, es decir, permite múltiples conexiones en simultaneo.

Se realiza una configuración adicional del host con la edición del archivo de configuración /etc/Nginx/Nginx.conf, esto se muestra en la Figura 40.

Figura 40

Archivo de configuración nginx.conf



Con estas configuraciones ya se puede ingresar a la página de bienvenida de Nginx, con la URL https://localhost, esto se muestra en la Figura 41.

Figura 41

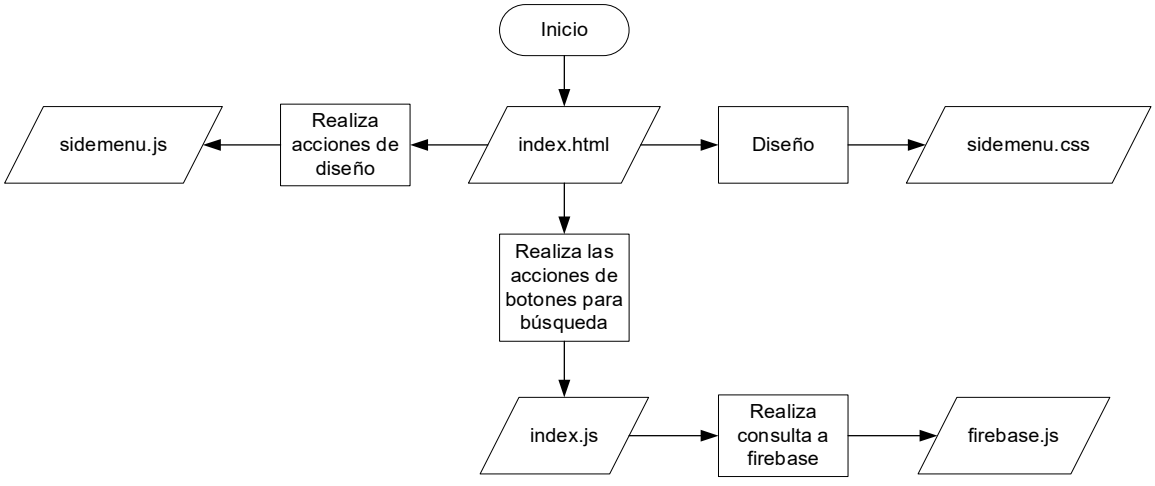
Página de bienvenida de Nginx

Nota. En esta captura se muestra la ventana de bienvenida de Nginx al ingresar al navegador con la IP 192.168.43.31 que es la dirección en dónde está el servidor local, después se subió este servidor en una máquina con una ip pública.

Una vez instalado e inicializado el servidor web, ya se puede diseñar la funcionalidad de la página, para ello se sigue el diagrama de flujo que se presenta en la Figura 42.

Figura 42

Diagrama de flujo de la página web



Nota. En este diagrama de flujo se muestra la relación entre las diferentes tecnologías utilizadas como: JS, HTML y CSS.

Con base al diagrama de flujo presentado, se muestra parte del código en el Código 29, del *head* y parte del *body* del archivo index.html en dónde se llama a los *script* sidemenu.js y index.js en donde se programa la funcionalidad de la página y al *script* sidemenu.css en donde se detallan estilos, fuentes y colores.

Código 29

Archivo index.html

```
<head>
    <meta charset=”UTF-8”>
    <meta http-equiv=”X-UA-Compatible” content=”IE=edge”>
    <meta name=”viewport” content=”width=device-width, initial-scale=1.0”>
    <title>AppVA</title>
    <link rel=”icon” href=”/logo.jpg”>
    <link rel=”stylesheet” href=”/Vista/Css/sidemenu.css”>
    <script
src=”https://cdnjs.cloudflare.com/ajax/libs/Chart.js/3.8.0/chart.js”></script>
</head>
```

Otra función importante dentro del *script* es la que permite calcular el valor a cancelar después de la obtención del consumo de agua potable y enérgía eléctrica, los valores de referencia fueron obtenidos de la Junta de Agua Potable, el código que permite esto se muestra en el Código 30.

Tanto la búsqueda de usuarios como sus respectivos consumos se obtienen de la base de datos que se describe anteriormente, el valor a pagar por el consumo puede ser ajustable a corde las regimenes de las empresas recaudadoras, para añadir esta funcionalidad en la página web es necesario el *script firebase.js*, los parámetros de la base se indican dentro de la pestaña de Configuración de proyecto de la consola en donde se levanta Firebase, esto se muestra en la Figura 43.

Código 30

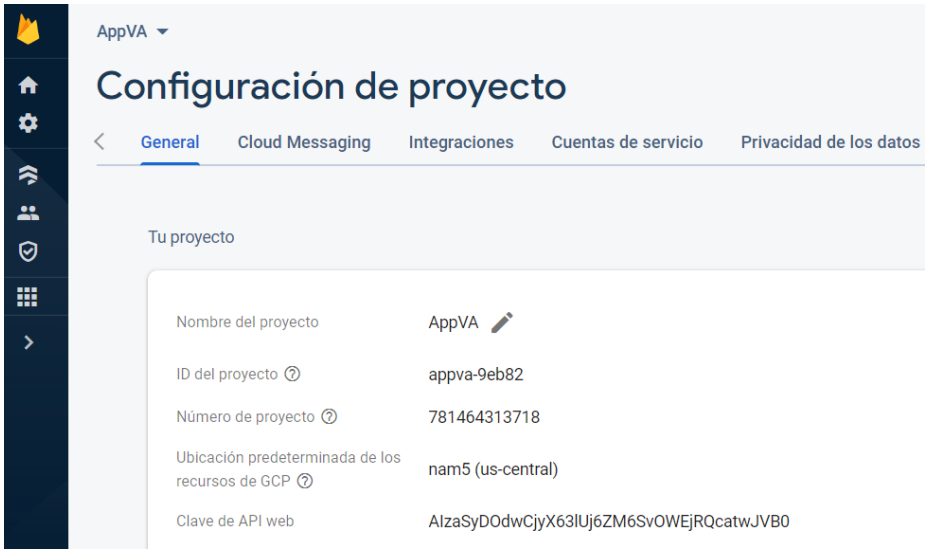
Calcular el valor a pagar.

```
            if (radio == 2) {
```

```
            subtotalConversion = consumoMensual * 0.092
            unidadMedida = “kW/h”
        }else {
            if ((consumoMensual) > 0 && (consumoMensual) < 21) {
                subtotalConversion = consumoMensual * 0.41
            } else if ((consumoMensual) >= 21 && (consumoMensual) < 26) {
                subtotalConversion = consumoMensual * 0.62
            } else if ((consumoMensual) >= 26 && (consumoMensual) < 41) {
                subtotalConversion = consumoMensual * 0.67
            } else if ((consumoMensual) >= 41) {
                subtotalConversion = consumoMensual * 0.72
            }
            unidadMedida = “m3”
        }
        totalAPagar = subtotalConversion + 3.50
```

Figura 43

Parámetros para el anclaje de la base de datos con la página web



Nota. Estos datos son necesarios tanto para las consultas desde la página web como para las consultas que se relizan desde la aplicación móvil.

En el Código 31 se muestra como se importa e inicializa las librerías de Firestore.

Código 31

Anclar la base datos con la aplicación web

```
const firebaseConfig = {
    apiKey: “xxxx”,
    authDomain: “xxxx”,
    projectId: “xxxx”,
    storageBucket: “xxxx”,
    messagingSenderId: “xxxx”,
    appId: “xxxx”
};
```

Nota. Existen varios métodos que permite anclar la base de datos con la página web, como npm, CDN y Config, para este proyecto se utilizó la URL.

Con todos los componentes desarrollados para la página web se compila el proyecto y el resultado fue el que se muestra en la Figura 44, como página de inicio.

Figura 44

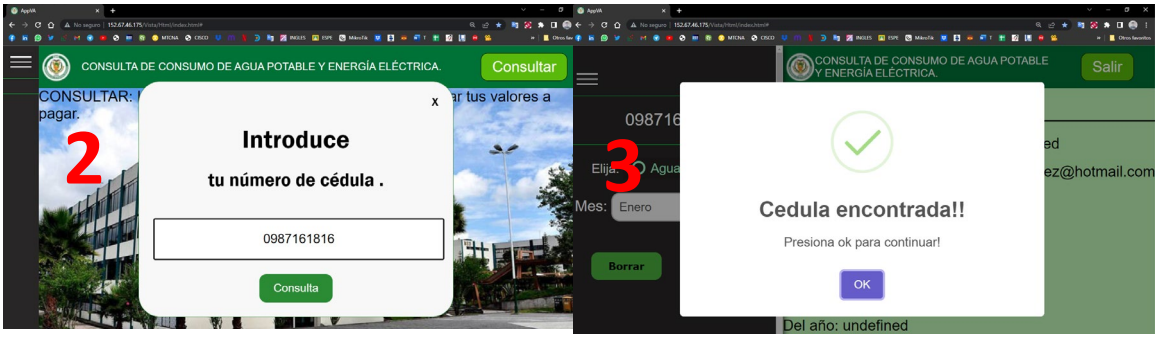
Ventana principal de la página web

Nota. En la figura se indica el botón que permite abrir la ventana en donde se ingresa el número de cédula de un cliente.

Una vez que se inserte un número de cédula correcto en la interfaz 2 que se muestra en la Figura 45, se despliega una ventana emergente que indica que el número de cédula fue encontrado esto se indica con en la interfaz 3 en la Figura 45.

Figura 45

Ventanas emergentes en la página web

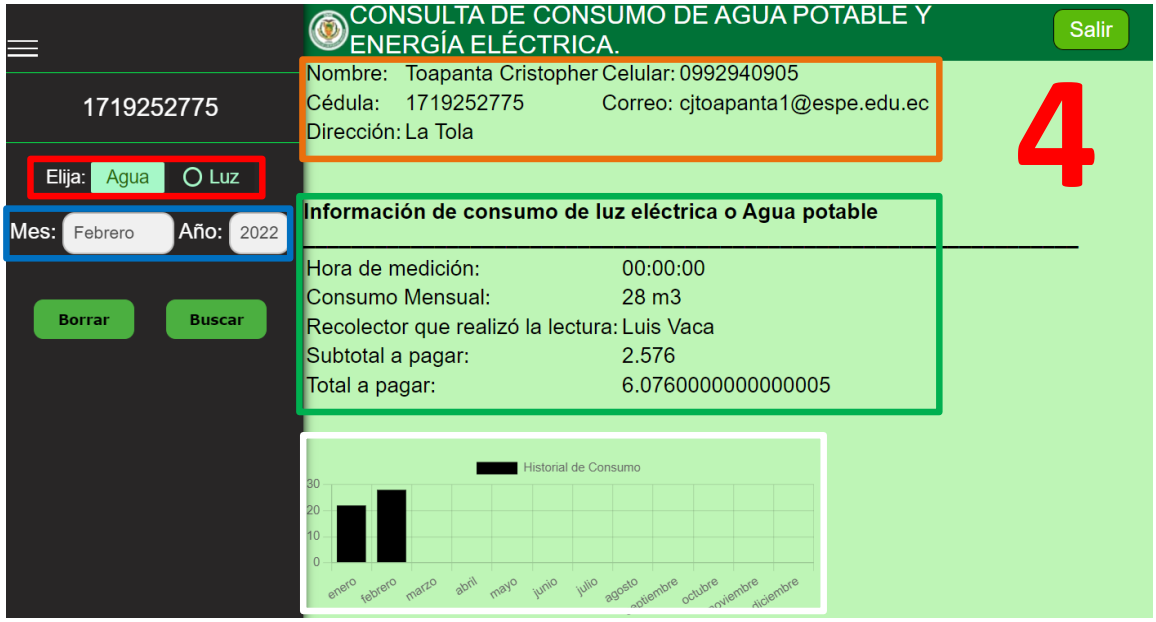


Nota. En el caso que se escriba un número de cédula incorrecto también se despliega una ventana emergente que indica al usuario que no está registrado.

Después de ingresar un número de cédula de un usuario, se muestra la ventana de la Figura 46, en donde se indican con un rectángulo naranja los datos del usuario, en la parte izquierda de la ventana con un rectángulo rojo se muestran las opciones de lectura que se desea consultar, es decir, si elije consultar el consumo de agua potable o energía eléctrica, de igual manera con un rectángulo azul se indican las opciones de consulta; se puede elegir el mes y el año. Con las opciones escogidas por el usuario en un rectángulo verde se muestran la información de consumo, datos como la hora de medición, recolector, consumo mensual, hora de medición y valor a pagar. Finalmente, en el rectángulo en blanco se muestra un historial de consumo de los meses anteriores.

Figura 46

Interfaz gráfica de ventana de informacion de consumo



Nota. Esta es la última ventana que se muestra en la página web, al presionar en el botón salir se despliega nuevamente la ventana principal.

Capítulo V

Pruebas y resultados

Para la evaluación del sistema implementado se planta 5 escenarios: tiempos de recolección en la ruta, validación del detector de dígitos, desempeño de la aplicación en diferentes dispositivos y parámetros de desempeño para aplicaciones de AV, en donde se compara la metodología actualmente utilizada para la recolección de lecturas, tanto en medidores de agua potable como de energía eléctrica, con la propuesta por el sistema desarrollado en este trabajo. También un escenario que muestra el desempeño de la aplicación en 2 diferentes tipos de dispositivos móviles, finalmente, los parámetros de desempeño comúnmente utilizamos en la AV.

Para iniciar con el detalle de cada escenario es primordial indicar que en el caso de la implementación de clasificadores y OCR en contadores de magnitudes físicas el universo es infinito, debido a que en cada casa del país mínimo se tiene un medidor de energía eléctrica y un medidor de agua potable. Por ello, se decide que la población a analizar sea el conjunto residencial La Colina, ubicado en la autopista Rumiñahui al frente de la Universidad de las Fuerzas Armadas “ESPE”, debido a que por parte de los administradores del conjunto residencial se brinda la autorización para tomar fotografías mientras dure el presente trabajo de titulación.

La fórmula para el cálculo de muestras para una población infinita y es la siguiente:

$$n = \frac{z^2 \times p \times q \times N}{(N-1) \times e^2 + z^2 \times p \times q}$$

En donde:

- z: nivel de confianza
- p: proporción de la población con la característica deseada

- q: proporción de la población sin la característica deseada
- e: error muestral máximo admitido
- N: tamaño de la población

Se define un nivel de confianza del 95% que determina un valor de z igual a 1.96, se especifica a p igual a 0.5 y por lo tanto q también es 0.5, esto debido a que los resultados son analizados según si la aplicación realiza bien la lectura o no. El error admitido en ingeniería es del 5%, finalmente, el conjunto residencial La Colina tiene 448 casas.

$$n = \frac{1.96^2 \times 0.5 \times 0.5 \times 448}{(448 - 1) \times 0.05^2 + 1.96^2 \times 0.5 \times 0.5}$$

$$n = 207.06$$

De acuerdo a la ecuación anterior se determina que el número de mediciones sea de 207 para medidores de agua potable y la misma cantidad para medidores de energía eléctrica.

Para comparar las mediciones entre la metodología actualmente utilizada por las empresas recolectoras de lecturas de energía eléctrica y agua potable; con el sistema implementado se realiza las mediciones una vez por mes durante tres meses, es decir, se necesita 69 casas dentro del conjunto con medidores de energía eléctrica y agua potable.

Con la muestra definida se determina una ruta que abarque la cantidad de casas necesarias, esta ruta se muestra en la Figura 47.

Primer escenario: tiempos de recolección en la ruta

Para el primer escenario se utiliza dos personas, la primera simula un recolector con la metodología comúnmente utilizada, es decir, se acerca al medidor del cliente, visualiza la lectura y la anota en su informe de recolección, y la otra persona sigue la misma ruta, pero utilizando la

aplicación creada en este trabajo. En la Tabla 12 se muestra los tiempos de recolección de lecturas obtenidos durante julio, agosto y septiembre.

Figura 47

Ruta de recolección de lecturas

Nota. Para simular una ruta de los recolectores se dio prioridad a que la misma sea continua, con el objetivo de no perder tiempo en el desplazamiento entre cada lectura.

Tabla 12

Tiempos de recolección de lecturas

Tiempo en ruta minutos: segundos	Persona 1 Metodología común	Persona 2 Aplicación
Julio	56:04	55:20
Agosto	54:37	54:58
Septiembre	53:12	52.19
Promedio	54:51	53:59

Nota. Los datos fueron obtenidos durante los 3 meses de recolección de lecturas siguiendo una misma ruta.

En la Tabla 12 se observa que la persona 1 se demora en promedio 54:51 minutos y la persona 2 tarda 53:59 minutos para completar la ruta que comprende 69 medidores de energía eléctrica y 69 medidores de agua potable. Sin embargo, es importante mencionar que en el primer caso cada lectura fue escrita manualmente, estos datos deben ser ingresados al sistema de facturación que tiene la empresa prestadora del servicio y para ello se necesita de 60 minutos adicionales por cada 69 lecturas. A diferencia del segundo caso, los datos recolectados ya son ingresados inmediatamente al sistema.

Segundo escenario: validación del detector de dígitos

En el segundo escenario se compara las lecturas realizadas por el recolector y las obtenidas con la aplicación. En el cálculo de la muestra se define que se necesita 207 lecturas de agua potable y 207 de energía eléctrica, en la Tabla 13 se muestra que se presenta una mayor cantidad de errores de lectura en medidores de agua potable, sin embargo, el porcentaje de error es 4% para medidores de energía eléctrica y 3% para medidores de agua potable, esto se muestra en la Figura 48.

Tabla 13

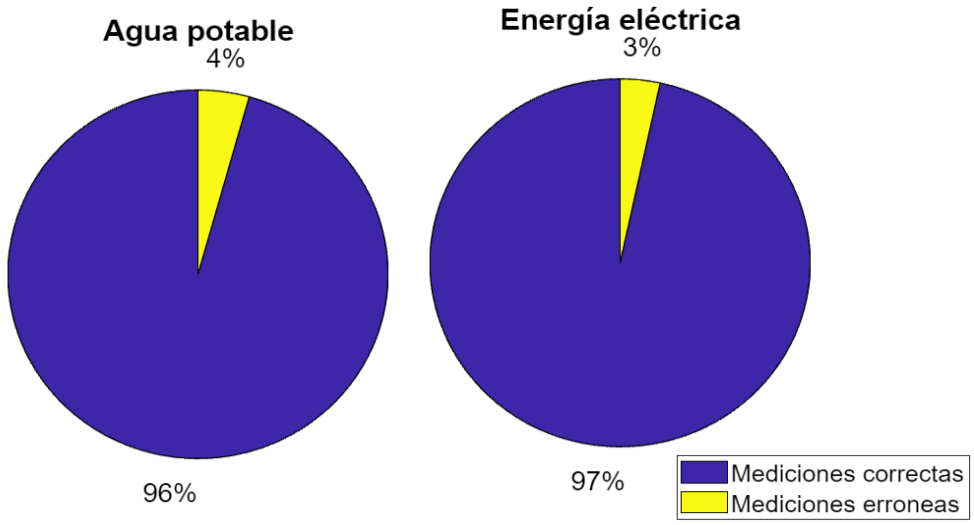
Errores en la lectura utilizando la aplicación

Errores	Agua potable	Energía eléctrica
Medidas correctas	198	200
Medidas erróneas	9	7

Nota. Se obtuvo una mayor cantidad de errores en los medidores de agua potable.

Figura 48

Porcentaje de error en las lecturas de los medidores de agua potable y energía eléctrica.



Nota. El porcentaje de error en las lecturas en medidores de energía eléctrica y agua potable están por debajo del 5% que es el margen aceptable en ingeniería.

Tercer escenario: desempeño de la aplicación en diferentes dispositivos

Para validar el desempeño de la aplicación se considera un dispositivo de gama media; Xiaomi Redmi Note 9 y un dispositivo móvil de gama baja; Huawei P9 Lite, en la Tabla 14 se muestra la comparación entre ambos equipos.

Ambos dispositivos móviles fueron puestos a prueba durante los 3 meses de recolección de lecturas con el objetivo de verificar los parámetros de desempeño mientras la aplicación se está ejecutando, en la Tabla 15 se muestra los parámetros más importantes para evidenciar el desempeño de la aplicación.

Tabla 14

Comparación entre los dispositivos móviles a prueba

Parámetros	Huawei P9 Lite	Xiaomi Redmi Note 9
Gama	baja	media
Pantalla	5.2”	6.53”
Procesador	Kirin 650 2GHz	Mediatek Helio G85 2GHz
RAM	2 GB	4 GB
Almacenamiento	16 GB	64 GB
Cámara	13 MP	48 MP
Batería	3000 mAh	5020 mAh
Sistema operativo	Android 6.0.1	Android 10

Nota. Se muestra las características más importantes de ambos dispositivos para diferenciar que el Xiaomi Redmi Note 9 tiene mayores prestaciones.

Tabla 15

Parámetros de desempeño en dos diferentes dispositivos móviles

Parámetros	Huawei P9 Lite	Xiaomi Redmi Note 9
Consumo de datos móviles	20 MB	25 MB
Consumo de la batería	95% - 80% (15%)	90% - 72% (18%)
Almacenamiento	120 MB	250 MB

Nota. Los dos dispositivos estuvieron a prueba durante la ruta de recolección de lecturas que en promedio dura 54 minutos.

Cuarto escenario: parámetros de desempeño para aplicaciones de AV

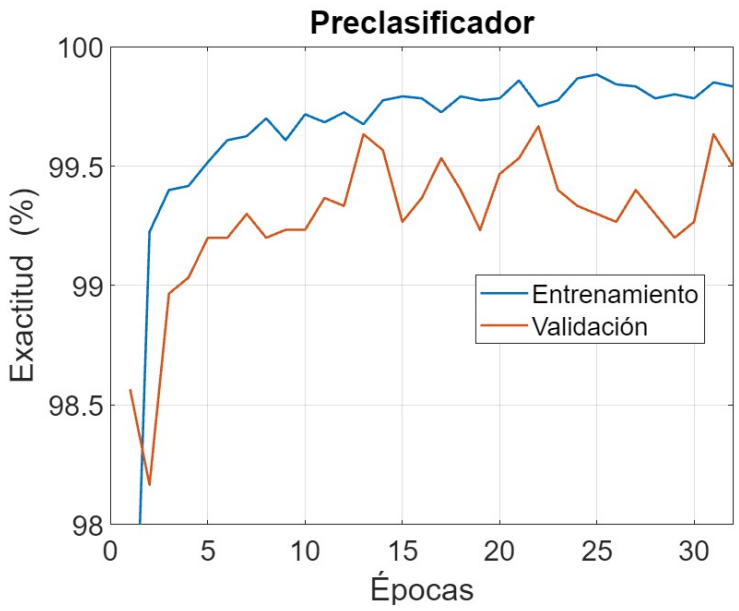
Parámetros de desempeño del preclasificador y clasificador de imágenes

Para ver las estadísticas del entrenamiento realizado tanto del preclasificador como del clasificador, se recurre a TensorBoard, el cual carga las gráficas de exactitud y pérdida.

En la Figura 49 y Figura 50 se muestra las estadísticas de exactitud del modelo entrenado del preclasificador y clasificador respectivamente, se aprecia que la exactitud de entrenamiento de ambos modelos es superior al 99%, con este valor se considera que el modelo de clasificación está apto para realizar predicciones buenas de las imágenes con las que fue entrenado.

Figura 49

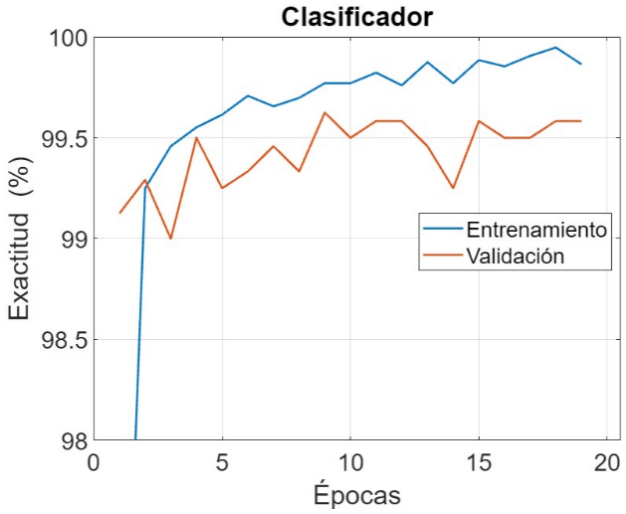
Gráfica de exactitud del preclasificador entrenado



Nota. La figura presenta las estadísticas de exactitud del modelo durante el entrenamiento del preclasificador.

Figura 50

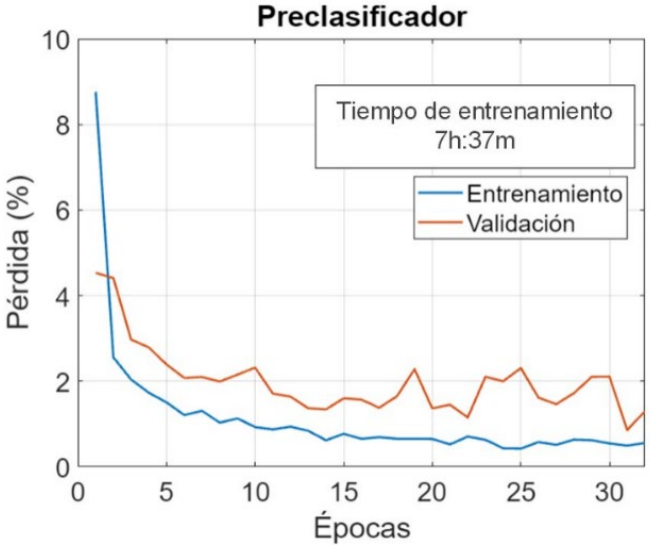
Gráfica de exactitud del clasificador entrenado



Nota. La figura presenta las estadísticas de exactitud del modelo durante el entrenamiento del clasificador.

Figura 51

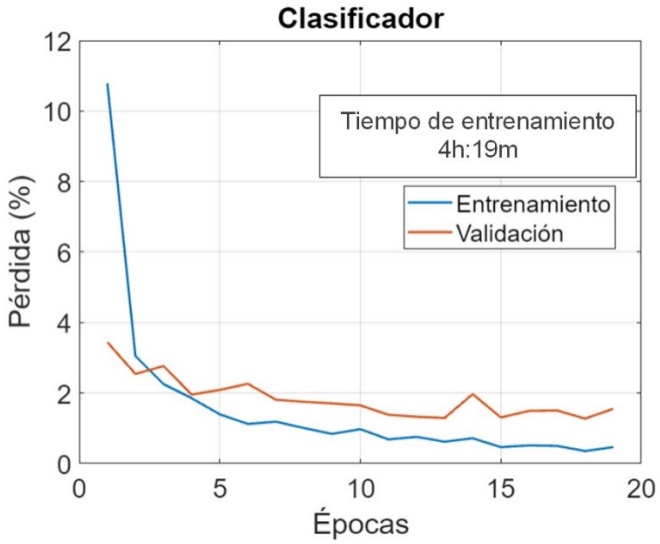
Gráfica de pérdida del preclasificador entrenado



Nota. La figura presenta las estadísticas de pérdida del modelo durante el entrenamiento del preclasificador.

Figura 52

Gráfica de pérdida del clasificador entrenado



Nota. La figura presenta las estadísticas de pérdida del modelo durante el entrenamiento del clasificador.

En la Figura 51 y Figura 52, se tiene las estadísticas sobre la pérdida del modelo entrenado del preclasificador y clasificador respectivamente, esto indica que tan mal se desempeñan los modelos por cada etapa de entrenamiento. Se muestra también el valor mínimo de pérdida de validación de ambos modelos que se fueron aproximadamente del 5%, mientras que el valor mínimo de perdida de entrenamiento fue de alrededor del 3%.

A continuación, también se presenta las gráficas de tiempo de entrenamiento requerido para el modelo de preclasificación y clasificación, en estas gráficas se aprecia cómo fue evolucionado los parámetros de desempeño en todo el tiempo de ejecución del aprendizaje de los modelos.

Matriz de confusión sobre modelo de clasificación entrenado

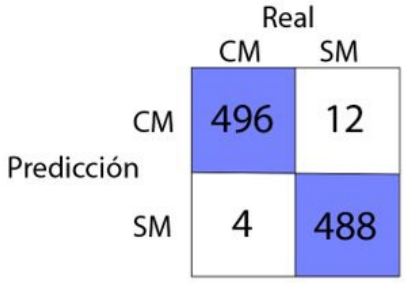
Con el modelo de clasificación entrenado y exportado en extensión .h5 se pone a prueba la eficiencia de la exactitud del modelo preclasificador y clasificador, para esto se realiza las pruebas con un conjunto de 1000 imágenes (500 entre medidores de agua y

medidores de electricidad y 500 aleatorias) para el pre clasificador y 1000 imágenes (500 de medidores de agua potable y 500 de medidores de energía eléctrica), es importante mencionar que estas son imágenes que no se utilizan durante el entrenamiento.

Por otro lado, con la matriz de confusión se presentan los resultados de las pruebas realizadas, en una matriz 2x2 que consta en su eje vertical las diferentes clases con las que fue entrenado el modelo (CM: con medidor y SM: sin medidor) para el caso del pre clasificador y (MA: medidor de agua y ME: medidor de electricidad) para el caso del clasificador y en el eje horizontal se encuentran las predicciones realizadas.

Figura 53

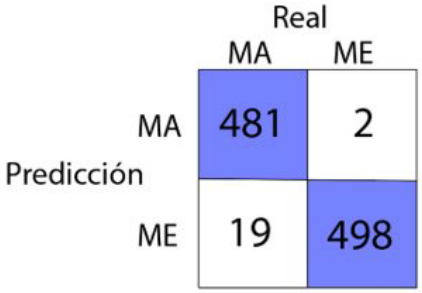
Matriz de confusión del pre clasificador



Nota. La figura muestra la tabla de confusión obtenida al poner al clasificador a prueba con 500 imágenes entre medidores de agua y de electricidad y 500 imágenes aleatorias.

Figura 54

Matriz de confusión del clasificador



Nota. La figura muestra la tabla de confusión obtenida al poner al clasificador a prueba con 500 imágenes de medidores de agua y 500 de medidores de electricidad.

En la Figura 53 se muestra la matriz de confusión obtenida al realizar las pruebas al pre clasificador, el porcentaje de predicciones correctas realizadas se presentan a continuación.

Predicciones de CM correctas:

$$PC_{CM} = \frac{496}{500} \times 100 = 99.2\%$$

Predicciones de SM correctas:

$$PC_{SM} = \frac{488}{500} \times 100 = 97.6\%$$

Predicciones total correctas:

$$PC_{T} = \frac{496 + 488}{1000} \times 100 = 98.4\%$$

Así mismo, en la Figura 54 se muestra la matriz de confusión obtenida al realizar las pruebas al clasificador, el porcentaje de predicciones correctas realizadas se presentan a continuación.

Predicciones de MA correctas:

$$PC_{MA} = \frac{471}{500} * 100 = 94.2\%$$

Predicciones de ME correctas:

$$PC_{ME} = \frac{499}{500} * 100 = 99.8\%$$

Predicciones total correctas:

$$PC_{T} = \frac{481 + 499}{1000} * 100 = 97{,}9\%$$

Con base a la matriz de confusión obtenida que se muestra en la Figura 53 y Figura 54 se realiza el cálculo de algunos de parámetros de desempeño del pre clasificador y clasificador por cada clase para las cuales fue entrenado, el valor de cada parámetro se presenta en la Tabla 16 y Tabla 17 respectivamente.

Tabla 16

Parámetros de desempeño del pre clasificador

Sensibilidad (%)	Especificidad (%)	BER
99,2	97,6	0,016

Nota. La tabla presenta los parámetros de desempeño del pre clasificador para cada clase, el valor de cada parámetro se lo obtuvo con las ecuaciones definidas en el capítulo anterior.

Tabla 17

Parámetros de desempeño del clasificador

Sensibilidad (%)	Especificidad (%)	BER
96,2	99,6	0,021

Nota. La tabla presenta los parámetros de desempeño del clasificador para cada clase, el valor de cada parámetro se lo obtuvo con las ecuaciones definidas en el capítulo anterior.

Parámetros de desempeño del detector de dígitos

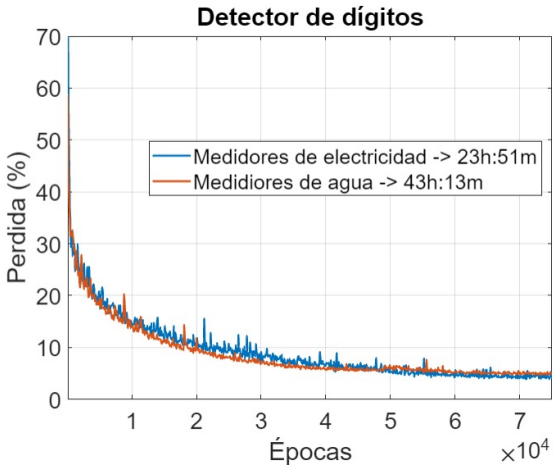
En este apartado se presentan la gráfica de error de los modelos entrenados para la detección de dígitos en los medidores. Esta gráfica presenta la disminución de la pérdida total duranta las épocas que fue entrenado. La pérdida total se compone principalmente por la pérdida de clasificación y pérdida de localización, dado que este modelo es capaz de clasificar un dígito entre los existen del 0 a 9 y además es capaz localizar en qué región de la imagen se encuentra. Este modelo es entrenado con 75000 épocas alcanzando una pérdida mínima de alrededor del 4%.

En la Figura 55 se muestra el porcentaje de pérdida para el detector de dígitos, en naranja se muestra los medidores de agua. En azul se muestra la pérdida para el detector de

dígitos en medidores de electricidad. Este modelo fue entrenando con las mismas épocas que el anterior modelo y de igual manera alcanza una pérdida mínima de alrededor del 4%.

Figura 55

Curvas de pérdida total para modelos de detección de dígitos.



Nota. La figura presenta en la curva azul la pérdida para la detección de dígitos en medidores de energía eléctrica y en la curva naranja para medidores de agua.

Se puede apreciar en ambas curvas que a partir de la época 50000 empieza a converger la curva de pérdida, por lo que ya no se requiere realizar el entrenamiento con más épocas y además se debe evitar el sobreajuste. Finalmente, para evaluar el rendimiento de los modelos de detección de dígitos se obtiene las curvas de precisión para cada clase de dígito en medidores de agua y de energía eléctrica. Esto nos permite evaluar tanto el rendimiento en clasificación como en localización de los dígitos, ya que los modelos de detección deben ser capaces de identificar acertadamente si un dígito está presente en la fotografía del medidor y cuál dígito es, además, debe predecir en que región de la imagen se encuentra éste.

La base de esta evaluación es el cotejo entre las detecciones teóricas y reales. En la Figura 56 se muestran en recuadros verdes las regiones de detección teórica y en recuadros rojos las regiones de detección reales obtenidas por el modelo, con esto se determinan dos

cosas, la probabilidad de que los recuadros teóricos coincidan con los reales (Precisión) y la probabilidad de que los dígitos en los recuadros reales se detecten fielmente (Sensibilidad).

Figura 56

Regiones de detección teórica y real en un medidor de electricidad.

En la Figura 57 se presentan las curvas de Precision – Recall obtenidas para los dígitos en medidores de agua potable, en éstas se evidencia que la precisión de coincidencia de recuadros teóricos y reales para todos los dígitos es bastante alta, con un valor de precisión media (mAP) de 98.13% y así mismo el *recall* donde los recuadros reales correspondan a la clase correcta.

En la Figura 58 se presentan las curvas obtenidas para los dígitos en medidores de electricidad, para este caso con una alta probabilidad de que los recuadros reales pertenezcan a la clase correcta tienen un valor de mAP de 91.34%.

Figura 57

Curvas Precision – Recall en modelo de detección de dígitos de medidores de agua.

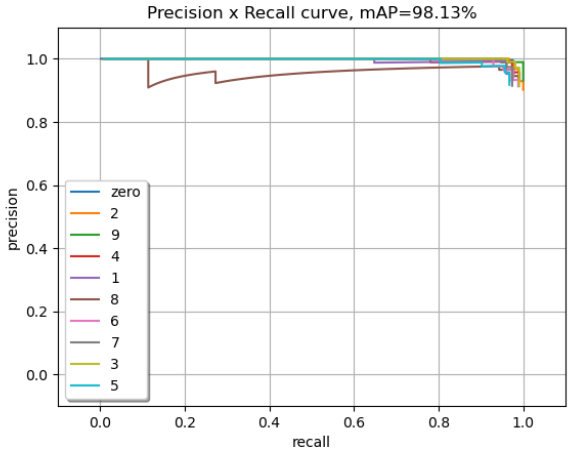
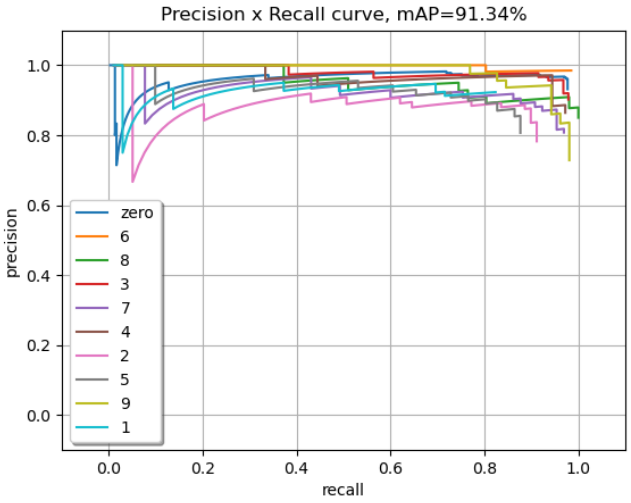


Figura 58

Curvas Precision – Recall en modelo de detección de dígitos de medidores de electricidad.



Ambos modelos de detección respondieron notablemente a la evaluación de su rendimiento, ya que se necesita que dichos modelos logren dos funciones localizar todos los dígitos de la imagen con los que fue entrenado y asignarlos correctamente a la clase apropiada.

Conclusiones

Mediante el estado del arte se obtuvo una descripción y evaluación del balance actual de aplicaciones desarrolladas utilizando AV, también permitió conocer los diferentes enfoques que se han aplicado en la actualidad y reutilizarlos, con la finalidad de ahorrar tiempo de desarrollo y recursos.

Para los bloques de clasificación y reconocimiento se utilizó el modelo MobileNetV2 debido a sus parámetros de desempeño, ya que presenta la mejor exactitud al utilizar una mínima cantidad de recursos, considerando que el modelo es ejecutado en un teléfono móvil.

Para elevar la exactitud sobre el 90% y generalización del aprendizaje de los modelos entrenados, se aplicaron varias técnicas de preprocesamiento de imágenes al *dataset* como la inversión, rotación, traslación e inyección de ruido. Adicional, se descartó imágenes con dimensiones menores a 224x224 con la finalidad de garantizar mayor detalle en los aspectos característicos de los medidores.

El aplicativo móvil se desarrolló en Android Studio con una interfaz simple e intuitiva de utilizar, de tal forma que permite al recolector tomar las lecturas de forma ágil y disminuye el tiempo con la respectiva ruta, la aplicación tiene un peso de 269 MB y es compatible a partir de la versión Android 6.0. Además, tiene la capacidad de almacenar hasta 100 lecturas en memoria caché cuando el dispositivo no tenga conexión a internet; siempre que el cliente ya haya sido consultado antes.

La página web desarrollada en este trabajo para la presentación del consumo obtenido de los medidores, fue implementada de manera local en una máquina virtual utilizando el servidor Nginx, esta realiza consultas a la base de datos Firestore de Firebase que permite alojar hasta 1 GB de información de manera gratuita, además, mantiene los datos sincronizados entre cliente y servidor, por lo que permite gestionar datos sin importar la latencia de la red ni la conectividad a la Internet, es decir, permite al usuario almacenar datos sin

conexión y sincronizarlos con la base de datos inmediatamente cuando se restablece la conectividad.

El sistema consta principalmente de un modelo de preclasificación, uno de clasificación y un detector; el primer modelo entrenado tiene una exactitud del 98.4% y un BER de 0.016; es decir; de 100 imágenes el preclasificador se equivoca 1, el modelo de clasificación tiene una exactitud del 97.9% y un BER de 0.021; es decir; de 100 imágenes se equivoca 2, además después de las pruebas en campo del sistema completo con la aplicación se obtuvo un porcentaje de error del 4% para el caso de medidores de agua potable y 3% para medidores de energía eléctrica al realizar lecturas. Con ello se concluye que todas las métricas de desempeño están dentro del rango establecido al principio de la investigación.

Trabajos futuros

Actualmente, siguen surgiendo nuevas CNN para aplicaciones móviles. La familia de Efficientnet es una de ellas, la cual promete ofrecer mejorar sustanciales con respecto a la familia Mobilnet. Se recomienda implementar un modelo de visión artificial perteneciente a la familia Efficientnet y realizar una comparación en cuanto a parámetros de rendimiento y uso de recursos.

La aplicación de la plataforma de Tensorflow Serving abre la puerta a nuevas metodologías e implementación de modelos de AV de alto rendimiento, dicha plataforma permite la integración fácil de modelos de Tensorflow. La implementación de modelos ejecutados en la nube será una nueva línea trabajo para aplicaciones móviles, la cual evitará que todo el procesamiento de AV se realice sobre el teléfono móvil.

Se recomienda anclar este sistema a uno de facturación en donde se permita revisar facturas vencidas, saldos y demás herramientas contables necesarios para una empresa de recaudación.

La aplicación desarrollada es capaz de enviar imágenes a una base de datos, sin embargo, la base actualmente utilizada tiene recursos limitados por ser una versión gratuita, se

recomienda implementar una base de datos en un servidor local que tenga mayor capacidad de almacenamiento con el fin de tener un respaldo de la lectura.

Bibliografía

Acens. (2019). *Servidor web Nginx, una clara alternativa a Apache*. https://www.acens.com/wp-content/images/2013/09/servidor-web-nginx-white-paper-acens.pdf

Ahn, D. (2021). *Accuracy and Loss: Things to Know about The Top 1 and Top 5 Accuracy*. Towards Data Science. https://towardsdatascience.com/accuracy-and-loss-things-to-know-about-the-top-1-and-top-5-accuracy-1d6beb8f6df3#:~:text=Top-5 accuracy means any,%3D 3%2F5 %3D 0.6.

Aliferi, C. (2016). *Android Programming CookBook* (2nd ed.). Exelixis PC. https://enos.itcollege.ee/~jpoial/allalaadimised/reading/Android-Programming-Cookbook.pdf

Alvarellos, A., Picón, D., Puertas, J., & Rabuñal, J. (2017). Sistema para medición de manómetros analógicos mediante visión artificial. *Linea Temática C | Agua y Ciudad*, *24*(26), 1–12. http://www.ingenieriadelagua.com/2004/JIA/Jia2017/wp-content/uploads/ponencias/posters/rb8.pdf

Alvares, A., Alves, A., & Feitosa de Castro, M. (2020). Implementación de una aplicación móvil (App) para lectura medidores de agua y energía basados en visión artificial. *Revista de Computación Aplicada*, *12*(3), 107–121. https://doi.org/http://orcid.org/0000-0001-6745-1437

Batschinski, G. (2021a). *Cloud Firestore*. https://blog.back4app.com/what-is-cloud-firestore/

Batschinski, G. (2021b). *Cloud Firestore*.

Boesch, G. (2020). *LabelImg for Image Annotation*. Computer Vision. https://viso.ai/computer-vision/labelimg-for-image-annotation/

Carrasco, M. (2021). Contratista de CNEL que puso medidores en Guayaquil ahora se encarga de las lecturas, cuestionadas por alto costo de planillas. *El Universo*. https://www.eluniverso.com/noticias/ecuador/contratista-de-cnel-que-puso-medidores-en-guayaquil-ahora-se-encarga-de-las-lecturas-cuestionadas-por-alto-costo-de-planillas-nota/

Chang, O., Pruna, E., Pilatasig, M., Escobar, I., & Mena, L. (2015). Calibration of residential water meters by using computer vision. *2015 CHILEAN Conference on Electrical, Electronics Engineering, Information and Communication Technologies (CHILECON)*, 351–356. https://doi.org/10.1109/Chilecon.2015.7400401

Cuenca, J., & Ortega, J. (2017). Ingreso de lecturas de consumo de agua potable en EMAPAL-Azogues, a través de dispositivos móviles. *Revista Científica y Tecnología UPSE*, *4*(2), 2–11. https://doi.org/https://doi.org/10.26423/rctu.v4i2.228

D'Arc, T. (2020). *Comprender la inteligencia artificial y ver ejemplos*. SmartHint. https://www.smarthint.co/es/o-que-e-inteligencia-artificial-exemplos/?utm_source=blog&;utm_medium=post&;utm_campaign=buscainteligente

Deng, Y. (2019). Deep Learning on Mobile Devices – A Review. *FAST Labs, BAE Systems*, *19*(5), 1–15.

EPMAPS. (2021). *Plan de negocios*. https://www.aguaquito.gob.ec/Alojamientos/Transparencia/K 2021/PLAN DE NEGOCIOS 2021.pdf

Garcia, I., & Caranqui, V. (2015). La Visión Artificial y los Campos de Aplicación. *TIERRA INFINITA*, *1*(1), 94–103. https://doi.org/10.32645/26028131.76

García, J. (2021). *El uso de la telemetría en la gestión del agua optimiza su funcionamiento, disminuye el desperdicio, reduce costos y favorece el ambiente.* Telcel. https://www.telcel.com/empresas/tendencias/notas/mejorar-gestion-de-agua-con-telemetria

Gevorkyan, M. N., Demidova, A., Demidova, T., & Sobolev, A. (2019). Computer Scienceand Computer Engineering. *Discrete y Continuous Models*, *27*(4), 305–315. https://doi.org/0.22363/2658-4670-2019-27-4-305-315

Giang, N., & Dlugolinsky, S. (2019). Machine Learning and Deep Learning frameworks and libraries for large-scale data mining: a survey. *Artificial Intelligence Review*, *52*(1), 77–124. https://doi.org/10.1007/s10462-018-09679-z

GitHub, I., & Tzutalin. (2022). *EtiquetaImg*. EtiquetaImg. https://github.com/heartexlabs/labelImg

Google. (2022). *Cloud Firestore*. https://firebase.google.com/docs/firestore

Hernández, F., Manuel, C., & Espejel, D. (2020). Sistema modular de telemetría para medición de parámetros en el hogar, basado en el internet de las cosas. *Revista de Tecnologías Computacionales*, *4*(14), 17–27. https://doi.org/10.35429/JOCT.2020.14.4.17.27

Howard, A. G., Zhu, M., Chen, B., Kalenichenko, D., Wang, W., Weyand, T., Andreetto, M., & Adam, H. (2017). Mobilenets: Efficient convolutional neural networks for mobile vision applications. *ArXiv Preprint ArXiv:1704.04861*.

Karimi, Z. (2021). Confusion Matrix. *ResearchGate*, *1*(1), 4. https://www.researchgate.net/publication/355096788_Confusion_Matrix

Keras. (n.d.). *Keras Applications*. Keras. Retrieved July 6, 2022, from https://keras.io/api/applications/

Laroca, R., Barroso, V., Diniz, M. A., Gonçalves, G. R., Schwartz, W. R., & Menotti, D. (2019). Convolutional neural networks for automatic meter reading. *Journal of Electronic Imaging*, *28*(1), 1–14. https://doi.org/10.1117/1.JEI.28.1.013023

Lee, K., & Street, W. (2003). Model-based detection, segmentation and classification for image analysis using on-line shape learning. *Machine Vision and Applications*, *13*(2003), 222–233. https://doi.org/10.1007/s00138-002-0061-6

Mitsa, T. (2019). *How Do You Know You Have Enough Training Data?* Towards Data Science. https://towardsdatascience.com/how-do-you-know-you-have-enough-training-data-ad9b1fd679ee

Navarro, J. (2013). *Software de Adquisición de Imágenes y Reconocimiento Óptico de Caracteres para Android* [Tesis de Master, Universitat Oberta de Catalunya]. http://openaccess.uoc.edu/webapps/o2/bitstream/10609/18691/9/jaimenavarrosantapauTFM0113memoria.pdf

Nelson, J. (2020). *LabelImg for Labeling Object Detection Data*.

https://blog.roboflow.com/labelimg/

Pang, B., Nijkamp, E., & Wu, Y. (2019). Deep Learning With TensorFlow: A Review. *Journal of Educational and Behavioral Statistics*, *20*(10), 1–22. https://doi.org/10.3102/1076998619872761

Planche, B., & Andres, E. (2019). *Computer Vision with TensorFlow 2*. Packt. https://github.com/PacktPublishing/Hands-On-Computer-Vision-with-TensorFlow-2

Planeta ChatBot. (2019). *Inteligencia Artificial, Aprendizaje Automático y Aprendizaje Profundo*. Planeta ChatBot. https://planetachatbot.com/inteligencia-artificial-aprendizaje-automatico-y-aprendizaje-profundo/

Praveen, J., Prasanna, H., & Chiplunkar, N. (2021). Image classification and prediction using transfer learning in colab notebook. *KeAi Chinese Roots Global Impact*, *2*(2021), 282–385. https://doi.org/10.1016/j.gltp.2021.08.068

Programador Clic. (n.d.). *Introducción a la clasificación y detección de objetivos*. Programmer Click.

Roman, K. (2020). *Water Meters Dataset [Kaggle - Dataset]*. https://www.kaggle.com/datasets/tapakah68/yandextoloka-water-meters-dataset

Rouhiainen, L. (2018). *Inteligencia artificial: 101 cosas que debes saber hoy sobre nuestro futuro*. Alienta Editorial. https://books.google.com.ec/books?id=tlWavgEACAAJ

Sandler, M., Howard, A., Zhu, M., Zhmoginov, A., & Chen, L.-C. (2018). Mobilenetv2: Inverted residuals and linear bottlenecks. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 4510–4520.

Sharma, P. (2021). *Image Classification vs. Object Detection vs. Image Segmentation*. Medium. https://medium.com/analytics-vidhya/image-classification-vs-object-detection-vs-image-segmentation-f36db85fe81

Shorten, C., & Khoshgoftar, T. (2019). A survey on Image Data Augmentation for Deep Learning. *Journal of Big Data*, *6*(60), 49. https://doi.org/https://doi.org/10.1186/s40537-019-

0197-0

Smyth, N. (2015). *Android Studio Development Essentials*. https://www.ebookfrenzy.com/pdf_previews/AndroidStudioEssentialsPreview.pdf

Tatic. (2021). *Image Classification vs. Object Detection vs. Image Segmentation*. Medium. https://medium.com/analytics-vidhya/image-classification-vs-object-detection-vs-image-segmentation-f36db85fe81

TensorFlow. (2021). *Mejores prácticas de desempeño*. TensorFlow. https://www.tensorflow.org/lite/performance/best_practices

Vakili, M., Ghamsari, M., & Rezaei, M. (2020). Performance Analysis and Comparison of Machine and Deep Learning Algorithms for IoT Data Classification. *Sharif University of Technology*. https://arxiv.org/ftp/arxiv/papers/2001/2001.09636.pdf

Valencia, J., Ramirez-Guerrero, T., Castañeda, L. F., & Toro, M. (2020). Detection of infractions and license plates in motorcycles by means of artificial vision, applied to Intelligent Transport Systems. *RISTI - Revista Iberica de Sistemas e Tecnologias de Informacao*, 1–15. https://doi.org/10.17013/risti.37.1-15

Wong, S., Gatt, A., Stamatescu, V., & McDonnel, M. (2016). Understanding Data Augmentation for Classification: When to Warp? *IEEE*, *2*(1), 6. https://doi.org/10.1109/DICTA.2016.7797091

Yuan, L., Qu, Z., Zhao, Y., Zhang, H., & Nian, Q. (2017). A Convolutional Neural Network based on TensorFlow for Face Recognition. *Advanced Information Technology, Electronic and Automation Control Conference IAEAC*. https://doi.org/10.1109/IAEAC.2017.8054070