Artículo Científico / Scientific Paper

UNIVERSIDAD POLITÉCNICA SALESIANA ECUADOR
INGENIUS
Revista de Ciencia y Tecnología
https://doi.org/10.17163/ings.n27.2022.04
pISSN: 1390-650X / eISSN: 1390-860X

Evaluación de AIoT en modelos computacionales en la nube y en el borde aplicado a la detección de mascarillas

Evaluation of AIoT performance in Cloud and Edge computational models for mask detection

Felipe Quiñonez-Cuenca[1,*], Cristian Maza-Merchán[1], Nilvar Cuenca-Maldonado[1], Manuel Quiñones-Cuenca[1], Rommel Torres[1], Francisco Sandoval[1], Patricia Ludeña-González[1]

Recibido: 15-11-2021, Recibido tras revisión: 20-12-2021, Aceptado: 27-12-2021, Publicado: 01-01-2022

Resumen

La COVID-19 ha provocado graves daños a la salud: centenas de millones de personas infectadas y varios millones de fallecidos en el mundo. Los programas de vacunación de cada Gobierno han influido en el decaimiento de estos índices, pero con la aparición de nuevas mutaciones del coronavirus más contagiosas, la preocupación sobre la efectividad de las vacunas se hace presente. Frente a esta situación el uso de mascarillas sigue siendo eficaz para prevenir la transmisión y contagio de la COVID-19. Lo que ha generado una creciente demanda de servicios de detección automática de mascarillas, que permita recordar a las personas la importancia del empleo de estas. En este trabajo se plantea un análisis del rendimiento de un sistema AIoT para la detección del uso correcto, incorrecto y sin mascarilla basado en dos modelos computacionales de Cloud y Edge, con la finalidad de determinar qué modelo se adecua mejor en un entorno real (interior y exterior) sobre la base de la confiabilidad del algoritmo, uso de recursos computacionales y tiempo de respuesta. Los resultados experimentales demuestran que el modelo computacional Edge presentó un mejor desempeño en comparación con el Cloud.

Palabras clave: AIoT, COVID-19, computación en la nube, computación de borde, detección de máscara facial, YOLO

Abstract

COVID-19 has caused serious health damage, infecting millions of people and unfortunately causing the death of several ones around the world. The vaccination programs of each government have influenced in declining those rates. Nevertheless, new coronavirus mutations have emerged in different countries, which are highly contagious, causing concern with vaccination effectiveness. So far, wearing facemasks in public continues being the most effective protocol to avoid and prevent COVID-19 spread. In this context, there is a demand of automatic facemask detection services to remind people the importance of wearing them appropriately. In this work, a performance evaluation of an AIoT system to detect correct, inappropriate, and non- facemask wearing, based on two computational models: Cloud and Edge, was conducted. Having as objective to determine which model better suites a real environment (indoor and *outdoor*), based on: reliability of the detector algorithm, use of computational resources, and response time. Experimental results show that Edge-implementation got better performance in comparison to Cloud-implementation.

Keywords: AIoT, COVID-19, Cloud Computing, Edge Computing, Face mask detection, YOLO

[1,*]Departamento de Ciencias de la Computación y Electrónica, Universidad Técnica Particular de Loja, Loja, Ecuador. Autor para correspondencia ✉: fdquinones@utpl.edu.ec.

Forma sugerida de citación: Quiñonez-Cuenca, F.; Maza-Merchán, C.; Cuenca-Maldonado, N.; Quiñones-Cuenca, M.; Torres, R.; Sandoval, F. y Ludeña-González, P. "Evaluación de AIoT en modelos computacionales en la nube y en el borde aplicado a la detección de mascarillas," *Ingenius, Revista de Ciencia y Tecnología*, N.° 27, pp. -, 2022. DOI: https://doi.org/10.17163/ings.n27.2022.04.

1. Introducción

Actualmente, la humanidad está atravesando una crisis sanitaria debido a la pandemia COVID-19, provocada por la nueva cepa de coronavirus SARS-CoV-2 [1], la misma que ha afectado inesperada y drásticamente la salud, la economía y el estilo de vida de las personas a nivel mundial. El origen del virus fue identificado a finales del año 2019 en la ciudad de Wuhan, China. Posteriormente, el virus se expandió por todo el mundo y el 11 de marzo de 2020 la Organización Mundial de la Salud (OMS) declaró como pandemia global COVID-19 [2]. El SARS-CoV-2 ataca el sistema inmunológico de las personas. Aunque la mayor parte de personas que se infectan con el virus tienen síntomas leves y moderados, un significativo de personas requieren de hospitalización, e incluso el uso de camas UCI (unidad de cuidados intensivos). Dicho virus se transmite principalmente a través de microscópicas gotas de plasma celular expulsado por la persona infectada al toser, estornudar o espirar [3].

Veinte meses de pandemia han dejado una huella en la historia de la salud de los seres humanos, lo que se evidencia a través de datos estadísticos oficiales. Hasta la fecha, 3 de diciembre de 2021, a nivel mundial, de acuerdo con las estadísticas de la OMS (2021), un poco más de doscientos sesenta y tres millones de casos han sido confirmados, de los cuales más de cinco millones fallecieron. A nivel continental, en América presenta el índice más alto de contagios en relación con los otros cinco continentes, cerca de 97 millones de casos confirmados se han reportado, de los cuales más de dos millones de personas fallecieron. Mientras que en el Ecuador, se registra más de 527 000 casos confirmados y más de 33 000 han perdido la vida [4].

Para neutralizar la pandemia COVID-19, los líderes de las naciones a nivel mundial tomaron cruciales decisiones, que desembocaron en problemas colaterales de los cuales la humanidad aún le cuesta trabajo recuperarse. Entre las medidas adoptadas están el confinamiento global, la implementación de protocolos de seguridad, e incluso la invención y distribución de vacunas.

En lo local, a mediados de marzo de 2020, el Gobierno ecuatoriano declaró el estado de emergencia, el cual involucró restricción a la movilidad, aislamiento y cierre de fronteras; medidas que afectaron gravemente la economía del país. Las empresas paralizaron su producción abruptamente, instituciones gubernamentales, educativas y financieras fueron forzadas a continuar sus actividades en línea.

Luego de cuatro meses y medio de confinamiento, la economía del país no podía continuar resistiendo y las restricciones fueron gradualmente eliminadas; pero para laborar presencialmente, protocolos de bioseguridad tenían que ser adoptados por la comunidad. Para prevenir el contagio y propagación del virus SARS-CoV-2 se requiere de cada individuo: usar obligatoriamente mascarillas buconasales, evitar las aglomeraciones, guardar distancia, lavarse regularmente las manos con jabón y desinfectar continuamente las superficies de uso común con alcohol. El seguimiento estricto de los protocolos de bioseguridad ha sido la clave fundamental para prevenir el virus, puesto que inicialmente, debido a que el SARS-CoV-2 era nuevo, no existían medicamento especializado ni vacuna para proteger a las personas.

Hoy en día existen varias vacunas contra la COVID-19 disponibles. La Organización Panamericana de Salud (OPS) [5], en su sitio web oficial, ha puesto a disposición del público en general información relacionada con las vacunas contra la COVID-19, entre las cuales están Pfizer/BioNTech, Moderna, AstraZeneca, Janseen, Sinopharm, y Sinovac. Aunque hay varias vacunas, el proceso de vacunación no avanza de acuerdo con lo planificado. Según las estadísticas de Ritchie *et al.* [6], hasta la fecha (diciembre 2021) el 54,9 % de la población mundial ha recibido al menos una dosis de la vacuna COVID-19. Se han administrado más de ocho mil millones de dosis en todo el mundo, y cada día se administran treinta y cuatro millones.

Simultáneamente al proceso de vacunación mundial, nuevas variantes del virus COVID-19 han sido identificadas en diferentes regiones del planeta. El elevado número de infectados aumenta el riesgo de mutaciones del virus. Cuanto más se propague el virus, pequeños cambios ocurren en su código genético; lo que le permite sobrevivir y reproducirse. Múltiples variantes circulan globalmente, por ejemplo: la variante del Reino Unido, conocida como alpha; la variante de Sudáfrica, conocida como beta; la variante brasileña, conocida como gamma, la variante de la India, conocida como delta; y la última denominada ómicron detectada en Sudáfrica. De acuerdo con los expertos, los virus mutan todo el tiempo y la mayoría de los cambios son intrascendentes. Pero otros pueden hacer que la enfermedad sea más infecciosa o amenazante, y estas mutaciones tienden a dominar [7].

El objetivo de la vacunación es lograr la inmunidad colectiva a nivel mundial para evitar que el SARS-CoV-2 continúe mutando, volviéndose más resistente a las vacunas actuales y provocando más períodos de mortalidad masiva. No obstante, la OPS, a finales de la primera semana de agosto, en vista de un alto índice de contagio de la "variante preocupante" (VOC, por su sigla en inglés) delta, en varios países, dentro y fuera de la región de las Américas, recomienda revisar los planes de contingencia y prepararse para un eventual aumento de casos y hospitalizaciones.

En este reporte se destaca que, el distanciamiento social, el uso de mascarillas buconasales y el uso de soluciones antisépticas continúan siendo las medidas más efectivas para reducir la transmisión de esta y todas las variantes.

A partir de los datos anteriormente mencionados, se puede deducir que la pandemia COVID-19 aún no se ha acabado, el futuro es incierto. Por ende, son necesarias la investigación e innovación para que den sus aportes tecnológicos a la sociedad en la lucha contra ella.

En este trabajo se describe el diseño, implementación y el análisis de desempeño de un sistema basado en dos modelos computacionales en la nube (Cloud) y en el borde (Edge), aplicado a la detección del uso de mascarilla en tiempo real mediante la aplicación de inteligencia artificial de las cosas (AIoT, por sus siglas en inglés). La importancia del desarrollo y evaluación del sistema radica en que se analiza el tiempo de respuesta, rendimiento del algoritmo de detección y recursos computacionales.

En la Sección II se describe el método propuesto, se detallan las arquitecturas computacionales y los componentes usados. En la Sección III se presentan los resultados del análisis de los modelos computacionales. Finalmente, en la Sección IV se evidencian las conclusiones y futuros trabajos.

2. Materiales y métodos

Mediante un diseño experimental y un enfoque cuantitativo el desarrollo de la investigación se dividió en cinco fases (ver Figura 1). A continuación, las actividades que se realizaron en cada una de las fases:

- Análisis del estado del arte de tecnologías habilitantes para AIoT, trabajos relacionados y algoritmos de detección, para detectar el uso de mascarillas en tiempo real.
- Diseño de una arquitectura y determinación de componentes de hardware y software, para el despliegue de los modelos computacionales en la nube y en el borde de la red.
- Desarrollo es la etapa en que se integran los componentes de software y hardware, y se implementa el algoritmo de detección y ejecución para detección de mascarillas en tiempo real, en los dos modelos computacionales.
- Evaluación es la etapa en que se determina un escenario controlado y escenarios reales, para realizar pruebas que permitan validar el rendimiento de los modelos computacionales.
- Análisis de los resultados para determinar el desempeño de los recursos utilizados en ambos modelos computacionales. Entre las métricas recopiladas se considera la demanda de recursos computacionales (CPU, RAM, memoria y almacenamiento), y el tiempo de respuesta del sistema. Para analizar el desempeño del algoritmo de detección se evalúan la exactitud, la precisión, la revocación, la media armónica y la media de la precisión promedio.

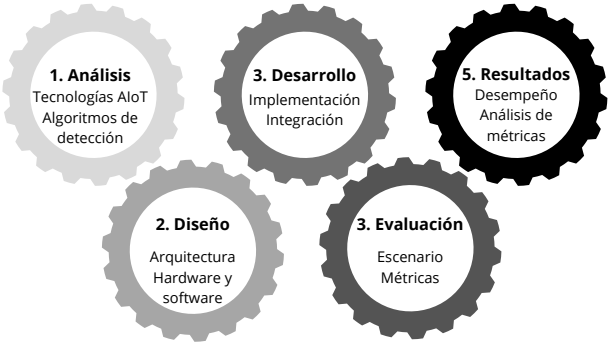


Figura 1. Metodología

2.1. Arquitectura

Para el desarrollo del sistema se diseñó una estrategia de comparación que permita que los dos modelos computacionales sean dotados con las mismas o similares capacidades. Sin embargo, debido a que cada implementación tiene características intrínsecas que las diferencian una de la otra, la estrategia de comparación fue diseñada con dos estructuras, una común para ambos modelos computacionales y la específica de cada uno. En la Figura 2 se puede visualizar la disposición de cada uno de los componentes de hardware y software que forman parte de la arquitectura para el despliegue del sistema, segmentado en escenario, sensores y actuadores, modelos computacionales y actuadores.

La arquitectura del sistema integra una cámara IP para obtener el video del escenario de prueba, este se envía por medio del protocolo de transmisión en tiempo real (RTSP, por sus siglas en inglés) a los dos modelos computacionales, en estos se ejecuta el procesamiento del video aplicando el algoritmo de detección de objetos en tiempo real basado en YOLOv3 («You only look once v3») para determinar el uso correcto e incorrecto y sin mascarilla. El modelo computacional Edge realiza la inferencia en tiempo real y presenta los resultados en una pantalla marcando en un recuadro los rostros dependiendo del resultado con «Mask correct» (mascarilla correcta), «Mask incorrect» (mascarilla incorrecta) y «No mask» (sin mascarilla), se generan alarmas sonoras y visuales; luego se procede a enviar a una plataforma web en la nube para almacenar las imágenes procesadas. Mientras que en el modelo computacional Cloud se realiza el procesamiento y mediante un dispositivo se puede visualizar el resultado y se almacena la inferencia como se detalla en el modelo Edge.

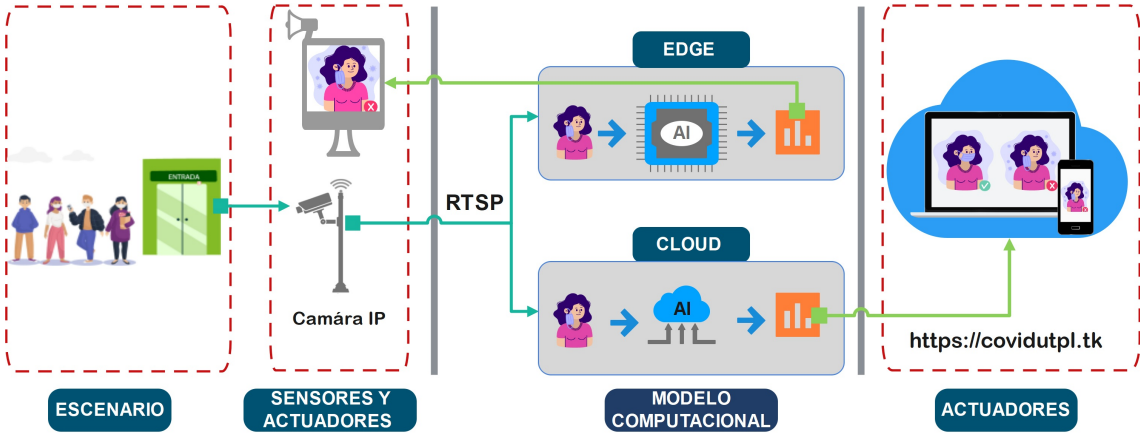


Figura 2. Arquitectura del sistema

2.1.1. Estructuración común para Cloud y Edge

De acuerdo con [8], el análisis comparativo se enfatiza en la explicación amplia de similitudes y diferencias de los fenómenos, para luego proporcionar razones válidas en un tema o área de interés. Consecuentemente, esta sección describe las «similitudes» de la estrategia de comparación entre los dos modelos computacionales Cloud y Edge. Para evaluar el desempeño del algoritmo AIoT, se planteó el uso del mismo protocolo de transmisión, códec de video, sensor visual (cámara IP), conjunto de datos (dataset) de entrenamiento, algoritmo de entrenamiento, dataset de evaluación, y algoritmo de ejecución para la detección de mascarillas en tiempo real, en ambos modelos.

2.1.2. Protocolo de transmisión

Actualmente, existen varios protocolos de transmisión como RTSP (Real Time Streaming Protocol), RTMP (Real-Time Messaging Protocol), SRT (Secure Reliable Transport), y WebRTC (RTC – Real Time Communication) [9]. Para seleccionar el protocolo de transmisión a utilizar en la estrategia comparativa de la presente investigación, se tomó en cuenta los resultados de investigaciones recientes y la compatibilidad del protocolo de transmisión con las cámaras de video de vigilancia disponibles en el mercado local. De acuerdo con Santos-González *et al.* [10], uno de los protocolos de transmisión más usados en video streaming es el RTSP, especialmente en ambientes con restricciones de ancho de banda, congestión de la red, eficiencia energética, costo, confiabilidad y conectividad. Mientras que, según Nurrohman y Abdurohman [11], el protocolo RTMP ofrece un mejor desempeño en la transmisión de video en directo en comparación con el protocolo RTSP.

Al revisar la compatibilidad de los protocolos de transmisión con los sensores visuales (cámaras de video vigilancia) ofertados en el mercado local, se encontró cámaras que disponían de dos protocolos RTMP y RTSP. Para implementar el mecanismo de comunicación *push* del protocolo RTMP, es necesario añadir otro componente (un servidor streaming) a la red, alterando el flujo de los datos en la red y la arquitectura de comparación, e incidiría directamente en el resultado del análisis de tráfico. Por otro lado, el mecanismo *pull* del protocolo RTSP, constituye una opción conveniente de implementar puesto que ambos modelos computacionales Cloud y Edge pueden conectarse como clientes al mismo sensor visual (cámara IP de vigilancia). Por lo tanto, RTSP fue seleccionado como protocolo de transmisión para la estrategia comparativa debido a que se adapta mejor a la arquitectura de comparación planteada.

2.1.3. Códec de video

De acuerdo con Nurrohman y Abdurohman [11], el códec de video H.264 es uno de los estándares de compresión más funcionales en aplicaciones de Internet de las cosas (IoT-Internet of Things, por sus siglas en inglés) debido a que ocupa menos capacidad cuando se almacena o se transmite. Asimismo, los estándares de compresión de video del códec H.264 están basados en compensación de movimiento; este códec es muy recomendable para la grabación, compresión y distribución de archivos de video en tiempo real [12]. También, el códec de video H.264 es compatible con el protocolo RTSP y está disponible en gran parte de los sensores visuales ofertados en el mercado local. Finalmente, en función de los antecedentes mencionados se determina que el códec H.264 es compatible con la arquitectura propuesta.

2.1.4. Algoritmo de detección de mascarillas

El AIoT combina dos enfoques, a saber el IoT y la Inteligencia Artificial (*AI-Artificial Inteligence*, por sus siglas en inglés) [13]. El enfoque IoT se refiere al

concepto de interconectar objetos a la red, de tal manera que la información pueda ser recolectada a través de sensores automáticamente, sin la intervención exclusiva de personas [14]. Mientras que la AI se refiere al enfoque de dotar a las máquinas inteligencia a través de algoritmos, para que puedan tomar decisiones según un entrenamiento previamente recibido [15]. Al unir ambos enfoques se intenta adherir una capa cognitiva a la red, para alcanzar la optimización de recursos a través de la autonomía que se logre dar a las máquinas para analizar situaciones y tomar decisiones.

El procesamiento de información multimedia es un gran desafío para los algoritmos AIoT debido a la gran cantidad de datos que esta actividad implica procesar y debido a los recursos limitados de IoT. Particularmente, en el presente caso de estudio «detección de uso de mascarilla en tiempo real», se pretende evaluar el rendimiento de los algoritmos AIoT en los modelos computacionales Cloud y Edge. Para lo cual, en el estado de arte se identificó que la solución técnica para este tipo de problemas de reconocimiento de objetos, es necesario el uso de algoritmos AIoT, basados en aprendizaje profundo (Deep Learning) con arquitecturas CNN (*Convolutional Neural Network*) [16].

En lo referente a la detección de objetos en tiempo real, el estado de arte resalta a los modelos detectores de una etapa por tener un mejor desempeño en comparación con los de dos etapas. El algoritmo YOLO, un modelo detector de una sola etapa, se caracteriza por una significativa diferencia de velocidad en comparación con los modelos de dos etapas R-CNN (*Region Based Convolutional Neural Networks*) y Fast-R-CNN. YOLO es mil veces más rápido que R-CNN y cien veces más rápido que Fast-R-CNN [16]. En contraste con el enfoque adoptado por los algoritmos de detección de objetos predecesores de YOLO, que reutilizan los clasificadores para realizar la detección, YOLO propone el uso de una red neuronal de extremo a extremo que hace predicciones de cuadros delimitadores y probabilidades de clase simultáneamente, en una sola iteración [14].

Para la presente estrategia de comparación del desempeño de algoritmos AIoT entre los modelos computacionales Cloud y Edge, se decidió seleccionar la versión reducida de YOLOv3, identificada como YOLOv3-tiny por dos razones. La primera es que dicha versión se trata de un modelo pequeño ideal para implementaciones donde los recursos computacionales son limitados; es decir, es compatible con la arquitectura de la implementación Edge, lo que es propicio para una comparación equitativa. Mientras que la segunda razón se debe al hecho que es la última versión reconocida como oficial, lo que significa que tiene acceso al soporte oficial de la comunidad de desarrollo.

El modelo YOLO es ampliamente implementado en soluciones que requieren la identificación de objetos en tiempo real, debido a su arquitectura y funcionamiento [16–19].

YOLOv3 fue propuesto como una solución que utiliza las CNN modernas, las cuales hacen uso de redes residuales y omiten conexiones. De acuerdo con, [14], autores de YOLOv3, esta versión utiliza la red neuronal convolucional DarkNet-53 mucho más compleja como la columna vertebral del modelo, de 106 capas con bloques residuales y redes de muestreo superior. Dicha arquitectura faculta al modelo YOLOv3 a predecir en tres escalas diferentes; los mapas de características se extraen en las capas 82, 94 y 106 para estas predicciones.

2.1.5. Conjunto de datos

Para el proceso de preparación del algoritmo de detección se requiere de un conjunto de imágenes de entrenamiento y de validación. En este trabajo se requiere entrenar al algoritmo detector de objetos en tiempo real YOLOv3-tiny, para que aprenda y esté en la capacidad de detectar tres clases. Es decir, para la implementación del presente caso de estudio «detección de uso de mascarillas» se entrenará el algoritmo de detección de personas con 1) mascarilla correcta, 2) sin mascarilla y 3) mascarilla incorrecta. La Figura 3 ilustra el conjunto de datos con las tres clases requeridas.

Figura 3. Conjunto de datos con tres clases: mascarilla correcta, sin mascarilla, y mascarilla incorrecta

El conjunto de datos debe contener información relevante para el contexto, es decir, imágenes de personas usando mascarillas de diferentes colores y modelos para la primera clase; imágenes de personas de diferentes edades y etnia para la segunda clase; e imágenes de personas usando la mascarilla de manera incorrecta, por debajo de la nariz o boca, para la tercera clase.

Para el entrenamiento del algoritmo, se personalizó el conjunto de datos seleccionando aleatoriamente imágenes de dos conjuntos de datos públicos Kaggle Medical Mask Dataset [20] y Masked Face (MAFA) dataset [21]. Debido a que dichos conjuntos de datos contienen imágenes reales de personas circulando en diferentes escenarios, a diferencia de otros conjuntos de

datos que contienen las tres clases, pero corresponden únicamente a la parte del rostro de la persona. Lo cual, de acuerdo con Sethi *et al.* [17], la información en contexto es otro enfoque utilizado para mejorar la precisión o la velocidad de detección. Adicionalmente, para el entrenamiento del algoritmo detector se planea dividir el conjunto de datos personalizado de acuerdo con la técnica tipo hold-out, que consiste en dividir el conjunto de datos en dos subconjuntos del 80 % para el entrenamiento y el 20 % restante de los datos para las pruebas [22]. De esta manera, luego del entrenamiento se podrá evaluar el desempeño del algoritmo en datos no vistos [23]. Mientras que, en la fase de ejecución del entrenamiento de la red neuronal especializada en la detección de mascarillas, se utiliza la técnica «*transfer learning*», que consiste en transferir el conocimiento de un modelo preentrenado en un contexto general, a uno más específico [24]. Es decir, para la detección de las tres clases (con mascarilla, sin mascarilla y mascarilla incorrecta) se tomará el modelo base de YOLOv3-tiny preentrenado en la detección de ochenta clases de objetos.

2.2. Algoritmo de ejecución en tiempo real

En el algoritmo 1 se describe mediante pseudocódigo el proceso para la detección de las tres clases (mascarilla correcta, sin mascarilla y mascarilla incorrecta). Este algoritmo tiene como entrada el flujo de imágenes (video stream) capturado por el sensor visual de la red, mientras que la salida será la o las detecciones de las personas clasificadas en las tres opciones.

Previo al proceso de detección se requiere preparar cada fotograma (*frame*) del video stream, mediante el redimensionamiento del *frame* a múltiples escalas de acuerdo con la arquitectura de YOLOv3-tiny. Luego se convierte cada *frame* a los canales RGB (*red, green, blue*). Finalmente, se almacena el *frame* y su fecha de captura.

A continuación, se procede a almacenar en el arreglo «Faces» todas las detecciones utilizando YOLOv3-tiny. Seguidamente, es necesario evaluar si existe alguna detección en el *frame* actual; es decir si el arreglo «Faces» contiene algún elemento (mayor a cero). En el caso de no existir alguna detección en el *frame* actual, el proceso culmina, y se procede a trabajar con el siguiente *frame* del video stream. Mientras que, si existe al menos una detección en el arreglo Faces, se requiere evaluar por cada detección la clase a la que pertenece para resaltar su cuadro delimitador y asignarle la etiqueta de clase (ID - identification) correspondiente. En el caso que: 1) Si el ID de la detección pertenece a la clase «con mascarilla», se resalta el cuadro delimitador de color verde; 2) si el ID de la detección pertenece a la clase «sin mascarilla», se resalta el cuadro delimitador de color rojo; y 3) si el id de la detección corresponde a la clase «mascarilla incorrecta», se resalta el cuadro delimitador de color naranja. En complemento, por cada detección se debe cortar y almacenar el cuadro delimitador de la detección en formato .jpg con la fecha y hora de detección.

Algoritmo 1 Detector de mascarillas a partir del video en tiempo real.

```
Stream RTSP Detección de mascarillas foreach <Frame in Stream> do
    Redimensionar Frame;
    Convertir Frame a RGB;
    Almacenar la fecha y hora;
    Indenticar Rostros en el Frame usando YOLOv3-tiny;
    if Rostros > 0 then
        foreach Rostro in Rostros do
            switch Rostro do
                case Mascarilla correcta do
                    Resaltar con VERDE en el Frame
                case No Mascarilla do
                    Resaltar con ROJO en el Frame
                case Mascarilla incorrecta do
                    Resaltar con Naranja en el Frame
            Cortar Rostro del Frame y almacenar Rostro como archivo JPG;
            Almacenar en la base de datos Rostro y Fechayhoradedetección;
```

2.3. Diseño de arquitectura de Edge

En la Figura 4 se ilustra el diseño de la arquitectura del modelo computacional Edge. Esta tiene como unidad central de procesamiento el componente de Jetson Nano. La cual tiene como entrada el video stream capturado por el sensor visual de la red (cámara IP TpLink Tapo C310 V2).

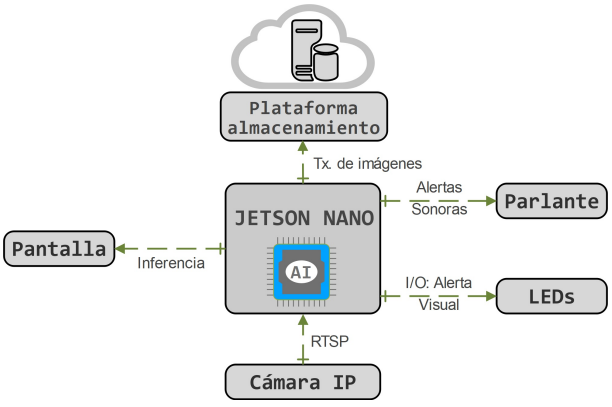


Figura 4. Arquitectura del modelo computacional Edge

En la unidad central de procesamiento se ejecuta el algoritmo AIoT para la detección de uso de mascarillas en tiempo real. Posteriormente, en el caso de

ocurrir una detección se producen cuatro salidas en los actuadores del sistema: 1) la transmisión de las imágenes correspondientes a las detecciones realizadas a un repositorio de la nube, 2) una alerta sonora que resalta la clase de la detección, 3) la activación de LED dependiendo de la clase detectada, y 4) la visualización de la detección en una pantalla. Finalmente, en la Figura 5, se evidencia los resultados de la inferencia en el modelo computacional Edge.

Figura 5. Resultados de la inferencia en el modelo computacional Edge

2.4. Diseño de arquitectura de Cloud

Existen plataformas Cloud que permiten a los usuarios finales desplegar sus soluciones AIoT basadas en transmisión y análisis de video en tiempo real, así como su almacenamiento, entre ellas *Amazon Web Service* (AWS), *Kinesis Video Stream* (KVS) y *SafetyRadar* [25]. Las primera plataforma, se caracterizan por ofrecer diferentes tipos de detección de objetos, por ejemplo, casas, autos, animales, etc. Al permitir implementar la detección de diferentes objetos, se requiere de conocimientos técnicos para utilizar estas plataformas. Mientras que la plataforma *SafetyRadar*, es especializada en la detección de mascarillas y otros implementos de bioseguridad, por lo que ofrece la tecnología de *plug and play*, es decir, no se requiere de conocimientos técnicos para su despliegue. Sin embargo, las tres plataformas son de pago.

Para el presente trabajo se optó por el diseño de una arquitectura propia para el modelo computacional en la nube (Cloud); con esto se tiene el control sobre los diferentes componentes de red. Por lo general, las plataformas no permiten la extracción de métricas que se requieren para la comparación, propósito de la presente investigación.

En la Figura 6 se ilustra el diseño de la arquitectura del modelo computacional Cloud. Desde la red local se accede a través del protocolo de transmisión RTSP al video stream capturado por el sensor visual de la red. Mientras que, en la nube, se tienen dispuestos dos componentes usando una arquitectura de contenedores, específicamente Docker [26]. Esto permite agilizar el proceso de despliegue y réplica de la implementación. El primer contenedor tiene la función de procesar el video stream, mientras que la función del segundo contenedor es publicar los resultados del procesamiento. En el contenedor 1, luego de la ingesta de datos multimedia se procesan los datos de entrada de acuerdo con la lógica del negocio; con la ejecución del algoritmo AIoT para la detección de uso de mascarillas en tiempo real. Posteriormente, los resultados de dicho procesamiento son almacenados; en este caso, las imágenes con las detecciones realizadas. Finalmente, se guardan los eventos realizados en un registro o «log». Mientras que, en el contenedor 2 se dispone de los resultados para que los usuarios finales puedan conectarse al servidor y consumir la información de interés, a través de un sitio Web.

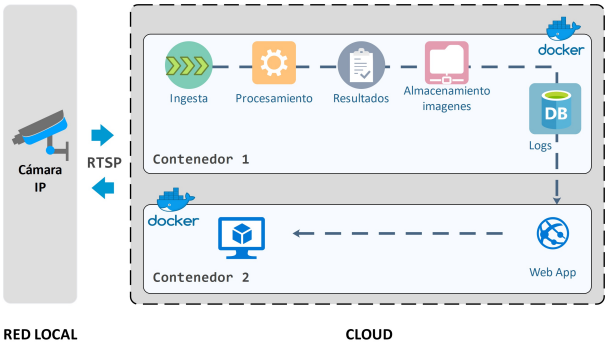


Figura 6. Arquitectura del modelo computacional Cloud

En la Figura 7 se evidencia los resultados de la inferencia en el modelo computacional Cloud.

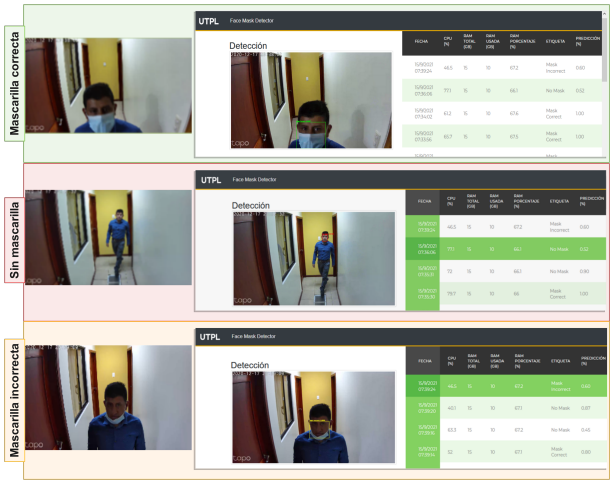


Figura 7. Resultados de la inferencia en el modelo computacional Cloud

2.5. Tecnologías utilizadas

La Tabla 1 muestra el resumen de las tecnologías consideradas en este trabajo. La primera columna identi-

fica la tecnología utilizada, la segunda columna corresponde al modelo computacional Cloud o Edge donde se aplica la tecnología; la tercera columna se señala en qué componente se va a usar. La última columna ofrece la justificación del uso de cada tecnología.

Tabla 1. Resumen de tecnologías usadas

Tecnología	Modelo	Componente	Justificación
JetPack SDK 4.5.1 [27]	Edge	Procesamiento	Incluye el paquete de controladores de la Jetson para Linux (L4T) con el sistema operativo Linux y las bibliotecas aceleradas por CUDA (TensorRT y cuDNN), para APIs de AI.
Python3 [28]	Edge Cloud	Procesamiento	Lenguaje base para el procesamiento de video, publicación de resultados, e integración con bibliotecas de visión e AI.
Deepstream SDK 5.1 [29]	Edge	Procesamiento	Este SDK de Nvidia proporciona un framework para construir aplicaciones de análisis de video aceleradas en la GPU que se ejecutan en la plataforma NVIDIA Jetson Nano.
Deepstream-services-library [30]	Edge	Procesamiento	Biblioteca de Python3 requerida para hacer uso de las funcionalidades del deepstream SDK.
MySQLdb [31]	Edge	Resultados	Proporciona la API de Python3 para acceder al servidor de bases de datos MySQL.
rclone [32]	Edge	Resultados	Requerido para gestionar archivos en el almacenamiento en la nube.
Docker [33]	Cloud	Procesamiento	Permite automatizar el despliegue de la aplicación en los contenedores ubicados en la nube.
Camgear [34]	Cloud	Procesamiento	Es un marco de procesamiento de video de alto rendimiento, multiplataforma que permite procesar video en tiempo real.
OpenCV [35]	Cloud	Procesamiento	Biblioteca que permite procesar visión artificial en Python3. Esta facilita en primera instancia la lectura del modelo entrenado de YOLOv3-tiny, y luego en la detección por cada frame procesado.
Firebase [36]	Cloud	Procesamiento	Permite almacenar los resultados de las detecciones en tiempo real.
Flask [37]	Cloud	Resultados	Framework que se utiliza para la creación de la aplicación web.
Bootstrap [38]	Cloud	Resultados	Biblioteca para el diseño de la aplicación que permite publicar los resultados.

2.6. Configuración experimental

Los procesos de etiquetación de imágenes y división del conjunto de datos personalizados a emplear en el entrenamiento del algoritmo detector de mascarillas se realiza, mediante el lenguaje de programación Python3.

El dataset fue dividido con una distribución de 80 % (716 imágenes) para entrenamiento y el 20 % (179 imágenes) restante para pruebas (ver Figura 8). Las imágenes del conjunto de datos personalizado, para el entrenamiento contienen la detección de una persona o varias personas dentro de la misma. Lo que representa para el entrenamiento 3070 personas con mascarilla correcta, 675 sin mascarilla y 113 con mascarilla incorrecta.

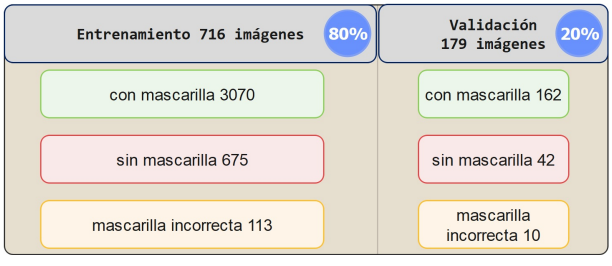


Figura 8. Distribución del conjunto de datos para entrenamiento y validación

La herramienta Google Colab [39] fue utilizada para ejecutar el proceso de entrenamiento del algoritmo AIoT. A este proceso se le provee de tres entradas: el 80 % de las imágenes del conjunto de datos, el archivo de configuración YOLOv3-tiny (ver en la Tabla 2 los parámetros de configuración), y el archivo de pesos de YOLOv3-tiny preentrenado con el dataset COCO [40].

Tabla 2. Valores de configuración para el entrenamiento de la red neuronal

Detalle	Cálculo	Valor
Número de clases	Mascarilla correcta, sin mascarilla y mascarilla incorrecta	classes=3
Tamaño máximo de batch	30000 (10000 por cada clase)	max_batches = 30000
Dimensiones de la imagen de entrada	ancho × alto = 416 × 416	width=416 height=416
Número de filtros en la capa convolucional antes de la capa YOLOv3-tiny	(classes+5)*3 = 24	filters=24
Tamaño de batch y sus subdivisiones para el entrenamiento		batch=64 subdivisions=2

El proceso de entrenamiento duró 16 horas, obteniendo una media de la precisión promedio (mAP) de 75.95 %. Finalmente, se validó la red neuronal entrenada con el 20 % restante del dataset (79 imágenes), esto representa 162 personas con mascarilla correcta, 42 sin mascarilla y 10 con mascarilla incorrecta. En la Tabla 3 se muestra los resultados correspondientes a la validación.

Tabla 3. Resultados de la validación del algoritmo de detección

Clase	Precisión
Mascarilla correcta	85,97 %
Sin mascarilla	68,72 %
Mascarilla incorrecta	73,15 %
mAP	**73,15 %**

3. Resultados y discusión

En esta sección se presentan los resultados de los diferentes experimentos en dos escenarios reales para determinar el rendimiento de los dos modelos computacionales. En este contexto, dos ambientes fueron seleccionados. El primer escenario *indoor* corresponde al ingreso de personas a una iglesia de la localidad para participar en la eucaristía dominical. Para el segundo escenario outdoor se aplicó en una calle con un alto índice de circulación de peatones, por turismo religioso al Santuario de El Cisne entre agosto y septiembre, ubicado en la provincia de Loja, sur del Ecuador.

En los dos ambientes indoor y *outdoor* se monitoreó por el lapso de una hora, de los cuales para un análisis más detallado se toman los 15 primeros minutos para el análisis cuantitativo de los resultados. En este contexto, se observó el ingreso de 82 personas en *indoor* y la circulación de 229 en outdoor. En la Figura 9 ilustran el total de detecciones en los ambientes *indoor* y outdoor.

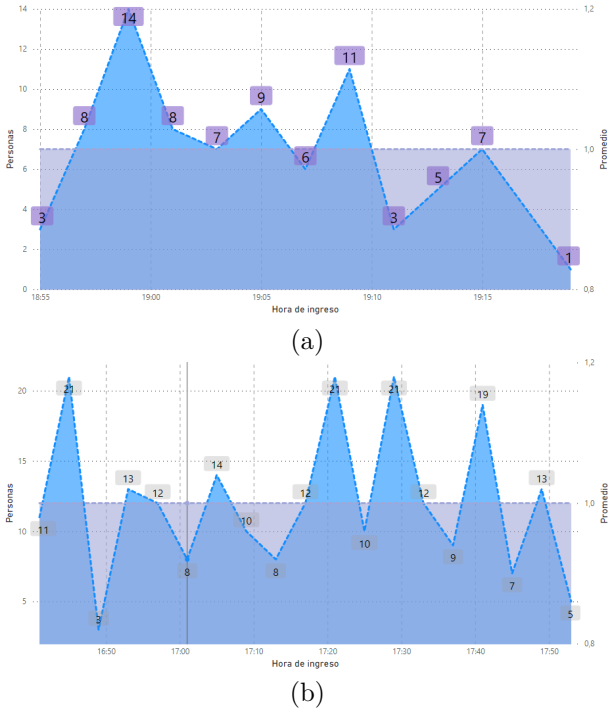


Figura 9. Flujo de personas: a) indoor y b) *outdoor*

Los resultados del experimento demuestran que el flujo de personas en el ambiente outdoor duplicó al de *indoor*. Teniendo como promedio un ingreso de 6 personas en el ambiente interno y circulación de 12 personas en el ambiente externo. El conteo fue realizado a través de la herramienta Camlytics. La Figura 10 muestra ejemplos de detecciones en los escenarios reales *indoor* y outdoor.

3.1. Métricas de evaluación

Para evaluar el desempeño del algoritmo AIoT en los modelos computacionales Cloud y Edge, en el caso de estudio «detección de uso de mascarillas de bioseguridad en tiempo real» se han considerado como métricas de evaluación:

- Confiabilidad del algoritmo de detección: Según [16], [18], [41–43], para determinar el rendimiento del algoritmo de detección se basa en cinco métricas a sabe: exactitud, precisión, sensibilidad, media armónica, media de la precisión promedio.
- Recursos computacionales: En función de lo presentado por [42] y el estándar ISO/IEC 25023, se recomienda analizar los recursos que ocupa la aplicación, analizando el tráfico de red, memoria RAM, CPU y almacenamiento.
- Tiempo de respuesta: Es otro parámetro que nos permite medir el tiempo que tarda el sistema en responder según lo investigado por Ashouri *et al.* [42] y por el estándar ISO/IEC 25023.

Figura 10. Resultados en escenario real interior y exterior

3.1.1. Confiabilidad del algoritmo de detección

Para evaluar la confiabilidad de los clasificadores basados en ML (Machine Learning) se utiliza una matriz de confusión para saber qué tan efectivo es el sistema. De acuerdo con Aslanpour *et al.* [41], una matriz de confusión, también conocida como matriz de errores, es un método para resumir el rendimiento del resultado de un modelo de clasificación, mostrando el número de predicciones correctas e incorrectas. Cuatro métricas son calculadas a partir de la matriz de confusión [16], [18], [41–43]: la exactitud (A: Accuracy, ver en la Ecuación 1), la precisión (P: Precision, ver en la Ecuación 2), la sensibilidad o revocación (R: Recall, ver en la Ecuación 3), y la media armónica (f_β, ver en la Ecuación 4). Adicionalmente, debido a las implementaciones AIoT Edge y Cloud del presente experimento se identifica tres clases (MC: mascarilla correcta, SM: sin mascarilla y MI: mascarilla incorrecta), también es necesario calcular la media de la precisión promedio (mAP, ver en la Ecuación 5) a partir de la precisión promedio (AP) de cada clase.

$$A = \frac{TP+TN}{(TP+FP)+(TN+FN)} \quad (1)$$

$$P = \frac{TP}{TP+FP} \quad (2)$$

$$R = \frac{TP}{TP+FN} \quad (3)$$

$$f_\beta = \frac{(1+\beta^2)*(P*R)}{\beta^2*P+R} \quad (4)$$

$$mAP = \frac{AP_{MC}+AP_{SM}+AP_{MI}}{3} \quad (5)$$

En donde: TP (*True Positive*) cuando la observación es positiva y predicha como positiva; FN (*False Negative*) cuando la observación es positiva, pero predicha como negativa; TN (*True Negative*) cuando la observación es negativa y es predicha como negativa; y FP (*False Positive*) cuando la observación es negativa, pero es predicha como positiva. Mientras que β es usado para asignar diferentes pesos a las medidas utilizadas en la Ecuación 4 [43]. En el presente trabajo a β se le asignó un valor de 1. La Tabla 4 muestra los resultados obtenidos al implementar en escenarios reales los modelos Edge y Cloud. Para lo cual, se realiza una depuración de detecciones repetidas en cada uno de los ambientes.

Tabla 4. Matriz de confusión

Modelo Edge en el escenario *indoor*				
		Valor verdadero		
		MI	SM	MC
Predicción	MC	2	29	**191**
	SM	0	**36**	9
	MI	**3**	3	0
Modelo Edge en el escenario outdoor				
Predicción	MC	2	15	**170**
	SM	0	**52**	30
	MI	**1**	1	0
Modelo Cloud en el escenario *indoor*				
Predicción	MC	5	36	**165**
	SM	7	**29**	12
	MI	**1**	2	6
Modelo Cloud en el escenario outdoor				
Predicción	MC	3	45	**123**
	SM	2	**41**	40
	MI	**1**	2	2

En la Tabla 5 se muestran los resultados obtenidos después de evaluar las implementaciones Edge y Cloud para la detección del uso de mascarillas buconasales por parte de las personas en los dos ambientes *indoor* y outdoor, en tiempo y escenarios reales.

Tabla 5. Resultados obtenidos de A, P, R y f_β

Exactitud (A) %				
Clase	**Cloud**		**Edge**	
	indoor	outdoor	*indoor*	outdoor
MC	76,7	64,7	85,2	82,6
SM	77,4	65	84,9	82,9
MI	90,7	94,8	97,9	98,7
Media	81,6	74,8	89,3	88,1
Precisión (P) %				
MC	80	71,9	86	90
SM	60,4	49,4	80	63,4
MI	11,1	20	50	50
mAP	50,5	47,1	72	67,8
Revocación (R) %				
MC	90,1	74,5	95,5	85
SM	43,3	46,6	52,9	76,5
MI	7,6	16,7	60	33,3
Media	47	45,9	69,5	64,8
Media armónica (f_β) %				
MC	84,8	73,2	90,5	87,4
SM	50,4	48	63,7	67,9
MI	9	18,2	54,5	40
Media	48,1	46,5	69,6	65,1

La Tabla 4 muestra los porcentajes alcanzados con respecto a las métricas de exactitud, precisión, revocación y media armónica. La exactitud se refiere a la cantidad de predicciones positivas correctas; en este contexto, los resultados muestran que el modelo Edge es 10,5 % más exacto que el modelo Cloud. Por otro lado, la precisión se refiere al porcentaje de casos positivos detectados; los resultados denotan que la precisión en ambiente *indoor* es mayor a la obtenida en ambiente *outdoor* en ambos modelos. Así mismo, se observa que el modelo Edge es un 39,9 % más preciso que Cloud. La precisión de Cloud es inferior a la de Edge, ya que durante la transmisión algunos fotogramas se pierden o se distorsionan en algunos casos. Adicionalmente, la precisión de la clase mascarilla incorrecta no puede ser considerada totalmente válida pues no existen las suficientes muestras de esta clase en los dos ambientes, en su lugar la distorsión de las imágenes en el modelo Cloud aumenta en número de falsos positivos. Con respecto a la métrica de revocación, la cual se refiere a la proporción de casos positivos que son correctamente identificados por el algoritmo, los resultados indican que el modelo Edge predijo un 20,7 % más correctamente que Cloud. Finalmente, la métrica de media armónica, es usada cuando el dataset no es balanceado al proveer entradas de las diferentes clases al clasificador. En este contexto, los resultados muestran que al no haber obtenido suficientes detecciones de la clase «mascarilla incorrecta» afecta drásticamente en la distribución de los datos.

3.2. Uso de recursos

Entre los recursos que se analizaron, para la comparación de los dos modelos computacionales Cloud y Edge, de acuerdo con, [42] y el estándar ISO/IEC 25023, están el tráfico de la red, la utilización de memoria RAM, CPU y almacenamiento en disco. Debido a que los resultados denotaron un mejor desenvolvimiento del detector de mascarillas buconasales en el ambiente *indoor*, en las siguientes secciones, se realiza un análisis del empleo de recursos en este ambiente. Asimismo, la Tabla 6 muestra las características de componentes de hardware de los equipos en los cuales se implementa cada uno de los dos modelos, Cloud y Edge.

Tabla 6. Características de hardware de las implementaciones Cloud y Edge

Componente	Cloud	Edge
GPU	—	NVIDIA Maxwell architecture with 128 NVIDIA CUDA® cores
CPU	Intel Xeon Processor (Skylake, IBRS) / 8 nucleos / 2100 MHz	Quad-core ARM Cortex-A57 MPCore processor
Memoria RAM	31 GB virtual	4 GB 64-bit LPDDR4, 1600MHz 25.6 GB/s
Almacenamiento	246 GB	16 GB eMMC 5.1
Conectividad	Gigabit Ethernet	Gigabit Ethernet, Wi-Fi
Pantalla	—	HDMI 2.0 and eDP 1.4

3.2.1. Tráfico de red

Las medidas del tráfico de la red generadas por la transmisión de video desde la cámara IP en el modelo Cloud, e imágenes desde de la Jetson Nano en el modelo Edge, fueron tomadas mediante el analizador de protocolos *Wireshark* en la interfaz de entrada del servidor donde se aloja modelo Cloud y la plataforma web. Se tomaron medidas de tráfico utilizando como parámetros de codificación de video a 1008 Kbps, observando el comportamiento del tráfico en volumen de Mbps. En la Figura 11, se puede observar el tráfico de red generado, la curva superior de color azul representa el tráfico desde la cámara IP al modelo computacional Cloud mediante el protocolo RSTP, y la curva inferior de color verde representa el tráfico generado por la transmisión de imágenes procesadas con el algoritmo de detección en el modelo Edge a la plataforma web (medido en Mbps). Dicha información está representada en forma de series a través del tiempo, la misma que corresponde a los primeros 15 minutos de monitoreo en el ambiente indoor.

Como resultado se puede evidenciar que el tráfico de la red es significativamente superior en el modelo Cloud en comparación al generado en el modelo Edge. La media de tráfico de red en Cloud es de 0,86 Mbps, con un máximo de 4,91 Mbps; mientras que la media de tráfico de red en Edge es de 0.07 Mbps, con un máximo de 0,26 Mbps. Esto se debe a que en el modelo Cloud se transmiten todos los fotogramas del sensor visual, sin hacer previamente un filtro de información; mientras que en el modelo Edge únicamente se transmite imágenes cuando existe una detección, lo que evita la saturación de la red.

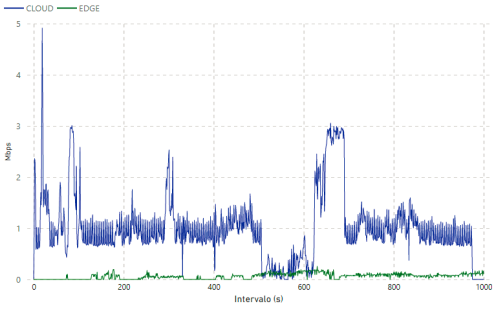


Figura 11. Tasa de transferencia de video al Cloud e imágenes procesadas desde el Edge a la plataforma web

3.2.2. Memoria RAM

Los modelos computacionales Cloud y Edge se caracterizan por la diferencia en la cantidad de recursos disponibles. Particularmente, en las implementaciones de este trabajo se dispuso de 4 GB de RAM en el modelo Edge y de 31 GB de RAM en el modelo Cloud. Los resultados indican (ver Figura 12) que el uso de RAM en el modelo Edge fluctuó entre 2,41 GB como mínimo, 2,43 GB como máximo, con una media de 2,42 GB. Mientras que en el modelo Cloud, el uso de RAM fluctuó entre 10,7 GB como mínimo, 12 GB como máximo, con una media de 11,55 GB. A partir de los datos obtenidos, se puede interpretar que, una vez designados los recursos al proceso YOLO, no existe mayor crecimiento durante la ejecución de las detecciones tanto en Cloud como Edge.

3.2.3. CPU

Con respecto a la capacidad de procesamiento de los modelos computacionales Cloud y Edge se diferencian notoriamente. A partir de la Figura 13, se determina que el modelo Edge con el Jetson Nano integra un procesador de 1,43 GHz; de los cuales el uso de la CPU durante los 15 primeros minutos de procesamiento fluctuó entre el 22,95 % como mínimo, el 62,01 % como máximo, con una media de 29,59 %. Mientras que en el modelo Cloud, el servidor contaba con un procesador de 4.8 GHz; de los cuales el uso de la CPU fluctuó entre 1,75 % como mínimo, 7,49 % como máximo, con una media de 3,31 %. Por lo tanto, se puede interpretar que debido a que el modelo Edge tiene menos recursos, el esfuerzo es mayor al momento de procesar las detecciones. Inversamente a ello, debido a que el modelo Cloud cuenta con mejores recursos el esfuerzo es mínimo al ejecutar el detector de uso de mascarillas buconasales.

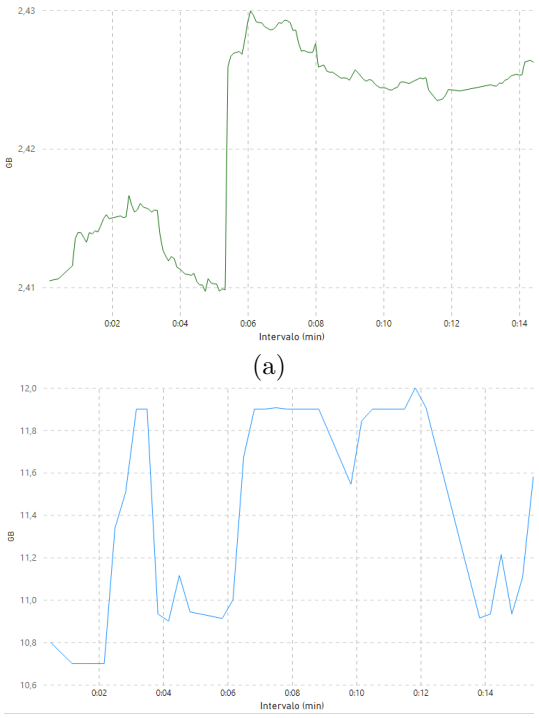


Figura 12. Uso de memoria RAM durante los primeros 15 minutos de monitoreo en el ambiente *indoor*: a) Modelo Edge y b) Modelo Cloud

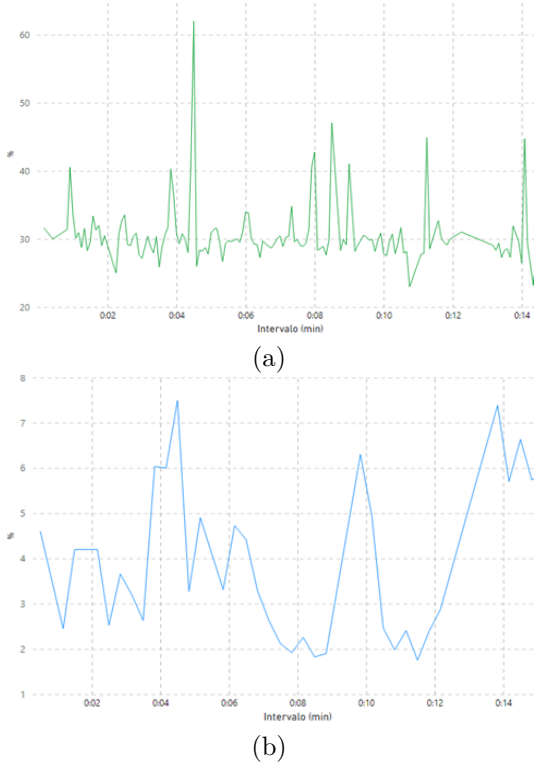


Figura 13. Uso de CPU durante los primeros 15 minutos de monitoreo en el ambiente *indoor*: a) Modelo Edge y b) Modelo Cloud

3.2.4. Almacenamiento

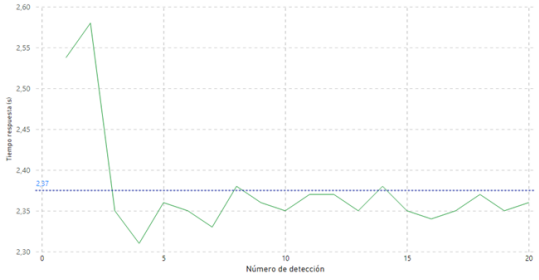
El empleo de espacio en disco de las detecciones crece exponencialmente en ambos modelos Cloud y Edge. En Cloud se puede incrementar este recurso en el caso de demandar un mayor espacio, mediante una actualización del contrato de alquiler del servidor en la nube. Mientras que, en Edge no es posible incrementar la capacidad de almacenamiento, ya que de naturaleza es limitado. Los resultados demuestran que el uso de espacio en disco es similar en ambos modelos Cloud y Edge, aproximadamente se ocupó 90,7 MB en el ambiente *indoor* y 200,5 MB en el ambiente outdoor. Es importante denotar que el detector de mascarillas buconasales almacena solamente el cuadro de la detección, no el contexto o la imagen completa, por lo que la capacidad de almacenamiento es optimizada.

3.3. Tiempo de respuesta

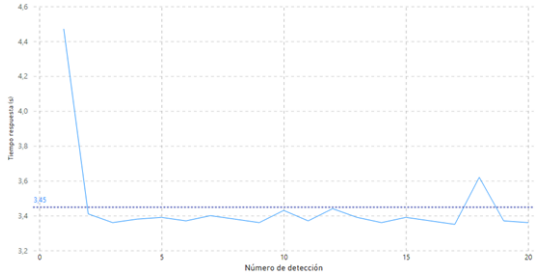
De acuerdo con Ashouri *et al.* [42] y el estándar ISO/IEC 25023, el tiempo de respuesta de un sistema es otra métrica que nos permite evaluar el desempeño de un sistema. Particularmente, en el presente trabajo, para obtener el valor del tiempo de respuesta se utilizaron los tiempos de eventos marcados en los archivos de log, y de esta manera identificar con exactitud el instante que ingresó la imagen al modelo, para luego calcular el tiempo transcurrido hasta su publicación.

En este contexto, para calcular el tiempo de respuesta en la implementación Edge, se consideró el lapso transcurrido entre el instante que se produce la detección hasta que el sistema termina de emitir la alerta sonora. La Figura 14a muestra gráficamente los tiempos de respuesta registrados en el modelo Edge durante las primeras veinte detecciones en el ambiente *indoor*, obteniendo un promedio de 2,37 segundos de respuesta.

Por otro lado, para calcular el tiempo de respuesta en el modelo Cloud se debe medir desde que se produce la captura en el sensor visual hasta la publicación de los resultados en la página web. Sin embargo, lo que es posible capturar con exactitud es el lapso de tiempo que se produce desde la detección en Cloud hasta su publicación en el portal; por lo que, en el tiempo Cloud se agregó un valor medio de 2 segundos a partir de la tasa de transferencia para una imagen de 300 kB con una resolución de 1920 × 1080, que corresponde al tiempo que le toma al sensor visual transmitir el video a la nube, el cual es indirectamente proporcional al ancho de banda proporcionado por la red. El ancho de banda con el que se contó para la implementación del modelo Cloud fue de 25 Mbps. La Figura 14b muestra gráficamente los tiempos de respuesta registrados en el modelo Cloud durante las primeras veinte detecciones en el ambiente *indoor*, logrando un promedio de 3,45 segundos de respuesta.



(a)



(b)

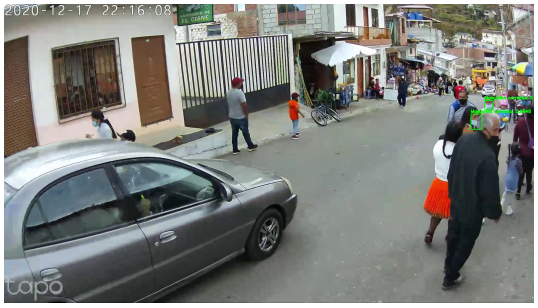
Figura 14. Tiempos de respuesta: a) Modelo Edge y b) Modelo Cloud

3.4. Discusión de resultados

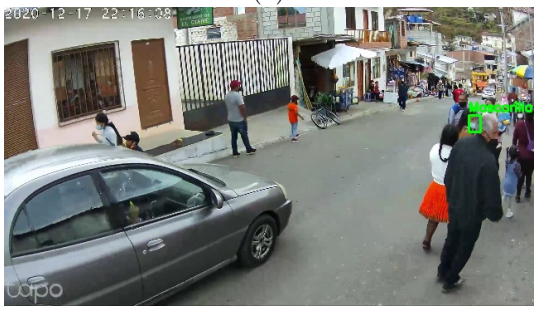
Después de haber desplegado las implementaciones Cloud y Edge en escenarios reales, a nivel general se identifica que el algoritmo AIoT (especializado en la identificación del uso de mascarillas de bioseguridad) tuvo un mejor desempeño en el ambiente *indoor* en los dos modelos, debido al flujo controlado (en una dirección) y exclusivo de personas que constituía el ingreso a la iglesia. Mientras que el desempeño del mismo algoritmo AIoT fue menos eficiente en el ambiente *outdoor* en ambas implementaciones, debido al flujo de personas no controlado (en varias direcciones) y no exclusivo de personas; otros objetos como automóviles, bicicletas, e incluso animales, formaron parte de las imágenes capturadas por el sensor visual. Con la implementación de algoritmos AIoT para la identificación de objetos en tiempo real, el modelo computacional Edge presentó un mejor desempeño en comparación con el modelo Cloud.

YOLO constituye una herramienta recomendable para implementar soluciones que utilizan AI en escenarios de la vida real, los cuales requieran de acciones inmediatas. Particularmente, en los resultados del presente trabajo arrojan métricas de exactitud de 78,2 % en el modelo Cloud y de 88,7 % en Edge. Aunque existe un descenso en la precisión del clasificador con un 48,8 % en Cloud y 69,9 % en Edge, se pudo evidenciar que la precisión en ambos modelos baja especialmente en el escenario *outdoor*, se debe al hecho que la red neuronal no fue lo suficientemente entrenada para identificar objetos pequeños. Adicionalmente, se notó que el detector también decae cuando existen múltiples personas en un mismo fotograma a ser detectadas en ambos modelos, para superar este inconveniente se podría incluir en el dataset de entrenamiento un mayor número de imágenes que contengan múltiples detecciones.

La calidad del flujo de información multimedia, entrada para el procesamiento de video en las implementaciones Cloud y Edge, incide en el desempeño del algoritmo AIoT. Se probó al detector de mascarillas en tiempo real con dos resoluciones del sensor visual, alta (1920 × 1080) y baja (640 × 360). Lo que permitió llegar a la conclusión de que a una la resolución baja, la detección presenta fallos u omisión en objetos pequeños; sin embargo, los objetos grandes (cercanos) son correctamente identificados en tiempo real (Figura 16), ya que el procesamiento del fotograma lo hace con fluidez.



(a)



(b)

Figura 15. Detección de mascarilla en diferentes resoluciones: a) 1920 × 080 y b) 640 × 360

Es necesario evaluar la relación entre la velocidad del flujo de video (fotogramas por segundo) y su relevancia en el contexto del negocio de la solución AIoT. Particularmente, en el caso de estudio de la presente investigación, detector de uso de mascarillas en tiempo real, se pudo evidenciar que no era necesario evaluar todos los fotogramas de la entrada. En primera instancia el algoritmo AIoT procesaba todos los fotogramas receptados (15 fotogramas por segundo), y como salida se obtuvo demasiadas detecciones pertenecientes a la misma persona. Por lo que se llegó a la conclusión que era necesario filtrar la cantidad de fotogramas de entrada al detector (un fotograma por segundo) y de esta manera evitar la saturación del procesador. En otras palabras, los recursos que se demanda para

procesar cada uno de los fotogramas son innecesarios. Para solventar este inconveniente se disminuye los fotogramas por segundo, la cual consiste en procesar un determinado fotograma por cada cierto intervalo. Como resultado se alcanzó una optimización de recursos de procesamiento al filtrar la entrada y de almacenamiento en la salida, ya que el número de detecciones de la misma persona bajó considerablemente, por ejemplo, de un 1 GB a 100 MB.

Finalmente, luego de haber experimentado con ambos modelos Cloud y Edge, se identifican varios desafíos para futuros trabajos. Existe incongruencia entre la velocidad de detección de los algoritmos de clasificación basados en Machine Learning (ML) y las tecnologías que permiten el almacenamiento y administración de actuadores en la red. Por ejemplo, el detector de uso de mascarillas de bioseguridad trabaja con una velocidad de 15 detecciones por segundo, pero al momento de querer almacenar esta información, la transacción tarda una detección por segundo, provocando de esta manera un cuello de botella. Los gestores de bases de datos actuales no alcanzan la velocidad que se requiere al trabajar con clasificadores basados en AI . Similarmente ocurre con los actuadores visuales o auditivos, es en vano que el algoritmo AIoT detecte múltiples objetos simultáneamente si el tiempo que le toma al parlante o pantalla mostrar esa información al público demanda de un segundo por objeto.

3.5. Evaluación del sistema en un ambiente real

Para evaluar la eficacia y la viabilidad del método propuesto, en esta parte presentamos un prototipo de sistema equipado con YOLOv3-tiny, que puede desplegarse en las entradas de los lugares públicos. El prototipo del sistema basado en el modelo computacional Edge se ilustra en la Figura **??** con la integración e implementación de los distintos componentes de hardware y software, incluyendo una cámara IP, un computador Jetson Nano, una pantalla con una interfaz HDMI.

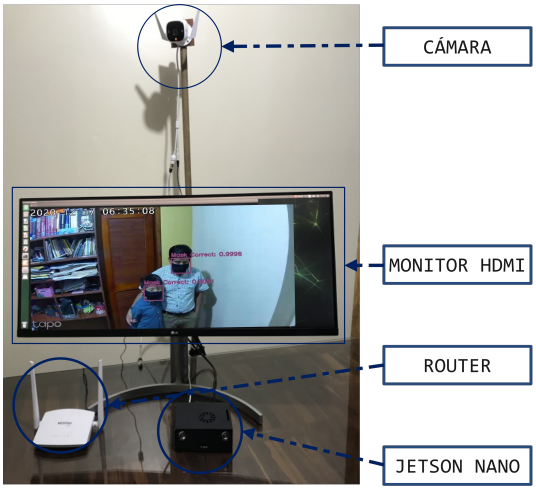


Figura 16. Implementación del sistema

En la Figura 5 se puede evidenciar el funcionamiento del prototipo para los tres casos (mascarilla correcta, sin mascarilla y mascarilla incorrecta). Así mismo, en los escenarios de evaluación las alertas visuales no captaron la atención de las personas cuando el algoritmo AIoT proporciona la retroalimentación de la detección. Inicialmente cuando las personas atravesaban el ángulo de visión de la cámara, en la pantalla se mostraba la salida de los cuadros delimitadores etiquetados y pintados de acuerdo con la clase detectada. Se evidenció que el propósito de la pantalla no era claro para el transeúnte, por lo que algunos saludaban o salían del área de acción de la cámara. Esto cambió, al agregar las alertas auditivas que capturaron más la atención de las personas. Los usuarios escuchaban los siguientes mensajes: «uso de mascarilla correcta», «mascarilla no detectada», o «uso de mascarilla incorrecta»; dependiendo de la clase de detección realizada por el algoritmo, los peatones identificaron el propósito del despliegue intuitivamente. Muchos de ellos corrigieron el mal uso o ausencia de mascarilla de bioseguridad.

4. Conclusiones

En este trabajo se ha diseñado, implementado y evaluado un sistema para la detección de mascarillas en contextos reales. Donde el desempeño del algoritmo de detección de objetos basados en AIoT, el modelo computacional Edge superó al modelo Cloud, con un 39,9 % en la métrica de precisión, que concierne al porcentaje de casos positivos detectados; y con un 10,5 % en la métrica de exactitud, la cual se refiere al porcentaje de predicciones positivas que fueron correctas. Adicionalmente, existen ventajas y desventajas inherentes a las características intrínsecas de cada modelo. Una de las ventajas de las implementaciones Cloud es que se caracterizan por disponer de altas capacidades de almacenamiento y procesamiento; y en el caso de requerir incrementar sus recursos de memoria RAM, disco, o procesamiento, el modelo ofrece la característica de escalabilidad al sistema. Sin embargo, la gran desventaja del modelo Cloud es el alto consumo de la red, ya que se transmite la secuencia de fotogramas en video sin previamente realizar una depuración de la información.

Mientras que el modelo Edge, a pesar de disponer de recursos limitados, esta desventaja es compensada con el hecho que es posible hacer una depuración de información, lo que permite a las aplicaciones ajustarse a la cantidad de recursos disponibles de RAM, procesamiento y ancho de banda. En donde se pudo evidenciar que, luego de ser asignados los recursos, el consumo de estos no fluctúa drásticamente. Además, en este lado de la red, se puede usar actuadores visuales o auditivos para una interacción similar a la humana

en el contexto en donde se esté usando el modelo.

La aplicación de YOLOv3 constituye una opción adecuada para implementaciones especializadas en detección de objetos en tiempo real ya sea en el modelo computacional Cloud o Edge, YOLOv3 por tratarse de un modelo detector de objetos de una sola etapa, clasifica y predice directamente el objetivo en cada ubicación de la imagen original completa. Todas estas características distinguen a YOLOv3 de otros modelos detectores, sin embargo, las métricas de exactitud y precisión son alcanzadas cuando se efectúa un proceso de entrenamiento con un amplio conjunto de datos. El mismo que debe contener múltiples detecciones en la misma imagen, las cuales deben ser capturadas en diferentes contextos y escalas. Así mismo, se puede diversificar el dataset a través de la utilización de técnicas de aumentación de imágenes.

Finalmente, como trabajo futuro se plantea el análisis del sistema en la persuasión a las personas para que usen mascarillas. También, se plantea la integración de sensores adicionales al sistema que permitan analizar la calidad del aire en especial en ambientes *indoor*.

Agradecimientos

Los autores queremos agradecer a la Universidad Técnica Particular de Loja (UTPL) por financiar este trabajo por medio de la segunda convocatoria para el «FINANCIAMIENTO DE TRABAJOS DE FIN DE TITULACIÓN Y MÁSTER 2020».

Referencias

[1] R. Aragón Nogales, I. Vargas Almanza, and M. G. Miranda Novales, "COVID-19 por SARS-CoV-2: la nueva emergencia de salud," *Revista Mexicana de Pediatría*, vol. 86, pp. 213–218, 2020. [Online]. Available: https://dx.doi.org/10.35366/91871

[2] WHO, "Listings of WHO's response to COVID-19," World Health Organization. [Online]. Available: https://bit.ly/3mAZ6LH

[3] ——, "Vías de transmisión del virus de la COVID-19: Repercusiones para las recomendaciones relativas a las precauciones en materia de prevención y control de las infecciones." [Online]. Available: https://bit.ly/3epu4Sq

[4] OMS, "Who coronavirus (COVID-19) dashboard," 2021. [Online]. Available: https://bit.ly/3mDAO3r

[5] OPS, "Vacunas contra la COVID-19," 2020. [Online]. Available: https://bit.ly/3z0JGFs

[6] H. Ritchie, E. Mathieu, L. Rodés-Guirao, C. Appel, C. Giattino, E. Ortiz-Ospina, J. Hasell, B. Macdonald, D. Beltekian, M. Roser, and et al., "Coronavirus (covid-19) vaccinations - statistics and research," 2020. [Online]. Available: https://bit.ly/3sEmtro

[7] C. Costa and C. Tombesi, "COVID-19: Cuánto tiempo se demoró en encontrar la vacuna para algunas enfermedades (y por qué este coronavirus es un caso histórico)," 2020. [Online]. Available: https://bbc.in/3pEV0Eh

[8] "Comparative research grant," *Anthropology News*, vol. 36, no. 8, pp. 43–43, 1995. [Online]. Available: https://anthrosource.onlinelibrary.wiley.com/doi/abs/10.1111/an.1995.36.8.43.1

[9] S. S. Bibak Sareshkeh, E. Magli, and P. Dal Zovo, "Combined ict technologies for supervision of complex operations in resilient communities," Master's thesis, 2020. [Online]. Available: https://bit.ly/3HaioPE

[10] I. Santos-González, A. Rivero-García, J. Molina-Gil, and P. Caballero-Gil, *Implementation and Analysis of Real-Time Streaming Protocols*, vol. 17, no. 4, 2017. [Online]. Available: https://doi.org/10.3390/s17040846

[11] A. Nurrohman and M. Abdurohman, "High performance streaming based on h264 and real time messaging protocol (rtmp)," in *2018 6th International Conference on Information and Communication Technology (ICoICT)*, 2018, pp. 174–177. [Online]. Available: https://doi.org/10.1109/ICoICT.2018.8528770

[12] S. Basu, "What are video streaming codecs & container formats: Muvi live server," 2020. [Online]. Available: https://bit.ly/3ErJPCZ

[13] J. S. Katz, "Aiot: Thoughts on artificial intelligence and the internet of things," *IEEE Internet if Things*, 2019. [Online]. Available: https://bit.ly/3sBwGEZ

[14] J. Redmon and A. Farhadi, "Yolov3: An incremental improvement," *ArXiv*, vol. abs/1804.02767, 2018. [Online]. Available: https://bit.ly/3psJLyp

[15] A. M. Porcelli, "La inteligencia artificial y la robótica: sus dilemas sociales, éticos y jurídicos," *Derecho global. Estudios sobre derecho y justicia*, vol. 6, pp. 49–105, 2020. [Online]. Available: https://doi.org/10.32870/dgedj.v6i16.286

[16] X. Jiang, T. Gao, Z. Zhu, and Y. Zhao, "Real-time face mask detection method based on yolov3," *Electronics*, vol. 10, no. 7, p. 837, 2021. [Online]. Available: https://doi.org/10.3390/electronics10070837

[17] S. Sethi, M. Kathuria, and T. Kaushik, "Face mask detection using deep learning: An approach to reduce risk of coronavirus spread," *Journal of Biomedical Informatics*, vol. 120, p. 103848, 2021. [Online]. Available: https://doi.org/10.1016/j.jbi.2021.103848

[18] D. Gonzalez Dondo, J. A. Redolfi, R. G. Araguás, and D. Garcia, "Application of deep-learning methods to real time face mask detection," *IEEE Latin America Transactions*, vol. 19, no. 6, pp. 994–1001, 2021. [Online]. Available: https://bit.ly/3pw7DkM

[19] S. Sethi, M. Kathuria, and T. Kaushik, "A real-time integrated face mask detector to curtail spread of coronavirus," *Computer Modeling in Engineering & Sciences*, vol. 127, no. 2, pp. 389–409, 2021. [Online]. Available: https://doi.org/10.32604/cmes.2021.014478

[20] I. Vich, "Medical masks dataset images tfrecords," Kaggle, 2020. [Online]. Available: https://bit.ly/3er0tb8

[21] S. Ge, J. Li, Q. Ye, and Z. Luo, "Mafa," 2018. [Online]. Available: https://bit.ly/3FBC52o

[22] S. Yadav and S. Shukla, "Analysis of k-fold cross-validation over hold-out validation on colossal datasets for quality classification," in *2016 IEEE 6th International Conference on Advanced Computing (IACC)*, 2016, pp. 78–83. [Online]. Available: https://doi.org/10.1109/IACC.2016.25

[23] E. Allibhai, "Holdout vs. cross-validation in machine learning." 2018. [Online]. Available: https://bit.ly/3z2TbE0

[24] F. Zhuang, Z. Qi, K. Duan, D. Xi, Y. Zhu, H. Zhu, H. Xiong, and Q. He, "A comprehensive survey on transfer learning," *Proceedings of the IEEE*, vol. 109, no. 1, pp. 43–76, 2021. [Online]. Available: https://doi.org/10.1109/JPROC.2020.3004555

[25] R. K. Indla, "An overview on amazon rekognition technology," 2021.

[26] L. Herrera-Izquierdo and M. Grob, "A performance evaluation between docker container and virtual machines in cloud computing architectures," *Maskana*, vol. 8, pp. 127–133, 2017. [Online]. Available: https://bit.ly/3z12oNf

[27] NVIDIA, "Jetpack sdk 4.5.1 archive," 2021. [Online]. Available: https://bit.ly/32BxzT1

[28] Python, "Welcome to python.org," 2021. [Online]. Available: https://bit.ly/3qqTd4Q

[29] NVIDIA, "Quickstart guide - deepstream 6.0 release documentation," 2021. [Online]. Available: https://bit.ly/3sDTa8s

[30] ProminenceAI, "Prominenceai/deepstream-services-library: A shared library of on-demand deepstream pipeline services for python and c/c++," GitHub. [Online]. Available: https://bit.ly/3pyxM2y

[31] MongoDB, "The application data platform," MongoDB. [Online]. Available: https://bit.ly/3qrRsUL

[32] N. Craig-Wood, "Rclone syncs your files to cloud storage," 2014. [Online]. Available: https://bit.ly/3JlPNsu

[33] Docker, "Empowering app development for developers," 2020. [Online]. Available: https://www.docker.com/

[34] A. Thakur, C. Clauss, C. Hollinger, V. Boivin, B. Lowe, M. Schoentgen, and R. Bouckenooghe, "abhitronix/vidgear: Vidgear v0.2.3," Oct. 2021. [Online]. Available: https://doi.org/10.5281/zenodo.5602375

[35] OpenCV. (2021) Opencv courses holiday sale. [Online]. Available: https://bit.ly/3ezvAS1

[36] Google Developers, "Firebase," 2020. [Online]. Available: https://bit.ly/3JinCeh

[37] Pallets, "Flask web development, one drop at a time," Pallet, 2010. [Online]. Available: https://bit.ly/3Hemy9h

[38] J. T. Mark Otto. (2021) Build fast, responsive sites with bootstrap. [Online]. Available: https://bit.ly/32Nl5rK

[39] Google. (2021) Colaboratory. Google Research. [Online]. Available: https://bit.ly/3EC3mk0

[40] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick, "Microsoft coco: Common objects in context," in *Computer Vision – ECCV 2014*, D. Fleet, T. Pajdla, B. Schiele, and T. Tuytelaars, Eds. Springer International Publishing, 2014, pp. 740–755. [Online]. Available: https://bit.ly/3sxpZUu

[41] M. S. Aslanpour, S. S. Gill, and A. N. Toosi, "Performance evaluation metrics for cloud, fog and edge computing: A review, taxonomy, benchmarks and standards for future research." *Internet of Things*, vol. 12, p. 100273, 2020. [Online]. Available: https://doi.org/10.1016/j.iot.2020.100273

[42] M. Ashouri, F. Lorig, P. Davidsson, and R. Spalazzese, "Edge computing simulators for iot system design: An analysis of qualities and metrics," *Future Internet*, vol. 11, no. 11, p. 235, 2019. [Online]. Available: https://doi.org/10.3390/fi11110235

[43] F. Oliveira-Teixeira, T. P. Donadon-Homem, and A. Pereira-Junior, "Aplicación de inteligencia artificial para monitorear el uso de mascarillas de protección," *Revista Científica General José María Córdova*, vol. 19, no. 33, pp. 205–222, 2021. [Online]. Available: https://doi.org/10.21830/19006586.725