"Desarrollo de un prototipo para detección de mascarillas y control de aforo de personas en espacios cerrados"

Zapata Hidalgo, Andrés Miguel

Departamento de Eléctrica, Electrónica y Telecomunicaciones

Carrera de Ingeniería en Electrónica, Automatización y Control

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

Ing. Silva Tapia, Rodrigo

10 de febrero del 2022

Trab_Titulacion_AZapata final.pdf

Scanned on: 22:48 February 10, 2022 UTC

Identical Words	326
Words with Minor Changes	106
Paraphrased Words	237
Ommited Words	0

Ing. Silva Tapia, Rodrigo
C.I: 0602199523

DEPARTAMENTO DE ELÉCTRICA, ELECTRÓNICA Y TELECOMUNICACIONES

CARRERA DE INGENIERÍA EN ELECTRÓNICA, AUTOMATIZACIÓN Y CONTROL

CERTIFICACIÓN

Certifico que el trabajo de titulación, "**Desarrollo de un prototipo para detección de mascarillas y control de aforo de personas en espacios cerrados**" fue realizado por el señor **Zapata Hidalgo, Andrés Miguel** el cual ha sido revisado y analizado en su totalidad por la herramienta de verificación de similitud de contenido; por lo tanto cumple con los requisitos legales, teóricos, científicos, técnicos y metodológicos establecidos por la Universidad de las Fuerzas Armadas ESPE, razón por la cual me permito acreditar y autorizar para que lo sustente públicamente.

Sangolquí, 09 de febrero de 2022

Firmado electrónicamente por:
RODRIGO SILVA TAPIA

Ing. Silva Tapia, Rodrigo

C.I 0602199523

DEPARTAMENTO DE ELÉCTRICA, ELECTRÓNICA Y TELECOMUNICACIONES

CARRERA DE INGENIERÍA EN ELECTRÓNICA, AUTOMATIZACIÓN Y CONTROL

RESPONSABILIDAD DE AUTORÍA

Yo, **Zapata Hidalgo, Andrés Miguel** con cédula de ciudadanía n°1723015499, declaro que el contenido, ideas y criterios del trabajo de titulación**: Desarrollo de un prototipo para detección de mascarillas y control de aforo de personas en espacios cerrados** es de mi 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, 09 de febrero de 2022

Firma

Zapata Hidalgo, Andrés Miguel

C.C.: 1723015499

DEPARTAMENTO DE ELÉCTRICA, ELECTRÓNICA Y TELECOMUNICACIONES

CARRERA DE INGENIERÍA EN ELECTRÓNICA, AUTOMATIZACIÓN Y CONTROL

AUTORIZACIÓN DE PUBLICACIÓN

Yo **Zapata Hidalgo, Andrés Miguel** con cédula de ciudadanía n°1723015499, autorizo a la Universidad de las Fuerzas Armadas ESPE publicar el trabajo de titulación: **Desarrollo de un prototipo para detección de mascarillas y control de aforo de personas en espacios cerrados** en el Repositorio Institucional, cuyo contenido, ideas y criterios son de mi responsabilidad.

Sangolquí, 09 de febrero de 2022

Firma

Zapata Hidalgo, Andrés Miguel

C.C.: 1723015499

Dedicatoria

Dedico este trabajo de manera especial a mis padres Miguel Zapata y Fanny Hidalgo por su incesante esfuerzo, su incondicional apoyo y por ser mi mayor fuente de inspiración para haber llegado hasta este punto.

A mis hermanos Daniel y Erika por haber demostrado un gran apoyo durante toda mi carrera en todos los aspectos posibles, y a la vez por servir como ejemplo de esfuerzo y dedicación.

ANDRÉS M. ZAPATA HIDALGO

.

Agradecimientos

Mi principal agradecimiento a mis padres por todas las enseñanzas que me han dado a lo largo de mi vida y que me han servido para lograr absolutamente todos mis propósitos; por su ejemplo de humildad, esfuerzo y responsabilidad que sin duda es mi mayor herencia; por nunca haber soltado mi mano durante toda mi carrera incluso cuando hubo momentos difíciles, en donde fue cuando más recibí su apoyo y su motivación y por ser la luz que guía mi camino.

A mis hermanos que fueron una parte fundamental durante todo este proceso, por su preocupación, por su apoyo en todo sentido y darme el impulso para seguir adelante cuando en realidad lo necesitaba.

A todas las personas que conocí durante este trayecto y que sin duda alguna trajeron con su presencia enseñanzas y experiencias que siempre quedaran guardadas y de manera especial a los amigos que permanecen a pesar del tiempo y que son aquellos que se convierten en familia.

A mi tutor, por ser una verdadera guía y un gran apoyo para que la realización de este proyecto haya sido posible.

Finalmente quiero agradecer a la Universidad de las Fuerzas Armadas – ESPE por ser el templo en donde aparte de recibir conocimiento, se reciben enseñanzas que perduraran en nuestra forma de ser y actuar

ANDRÉS M. ZAPATA HIDALGO

ÍNDICE

CONTENIDO

Índice de tablas

Índice de figuras

Resumen

La situación actual de emergencia sanitaria debido a la pandemia por el virus COVID 19, exige el desarrollo de soluciones tecnológicas e inteligentes que faciliten el control de las medidas de protección tales como el distanciamiento social, el uso de mascarilla, las aglomeraciones en espacios cerrados, entre otros. El presente proyecto tiene como finalidad el desarrollo de un prototipo basado en algoritmos de visión por computadora capaz de realizar la detección automática de mascarilla y la cuenta de personas, que ingresan en lugares tales como bares, restaurantes, gimnasios, cines, micro mercados, entre otros.

El prototipo se implementa en una tarjeta NVIDIA modelo Jetson Nano equipada con dos cámaras de video utilizadas conjuntamente con los algoritmos SSD MobileNetV1-COCO, SORT y la implementación de alertas audibles en la detección de mascarilla y la cuenta de personas respectivamente. Además, los datos del contador de personas se visualizan en el monitor de un servidor externo para la gestión del administrador del sistema.

PALABRAS CLAVE:

- **PANDEMIA COVID 19**
- **VISION POR COMPUTADORA**
- **ALGORITMOS DE DETECCIÓN DE OBJETOS**
- **DETECCION DE MASCARILLA**
- **CONTADOR DE PERSONAS**

Abstract

The current health emergency due to the COVID 19 virus pandemic requires the development of technological and intelligent solutions that facilitate the control of protection measures such as social distancing, the wearing masks, crowds in closed spaces, among others. The purpose of this project is to develop a prototype based on computer vision algorithms capable of automatically detecting masks and counting people who enter places such as bars, restaurants, gyms, cinemas, micro markets, among others. others.

The prototype is implemented on an NVIDIA Jetson Nano model card equipped with two video cameras jointly used with the SSD MobileNetV1-COCO, SORT algorithms and the implementation of audible alerts in mask detection and people counting, respectively. In addition, the people counter data is displayed on the monitor of an external server for management by the system administrator.

KEYWORDS:

- **COVID-19 PANDEMIC**
- **COMPUTER VISION**
- **OBJECT DETECTION ALGORITHMS**
- **MASK DETECTION**
- **PEOPLE COUNTER**

Capítulo I. Introducción

Antecedentes

La emergencia sanitaria causada por el Coronavirus supuso un cambio dramático en cuanto a la situación mundial, tanto en el ámbito económico, social, y sobre todo de salud; en donde se empezaron a tomar algunas medidas preventivas debido a la rápida expansión del virus. (Diaz & Toro, 2020). Sin embargo, con el paso de los meses, el Coronavirus logró expandirse rápidamente hacia todos los países del mundo, por lo que el 11 de marzo de 2020, la Organización Mundial de la Salud (OMS) declaró la ocurrencia de la pandemia de COVID-19, exhortando a todos los países a tomar medidas y aunar esfuerzos de control sobre la mayor emergencia en la salud pública mundial de los tiempos modernos. (Diaz & Toro, 2020)

Dentro del conjunto de medidas aconsejables para evitar el contagio de dicha enfermedad, se tiene entre las más importantes el uso de la mascarilla y el distanciamiento social, mismas que en la mayoría de los países han sido impuestas, e incluso su incumplimiento es motivo de penalizaciones. Sin embargo, el incumplimiento de ellas se ha convertido actualmente en el mayor problema que impide vencer a esta enfermedad. (Sedano, Rojas, & Vela, 2020)

El intento de las autoridades locales en Ecuador por crear conciencia social en la ciudadanía ha tenido leve impacto, pues el incumplimiento de las medidas de prevención es una situación que se vive a diario, y la cual ha traído como consecuencia que los hospitales trabajen al límite de su capacidad, e incluso haya personas esperando por atención o por una cama en las unidades de cuidados intensivos, hechos que generan preocupación a pesar de que el plan de vacunación del actual gobierno haya avanzado a ritmo acelerado. (Chauca , 2020)

Por esta razón la tecnología ha tomado un papel importante en el aporte de soluciones tecnológicas en la pandemia, pues el control de las medidas de prevención supone una tarea demasiado engorrosa pero

necesaria, ya que el uso de la mascarilla y el cumplimiento de los aforos permitidos dentro de espacios cerrados contribuye de sobremanera para frenar la ola de contagios. Por esta razón el desarrollo de dispositivos autónomos y que, mediante Visión Artificial son capaces de realizar acciones como: detección de mascarilla, medición del distanciamiento social, conteo de personas, se convierte en un gran aliado para poder vencer a la pandemia

Justificación

La situación actual debido a la emergencia sanitaria causada por el Coronavirus en nuestro país es una realidad compleja hasta día de hoy, pues a pesar de todos los esfuerzos que se realizan por evitar la propagación del virus, existen una inconsciencia social de gran magnitud en la ciudadanía que no permite superar de manera óptima esta situación. Es así como a diario se puede observar en lugares de acceso público, personas que no llevan mascarillas o que en su defecto la llevan en una posición incorrecta, fiestas clandestinas, lugares en donde no se respeta el aforo permitido, unidades de transporte al máximo de su capacidad; factores que sin duda alguna son los que no permiten que de a poco vayamos venciendo a esta pandemia.

Por esta razón el control del uso de la mascarilla y del distanciamiento social, que son los dos ejes principales para evitar el contagio del virus, deben ser atendidos con prioridad en nuestro país, y de manera especial en ambientes cerrados, en donde el contagio puede darse de forma masiva. Esto debido a que el control de todos los espacios y en todo momento resulta una tarea casi imposible para las autoridades, por lo que se requiere una solución automática. En vista de lo señalado, se puede notar que se requiere una solución tecnológica que pueda verificar de manera automática el uso de la mascarilla, y controlar el aforo de personas dentro de lugares concurridos, como pueden ser bares, restaurantes, gimnasios, cines, micro

mercados, entre otros; con el fin de respetar las medidas preventivas de uso correcto de la mascarilla y además de los aforos permitidos

Debido a esto, la tecnología actual resulta ser un gran aliado, pues el avance de la Inteligencia Artificial cada vez proporciona más herramientas que pueden ser aplicadas para la solución de problemas que afectan a la sociedad. Por ejemplo, los algoritmos de visión por computadora, mismos que permiten obtener, procesar y analizar secuencias de video en conjunto con modelos de detección de objetos resultan una integración de tecnologías idónea para cuando se requiere analizar en tiempo real la detección de algún objeto en particular.

La importancia de esta solución es de gran impacto dentro de nuestro país, pues a través de la implementación de un prototipo que sea autónomo, y capaz de realizar un control de acceso mediante la detección del uso de la mascarilla, y controlar el aforo mediante el conteo de personas que ingresan a un espacio cerrado, permitirá que las normativas que se dictan por los organismos reguladores sean acatadas, y en consecuencia la expansión del virus sea cada vez menor.

Alcance

En el presente proyecto se plantea implementar un prototipo con el cual se pueda realizar el control de acceso de personas en la entrada principal de locales o espacios cerrados, mediante la detección del uso de la mascarilla y también la cuenta del aforo de personas dentro de los mismos.

El prototipo será un dispositivo portable y autónomo, que estará compuesto como parte central de procesamiento de una tarjeta de desarrollo NVIDIA Jetson Nano, encargada de correr los modelos desarrollados para la detección de mascarilla y el conteo de personas; además de 2 cámaras de video como periféricos de entrada, que permitan la captura de video para dichos modelos, de un par de parlantes que permitan reproducir las señales audibles de alerta, tanto para el control de acceso mediante la detección

de mascarilla, como para el control de aforo mediante el conteo de personas. Además, contará con conectividad Wi-Fi mediante el uso de un adaptador de tarjeta de red, que permita enviar los datos asociados al contador de personas hacia el servidor externo, de manera que esta información sirva para la administración del local en donde se implemente el prototipo, como también de los usuarios. Teniendo como objetivo final que este prototipo pueda ser instalado y puesto a prueba en al menos 3 escenarios como bares, restaurantes, gimnasio, micro mercados, entre otros

El modelo de detección de mascarilla se realizará a través del reentrenamiento de la red SSD MobilenetV1 y un conjunto de datos personalizado, con su correspondiente inferencia a través de Tensor RT, con lo cual se pueda realizar la detección de: Rostro con mascarilla, Rostro sin mascarilla y Rostro con mascarilla mal ubicada. El modelo de conteo de personas se lo realizará mediante el uso de un modelo entrenado de detección de personas basado en SSD MobilenetV1-COCO y su correspondiente inferencia a través de TensorRT, integrado con el algoritmo de seguimiento SORT (Simple Online Real-time Tracking) con el cual se pueda realizar el conteo de personas que ingresan o salen de un espacio cerrado. Además, se tendrá el diseño de una función de alarmas que utilice señales audibles, para poder realizar el control de acceso en base a las detecciones del modelo de detección de mascarillas, y de manera similar para el conteo de personas, con lo cual se generen las alertas audibles y se pueda dictaminar cuando el aforo se encuentre al límite. Finalmente se implementará sobre un servidor externo los datos asociados al conteo de personas, con lo cual se pueda observar en tiempo real, el número de personas que se encuentran dentro de un local y también si el aforo se encuentra incompleto o en su defecto el aforo ya se ha completado.

Objetivos

General

Desarrollar un prototipo para la detección de mascarilla y control de aforo en lugares cerrados y de afluencia pública para mitigar el contagio del Covid-19, utilizando algoritmos de visión por computadora implementados sobre la tarjeta de desarrollo NVIDIA Jetson Nano.

Específicos

- Reentrenar un modelo de red SSD MobilenetV1 para la detección de 3 clases de objetos: rostro con mascarilla, rostro sin mascarilla y rostro con mascarilla mal ubicada e implementar el modelo entrenado (inferencia) en la tarjeta NVIDIA
- Implementar el algoritmo SORT (Simple Online Real-Time Tracking) en la tarjeta NVIDIA para realizar la cuenta de personas en el control de aforo en los locales.
- Integrar en el prototipo señales audibles de alerta del detector de mascarilla y control de aforo para las personas que ingresan a los locales
- Implementar un servidor externo que muestre en tiempo real al administrador y los usuarios, los datos asociados al conteo de personas.
- Realizar pruebas con el prototipo en al menos 3 lugares de acceso público de los mencionados anteriormente para verificar su funcionamiento en entornos reales.

Capítulo II. Marco teórico

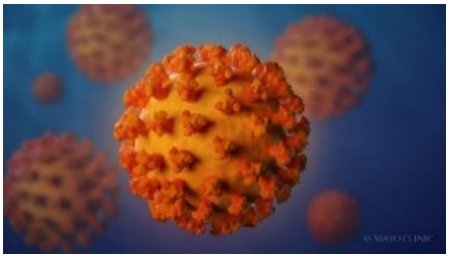
La pandemia de COVID-19

Una de las emergencias sanitarias más desastrosas a nivel mundial, empezó con el brote de un virus denominado síndrome respiratorio agudo grave (SARS-CoV-2), que en el mes de marzo de 2020, fue declarado por la Organización Mundial de la Salud (OMS) como pandemia, afectando de sobremanera a todos los países del mundo, desencadenando en la mayoría de ellos en el desbordamiento del sistema de salud y la incapacidad de brindar atención médica a todas las personas que contraían el virus.

En la denominada primera ola, pese a los esfuerzos de los entes gubernamentales de cada país fue un reto casi imposible el poder controlar la cadena de contagios y mucho menos el brindar una atención adecuada para hacer frente al virus, teniendo como resultado una ola de muertes desenfrenada, que sembró el temor dentro de toda la población mundial. Sin embargo, por mucho tiempo no se tomó a esta enfermedad con la seriedad del caso, pues muchas personas no se sujetaban a las medidas de prevención contra contagios de dicho virus, medidas como: el uso de mascarillas, el distanciamiento social, el uso de alcohol, el confinamiento, entre otras. Esto trajo como consecuencia un contagio masivo y simplemente agravar la situación de la pandemia cada día más.

Figura 1

SARS-CoV-2



Nota. Tomado de (Mayo Clinic, 2022)

Medidas de prevención

Desde el comienzo de la pandemia se empezaron a dictar lineamientos para poder ayudar a frenar el contagio del coronavirus dentro de las cuales destacan:

- Quedarse en casa si te sientes mal, aun si lo síntomas son leves
- Lavado frecuente de manos con agua y jabón
- Uso de alcohol en caso de contacto con superficies
- Uso de mascarilla
- Distanciamiento social de al menos 1 metro
- Evitar el contacto de ojos y boca con las manos
- Evitar aglomeraciones y permanecer en espacios poco ventilados
- Limpieza y desinfección de las superficies de alto contacto

Figura 2

Medidas de prevención

Nota. Tomado de (Organización Panamericana de la Salud, 2020)

Tecnología y Pandemia COVID19

La tecnología empezó a tomar un papel importante durante la pandemia, y los desarrolladores de software no perdieron tiempo en la innovación de nuevas plataformas o dispositivos que de alguna manera ayuden en la pandemia. Un claro ejemplo son las plataformas que se desarrollaron para realizar telemedicina, con lo cual se evitó de sobremanera las aglomeraciones dentro de las instituciones de salud. Otro ejemplo son los dispositivos medidores de temperatura y dispensadores de alcohol automático que hoy en día nos podemos encontrar en centros comerciales, bares, restaurantes, cines y gran cantidad de espacios cerrados.

Pero no solo esto ha surgido, pues debido a la pandemia la investigación de la inteligencia artificial también se centró en brindar soluciones que aporten en la situación de pandemia, es por ello que los detectores de mascarillas fueron quizá el principal objetivo, pues el uso de mascarilla es el principal agente que puede librarnos del contagio del Coronavirus, y a pesar de que se ha desarrollado muchos avances sobre el tema, aún resulta una tarea compleja el implementarlo sobre un dispositivo que sea capaz de realizar dicha tarea. De igual manera la IA también se ha visto enfocada en el aspecto del distanciamiento social, dispositivos capaces de medir el distanciamiento entre personas han surgido como soluciones, sin embargo, es claro que, si bien es cierto que la investigación sobre el tema ha sido abundante, aún existen muchas limitaciones que frenan el desarrollo, pero que a la vez han permitido el surgir de nuevas soluciones a través de la tecnología IA.

Inteligencia Artificial

El concepto de inteligencia artificial fue acuñado en 1956 por John McCarthy quien la definió como la ciencia y el ingenio de hacer maquinas inteligentes, especialmente programas de cómputo. La inteligencia artificial vista de otra perspectiva hacer referencia a una simulación de los procesos propios de la

inteligencia humana por parte de máquinas, dichos sistemas incluyen el aprendizaje, que se lo realiza mediante la obtención de información y reglas para el uso de esta en la aplicación deseada, el razonamiento, que fundamentalmente hace uso de dichas reglas con el fin de llegar a obtener un resultado aproximado y finalmente la autocorrección. De manera equivalente podemos llamar a Inteligencia Artificial a todo aquel software que tenga la capacidad de imitar capacidades humanas, dirigido hacia la realización de tareas específicas o tareas un tanto más generalizadas. Sin embargo, el reto más importante es que dichos softwares cada vez sean capaces de poder reaccionar en entornos a un nivel más profundo.

De manera general se puede afirmar que la Inteligencia Artificial moderna se encuentra dividida en dos escuelas: la primera que se enfoca en el estudio del comportamiento humano denominada convencional y la que se enfoca en un aprendizaje interactivo llamada computacional. Dentro de la sociedad moderna día a día podemos visualizar diferentes aplicaciones de la inteligencia artificial como, por ejemplo, los robots que realizan tareas específicas, coches autónomos, dispositivos de detección, entre otros. Pero esto es solo un pequeño comienzo del gran abanico de aplicaciones que en el futuro se podrá desprender de la inteligencia artificial, pues esta tecnología verdaderamente será un gran salto a una nueva era, por lo que la investigación sobre este campo ha tomado gran importancia actualmente.

Redes Neuronales Artificiales

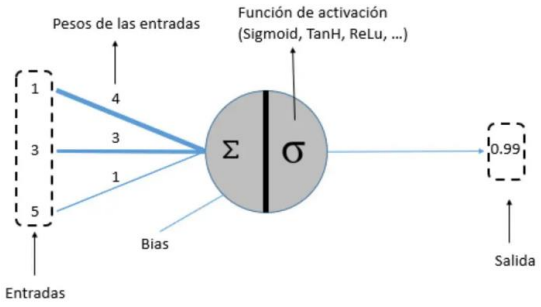
Cuando se habla de redes neuronales artificiales, inmediatamente se produce la relación con las redes neuronales humanas, y es que no está por demás realizar dicha comparativa ya que el objetivo principal de las redes neuronales artificiales es emular el funcionamiento de las redes de neuronas biológicas propias del ser humano. Por esta razón , partiendo de un concepto biológico, se dice que las neuronas que se encuentran alojadas en nuestro cerebro se encuentran compuestas de dendritas, el soma y el axón: todas las mencionadas anteriormente tiene una función específica, las dendritas son las

encargadas de captar los impulsos nerviosos que emiten otras neuronas, estos son procesados en el soma y posterior son enviados a través del axón, el cual emite un impulso nervioso hacia las neuronas contiguas.

De manera general y esquemática una neurona artificial puede ser representada como se observa en la siguiente figura:

Figura 3

Esquema de una Neurona Artificial



Nota. Tomado de (García , 2019)

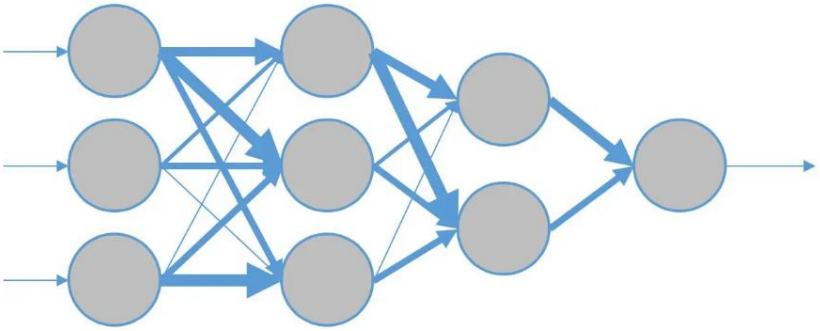
De manera similar a la concepción de cómo está compuesto el cerebro, internamente por millones de neuronas interconectadas entre sí, las redes neuronales artificiales se encuentran formadas por un número determinado de neuronas artificiales que se encuentran relacionadas entre sí y que se agrupan por niveles, lo que se lo denomina capas.

Dicho esto, se pude definir a una capa de una red neuronal artificial como la asociación de neuronas que tiene entradas provenientes de una capa anterior, y de la cual se obtienen salidas que a la vez se transforman en las entradas de una capa posterior.

Una red neuronal compuesta de 4 capas se muestra a continuación:

Figura 4

Estructura de una Red Neuronal de 4 capas



Nota. Tomado de (García , 2019)

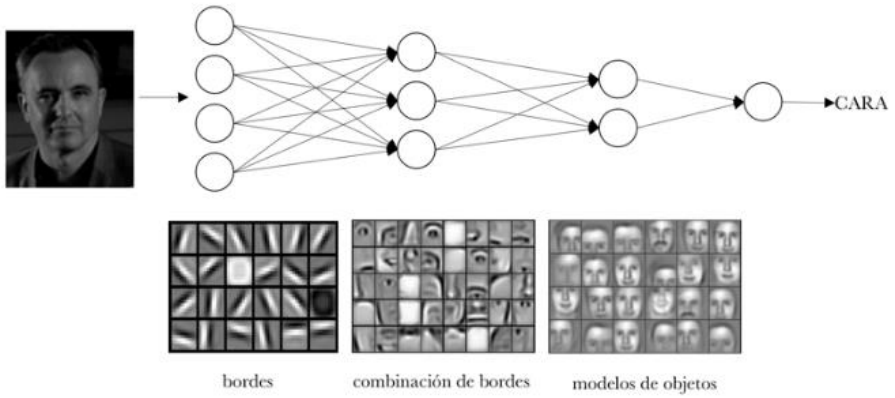
Por lo tanto, una red neuronal se la puede definir con los siguientes componentes: capa de entrada, capas ocultas y capa de salida. Cada capa puede estar formada por una o un número determinado de neuronas, el número de capas ocultas puede ser un numero extremadamente alto, de donde surge la idea de Deep Learning o Aprendizaje profundo, en el cual se trabaja con una gran cantidad de capas ocultas, que permiten a la vez la solución de tareas más complejas y también de la mejora del rendimiento de las redes neuronales artificiales.

Redes Neuronales Convolucionales

Cuando se habla de redes neuronales convolucionales, la principal característica de este tipo de redes es que están especializadas en la apreciación de las propiedades visuales de los objetos, es decir formas, líneas, bordes, texturas, entre otros. La visión humana se da de una forma parecida, pues estas características son las que permiten que se dé una identificación visual, y que de forma similar lo realizan las redes neuronales mediante una identificación artificial. Por esta razón mediante la discriminación de dichas características que tienen preponderancia sobre otras que no lo son, se logra generar el efecto de este tipo de redes neuronales.

Figura 5

Funcionamiento de una red Neuronal para detección de rostro



Nota. Tomado de (ViewNext, 2019)

Matemáticamente hablando se puede definir al término convolución como una función aplicada sobre otra, y está fundamentada como una operación básica en el proceso de extraer información de parte de este tipo de redes neuronales, procedimiento que intenta imitar a la forma en que trabaja el córtex cerebral, mismo que es quien se encarga de dotar de visión al ser humano. De manera que dicho proceso artificial, el cual imita la visión humana es capaz de poder diferenciar las particularidades internas dentro de un grupo de datos que pertenecen a una categoría de objetos. Con ello, se puede asegurar una asignación de etiquetas que posea un alto índice de probabilidad de ser la correcta.

Sin embargo, para lograr dicho objetivo, es necesario saber que se requiere de un gran volumen de datos, que generalmente son imágenes, o se los analiza de esa manera, debido a que el análisis de estos elementos gráficos se los lleva a cabo mediante la diferenciación de cada uno de sus partes en base al color. La máquina es quien realiza la división mediante la separación del espacio superficial en retículas pixeladas, mismas que a su vez son quienes entregan información numérica.

Precisamente en este punto es donde toman importancia las convoluciones, pues estas consisten en ingresar un grupo de pixeles cercanos, pero que en términos de tamaño son inferiores a los de la imagen original. De esta manera es como se irán identificando patrones de comparación con la imagen objeto que representa la clasificación automática. Cuando sobre la imagen se realizan las superposiciones se produce una nueva matriz de datos, estos grupos de pixeles se los llama Kernel y su unión representan un filtro.

Una vez que sean desarrolladas, se añade una función de activación, con lo cual se consigue un mapeado de dimensiones idénticas a la original. Cabe mencionar que mientras se trabaje con figuras de más complejidad, pues el número de convoluciones será proporcionalmente con dicha característica.

Entrenamiento de una red neuronal

Cuando se habla del entrenamiento de una red neuronal se hace referencia a el ajuste de los pesos asociados a cada neurona como entrada de lo cual se habló anteriormente, todo esto con el objetivo de que las respuestas de la capa de salida sean las esperadas y que se ajusten en lo posible de mejor manera a los datos que son conocidos

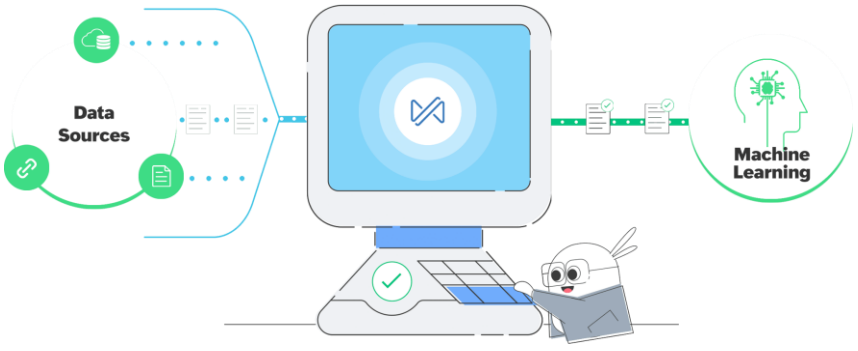
Por ejemplo, si se desea realizar el entrenamiento de una red neuronal artificial para la detección de cualquier objeto, en cada nivel de las capas ocultas los pesos de cada neurona se van ajustando, de manera que el error vaya disminuyendo y tendiendo hacia cero, en otras palabras menos técnicas, para que una red neuronal artificial pueda generalizar e identificar un determinado objeto sobre la entrada será necesario etiquetar imágenes que pertenezcan a la clase del objeto que se desea identificar, misma que se le daría una valor verdadero y también de imágenes que no pertenezcan a la clase del objeto identificadas como falsos, con lo cual, la red sea capaz de ajustar los parámetros internos de la red, extrayendo las particularidades principales para poder identificar si en efecto se trata del objeto que se desea identificar o no.

Machine Learning

El termino Machine Learning comprende una de las diferentes ramas de la IA, y se encuentra orientado a que las maquinas a través de su aplicación sean capaces de aprender una tarea especifica. Todo esto se realiza con el finde obtener sistemas computacionales que puedan identificar patrones o extraer características de un conjunto de datos, para posteriormente, y haciendo uso de dicha información realizar una inferencia y obtener una predicción acertada de lo que se busca. Actualmente este tipo de algoritmos son usados comúnmente en nuestra vida diaria, y unos claros ejemplos se dan cuando obtenemos recomendaciones en redes sociales como por ejemplo Facebook o Instagram de las necesidades que actualmente requerimos, en base a lo que vemos dentro de dichas redes, de igual manera cuando se recomienda la visualización de un video, serie o película en Plataformas como YouTube, Netflix, Spotify y también los sistemas inteligentes de interacción con humanos como son Siri o Alexa.

Figura 6

Machine Learning



Nota. Tomado de (Amat, 2021)

Por lo que de manera general y precisa podemos definir el Machine Learning como el encargado del reconocimiento de patrones dentro de los sistemas informáticos, con lo cual logra la capacidad de, a partir

de un determinado conjunto o muestra de datos en un software informático, extraer nuevas inferencias de otros conjuntos de datos, con los cuales no se ha realizado un entrenamiento previo.

Dentro de los tipos de aprendizaje que se puede tener en Machine Learning destacan los siguientes:

Aprendizaje por refuerzo

Este tipo de aprendizaje se encuentra relacionado con las maquinas que realizan su aprendizaje mediante métodos de prueba y error, de manera que con este proceso se pueda alcanzar un performance deseado para poder realizar una tarea especifica.

El más claro y conocido ejemplo es cuando DeepMind desarrollo un tipo de Inteligencia Artificial denominada AlphaGo, misma que tras un entrenamiento que duro algunos meses mediante el análisis de partidas desarrolladas por humanos tuvo como resultado, el hecho de vencer al campeón vigente de Go, lo realmente interesante es que debido a la complejidad del juego las maquinas convencionales nunca habían podido obtener un resultado positivo al encontrarse con la dificultad de incorporar estrategia al código, entonces surgió la duda de cuál fue el secreto para obtener dicho resultado. Precisamente la mejora no tuvo nada que ver con la capacidad de procesamiento o capacidad computacional de la máquina, sino el tipo de aprendizaje que se le otorgó, pues prácticamente la maquina aprendió a jugar sola mediante este tipo de aprendizaje, jugando un sinfín de partidas en su contra hasta que finalmente aprendió las mejores maneras posibles de obtener el mejor resultado.

Aprendizaje Supervisado

Este tipo de aprendizaje tiene como objetivo el entrenamiento de las maquinas haciendo uso de datos completamente etiquetados, es decir que cada dato posea una descripción concisa, con lo cual el algoritmo sea capaz de seleccionar dichas etiquetas en otro conjunto de datos. Por ejemplo, se puede citar

un sistema de detección de personas, entonces el conjunto de datos deberá estar correctamente etiquetado con esta clase, con lo cual la maquina podrá ser capaz de identificar imágenes similares en donde se encuentren personas.

Aprendizaje no supervisado

El Aprendizaje no supervisado prácticamente se encarga de inferir patrones de un determinado conjunto de datos, pero esta tarea se la realiza sin una referencia previa a o en base a resultados ya conocidos o etiquetados, de manera que, a diferencia del Aprendizaje Supervisado, esto algoritmos no se puede aplicar a una tarea de regresión o clasificación, y esto es debido a que en dicho algoritmo los valores de salida son desconocidos.

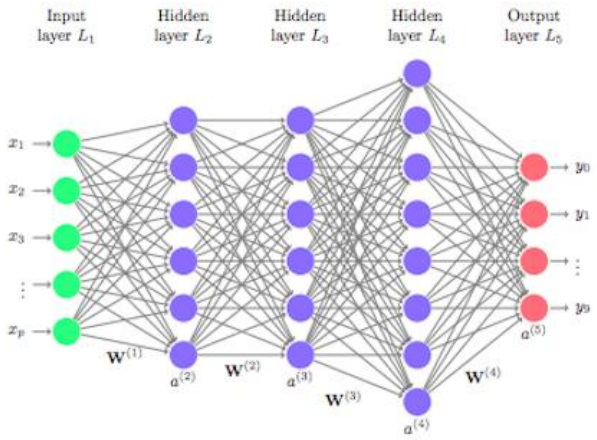
Deep Learning

El denominado Deep Learning o aprendizaje profundo, prácticamente se desprende como una rama del Machine Learning, en el caso específico en donde la cantidad de capas ocultas se vuelve un numero bastante mayor, y también en la cual la cantidad de datos que se procesan es bastante alta, de manera que este tipo de algoritmos requieren en términos computacionales una gran capacidad tanto para poder trabajar con una gran cantidad de datos como para poder procesar dichos datos a través de las numerosas capas ocultas.

El Deep Learning tiene como objetivo conseguir que un ordenador o un procesador realice el aprendizaje por cuenta propia y que realice determinadas tareas que sean similares a las de los humanos, como, por ejemplo, la identificación de imágenes, reconocimiento de habla o incluso la capacidad de realizar predicciones en sistemas en donde se logre identificar patrones y a la vez aplicarlos a dichas predicciones.

Figura 7

Representación de una red neuronal profunda



Nota. Tomado de (Amat, 2021)

Por ejemplo, un caso ideal en donde se aplica Deep Learning es en el reconocimiento de letras o símbolos, es decir si una persona escribe una letra cualquiera, habrá miles de letras diferentes, cada una con un estilo o con una forma propia de quien la escribe, pero al fin y al cabo todas representan lo mismo. Las aplicaciones de Deep Learning tienen la capacidad de identificar los trazos, ya sea de cualquier estilo de escritor o de cualquier forma de escritura, todo esto con un porcentaje de acierto alto, ya que precisamente es ahí en donde la alta cantidad de datos y procesamiento son los que permiten que se de este performance.

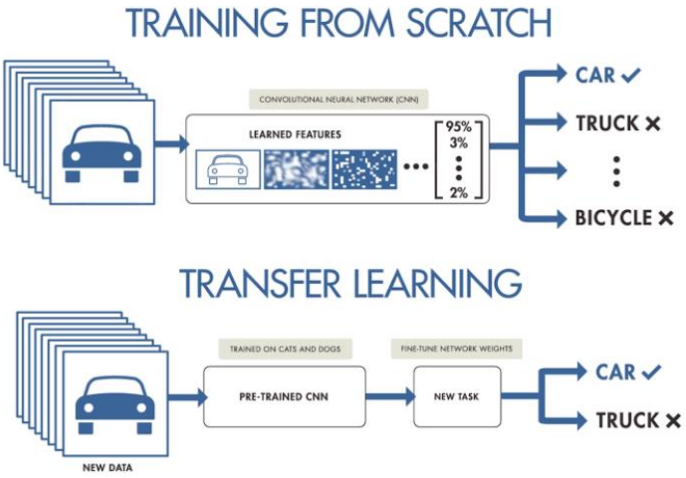
Transfer Learning

La transferencia de aprendizaje se muestra como una técnica relativamente nueva surgida desde el Deep Learning y que específicamente es aplicado dentro del campo de aprendizaje automático con Inteligencia Artificial. Y en términos breves consiste en el aprovechamiento de una gran cantidad de información que ha sido obtenida en la realización de una tarea o problema, para de esta manera, utilizarla en otra tarea especifica, pero que, de alguna manera, comparte características o patrones de similitud.

Hoy en día existen gran variedad de modelos de redes neuronales que ya se encuentran realizando una tarea, ya sea de detección, clasificación, reconocimiento, entre otros. Dichas redes han pasado por un proceso largo de optimización y de verificación de mejora en cuanto al rendimiento, por lo que en términos de investigación, si se desea realizar por ejemplo un detector de objetos, resulta un camino muy engorroso el empezar por diseñar una nueva red neuronal que en términos de rendimiento se asemeje a las redes que actualmente dominan este circuito, es por ello que si se realiza una detección por ejemplo de perros, dicho modelo perfectamente puede ser usado para detectar gatos, caballos o todos los animales que comparten características esenciales y que a la vez el modelo sea capaz de identificar

Figura 8

Diferencia entre un entrenamiento desde cero y con Transfer Learning



Nota. Tomado de (ViewNext, 2019)

Nvidia Jetson Nano

La inteligencia Artificial ha tomado mucha importancia en la actualidad, sin embargo, no en cualquier dispositivo se puede desarrollar este tipo de tecnología, por esta razón NVIDIA ha creado un hardware que se encuentra disponible para creadores, estudiantes y desarrolladores que estén incursionando en el mundo de la IA.

Nvidia Jetson Nano Developer Kit es un ordenador de capacidades limitadas, pero a la vez de gran alcance, mismo que está desarrollado con el fin de poder realizar la ejecución de múltiples redes neuronales trabajando en paralelo ya sea para tareas de detección de objetos, clasificación de imágenes, procesamiento de imágenes o del habla, entre otros. Todas estas funciones se pueden realizar sobre una plataforma portable pero potente.

En muchas ocasiones las aplicaciones de inteligencia artificial se las desarrolla sobre ordenadores de gran capacidad computacional, con tarjetas gráficas potentes, memoria RAM por encima de 8 GB y procesadores de última generación, sin embargo, las aplicaciones de inteligencia artificial orientadas al servicio de la comunidad requieren de un dispositivo pequeño pero eficiente en el cual se pueda correr dichos algoritmos. Por esta razón surge la línea de dispositivos Jetson. A la espera de que los desarrolladores de software comiencen a desarrollar sus aplicaciones y que, a la vez, estas puedan funcionar con un dispositivo que sea implementado en el lugar que la aplicación requiera.

La Jetson nano es una microcomputadora lanzada pro NVIDIA sobre todo enfocado en proyectos de robótica o dispositivos autónomos, misma que contiene un GPU, un procesador, una memoria RAM y los dispositivos de entradas y salidas para poder interactuar con dicha microcomputadora.

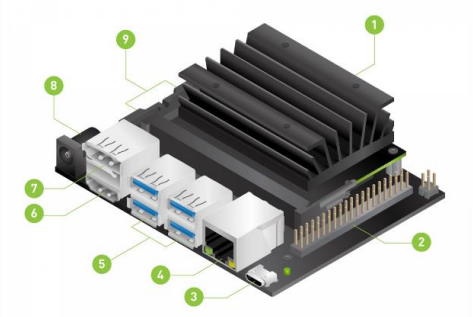
Cuenta con un procesador de 4 núcleos, cada uno corriendo a 1.4 GHz, su GPU contiene 128 núcleos de la arquitectura Maxwell de NVIDIA, estos GPUs son los que nos van a ayudar a poder realizar inferencias

y poder correr modelos de Inteligencia Artificial, puesto que estos ayudan al procesamiento en paralelo de operaciones que son cruciales para inferencia y entrenamiento de modelos IA.

El sistema operativo no viene instalado por default, pero Nvidia libero una versión de Ubuntu, la cual es denominada JetPack, la cual es una modificación de Ubuntu para que pueda funcionar específicamente sobre la Jetson Nano, y que viene internamente precargado con algunas librerías orientadas hacia aplicaciones de IA.

Figura 9

Tarjeta Nvidia Jetson Nano



Nota. Tomado de (Nvidia, 2022)

La Jetson nano contiene varios puertos de entrada y de salida, los cuales son:

1. Entrada para tarjeta microSD (Sistema Operativo)
2. 40 pines de entrada y salida (Sensores y actuadores compatibles)
3. Entrada micro USB para energización
4. Entrada Ethernet
5. 4 puertos USB 3.0
6. Puerto HDMI

7. Display port
8. Entrada de barril para energización (MAX POWER)
9. Conectores de cámara MIPI CSI-2

Tensor RT

Con TensorRT, puede optimizar los modelos de redes neuronales entrenados en todos los marcos principales, calibrar para obtener una precisión más baja con una precisión alta e implementar en centros de datos de hiper escala, plataformas de productos automotrices o integradas. Las aplicaciones basadas en TensorRT funcionan hasta 40 veces más rápido que las plataformas solo de CPU durante la inferencia.

TensorRT se basa en CUDA, el modelo de programación paralela de NVIDIA, y le permite optimizar la inferencia aprovechando bibliotecas, herramientas de desarrollo y tecnologías en CUDA-X™ para inteligencia artificial, máquinas autónomas, computación de alto rendimiento y gráficos. Con las nuevas GPU NVIDIA Ampere Architecture, TensorRT también aprovecha los núcleos de tensor dispersos que brindan un impulso de rendimiento adicional. (Nvidia, 2022)

TensorRT proporciona INT8 mediante Quantization Aware Training y Post Training Quantization, y optimizaciones FP16 para implementaciones de producción de aplicaciones de inferencia de aprendizaje profundo, como transmisión de video, reconocimiento de voz, recomendación, detección de fraude, generación de texto y procesamiento de lenguaje natural. La inferencia de precisión reducida reduce significativamente la latencia de la aplicación, que es un requisito para muchos servicios en tiempo real, así como para aplicaciones integradas y autónomas. (Nvidia, 2022)

Con TensorRT, los desarrolladores pueden concentrarse en crear nuevas aplicaciones impulsadas por IA en lugar de ajustar el rendimiento para la implementación de inferencias.

Algoritmo SORT (Simple Online Real-time Tracking)

SORT es una implementación básica de un marco de seguimiento visual de múltiples objetos basado en técnicas rudimentarias de asociación de datos y estimación de estado. Está diseñado para aplicaciones de seguimiento en línea donde solo están disponibles los fotogramas pasados y actuales y el método produce identidades de objetos sobre la marcha. Si bien este rastreador minimalista no maneja la oclusión o el reingreso de objetos, su propósito es servir como referencia y banco de pruebas para el desarrollo de futuros rastreadores. (GitHub, 2019)

Detección

El algoritmo original determina qué para aprovechar el rápido desarrollo de la detección basada en CNN, se usa el marco de detección FrRCNN el cual es un marco de extremo a extremo, que consta de dos pasos:

Paso 1: Extrae características y propone región

Paso 2: Clasificación de objetos dentro del área sugerida anteriormente

Sin embargo, menciona que se puede reemplazar con cualquier otra arquitectura de diseño.

Propagación de estados de objetos en fotogramas futuros (filtro de Kalman)

El desplazamiento de los objetos en los fotogramas consecutivos se aproxima mediante un modelo lineal de velocidad constante que es independiente de otros objetos y del movimiento de la cámara. El estado de cada objetivo se denota como:

$$x = [u, v, s, r, \dot{u}, \dot{v}, \dot{s}]$$

Donde u & v representan el centro del cuadro delimitador r & u indican la escala y la relación de aspecto. Con lo cual se obtienen 2 escenarios:

Escenario 1 (cuando se detecta un objeto): Utilizando el marco de filtro de Kalman y el cuadro delimitador detectado, se predice el estado futuro. Aquí las velocidades se calculan de manera óptima.

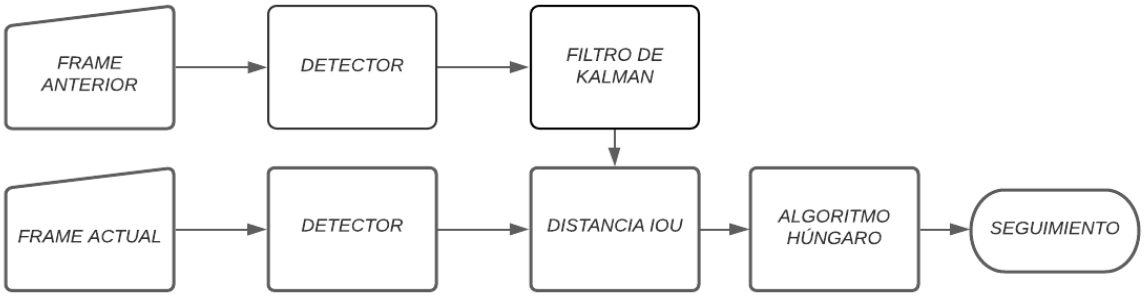
Escenario 2 (Sin detección asociada con el objetivo): El estado simplemente pasa sin corrección (es decir) utilizando un modelo de velocidad lineal simple.

Asociación de detecciones actuales con objetos existentes (algoritmo húngaro)

Utilizando los nuevos estados objetivo, predecimos los cuadros delimitadores que luego se comparan con los cuadros detectados en el marco de tiempo actual. La métrica IOU y el algoritmo húngaro se utilizan para elegir la casilla óptima para transmitir la identidad. También se establece un límite de corte para el IOU y se eliminarán todas las casillas que resulten en una puntuación del IOU por debajo de este límite. (Bewley & Ge, 2017)

Figura 10

Asociación de detecciones con objetos existentes



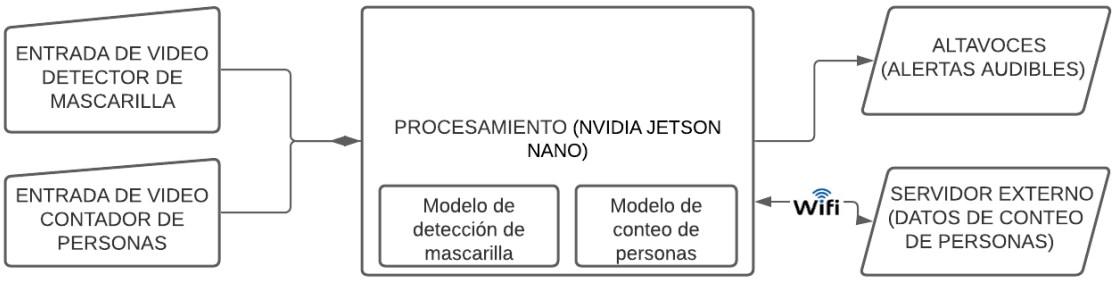
Capítulo III. Diseño

Concepción general del diseño del prototipo

Para el desarrollo del prototipo, se tiene una concepción de un dispositivo capaz de realizar el control de acceso mediante la detección de mascarillas y también del control de aforo de personas dentro de espacios cerrados, mediante un contador de personas. Con lo cual se requiere un sistema que en primera instancia sea autónomo y pueda ser instalado en cualquier espacio cerrado en el que lleve a cabo su función. Para ello se plantea el siguiente esquema:

Figura 11

Diseño conceptual del prototipo



Como se observa en el diagrama, el prototipo debe contar con dos periféricos de entrada que proporcionen la señal de video a los modelos de detección de mascarilla y conteo de personas, debido a que estas tareas se deben realizar de manera independiente. Como unidad de procesamiento se tiene la tarjeta de desarrollo NVIDIA Jetson Nano, misma que debe correr los modelos de detección, además de unos parlantes como periférico de salida, encargados de reproducir las señales audibles de las alertas de los modelos de detención de mascarilla y conteo de personas.

En vista de la necesidad de mostrar los valores de detección asociado al contador de personas, también se requiere conectividad Wi-Fi para poder enviar dichos datos hacia un servidor externo en el cual, tanto para la administración como para visualización de los usuarios, se indique las personas que se encuentran dentro del establecimiento y también si el aforo se encuentra incompleto o en su defecto si el aforo se encuentra lleno.

Consideraciones para el posicionamiento de las cámaras

La estructura del prototipo está dada por criterios técnicos que se manejan dentro de los detectores tanto de personas como de mascarillas. Si bien es cierto que ambas tareas eventualmente podrían ser realizadas por una sola cámara es necesario saber que el enfoque que se tiene para un modelo de detección depende en gran medida de la aplicación que se le va a dar.

Posicionamiento de cámara para detección de mascarilla

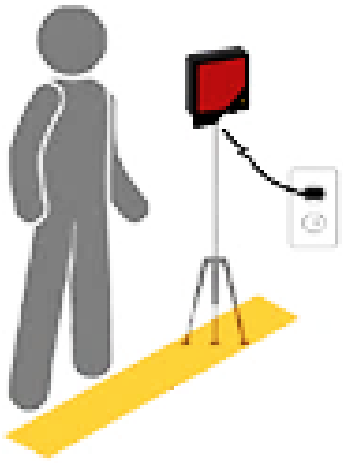
La detección de mascarilla es una tarea que podría parecer sencilla, sin embargo, durante el largo camino de investigación del cual ha sido objeto, se han venido creando redes neuronales más precisas y confiables en la detección de objetos, pero esto no solo depende del rendimiento de la red neuronal, sino también del conjunto de datos, el entrenamiento, el motor de inferencia; todo esto en el aspecto técnico.

Cuando un modelo de detección es puesto a prueba también existen otros valores de evaluación del rendimiento que intervienen, en este caso hablamos de factores externos como la iluminación, el posicionamiento, la distancia a la que trabajan las cámaras que capturan el video, entre otros.

En el caso del control de acceso mediante la detección de mascarilla, se propone un posicionamiento de la cámara que reciba a la persona a ser detectada, de manera frontal, con lo cual, se asegure una detección más precisa.

Figura 12

Modo frontal de detección de mascarilla



Esto debido a que en ocasiones los detectores presentan bajo rendimiento cuando, por ejemplo, el objeto de detección se encuentra demasiado lejano, o por factores como iluminación o perspectiva.

Posicionamiento de la cámara para el conteo de personas

El conteo de personas es una tarea que puede ser realizada con varios dispositivos electrónicos, uno de ellos mediante el uso de cámaras de video, manera en la cual se plantea en este prototipo. El uso de cámaras de video para el conteo de personas tradicionalmente trae consigo un enfoque de detección desde una vista superior, que se puede evidenciar en cámaras especializadas para esta tarea, mismas que se ubican en lugares altos de manera que la detección se realice desde una vista superior.

Pero este enfoque trae consigo consideraciones técnicas que lo sustentan, una de ellas es el problema de sobre posicionamiento que generalmente sucede cuando se realiza detección de personas. Cuando se tiene una vista frontal o lateral, este problema tiene más incidencia debido a la perspectiva con la que se detecta el objetivo, a manera de ejemplo se muestra en la siguiente figura.

Figura 13

Detección Frontal con evidencia de sobre posicionamiento

Pero si se toma una vista superior estos problemas son menos perjudiciales para las detecciones y en consecuencia para el conteo de personas, pues en una vista superior el sobre posicionamiento tiene una baja incidencia debido a la perspectiva de detección. Por esta razón para el desarrollo del prototipo se considera una vista superior de la cámara que se encarga de la detección de personas.

Figura 14

Detección desde una vista superior

Consideraciones para la carcasa del prototipo

En vista a las consideraciones anteriores, se plantea que el prototipo requiere una carcasa en la cual pueda montarse todos los elementos requeridos por el sistema de detección. Debido a esto se tiene en cuanta las siguientes consideraciones para su diseño:

Estructura

La carcasa debe tener una estructura robusta y de fácil manejo debido a que su instalación no debe suponer un gran reto. En base a esto se propone una estructura rectangular para que pueda ser colocado sobre una base fija sobre una estructura fuerte que puede ser una columna o una pared.

Material

El material de la carcasa que se debe usar debe ser resistente y llamativo, debido a que el prototipo estará en contacto visual con los usuarios, por lo cual se considera que la mejor opción es trabajar con impresión en 3D y usar el material PLA que es un material resistente y a la vez que resulta innovador para este tipo de aplicaciones.

Ventilación

Debido a que la tarjeta NVIDIA Jetson Nano deber correr los modelos de detección de mascarilla y el conteo de personas en tiempo real, esta tarea supone una gran cantidad de procesamiento que a la vez provoca energía en forma de calor, por lo que es necesario que la carcasa tenga orificios de ventilación por donde se pueda disipar dicha energía que emana la tarjeta.

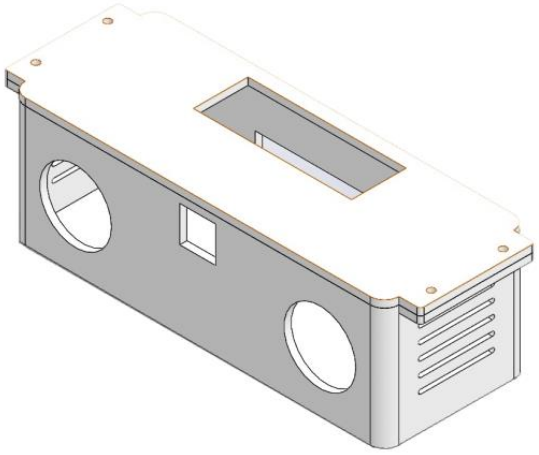
Consideraciones adicionales

La carcasa además requiere de orificios para la salida de audio (parlantes), para que el usuario pueda escuchar de forma clara las alertas audibles. Además de un par de orificios por los cuales se puedan realizar las conexiones tanto de alimentación del dispositivo, como de los periféricos de entrada y salida.

En vista a las consideraciones anteriores, se diseña la carcasa del prototipo mediante el software Inventor, con el objetivo de obtener el archivo correspondiente para poder realizar la impresión en 3D de la misma. A continuación, se indica la carcasa que contendrá todos los elementos del prototipo:

Figura 15

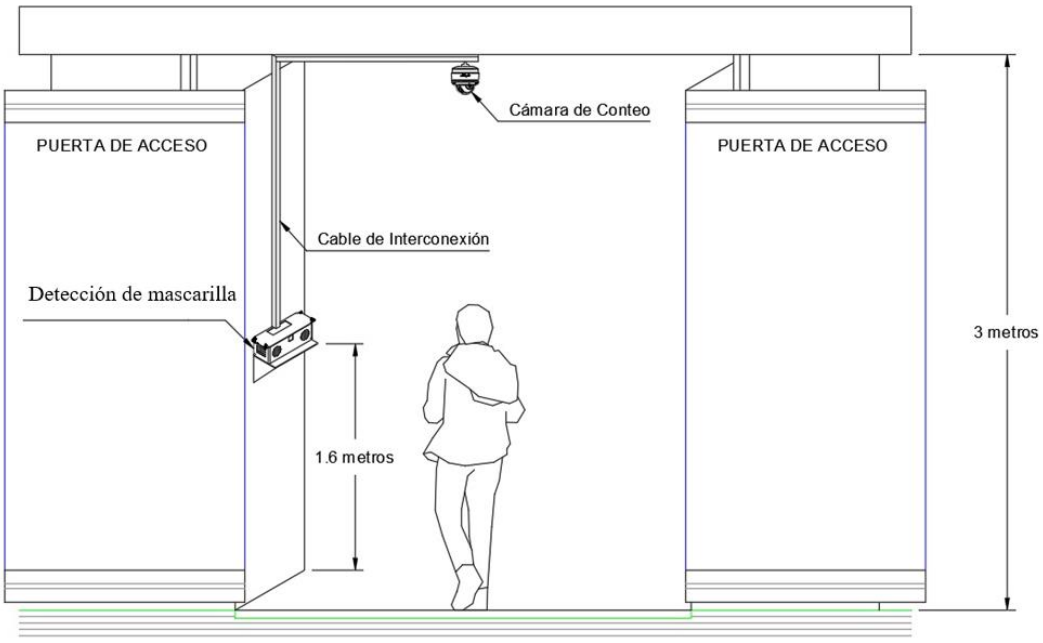
Diseño de la carcasa del prototipo



Dentro de la carcasa se tiene el espacio correspondiente para la tarjeta Jetson Nano, la cámara para la detección de mascarillas, los altavoces, el módulo wifi y un extensor de puertos USB en el caso de requerirse más salidas de este tipo. Mientras que la cámara encargada del conteo de personas se encuentra fuera de la carcasa y se la ubica en una posición alta con una vista superior y centrada en el acceso del lugar en donde vaya a ser instalada. Esto se muestra a continuación:

Figura 16

Forma de instalación del prototipo



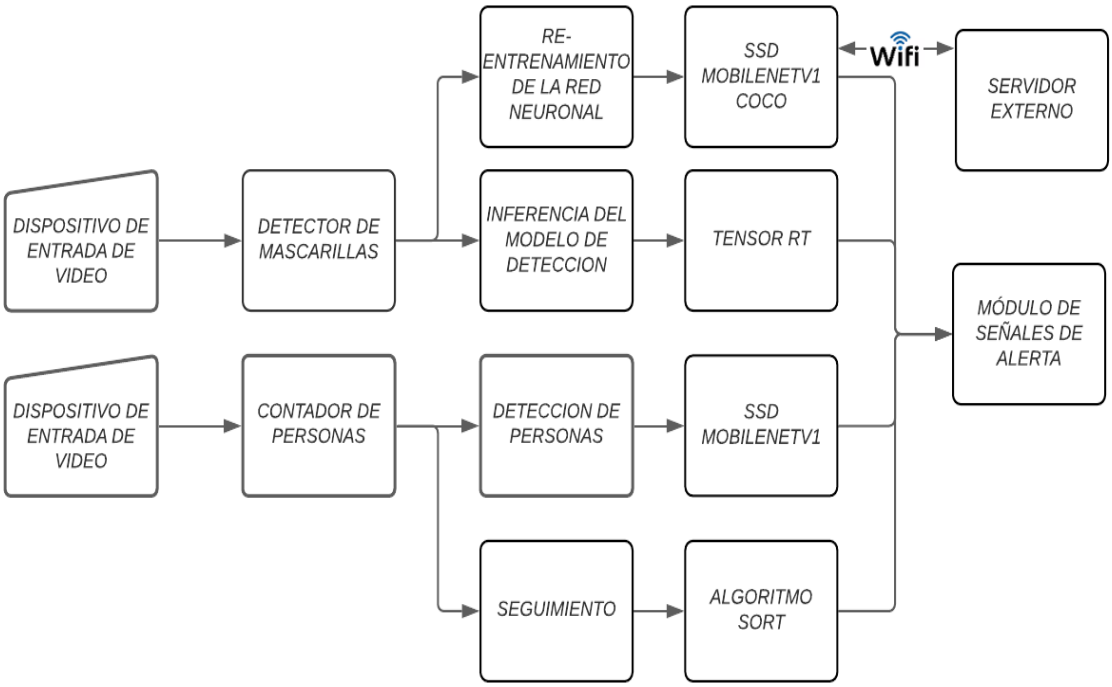
Como se observa en la figura el prototipo se debe ubicar a una altura propicia para que la detección de mascarillas sea aplicable a personas de toda estatura y en el caso de la cámara para el conteo de personas ubicarse a una altura en la cual se pueda tener un enfoque correcto de la entrada principal por donde ingresen los usuarios al establecimiento. Los valores que se muestran en la figura son tentativos, pues más adelante mediante las pruebas de funcionamiento se determinará las posiciones y alturas óptimas para las cámaras.

Descripción general del sistema

El sistema se encuentra enfocado en la tarea de realizar un control de acceso mediante la detección de mascarilla y a la vez el control de aforo mediante el conteo de personas en el ingreso a lugares cerrados como bares, restaurantes, gimnasios, micro mercados, entre otros. Por esta razón se presenta a continuación un esquema técnico general de cómo está estructurado el sistema.

Figura 17

Estructura general del sistema



Como se observa en el diagrama de bloques anterior, existen dos componentes principales dentro del prototipo que son el detector de mascarilla y el contador de personas.

El detector de mascarillas se encuentra formado por dos elementos principales que son: el reentrenamiento de la red neuronal SSD MobileNetV1 y también por la inferencia del modelo generado para la detección de mascarilla, con lo cual se realiza un control de acceso mediante la generación de alertas auditivas en caso de llevar la mascarilla puesta correctamente, no llevar la mascarilla puesta o llevarla en una posición incorrecta.

La generación de dichas alarmas permite que a la entrada de los escenarios de prueba se asegure que todas las personas que ingresan cumplan con la medida de protección de llevar la mascarilla en su posición correcta.

Por otro lado, el contador de personas también se encuentra compuesto por dos partes que son la detección de personas, el cual se lo realiza con un modelo ya entrenado para la detección de personas, con la red SSD MobileNetV1-COCO y la inferencia mediante TensorRT. Mientras que la parte de seguimiento se lo realiza mediante el algoritmo SORT(Simple Online Real time Tracking), con lo cual se logra realizar el conteo de las personas que ingresan y salen de un espacio cerrado, obteniendo el valor de personas que se encuentran dentro de un escenario de prueba.

Con esto se realiza el control de aforo dentro de los dichos escenarios, y de igual manera mediante una alerta auditiva se informa cuando el aforo se encuentre a su capacidad máxima, de manera que el ingreso de personas quede prohibido mientras el aforo vuelva a estar disponible.

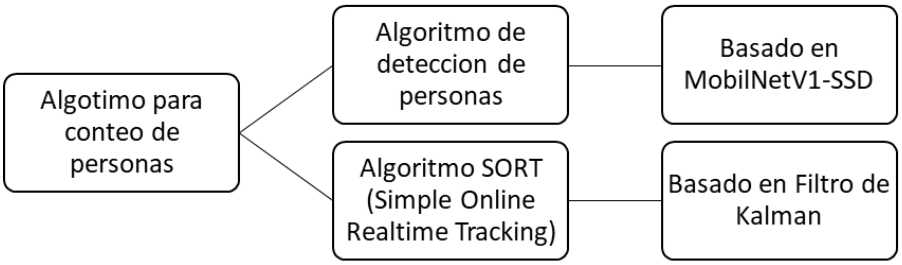
Mediante conectividad Wi-Fi se realiza el envío de los datos del contador de personas hacia un servidor externo, con lo cual sea posible supervisar dichos valores en cualquier dispositivo con conexión a Internet.

Algoritmo para el conteo de personas

Se propone un algoritmo para el conteo de personas formado por dos partes, como se muestra en la siguiente figura:

Figura 18

Diagrama de bloques del algoritmo para conteo de personas



A su vez cada parte de los algoritmos presentados anteriormente tienen una estructura interna que va a ser analizada a continuación.

Algoritmo de detección de personas

En este caso, se plantea utilizar un modelo entrenado para la detección de personas, basado en MobilenetV1, mismo que realice la detección de personas en tiempo real y que posteriormente se integre con el algoritmo de seguimiento SORT (Simple Online Real-Time Tracking). Mediante el modelo entrenado se propone crear un motor de inferencia del modelo a través de Tensor RT que es un optimizador para modelos de Inteligencia Artificial, con el cual se realice la inferencia del modelo de manera rápida y precisa.

El modelo de detección de personas basado en MobilenetV1 que se desea implementar es un modelo entrenado con Tensorflow, sin embargo, Nvidia presta scripts de demostración para convertir dichos

modelos a un formato UFF con el cual se pueda realizar la inferencia a través de Tensor RT, por lo que para esta acción se requiere del repositorio de demos de Tensor RT para modelos de detección.

Algoritmo SORT (Simple Online Realtime Tracking)

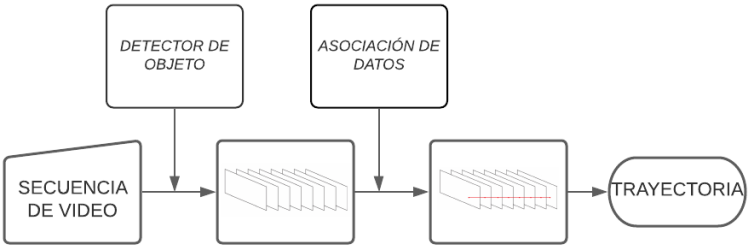
El algoritmo de seguimiento SORT es un algoritmo que se encuentra disponible para su uso libre, por lo que se lo usa para esta aplicación, este algoritmo en su forma original trabaja con la detección de objetos mediante Faster RCNN, sin embargo, se lo debe acoplar a nuestra necesidad, para la detección de personas mediante el modelo en MobilenetV1 SSD.

La implementación de dicho algoritmo está basada en la creación de un marco de seguimiento, más conocido como cuadro delimitador, el cual encierra al objetivo de la detección en cada cuadro que se está procesando, de manera que el seguimiento del objeto sucede entre el análisis de los fotogramas anteriores y actuales a lo largo del tiempo, en este caso del video que se está capturando.

Por esta razón fundamentalmente el algoritmo hace uso del tamaño del cuadro delimitador de la detección y de su posición para poder realizar una estimación del movimiento, con lo cual se puede realizar una asociación de datos que nos permita determinar el seguimiento de un objeto, en este caso el seguimiento de las personas.

Figura 19

Secuencia general de algoritmo de seguimiento



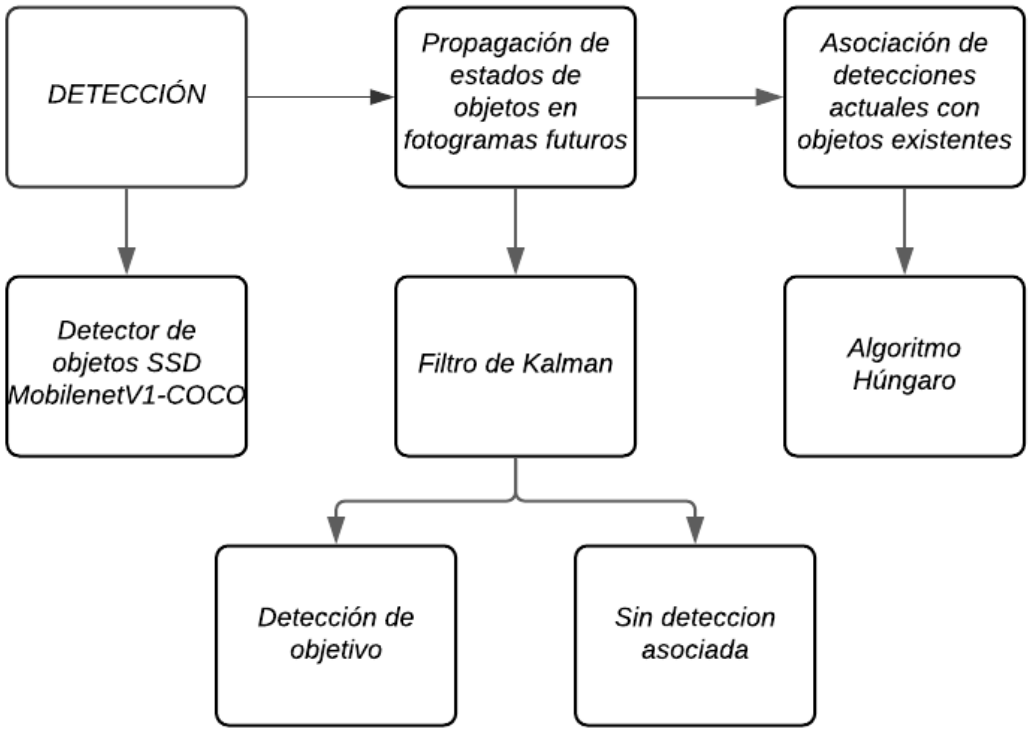
Integración del modelo de detección de personas y el algoritmo SORT

Como se detalló anteriormente para este aplicativo se realiza la detección de personas basado en un modelo pre-entrenado de la red SSD MobileNetv1, mismo que es el que proporciona al algoritmo SORT las detecciones y a partir de ello, este algoritmo hace uso del filtro de Kalman y el Método Húngaro para la identificación y posterior asociación de las detecciones de los marcos mediante la evaluación de las superposiciones de los cuadros delimitadores.

A pesar de que el filtro de Kalman y el Método Húngaro son técnicas poco complejas, alcanzan un alto rendimiento en cuanto a la gran velocidad de fotogramas que puede alcanzar.

Figura 20

Estructura del algoritmo SORT



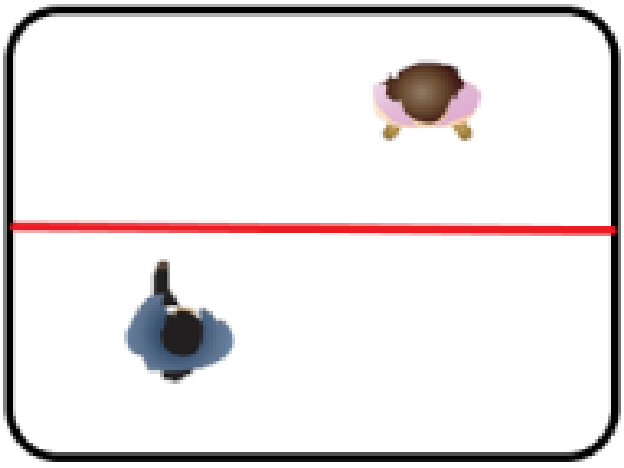
Conteo de personas

Una vez que se cuenta con los dos algoritmos detallados anteriormente funcionando se procede a realizar la configuración para realizar el conteo de personas. Esto se lo realiza mediante el uso de las detecciones y los rastreadores.

Para esto se define un límite justo por el centro de la imagen que nos va a servir para poder discriminar entre si una persona está ingresando o saliendo de la siguiente manera:

Figura 21

Límite establecido para el conteo de personas



Esto se lo realiza mediante la siguiente programación, en donde se definen los contadores que se estarán actualizando en base a los rastreadores, mismos que serán divididos en el caso de sobrepasar el límite establecido tanto de un lado hacia el otro para registrar la entrada de una persona, como en el sentido contrario para registrar la salida de la persona.

Modelo de detección de mascarillas

El modelo de detección de mascarillas se lo debe realizar usando Transfer Learning, ya que se va a utilizar un modelo pre-entrenado de la red SSD MobilenetV1. Para realizar esta tarea se considera la utilización de un contenedor proporcionado por NVIDIA denominado "JETSON-INFERENCE "el cual contiene todas las librerías y complementos necesarios para poder realizar la transferencia de aprendizaje de diferentes modelos de detección y de esta manera poder reentrenar los modelos según nuestra conveniencia. En este caso este contenedor implementa internamente Pytorch para la tarea de transfer Learning y Tensor RT para la inferencia de los modelos

Conjunto de datos

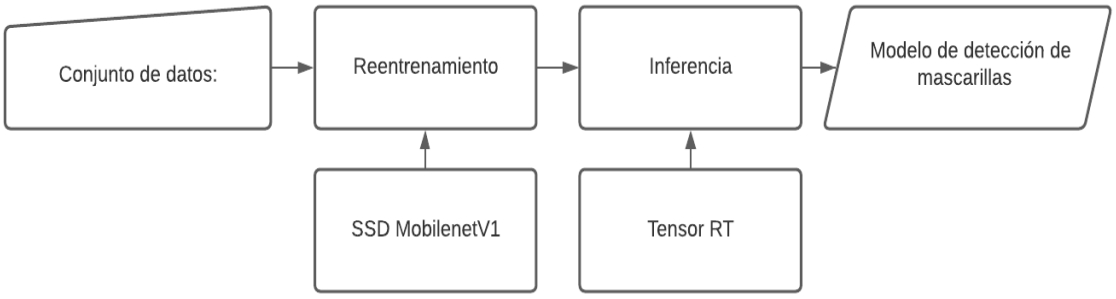
En este caso se considera la posibilidad de usar un conjunto de datos propio, o uno previamente preparado para esta tarea. En base a las consideraciones técnicas del volumen de datos, de la calidad de las imágenes, de la variabilidad y otros factores se opta por utilizar un conjunto de datos proporcionado por Keaggle, el cual contiene 853 imágenes de las 3 clases: Rostro con mascarilla, Rostro sin mascarilla y rostro con mascarilla mal ubicada

Diseño del modelo de detección

Como se observa en la figura, se presenta el desarrollo necesario para obtener el modelo de detección, teniendo como primer paso la obtención del conjunto de datos, con el cual se realice el reentrenamiento mediante la red MobilnetV1, para su posterior inferencia del modelo a través de Tensor RT. Tener en cuenta que todo este desarrollo se lo realiza sobre el Docker proporcionado por NIVDIA Jetson-Inference.

Figura 22

Componentes del modelo de detección de mascarillas



Diseño de la función de alarmas

Para la función de alarmas orientadas hacia el control de acceso y el control de aforo se tiene un mismo concepto, el cual estaba basado en trabajar como una función independiente pero que a la vez trabaje en paralelo con los modelos de detección de mascarillas y conteo de personas.

Este requerimiento surge debido a que los modelos de detección en tiempo real se encuentran en todo momento dando los resultados de detección, por lo que para activar una alarma se requiere establecer un punto de quiebre en el cual, se active la alarma y se ejecute por una sola vez. Por esta razón se plantea el uso de hilos, que es un algoritmo que permite que una tarea pueda ser ejecutada al mismo tiempo que otra tarea.

para ello se toma en cuenta el uso de la librería Threading para el procesamiento en paralelo con usando los objetos Thread y Lock:

threading.Thread:

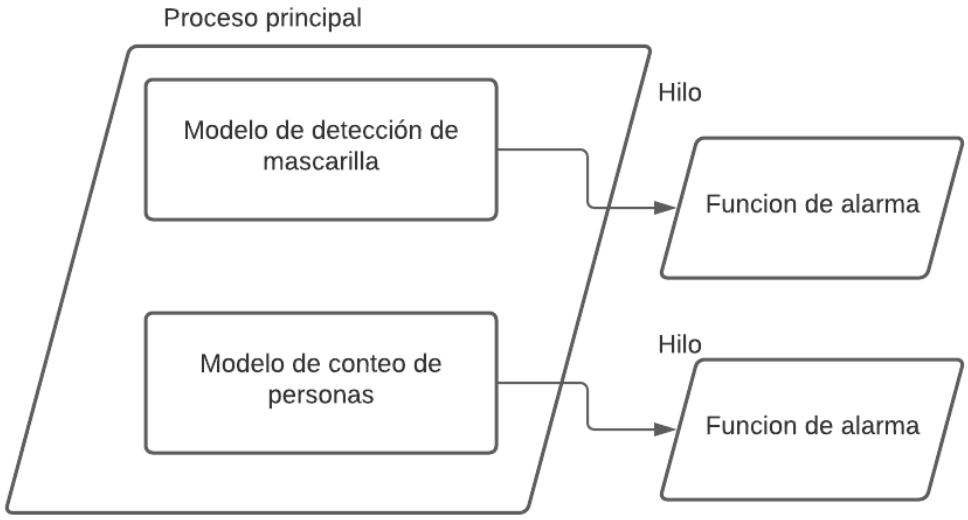
Es un objeto que permite crear procesos paralelos o *hilos*. Para crear un hilo, se debe tener definida una función y usarla como objetivo al momento de instanciar el objeto de la siguiente manera: Hilo = threading. Thread(target=function). Y para iniciarlo se utiliza el método *start:* Hilo.start()

threading.Lock:

Este objeto permite sincronizar los procesos entre hilos. Para crear un objeto de este tipo solo es necesario llamar al constructor: *lock = thread. Lock ()*. Un Lock sólo puede ser adquirido por un único hilo, pero puede ser liberado por cualquiera. Esto permite realizar una configuración tal que se ejecute un proceso solo si su Lock es liberado.

Figura 23

Función de alarma mediante Hilos



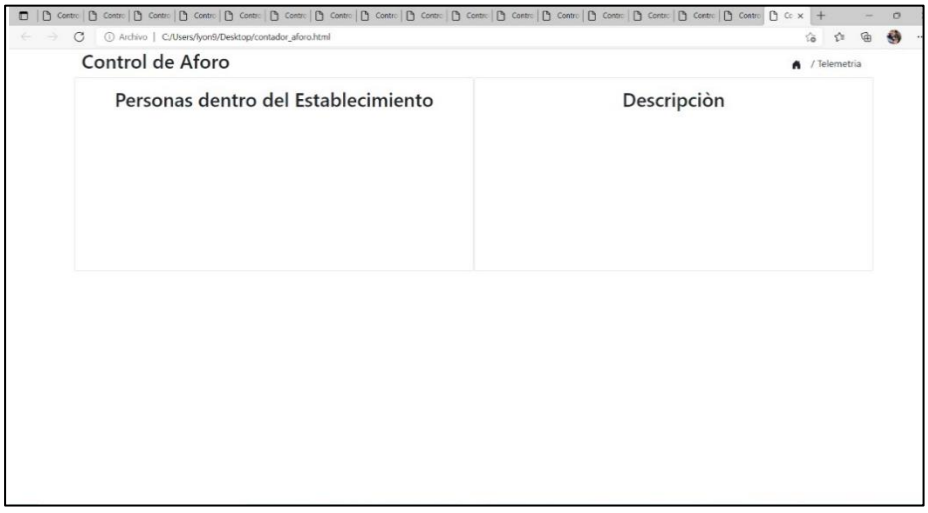
Consideraciones para el servidor externo

Debido al requerimiento de mostrar en un servidor externo el número de personas que se encuentren dentro de un local o espacio cerrado y la alerta de si el aforo se encuentra lleno, se propone usar un servidor ya establecido, que preste sus servicios para poder realizar una interfaz en la cual se pueda indicar los datos señalados anteriormente.

Para el envío de datos desde el contador de personas, mismo que se desarrolla en Python, se plantea el uso de la biblioteca MQTT que permite realizar el nexo entre el servidor y el modelo, con lo cual el envío de datos en tiempo real se convierte en una tarea sencilla.

Figura 24

Diseño genérico de la interfaz principal del servidor externo



Capitulo IV. Desarrollo e Implementación

Instalación de Jet Pack en la Jetson Nano

Para empezar, se usa Balena Etcher, el cual nos permite flashear el sistema operativo en la tarjeta SD, de modo que la imagen del sistema operativo pueda correr en la Nvidia Jetson Nano. Por otro lado, se requiere descargar de la página oficial de Nvidia el JetPack SDK, mismo que contiene por un lado el Sistema Operativo, la cual consta de una versión de Ubuntu y tenemos también el JetPack 4.5.1, es cual es un SDK que viene integrado en el sistema operativo con librerías instaladas como Opencv, Drivers de conexión con GPU, optimizadores de código como CUDA, CUDNN, para sacar el máximo provecho a la GPU.

Para realizar este procedimiento en primer lugar se procede a descargar él .zip del JetPack 4.5.1 de NVIDIA, para nuestro Kit de desarrollo Jetson Nano de 4GB, que se muestra a continuación:

Figura 25

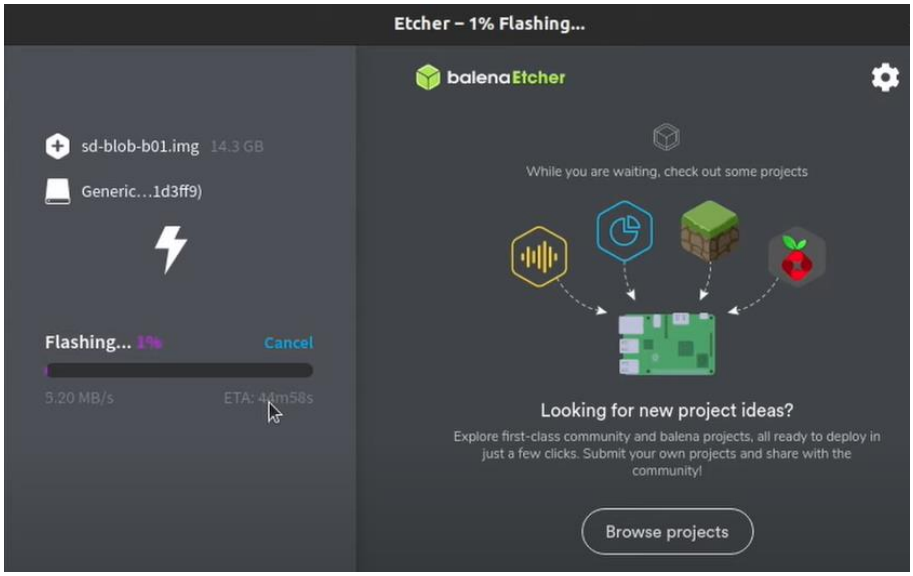
Sitio oficial para descarga de JetPack 4.5.1

A continuación, se presenta el software de Balena Etcher, en donde se realiza el siguiente procedimiento:

1. Se selecciona la pestaña **Flash from file** para indicar la ubicación de la imagen de JetPack 4.5.1
2. Seleccionamos en que dispositivo se desea instalarlo, en nuestro caso sobre nuestra tarjeta SD
3. Para finalizar se selecciona el botón Flash y se espera hasta que el procedimiento se complete.

Figura 26

Flasheo del Sistema Operativo



Una vez terminado el flasheo, se procede a insertar la tarjeta microSD en la Jetson Nano, y se procede a encenderla.

A continuación, se encuentra con una configuración de típica en donde se aceptan términos y se configuran valores determinados. En donde se aceptan los términos y condiciones, se define el idioma en que se va a trabajar, la disposición del teclado, y una parte importante es la de conexión a una red

inalámbrica, pues para la configuración del dispositivo es necesario que esté conectado a una red wifi, o en su defecto mediante un cable Ethernet. Luego también se escoge el huso horario en el que se encuentra, y a continuación se despliega una ventana en donde se configura el nombre del equipo y para el Usuario principal y también la contraseña para inicio de sesión.

A continuación, se asigna el valor en MB del espacio que la tarjeta que se destina para el Sistema Operativo, y la selección del NVPModel Mode, que en nuestro caso al estar alimentado por el conector de barril será el modo MAXN. De esta manera comienza la instalación del Sistema Operativo.

Figura 27

Interfaz principal de la tarjeta Jetson Nano

Implementación del Modelo de detección de Mascarillas

Para este diseño se va a utilizar un Docker proporcionado por Nvidia para la plataforma de la Jetson Nano, en el cual está instalado internamente todas las librerías necesarias para el desarrollo de modelos de detección. Este Docker nos ofrece una plataforma integrada que permite usar TensorRT para implementar modelos de redes neuronales sobre la tarjeta Jetson Nano con alta eficiencia.

Figura 28

Interfaz para Jetson Nano e Inteligencia Artificial

Nota. Tomado de (Nvidia, 2022)

Para obtener este repositorio en nuestra tarjeta Jetson Nano, en un terminal se procede a ejecutar los siguientes comandos, con los cuales se clona el repositorio y también se corre el Docker, con lo cual ya estaríamos trabajando sobre el mismo.

```
git clone --recursive https://github.com/dusty-nv/jetson-inference
cd jetson-inference
docker/run.sh
```

Obtención del modelo pre-entrenado

El modelo pre-entrenado de la red SSD MobilenetV1 es una agrupación de modelos enfocados en la detección de objetos que se encuentran pre-entrenado y listos para ser entrenados en base a la aplicación deseada. Para ejecutar esta acción se ejecuta el siguiente código en un terminal

```
wget https://nvidia.box.com/shared/static/djf5w54rjvpqocsiztzaandq1m3avr7c.pth -O models/mobilenet-v1-ssd-mp-0_675.pth
```

Obtención del conjunto de datos personalizado de Kaggle

Para un entrenamiento de algoritmos de detección de objetos siempre es necesario una gran cantidad de información que son utilizados como datos de entrenamiento, y el porqué de este requerimiento es debido a que mientras se realiza un entrenamiento con más datos, los resultados del entrenamiento tendrán mejores resultados. Pero no solo se trata de entrenar el algoritmo con miles de imágenes que finalmente se asemejan o que tienen una perspectiva similar, pues lo que verdaderamente contribuye a tener un mejor entrenamiento es la variabilidad de dichos datos, y es algo de lo que constantemente se menciona en foros o discusiones sobre entrenamiento de redes neuronales.

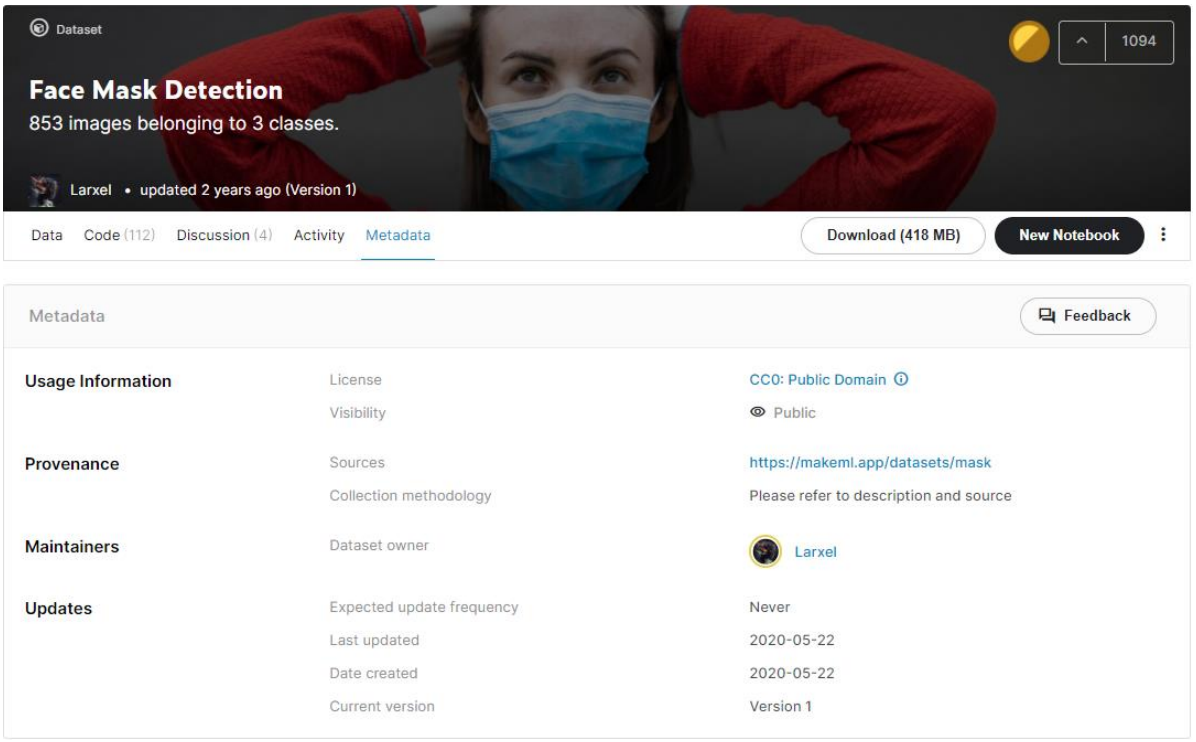
La variabilidad de los datos hace referencia a que los datos de entrenamiento sean tomados en diferentes entornos por ejemplo de iluminación; de perspectivas como frontal, lateral, superior, entre otras; rotación o incluso enfoque. Todas estas variantes permiten que un conjunto de datos sea el más adecuado para un entrenamiento, y sobre todo si dicho modelo va a trabajar en tiempo real, que va a ser el modo de trabajo en el prototipo, pues de esta manera se asegura que en la captura del video, el objeto a detectar no necesariamente este de manera frontal para ser detectado, sino indiferente de su posición, dicho objeto se detecte, de igual manera que los porcentajes de confianza de la detección no se vean tan afectados cuando el entorno en el que se está desarrollando la detección carezca de iluminación, u otros factores que puedan afectar a la calidad de la imagen.

Por esta razón en el presente trabajo se hace uso de un conjunto de datos personalizado que proporciona Kaggle, que como se muestra en la siguiente imagen es un Dataset de Dominio Público, y que contiene 853 imágenes pertenecientes a 3 clases que son:

- Con mascarilla
- Sin mascarilla
- Mascarilla usada incorrectamente

Figura 29

Dataset para detección de mascarilla de Kaggle



Nota. Tomado de (Kaggle, 2020)

Este directorio contiene 2 carpetas que vienen a ser las carpetas de las imágenes y la carpeta que contiene las anotaciones

Carpeta de imágenes

En esta carpeta se encuentran localizadas las 853 imágenes de extensión PNG de las 3 clases que se muestra a continuación:

Figura 30

Ejemplos de imágenes de las 3 clases del Dataset

Nota. Tomado de (Kaggle, 2020)

Como se observa en la figura, se tiene un ejemplo de las imágenes de cada clase, en la primera una persona con la mascarilla usada correctamente, en la segunda una persona con la mascarilla ubicada por debajo de la nariz, por lo que esta imagen pertenece a la clase de Mascarilla usada incorrectamente, y la tercera imagen una persona que no lleva mascarilla.

Carpeta de Anotaciones

En esta carpeta se encuentran los archivos de extensión HTML asociados a cada una de las imágenes, mismas que contienen los cuadros delimitadores o bounding box, situados en las regiones de interés.

A continuación, se muestra un ejemplo de un archivo HTML con su imagen original en la cual se detalla el cuadro delimitador enfocado en el cuadro que contiene el objeto de detección, en este caso la mascarilla.

Figura 31

Imagen JPG y su correspondiente archivo HTML

```
<annotation>
<folder>images</folder>
<filename>maksssksksss10.png</filename>
<size>
<width>301</width>
<height>400</height>
<depth>3</depth>
</size>
<segmented>0</segmented>
<object>
<name>with_mask</name>
<pose>Unspecified</pose>
<truncated>0</truncated>
<occluded>0</occluded>
<difficult>0</difficult>
<bndbox>
<xmin>98</xmin>
<ymin>267</ymin>
<xmax>194</xmax>
<ymax>383</ymax>
</bndbox>
</object>
    </annotation>
```

Nota. Tomado de (Kaggle, 2020)

El código anterior, es el generado cuando se realiza la etiqueta de la imagen, internamente obtiene los datos de largo y ancho de la imagen, y a la vez realiza un cuadro delimitador en la región de interés, en este caso en la región en donde se lleva la mascarilla, con lo que dicha etiqueta se asigna a la clase que corresponda.

Mediante estas asignaciones es posible realizar el entrenamiento de manera directa, pues el algoritmo de entrenamiento es capaz de discriminar en una secuencia de video las diferentes clases para las que fue entrenado.

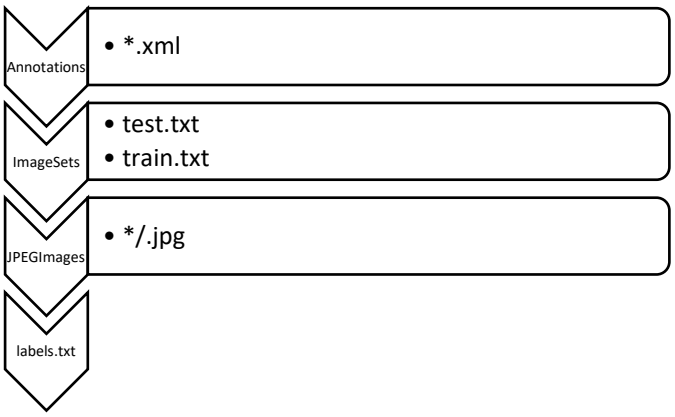
Conversión del Dataset a formato VOC de Pascal

El conjunto de datos que previamente fue obtenido desde Keaggle se lo convierte en un formato especifico, el formato VOC de Pascal es un formato que pues como indican sus siglas PASCAL (Pattern Analysis, Statistical Modelling, and Computational Learning) es una Red de Excelencia de la UE precisamente orientado a este tipo de aplicaciones.

Para ello primero se necesita cumplir con una estructura de datos que es necesaria para cualquier conjunto de datos que estén siendo preparados para un entrenamiento, en la siguiente figura se muestra cómo debe estar estructurado el directorio del conjunto de datos.

Figura 32

Estructura de directorios de Pascal VOC



A continuación, se detalla que contiene cada uno de los directorios y archivos detallados en el esquema anterior:

Carpeta Annotations: Esta carpeta contiene todos los archivos HTML asociadas a las imágenes, mismos que son proporcionados por el Dataset descargado.

Carpeta ImageSets: Esta carpeta contiene los archivos test.txt y trainval.txt, mismos que son los archivos que contendrán las rutas de los datos tanto de entrenamiento como de validación

Carpeta JPEGImages: Esta carpeta contiene los archivos .jpg otorgados por el Dataset descargado desde Keaggle

Archivo Labels.txt: Este archivo de texto contiene en su interior la lista de clases que se van a detectar que son: Con mascarilla, Sin mascarilla y Mascarilla usada incorrectamente.

Script para creación de archivos trainval.txt y test.txt

En contenedor de del proyecto de detección de mascarilla se encuentra script en Python denominado prepare.py que permite realizar la tarea de obtener todas las rutas a los datos de entrenamiento y validación. En este caso se realiza una asignación de datos, el 10% para testeo y 90% para entrenamiento, que es una asignación convencional para entrenamiento de modelos de detección.

Una vez que se corre este script mediante el siguiente comando, ya se tiene toda la estructura mostrada anteriormente de manera correcta, con lo cual el modelo está listo para entrenar.

```
python prepare.py
```

Adición de memoria de intercambio para el entrenamiento

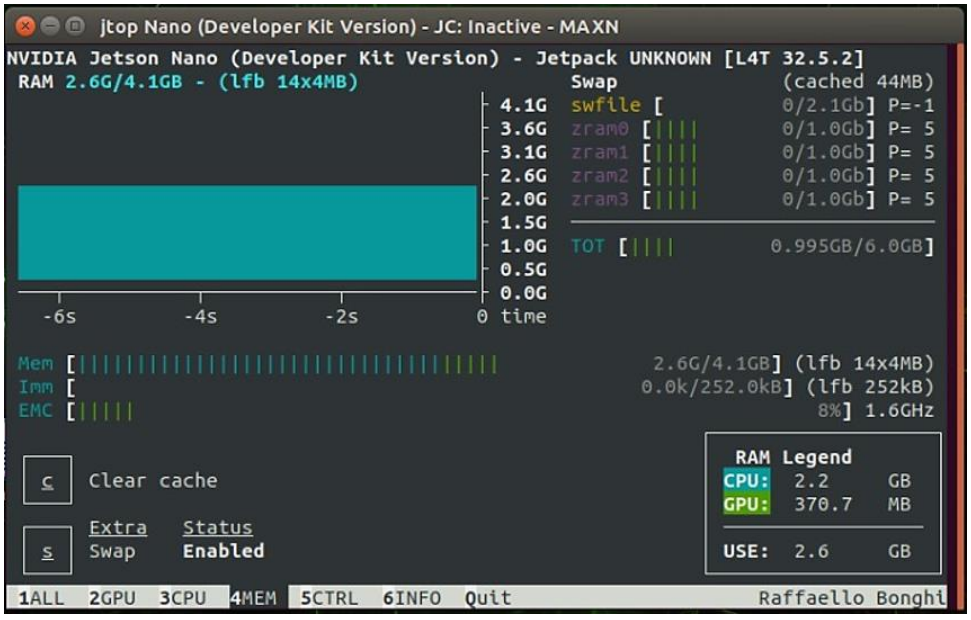
El entrenamiento de un modelo de detección sugiere una gran capacidad de procesamiento, por lo que las características del dispositivo como su procesador, su GPU, su memoria RAM son elementos de gran importancia en términos de rapidez del entrenamiento.

Debido a que la tarjeta Jetson Nano presenta características limitadas en cuanto a memoria, se realiza una configuración de la memoria de intercambio para tener mayor disponibilidad de memoria durante la operación de entrenamiento.

Esto se lo realiza mediante la instalación de una herramienta de depuración llamada JTOP , la cual muestra toda información de las CPUs, GPU, memorias y otros componentes de la tarjeta Jetson Nano, y también el poder adicionar la memoria de intercambio en la interfaz que se muestra a continuación:

Figura 33

Herramienta JTOP para adición de memoria de intercambio



Reentrenamiento de SSD-MobileNet

Para realizar el reentrenamiento de la red SSD-MobileNet el Docker en el que se está trabajando proporciona un algoritmo de entrenamiento, mismo que requiere de algunos argumentos que se muestran a continuación:

Tabla 1

Parámetros para reentrenamiento

Argumento	Valor	Descripción
--data	data/	Ubicación del directorio del conjunto de datos
--model-dir	models/	Directorio para generar los puntos de control del modelo entrenado
--batch-size	4	Cantidad de imágenes tomadas en cada iteración
--epochs	50	Numero de épocas para el entrenamiento

Entonces se corre el siguiente código sobre un terminal:

```
python3 train_ssd.py --dataset-type=voc --data=data/MaskDetector --model-dir=models/mymodel –batch-size=4 –-epochs=50
```

El entrenamiento es un proceso bastante largo, y mucho más tomando en cuenta que el entrenamiento se lo está llevando a cabo en la tarjeta Jetson Nano, que en términos de hardware es bastante limitada, sin embargo, a continuación, se presenta una tabla con el rendimiento de la tarjeta durante el entrenamiento en términos de procesamiento de imágenes por segundo y duración de entrenamiento por época.

Tabla 2

Parámetros de duración del entrenamiento

	Imágenes/ s	Tiempo por época [s]
Jetson Nano	≈ 5	≈ 2 min 50 seg
Duración del entrenamiento	≈ 2h 22min	

Tabla 3

Pérdidas de validación durante el entrenamiento

Época	Perdida De Validación	Época	Perdida De Validación	Época	Perdida De Validación
1	4.8503	18	2.7249	35	2.5462
2	3.9592	19	2.8452	36	2.5098
3	3.6149	20	2.8517	37	2.5002
4	3.4253	21	2.8152	38	2.4852
5	3.2256	22	2.6800	39	2.4452
6	3.5519	23	2.7397	40	2.4516
7	3.2213	24	2.7031	41	2.4447
8	3.1783	25	2.6682	42	2.5236
9	3.4477	26	2.7556	43	2.4309
10	3.0455	27	2.7073	44	2.3646
11	2.9420	28	2.5765	45	2.4424
12	2.8204	29	2.5960	46	2.3707
13	2.9786	30	2.5105	47	2.4106
14	2.8404	31	2.6045	48	2.3817
15	2.8054	32	2.6469	49	2.3824
16	2.9604	33	2.6173	50	2.3112
17	2.8425	34	2.6847		

Conversión del modelo a ONNX

Para que se pueda realizar la inferencia con TensorRT es necesario un modelo de formato ONNX (Open Neural Network Exchange), para lo cual se usa el script proporcionado por el contenedor de NVIDIA, el cual nos proporciona el modelo en dicho formato, esto se realiza mediante el siguiente comando:

```
python3 onnx_export.py --model-dir=models/mymodel
```

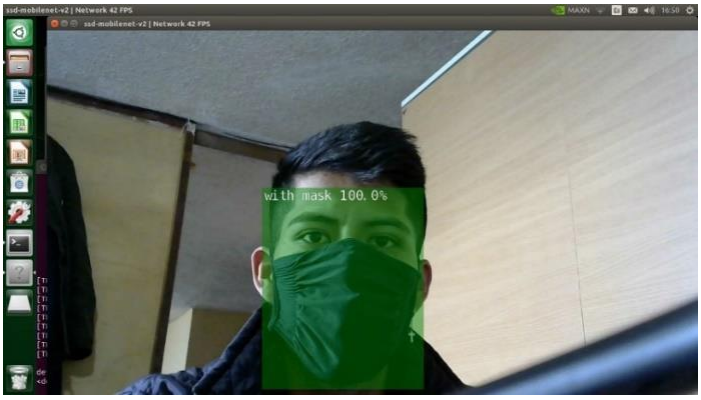
Una vez que tenemos el modelo en formato ONNX ya es posible correr el modelo a través del archivo detectnet.py, dicho script realiza la inferencia del modelo y permite realizar la detección de mascarillas en tiempo real con el siguiente comando:

```
detectnet --model=$NET/ssd-mobilenet.onnx --labels=$NET/labels.txt --input-blob=input_0 --
output-cvg=scores --output-bbox=boxes --threshold=0.2 --flip-method=rotate-180 csi://0
```

Al ejecutar el comando anterior se obtiene el modelo de detección de mascarilla que se muestra a continuación:

Figura 34

Modelo de detección de mascarilla



Implementación del contador de personas

Detección de personas

La detección de personas es una aplicación que por mucho tiempo se ha venido realizando, puesto que es un objetivo muy común en los detectores, Por esta razón la tarea de entrenar un detector de personas resulta una tarea repetitiva, tomando en cuenta que se tiene acceso libre a modelos de detección de personas que en primera instancia han sido resultado de varias pruebas y que además ofrecen buenos resultados de rendimiento. Por esta razón para el conteo de personas se usa un modelo entrenado para la detección de personas basado en la red MobilenetV1 SSD-COCO. Pero este modelo escogido se encuentra entrenado con Tensorflow, por lo que el reto aquí se trata en convertir el modelo a un formato en el que se pueda crear un motor de inferencia a través de Tensor RT para que el modelo pueda correr sobre la tarjeta Jetson Nano. Para ello se hace uso del repositorio otorgado por NVIDIA, en donde se tienen ejemplos que demuestran cómo optimizar modelos Caffe, Tensorflow, DarkNet, PyTorch con TensorRT y hacer inferencias en plataformas NVIDIA Jetson. Para ello se clona el repositorio y se hace uso del demo TRT_SSD.

```
git clone https://github.com/jkjung-avt/tensorrt_demos.git
cd tensorrt_demos/ssd
./install.sh
$ ./build_engines.sh
```

Con los comandos anteriores se clona el repositorio en la tarjeta Jetson Nano, se accede al directorio y se mediante script install.sh se realiza la instalación de las dependencias necesarias para poder correr el demo, dentro de este archivo se realiza la instalación de PyCUDA en caso de ser necesario, se crea un parche ‘graphsurgeon.py’ en TensorRT y se crea un enlace simbólico de libflattenconcat.so.

Además, con el archivo build_engines.sh se convierte un modelo SSD (pb) a UFF y posteriormente construye el motor TensorRT. Con lo cual el modelo de detección de personas se encentra listo para poder integrarse con la parte de detección del algoritmo SORT.

Algoritmo SORT Para el uso de este algoritmo se accede a la descarga del algoritmo SORT original, desarrollado por Alex Bewley, repositorio que se encuentra en GitHub y que nos proporciona una carpeta en donde se encuentran todos los componentes para que el algoritmo pueda funcionar. A continuación, con los parámetros mostrados se clona el repositorio y además se instala los requerimientos necesarios.

```
git clone https://github.com/abewley/sort.git
cd sort-master
pip install -r requirements.txt
```

El algoritmo original SORT trabaja con detecciones de objetos basada en Faster RCNN, por lo cual se realiza una configuración en cuanto a la parte de detección de este algoritmo, pues para la aplicación del contador de personas, de debe ingresar como argumento de entrada al algoritmo SORT la detección de personas, para que en base a ello se realice la tarea de seguimiento.

Integración y de algoritmos de detección de personas y SORT

Una vez que se tiene ambos algoritmos se procede a realizar una fusión de los mimos, en donde se realiza el reemplazo de la componente de detección en el algoritmo SORT, implementando de esta manera la inferencia del modelo de detección de personas como argumento de entrada. Si bien es cierto que el algoritmo SORT se encarga de realizar el seguimiento de los objetos detectados, para el conteo de personas se requiere una configuración adicional con la cual se pueda determinar que las asociaciones que han sido asignadas a los rastreadores puedan determinarse entre un límite que permita discernir si la persona se

encuentra ingresando o saliendo, para ello, en donde se asigna los rastreadores coincidentes con detecciones asignadas, se implementa el siguiente código:

```
for t, trk in enumerate(trackers):
        if t not in unmatched_trks:
            d = matched [np.where(matched[:, 1] == t)[0], 0]
            trk. update(boxes[d, :][0])
            xmin, ymin, xmax, ymax = boxes [d, :][0]
            cy = int ((ymin + ymax) / 2)
            #IN count
            if idstp[trk.id][0][1] < H // 2 and cy > H // 2 and trk.id not in idcnt:
                incnt += 1
                idcnt. append(trk.id)
            #OUT count
            elif idstp[trk.id][0][1] > H // 2 and cy < H // 2 and trk.id not in idcnt:
                outcnt += 1
                idcnt. append(trk.id)
```

En el código anterior se realiza una validación para que en base a los rastreadores que han sido asignados con las detecciones se los divida entre un límite imaginario dado por los factores cy que realiza la división de la imagen en el eje y H/2 que determina la altura a nivel medio, con lo cual aquellos rastreadores que sobrepasen el límite establecido desde la parte superior de la altura media contara como ingreso, y en si defecto los que sobrepasen el límite desde la parte inferior de la altura media se considerara como salida.

Implementación de las funciones de alarma

Función de alarma para detector de mascarilla

La función de alarma para el modelo de detección de mascarilla requiere previamente una validación de los parámetros de las detecciones, con lo cual se pueda asegurar que la activación de dicha alarma solamente se ejecute cuando la detección se cumpla entre los limites definidos.

Validación de parámetros del detector de mascarillas para activación de alarmas

Sobre el algoritmo detectnet.py que permite correr el modelo de detección de mascarillas se realiza validaciones para que el control de acceso sea efectivo.

Las salidas del modelo de detección con las cuales se va a trabajar son la siguientes:

len(Detection): Salida que indica el valor del número de detecciones

Detection.Confidence: Salida que entrega el porcentaje de confianza de la detección

Detection.ClassID: Salida que entrega la clase que se está detectando

Validación de detección

En primera instancia se realiza la validación para que se trabaje con una sola detección, esto debido a que al ser un control de acceso el proceso de detección debe ser uno a uno, para que la detección sea controlada y ordenada. De igual manera esta validación permite que la detección sea correcta puesto que el modelo en condiciones de lejanía o perspectiva suele realizar detecciones múltiples, con lo cual se puede asegurar que la persona que es objeto de la detección se encuentre posicionada correctamente.

Validación de porcentaje de confianza

En vista de los resultados de los porcentajes de confianza para cada clase, se establece una validación con este parámetro, con el objetivo de asegurar una correcta detección, en el caso de las clases Con mascarilla y Sin Mascarilla la validación se realiza para cuando las detecciones superen el 90% de porcentaje de confianza, mientras que para la clase Mascarilla usada incorrectamente se establece un porcentaje de confianza de 60% debido al rendimiento del modelo de detección

Validación de clase

Debido a que para el control de acceso se requiere alarmas en base a cada clase, se realiza la validación de cada clase, con el fin de tener la capacidad de activar las alarmas para cada clase de las detecciones. Estas validaciones se las realiza mediante una cadena de IFS anidados que permiten que las alarmas se activen siempre y cuando estas validaciones se cumplan, a continuación, se muestra el código implementado para realizar las validaciones correspondientes:

```
if (len(detections)==1):
        if(detection.Confidence>0.90):
                if(detection.ClassID==2):
                alarma.sin_mascarilla()
                if(detection.ClassID==1):
                alarma.con_mascarilla()
        if(detection.Confidence>0.40):
                if(detection.ClassID==3):
                alarma.mal_mascarilla()
```

Función de alarma para contador de personas

El contador de personas requiere las validaciones basadas en el parámetro de personas que se encuentran dentro del local o espacio cerrado, debido a que, en base a este se debe activar la alarma para cuando el aforo se encuentre al 50,75 y 100 %.

Validación de parámetros del contador de personas para activación de alarmas. En este caso la única validación requerida es tomar en cuenta el porcentaje del aforo permitido y realizar una comparativa con el número de personas que se encuentran dentro del establecimiento para los porcentajes de 50, 75 y 100% en donde deben activarse las alarmas, esto se realiza mediante el siguiente código:

```
CNT = incnt-outcnt
if(CNT == round(0.5*AFORO)):
        alarma.aforo50()
if(CNT == round(0.75*AFORO)):
        alarma.aforo75()
if(CNT > AFORO):
        alarma.aforolleno()
```

Implementación de las alarmas para el control de acceso y para el control de aforoPara la aplicación del control de acceso y control de aforo se requiere una función que permita ejecutar las alarmas, sin embargo, esta tarea debe realizarse en paralelo a las detecciones que se realizan todo el tiempo, por lo que se realiza el siguiente código.

En el siguiente fragmento de código se realiza la importación de librerías, la creación de la clase para la alarma y la inicialización de las variables de estado de la clase.

```
import threading
import playsound
class alarm():
    __status = 0
    __alarms = [
        None,
        playsound.playsound,"track.mp3",  # alerta1
        playsound.playsound,"track1.mp3",  # alerta2
        playsound.playsound,"track.2mp3",  # alerta3
    ]
```

A continuación, se crea el objeto Lock, mismo que sincronizará al hilo de alarmas y el proceso principal , la función de inicialización del hilo y la función de destrucción que permite finalizar el hilo.

```
__lock_alarm = threading.Lock()
    def __init__(self) -> None:
        self.__lock_alarm.acquire()
        self.__hilo = threading.Thread(target=self.__alarma)
        self.__hilo.start()
        pass
    def __del__(self):
        print("Terminando Hilo")
        while not self.__alarm_activate(-1):
            print('.',end='')
            pass
        self.__hilo.join()
```

Finalmente, en el siguiente extracto de código se define el método para reproducir la alarma en base al estado que se encuentre, se define el método que permite el registro de un nuevo estado y finalmente la definición de los métodos que registran los nuevos estados o su reinicializarían

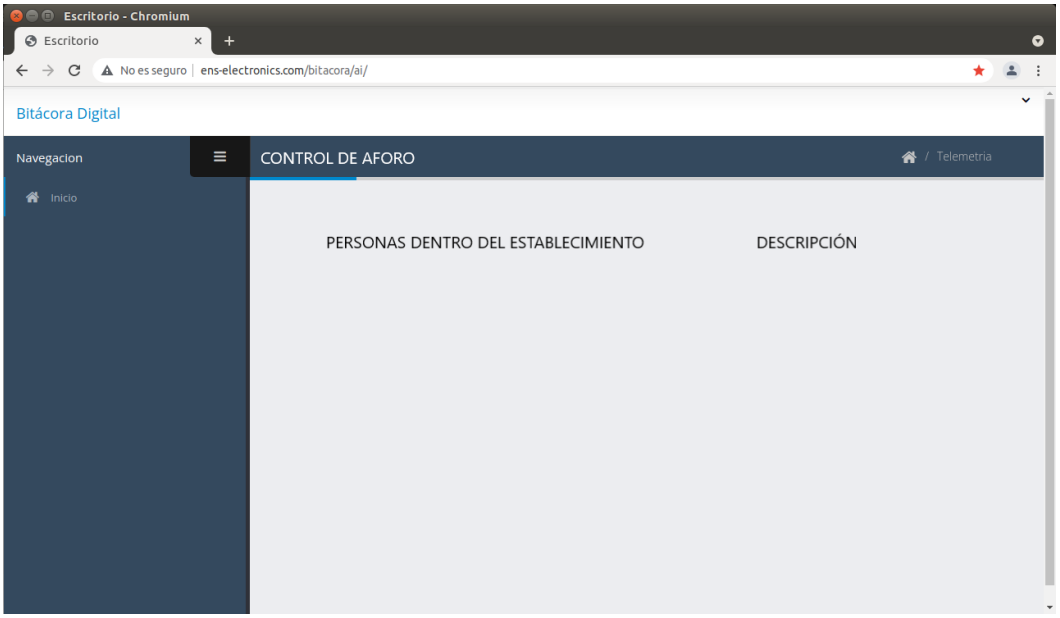
```
        def __alarma(self):
            while True:
                if self.__status >=0:
                    if self.__lock_alarm.acquire(timeout=5):
                        engine.say(self.__alarms[self.__status])
                        engine.runAndWait()
                    else:
                        self.__status=0
                else:
                        break
        def __alarm_activate(self, n):
            if self.__status != n:
                self.__status = n
                if self.__lock_alarm.locked():
                    self.__lock_alarm.release()
                    return True
                return False
        def sin_personas(self):
            self.__status = 0
        def con_mascarilla(self):
            self.__alarm_activate(1)
        def sin_mascarilla(self):
            #if self.__status == 0:
                self.__alarm_activate(2)
        def mal_mascarilla(self):
            #if self.__status in [0,2]:
                self.__alarm_activate(3)
```

Implementación del servidor externo

Para la implementación del servidor se hace uso de un servidor particular, sobre el cual se diseña una interfaz sencilla en donde se muestran los dos valores asociados al conteo de personas que son: número de personas que se encuentran dentro del establecimiento y alerta de aforo si se encuentra incompleto o lleno, de esta manera se tiene la implementación que se muestra a continuación:

Figura 35

Servidor externo



Para el envío de los datos hacia el servidor se implementó el siguiente código, con el cual se envía el dato CNT que contiene el número de personas que se encuentran dentro del establecimiento y la cadena de texto que tiene dos estados, Aforo incompleto y aforo lleno. Estas validaciones se las realiza en el algoritmo SORT haciendo uso de los valore CNT que es el número de personas que se encuentran dentro

del establecimiento, y del valor configurable Aforo, el cual depende del aforo permitido en el escenario de prueba en que se vaya a poner a funcionar.

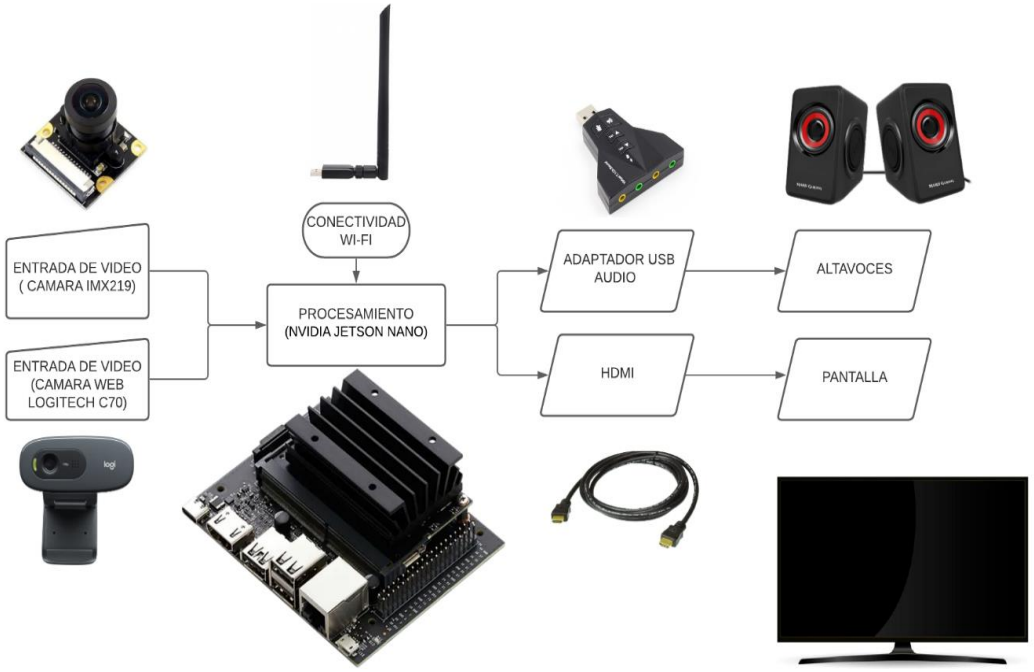
```
import paho.mqtt.client as mqtt
mqtt_user = 'carlosluis'
mqtt_pass = '4tablasmqws***'
client = mqtt.Client()
client.username_pw_set(mqtt_user, mqtt_pass)
client.connect('157.230.209.226', 1883, 60)
print("Estamos listos")
client.publish('contador1/', "CNT")
if(CNT ≥ Aforo):
        client.publish('contador2/', "Aforo lleno")
else:
        client.publish('contador2/', "Aforo incompleto")
client.loop_forever()
```

Montaje del prototipo

El prototipo tiene una estructura convencional de tipo: entradas, unidad de procesamiento y salidas, sin embargo, en este caso al trabajar con una Jetson Nano, cabe recalcar que esta tarjeta de desarrollo no cuenta por ejemplo con un módulo de conexión WI-Fi, o salidas de tipo audio directas, por lo que existen también los componentes de conectividad y adaptación.

Figura 36

Componentes del prototipo



Periféricos de entrada

Cámara IMX 219La cámara IMX 219 está conectada hacia el puerto para cámara tipo MIPI CSI de la Jetson Nano, misma que está encargada de entregar una señal de video como entrada para la detección de mascarillas. Consta con una resolución máxima de 3280 x 2464 pixeles.

Cámara Logitech C70

Por otro lado, la cámara web Logitech C70 conectada en un Puerto USB de la Jetson Nano, se encarga de dotar de la señal de video para el contador de personas. Consta con una resolución máxima: 720p/30 fps.

Unidad de procesamiento

El prototipo tiene como parte central y enfocada en el procesamiento de los algoritmos de detección, la tarjeta de desarrollo Nvidia Jetson Nano, misma que específicamente ha sido desarrollada para la aplicación de modelos basados en Inteligencia artificial.

Para ello cuenta con una GPU de 128 núcleos Maxwell, un procesador Quad-core ARM A57 que trabaja a 1.43 GHz y memoria RAM de 4GB con 64-bit LPDDR4. Estas especificaciones son de gran importancia debido a que en base a ellas es posible el tener un buen rendimiento en el entrenamiento e inferencia de los modelos de detección.

Este dispositivo se encarga específicamente del entrenamiento del modelo de detección de mascarillas y también de la inferencia de dicho modelo a través de TensorRT, así también como de la inferencia del modelo de detección de personas y de la ejecución del algoritmo de seguimiento SORT.

Conectividad

La tarjeta Jetson Nano como tal no cuenta con un módulo incorporado para la conexión WI-FI, por lo que es necesario añadir un adaptador tipo USB que nos provea de este requerimiento. Se hace uso de un adaptador de tarjeta de red USB WiFi de 1200 Mbps, el cual no requiere de un driver de instalación y que es compatible con Linux, sistema en el cual está basado el Sistema Operativo de la Jetson Nano.

Adaptadores

Adaptador de audio

Debido a que la tarjeta Jetson Nano no posee salidas de audio incorporadas, es necesario el uso de un adaptador de audio tipo USB tipo 2.0, que tiene 2 entradas de audífono tipo 3.5 mm y 2 entradas de

micrófono de tipo 3.5 mm. Con lo cual la tarjeta Jetson Nano obtiene la capacidad de conectar directamente dispositivos de audio como Altavoces

HDMI

La tarjeta Jetson Nano cuenta con un puerto de salida tipo HDMI para poder realizar una conexión hacia un periférico de salida, por lo que se hace uso de este para obtener una salida visual sobre una pantalla.

Periféricos de salida

Altavoces

Debido al requerimiento del control de acceso mediante la detección de mascarillas, es necesario un par de altavoces que sean capaces de reproducir una señal auditiva cuando se realice la detección.

Pantalla (opcional)

En vista de que la detección de mascarilla y el conteo de personas va a ser realizado en tiempo real, dichas aplicaciones generan una salida que puede ser de interés para la supervisión del control de acceso y del conteo de personas, por lo que se requiere una pantalla, en donde se pueda observar en tiempo real la salida de video con sus respectivas detecciones. Por otro lado, para poder observar los datos en el servidor externo, de igual manera se requiere una pantalla que tenga conectividad a internet, que en realidad puede ser visualizado en cualquier dispositivo, pero en este caso la pantalla puede ser usada también para visualizar los valores mostrados en el servidor externo.

Todos los elementos mencionados anteriormente se ensamblaron sobre la carcasa impresa en 3D, de manera que el prototipo quedo de la siguiente manera:

Figura 37

Ensamble del prototipo

Capítulo IV. Pruebas y Resultados

Pruebas sobre diferentes entornos de iluminación

Para las pruebas que se realizan a continuación cabe mencionar que los porcentajes mostrados en las tablas son resultado de un promedio de las pruebas recurrentes que se realizó con el prototipo. En el caso del detector de mascarillas, los valores son un promedio aproximado de 10 pruebas de detección recurrente, mientras que para el contador de personas se trabaja con 8 detecciones recurrentes.

Detector de mascarilla

Pruebas de posición con respecto a la luz

Como es sabido, los niveles de luz son un factor importante dentro de los modelos de detección de objetos, por lo cual se realizaron pruebas en diferentes posiciones con respecto a la luz.

Para esta tarea se colocó el dispositivo que realiza la detección de mascarillas en 4 posiciones diferentes, respecto a la posición de la fuente de luz, con lo cual se pudo obtener los diferentes resultados de porcentajes de confianza en las detecciones que se muestra en la siguiente tabla:

Tabla 4

Pruebas de posición con respecto a la luz

Posición respecto a la fuente de luz	Con mascarilla (% de confianza)	Mascarilla usada incorrectamente (% de confianza)	Sin mascarilla (% de confianza)
Frontal (CONTRALUZ)	60%	Detección errónea	Detección errónea
Lateral izquierda	98%	66%	98%
Lateral derecha	98%	68%	98%
Opuesta	99%	75%	99%

Como se puede observar en la tabla anterior, cuando se coloca el dispositivo en posición frontal a la fuente de luz, su rendimiento es bajo, e incluso presenta detecciones falsas de las clases de Mascarilla usada incorrectamente y Sin mascarilla. Esto debido a que el objeto de detección (persona), se encuentra justo delante de la fuente de luz; entonces el efecto será que se detecte una silueta oscura, con lo cual la detección es inviable.

En el caso de las posiciones laterales del dispositivo se observa que las detecciones de las 3 clases son acertadas, y que los porcentajes de confianza de las detecciones son óptimos en el caso de las clases Con Mascarilla y Sin Mascarilla, y aceptable en el caso de la clase Mascarilla usada incorrectamente.

Y finalmente, en el caso de la posición opuesto del dispositivo respecto a la fuente de luz se obtuvieron resultados similares a las posiciones laterales, sin embargo, en la clase Mascarilla usada incorrectamente si se observó un incremento un tanto significativo.

En vista de estos resultados, y tomando en cuenta que el dispositivo debe ser ubicado en el acceso de los escarnios de prueba se determinó que la mejor posición de instalación es la posición lateral, puesto que la detección opuesta a la fuente de luz no es consecuente con la aplicación para el control de acceso.

Pruebas de detección a diferentes horarios y nivel de iluminación

En vista de los resultados anteriores, esta prueba se realizó con una posición lateral con respecto a la fuente de luz, en diferentes horarios, con intervalos de 2 horas durante todo el día. Con lo cual se obtuvo diferentes niveles de iluminación, y precisamente de esa forma se pudo observar la variación de los porcentajes de confianza del detector de mascarillas en las detecciones.

A continuación, se muestra una tabla en la cual se visualiza los porcentajes marcados en cada horario de cada una de las clases del detector de mascarilla.

Tabla 5

Pruebas de detección a diferentes horarios y nivel de iluminación

Hora	Con mascarilla (% de confianza)	Mascarilla usada incorrectamente (% de confianza)	Sin mascarilla (% de confianza)
8:00(con iluminación)	91%%	45%	92%
10:00	92%	56%	95%
12:00	99%	72%	98%
14:00	99%	70%	99%
16:00	96%	63%	96%
18:00(con iluminación)	95%	55%	93%
20:00(con iluminación)	92%	52%	92%

Como se puede observar en la tabla anterior, los porcentajes de confianza de las detecciones se encuentran estrechamente relacionados con el nivel de luz.

Tomando en cuenta que desde la mañana empieza un aumento progresivo de luz, se realizó las pruebas cada 2 horas y se realizó varias detecciones, con lo cual el valor presentado en las tablas es un valor promedio aproximado de todas las detecciones realizadas. En el caso de los horarios de 8:00, 18:00 y 20:00 horas las pruebas se realizaron con iluminación artificial.

Pruebas de detección respecto a la distancia

La distancia del objetivo de detección frente a la cámara es otro factor muy importante dentro de los modelos de detección, por lo cual las siguientes pruebas se enfocaron en determinar una distancia ideal para dicho objetivo.

Tabla 6

Pruebas de detección respecto a la distancia

Distancia [m]	Con mascarilla (% de confianza)	Mascarilla usada incorrectamente (% de confianza)	Sin mascarilla (% de confianza)
0.2	80%	27%	80%
0.4	89%	45%	88%
0.6	97%	68%	97%
0.8	99%	75%	99%
1	98%	65%	98%
1.2	95%	55%	94%
1.4	92%	42%	93%
1.6	88%	36%	88%
1.8	85%	No hay detección	84%
2	85%	No hay detección	83%

Los resultados obtenidos en la tabla anterior fueron realizados en condiciones óptimas de iluminación. Como se puede observar la distancia también juega un papel importante dentro los porcentajes de detección de las mascarillas.

Como se observa cuando las detecciones son demasiado cercanas, como ocurre a 0.2 y 0.4 m se tienen resultados de detección poco favorables, y esto es debido a que al estar muy cerca de la cámara se realizan detecciones múltiples, además de que no se logra enfocar en su totalidad el rostro, por lo cual los porcentajes de confianza de las detecciones bajen.

Por otro lado, se pudo ver que en las distancias entre 0.6 y 0.8 m se obtienen los mejores resultados de detección, debido a que el enfoque del rostro se realiza completamente y es una distancia prudente para que el detector pueda detectar las mascarillas.

Finalmente, se observó que a partir de 1 metro de distancia los porcentajes de confianza de las detecciones van bajando paulatinamente, sin embargo, hasta 1.6 metros no existe una baja significativa. Pero en los rangos de 1.8 a 2 metros en la clase de Mascarilla usada incorrectamente no tiene una detección adecuada. Y esto se justificó debido a que a distancias mayores el objetivo de detección (mascarilla) al ser pequeño, no se lo puede detectar con precisión.

En base a los resultados anteriores se determinó la distancia a la que debe estar el objetivo de detección de la cámara en 0.70 metros, valor que se fijó para las posteriores pruebas sobre los escenarios definidos

Contador de personas

En el caso del contador de personas el principal factor determinante en cuanto a su funcionamiento es la distancia de detección de las personas, pues debido a que su ubicación va a estar desde arriba con un enfoque de vista superior, esta distancia con respecto a los objetivos de detección (personas), fue probada y en la siguiente tabla se indica los resultados obtenidos. El factor de iluminación, si bien es cierto que también tiene un impacto mínimo, no es motivo de análisis, debido a que la posición de la cámara desde una vista superior no supone problemas

Tabla 7

Pruebas de funcionamiento del contador

	Ingreso		Salida	
Distancia [m]	Detección correcta y conteo	Sin detección	Detección correcta y conteo	Sin detección
2	6	4	8	2
2.25	7	3	8	2
2.5	9	1	10	0
2.75	10	0	10	0
3	10	0	10	0
3.25	10	0	10	0
3.50	10	0	10	0
3.75	10	0	10	0
4	8	2	9	1

Como se observa en la tabla anterior, el principal problema sobre las detecciones y el conteo se da cuando la cámara no tiene una altura apropiada para la detección, y esto es debido a que al trabajar a alturas menores a 2.25 m la detección cubre cas todo el marco de detección de la cámara, por lo cual la tarea de conteo se vuelve difícil de realizar.

A partir de los 2.75 hasta 3.75 metros de altura se observa que la detección, el seguimiento y el conteo se realiza correctamente debido a que el enfoque de la cámara al estar a la altura correspondiente toma al objetivo de detección de manera correcta y puede realizar el seguimiento y el discernimiento de si la persona está ingresando o saliendo.

A la altura de 4 metros se puede observar que también existen falta de detecciones, debido a que a una altura muy mayor los objetivos de detección empiezan a perder visibilidad para el modelo de detección de personas. En base a los resultados obtenidos con las pruebas anteriores se definió que, para las posteriores pruebas sobre los escenarios de prueba, la altura optima debería ser de 3 metros de altura dentro de lo posible y en concordancia con los espacios físicos de los escenarios.

Pruebas en los escenarios propuestos

Para la realización de las pruebas es necesario tomar en cuenta el funcionamiento integrado del modelo de detección de mascarillas, como del contador de personas , las alertas audibles y la presentación de los valores asociados al contador en el servidor externo.

Es por ello que para las pruebas de funcionamiento se debe tomar en consideración lo siguiente:

La instalación de la cámara para el conteo de personas se encuentra ubicada justamente en el limite del ingreso, con lo cual es la primera funcionalidad que realiza el prototipo, y en base a ello se reproducirán las alarmas asociadas al contador de personas.

El prototipo con la cámara para la detección de personas se encuentra ubicado dentro del establecimiento de prueba, con lo cual se encontrará a una distancia prudente de la cámara del conteo de personas, esto permite que la función de detección de mascarillas se realice posterior al registro de ingreso de la persona. Este funcionamiento se encuentra justificado debido a que el primer objetivo es controlar el ingreso de personas en base al aforo permitido, y una vez que este parámetro se cumpla, verificar el uso de la mascarilla, pues no tendría sentido que si el aforo está lleno y las personas ya no puedan ingresar se realice la detección de mascarilla, y con lo cual también se asegura sincronía entre el trabajo de ambas funcionalidades.

Escenario 1: Gimnasio "TEAM ÉLITE"

El primer escenario de prueba es un gimnasio ubicado al sur de la Ciudad de Quito, el cual brinda servicio de gimnasio y entrenamiento formativo en Kick Boxing, dicho establecimiento es un espacio cerrado por lo que la aplicación del prototipo es viable.

Figura 38

Escenario 1: Gimnasio "TEAM ÉLITE"

Las pruebas en este escenario fueron realizadas con los siguientes parámetros.

Horario

Las pruebas de funcionamiento del prototipo se realizaron entre las 10:00 y 12:00 horas.

Nivel de luminosidad

Debido a la situación climática y el horario en el que se probó el prototipo, se tiene baja iluminancia pues la prueba se realizó en un típico día nublado y además el acceso del establecimiento no cuenta con iluminación adicional.

Distancia de detección de las cámaras

Para la detección de mascarillas la cámara se colocó a una altura de 1.60 metros, con lo cual se obtiene un rango amplio para detectar personas de diferente estatura, además se tiene un límite establecido para la posición de la persona que va a realizar la prueba, en este caso la distancia optima con la cámara es a 0.70 metros de distancia, con lo cual se asegura una mejor detección.

En el caso del conteo de personas la cámara se ubicó a 2.50 metros de altura debido a la limitante del espacio físico del local, pues los resultados óptimos de detección se obtienen a una distancia de 3 metros.

Con ello se procedió a poner en funcionamiento el prototipo.

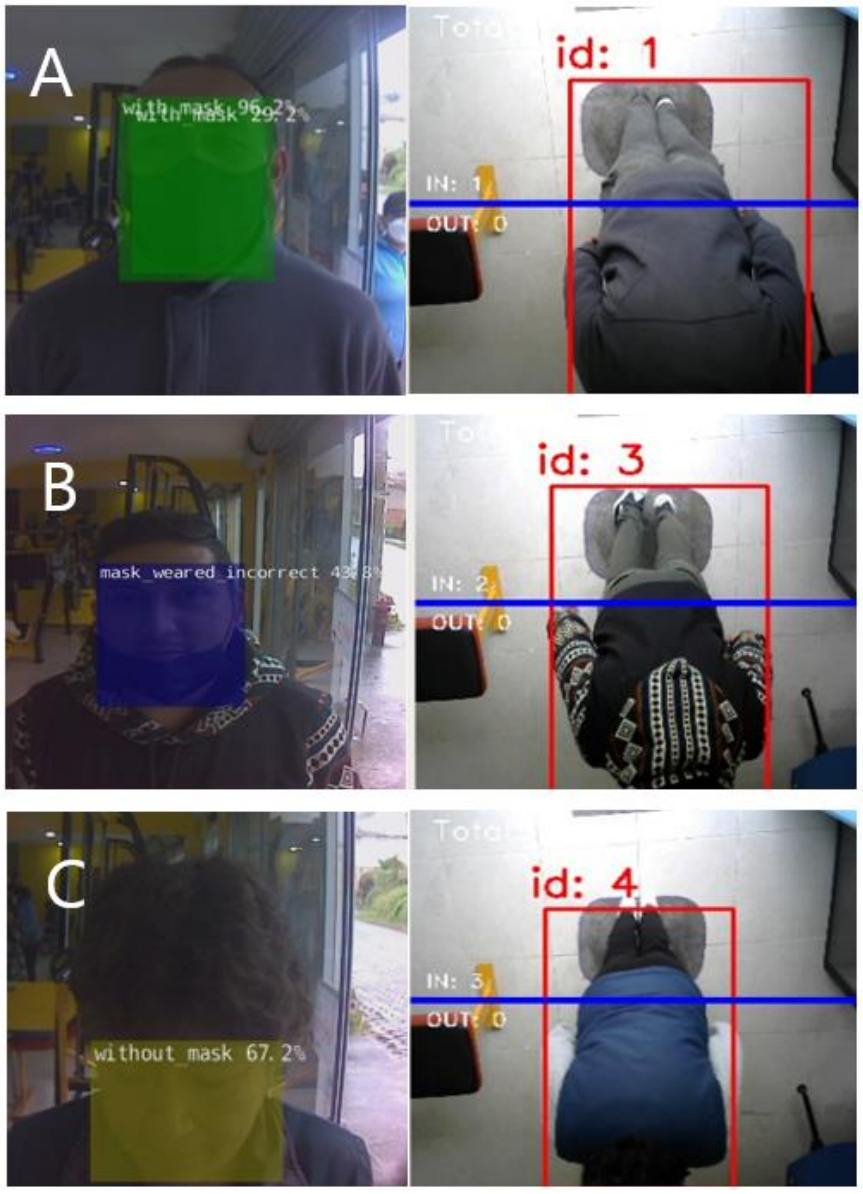
Figura 39

Instalación del prototipo

En la imagen anterior se puede observar cómo se realiza la detección de la mascarilla y a la vez con la cámara superior se realiza el conteo de personas.

Figura 40

Detecciones de mascarilla y seguimiento de objetivos para el contador A) Con mascarilla, B) Mascarilla en posición incorrecta, C)Sin mascarilla



En la siguiente tabla se muestra todas las detecciones que se realizaron durante el periodo de prueba del prototipo en este escenario

Tabla 8

Pruebas de funcionamiento del prototipo en el escenario 1

USUARIO	DETECCIÓN DE MASCARILLA	CONTEO DE PERSONAS	ALARMAS
1	Con mascarilla (96.2 %)	Ingreso (1)	Mascarilla detectada, Bienvenido
2	Sin mascarilla (93.3 %)	Ingreso (2)	Póngase la mascarilla
3	Con mascarilla (82.1 %)	Ingreso (3)	Mascarilla detectada, Bienvenido
4	Con mascarilla (91.1 %)	Ingreso (4)	Mascarilla detectada, Bienvenido
5	Con mascarilla (85.8 %)	Ingreso (5)	Mascarilla detectada, Bienvenido
6	Con mascarilla (93.3 %)	Ingreso (6)	Mascarilla detectada, Bienvenido Aforo al 50 %
7	No hay detección	Salida (5)	
8	No hay detección	Salida (4)	
9	Con mascarilla (88.1 %)	Ingreso (5)	Mascarilla detectada, Bienvenido
10	Masc. en posición incorrecta (43.8 %)	Ingreso (6)	Coloque su mascarilla correctamente
11	Sin mascarilla (94.7 %)	Ingreso (7)	Póngase la mascarilla
12	Con mascarilla (94.8 %)	Ingreso (8)	Mascarilla detectada, Bienvenido
13	Con mascarilla (92.1 %)	Ingreso (9)	Mascarilla detectada, Bienvenido Aforo al 75%
14	Con mascarilla (88.0 %)	Ingreso (10)	Mascarilla detectada, Bienvenido
15	No hay detección	Salida (9)	
16	Con mascarilla (93.3 %)	Ingreso (10)	Mascarilla detectada, Bienvenido
17	Con mascarilla (87 %)	Ingreso (11)	Mascarilla detectada, Bienvenido
18	Con mascarilla (91.9 %)	Ingreso (12)	Mascarilla detectada, Bienvenido Aforo al 100 %
19	Masc. en posición incorrecta (56.5 %)	Ingreso (13)	Aforo al 100% ,No ingrese

Figura 41

Presentación de valores en el servidor

Resultados

En esta prueba de funcionamiento se tuvieron que configurar los parámetros de detección de la mascarilla debido al bajo porcentaje de luminosidad, efecto que se presentó debido a la poca iluminación en el acceso del gimnasio, como de la situación climática al momento de la realización de la prueba.

Debido a esto se configuró las validaciones de los porcentajes de confianza para las detecciones en el caso de las clases Con Mascarilla y Sin Mascarilla en el 80%, mientras que en la clase Mascarilla usada Incorrectamente en 40%. Con ello se obtuvo un funcionamiento correcto de las detecciones, ya que como se realizó en las pruebas en diferentes entornos de iluminación, en lugares con poca iluminación los porcentajes de confianza disminuyen en las detecciones.

De esta manera en cuanto al detector de mascarilla, durante todo su funcionamiento ejecuto las detecciones sin errores de confusión entre clases.

En el caso del contador de personas el rendimiento se vio un tanto afectado debido a la altura en que se posicionó la cámara que realiza esta acción, teniendo una detección errónea de un usuario que ingresaba al establecimiento durante toda la prueba de funcionamiento del prototipo

Escenario 2: Restaurante "POLLO MAS RICO"

El segundo escenario de prueba es un Restaurante Asadero de Pollos denominado "POLLO MAS RICO", se encuentra ubicado en el sur de Quito, sector el Conde. Este escenario presenta las condiciones previstas para la instalación del prototipo en el acceso para la detección de mascarillas y también con una altura de 3 metros para la ubicación de la cámara del conteo de personas.

Figura 42

Escenario 2: Restaurante "POLLO MAS RICO"

Las pruebas en este escenario fueron realizadas con los siguientes parámetros.

Horario

Las pruebas de funcionamiento del prototipo se realizaron entre las 13:00 y 15:00 horas.

Nivel de luminosidad

Debido a la situación climática y el horario en el que se probó el prototipo, se tiene una luminosidad óptima para la realización de las pruebas.

Distancia de detección de las cámaras

Para la detección de mascarillas la cámara se colocó a una altura de 1.60 metros, con lo cual se obtiene un rango amplio para detectar personas de diferente estatura, además se tiene un límite establecido para la posición de la persona que va a realizar la prueba, en este caso la distancia optima con la cámara es a 0.70 metros de distancia, con lo cual se asegura una mejor detección.

En el caso del conteo de personas la cámara se ubicó a 3 metros de altura.

Con ello se procedió a poner en funcionamiento el prototipo.

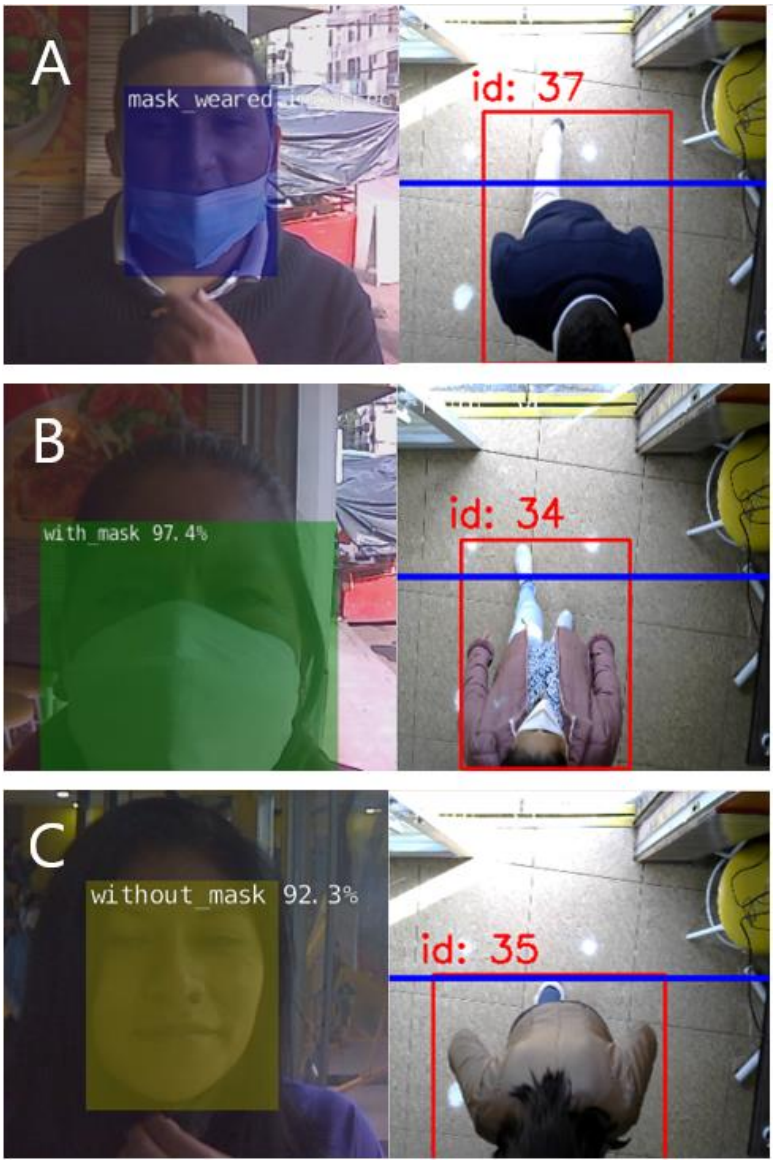
Figura 43

Instalación del prototipo

En la imagen anterior se puede observar cómo se realiza la detección de la mascarilla y a la vez con la cámara superior se realiza el conteo de personas.

Figura 44

Detecciones de mascarilla y seguimiento de objetivos para el contador A) Mascarilla en posición incorrecta, B) Con mascarilla, C)Sin mascarilla



En la siguiente tabla se muestra todas las detecciones que se realizaron durante el periodo de prueba del prototipo en este escenario

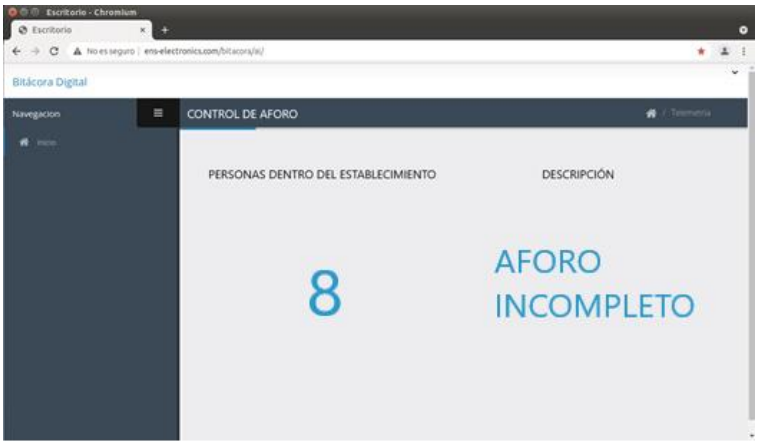
Tabla 9

Pruebas de funcionamiento del prototipo en el escenario 2

USUARIO	DETECCIÓN DE MASCARILLA	CONTEO DE PERSONAS	ALARMAS
1	Con mascarilla (97.4 %)	Ingreso (1)	Mascarilla detectada, Bienvenido
2	Con mascarilla (92.1 %)	Ingreso (2)	Mascarilla detectada, Bienvenido
3	Sin mascarilla (96.3 %)	Ingreso (3)	Póngase la mascarilla
4	Masc. en posición incorrecta (76.6 %)	Ingreso (4)	Coloque su mascarilla correctamente
5	No hay detección	Salida (3)	
6	No hay detección	Salida (2)	
7	Con mascarilla (95.5 %)	Ingreso (3)	Mascarilla detectada, Bienvenido
8	Sin mascarilla (97.3 %)	Ingreso (4)	Póngase la mascarilla
9	No hay detección	Salida (3)	
10	Con mascarilla (99.1 %)	Ingreso (4)	Mascarilla detectada, Bienvenido
11	Sin mascarilla (99.3 %)	Ingreso (5)	Póngase la mascarilla
12	Con mascarilla (97.8 %)	Ingreso (6)	Mascarilla detectada, Bienvenido Aforo al 50 %
13	Con mascarilla (98.1 %)	Ingreso (7)	Mascarilla detectada, Bienvenido
14	Con mascarilla (98.0 %)	Ingreso (8)	Mascarilla detectada, Bienvenido
15	Con mascarilla (98.1 %)	Ingreso (9)	Mascarilla detectada, Bienvenido Aforo al 75%
16	No hay detección	Salida (8)	
17	No hay detección	Salida (7)	
18	No hay detección	Salida (6)	Aforo al 50 %
19	Con mascarilla (95.8 %)	Ingreso (7)	Mascarilla detectada, Bienvenido

Figura 45

Presentación de valores en el servidor



Resultados

En esta prueba de funcionamiento se mantuvieron los rangos de validación de las detecciones óptimos que se detallaron en las pruebas sobre diferentes entornos de iluminación, esto debido a que el escenario presentó una buena iluminación y además la situación climática también fue favorable. Por lo que como se observa en la tabla anterior, se obtuvieron detecciones con valores de confianza muy altos, cercanos al 100 % en el caso de las clases Con mascarilla y Sin mascarilla.

En el caso de la clase Mascarilla usada incorrectamente se vio un rendimiento muy mejorado, ya se alcanzó porcentajes superiores al 70% de confianza en la detección, con lo cual el rendimiento de la detección de mascarillas fue optimo. En cuanto al contador de personas, de igual manera se tuvo un rendimiento optimo, pues se obtuvo todas las detecciones de entrada y salida correctamente, es decir nunca se perdió ningún objetivo de detección.

Escenario 3: Centro Fisioterapéutico "FISIO ON"

El tercer escenario es un centro de Fisioterapia ubicado en el sur de la ciudad, el cual ofrece servicios de fisioterapia y rehabilitación. Este escenario de igual manera cumple con los requisitos necesarios para la instalación del prototipo, con lo cual se procedió a realizar las pruebas correspondientes. Para este escenario se maneja un aforo de 8 personas.

Figura 46

Escenario 3: Centro Fisioterapéutico "FISIO ON"

Las pruebas en este escenario fueron realizadas con los siguientes parámetros.

Horario

Las pruebas de funcionamiento del prototipo se realizaron entre las 17:00 a 19:00 horas

Nivel de luminosidad

Debido a la situación climática y el horario en el que se probó el prototipo, se tiene se tiene una luminosidad aceptable debido a la iluminación artificial en el acceso de dicha entidad.

Distancia de detección de las cámaras

Para la detección de mascarillas la cámara se colocó a una altura de 1.60 metros, con lo cual se obtiene un rango amplio para detectar personas de diferente estatura, además se tiene un límite establecido para la posición de la persona que va a realizar la prueba, en este caso la distancia optima con la cámara es a 0.70 metros de distancia, con lo cual se asegura una mejor detección.

En el caso del conteo de personas la cámara se ubicó a 2.80 metros de altura debido a la limitante del espacio físico del local, pues los resultados óptimos de detección se obtienen a una distancia de 3 metros.

Con ello se procedió a poner en funcionamiento el prototipo.

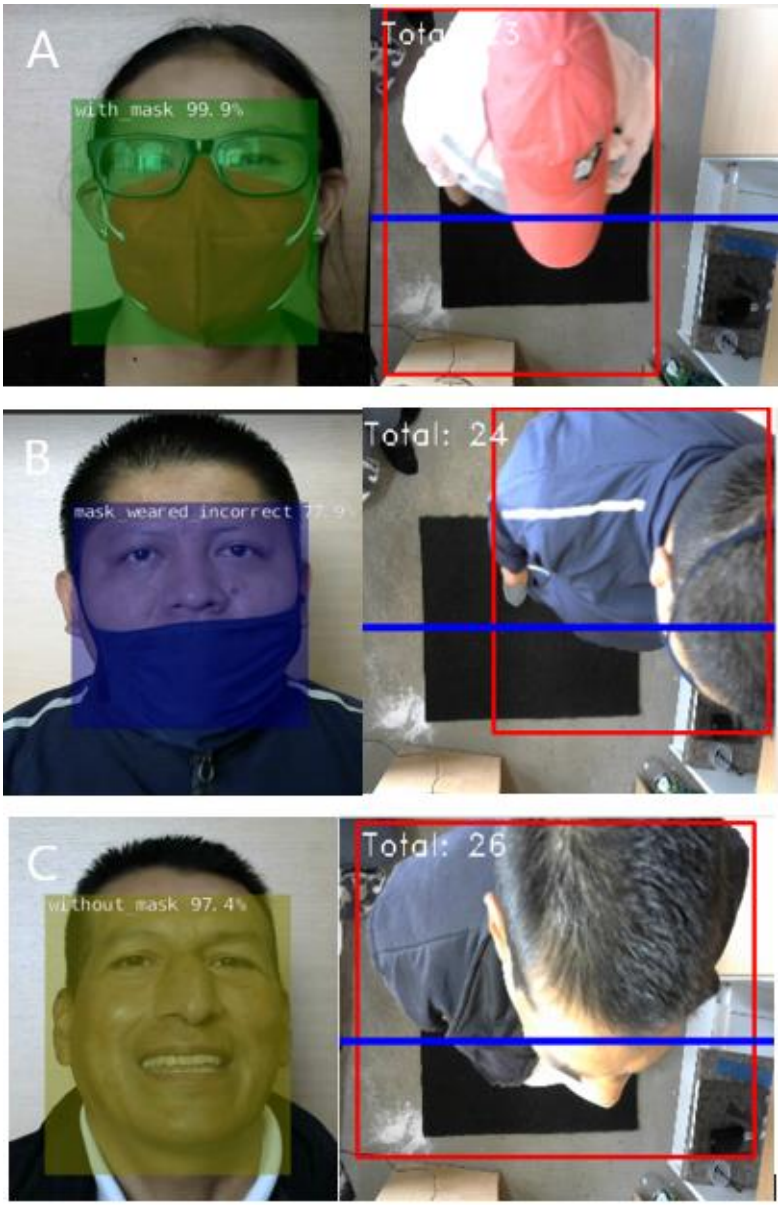
Figura 47

Instalación del prototipo

En la imagen anterior se puede observar cómo se realiza la detección de la mascarilla y a la vez con la cámara superior se realiza el conteo de personas.

Figura 48

Detecciones de mascarilla y seguimiento de objetivos para el contador A) Con mascarilla, B) Mascarilla en posición incorrecta, C)Sin mascarilla



En la siguiente tabla se muestra todas las detecciones que se realizaron durante el periodo de prueba del prototipo en este escenario

Tabla 10

Pruebas de funcionamiento del prototipo en el escenario 3

USUARIO	DETECCIÓN DE MASCARILLA	CONTEO DE PERSONAS	ALARMAS
1	Con mascarilla (94.2 %)	Ingreso (1)	Mascarilla detectada, Bienvenido
2	Sin mascarilla (91.3 %)	Ingreso (2)	Póngase la mascarilla
3	Con mascarilla (92.1 %)	Ingreso (3)	Mascarilla detectada, Bienvenido
4	Masc. en posición incorrecta (63.8 %)	Ingreso (4)	Coloque su mascarilla correctamente
			Aforo al 50 %
5	Con mascarilla (95.8 %)	Ingreso (5)	Mascarilla detectada, Bienvenido
6	Con mascarilla (93.3 %)	Ingreso (6)	Mascarilla detectada, Bienvenido
			Aforo al 75 %
7	No hay detección	Salida (5)	
8	No hay detección	Salida (4)	Aforo al 50 %
9	Con mascarilla (96.1 %)	Ingreso (5)	Mascarilla detectada, Bienvenido
10	Masc. en posición incorrecta (53.8 %)	Ingreso (6)	Coloque su mascarilla correctamente
			Aforo al 75 %
11	Con mascarilla (93.0 %)	Ingreso (7)	Mascarilla detectada, Bienvenido
12	Con mascarilla (94.8 %)	Ingreso (8)	Mascarilla detectada, Bienvenido
13	Con mascarilla (96.8 %)	Ingreso(9)(8)	Aforo al 100% no ingrese
14	No hay detección	Salida (7)	
15	No hay detección	Salida (6)	Aforo al 75%

Figura 49

Presentación de valores en el servidor

Resultados

En esta prueba de funcionamiento se trabajó con los niveles óptimos de la detección de mascarilla obtenidos en la prueba anteriores. Se trabajo con dichos niveles debido a que el nivel de iluminación artificial favoreció al rendimiento del modelo de detección. En el caso del contador de personas, se trabajó con la posición de la cámara a una altura de 2.80 metros teniendo en cuenta el espacio físico del local. Se trabajo con de los porcentajes de confianza para las detecciones de las clases Con Mascarilla y Sin Mascarilla en el 90%, mientras que en la clase Mascarilla usada Incorrectamente en 60%. Con ello se obtuvo un funcionamiento correcto de las detecciones.

Como se observa en la tabla el rendimiento del prototipo fue optimo, pues en cuanto a los porcentajes de detección superar los porcentajes establecidos de validación perfectamente, por lo que no existen falsas detecciones o confusión entre una clase y otra. Con lo cual se aseguró que el control de acceso sea confiable. En el caso del contador de personas, a pesar de no instalar la cámara a la altura establecida como optima, su rendimiento fue perfecto, con lo cual no se presentaron perdida de los objetivos de detección, y el conteo de personas se realizó de manera correcta.

Capítulo V: Conclusiones, recomendaciones y trabajos futuros

Conclusiones

- El modelo de red SSD-MobilenetV1 que de manera específica se encuentra entrenado para detección facial se reentreno para implementar la aplicación de detección de mascarillas, mismo que dio resultados satisfactorios en las pruebas del prototipo, concluyendo que con esta red se puede obtener un buen rendimiento en la velocidad de procesamiento de las imágenes, tomando en cuenta las limitaciones computacionales de la tarjeta de desarrollo Nvidia Jetson Nano.

- En el caso del contador de personas, el uso de la red SSD MobilenetV1-COCO entrenada para la función de detección de personas, integrada con el algoritmo de seguimiento SORT, es una solución aceptable, aunque existen otros algoritmos más complejos con los cuales se puede mitigar el problema de oclusión y reingreso de objetivos, sin embargo estos algoritmos también requieren más potencia de procesamiento que la que dispone el hardware utilizado.

- En la detección de mascarillas y personas, se determinó que el nivel de iluminación del ambiente juega un papel preponderante, por lo que es necesario acomodar tanto la posición de la cámara para evitar la contraluz, como también la distancia de la persona con respecto a la misma.

- El uso de hardware liviano limita de cierto modo el desarrollo de aplicaciones basadas en modelos de redes neuronales implementadas para la detección de objetos en videos capturados en tiempo real. Sin embargo, la tarjeta NVIDIA Jetson Nano tuvo un desempeño

muy aceptable en las pruebas realizadas con este prototipo en distintos escenarios controlados.

- El desarrollo de un prototipo implica varias pruebas de funcionamiento en entornos controlados cuyos factores internos y factores externos podrían afectar el desempeño del mismo, tal como sucedió con la posición de las cámaras, las alturas a las que fueron ubicadas, la distancia de los objetivos de detección, el nivel de iluminación o incluso las condiciones climáticas.

- La implementación de los mensajes audibles del prototipo requiere de un código de programación en paralelo que se sincronice solo con determinados fotogramas del video, pues de lo contario se activarán con todos los fotogramas procesados en el algoritmo de detección.

Recomendaciones

- En el entrenamiento de modelos de detección es importante configurar sus parámetros en base a la capacidad computacional del dispositivo, por lo que se debe tener cuidado con el número de iteraciones de entrenamiento así como también con el lote de imágenes procesadas, pues el tiempo prolongado de trabajo eleva la temperatura de la Tarjeta Jetson Nano y se pudo verificar que el incremento de la misma puede ocasionar errores que en ocasiones impiden concluir con el proceso de entrenamiento.

- Para la inferencia de los modelos de redes neuronales sobre la tarjeta Jetson Nano se requiere realizar el cambio en el código implementado con Tensorflow a Tensor RT con lo cual es posible correr una variedad de modelos sobre dicha tarjeta.

Trabajos Futuros

- La aplicación de modelos basados en Inteligencia Artificial cada vez va en aumento, sin embargo el enfoque en los cuales de desarrolla, pocas veces se orientan en solucionar problemas sociales. Es por ello que un tema importante en donde tiene un gran campo de acción la IA es en la seguridad, aspecto en el cual nuestro país tiene grandes deficiencias. Debido a esto, como trabajo futuro se propone utilizar la misma concepción del presente proyecto, orientado en la detección de armas dentro de establecimiento cerrados, con lo cual se puedan generar alertas, cuando este tipo de situaciones pongan en peligro a los usuarios de un establecimiento. Dichas alertas pueden ser enviadas a un servidor externo en el momento de la detección de armas u objetos peligrosos, con el fin de que los agentes de seguridad ya sean públicos o privados puedan tomar una acción inmediata ante estos hechos.

- El presente proyecto también puede verse reorientado hacia su aplicación sobre el transporte público, pues a día de hoy el uso de mascarilla y el control de aforo dentro de las unidades de transporte es uno de los principales factores de supervisión en la actual pandemia. Sin embargo, esta es una tarea que resulta imposible de ser realizada en todas las unidades por lo que un prototipo autónomo que realice esa función, es una excelente opción para poder controlar estos requerimientos.

- Este prototipo puede ser potenciado a través de una mejor tarjeta de procesamiento como la NVIDIA Jetson TX, con lo cual se podrían correr modelos más potentes y precisos, y a la vez buscar un enfoque para las detecciones que permita que se utilice solo una cámara de video para realizar las tareas de detección de mascarillas y conteo de personas.

Referencias Bibliográficas

Amat, J. (Mayo de 2021). *Ciencia de Datos.* Obtenido de https://www.cienciadedatos.net/documentos/py35-redes-neuronales-python.html

Andrade, H., Sinche, S., & Hidalgo, P. (2021). Modelo para detectar el uso correcto de mascarillas en tiempo real. *Revista de Investigación en Tecnologías de la Información, 9*(17), 111.

Bewley, A., & Ge, Z. (2017). *SIMPLE ONLINE AND REALTIME TRACKING.* Queensland, Sydney.

Borja, A., & Tofael, A. (2021). *Real Time Pear Fruit Detection and Counting Using YOLOv4.*

Chauca , R. (2020). La covid-19 en Ecuador: fragilidad política. *Scielo, 28*(2), 587.

Diaz, F., & Toro, A. (2020). SARS-CoV-2/COVID-19: el virus, la enfermedad y la pandemia. *24*(3), 21.

García , O. (Septiembre de 2019). *Xeridia.* Obtenido de https://www.xeridia.com/blog/redes-neuronales-artificiales-que-son-y-como-se-entrenan-parte-i

GitHub. (2019). *GitHub*. Obtenido de https://github.com/abewley/sort

Kaggle. (2020). *Kaggle.* Obtenido de https://www.kaggle.com/andrewmvd/face-mask-detection/metadata

Mayo Clinic. (2022). *Mayo Clinic*. Obtenido de https://www.mayoclinic.org/es-es/diseases-conditions/coronavirus/symptoms-causes/syc-20479963

Montesdeoca, E. (2020). *IMPLEMENTACIÓN DE UN SISTEMA DE RECONOCIMIENTO DEL USO.* Machala.

Mosquera, C., & Dussan, G. (2020). *Detección y seguimiento de múltiples objetos en tiempo real para vehículos autonomos.* Santiago de Cali.

Nvidia. (2022). *Nvidia Developer.* Obtenido de https://developer.nvidia.com/embedded/learn/get-started-jetson-nano-devkit

Nvidia. (2022). *Nvidia Developer*. Obtenido de https://developer.nvidia.com/tensorrt

Organización Panamericana de la Salud. (2020). *Organización Panamericana de la Salud.*

Sedano, F., Rojas, C., & Vela, J. (2020). COVID-19 DESDE LA PERSPECTIVA DE LA PREVENCIÓN PRIMARIA. *Scielo*, 494.ViewNext. (19 de Agosto de 2019). *ViewNext.* Obtenido de https://www.viewnext.com/transfer-learning-y-redes-convolucionales/page/2/

Anexos