PLATAFORMA PARA LA IMPLEMENTACIÓN DE ALGORITMOS DE VISIÓN POR COMPUTADORA: CASO DE ESTUDIO ALWAYSAI

PLATFORM FOR THE IMPLEMENTATION OF COMPUTER VISION ALGORITHMS: ALWAYSAI CASE STUDY

Luis Ángel Mejía Genis
Universidad Tecnológica Emiliano Zapata del Estado de Morelos, México
mejiagen.luisangel@gmail.com

Jonathan Villanueva Tavira
Universidad Tecnológica Emiliano Zapata del Estado de Morelos, México
jonathanvillanueva@utez.edu.mx

Recepción: 30/octubre/2020 **Aceptación:** 10/diciembre/2020

Resumen

La visión por computadora es una rama de la inteligencia artificial que se ha desarrollado significativamente en los últimos años y se van incorporando cada vez más en nuestra vida cotidiana. Es una de las herramientas clave utilizadas en el desarrollo de proyectos actuales, sobre todo en los robots autónomos. Sin embargo, dotar de visión por computadora a estos proyectos es una tarea hasta cierto punto complicada. Este artículo se concentra en documentar sobre alwaysAI, una plataforma surgida recientemente, la cual tiene como objetivo facilitar a los desarrolladores la implementación de visión por computadora en proyectos, proporcionando diversos algoritmos de visión que hacen uso de modelos previamente entrenados de acuerdo a su aplicación. Los algoritmos proporcionados por esta plataforma ofrecen dos tipos de análisis de imágenes, uno es por medio de video capturado en tiempo real mediante una webcam y el otro método utiliza imágenes previamente cargadas. Se realizaron pruebas piloto de ambos tipos de algoritmos en una computadora con sistema operativo Windows y se documentó sobre sus resultados exitosos.

Palabras Clave: Algoritmos, aprendizaje automático, aprendizaje profundo, modelo, visón por computadora.

Abstract

Computer vision is a branch of artificial intelligence that has developed significantly in recent years and is increasingly incorporated into our daily lives. It is one of the key tools used in the development of current projects, especially in autonomous robots. However, providing computer vision to these projects is a somewhat complicated task. This article concentrates on documenting on a alwaysAI, recently emerged platform, which aims to facilitate developers the implementation of computer vision in projects, providing various vision algorithms that make use of previously trained models according to their application. The algorithms provided by this platform offer two types of image analysis, one is by means of video captured in real time through a webcam and the other method uses previously loaded images. Pilot tests of both types of algorithms were performed on a Windows computer and their successful results were documented.

Keywords**:** *Algorithms, computer vision, deep learning, machine learning, model.*

1. Introducción

Las tecnologías de IA (Inteligencia Artificial) han comenzado a desarrollar como nunca antes la capacidad de ver (visión artificial), oír (reconocimiento de voz) y entender (procesamiento del lenguaje natural). La figura 1 muestra claramente este concepto. Antes, estas habilidades pertenecían únicamente a los seres humanos, pero en el futuro próximo las máquinas y los robots las podrán desarrollar gracias a la IA.

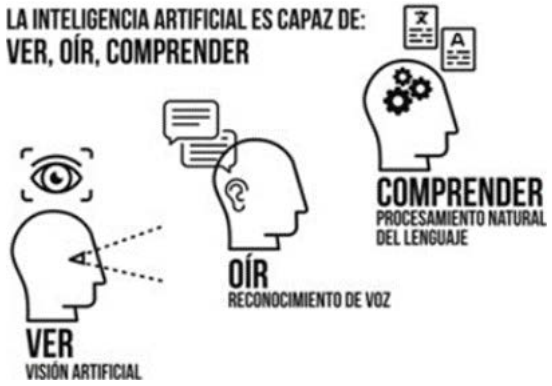


Figura 1 Inteligencia artificial es capaz de ver, oír y comprender [Rouhiainen, 2018].

Uno de los beneficios de la IA es que permite que las máquinas y los robots realicen tareas que los humanos consideran difíciles, aburridas o peligrosas, lo que repercutirá a su vez en que el ser humano pueda realizar aquello que antes creía imposible [Rouhiainen, 2018].

Las dificultades que enfrentan los sistemas que dependen del conocimiento codificado, sugieren que los sistemas de IA necesitan la capacidad de adquirir su propio conocimiento, extrayendo patrones de datos en bruto. Esta capacidad se conoce como aprendizaje automático. La introducción del aprendizaje automático permitió que las computadoras abordaran problemas relacionados conocimiento del mundo real y tomar decisiones que parecen subjetivas [Goodfellow, 2017]. El aprendizaje automático se considerada una buena herramienta para resolver un problema cuando no se cuenta con un modelo un matemático o una ecuación adecuada. Es una técnica que utiliza múltiples datos para crear un programa y realizar una tarea específica [Anderson, 2017].

Una fuente importante de dificultad en muchas aplicaciones de inteligencia artificial del mundo real es que muchos de los factores de variación influyen en cada dato que somos capaz de observar. Los píxeles individuales en una imagen cambian constantemente debido a cambio de iluminación, del ángulo de visión, entre algunos otros factores. Por supuesto, puede ser muy difícil extraer tales características abstractas de alto nivel de datos en bruto. El aprendizaje profundo resuelve este problema central en el aprendizaje automático mediante la introducción de representaciones que se expresan en términos de otras representaciones más simples. El aprendizaje profundo es un particular tipo de aprendizaje automático que logra un gran poder y flexibilidad al aprender a representar el mundo como una jerarquía anidada de conceptos, con cada concepto definido en relación con conceptos más simples y representaciones más abstractas calculadas en términos de los menos abstractos.

La figura 2 muestra cómo un sistema de aprendizaje profundo puede representar el concepto de una imagen de una persona combinando conceptos más simples, como esquinas y contornos, que a su vez se definen en términos de bordes. [Goodfellow, 2017].

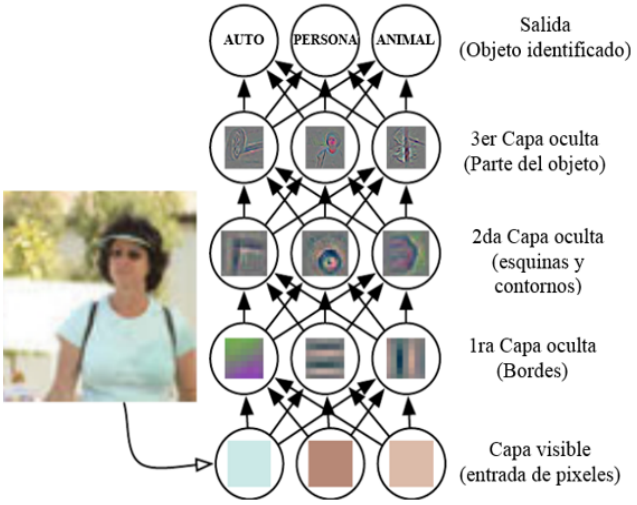


Figura 2 Modelo de aprendizaje profundo [Goodfellow, 2017].

El término visión integrada implica un híbrido de dos tecnologías, sistemas embebidos y visión por computadora. Un sistema embebido es un sistema basado en microprocesador que no es una computadora de uso general, mientras que la visión por computadora se refiere al uso de procesamiento digital y algoritmos inteligentes para interpretar el significado de imágenes o videos [Vijayalakshmi, 2020]. Un algoritmo es una secuencia de instrucciones que deben llevarse a cabo para transformar la entrada a la salida. Para la misma tarea, puede haber varios algoritmos y se busca encontrar el más eficiente, requiriendo el menor número de instrucciones, memoria o ambos [Alpaydin, 2010]. La estructura básica de todo sistema de visión por computador desde el punto de vista hardware, es el sistema de adquisición de imágenes, como se muestra en la figura 3. Es el encargado de todo proceso de información y captación de imágenes, es decir, el que trasmite la información de mundo físico hasta la memoria del computador [Somolinos, 2002].

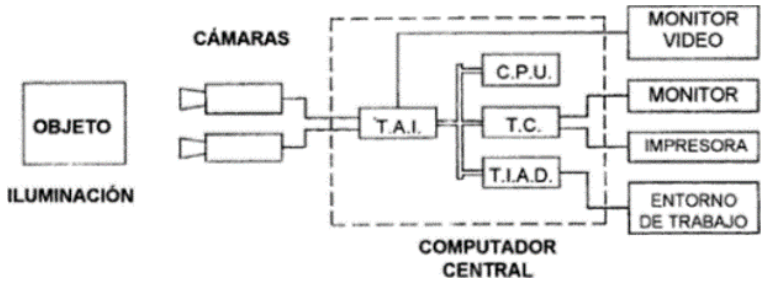


Figura 3 Estructura de un sistema de visión por computadora [Somolinos, 2002].

Entonces, simplemente, la visión integrada se refiere a máquinas que entienden su entorno a través de medios visuales. Un sistema de visión integrado consiste, por ejemplo, en una cámara, un llamado cámara de nivel de placa, que está conectada a una placa de procesamiento como se muestra en la figura 4. Las placas de procesamiento se hacen cargo de las tareas de la PC desde la clásica configuración de visión artificial [Vijayalakshmi, 2020].

Figura 4 Sistema de visión integrado [Vijayalakshmi, 2020].

Sin embargo, la visión integrada es una de las tareas más compiladas y tediosas al desarrollar proyectos. La implementación algoritmos de visión por computadora en sistemas embebidos como Raspberry Pi, muchas veces presentan algunos problemas como falta compatibilidad y fallas en la instalación de librerías, además, son algoritmos muy robustos por lo que se complica la configuración e implementación de los mismos. Por lo anterior, han surgido algunas plataformas que tienen como objetivo facilitar la integración de visión por computadora en algunos microprocesadores o computadoras de placa única, proporcionando algoritmos de visión con modelos previamente entrenados.

AlwaysAI (figura 5) es una plataforma esencial de desarrollo de visión por computadora para crear e implementar aplicaciones de aprendizaje automático en dispositivos de borde. El motor de plataforma, edgeIQ, permite a los desarrolladores crear aplicaciones de borde utilizando API de Python simples para servicios centrales de visión por computadora, como detección de objetos, clasificación, seguimiento, conteo y segmentación semántica.

El catálogo de alwaysAI proporciona modelos de aprendizaje profundo y pre-entrenados que simplemente funcionan en su aplicación. Es muy fácil agregar, eliminar o intercambiar modelos, lo que le permite experimentar y encontrar lo que

funciona mejor para la aplicación. También si se tiene un modelo puede ser subido al catálogo de awaysAI y usarlo en la aplicación. AlwaysAI se normaliza en los principales marcos como Caffe, TensorFlow y Darknet en varios parámetros clave.

Figura 5 Logo de la Plataforma alwaysAI.

2. Métodos

La plataforma alwaysAI permite desarrollar una aplicación de visión por computadora en una computadora local, y luego implementar la aplicación localmente o en un dispositivo periférico con la CLI alwaysAI.

Algoritmos

La plataforma proporciona diferentes algoritmos de visión por computadora los cuales se enlistan a continuación:

- Clasificador de género por edad.
- Seguimiento de objetos.
- Contador de cara.
- Clasificador de imagen.
- Segmentación semántica de vehículos autónomos para nvidia.
- Detector de objetos en tiempo real para nvidia.
- Detector de objetos.
- Detector facial en tiempo real.
- Detector de objetos en tiempo real.
- Estimación de pose en tiempo real.
- Segmentación semántica de paisaje urbano.
- Segmentación semántica.
- Contador de objetos simple.

Modelos

El Catálogo de modelos alwaysAI contiene modelos de aprendizaje automático previamente capacitados, actualmente abarca cuatro categorías: detección de objetos (Tabla 1), clasificación (Tabla 2), estimación de pose (Tabla 3) y segmentación semántica (Tabla 4).

Tabla 1 Modelos de la categoría detección de objetos.

Modelo	Autor	Marco de referencia	Conjunto de datos	AVG tiempo de interferencia
res10_300x300_ssd_iter_140000	alwaysai	caffe	desconocido	132,85 ms
squeezenet_ssd	alwaysai	caffe	VOC0712	141,75 ms
mobilenet_ssd_face_longrange	alwaysai	caffe	WIDER	158.07 ms
mobilenet_ssd_face	alwaysai	caffe	WIDER	175.24 ms
mobilenet_ssd	alwaysai	caffe	Pascal VOC	181.25 ms
2019_FRC_GamePieces_Dev_v3	CAP1Sup	tensorflow	desconocido	184.59 ms
ssd_mobilenet_v1_coco_2018_01_28	alwaysai	tensorflow	COCO	200.31 ms
ssd_mobilenet_v2_oidv4	alwaysai	tensorflow	OIDv4	389 ms
yolo_v3	alwaysai	darknet	COCO	3117.700 ms
ssd_inception_v2_coco_2018_01_28	alwaysai	tensorflow	COCO	1266 ms

Tabla 2 Modelos de la categoría clasificación.

Modelo	Autor	Marco de referencia	Conjunto de datos	AVG tiempo de interferencia
shufflenet_1x_g3	alwaysai	caffe	ILSVRC-2014	38 ms
squeezenet_v1.1	alwaysai	caffe	ILSVRC-2015	53.72 ms
oxford102_alexnet	alwaysai	caffe	Oxford102	78 ms
agenet	alwaysai	caffe	Adience-Benchmark	78.770 ms
finetune_flickr_style	alwaysai	caffe	Flickr-Style	82 ms
alexnet	alwaysai	caffe	ILSVRC-2010	83.14 ms
gendernet	alwaysai	caffe	Adience-Benchmark	87.33 ms
squeezenet_v1.0	alwaysai	caffe	ILSVRC-2015	ILSVRC-2015
mobilenet_v1_1.0_224	alwaysai	tensorflow	ILSVRC-2012	110.17 ms
resnet_18_grocery	alwaysai	onnx	Desconocido	173.4 ms

Tabla 3 Modelos de la categoría estimación de pose.

Modelo	Autor	Marco de referencia	Conjunto de datos	AVG tiempo de interferencia
human_pose_nano	alwaysai	tensor-rt	COCO	Desconocido
human_pose_myriad	alwaysai	caffe	COCO	Desconocido
human_pose_tx2	alwaysai	tensor-rt	COCO	Desconocido
human-pose	alwaysai	onnx	COCO	Desconocido

Tabla 4 Modelos de la categoría segmentación semántica.

Modelo	Autor	Marco de referencia	Conjunto de datos	AVG tiempo de interferencia
fcn_resnet18_cityscapes_512x256	alwaysai	onnx	Cityscapes	1238 ms
fcn_resnet18_pascal_voc_512x320	alwaysai	onnx	Pascal VOC	1495 ms
fcn_alexnet_cityscapes_sd	alwaysai	caffe	Cityscapes	5796 ms
enet	alwaysai	enet	Desconocido	Desconocido
fcn_alexnet_pascal_voc	alwaysai	caffe	Desconocido	Desconocido

Nota: De las categorías detección de objetos y clasificación solo se enlistan los modelos con menor tiempo de interferencia, sin embargo, existen más modelos que se pueden implementar en los algoritmos y se encuentran en la plataforma, en la cual también se puede observar a detalle las características completas de todos los modelos.

Para elegir el modelo adecuado para implementar en las aplicaciones de visión, se debe tener en cuenta diferentes criterios, el principal son las etiquetas que contiene el modelo, las etiquetas son todos los objetos con los que fue entrenado el modelo y que es capaz de detectar y clasificar, por otro lado, se debe considerar la precisión del modelo y los tiempos de interferencia; la información de precisión de un modelo se muestra en Media Precisión Media (mAP). Se dan dos valores de mAP según la frecuencia con la que se predice correctamente un objeto dentro de las primeras predicciones (top-1) o dentro de las cinco predicciones principales (top-5) devueltas por el motor de inferencia. Los tiempos de inferencia para los modelos miden cuánto tiempo tarda el motor de inferencia en procesar una imagen y devolver predicciones. Los tiempos de inferencia se dan en segundos. En el catálogo de modelos de la plataforma se visualiza a detalle las especificaciones de cada modelo para elegir el adecuado de acuerdo con lo que se menciona anteriormente.

Biblioteca Python edgeIQ

La API edgeIQ es una biblioteca utilizada en los algoritmos de la plataforma la cual sirve como una capa intermedia entre los modelos de aprendizaje automático y las aplicaciones de visión por computadora, y proporciona todo lo necesario para construir una aplicación. La documentación completa sobre esta biblioteca esta publicada directamente en la página web de alwaysIA.

Implementación de los algoritmos de alwaysAI en un sistema embebido

AlwaysAI es compatible con los sistemas basados en ARM32, ARM64 y x86 que ejecutan Docker en Linux. Esto incluye Raspberry Pi 3B +, NVIDIA Jetson, ASUS Tinker Board y sistemas basados en los procesadores Snapdragon de Qualcomm. Además, alwaysAI es compatible con los aceleradores Myriad de Intel en todas las plataformas anteriores y el acelerador Jetson Nano en el kit para desarrolladores NVIDIA Jetson Nano.

Para cumplir con los requisitos mínimos, el dispositivo perimetral debe tener instalado un sistema operativo basado en Linux para ser compatible con Docker. Se ha probado los siguientes sistemas operativos con la imagen edgeIQ Docker:

- Raspbian Buster
- Raspbian Stretch
- Debian Buster
- Debian Stretch

El dispositivo también debe tener configurado el acceso SSH (se requiere SSH para que la CLI de alwaysAI se comunique con el dispositivo) y debe ser compatible con una cámara USB o una cámara de cinta (para aplicaciones que utilizarán una alimentación de video en tiempo real desde una cámara conectada).

Configuración de la computadora de desarrollo

El primer paso para usar la plataforma alwaysAI es configurar la computadora de desarrollo con la interfaz de línea de comandos (CLI) alwaysAI. La plataforma es compatible con los sistemas operativos Windows, Linux y MAC, y en cada caso la

instalación del CLI es diferente. En este proyecto se utilizó Windows como sistema operativo de la computadora de desarrollo.

La plataforma facilita un software de escritorio (Figura 6) el cual se descarga desde el sitio web y se instala en la computadora de desarrollo. Dicho software incluye todo lo que se necesita para crear aplicaciones de visión por computadora alwaysAI en el host de desarrollo.

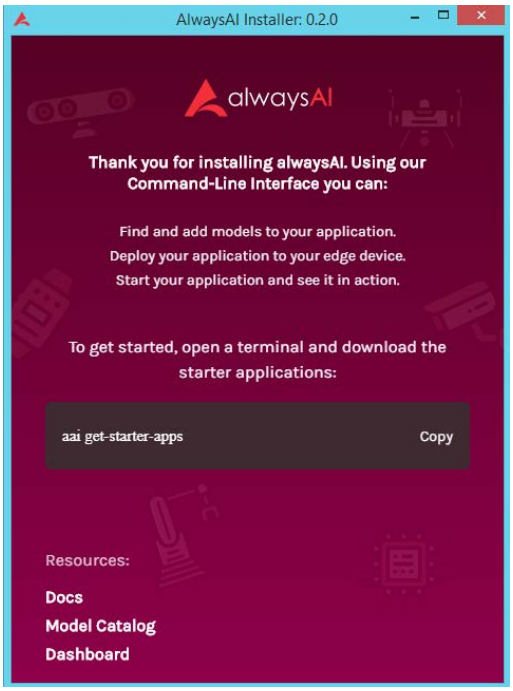


Figura 6 Software de escritorio de alwaysAI.

En la figura 7 se observa una terminal Powershell de Windows en la cual se ejecuta el comando *aai -v*. Una vez ejecutado muestra la versión de alwaysAI instalada y sirve para verificar que la CLI alwaysAI esté completamente instalada y accesible.

Figura 7 Versión de alwaysIA en PowerShell.

Descarga de los archivos necesarios

Para comenzar, se descargan todas las aplicaciones de inicio, las cuales son carpetas (Figura 8) en donde se encuentras scrip's, archivos de texto, imágenes, modelos entre otras cosas requeridas para ejecutar los programas de visión por computadora. La descarga de las carpetas se puede hacer de dos maneras; descargando un archivo ZIP con las carpetas necesarias desde la página web de alwaysIA a la computadora de desarrollo, o como se realizó en este proyecto, descargándolas desde el terminal PowerShell de Windows ejecutando el comando aai get-starter-apps (Figura 9).

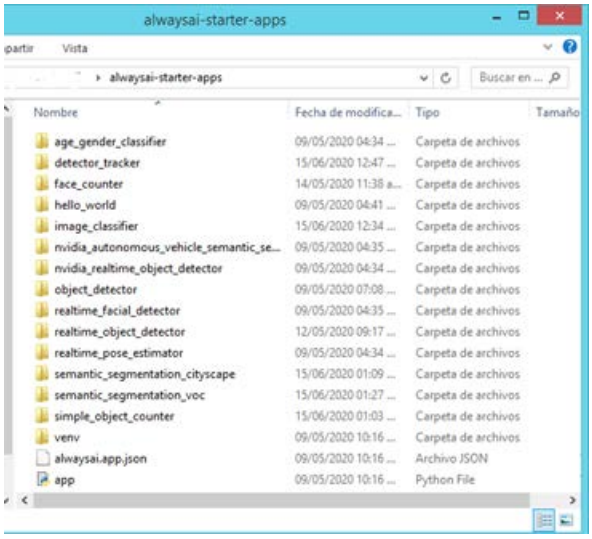


Figura 8 Carpetas de las aplicaciones de inicio.

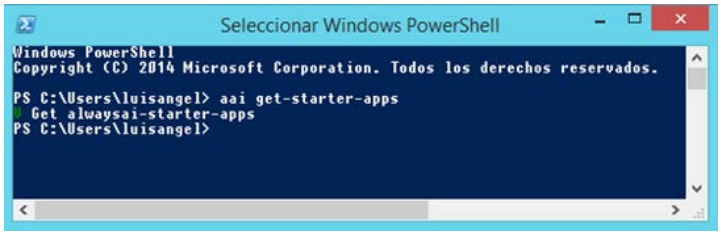


Figura 9 Descarga de desde el PowerShell.

Ejecución de los algoritmos

Una vez con los archivos descargados, se navega al directorio de la aplicación que se desea correr, esto ejecutando el comando cd <Dirección de las carpetas

descargadas> / <Aplicación> en la terminal PowerShell. Si fueron descargadas desde el PowerShell, como en este caso, el comando será: cd alwaysai-starter-apps/<Aplicación>. En la tabla 5 se muestra los comandos que se deben ejecutar para navegar al directorio de acuerdo con la aplicación que se requiere.

Tabla 5 Lista de aplicaciones y su comando.

Aplicación	Comando
Clasificador de género por edad.	*cd alwaysai-starter-apps/age_gender_classifier*
Seguimiento de objetos.	*cd alwaysai-starter-apps/detector_tracker*
Contador de caras.	*cd alwaysai-starter-apps/ face_counter*
Clasificador de imagen.	*cd alwaysai-starter-apps/image_classifier*
Segmentación semántica de vehículos autónomos para nvidia.	*cd alwaysai-starter-apps/nvidia_autonomous_vehicle_semantic_segmentation*
Detector de objetos en tiempo real para nvidia.	*cd alwaysai-starter-apps/nvidia_realtime_object_detector*
Detector de objetos.	*cd alwaysai-starter-apps/object_detector*
Detector facial en tiempo real.	*cd alwaysai-starter-apps/realtime_facial_detector*
Detector de objetos en tiempo real.	*cd alwaysai-starter-apps/realtime_object_detector*
Estimación de pose en tiempo real.	*cd alwaysai-starter-apps/realtime_pose_estimator*
Segmentación semántica de paisaje urbano.	*cd alwaysai-starter-apps/ semantic_segmentation_cityscape*
Segmentación semántica.	*cd alwaysai-starter-apps/semantic_segmentation_voc*
Contador de objetos simple	*cd alwaysai-starter-apps/simple_object_counter*

Una vez en el directorio de la aplicación requerida, se ejecuta el comando aai app install, Este comando solicita las credenciales de su cuenta alwaysAI y luego instala los modelos de la aplicación y un entorno virtual de Python (venv) en el directorio de la aplicación. Una vez instalado correctamente todo, la terminal de PowerSehell se muestra notificaciones como se observa en la figura 10.

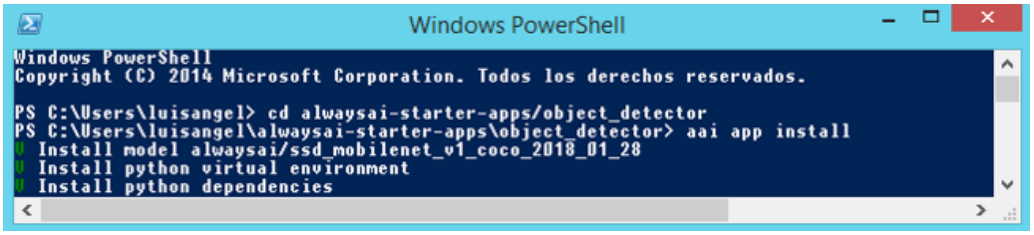


Figura 10 Ventana del PowerShell con los archivos necesarios instalados.

Por último, se inicia la aplicación con el comando aai app start. La salida del registro de app.py se reenvía a través de la CLI a la terminal, incluidas las etiquetas que el modelo ha sido entrenado para reconocer.

Una vez que se inicia la aplicación, se imprime un registro con el enlace, el cual indica que la aplicación o algoritmo de visión está corriendo:

[INFO] Streamer comenzó en http: // localhost: 5000

Para ver los datos de video y texto generados por la aplicación se abre un navegador de internet en donde se ingresa al URL http: // localhost: 5000. Para detener la aplicación, simplemente se presiona CTRL-C. En la figura 11 de observa la interfaz donde se muestran los resultados de los algoritmos.

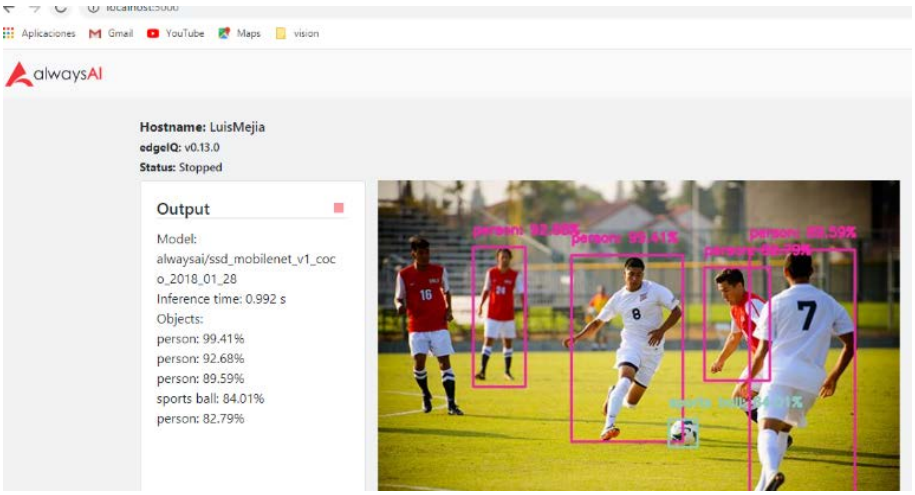


Figura 11 Interfaz de resultados de los algoritmos.

En la parte izquierda se encuentra un cuadro en el cual muestra el nombre de Host, la versión de edgeIQ y el estado:

- Hostname: xxxxxx

- edgeIQ: v0.13.0
- Status: detenido.

También se visualiza el modelo utilizado, el tiempo de interferencia y los objetos identificados junto con su porcentaje de precisión, por ejemplo:

- Model: alwaysai/ssd_mobilenet_v1_coco_2018_01_28
- Inference time: 0.992 s
- Objects:
- person: 99.41%
- person: 92.68%
- person: 89.59%
- sports ball: 84.01%
- person: 82.79%

Por último, se muestra en grande y al centro la imagen analizada o en caso contrario el video capturado por la webcam.

Cambio del modelo de visión por computadora.

Se puede modificar la aplicación fácilmente para usar un modelo diferente y ver cómo se comporta. Como se mencionó anteriormente, la plataforma de alwaysAI facilita un catálogo extenso de modelos que se pueden implementar. Para esto se tiene que cambiar manualmente el modelo siguiendo algunos sencillos pasos que se describen a continuación.

- Para empezar, se accede al directorio de la aplicación de la cual se quiera cambiar el modelo en la terminal de PowerShell.
- Como primer paso se agrega el nuevo modelo a la configuración de su aplicación con el siguiente comando:

  aai app models add alwaysai/<Modelo>

- Después se abre el scrip con el IDLE de Python app.py, el cual se encuentra en la carpeta de aplicación las cuales fueron descargadas en el paso G, se ubica la linea de código:

obj_detect = edgeiq.ObjectDetection("<Modelo>")

- Y se remplaza el modelo por defecto que tiene el algoritmo, por el que se desea implementar, por ejemplo:

  obj_detect = edgeiq.ObjectDetection(" ID modelo por defecto")

  se remplaza por:

  obj_detect = edgeiq.ObjectDetection("ID del modelo nuevo")

- Los ID de los modelos los obtenemos de la plataforma alwaysAI/ catalogo de modelos.
- Como paso opcional, se puede eliminar el modelo anterior de la configuración de su aplicación, con el comando:

aai app models remove alwaysai/<Modelo>

3. Resultados

Una vez analizados los algoritmos, la documentación de la plataforma alwaysai en su página web (Figura 12) y de haber realizado los pasos F y G de la sección anterior, se procedió a realizar las pruebas en un sistema operativo con Windows 10 siguiendo las instrucciones del apartado H de la misma sección. La plataforma proporciona dos tipos de algoritmos; los que requieren una cámara web para su ejecución y estos hacen la parte de visión en tiempo real y por otra parte se encuentran los que analizan imágenes previamente agregadas a una carpeta en el directorio de la aplicación, un ejemplo de esto se muestra en la figura 13. En el desarrollo de este proyecto se realizaron pruebas a ambos tipos de aplicaciones.

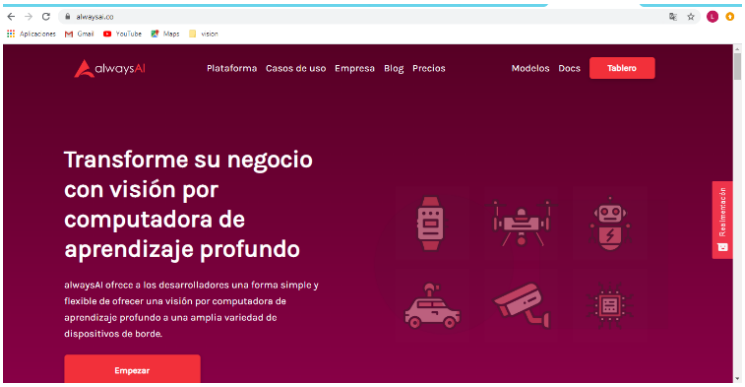


Figura 12 Página principal del sitio web alwaysIA.

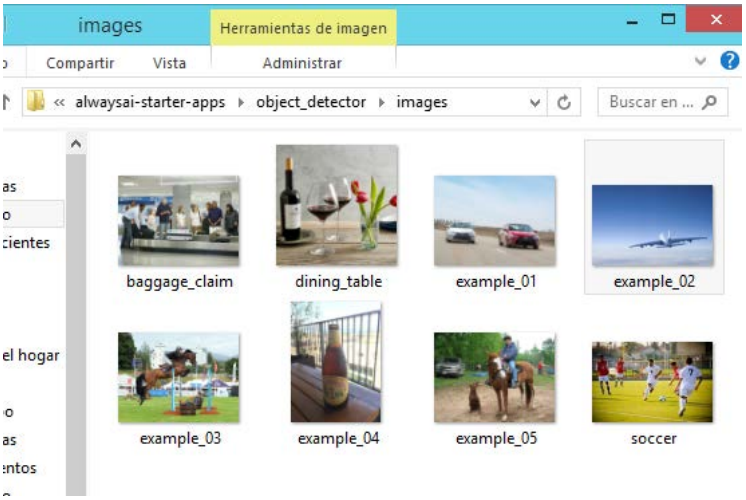


Figura 13 Imágenes analizadas por el algoritmo de detección de objetos.

Detección de Objetos

El algoritmo de detección de objetos analiza imágenes previamente agregadas a la carpeta *images* que se encuentran dentro del directorio del algoritmo, como se muestra en la figura 13. Las imágenes deben ser formato *JPEG (.jpg)*.

Después de agregar las imágenes que se desean analizar mediante técnicas de visión por computadora, se instala lo necesario y ejecuta siguiendo los procedimientos del apartado H de la sección anterior (Figura 14).

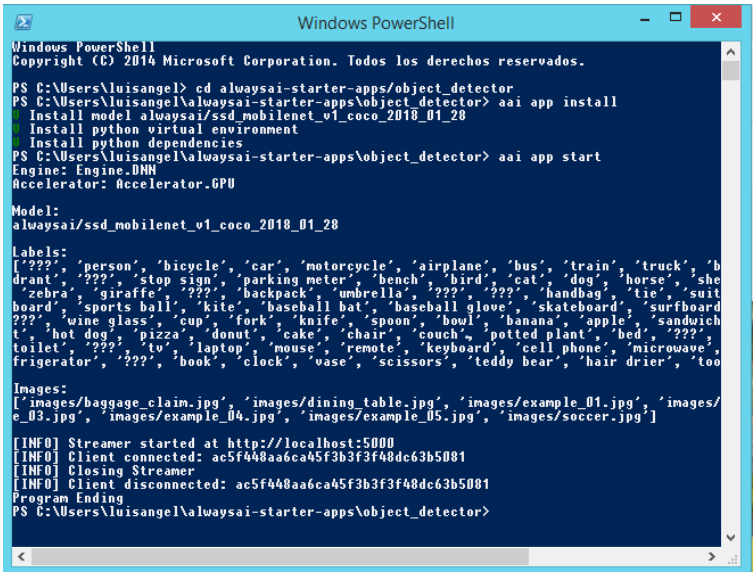


Figura 14 Ejecución del algoritmo de detección de objetos en terminal de PowerShell.

Por último, visualizamos los resultados (Figuras 15 y 16) como se indica en la misma sección. Como se observa, inserta un cuadro delimitador junto con su etiqueta de cada objeto identificado.

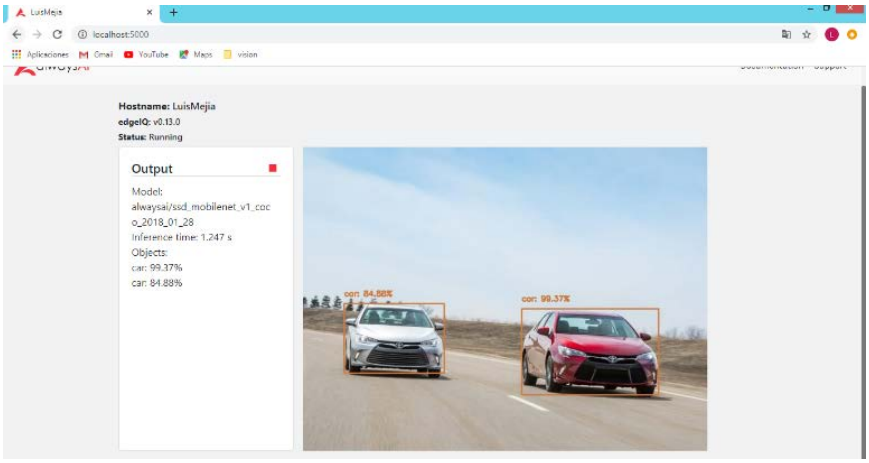


Figura 15 Resultado del algoritmo Detección de Objetos en el cual reconoce dos carros.

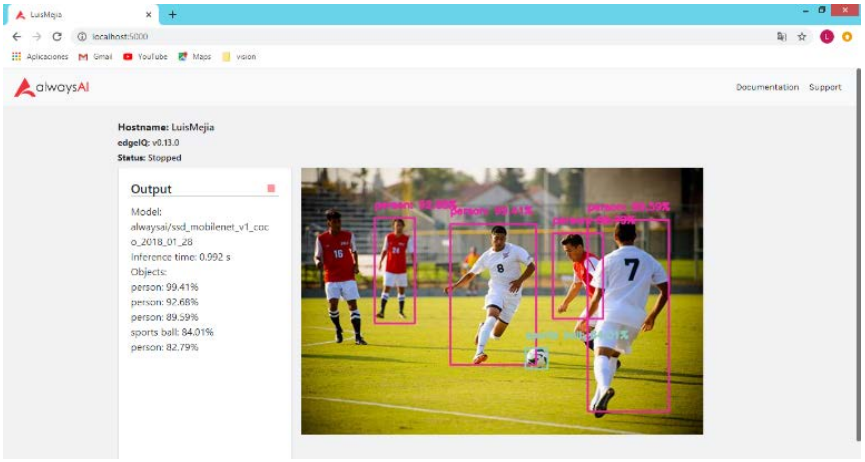


Figura 16 Resultado del algoritmo Detección de Objetos de un juego de Futbol.

Clasificador de imágenes

El algoritmo clasificador imágenes también hace uso de imágenes previamente agregadas, siguiendo la metodología del algoritmo anterior, se ejecutó el clarificador de imágenes, solo que en este caso las imágenes deben de tener un formato *PNG (.png)*. Los resultados se visualizan en las figuras 17 y 18, se observa que delimita con un cuadro e inserta una etiqueta de cada objeto identificado.

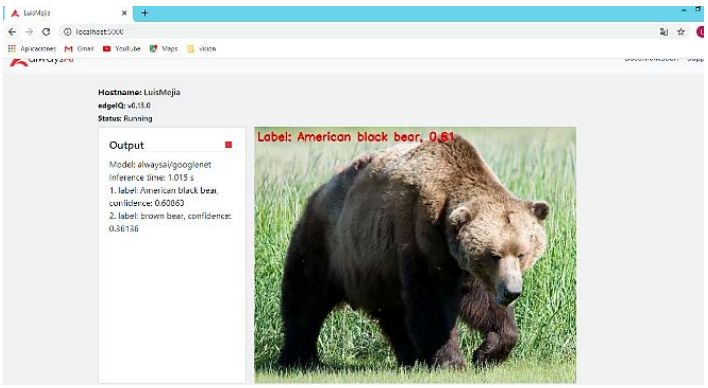


Figura 17 Resultado del algoritmo Clasificador de imágenes Reconociendo un oso.

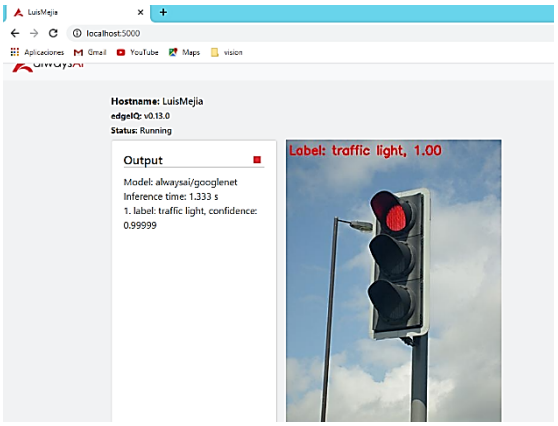


Figura 18 Resultado del algoritmo Clasificador de imágenes reconociendo un semáforo.

Segmentación semántica de paisaje urbano

El algoritmo de Segmentación semántica de paisaje urbano se ejecuta siguiendo la metodología de los algoritmos anteriores, solo que este acepta formato de imágenes en dos formatos; *PNG (.png) y JPEG (.jpg)*. Los resultados se observan en las figuras 19 y 20. Como se observa en la figura 21, la interfaz muestra las diferentes tonalidades de colores con la que segmenta cada objeto identificado en las imágenes analizadas.

Segmentación semántica

Este algoritmo, al igual que los anteriores, también hace uso de imágenes previamente cargadas. Su objetivo es mostrar una visión semántica de diferente tipo de objetos y segmentarla de acuerdo con lo que analice, como se muestra en las

figuras 22, 23 y 24. Del mismo modo que el algoritmo anterior, la interfaz muestra las diferentes tonalidades de colores con la que segmenta cada objeto identificado y acepta formatos de imagen *PNG (.png) y JPEG (.jpg)*.

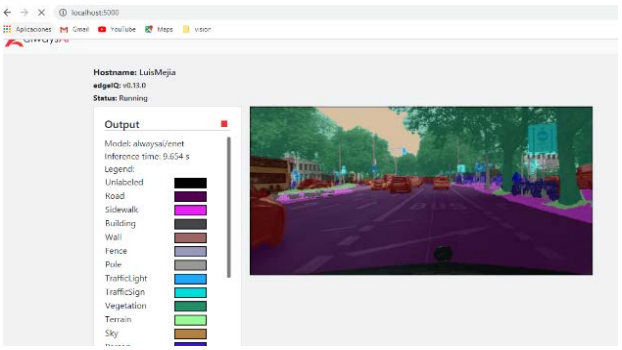


Figura 19 Resultado 1 del algoritmo Segmentación semántica de paisaje urbano.

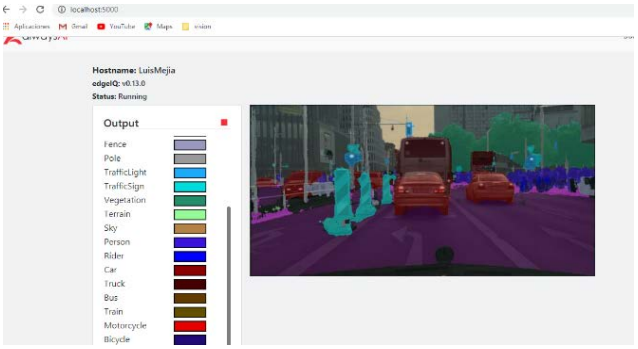


Figura 20 Resultado 2 del algoritmo Segmentación semántica de paisaje urbano.

Figura 21 Tonalidades con las que segmenta los objetos el algoritmo.

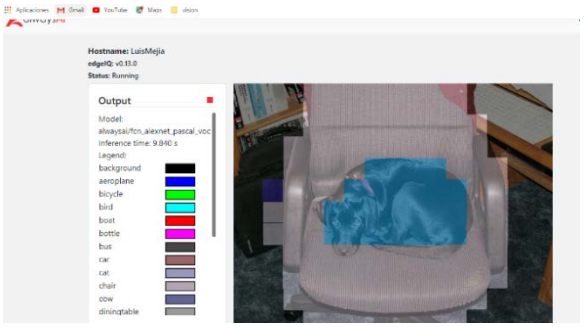


Figura 22 Resultado del algoritmo Segmentación semántica detectando un perro.

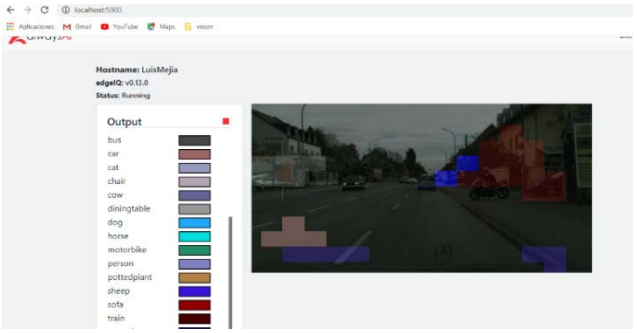


Figura 23 Resultado del algoritmo Segmentación semántica de una calle.

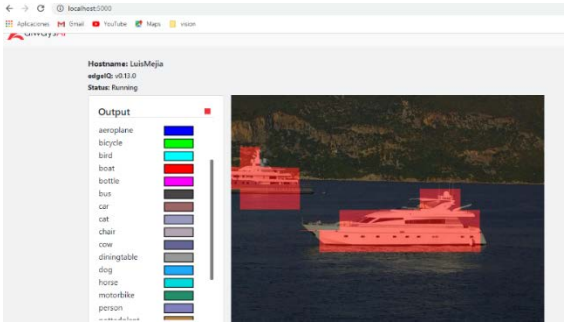


Figura 24 Resultado del algoritmo Segmentación semántica de un paisaje.

Contador de objetos simple

El algoritmo contador de objetos requiere de una webcam para el análisis de video en tiempo real a través de esta, en este caso la computadora que se utilizó para realizar las prueba cuenta con una integrada por lo que no fue necesario conectar una webcam externa. Lo resultados se pueden observar en las figuras 25 y 26. En la interfaz de resultados, en la parte izquierda se observa un recuadro con las etiquetas que analiza el algoritmo y muestra la cantidad de cada uno de los

objetos identificados mediante la webcam (Figura 27). También muestra el video capturado en donde enmarca con un cuadro delimitador los objetos y en la parte inferior se muestra la fecha y hora en la que se captura el video (Figuras 25 y 26).

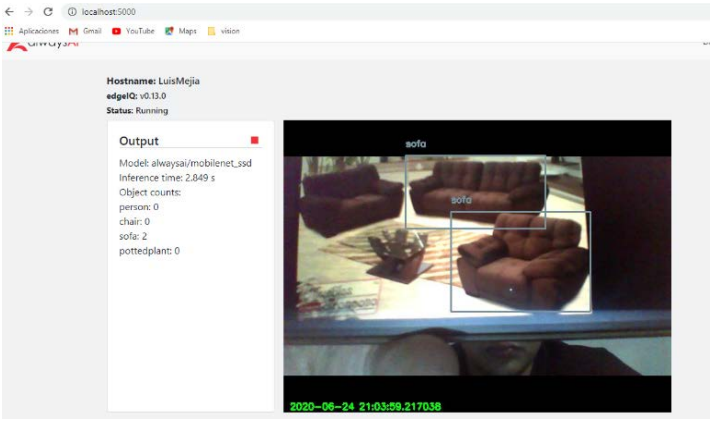


Figura 25 Resultado del algoritmo Contador de objetos simple en reconocer 2 de 3 sofás.

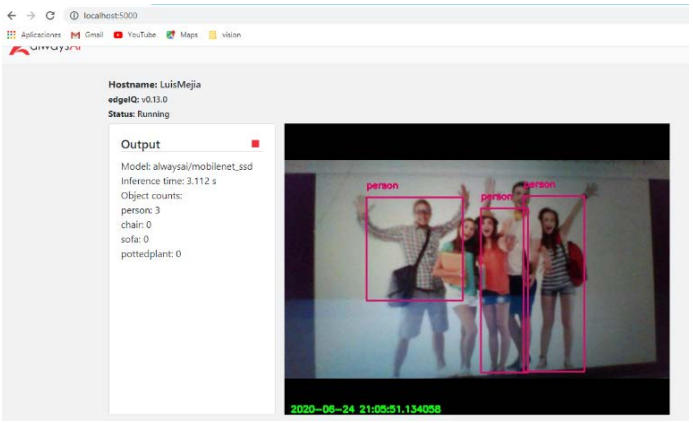
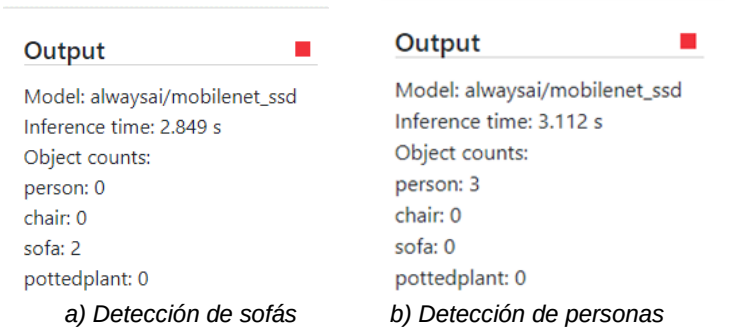


Figura 26 Resultado del algoritmo Contador de objetos reconociendo 2 personas.



a) Detección de sofás *b) Detección de personas*

Figura 27 Interfaz de la cantidad de los objetos identificados por el algoritmo.

Detector facial en tiempo real

El algoritmo de Detector facial en tiempo real captura el video mediante una webcam y lo analiza el algoritmo. Una vez analizado, se muestra el video capturado y enmarca una cara, si es que la encuentra, colocando el porcentaje de precisión junto al recuadro, como se muestra en la figura 28.

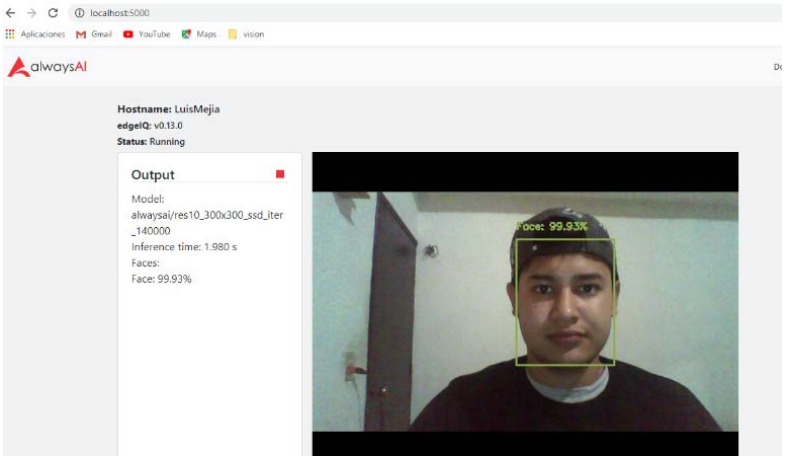


Figura 28 Resultado de las pruebas realizadas al algoritmo Detector facial.

Detector de objetos en tiempo real

Al igual que el algoritmo anterior, se captura video mediante una webcam y el algoritmo analiza lo que se encuentra frente de ella, marcando con un cuadro y colocando la etiqueta del objeto detectado en el instante. Los resultados de las pruebas de observan en las figuras 29 y30.

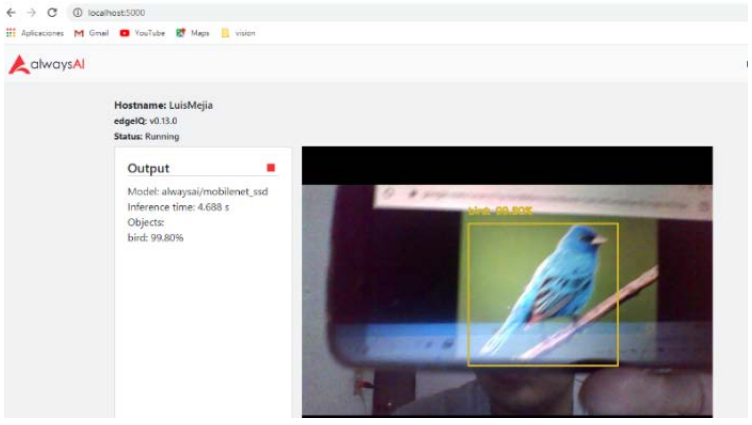


Figura 29 Resultado del algoritmo Detector de objetos en tiempo real detectando un ave.

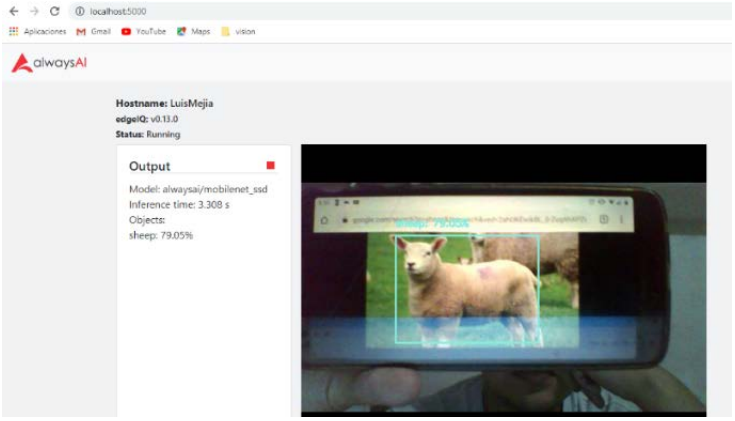


Figura 30 Resultado del algoritmo Detector de objetos en tiempo real.

Seguimiento de objetos

El algoritmo enmarca el objeto identificado con un cuadro delimitador y le inserta su respectiva etiqueta, los cuales, como su nombre lo indica, siguen al objetivo capturado mediante la webcam por todo espacio de la escena a donde se mueva. En la figura 31 se puede visualizar el resultado.

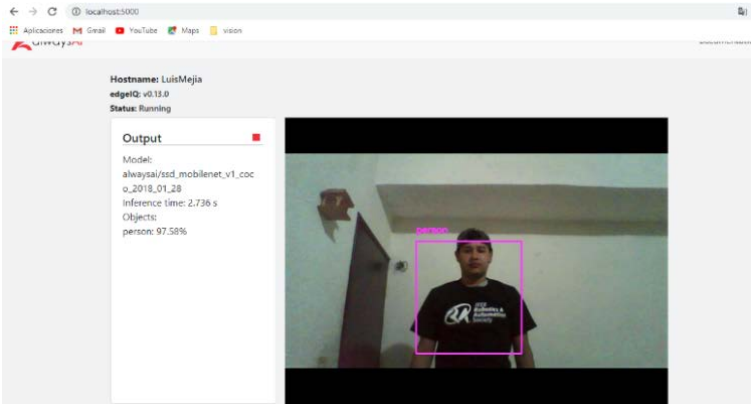


Figura 31 Resultado de las pruebas realizadas al algoritmo Seguimiento de objetos.

4. Discusión

En el Laboratorio de Internet de las Cosas realizamos varios proyectos que emplean la detección de objetos. No obstante, se requiere una plataforma que pueda correr perfectamente en sistemas embebidos como la Raspberry Pi. Lo anterior, ya que todos los esfuerzos tienen que ser canalizados hacia los robots de asistencia y servicio con los que se participa en distintos certámenes.

5. Conclusiones

En este artículo se documenta acerca del procedimiento para ejecutar los algoritmos de visión por computadora de alwaysAI, así como los resultados obtenidos a la hora de probarlos en una computadora de desarrollo, los cuales fueron satisfactorios. No obstante, en futuras actualizaciones del proyecto, se tiene como objetivo llevar los algoritmos hacia un sistema embebido como la Raspberry Pi para visualizar y documentar resultados.

6. Bibliografía y Referencias

[1] Alpaydin E., Introduction to Machine Learning, London: MIT Press, 2010.

[2] Anderson J., La era de las maquinas inteligentes, Buenos Aires: Cambridge: MIT Press, 2017.

[3] Goodfellow I., Bengio Y. y Courville A., Deep Learning, Cambridge: MIT Press, 2017.

[4] Rouhiainen L., Inteligecia Artificial, 101 cosas que debes saber hoy sobre nuestro futuro, Barcelona: Alienta Editorial, 2018.

[5] Somolinos J. A., Avances en Robótica y Visión por Computador, Ciudad Real: Universidad de Castilla-La Macha, 2002.

[6] Vijayalakshmi S. R. y Muruganand S., Embedded Vision: An Introduction, Dulles: Mercury Learning and Information, 2020.