Universidad Católica de Santa María

Facultad de Ciencias e Ingenierías Físicas y Formales

Escuela Profesional de Ingeniería Electrónica

Diseño e implementación de un prototipo de un sistema IOT para la identificación de actos delictivos y ejecución de acciones disuasorias mediante el uso de la tarjeta ESP32-CAM, servidor web y algoritmos de aprendizaje automático

Tesis presentada por el Bachiller:

Diaz Lima, Renee Edmundo

(0009-0004-2388-3683)

para optar el Título Profesional de Ingeniero Electrónico con especialidad en Telecomunicaciones

Asesor (a):

Mg. Valdivieso Herrera, Diana Isabel

(0009-0008-2496-9445)

Arequipa- Perú

2024

UCSM-ERP

UNIVERSIDAD CATÓLICA DE SANTA MARÍA

INGENIERIA ELECTRONICA

CON ESPECIALIDAD EN TELECOMUNICACIONES

TITULACIÓN CON TESIS

DICTAMEN APROBACIÓN DE BORRADOR

Arequipa, 18 de Diciembre del 2023

Dictamen: 008130-C-EPIE-2023

Visto el borrador del expediente 008130, presentado por:

2015245341 - DIAZ LIMA RENEE EDMUNDO

Titulado:

DISEÑO E IMPLEMENTACIÓN DE UN PROTOTIPO DE UN SISTEMA IOT PARA LA IDENTIFICACIÓN ACTOS DELICTIVOS Y EJECUCIÓN DE ACCIONES DISUASORIAS MEDIANTE EL USO DE LA TARJETA ESP32-CAM, SERVIDOR WEB Y ALGORITMOS DE APRENDIZAJE AUTOMÁTICO

Nuestro dictamen es:

APROBADO

29267682 - RODRIGUEZ GONZALES PEDRO ALEX
DICTAMINADOR

29715414 - COAGUILA GOMEZ RONALD PERCING
DICTAMINADOR

29410027 - SULLA TORRES RAUL RICARDO
DICTAMINADOR

Dedicatorias

Dedico este trabajo de tesis a:

A mi madre Paula, que a su manera supo motivarme, darme su bendición, pedirla a dios, guiarme y cuidarme para que pudiera llegar a esta instancia en mis estudios.

A mi padre Renee, que siempre me preguntaba que tal estaba el desarrollo y aun cuando le decía que aún faltaba que nunca dudó de que lo lograría. Siempre me apoyó, me entregó las piezas para su construcción, para las pruebas y para su realización. Aún me sigue apoyando para continuar con mis estudios.

A mi abuelo Eusebio. Gracias a él, quien me llevó a conocer la fábrica de harina de pescado donde trabajaba. Me mostró todas las máquinas operando, grandes molinos girando con solo presionar botones, y compartió conmigo sus historias sobre cómo reparaba las máquinas y las fallas que encontraba. Aprendí que a veces un simple relé oxidado puede detener toda la maquinaria, pero solo observando, escuchando e incluso oliendo puedes encontrar las fallas. Fue esto lo que me motivó a estudiar y convertirme en ingeniero, como él.

Agradecimiento

Quisiera expresar mi profundo agradecimiento a todos aquellos que han contribuido de manera significativa a la culminación de este trabajo de investigación. En primer lugar, quiero agradecer a mi asesora, Ing. Valdivieso Herrera Diana Isabel, Su dedicación y apoyo incansable han sido la brújula que guio mis esfuerzos hacia la excelencia académica.

Agradezco sinceramente a todos los profesores de la Escuela profesional de Ingeniería Electrónica por compartir sus conocimientos y experiencias, moldeando así mi comprensión y aprecio por el vasto campo de la electrónica. Sus enseñanzas han sido la base sobre la cual construí este trabajo.

También agradezco a mi KY me motivo cuando había varios errores de diseño y los modelos no funcionaban, me dio ánimos para que encontrara las soluciones, gracias a ti no deje de intentarlo hasta lograrlo ya que tú nunca dejas de intentarlo hasta lograrlo

A mi familia y seres queridos, les agradezco por su inquebrantable apoyo emocional y motivacional. Sus palabras alentadoras y paciencia infinita fueron mi inspiración constante.

En resumen, mi más sincero agradecimiento a cada persona que formó parte de este viaje. Sus contribuciones no solo han enriquecido mi experiencia académica, sino que también han dejado una huella imborrable en mi crecimiento personal y profesional.

A todos ustedes, mi más profundo agradecimiento.

RESUMEN

La presente tesis se centra en abordar la problemática de las elevadas tasas de delincuencia existentes en el Perú, especialmente en las áreas urbanas; atribuidas a factores sociales y a la afluencia migratoria. En este contexto, se propone el diseño de un sistema que ofrezca asistencia tanto a la ciudadanía como a las fuerzas policiales y centros de vigilancia ciudadana. Este sistema se basará en la aplicación de inteligencia artificial, para la detección de actividades delictivas; con el propósito de notificar a los operadores de los sistemas de videovigilancia ciudadana. Esta notificación y las medidas disuasorias se plantean como una medida activa destinada a prevenir o disuadir la comisión de delitos, en consecuencia; se espera que este sistema contribuya a aumentar la eficiencia de la vigilancia ciudadana.

El enfoque principal de este trabajo se concentra en los delitos de robo y asalto, que presentan un alto número de denuncias. Con el objetivo de mejorar la seguridad ciudadana, se plantea el desarrollo e implementación de un prototipo de sistema de Internet de las cosas (IoT); diseñado para la identificación de actividades delictivas y la ejecución de acciones disuasorias. Este prototipo empleará la tarjeta ESP32-CAM, servidores web y algoritmos de aprendizaje automático.

Para la identificación de actividades delictivas, se utilizará el modelo MoViNet basado en redes convolucionales 3D; debido a su mayor eficacia en comparación con otros modelos evaluados.

La interfaz de control y clasificación permitirá la conexión a las cámaras IP para procesar y clasificar las imágenes captadas, esta clasificación se llevará a cabo en el lado del cliente mediante Tensorflow JS y en el servidor a través del modelo publicado en Tensorflow Serving; situación que permitirá la ejecución y clasificación en tiempo real de las imágenes.

Se utilizarán medidas disuasorias de tipo lumínico y sonoro para truncar el acto delictivo y/o advertir y salvaguardar al resto de ciudadanos; mediante el módulo disuasorio ESP32-CAM.

Palabras claves:

Videovigilancia, Redes neuronales, IOT.

ABSTRACT

The following text focuses on addressing the issue of high crime rates in Peru, especially in urban areas, attributed to social factors and migratory influx. In this context, the proposal is to design a system that provides assistance to both the public and law enforcement agencies and citizen surveillance centers. This system will be based on the application of artificial intelligence for the detection of criminal activities, with the purpose of notifying operators of citizen vídeo surveillance systems. This notification and deterrent measures are envisioned as an active measure aimed at preventing or deterring the commission of crimes. Consequently, it is expected that this system will contribute to increasing the efficiency of citizen surveillance.

The main focus of this work is on theft and assault crimes, which have a high number of reports. In order to enhance public safety, the development and implementation of an Internet of Things (IoT) prototype system designed for the identification of criminal activities and the execution of deterrent actions are proposed. This prototype will use the ESP32-CAM card, web servers, and machine learning algorithms.

For the identification of criminal activities, the MoViNet model based on 3D convolutional networks will be used, due to its greater efficiency compared to other evaluated models. The control and classification interface will allow the connection to IP cameras to process and classify captured images. Classification will be done on the client side using Tensorflow JS and on the server through the model published in Tensorflow Serving, allowing real-time execution and classification of images. Deterrent measures of a luminous and auditory nature will be used to thwart criminal acts and/or warn and safeguard the rest of the citizens, and this will be carried out by the ESP32-CAM deterrent module.

Key words:

Vídeo surveillance, Neural networks, IOT.

ÍNDICE

ÍNDICE DE FIGURAS

INTRODUCCIÓN

El propósito fundamental de este trabajo de tesis consiste en concebir y poner en marcha un prototipo de sistema que, haciendo uso de cámaras inalámbricas y estaciones de videovigilancia ciudadana; tenga la capacidad de detectar y reconocer actividades ilícitas dentro del ámbito urbano.

El objetivo principal de este sistema radica en la posibilidad de alertar al operario de las estaciones de videovigilancia, acerca de posibles actos delictivos y llevar a cabo acciones disuasorias; que obstaculicen o prevengan la continuación de dichas actividades delictivas por parte del individuo identificado. Este trabajo de tesis busca contribuir a la protección inmediata de la comunidad y a la reducción de la incidencia delictiva en áreas urbanas.

Cabe destacar que, en los últimos años, la tasa de criminalidad en el Perú ha experimentado un preocupante incremento. En este sentido, según los datos proporcionados por el Sistema de Denuncias Policiales (Sidpol), en el año 2020 se registraron un total de 79,920 actos delictivos; mientras que en 2021 esta cifra aumentó en un 18%, alcanzando los 94,789 incidentes delictivos. Los delitos de robo, asalto y sicariato son los que concentran la mayor cantidad de denuncias.

El sistema que se pretende diseñar y desarrollar tiene como finalidad principal, notificar al operador de la estación de videovigilancia y tomar medidas disuasorias; tales como la activación de una sirena que alerte sobre la actividad delictiva en curso. Esto se hace con el fin de proteger a los transeúntes de la zona y generar una presión social que pueda disuadir al individuo infractor. Además, permite la emisión de luz para iluminar la zona afectada, con el propósito de aumentar la visibilidad de los hechos y facilitar la observación de los transeúntes; lo que también actúa como medida disuasoria al eliminar el sigilo del acto delictivo.

Con esta propuesta, se espera incrementar la eficacia de la detección de eventos delictivos a través de la expansión de la red de cámaras de videovigilancia sin necesidad de aumentar la contratación de operadores, dado que el sistema optimizaría la detección de eventos delictivos y la percepción de estos por el operario; permitiendo la implementación de medidas disuasorias y acción de medidas policiales de manera eficiente.

CAPÍTULO I. PLANTEAMIENTO TEÓRICO

1. Título

DISEÑO E IMPLEMENTACIÓN DE UN PROTOTIPO DE UN SISTEMA IOT PARA LA IDENTIFICACIÓN DE ACTOS DELICTIVOS Y EJECUCIÓN DE ACCIONES DISUASORIAS MEDIANTE EL USO DE LA TARJETA ESP32-CAM, SERVIDOR WEB Y ALGORITMOS DE APRENDIZAJE AUTOMÁTICO

2. Identificación del Problema

Actualmente, en el Perú las tasas delictivas existentes en el ámbito urbano son muy elevadas debido a los distintos factores sociales que se vienen presentando en el país como la ola migratoria; que incrementó las tasas delictivas preexistentes.

Según la recomendación de la ONU, se debería contar con (01) efectivo policial por cada 357 habitantes, sin embargo, a pesar de la problemática que se vive en el País; solo se tiene a razón de (01) efectivo policial por cada 1000 habitantes. (Arango, 2022).

Siendo notable la deficiencia de disponibilidad de efectivos policiales, es que se consideró el diseño de un sistema que pueda brindar ayuda y asistencia tanto a la ciudadanía como a la policía; mediante la detección de estos eventos y la aplicación de medidas disuasorias que no representen ningún riesgo para los involucrados. Asimismo, que permita dar aviso de manera rápida a los operadores de las salas de videovigilancia de seguridad ciudadana y/o de los centros policiales.

Los crímenes que el prototipo de Sistema IOT espera abordar, serían los de robo y asalto; esto debido a que la tasa de denuncias de este tipo de crímenes es el mayor de los que se registran en el Sistema Informático de Denuncias Policiales (SIDPOL). Es preciso indicar, que estos crímenes son los que mayor probabilidad de disuasión tienen, esto debido a que la mayoría ellos se realizan de manera rápida y efectiva.

3. Descripción del Problema

Las actividades delictivas como robo y asalto son las que tienen una mayor tasa de denuncias en nuestro país y en la ciudad de Lima, según datos de SIDPOL (Sistema Informático de Denuncias Policiales); se tiene que por cada 100 habitantes 16 son víctimas de estos actos delictivos.

(Instituto Nacional de Estadística e Informática, 2017)Según la encuesta realizada por INEI en el año 2021**,** los factores que más influyen; en las elevadas tasas de inseguridad ciudadana serían:

- Presencia de delincuentes.
- Poca presencia de personal policial y/o serenazgo.
- Poca eficiencia en la detección de actos delictivos dada la gran cantidad de puntos de videovigilancia.
- Consumo de drogas.
- Presencia de pandilleros.
- Poca iluminación.

Las principales causas de los altos niveles de inseguridad se deben a dos factores:

- Dimensión de la ciudad.
- Cantidad de Habitantes. (pág. 98)

3.1. Para la situación actual tenemos:

La mayoría de las ciudades en el País tienen deficiencias en la cantidad de personal de la policía que patrullan y vigilan las calles, aun cuando las actividades de patrullaje pueden tener apoyo del personal de serenazgo, juntas vecinales y otros.

La implementación de la videovigilancia mejora la seguridad de la zona, pero al ser elementos pasivos que solo recaban información resultan poco útiles cuando las áreas a intervenir tienen gran magnitud. Es importante indicar que la información obtenida de la videovigilancia en formato digital, es casi siempre empleada para el identificar al

responsable del acto delictivo más no permite actuar de manera inmediata al momento del suceso del mismo.

Algunas de las medidas disuasorias más comunes que se tienen como defensa para los actos delictivos de robo y asalto, son de tipo Sonoro (Alarmas, silbatos, sirenas, otros), de Tipo Visible (Carteles, Rejas, Cámaras, otros); y de tipo activo (Perros, Guardias y otros)

Adicionalmente a ello, es necesario precisar que los actuales centros de videovigilancia destinados a la seguridad ciudadana; cuentan con una cantidad considerable de monitores y cámaras de seguridad. Esta situación genera la necesidad de contar también con un número elevado de operadores encargados de supervisar las cámaras. Si bien es cierto, que esta disposición incrementa la capacidad de respuesta frente a eventos delictivos, los que son detectados por los operadores; también se da el caso de no ser detectados estos eventos por la gran cantidad de cámaras que deben ser vigiladas por ellos.

3.2. Para la situación deseada:

Con el desarrollo de esta tesis se pretende proponer un Prototipo de Sistema IOT que permita identificar los actos delictivos de robo y asalto, mediante el uso de las imágenes capturadas por una cámara y el uso de redes neuronales; con ello se espera poder comunicar a la policía y/o serenazgo para que puedan acudir en auxilio de la persona afectada oportunamente.

Los centros de videovigilancia equipados con este tipo de sistema, experimentarían una mejora significativa en la detección de eventos delictivos. Esto debido a que los operadores serían alertados oportunamente sobre eventos delictivos detectados por la red neuronal, reduciendo en gran medida la cantidad de casos que pasarían desapercibidos en condiciones normales.

El sistema tiene la capacidad de analizar las cámaras de manera mucho más eficiente que los operadores, lo que permite aumentar el número de puntos de vigilancia sin necesidad de incrementar la cantidad de operadores requeridos para esta tarea. Esto, a su vez, abre la posibilidad de aumentar el número de efectivos disponibles para la seguridad ciudadana.

También, se espera que este sistema pueda ejecutar medidas disuasorias para el truncamiento del acto delictivo; con ello se espera que el sistema pueda aportar una mayor seguridad a la ciudadanía.

Las medidas disuasorias consideradas para este proyecto serían:

- Tipo sonoro: Mediante el uso de una sirena que pueda alertar del hecho delictivo en la zona.
- Tipo visual: Mediante el uso de luces LED para mejorar la visibilidad del evento y advertir a los transeúntes.

4. Objetivos

4.1. Objetivo General

Diseñar e implementar un prototipo de sistema IOT para la identificación de actos delictivos y ejecución de acciones disuasorias mediante el uso de la tarjeta ESP32-CAM, servidores Web y algoritmos de aprendizaje automático.

4.2. Objetivos Específicos

- Seleccionar un algoritmo de Aprendizaje automático que permita identificar actos delictivos
- Entrenar el algoritmo de Aprendizaje automático para la identificación de los actos delictivos de robo y asalto
- Diseñar el código fuente para la identificación del acto delictivo y la ejecución de medidas disuasorias
- Diseñar e implementar la estructura prototipo del sistema
- Diseñar e implementar el circuito electrónico.

5. Alcances

Como resultado de este trabajo de tesis se pretende proponer un prototipo que permita identificar actos delictivos y ejecución de acciones disuasorias de actos delictivos de robo y asalto, contribuyendo de esta forma a la lucha contra el crimen e incrementando la seguridad ciudadana.

El dispositivo a implementar será del tipo activo, es decir, podrá actuar en el momento del evento. Como consecuencia de ello, se espera mejorar la eficiencia de las centrales de videovigilancia; al permitir la identificación de acciones delictivas y aumentar la respuesta ante la ocurrencia de estos eventos.

Las medidas disuasorias tienen como principal objetivo el truncar o dificultar la ejecución del acto delictivo, esto permitirá que el sistema pueda actuar durante la ocurrencia del hecho. Las medidas de tipo sonora y las medidas lumínicas, también tienen el objetivo de poder alertar del evento al resto de transeúntes; permitiendo que estos puedan salvaguardar su integridad.

La advertencia o notificación al operador, será una mejora en la eficiencia del despliegue y asistencia a los afectados; con ello se espera reducir el tiempo de respuesta, aumentar la capacidad de captura del sospechoso y disminuir la tasa de criminalidad en la ciudad donde se implemente el sistema.

CAPÍTULO II. MARCO TEÓRICO

1. Antecedentes de Investigación

1. (HUARAYA APAZA, 2015) "SISTEMA WEB PARA MEJORAR EL CONTROL DE OPERACIONES DE LA SEGURIDAD CIUDADANA EN LA MUNICIPALIDAD DE SANTIAGO DE SURCO". En el trabajo de investigación se realizó el análisis del funcionamiento del sistema de seguridad ciudadana de la municipalidad de Santiago de Surco, las deficiencias que poseen a la hora de coordinar acciones en contra de las actividades delictivas que se den en el sector de cobertura. También se desarrolló una página web que mejore el control de las operaciones de la seguridad ciudadana, mejorando la eficacia del proceso de control de operaciones y automatización del proceso de control de personal; asimismo este trabajo se considera un antecedente debido a que abordó el tema de seguridad ciudadana, las deficiencias que se tiene en el país y las propuestas de mejora que se tomaron.

2. (López Bonilla & Espinoza Quezada, 2022) "ESTUDIO TÉCNICO DE UN NUEVO SERVICIO DE SEGURIDAD CIUDADANA". En este trabajo de investigación se observa el desarrollo de un sistema de seguridad ciudadana, que se encuentra basado en una red wifi montada en la red de un tranvía que circula la ciudad de Cuenca; la red tiene el objetivo de brindar el acceso a internet para todos los usuarios del servicio de tranvía.

   El trabajo tuvo como objetivo usar de manera eficiente el ancho de banda de la red para brindar el servicio adicional de seguridad, que sería la implementación de videovigilancia usando la segmentación de red MPLS L2VPN, políticas de firewall, control de ancho de banda, compresión de vídeo y capacidad optimizada de almacenamiento.

   El desarrollo de este proyecto se realizó por partes siendo la primera de implementación, la siguiente parte intenta demostrar la arquitectura del sistema propuesto y su integración en la infraestructura de la red. Como parte final se

analizó, si el sistema de videovigilancia permitiría una grabación de vídeo eficiente; incluso en entornos de alto voltaje.

Es importante mencionar que este trabajo está muy relacionado al tema de esta tesis, que tiene como uno de sus objetivos desarrollar un sistema de videovigilancia; que será utilizado posteriormente para analizar los fotogramas capturados y determinar si éstos representan o no una conducta delictiva.

3. (Barrientos Avendaño, Pérez Yáñez, & Rico-Bautista) "DISPOSITIVO DE SEGURIDAD IOT SOPORTADO POR SENSORES PARA EL MONITOREO DE BICICLETAS". Este trabajo propuso un sistema IOT para la detección del robo de bicicletas, mediante el uso de sensores como GPS, con la finalidad de brindar apoyo rápido al momento que se realice el robo, aumentado la posibilidad de recuperar la bicicleta. En este trabajo de investigación vió la capacidad que tienen los dispositivos IOT en la prevención de delitos y aumento de la seguridad ciudadana; información que servirá de guía para el desarrollo de este trabajo de tesis.

4. (Calderón Ñaccha, Santillán Paredes, & Masías Donayre)" PLAN DE NEGOCIOS PARA IMPLEMENTAR UN SISTEMA DE DETECCIÓN Y ALERTAS"). En este trabajo de investigación se realizó un análisis de las ventajas del uso del reconocimiento facial y su relación con temas de seguridad civil, además se realizó la descripción de indicadores percibidos de ansiedad, viviendas afectadas por robo, evidencia de la ineficiencia de la policía, patrulla policial y falta de iniciativa de las entidades del Estado Peruano. Este trabajo abordó la percepción que tendría la población en el uso de reconocimiento facial y tecnologías similares para combatir la delincuencia. Considerando la inteligencia artificial y la seguridad ciudadana, este trabajo es un antecedente a considerar en la tesis debido a que se desarrollaron temas relacionados con reconocimiento facial; por lo que permitirá profundizar un poco más a detalle la capacidad de aceptación que tendrá el sistema propuesto.

5. (Machaca Arcana, 2019) "RECONOCIMIENTO DE EVENTOS ANÓMALOS EN VÍDEOS OBTENIDOS DE CÁMARAS DE VIGILANCIA, USANDO REDES CONVOLUCIONALES". En este trabajo de investigación se observó el uso de redes neuronales convulsiónales para el campo de reconocimiento de movimiento y acciones en videovigilancia, destacando su importancia y las aplicaciones potenciales. Se resaltó la utilidad de los modelos basados en deep learning para abordar la incertidumbre asociada con las acciones humanas y la detección de objetos en tiempo real. Se mencionaron diferentes técnicas de detección de objetos, como YOLO, RCNN y SSD, que presentan un nuevo enfoque en la detección de objetos en imágenes. Como parte de la inteligencia artificial y la seguridad ciudadana, este trabajo se considera como un antecedente, porque trata temas de reconocimiento de eventos anómalos en una zona vídeo vigilada; que contribuirá en profundizar en las redes neuronales convolucionales.

2. Base teórica

2.1. Efecto Disuasorio

El efecto disuasorio es el resultado de varios factores que disuaden e impiden que un ladrón robe antes de cometerlo, eso incluye cambiar la mente de un ladrón; que se puede dar cuando un ladrón intenta cometer un robo o un delito.

2.1.1. Tipo de ladrones

(Sánchez , Laura, 2023)Según el informe de una ex agente del FBI y psicóloga criminal, la clasificación de criminales se da con base en la personalidad y comportamiento, siendo estos:

- **Organizados:** Se refiere a individuos delincuentes con historial de reincidencia que llevan a cabo sus acciones de manera premeditada.
- **Desorganizados:** Generalmente, se trata de jóvenes ladrones que actúan de forma intermitente, focalizándose principalmente en objetos de fácil acceso.
- **Interpersonales:** Estos criminales están familiarizados con sus víctimas y suelen cometer robos con el propósito de ejercer control sobre ellas, generando un sentido de temor.
- **Oportunistas:** Este tipo de ladrón es un principiante que aprovecha circunstancias propicias, como una ventana o puertas abiertas para delinquir.

Además, encontramos las bandas criminales, dedicadas exclusivamente al robo. Estos grupos bien organizados y estructurados cuentan con un profesional para cada tarea dentro del robo, esta organización existente permite que se puedan perpetrar varios asaltos y robos en poco tiempo. (pág. 1)

2.1.2. Razones por las cuales las medidas disuasorias son efectivas en la prevención de robos y asaltos

- **Incrementan el riesgo percibido:** Las medidas disuasorias, como la presencia visible de cámaras de seguridad, alarmas, sistemas de vigilancia y señales de advertencia, aumentan la percepción de riesgo para los posibles delincuentes. Saber que hay medidas de seguridad en un determinado lugar, puede disuadir a los criminales de cometer un delito; ya que temen ser detectados o atrapados en el acto.
- **Reducen la oportunidad**: Las medidas disuasorias a menudo incluyen el fortalecimiento de la seguridad física, como puertas y ventanas resistentes, cerraduras de alta calidad y sistemas de control de acceso. Estos obstáculos hacen que sea más difícil para los delincuentes acceder a una propiedad, lo que reduce la oportunidad de cometer un robo o asalto.
- **Aumentan la conciencia pública:** Las medidas disuasorias también pueden aumentar la conciencia pública sobre la importancia de la seguridad. Cuando las personas saben que existe una amenaza y que se están tomando medidas para prevenirla, tienden a estar más alerta y tomar precauciones adicionales; lo que puede disuadir a los delincuentes.
- **Facilitan la detección y la respuesta rápida:** Los sistemas de seguridad modernos, suelen estar conectados a centrales de monitoreo o a aplicaciones móviles; que permiten a los propietarios o a las autoridades recibir alertas en tiempo real en caso de intrusión o actividad sospechosa. Esto facilita la detección temprana y respuesta rápida, lo que puede prevenir robos y asaltos; o permitir la captura de los delincuentes.
- **Crea un entorno hostil para los delincuentes:** Cuando un lugar está bien protegido con medidas disuasorias efectivas, puede crear un entorno; en el que los delincuentes sientan que es demasiado riesgoso cometer un delito.

> Esto puede disuadir a los criminales en busca de objetivos fáciles y alentarlos a buscar oportunidades en otros lugares. (Securitas, 2020, pág. 1)

2.2. Microcontroladores

Un microcontrolador es un circuito integrado programable, que puede ejecutar instrucciones almacenadas en la memoria; está compuesto de varios bloques funcionales que realizan tareas específicas. Un microcontrolador contiene las tres unidades funcionales principales de una computadora:

- Unidad central de procesamiento
- Memoria
- Periféricos de entrada/salida.

Algunos microcontroladores utilizan palabras de 4 bits, funcionan a velocidades de reloj tan bajas como 4 kHz y funcionan con un bajo consumo de energía en el orden de los micro watts (mW). Por lo general, tienen la capacidad de esperar eventos como pulsaciones de botones u otras interrupciones.

Dependiendo del uso a darle o del sistema donde se encontrarán incluidos, el factor más importante para su selección sería el rendimiento. Cabe indicar que, a medida que los requerimientos crezcan, como por ejemplo en los microcontroladores donde deben comportarse como procesadores de señal digital (DSP); se tendrá que trabajar con altas velocidades de reloj y por ende se tendrá un alto consumo de energía. Para controlar el proceso, se debe escribir un programa y almacenarlo en una EEPROM; este programa puede estar escrito en lenguaje ensamblador u otro lenguaje para microcontroladores. Sin embargo, para almacenar un programa en la memoria del microcontrolador; debe estar codificado en un sistema hexadecimal (Valdéz Peréz & Pallás Areny, 2007, pág. 11).

Ilustración 1: Microcontroladores de la marca Microchip,
Fuente: https://microcontroladoressesv.wordpress.com/microcontroladores-pic-y-sus-variedades/

2.2.1. Tipos de arquitectura

Arquitectura VON NEUMANN

(Valdéz Peréz & Pallás Areny, 2007, pág. 21) En la arquitectura de Von Neumann el dispositivo de almacenamiento para instrucciones, es el mismo donde se almacenan datos; esto se usa en computadoras personales para ahorrar muchas líneas de E/S (Entradas y Salidas) que son muy costosas. Además, esta organización evita muchos problemas a los diseñadores de placas base y reduce el costo de estos sistemas. En una computadora personal, cuando un programa se carga en la memoria; asigna un espacio de direcciones de memoria segmentado.

Para los microcontroladores, hay dos tipos de memoria:

- Memoria de datos (generalmente un tipo de SRAM)
- Memoria de programa (ROM, PROM, EEPROM, flash u otro tipo no volátil).

Como se puede observar, las memorias se encuentran separadas lo que contradice la definición anteriormente dada; pero aún se ubican en esta categoría debido a que los buses de acceso a los dos tipos de memoria son los mismos. En conclusión, la forma en que la memoria está conectada al procesador, sigue los mismos principios definidos por la arquitectura subyacente; por lo que la arquitectura permanece sin cambios.

Arquitectura Harvard

(Valdéz Peréz & Pallás Areny, 2007, pág. 21) En la arquitectura Harvard la memoria de instrucciones y la de almacenamiento tiene buses separados, por lo que cada tipo de memoria tiene un bus de datos; un bus de direcciones y un bus de control. Una ventaja fundamental de esta arquitectura, es que el tamaño del bus se puede ajustar; a las características de cada tipo de memoria. Además, el procesador puede acceder a todos ellos simultáneamente; lo que acelera enormemente el procesamiento. En general, los sistemas con esta arquitectura, son el doble de rápidos que los sistemas similares; con arquitectura de Von Neumann. La desventaja es que consume muchas líneas de E/S (Entrada y

Salida) en los microcontroladores y otros sistemas embebidos, donde los datos y la memoria del programa a menudo comparten el mismo paquete que el procesador; la deficiencia anteriormente mencionada no es problema grave y es por eso que las arquitecturas de Harvard se encuentran en la mayoría de los microcontroladores.

2.2.2. Partes de un microcontrolador

Con referencia en (Gunther Gridling, 2007, págs. 11-71) las partes de los microcontroladores son:

Registros

Son un espacio de memoria muy pequeño, donde se almacenan los datos para las diversas operaciones; que se realizarán en el resto de circuitos del procesador. Los registros se utilizan para almacenar los resultados de ejecutar instrucciones, cargar datos desde una memoria externa o almacenar datos en una memoria externa.

Los registros definen uno de los parámetros más importantes del microprocesador, los tamaños de registros en los microcontroladores son de 4, 8, 16, 32 o 64 bits; esto determina gran parte del potencial de estas máquinas y la combinación de componentes que tendrá el microcontrolador.

Unidad de Control

Es el cerebro del microcontrolador y contiene la lógica necesaria para decodificar y ejecutar instrucciones y controlar registros, ALUs y buses.

Los bloques de control son uno de los determinantes fundamentales del rendimiento del procesador, y su tipo y estructura determinan parámetros como el tipo de conjunto de instrucciones que puede tener un sistema, la velocidad de ejecución, el tiempo de ciclo de la máquina, el tipo de bus, etc.

Unidad aritmético-lógica (ALU)

Es un circuito que realiza operaciones lógicas y matemáticas, toda la unidad está dedicada a este proceso y tiene cierto grado de independencia. Aquí es donde se realizan las operaciones típicas de suma, resta y lógica del álgebra booleana.

Adicionalmente a ello la mayoría de los microprocesadores modernos tienen varias ALU, dedicadas a realizar operaciones complejas; como operaciones de punto flotante. También tiene un gran impacto en el rendimiento del procesador, dependiendo de su potencia puede realizar tareas más o menos complejas; como calcular el seno flotante en muy poco tiempo.

Buses

Son los medios de comunicación utilizados por los diversos componentes del procesador, para intercambiar información entre ellos.

Para los microcontroladores, es común que los buses se reflejen en la encapsulación del circuito; ya que son principalmente para E/S de uso general y periféricos del sistema.

Existen tres tipos de buses:

- **Dirección:** Se utiliza para seleccionar al dispositivo con el cual se quiere trabajar, o en el caso de las memorias; seleccionar el dato que se desea leer o escribir.
- **Datos:** Se utiliza para mover los datos entre los dispositivos de hardware (entrada y salida).
- **Control:** Se utiliza para gestionar los distintos procesos de escritura lectura y controlar la operación de los dispositivos del sistema.

Periféricos

Los periféricos son circuitos digitales, los cuales permiten interactuar con las señales externas. Un microcontrolador puede tener diferentes tipos de periféricos como:

- **Entradas y Salidas:** Generalmente agrupados en puertos de 8 bits, nos dan un camino bidireccional para recibir o escribir información; con el fin de operar dispositivos simples como relés, luces y LED.
- **Temporizadores y Contadores:**
    - Contador: Es un circuito síncrono, que cuenta los pulsos que ingresan a la fuente de alimentación; para la entrada del

reloj. Si la fuente de los contadores grandes es el oscilador interno del microcontrolador, generalmente no tienen un pin de enlace; en cuyo caso actúan como temporizadores. Por otro lado, si la fuente del contador es externa, tiene un pin de enlace configurado como una entrada; que está en modo de contador.

  - Temporizador: Es un circuito síncrono muy común en los microcontroladores y se utilizan para muchas tareas, como medir frecuencias, implementar relojes y comunicarse con otros dispositivos que requieren una base de tiempo.

- **Conversor Analógico Digital:** Circuitos eléctricos con la función de convertir señales analógicas a digitales (ADC) y convertir las señales digitales en análogas (DAC)
- **Puerto de comunicación:** Nos permite transmitir la data, en el microcontrolador; mediante el encapsulamiento en tramas. En los microcontroladores podemos observar los siguientes protocolos:
  - Puerto serie
  - SPI
  - I2C
  - USB
  - ETHERNET
- **Modulador de Ancho de Pulso (PWM):** Un PWM (Pulse Width Modulator), es un periférico muy útil; especialmente para el control de motores. Sin embargo, hay un grupo de aplicaciones que se pueden realizar con este dispositivo, entre las que se encuentran la inversión DC/AC de UPS, la conversión digital a analógica (D/A), el control de iluminación, la coordinación (desenfoque) entre otras aplicaciones, etc.

2.2.3. Microprocesador ESP32

Según la descripción general en (Espressif Systems, 2017) El microprocesador ESP32 es un chip de bajo coste y bajo consumo de energía que integra tecnologías Wi-Fi y Bluetooth. El ESP32, fue desarrollado por Espressif Systems; utiliza un procesador Tensilica Xtensa LX6 de doble núcleo que puede operar a 160 o 240 MHz. El ESP32 tiene varias características, que lo hacen adecuado para aplicaciones de Internet de las cosas (IoT); como domótica, control remoto, sensores y actuadores.

Componentes internos

Según la hoja técnica (Espressif Systems, 2019, págs. 3-5) El ESP32 tiene varios componentes internos que le permiten funcionar como un sistema en chip (SoC). Algunos de estos componentes son:

- **Interruptores de antena:** Permiten seleccionar la antena adecuada, para cada modo de comunicación (Wi-Fi o Bluetooth).
- **Balun de radiofrecuencia:** Convierte las señales eléctricas balanceadas, en desbalanceadas y viceversa; lo que facilita la transmisión y recepción de radiofrecuencia.
- **Amplificador de potencia:** Aumenta la potencia de la señal transmitida por el chip.
- **Amplificador receptor de bajo ruido:** Reduce el ruido de la señal recibida por el chip.
- **Filtros:** Elimina las frecuencias no deseadas de la señal de radiofrecuencia.
- **Módulos de administración de energía:** Regulan el voltaje y la corriente del chip y sus periféricos, optimizando el consumo de energía.
- **SDIO:** El ESP32 tiene dos interfaces SDIO (Secure Digital Input/Output) que pueden usarse para comunicarse con tarjetas SD o dispositivos compatibles usando un protocolo serie síncrono.

- **ADC:** El ESP32 tiene un conversor analógico a digital (ADC) de 12 bits, que puede medir hasta 18 canales con una resolución máxima de 4096 niveles; el ADC puede usarse para leer señales analógicas como voltaje o corriente.
- **DAC:** El ESP32 tiene dos conversores digitales a analógico (DAC), de 8 bits que pueden generar señales analógicas como voltaje o corriente; a partir de valores digitales.
- **Sensores táctiles:** El ESP32 tiene 10 sensores táctiles capacitivos, que pueden detectar el contacto humano; en superficies conductoras como metal o vidrio.

Unidades de procesamiento

(Espressif Systems, 2019, pág. 24) El ESP32 tiene dos unidades principales de procesamiento: la CPU y el co-procesador de ultrabaja energía (ULP).

- **CPU:** Es el núcleo principal del chip, encargado de ejecutar el código del usuario y las funciones del sistema operativo. La CPU es un procesador Tensilica Xtensa LX6 de doble núcleo que puede operar a 160 o 240 MHz y rendir hasta 600 DMIPS (millones de instrucciones por segundo).
- **Co-procesador ULP:** Es un núcleo secundario del chip, encargado de ejecutar tareas simples y repetitivas que requieren poco consumo de energía. El co-procesador ULP puede operar a 8 MHz y rendir hasta 150 DMIPS. El co-procesador ULP puede acceder a los periféricos del chip, como los sensores táctiles, el ADC, el DAC y los GPIOs, y puede despertar a la CPU cuando sea necesario.

Protocolos de comunicación

(Espressif Systems, 2019, págs. 31-32) El ESP32 soporta varios protocolos de comunicación inalámbrica y alámbrica, que le permiten interactuar con otros dispositivos y redes. Algunos de estos protocolos son:

- **Wi-Fi:** Es el protocolo más utilizado para la comunicación inalámbrica en Internet. El ESP32 soporta el estándar 802.11 b/g/n,

que opera en la banda de 2.4 GHz y puede alcanzar una velocidad máxima de 150 Mbps. El ESP32 puede funcionar como estación (STA), punto de acceso (AP) o ambos al mismo tiempo (STA+AP). El ESP32 también soporta Wi-Fi Direct (P2P), que permite la conexión directa entre dos dispositivos sin necesidad de un router.

- **Bluetooth:** Es el protocolo más utilizado para la comunicación inalámbrica entre dispositivos cercanos. El ESP32 soporta Bluetooth v4.2, que incluye los modos BR/EDR (Basic Rate/Enhanced Data Rate) y BLE (Bluetooth Low Energy). El modo BR/EDR permite una comunicación rápida y confiable entre dos dispositivos, mientras que el modo BLE permite una comunicación eficiente en energía entre varios dispositivos.

- **SPI:** Es un protocolo alámbrico que permite la comunicación sincrónica entre un dispositivo maestro y uno o varios dispositivos esclavos. El ESP32 tiene cuatro interfaces SPI que pueden funcionar como maestro o esclavo.

- **I2C:** Es un protocolo alámbrico que permite la comunicación asíncrona entre un dispositivo maestro y uno o varios dispositivos esclavos. El ESP32 tiene dos interfaces I2C que pueden funcionar como maestro o esclavo.

- **UART:** Es un protocolo alámbrico que permite la comunicación serie entre dos dispositivos. El ESP32 tiene tres interfaces UART que pueden funcionar como transmisor (TX) o receptor (RX).

- **SDIO:** Es un protocolo alámbrico, que permite la comunicación; entre un dispositivo host y una tarjeta SD o similar. El ESP32 tiene un controlador host SDIO, que puede soportar tarjetas SD/SDIO/CE-ATA/MMC/eMMC y un controlador esclavo SDIO/SPI; que puede funcionar como una tarjeta SD.

- **Ethernet:** Es un protocolo alámbrico, que permite la comunicación entre dispositivos en una red local. El ESP32 tiene una interfaz Ethernet MAC, con DMA dedicado y soporte para el protocolo

IEEE 1588 Precision Time Protocol; que permite sincronizar los relojes de los dispositivos en la red.

- **CAN:** Es un protocolo alámbrico, que permite la comunicación entre dispositivos en una red vehicular o industrial. El ESP32 tiene un controlador CAN 2.0, que puede soportar hasta 1 Mbps de velocidad.

- **Infrarrojo:** Es un protocolo inalámbrico, que permite la comunicación entre dispositivos mediante señales infrarrojas; el ESP32 tiene un controlador remoto infrarrojo (TX/RX) que puede soportar hasta 8 canales. (págs. 31 -32)

ESP32CAM

Según la hoja técnica (AI-Tinker, 2017) El ESP32CAM Ilustración 2 es un módulo, que combina el microprocesador ESP32 con una cámara OV2640 de 2 megapíxeles; que permite capturar imágenes y transmitirlas por Wi-Fi o Bluetooth.

El ESP32CAM puede alimentarse con una batería de litio de 3.7 V, tiene 16 pines GPIO que pueden usarse para diversas funciones, como entradas y salidas digitales, PWM, I2C, SPI, UART, etc.; además tiene una memoria flash de 4 MB y una memoria RAM de 520 KB.

Ilustración 2:ESP32-CAM,
Fuente:
https://www.mouser.ch/images/marketingid/2021/microsites/0/Espressif_ESP32-DevKitC-DA_comp.png

2.3. Conteiner de aplicaciones(contenedorización)

La tecnología de contenedorización, es un enfoque que permite empaquetar una aplicación y todas sus dependencias; en un entorno aislado y estandarizado llamado "contenedor". Estos contenedores son unidades autónomas que incluyen todo lo necesario, para que una aplicación se ejecute correctamente, como el código, las bibliotecas, las configuraciones y las herramientas del sistema operativo (Redhat, 2023).

Según (Hewlett Packard, 2023) La tecnología de contenedorización, se utiliza ampliamente en la industria de la tecnología y ofrece varias ventajas y desventajas:

2.3.1. Ventajas de la Tecnología de Contenedorización:

- **Portabilidad:** Los contenedores son altamente portátiles, pueden ejecutarse en cualquier entorno que admita contenedores, ya sea un servidor local, una nube pública o un clúster de orquestación como Kubernetes; que garantiza que las aplicaciones funcionen de manera coherente en diferentes entornos.
- **Aislamiento:** Los contenedores ofrecen un alto grado de aislamiento entre aplicaciones. Cada contenedor es independiente y no afecta a otros contenedores en el mismo sistema, que mejora la seguridad y la confiabilidad de las aplicaciones.
- **Eficiencia:** Los contenedores comparten recursos del sistema operativo anfitrión, lo que los hace livianos y eficientes en términos de recursos; lo que permite ejecutar más contenedores en un servidor físico que las máquinas virtuales tradicionales.
- **Rápida Implementación:** Los contenedores se inician y detienen rápidamente, lo que facilita la implementación y escalabilidad de aplicaciones; ya que los contenedores pueden adaptarse a las demandas cambiantes de tráfico.
- **Facilita la Gestión de Dependencias:** La tecnología de contenedorización, resuelve problemas de dependencias al empaquetar todas las bibliotecas y dependencias junto con la aplicación; lo que elimina conflictos de dependencias y simplifica la administración.

- **Orquestación:** Las herramientas como Kubernetes y Docker Swarm permiten orquestar y administrar fácilmente múltiples contenedores en entornos de producción complejos.

2.3.2. Desventajas de la Tecnología de Contenedorización:

- **Rendimiento Ligeramente Inferior:** Aunque los contenedores son eficientes, pueden experimentar una ligera sobrecarga; en comparación con aplicaciones nativas en el sistema operativo.
- **Dificultad con Aplicaciones Monolíticas:** Las aplicaciones monolíticas grandes y complejas, pueden ser difíciles de dividir en contenedores de manera efectiva.
- **Complejidad de Orquestación:** La orquestación de contenedores puede ser compleja, especialmente en entornos de producción a gran escala; requiere herramientas y habilidades adicionales.
- **Tamaño de Imagen:** Las imágenes de contenedores pueden volverse grandes si no se administran adecuadamente, situación que puede afectar los tiempos de descarga y almacenamiento.
- **Compatibilidad Limitada:** Algunas aplicaciones y cargas de trabajo, pueden no ser compatibles con la contenedorización; debido a restricciones específicas de hardware o sistema operativo.

En general, la tecnología de contenedorización es una herramienta poderosa para el desarrollo y la implementación de aplicaciones, pero es importante comprender sus ventajas y desventajas para utilizarla de manera efectiva en diferentes escenarios; su elección depende de las necesidades específicas del proyecto y de los recursos disponibles.

2.3.3. Docker

Con referencia en el artículo (Microsoft, 2021) Docker es una plataforma popular para desarrollar, enviar y ejecutar aplicaciones. Docker utiliza la tecnología de contenedorización para empaquetar una aplicación y sus dependencias en una unidad estandarizada llamada "contenedor". Estos contenedores se pueden

implementar fácilmente en diferentes entornos, como desarrollo, pruebas y producción, sin preocuparse por problemas de compatibilidad o diferencias en la infraestructura subyacente.

Funciones de Docker

- **Contenedor Docker:** Se trata de un paquete ligero, independiente y ejecutable que incluye todo lo necesario para ejecutar un software, incluido el código, tiempo de ejecución, herramientas del sistema, bibliotecas y configuraciones.
- **Imagen Docker:** Es una plantilla de solo lectura que se utiliza para crear contenedores Docker. Las imágenes se crean a partir de un conjunto de instrucciones llamado Dockerfile y se almacenan en un registro de Docker.
- **Registro de Docker:** Es un repositorio para imágenes de Docker, Docker Hub es un registro público popular; pero también puedes crear registros privados para las imágenes de tu organización.
- **Dockerfile:** Se refiere a un archivo de texto que contiene un conjunto de instrucciones para construir una imagen de Docker. Estas instrucciones incluyen la especificación de una imagen base, la adición de archivos, la ejecución de comandos y la configuración de ajustes.
- **Docker Compose:** Es una herramienta para definir y ejecutar aplicaciones Docker de múltiples contenedores; permite definir aplicaciones complejas con varios servicios y gestionar sus dependencias.
- **Motor de Docker:** Es el componente central de Docker, que ejecuta y gestiona contenedores en una máquina anfitriona.

Ventajas de Docker:

- **Portabilidad:** Los contenedores Docker, se ejecutan de manera uniforme en cualquier entorno que admita Docker; facilitando la migración de aplicaciones entre diferentes servidores y nubes.

- **Aislamiento:** Los contenedores ofrecen un alto grado de aislamiento entre aplicaciones, lo que significa que una aplicación y sus dependencias se ejecutan en su propio entorno; sin interferir con otras aplicaciones en el mismo sistema.

- **Eficiencia:** Los contenedores son livianos y comparten recursos del sistema operativo anfitrión, haciéndolos eficientes en términos de recursos y rápidos para iniciar y detener.

- **Escalabilidad:** Docker permite escalar aplicaciones fácilmente, agregando o eliminando contenedores según sea necesario; que facilita la administración de cargas de trabajo en constante cambio.

Casos de Uso de Docker:

- **Desarrollo de Aplicaciones:** Los desarrolladores pueden crear entornos de desarrollo reproducibles usando Docker, que garantiza que todos los miembros del equipo; trabajen en la misma configuración.

- **CI/CD:** Docker es fundamental en las prácticas de Integración Continua y Despliegue Continuo, ya que los contenedores pueden usarse para crear imágenes de aplicaciones y desplegarlas de manera consistente en diferentes etapas del proceso de desarrollo.

- **Micro servicios**: Muchas arquitecturas de micro servicios utilizan Docker para empaquetar y distribuir servicios individuales en contenedores, situación que facilita la escalabilidad y la administración.

- **Pruebas y QA:** Docker permite crear entornos de prueba aislados que coinciden con el entorno de producción, garantizando una mayor consistencia en las pruebas de aplicaciones.

- **Despliegue en la Nube:** Las aplicaciones empaquetadas en contenedores Docker, se pueden mover fácilmente a entornos de nube pública o privada.

Comandos Básicos de Docker:

Los comandos indicados fueron obtenidos de (Docker, 2022, pág. 1)

- **Docker run:** Utilizado para ejecutar un contenedor a partir de una imagen.
- **Docker build:** Crea una nueva imagen a partir de un Dockerfile.
- **Docker pull:** Descarga una imagen de un registro de Docker.
- **Docker ps:** Lista los contenedores en ejecución.
- **Docker images:** Lista las imágenes disponibles en el sistema.

2.4. Servidor Web

(Universidad de Chile, 2008) Un servidor web es un programa informático, que se encarga de procesar y enviar la información que solicitan los usuarios a través de internet; utilizando un protocolo de comunicación como el HTTP. Un servidor web puede ser tanto el software, que realiza esta función como el hardware; que almacena los datos y ejecuta el software.

2.4.1. Funciones de un servidor web:

- Escuchar las peticiones de los clientes que llegan por el protocolo HTTP o HTTPS.
- Interpretar las peticiones y determinar qué recursos o acciones se solicitan.
- Buscar los recursos solicitados en el sistema de archivos o en una base de datos.
- Ejecutar los scripts o programas necesarios para generar contenido dinámico.
- Enviar las respuestas a los clientes con el código de estado, las cabeceras y el cuerpo del mensaje.

2.4.2. Partes de un servidor web:

- El software del servidor web, es el encargado de recibir las peticiones de los clientes, buscar los archivos correspondientes y enviarlos al navegador web del cliente.
- Los archivos que contienen la información que se muestra en las páginas web, pueden ser de diferentes tipos como HTML, CSS, JavaScript, imágenes, etc.
- Las aplicaciones y bases de datos, permiten generar contenido dinámico, es decir, se actualiza según los datos del cliente o del servidor.

2.4.3. Tipos de servidores web:

Según (Brunero, 2020) los tipos de servidores web son:

- Servidores síncronos, son aquellos que responden a las peticiones de los clientes de forma secuencial, es decir, una por una. Lo que implica que el servidor debe esperar a que se complete una petición, antes de atender la siguiente; lo que puede provocar retrasos o bloqueos si hay muchas peticiones simultáneas.
- Servidores asíncronos, son aquellos que responden a las peticiones de los clientes de forma paralela, es decir, varias a la vez. Es decir, el servidor puede atender varias peticiones al mismo tiempo, sin esperar a que se complete una para pasar a la siguiente, lo que mejora el rendimiento y la eficiencia del servidor.

2.4.4. Servidor web en una tarjeta ESP32

Un servidor web puede usar una tarjeta ESP32, para ofrecer servicios web; desde un dispositivo embebido con conexión WiFi o Bluetooth. Una tarjeta ESP32 es un módulo que integra un microcontrolador, una memoria flash, una antena y otros componentes electrónicos que le permiten conectarse a Internet y ejecutar código. Algunas ventajas de usar una tarjeta ESP32 como servidor web son:

- Bajo costo: Una tarjeta ESP32 tiene un precio muy asequible, alrededor de 10 euros; lo que la hace ideal para proyectos de bajo presupuesto o prototipos.
- Bajo consumo: Una tarjeta ESP32 consume muy poca energía, que la hace adecuada para aplicaciones que funcionan con baterías o paneles solares.
- Facilidad de programación: Una tarjeta ESP32 se puede programar con diferentes lenguajes y entornos, como Arduino, MicroPython, Lua o C/C++.
- Versatilidad: Una tarjeta ESP32 se puede usar para crear diferentes tipos de servicios web, como páginas estáticas, dinámicas, API REST, MQTT, WebSocket o WebRTC.

Servidor web Asíncrono basado en ESPAsyncWebServer

Usando como referencia la hoja técnica de la librería (Kravets, 2022) Un servidor web asíncrono es un tipo de servidor web, que utiliza un enfoque de programación asíncrona; para manejar varias peticiones de forma simultánea. En lugar de crear un hilo o un proceso para manejar cada petición, un servidor web asíncrono utiliza una técnica llamada "multiplexación de eventos"; para manejar varias peticiones de forma simultánea utilizando un solo hilo o proceso. Esto permite al servidor web manejar un gran número de peticiones, sin consumir demasiados recursos del sistema.

La librería ESPAsyncWebServer, utiliza esta técnica para crear servidores web asíncronos en el ESP32; que permite al dispositivo manejar varias peticiones de forma eficiente y escalable. Algunas de las características interesantes de esta librería incluyen:

- Manejo de peticiones y respuestas en formato HTTP: ESPAsyncWebServer, proporciona una interfaz fácil de usar para crear y manejar peticiones y respuestas HTTP.
- Manejo de rutas: La librería permite definir diferentes rutas; para manejar diferentes peticiones. Por ejemplo, se pueden definir diferentes rutas para manejar peticiones GET, POST y otros tipos de peticiones.

- Interfaz para manejar parámetros de petición: La librería permite fácilmente acceder y manejar parámetros de petición en una petición HTTP.
- Interfaz para manejar archivos subidos: La librería proporciona una interfaz para manejar archivos subidos en una petición HTTP.
- Interfaz para manejar autenticación: ESPAsyncWebServer permite fácilmente definir y manejar la autenticación básica en un servidor web.
- Interfaz para manejar WebSockets: La librería proporciona una interfaz para manejar comunicaciones a través de WebSockets.

La librería ESPAsyncWebServer proporciona una interfaz de programación de aplicaciones (API), para crear servidores web asíncronos en el ESP32; permitiendo manejar varias peticiones de forma eficiente y escalable mediante el uso de técnicas de multiplexación de eventos.

Además, proporciona una serie de características interesantes para manejar diferentes aspectos de un servidor web, como rutas, parámetros de petición, archivos subidos, autenticación y comunicación a través de WebSockets.

En conclusión, un servidor web, es un programa informático que permite ofrecer contenido web a los clientes que lo solicitan. Un servidor web puede estar compuesto por varias partes y existen diferentes tipos de servidores web según sus características, pueden usar una tarjeta ESP32 para crear servicios web desde un dispositivo embebido con conexión inalámbrica.

2.5. Internet de las Cosas (IOT)

IOT es el acrónimo de Internet de las cosas, que se refiere a la interconexión de objetos físicos con la red mediante sensores, actuadores y otros dispositivos. El objetivo de IOT es facilitar la comunicación, el control y la automatización de los procesos y las actividades cotidianas (AWS, s.f.). En este trabajo se presentan los conceptos básicos de IOT, sus partes principales, algunos ejemplos con ESP32 y su uso en clasificación de imágenes.

2.5.1. Conceptos básicos de IOT

IOT se basa en tres conceptos fundamentales: la identificación, la comunicación y la inteligencia.

- La identificación consiste en asignar un identificador único a cada objeto que se quiere conectar a la red, como por ejemplo un código QR, un código de barras o una etiqueta RFID.
- La comunicación se refiere al intercambio de datos entre los objetos y la red, o entre los propios objetos, mediante protocolos como Wi-Fi, Bluetooth, ZigBee o LoRa.
- La inteligencia se refiere a la capacidad de procesar y analizar los datos recogidos por los sensores, y de tomar decisiones o acciones en función de ellos, mediante algoritmos o aplicaciones. (Quiñonez Muñoz, 2019)

2.5.2. Partes de IOT

Un sistema IOT se compone de cuatro partes principales: los dispositivos, la red, la plataforma y las aplicaciones.

- Los dispositivos son los objetos físicos que se conectan a la red mediante sensores y actuadores.
    - Los sensores son capaces de medir variables físicas como la temperatura, la humedad, el movimiento o la luz.
    - Los actuadores son capaces de modificar el entorno físico mediante acciones como encender una luz, abrir una puerta o activar una alarma.
- La red es el medio que permite la comunicación entre los dispositivos y la plataforma, puede ser una red local o una red global como Internet.
- La plataforma es el software que gestiona los datos y los servicios de los dispositivos, puede ser una plataforma propia o una plataforma en la nube como AWS, Google Cloud o Azure.

- Las aplicaciones son las interfaces que permiten al usuario interactuar con el sistema IOT, éstas pueden ser aplicaciones web, móviles o de escritorio. (Quiñonez Muñoz, 2019)

2.6. Machine Learning

En los siguientes párrafos citaremos a dos autores que explican de manera básica y fundamental el Machine Learning.

“La definición "sin estar programada explícitamente" se atribuye a Arthur Samuel, quien acuñó el término "aprendizaje automático"” en 1959.

Michie, D., Spiegelhalter “Sería útil que las computadoras pudieran aprender de la experiencia y así mejorar automáticamente la eficiencia de sus propios programas durante la ejecución. Se puede proporcionar una facilidad de aprendizaje de memoria simple pero eficaz en el marco de una programación adecuada”

Es el conjunto de algoritmos pertenecientes a la rama de Inteligencia Artificial, que permite a los algoritmos de aprendizaje automático construir un modelo basado en datos de muestra, conocidos como "datos de entrenamiento"; con el fin de hacer predicciones o decisiones sin estar programados explícitamente para hacerlo. El algoritmo tiene la capacidad de evolucionar conforme se recopile más datos cuando esté en uso, lo cual generalmente mejora las capacidades del algoritmo en sí.

2.6.1. Regresión Lineal:

Una gradiente se define como un vector compuesto por las derivadas parciales de una función. Este vector, nos proporciona información acerca de la dirección y la magnitud de cambio en dicha función; permitiéndonos determinar qué direcciones nos acercan o alejan de su mínimo valor.

La distancia entre los puntos sucesivos que el algoritmo recorre, se conoce como pasos. Es importante destacar que esta distancia es ajustable por parte del usuario, lo que permite controlar la velocidad y la precisión del proceso de optimización.

La ecuación para el modelo sería:

$$Y=b+w1X1$$

- Y es la etiqueta (El valor a Predecir).

- b es la ordenada de origen (Valor de Y donde X=0).
- w es la ponderación (Sería la pendiente).
- X es el atributo (Una entrada conocida). (Google, 2022)

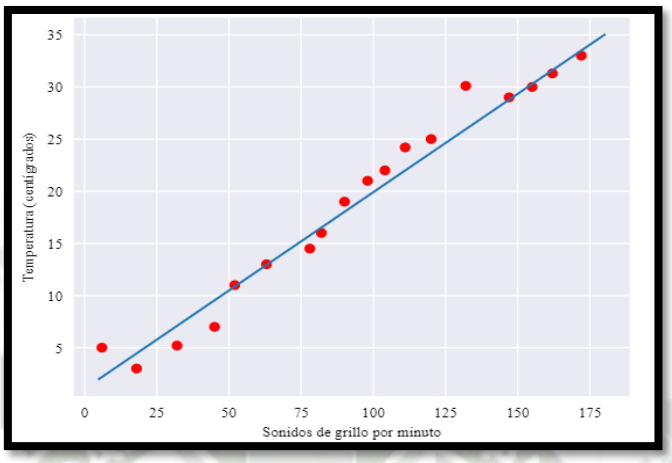


Ilustración 3: Descending into ML: Linear Regression | Machine Learning Crash Course, Fuente: https://developers.google.com/machine-learning/crash-course/descending-into-ml/linear-regression

2.6.2. Entrenamiento y pérdida

Entrenar un modelo, significa determinar los valores correctos; para todos los pesos e intersecciones de muestras etiquetadas. En el aprendizaje supervisado, un algoritmo de aprendizaje automático; construye un modelo observando varios ejemplos e intentando encontrar un patrón que minimice la pérdida.

El valor de pérdida, es un número que es el resultado de la diferencia entre el valor predicho y el valor real; calculado cuando probamos el algoritmo usando valores de PRUEBA. Si la predicción del modelo es perfecta, entonces la pérdida es cero; de lo contrario la pérdida será mayor. Figura 4

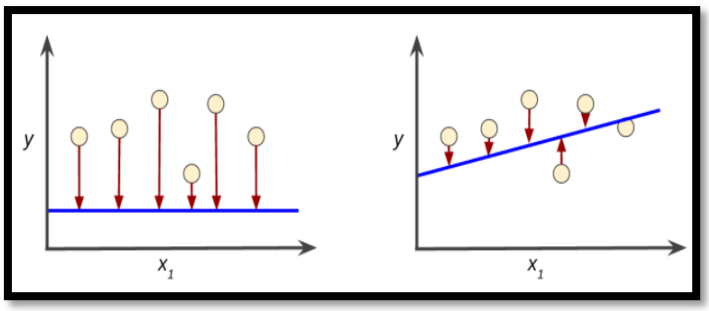


Ilustración 4: Modelo de alta pérdida y modelo de baja pérdida, Fuente: https://developers.google.com/machine-learning/crash-course/descending-into-ml/linear-regression

- Pérdida al Cuadrado: Debido a que el objetivo del machine learning, es obtener un valor predicho "Y" con una tasa de acierto alta; es que para medir la cantidad el error se aplica la siguiente ecuación

$$\text{Pérdida al Cuadrado} = (Y - Y)^2$$

Y es el valor observado

Y' es el predecido

- Error Cuadrático Medio: Es el valor promedio, calculado de los resultados de pérdida al cuadrado de cada valor TEST y si este valor es muy alto nos indican que los valores de ponderación o características del modelo están mal elegidos; o el modelo no es compatible con el problema

$$\text{ECM} = \frac{1}{\text{N}} \sum_{(X,Y)\epsilon D} (Pérdida\ al\ cuadrado) \rightarrow ECM = \frac{1}{N} \sum_{(X,Y)\epsilon D} (Y - Y')^2$$

2.6.3. Reducción de Pérdidas

Descenso de gradientes

Un gradiente es un vector con derivadas parciales; indica si está más cerca o más lejos. El algoritmo de descenso de gradiente, calcula la pendiente de la curva de pérdida desde el punto inicial e intenta estimar el valor de pérdida mínimo; para determinar el peso "w". La distancia entre puntos, se llama paso y este valor lo controla el usuario Ilustración 5. (Google, 2023)

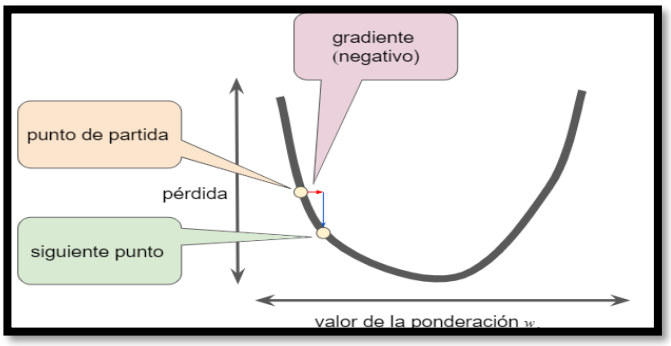


Ilustración 5: Descending into ML: Linear Regression,
Fuente: https://developers.google.com/machine-learning/crash-course/descending-into-ml/linear-regression

Tasas de Aprendizaje

Este es el tamaño del paso que se toma al ajustar el algoritmo de descenso de gradiente Ilustración 6: Descending into ML: Linear Regression,. Si elegimos una tasa de aprendizaje muy baja, el aprendizaje llevará mucho tiempo, pero si elegimos una tasa de aprendizaje muy alta, es posible que nos equivoquemos o nos acerquemos en el valor de pérdida mínimo; por lo que el objetivo sería encontrar la tasa de aprendizaje óptima para alcanzar el valor mínimo. Esta variable es un hiperparámetro, que es una variable que el programador define para ajustar el algoritmo de aprendizaje automático. (Google, 2023)

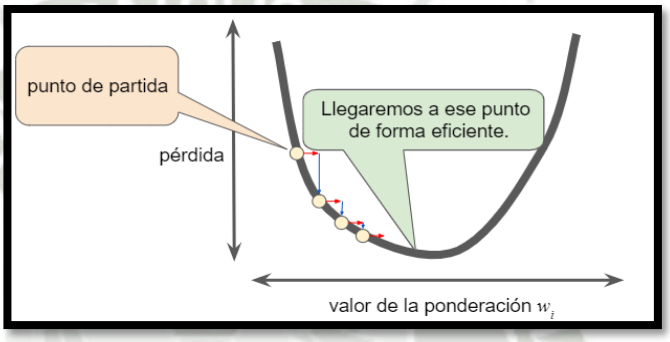


Ilustración 6: Descending into ML: Linear Regression,
Fuente: https://developers.google.com/machine-learning/crash-course/descending-into-ml/linear-regression

2.6.4. Tipos de algoritmos

Algoritmos de regresión:

En las tareas de regresión, el programa de aprendizaje automático debe estimar y comprender las relaciones entre las variables. El análisis de regresión, se enfoca en una variable dependiente y una serie de otras variables cambiantes; lo que lo hace particularmente útil para la predicción y el pronóstico.

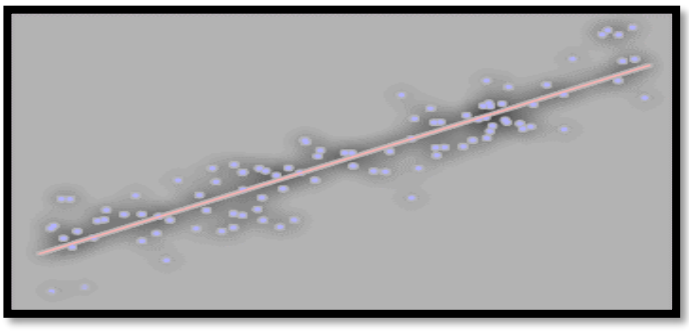


Ilustración 7: Análisis De Agrupamiento K-Medias,
Fuente: https://www.pngwing.com/es/free-png-ds

Algoritmos bayesianos

Este tipo de algoritmos, por clasificación están basados en el teorema de Bayes y clasifican cada valor; como independiente de cualquier otro.

Esto permite predecir una clase o categoría, en función de un conjunto dado de características; utilizando la probabilidad. A pesar de su simplicidad, el clasificador funciona sorprendentemente bien y se usa a menudo porque supera a los métodos de clasificación más sofisticados.

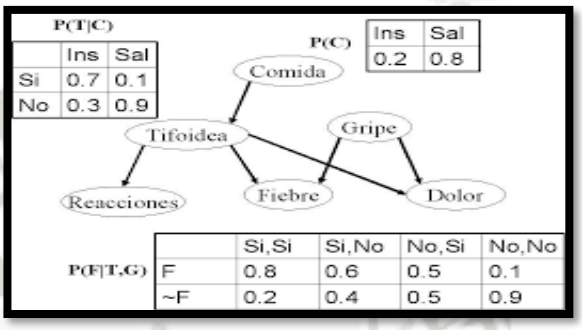


Ilustración 8: Árbol de decisiones,
Fuente: https://ccc.inaoep.mx/~esucar/Clases-mgp/caprb.pdf

Algoritmos de agrupación

Se utilizan en el aprendizaje no supervisado, y sirven para categorizar datos no etiquetados, es decir; datos sin categorías o grupos definidos. El algoritmo, funciona mediante la búsqueda de grupos dentro de los datos; con el número de grupos representados por la variable K. A continuación, funciona de manera iterativa; para asignar cada punto de datos a uno de los K grupos según las características proporcionadas.

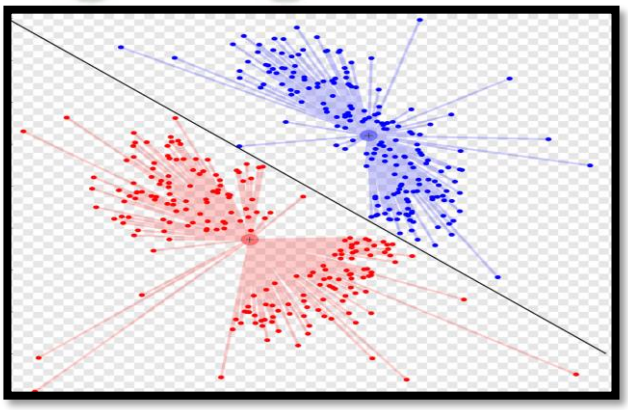


Ilustración 9: Visualización De Datos,
Fuente: https://www.pngwing.com/es/free-png-ds

Algoritmos de árbol de decisión

Un árbol de decisión, es una estructura de árbol similar a un diagrama de flujo; que utiliza un método de bifurcación para ilustrar cada resultado posible de una decisión. Cada nodo dentro del árbol, representa una prueba en una variable específica; y cada rama es el resultado de esa prueba.

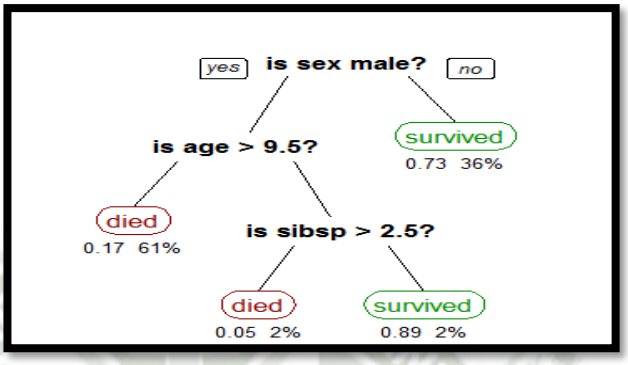


Ilustración 10: Árbol de Decisión,
Fuente:https://es.wikipedia.org/wiki/Aprendizaje_basado_en_%C3%A1rboles_de_decisi%C3%B3n#/media/Archivo:CART_tree_titanic_survivors.png

Algoritmos de redes neuronales

Una red neuronal artificial (RNA), comprende unidades dispuestas en una serie de capas; cada una de las cuales se conecta a las capas anexas. Las RNA se inspiran en los sistemas biológicos como el cerebro, y en cómo procesan la información, por lo que son esencialmente un gran número de elementos de procesamiento interconectados; que trabajan al unísono para resolver problemas específicos.

También aprenden con el ejemplo y la experiencia, y son extremadamente útiles para modelar relaciones no lineales en datos de alta dimensión; o donde la relación entre las variables de entrada es difícil de entender.

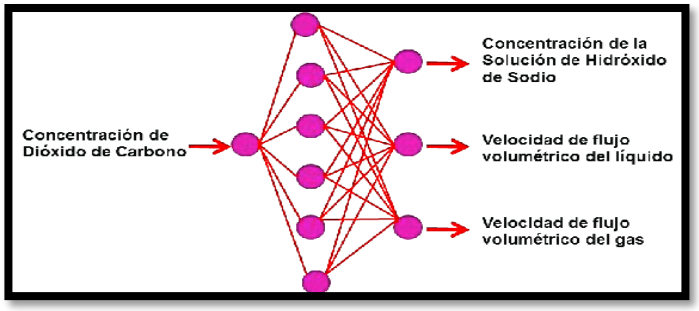


Ilustración 11: Redes Neuronales ,
Fuente: https://www.pngwing.com/es/free-png-d

2.6.5. Red neuronal convolucional

Una red neuronal convolucional (CNN), es un tipo de red neuronal artificial diseñada para procesar y analizar imágenes; utiliza una técnica llamada "convolución" para analizar patrones en las imágenes, lo que permite a la red aprender características específicas de las imágenes, como bordes, texturas y objetos. Estas características son aprendidas a través de capas de "convolución" y "submuestreo" (también conocido como "agrupamiento" o "pooling").

En la primera capa de la CNN, se utilizan filtros para analizar la imagen y extraer características, estos filtros son matrices pequeñas que se deslizan a lo largo de la imagen, y para cada posición; se calcula la suma ponderada de los píxeles que están debajo del filtro. Esto se conoce como una operación de convolución, y ayuda a la red a detectar patrones específicos en la imagen, como bordes y texturas.

Después de la capa de convolución, la red utiliza una capa de "submuestreo" o "agrupamiento", que se utiliza para reducir la dimensionalidad de la representación de la imagen. Esto se logra mediante el uso de una función de agrupamiento, como el promedio o el valor máximo; para combinar los valores de varias neuronas en una sola celda.

En las capas subsiguientes, la red aprende características cada vez más complejas; mediante la combinación de las características aprendidas en las capas anteriores. La última capa de la red se conecta a una capa de salida, donde se realiza la clasificación final.

CNNs son muy utilizadas en tareas de visión artificial como la clasificación de imágenes, reconocimiento de objetos, detección de objetos, segmentación de imágenes, análisis de vídeo, entre otras. Estas redes son muy eficientes en tareas que requieren entender la estructura espacial de las imágenes, y en general; las redes CNNs son las que tienen mejor desempeño en tareas de reconocimiento de imágenes. (Spiegato, 2023)

Entrenamiento y pruebas de una red neuronal convolucional

Las redes neuronales convolucionales (CNN), se entrenan y se prueban utilizando un conjunto de datos específico; el proceso de entrenamiento se divide en dos partes: el entrenamiento y la validación.

El entrenamiento, se realiza utilizando un conjunto de datos llamado conjunto de entrenamiento; que consta de un gran número de ejemplos de imágenes etiquetadas. La red neuronal se presenta con una imagen y su etiqueta correspondiente, y a través de un proceso de ajuste de pesos; la red aprende a asociar características específicas de la imagen con su etiqueta correspondiente.

La validación, se realiza utilizando otro conjunto de datos llamado conjunto de validación; que también consta de ejemplos de imágenes etiquetadas. La idea es, evaluar el rendimiento de la red en datos que no se utilizaron en el entrenamiento; con el objetivo de detectar overfitting.

Una vez entrenada la red, se utiliza el conjunto de prueba para evaluar el rendimiento de la red en datos que nunca ha visto antes. El rendimiento se mide utilizando una métrica como la precisión, la exhaustividad o el puntaje F1, entre otras.

En resumen, el proceso para entrenar y probar una CNN es el siguiente:

- Se divide el conjunto de datos en tres: entrenamiento, validación y prueba
- Se entrena la red utilizando el conjunto de entrenamiento
- Se valida el rendimiento de la red utilizando el conjunto de validación
- Se evalúa el rendimiento final de la red en el conjunto de prueba
- Si se desea mejorar el rendimiento se ajustan los hiperparámetro o se vuelve al paso 2.

2.6.6. Red neuronal recurrente

Las redes neuronales recurrentes (RNN), son usadas para clasificar y analizar datos; basadas en secuencia como audio, texto, vídeo, etc. A diferencia de las redes neuronales convencionales, las RNN tienen conexiones entre las unidades ocultas que forman un ciclo; lo que les permite mantener una memoria de los estados anteriores. Esta situación permite la captura las dependencias temporales y contextuales de los datos.

Para entrenar una RNN, se utiliza el algoritmo de retro propagación a través del tiempo (BPTT), que consiste en desenrollar la red a lo largo de la secuencia de entrada y aplicar la regla de la cadena para calcular los gradientes de los parámetros BPTT, usada para entrenar redes neuronales convencionales; pero adaptado al caso de las secuencias. El objetivo es minimizar una función de pérdida que mide el error entre la salida esperada y la salida real de la red para cada elemento de la secuencia. (IBM)

Estructura:

- **Capa de entrada (Input Layer):** Esta capa recibe la secuencia de entrada, que puede ser una secuencia de vectores; como palabras en un texto o puntos en una serie de tiempo.
- **Capa oculta (Hidden Layer**): La capa oculta, es donde reside la capacidad de las RNN; para mantener información sobre estados anteriores. Cada unidad en esta capa tiene conexiones recurrentes consigo misma, lo que permite que la información fluya desde el paso de tiempo anterior al siguiente. Cada unidad realiza una operación, que combina la entrada actual con la salida de la capa oculta en el paso de tiempo anterior; lo que crea una especie de "memoria" temporal.
- **Capa de salida (Output Layer):** La capa de salida puede ser una capa completamente conectada que produce la salida deseada, que puede ser una clasificación, una predicción en una serie de tiempo o cualquier otro tipo de salida requerida por la tarea.

Arquitecturas variantes de las Redes neuronales recurrentes

Redes Neuronales Recurrentes (RNN):

- Las RNN son un tipo de red neuronal diseñada para manejar datos secuenciales y establecer conexiones temporales en los datos.
- Utilizan conexiones recurrentes que permiten que la información fluya de un paso de tiempo a otro, lo que las hace adecuadas para tareas que implican secuencias de datos; como series temporales y procesamiento de lenguaje natural.
- Sin embargo, las RNN tradicionales pueden sufrir de problemas de desvanecimiento y explosión del gradiente; que dificulta el entrenamiento efectivo en secuencias largas.

Redes LSTM (Long Short-Term Memory):

- Las redes LSTM son una mejora de las RNN tradicionales, que abordan el problema de los gradientes; que desaparecen o explotan.
- Introducen unidades de memoria llamadas "celdas LSTM" que pueden almacenar y recuperar información durante largas secuencias de datos.
- Utilizan mecanismos de puerta, como puertas de olvido, puertas de entrada y puertas de salida, para controlar el flujo de información a través de la celda y regular la memoria a largo y corto plazo.
- Las LSTM son especialmente útiles para tareas que involucran dependencias a largo plazo en los datos, como traducción automática o generación de texto coherente.

Redes GRU (Gated Recurrent Unit):

- Las redes GRU son otra mejora de las RNN tradicionales que también abordan el problema de los gradientes que desaparecen o explotan.

- Son similares a las LSTM en términos de su capacidad para mantener información a largo plazo y corto plazo, pero utilizan una arquitectura más simple con menos puertas.
- Las GRU tienen una puerta de actualización y una puerta de reinicio, que controlan el flujo de información a través de la unidad; lo que las hace computacionalmente más eficientes que las LSTM.
- A menudo las GRU, se utilizan en tareas de procesamiento de secuencias cuando se busca un equilibrio entre el rendimiento y la eficiencia computacional.

BRNN (Redes Neuronales Bidireccionales):

- Capturan información, tanto de los pasos de tiempo anteriores como de los futuros en una secuencia de datos.
- Utilizan dos capas ocultas, una procesa la secuencia de izquierda a derecha y otra de derecha a izquierda.
- Beneficiosas para tareas que requieren comprender el contexto en ambas direcciones, como el procesamiento de lenguaje natural.

2.6.7. Red neuronal convolucional 3D (3D-CNN)

La red neuronal convolucional 3D (3D-CNN), es una versión de una red neuronal convolucional diseñada específicamente para trabajar con datos tridimensionales; como vídeos o volúmenes médicos.

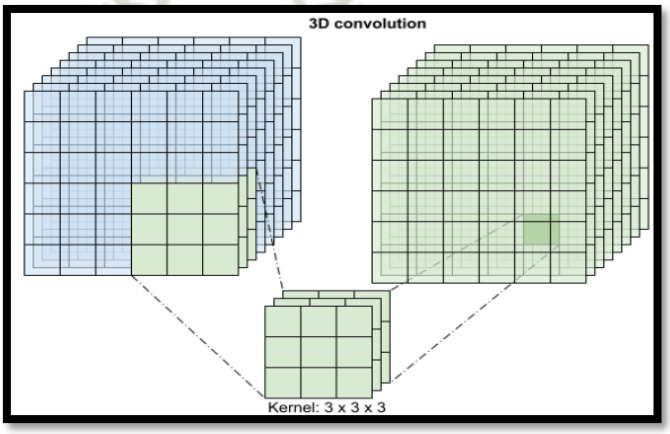


Ilustración 12: 3D convolutional neuronal network,
Fuente: https://www.tensorflow.org/tutorials/vídeo/vídeo_classification#visualize_the_results

Estas redes se utilizan para tareas de procesamiento de vídeo, análisis de imágenes 3D y otras aplicaciones que involucran datos volumétricos.

3D-CNN es una extensión de la CNN tradicional, que se utiliza principalmente para procesar imágenes 2D como instantáneas.

La principal diferencia entre CNN-3D y CNN 2D es la capacidad de trabajar con datos tridimensionales, significa que en lugar de trabajar únicamente con la altura y el ancho (como en las imágenes 2D), 3D-CNN; también tiene en cuenta la profundidad o el tiempo. Por lo que son ideales para tareas que involucran secuencias de imágenes en movimiento, como vídeos.

Algunas aplicaciones populares de CNN-3D incluyen: Análisis de vídeo:3D-CNN se puede utilizar para tareas como reconocimiento de objetos en movimiento, seguimiento de objetos por vídeo, detección de actividad por vídeo de vigilancia y más.

Asimismo, en tratamientos médicos se tienen aplicaciones en 3D-CNN que se utilizan para analizar datos de imágenes 3D, como resonancias magnéticas, tomografías computarizadas y ultrasonidos para diagnóstico y planificación de tratamientos.

Por otro lado, en la reconstrucción y modelado 3D:3D-CNN, también se utiliza en la reconstrucción de objetos tridimensionales a partir de secuencias de imágenes 2D; o para tareas de simulación y modelado tridimensional o para Vídeo juegos y animación.

Finalmente, en la industria del entretenimiento, 3D-CNN se utiliza para mejorar la calidad de los gráficos en videojuegos y dibujos animados; así como para detectar y rastrear objetos en entornos 3D.

Similar a 2D-CNN, 3D-CNN consta de capas convolucionales, que se utilizan para extraer características relacionadas de datos tridimensionales; seguidas de capas de agrupamiento y capas completamente conectadas para realizar tareas específicas de clasificación o regresión. Sin embargo, debido a la complejidad adicional de trabajar con datos 3D, 3D-CNN a menudo requiere una mayor potencia informática y grandes cantidades de datos de entrenamiento para lograr buenos resultados.

Estructura

- **Capa de Entrada (Input Layer):** Esta capa acepta datos tridimensionales, como un vídeo o una secuencia de imágenes 3D. La entrada puede tener tres dimensiones, por ejemplo, ancho, alto y profundidad (que podría ser el número de cuadros en un vídeo).
- **Capas de Convolución 3D (3D Convolutional Layers):** Estas capas son el núcleo de una CNN-3D, aplican filtros tridimensionales a la entrada para extraer características relevantes. Cada filtro se desliza a través de la entrada en las tres dimensiones (ancho, alto y profundidad) y produce mapas de características; el número de filtros determina la profundidad de la capa.
- **Capas de Activación (Activation Layers):** Después de cada capa de convolución, generalmente se aplica una función de activación, como ReLU (Rectified Linear Unit); para introducir no linealidad en la red y permitir que la red aprenda relaciones más complejas en los datos.
- **Capas de Pooling 3D (3D Pooling Layers):** Estas capas reducen la dimensionalidad de las características obtenidas mediante la convolución. Al igual que en las CNN 2D, las capas de pooling 3D reducen el tamaño de los mapas de características; lo que reduce la cantidad de parámetros en la red y ayuda a prevenir el sobreajuste.
- **Capas Totalmente Conectadas (Fully Connected Layers):** Después de una serie de capas de convolución y pooling, se suelen agregar una o varias capas totalmente conectadas. Estas capas transforman las características extraídas, en una representación adecuada para la tarea final; como clasificación o regresión.
- **Capa de Salida (Output Layer):** La capa de salida produce la predicción final de la red, la cantidad de neuronas en esta capa depende de la naturaleza de la tarea. Por ejemplo, en una tarea de clasificación; puede haber una neurona por clase con una función de activación softmax.

- **Conexiones Residuales (Residual Connections):** En algunas arquitecturas de CNN-3D más avanzadas, se utilizan conexiones residuales; para ayudar en el entrenamiento de redes más profundas. Estas conexiones permiten que la información fluya directamente a través de las capas sin ser alterada, lo que puede facilitar el aprendizaje profundo.

2.6.8. Modelo de Aprendizaje Profundo ResNET

ResNet es una arquitectura de aprendizaje profundo diseñada para entrenar redes neuronales convolucionales (CNN) desarrollada por Kaiming He, Xiangyu Zhang, Shaoqing Ren y Jian Sun en 2015.

ResNet representa un avance importante en esta área, sobre visión por computadora y reconocimiento de imágenes; ya que resuelve eficazmente el problema de la formación de redes neuronales muy profundas.

La innovación clave de ResNet es el uso de bloques residuales, diseñados para facilitar la formación de redes ultra profundas; estos bloques residuales crean una conexión interrumpida o una conexión acortada que omite una o más capas. Esto permite que la red aprenda funciones residuales, es decir, que tenemos una capa de entrada X y queremos calcular la salida F(X) de una capa en la red. En lugar de aprender una función completa F(X), ResNet aprende una función residual R(X), que representa la diferencia entre la salida deseada y la entrada actual.

Matemáticamente, esto se expresa como:

$$F(x) = R(x) + X$$

Donde:

- F(X) es la salida deseada de la capa.
- X es la entrada actual.
- R(X) es la función residual que se debe aprender.

Esta conexión residual alivia el problema de la supresión de gradiente, facilitando así la formación de redes muy profundas, las redes más profundas pueden aprender

funciones más complejas; lo cual es importante para tareas como clasificación de imágenes, detección de objetos y segmentación de imágenes.

La arquitectura ResNet viene en varias profundidades, comúnmente conocidas como ResNet-18, ResNet-34, ResNet-50, ResNet-101 y ResNet-152, entre otras; el número indica el número total de capas en la red. Por ejemplo, ResNet-50 tiene 50 clases.

ResNet se ha utilizado y adaptado ampliamente para diversas tareas de visión por computadora, como clasificación de imágenes, detección de objetos, segmentación semántica, etc.; ha producido constantemente resultados superiores en pruebas comparativas como ImageNet y se ha convertido en la arquitectura fundamental del aprendizaje profundo de visión por computadora. (geeksforgeek, 2023)

2.6.9. Modelos de Machine Learning considerados para la Clasificación de Acciones Delictivas

Para la clasificación de imágenes delictivas y no delictivas, se utilizará un conjunto de datos etiquetados; que contengan imágenes de ambos tipos. Durante el entrenamiento, el modelo aprenderá a distinguir entre las dos categorías y luego; se puede evaluar su rendimiento en un conjunto de datos de prueba para medir su precisión.

Modelo Secuencial Denso

Un modelo secuencial denso (Dense Sequential Model) Ilustración 13, es un tipo de modelo de red neuronal artificial; que se utiliza en el aprendizaje profundo (deep learning). Es un modelo básico y fundamental, que consta de una secuencia lineal de capas densas (también llamadas capas completamente conectadas); cada capa se conecta a la capa anterior y posterior que permitirá que la información fluya a través del modelo.

En una capa densa, cada nodo (neurona) está conectado a todos los nodos de la capa anterior y posterior. Cada nodo realiza una operación matemática simple en su entrada, y la salida se pasa a todos los nodos de la siguiente capa.

En el modelo secuencial denso, los nodos en cada capa están organizados en una sola dimensión; y la salida de una capa se alimenta como entrada a la siguiente capa que hace que la red sea secuencial.

La funcionalidad principal de los modelos secuenciales densos es la clasificación de imágenes, los modelos reciben como entrada una imagen (representada por una matriz de píxeles) y la procesan a través de una serie de capas de neuronas; para producir una salida que es la probabilidad de que la imagen pertenezca a cada una de las clases posibles. El modelo aprende a clasificar las imágenes, a partir de un conjunto de datos de entrenamiento previamente etiquetados; donde cada imagen está asociada a una etiqueta de clase conocida.

Durante el entrenamiento, el modelo ajusta los pesos y los sesgos de las neuronas en cada capa para minimizar la función de pérdida, que mide la diferencia entre las etiquetas de clase conocidas y las predicciones del modelo; para cada imagen en el conjunto de entrenamiento. El proceso de entrenamiento utiliza un algoritmo de optimización como el descenso del gradiente, para ajustar los pesos y sesgos de las neuronas; en cada capa para minimizar la función de pérdida.

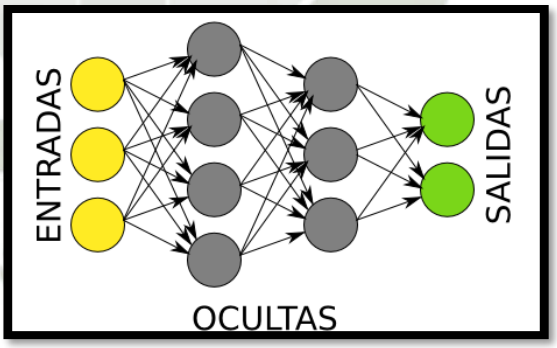


Ilustración 13: Modelo Secuencial Denso ,
Fuente: https://rubenlopezg.wordpress.com/2014/05/07/que-es-y-como-funciona-deep-learning/

Modelo Red Neuronal Convolucional

Un modelo de Red Neuronal Convolucional (Convolutional Neural Network, CNN en inglés) Ilustración 14, es una arquitectura de red neuronal profunda; diseñada para procesar datos de entrada con una estructura de cuadrícula como imágenes.

Este tipo de modelo, se compone de capas de convolución y capas de agrupamiento (pooling); seguidas de una o varias capas densas.

Las capas de convolución aplican un conjunto de filtros convolucionales a la entrada, que detectan características específicas de la imagen; como bordes o esquinas. Las capas de agrupamiento, reducen el tamaño de la imagen; al agrupar varias características cercanas en una sola. Finalmente, las capas densas procesan la información resultante; para generar la salida deseada.

Para implementar un modelo secuencial convolucional en Python, se puede utilizar la librería Keras, que es una API de alto nivel para construir y entrenar modelos de aprendizaje profundo. Keras se puede ejecutar en Google Colab, una plataforma de computación en la nube que permite escribir y ejecutar código en Python.

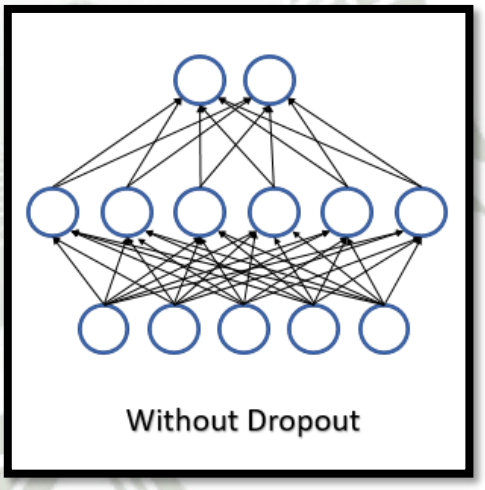


Ilustración 14: CNN sin dropout,
Fuente: https://c.mql5.com/2/42/2264830929419.png

Modelo de Aprendizaje Profundo MoViNet

MoViNet Ilustración 15 representa un avanzado modelo de aprendizaje profundo, diseñado para el reconocimiento de acciones en vídeos; aprovechando las redes neuronales convolucionales 3D.

Una extensión de sus contrapartes 2D de las redes, aplican filtros a los vóxeles de los vídeos; unidades tridimensionales que contienen información espacial y temporal.

Un "vóxel" es la contracción de "volumetric pixel" (píxel volumétrico), y a diferencia de los píxeles en imágenes 2D, los vóxeles representan unidades tridimensionales en el espacio y el tiempo, lo que significa que contienen información no solo sobre la ubicación en el cuadro; sino también sobre cómo cambian con el tiempo.

Este enfoque permite a MoViNet capturar con precisión tanto la estructura espacial, como la dinámica temporal de los contenidos audiovisuales.

La arquitectura de MoViNet se fundamenta en bloques residuales móviles, una variante que disminuye la cantidad de parámetros y la carga computacional. MoViNet se puede entrenar de manera supervisada o auto-supervisada, y se adapta sin problemas a diversas resoluciones y tasas de fotogramas, lo que lo convierte en una herramienta versátil.

Este modelo demuestra una notable capacidad para clasificar acciones con elevada precisión y eficiencia, sus aplicaciones abarcan diversos campos, desde el análisis deportivo hasta la vigilancia; la interacción humano-máquina y la educación.

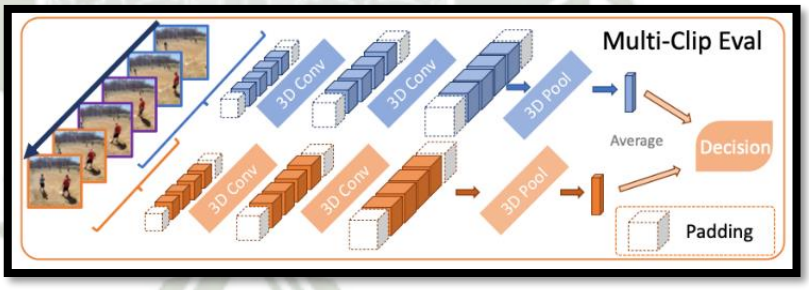


Ilustración 15: MoviNet Red para clasificación en Streaming,
Fuente: https://arxiv.org/pdf/2103.11511.pdf

CAPÍTULO III. DISEÑO DEL SISTEMA

1. Esquema Conceptual

El trabajo de investigación tiene como objetivo diseñar e implementar un prototipo de sistema IOT, para la identificación de actos delictivos y ejecución de acciones disuasorias, en un entorno de casa u hogar mediante el uso de la tarjeta ESP32-Cam, servidor web asíncrono y algoritmos de aprendizaje automático.

Para ello se dividirá el desarrollo en tres secciones:

- **Sección de software:** En esta sección del proyecto, se lleva a cabo la programación de los distintos aplicativos y dispositivos:
  - La placa ESP-32CAM, que es un microcontrolador de bajo costo y bajo consumo de energía; con una gran cantidad de recursos para conectividad y procesamiento.

    Para programar la placa ESP-32CAM, se utilizará ArduinoIDE, una plataforma de desarrollo integrado de código abierto para microcontroladores.

    El objetivo es que el ESP32-CAM actúe como una cámara y envíe las imágenes al servidor de TensorFlow Serving, para recibir la clasificación correspondiente, con base en esta información se decidirá si se ejecutarán medidas disuasorias o no.
  - El servidor web será montado por Live Server, un complemento de Visual Studio Code que permitirá cargar una página HTML y JavaScript en el navegador.

    Esta página será el medio de control y mostrará los resultados, sirviendo como la interfaz gráfica intuitiva; para el manejo del prototipo. La codificación de la página HTML se realizó utilizando Visual Studio Code, un editor de código fuente de código abierto; que permite trabajar con varios lenguajes de programación.
- **Sección de entrenamiento y uso de algoritmos de aprendizaje automático:** En esta sección, se analizarán diferentes tipos de modelos de Machine Learning para el entrenamiento y uso en el proyecto; se considerará el uso de una red ya entrenada y de modelos de creación propia:

- **MoViNet:** Es una arquitectura de red neuronal convolucional 3D y redes residuales ligera desarrollada por Google para dispositivos móviles y computadoras de baja potencia. Esta red neuronal, de acceso gratuito, será utilizada como núcleo del nuevo modelo. Para ello, se usará la columna vertebral del modelo para que se adapte para clasificar dos acciones.

  MoViNet utiliza una técnica llamada depthwise separable convolution, para reducir significativamente el número de parámetros y operaciones necesarias en el entrenamiento e inferencia; que permite un mejor rendimiento en dispositivos con recursos limitados.
- **Arquitectura de modelos propios:** Se desarrollarán dos arquitecturas de modelos para la clasificación de imágenes, con el objetivo de analizar y encontrar el mejor modelo para la detección de acciones delictivas; para ello se utilizará una arquitectura de modelo Secuencial y una arquitectura Convolucional.

Para el entrenamiento de estas redes, se usarán algunos datasets de acceso gratuito de la página https://dasci.es/es/transferencia/open-data/deteccion-de-armas/; uno de los datasets utilizados será "Detección de pistolas".

El conjunto de datos Pistol detection, contiene 3000 imágenes de armas de fuego cortas con un fondo rico en contexto. Estas imágenes seleccionadas de internet, contienen una o más pistolas en situaciones diversas; incluyendo contextos de videovigilancia.

Otro de los datasets será "Detección de armas blancas", el conjunto de datos Knife detection contiene 2078 imágenes; donde al menos aparece un cuchillo. Estas imágenes seleccionadas fueron descargadas de internet y algunas pertenecen a fotogramas extraídos de vídeos de YouTube o de videovigilancia.

- **Sección de Hardware:** En esta sección del proyecto se llevará a cabo la descripción detallada de los componentes, módulos y tarjetas de desarrollo que serán utilizados para la implementación de un sistema de Internet de las cosas (IoT); cuyo enfoque se centrará en la identificación de actos delictivos y la ejecución de medidas disuasorias.

Para la consecución de este proyecto se empleará, como dispositivo central, una PC, la cual actuará como servidor para la página web y para la distribución del modelo con TensorFlow Serving, tarjeta ESP32-CAM; la cual estará encargada de ejecutar las acciones disuasorias y capturar las imágenes para la clasificación. Dicha tarjeta de desarrollo está basada en el microcontrolador ESP32 y dispone de una cámara integrada junto con un módulo wifi, lo cual la convierte en una solución eficiente; para aplicaciones IoT.

A fin de asegurar un funcionamiento óptimo y fiable del sistema, se usará una fuente de alimentación que proporcione una tensión constante de 5 voltios y una corriente de 2 amperios para cargar la batería de Litio y/o alimentar a la placa del módulo disuasorio; se hará uso del módulo Relay para el control del encendido y apagado de luces, los cuales serán parte de las medidas disuasorias implementadas en el sistema.

El módulo Relay es un dispositivo electrónico que permite el control de cargas eléctricas de alta potencia, a través de señales de baja potencia; lo cual resulta ideal para la implementación de sistemas de IoT.

Es importante destacar, que la selección y elección de los componentes y módulos se realizará con suma precaución; con el propósito de garantizar que estos sean compatibles entre sí y cumplan con los requerimientos técnicos necesarios para el adecuado funcionamiento del sistema.

Es igualmente importante llevar a cabo una programación y configuración adecuadas de los componentes, a fin de garantizar la efectividad del sistema; en la identificación de actos delictivos y la ejecución de medidas disuasorias. La programación implica la elaboración y depuración del código, para controlar el comportamiento de los diferentes componentes; mientras que la configuración consiste en la adaptación de los parámetros del sistema a las necesidades específicas del proyecto.

1.1. Diagrama de Flujo para el Prototipo

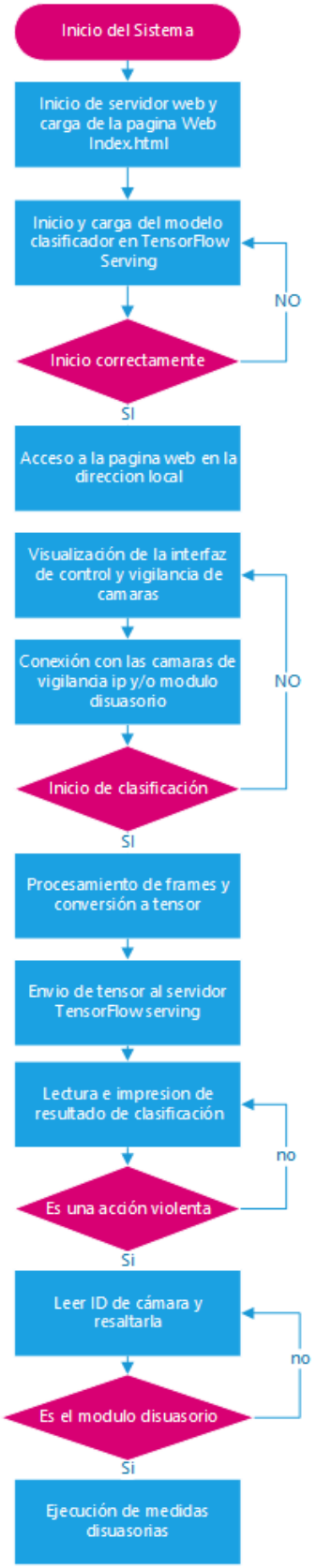


Ilustración 16: Diagrama de flujo para el prototipo a diseñar, Fuente Propia

1.1.1. Explicación del diagrama de flujo

- El inicio del sistema empieza con el arranque del sistema operativo y la ejecución de aplicaciones como Docker y Live Server, para poner en marcha el servidor web y el servidor de TensorFlow Serving; donde estará alojado el modelo de clasificación.

- Una vez que el sistema se haya inicializado correctamente, se puede acceder a la dirección local en nuestro Dispositivo Cliente-Servidor para obtener acceso a la página web "Index.html"; que servirá como nuestra interfaz de control, conexión y supervisión de nuestras cámaras IP.

- Una vez que las cámaras estén conectadas, se puede iniciar el proceso de clasificación presionando el botón "Iniciar clasificación" en la interfaz web de control; también se cuenta con un botón "Detener clasificación" para interrumpirla en caso de ser necesario.

- El proceso de clasificación seleccionará una a una las cámaras, capturará los frames actuales del flujo de vídeo en curso; los almacenará junto con la ID de la cámara y los procesará.

- Para llevar a cabo este procesamiento, los frames serán escalados al tamaño de 224 píxeles de alto por 224 píxeles de ancho, esta dimensión se debe a que el modelo se ha entrenado con estas dimensiones; para la conversión a tensor, utilizaremos TensorFlow JS.

- Una vez que se tenga el tensor, debe ser enviado al servidor TensorFlow Serving para su clasificación. Este servidor al recibir los datos, proporcionará una respuesta; que contendrá el resultado de si la clasificación es o no delictiva.

- En caso de que la clasificación sea delictiva, se resaltará la cámara en cuestión con un color rojo; esto tiene como objetivo alertar al operador. Además, en caso de que la cámara sea la del módulo disuasorio; también se iniciará con las medidas disuasorias.

1.2. Diagrama de bloques del prototipo

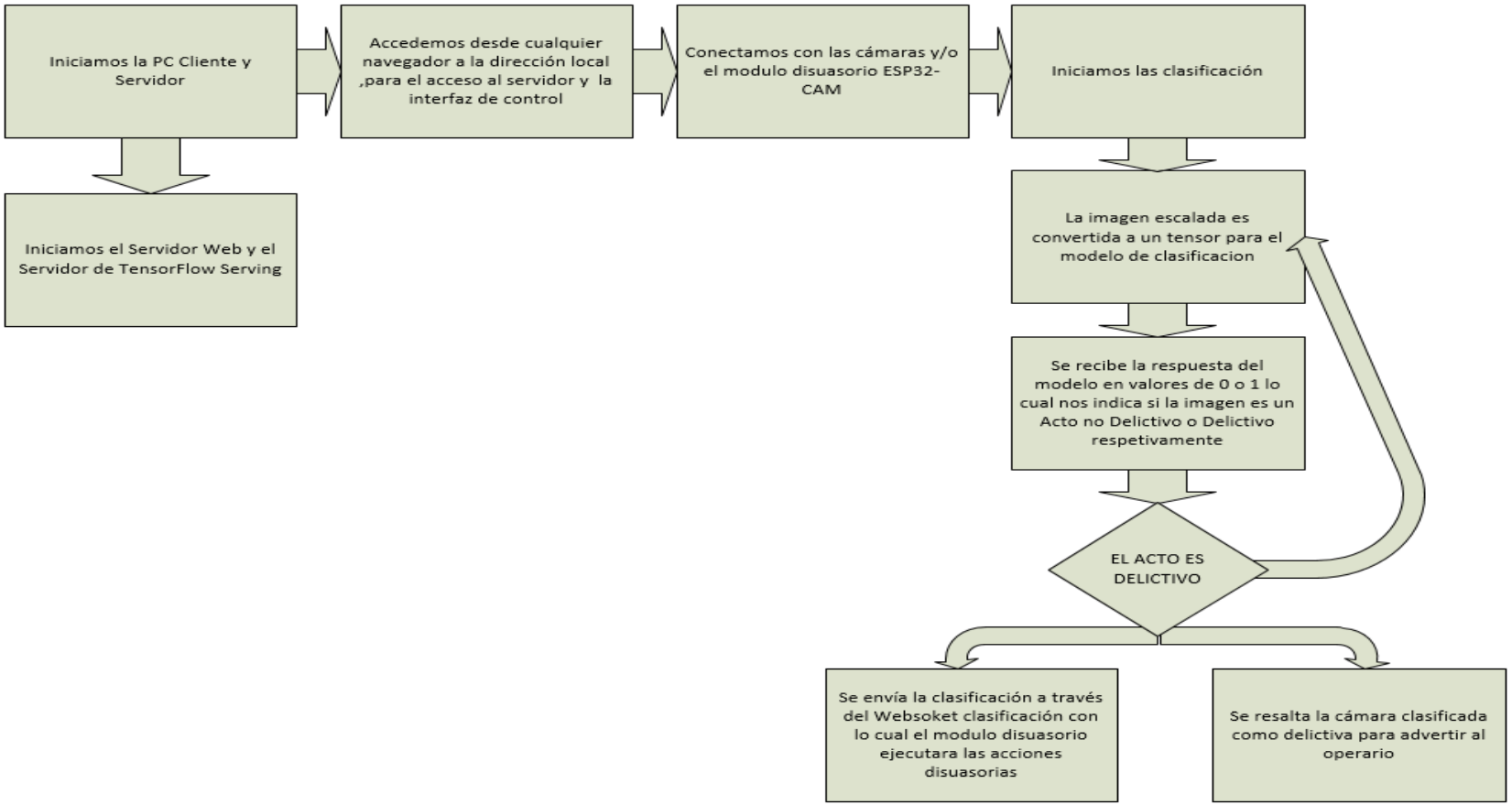


Ilustración 17: Diagrama de bloques para el prototipo a diseñar, Fuente Propia

1.3. Diagrama de Red

1.3.1. Sistema prototipo de clasificación de acciones delictivas

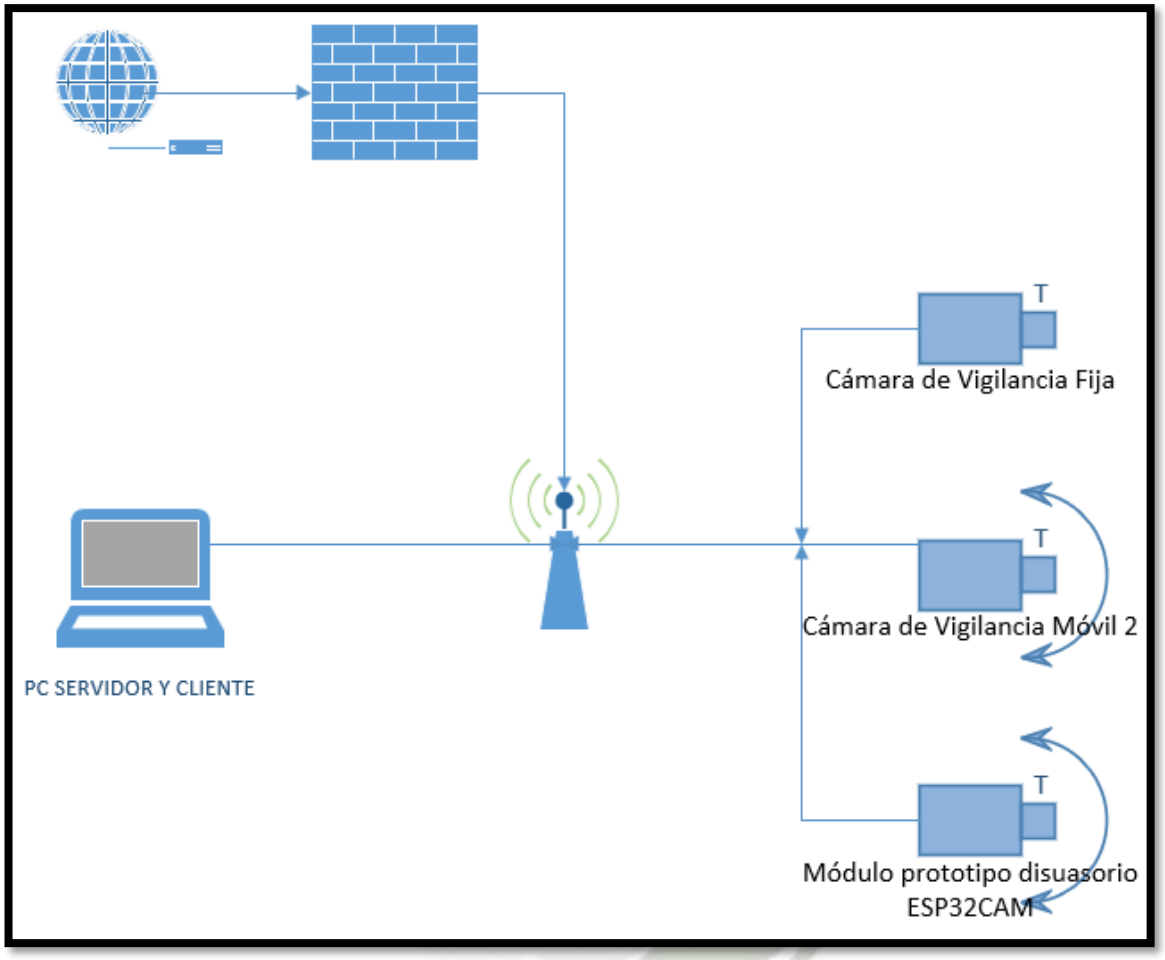


Ilustración 18: Diagrama de red para el sistema prototipo de clasificación, Fuente Propia

El intercambio de datos se llevará a cabo a través de la red WiFi, donde las cámaras de vídeo inalámbricas serán accedidas; mediante las direcciones IP asignadas por el router WiFi correspondiente.

El servidor web, alojado en la PC del usuario, desempeñará la función de recibir, procesar y entregar el flujo de vídeos provenientes de las cámaras de seguridad inalámbricas al modelo para su clasificación; dependiendo de la respuesta de esta clasificación se activarán medidas disuasorias en el módulo prototipo de disuasión.

La comunicación de la clasificación al módulo de disuasión, se llevará a cabo a través de WebSocket; una tecnología que permite una interacción fluida entre el servidor y el cliente.

Es importante destacar que, en este escenario, el cliente también actúa como servidor debido a razones de prueba; lo que significa que la misma configuración se puede aplicar en un entorno real.

Sin embargo, es relevante mencionar que esta arquitectura conlleva una carga computacional significativa en un solo dispositivo, ya que debe gestionar la entrega de la página web y la visualización en tiempo real de los flujos de vídeo provenientes de las cámaras; además de la clasificación de los mismos para determinar si se trata o no de actividades delictivas. En caso de que se clasifiquen como actividades delictivas, se establecerá una conexión WebSocket; con el módulo de disuasión para que este inicie las medidas de disuasión correspondientes.

2. Componentes del Prototipo disuasorio ESP32-CAM

2.1. ESP32-CAM

ESP32-CAM es un módulo de cámara de bajo costo basado en el microcontrolador ESP32 de Espressif Systems, este módulo integra una cámara OV2640 de 2 megapíxeles; una antena WiFi y una memoria flash en un solo paquete compacto.

El ESP32-CAM es compatible con los protocolos WiFi y Bluetooth y es capaz de conectarse a Internet para transmitir imágenes o vídeo en tiempo real a través de la red. Además, este módulo cuenta con una amplia variedad de interfaces, incluyendo UART, I2C, SPI, SDIO y PWM; lo que lo hace ideal para una amplia gama de aplicaciones de IoT y de desarrollo de proyectos.

El ESP32-CAM se puede programar utilizando el IDE de Arduino y es compatible con varios lenguajes de programación, incluyendo C++ y MicroPython, debido a su bajo costo y su amplia gama de características, el ESP32-CAM es una excelente opción para desarrolladores de proyectos de IoT; que necesiten una solución de cámara de bajo costo y fácil de usar

2.1.1. Funcionamiento en el prototipo

Este dispositivo funcionará como una cámara de videovigilancia destinada a las pruebas del prototipo, estará programado para transmitir vídeo a través de la red Wi-Fi utilizando la dirección IP estática pre configurada; su principal función será

transmitir los datos al modelo servido en TensorFlow Serving y en caso de detectar una acción delictiva, activará las medidas disuasorias correspondientes.

2.2. Módulo Buzzer

Un módulo de zumbador, a veces llamado módulo de zumbador o módulo de altavoz; es un dispositivo electrónico que se utiliza para generar sonidos o tonos audibles en proyectos electrónicos. Estos módulos son componentes populares en proyectos de electrónica y robótica, ya que permiten agregar retroalimentación audible a los sistemas, indicar estados o condiciones, o incluso crear música simple.

El módulo buzzer puede generar un pitido fuerte, el cual permite advertir de un cambio de estado o la activación de un evento; para este proyecto en caso se detecte un hecho delictivo

2.2.1. Funcionamiento en el prototipo

Este módulo tiene la función de emitir un pitido, el cual sustituye de manera prototipo a una sirena de mayor envergadura y rango de audición; debido a que estas son costosas. Este sonido tiene como función alertar en caso de que se produzca un acto delictivo, con el propósito de disuadir dicha acción y advertir a las personas presentes en el entorno; para que puedan tomar precauciones adecuadas y evitar que se conviertan en víctimas adicionales del incidente. Además, se busca brindar una respuesta rápida ante situaciones de emergencia, incrementando así la seguridad y protección de las personas afectadas.

2.3. Módulo de Relés

Un módulo de relés de 5 voltios a 240 voltios, es un dispositivo electromagnético que se utiliza para controlar la apertura y cierre de un circuito eléctrico; su función principal es actuar como un interruptor controlado por una corriente eléctrica externa.

Para prender un foco con un módulo de relés de 5 V y 240 V, se debe conectar el foco a la fuente de alimentación y luego conectar el módulo de relés entre la fuente de alimentación y el foco. Luego, se debe aplicar una señal de voltaje de 5 V al relé correspondiente para activarlo y permitir que la corriente eléctrica fluya hacia el foco; encendiéndolo. Para apagar el foco, simplemente se debe desactivar el relé correspondiente, interrumpiendo el flujo de corriente eléctrica.

2.3.1. Funcionamiento en el prototipo

Este módulo tiene como finalidad facilitar la implementación de medidas disuasorias, mediante la activación de luces luminosas al detectar actos delictivos; iluminando la zona afectada para alertar y proteger a las personas presentes. Esto busca crear una presencia visual impactante, para los perpetradores y concientizar a los individuos circundantes sobre posibles peligros; promoviendo una respuesta más segura y rápida.

La combinación de luces disuasorias e iluminación busca maximizar la efectividad en la protección de la comunidad, reducir delitos y aumentar la sensación de seguridad; contribuyendo a la prevención y disuasión de conductas delictivas futuras.

2.4. Batería de Ion de Litio

Una batería de iones de litio también conocida como batería de litio o Li-ion, es una batería recargable que utiliza el litio; como uno de los componentes principales en un proceso químico. Estas baterías se utilizan ampliamente en una variedad de dispositivos electrónicos portátiles, vehículos eléctricos, sistemas de almacenamiento de energía y muchas otras aplicaciones debido a su alta densidad de energía, baja tasa de auto descarga y con capacidad para ser recargada muchas veces.

2.4.1. Funcionamiento en el prototipo

Este componente tiene como función almacenar energía para el prototipo del módulo disuasorio con el ESP32-CAM, la capacidad de este componente determinará el tiempo en funcionamiento sin energía externa que puede mantenerse encendido el prototipo; esto debido a que las cámaras deben mantenerse en funcionamiento aún si el suministro eléctrico se desconecta o se corta temporalmente.

2.5. Módulo cargador y elevador de tensión TP4056

Un cargador de baterías de litio y elevador de tensión a 5 V es un dispositivo que combina dos funciones en uno, permite cargar una batería de litio y al mismo tiempo proporcionar una salida de 5 voltios (generalmente a través de un puerto USB) para alimentar otros dispositivos electrónicos.

El microcontrolador TP4056, está diseñado para controlar la carga y descarga de baterías de iones de litio; comúnmente utilizado en dispositivos portátiles y equipos eléctricos portátiles. Es un componente electrónico, encargado de controlar los procesos de carga y descarga de una batería de iones de litio; protegiéndola de daños y alargando su vida útil. TP4056 incluye protección contra sobretensión, sobre temperatura y cortocircuito, así como función de carga constante y función de descarga constante como también hay un LED que muestra el estado de carga de la batería.

2.5.1. Funcionamiento en el prototipo

Este módulo es el encargado de cargar y descargar de manera segura la batería del prototipo, para que la vida útil de este sea larga, también tiene la función de elevar el voltaje de salida de la batería a 5v; lo que permite alimentar a la placa del módulo disuasorio con el ESP32-CAM porque la batería de litio solo nos suministra 3.7 voltios.

2.6. Estructura 3D para el Módulo disuasorio ESP32-Cam

La estructura tridimensional (3D), ha sido concebida como una plataforma diseñada para alojar una cámara ESP32-CAM; así como la placa del módulo disuasorio. Su funcionalidad principal reside en la capacidad de almacenar una batería y de facilitar el acceso a los módulos de relé, para alimentar y controlar tanto la lámpara como el buzzer; los cuales emiten señales auditivas como parte de las medidas disuasorias.

Adicionalmente a ello, se ha contemplado la posibilidad de acceder a los puertos USB del módulo TP4056; ya que a través de estos puertos se llevará a cabo la recarga de la batería de litio. Asimismo, se ha previsto la inclusión de un mecanismo de cierre; que permita acceder a la placa y observar su interior de manera conveniente.

3. Perfil Económico

Para el desarrollo y pruebas de este trabajo de investigación, se llevó a cabo el diseño y construcción de un prototipo. Este prototipo se encuentra limitado, en términos de sus características y accesorios reales; destinados para su uso en un entorno urbano.

En este sentido, se han elaborado dos perfiles económicos: uno para el módulo prototipo desarrollado y otro para el módulo que se utilizaría en un entorno urbano. Este último

se basa en los precios del mercado y se presenta con el propósito de proporcionar una estimación del costo.

El cálculo de estos perfiles incluye elementos disuasorios, como una lámpara de alta potencia para iluminar la zona donde se produce el evento; así como una sirena destinada a alertar a los transeúntes y/o disuadir cualquier evento no deseado. Además, se requiere que este módulo cuente con una protección IP23 (Ingress Protection 23; que garantiza su resistencia a la lluvia).

3.1. Perfil económico del módulo prototipo disuasorio ESP32-CAM

Componente	Cantidad	Unidad	Precio
ESP32-cam	1	Unitario	S/ 40
Módulo Buzzer	1	Unitario	S/ 02
Módulo de Relé	1	Unitario	S/ 05
Batería de Litio CL-503450	1	Unitario	S/ 15
Módulo cargador y elevador de tensión TP4056	1	Unitario	S/ 17
Foco de luz	1	Unitario	S/ 17
Cables de protoboard	3	Paquete	S/ 11
Otros	1	Paquete	S/ 20
Estructura 3D	1	Paquete	S/ 45
Total			S/ 172

Tabla 1 Perfil económico módulo prototipo

3.2. Perfil económico del módulo disuasorio ESP32-CAM

Componente	Cantidad	Unidad	Precio
ESP32-cam	1	Unitario	S/ 40
Módulo Buzzer	1	Unitario	S/ 02
Módulo de Relé	2	Unitario	S/ 10
Batería de Litio CL-503450	1	Unitario	S/ 15
Fuente Switching 5v	1	Unitario	S/ 15
Sirena de 60W modelo (OP-58S)	1	Unitario	S/ 70
Módulo cargador y elevador de tensión TP4056	1	Unitario	S/ 17
Reflector Led 50W	1	Unitario	S/ 90
Cables de protoboard	3	Paquete	S/ 11
Otros	1	Paquete	S/ 20
Estructura 3D con protección IP23	1	Paquete	S/ 80
Total			S/ 370

Tabla 2 Perfil económico módulo disuasorio

4. Modelado electrónico

4.1. Cálculo de duración de batería

Datos:

- Batería modelo CL-503450
  - Voltaje: 3,7 V
  - Capacidad: 1000 mAh, 1C de descarga
  - La corriente máxima de carga: 1A
  - Corriente máxima de descarga: 1A
  - Voltaje de corte de descarga: 2,75 V
  - Función de protección: sobrecarga, sobre descarga, sobre corriente, protección contra cortocircuitos, protección contra sobrecalentamiento
- Prototipo de sistema disuasorio (Alimentación por Batería)
  - Voltaje: 3.7 V
  - Corriente consumida: 500Ma

Cálculos:

- Tiempo de funcionamiento (en horas) = Capacidad de la batería (en mAh) / Consumo del equipo (en mA)
- Tiempo de funcionamiento = 1000mAh / 500mA = 2 horas

Conclusión

Al ser un prototipo, se considera que dos horas de funcionamiento en caso de corte eléctrico; proporciona una autonomía suficiente para el sistema.

4.2. Comparación de transistores

- El 2N2222 es un transistor de propósito general que ha sido ampliamente utilizado en aplicaciones electrónicas durante décadas debido a su disponibilidad, confiabilidad y versatilidad.
- 2N3904: Es un transistor de propósito general muy común. Ambos transistores tienen características eléctricas similares y se pueden utilizar en muchas aplicaciones intercambiablemente. Sin embargo, el 2N2222 puede tener una capacidad de corriente ligeramente mayor, lo que lo hace más adecuado para aplicaciones que requieren manejo de corriente más alto.
- BC548: Este es otro transistor NPN de propósito general que es ampliamente utilizado. Si bien el BC548 es similar al 2N2222 en términos de especificaciones eléctricas básicas, el 2N2222 puede ser preferido en aplicaciones donde se necesita una mejor tolerancia a altas temperaturas o donde se espera una mayor durabilidad.
- PN2222: Esta es una versión similar al 2N2222, pero con encapsulado de montaje superficial (SMD). Si bien puede ser más conveniente en aplicaciones donde el espacio es limitado y se prefiere un diseño de montaje superficial, el 2N2222 en encapsulado de orificio pasante sigue siendo más fácil de usar en prototipos y proyectos de aficionados.

Conclusión

- Disponibilidad: El 2N2222 es ampliamente disponible en muchas tiendas de componentes electrónicos, lo que lo hace conveniente para prototipos y proyectos de aficionados.
- Durabilidad: Tiene una buena resistencia a condiciones ambientales adversas y puede manejar temperaturas más altas en comparación con algunos de sus competidores.
- Rango de aplicaciones: Con su capacidad de corriente razonablemente alta y especificaciones eléctricas versátiles, el 2N2222 es adecuado para una amplia gama de aplicaciones, desde amplificadores de audio hasta circuitos de conmutación.
- En resumen, aunque existen otros transistores similares al 2N2222, su disponibilidad, durabilidad y versatilidad lo convierten en una opción preferida en muchas aplicaciones electrónicas. Sin embargo, la elección del transistor adecuado dependerá de las especificaciones y requisitos específicos de cada proyecto.

4.3. Cálculo de resistencia de transistor para Relé

Datos:

- ESP32-CAM
  - Corriente de entrada: 500mA
  - Corriente máxima GPIO: 40mA
  - Voltaje de entrada: 3.3 v o 5v
  - Voltaje de salida: 3.3v o 5v

- Transistor 2N2222
  - VCBO Máxima Colector-Base: 75 V
  - Tensión VEBO Máxima Emisor-Base: +6 V
  - Tensión VCEO Máxima Colector-Emisor: 40 V
  - Intensidad de colector: 500mA
  - Impedancia de entrada: 8KΩ
  - Frecuencia Máxima de Funcionamiento: 300 MHz
  - Hfe: 300

- Relé SRD-05VDC-SL-C
  - Voltaje de Operación bobina: 5V DC
  - Corriente bobina: 75 mA
  - Voltaje máximo de carga: 240V AC/ 30V DC
  - Corriente máxima de carga: 10A
  - Contactos: 1 NO, 1 NC
  - Tiempo de acción: 10 ms / 5 ms

Cálculos:

- $\text{Ib} = \frac{\text{Ic}}{\text{hfe}} = \frac{500*10^{-3}}{300} = 1.7\text{mA}$
- $\text{Vrb} = \text{Vesp32} - \text{Vbe} = 5\text{v} - 0.7 = 4.3\text{v} \rightarrow 4\text{v}$
- $\text{Rb} = \frac{\text{Vrb}}{\text{Ib}} = \frac{4}{1.7*10^{-3}} = 2.35 * 10^{3} \rightarrow 2\text{k}\Omega$

Conclusión

Para el control del relé, se ha evaluado la utilización de un transistor convencional, en particular el modelo 2N2222. Este dispositivo, se ha seleccionado debido a su capacidad para gestionar; tanto la tensión como la corriente requerida para el funcionamiento del relé.

El mencionado dispositivo, tiene una capacidad de voltaje nominal de 40 voltios es significativamente superior a la demanda; que es de tan solo 5 voltios. En cuanto a la corriente necesaria, el transistor tiene la capacidad de soportar hasta 500 mA, lo cual supera con creces los 75 mA; necesarios para activar la bobina del relé.

Además, mediante cálculos realizados, se ha determinado que la resistencia utilizada en este circuito; no debe superar los 2k ohmios para garantizar un funcionamiento adecuado.

4.4. Diagrama Electrónico

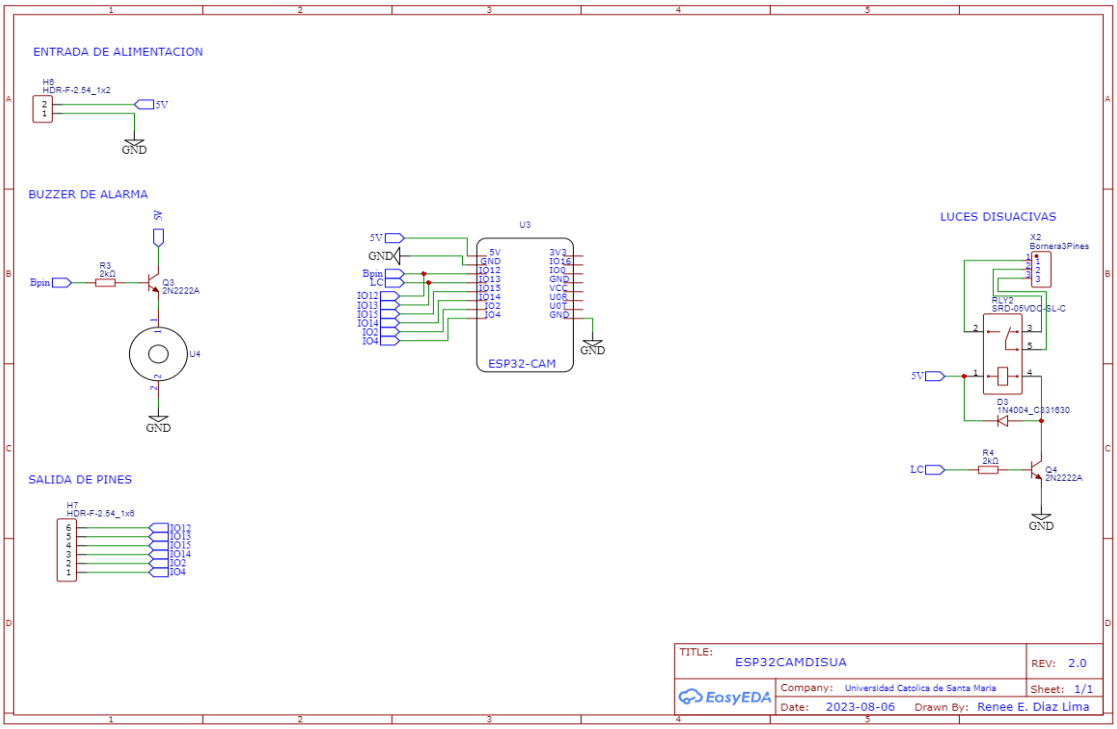


Ilustración 19: Imagen de Diagrama Electrónico,
Fuente Propia, Fuente: Anexo 1 Diagrama electrónico

El diagrama electrónico anteriormente mostrado, es una imagen reducida del que se adjunta en la sección ANEXOS - "Diagrama Electrónico"; este diagrama nos muestra las conexiones

existentes en la tarjeta principal ESP32-DevKit con los demás módulos y componentes que se describen a continuación:

- **Entrada de Alimentación:** En esta sección se observa la colocación de una bornera que recibirá la alimentación del módulo cargador y elevador de tensión TP4056, debido a que este se encargará de elevar el voltaje de la batería de 3.7v a 5v; que es el voltaje que usará el ESP32-CAM y el resto de componentes.
- **Buzzer de alarma:** En esta sección se presentan las conexiones necesarias para activar el zumbador, dado que el zumbador requiere una corriente de 35 mA que puede ser suministrada por el ESP32; se ha optado por utilizar el mismo circuito de activación que se empleó para el relé, por motivos de seguridad.
- **Salidas de Pines:** En esta sección se observa la colocación de espadines, que nos dará acceso a los pines disponibles ya que con esto y mediante la programación; se puede añadir más medidas disuasorias y/o mejorar la capacidad de las actuales.
- **Luces Disuasivas:** En esta sección, se pueden observar las conexiones necesarias para utilizar los relés; que activarán las lámparas lumínicas utilizadas como medidas disuasorias. En este diagrama, se requiere un pulso constante de 5 voltios en la entrada del transistor 2N2222; que funcionará en modo corte-saturación. Dado que se trata de una señal de control digital, se utilizará el pin digital 13 de la tarjeta ESP32-CAM.

4.5. Diagrama PCB del módulo prototipo disuasorio ESP32-CAM

4.5.1. Máscara de componentes

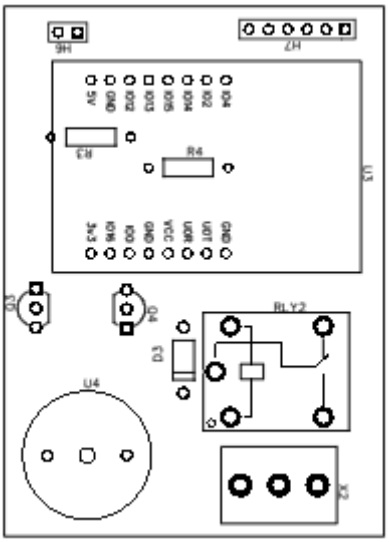


Ilustración 20: Imagen máscara de componentes,
Fuente Propia, Fuente: Anexo 1 Máscara de componente

4.5.2. Cara de pistas

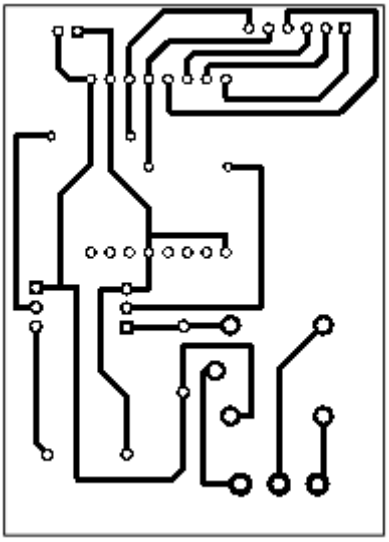


Ilustración 21: Imagen máscara de pistas,
Fuente Propia, Fuente: Anexo 1 Pistas

CAPÍTULO IV. DESARROLLO E IMPLEMENTACIÓN DEL SISTEMA

1. Modelo de Machine Learning para Clasificación de Acciones Delictivas

Para la clasificación de imágenes delictivas y no delictivas, se utilizará un conjunto de datos etiquetado; que contenga imágenes de ambos tipos. Durante el entrenamiento, el modelo aprenderá a distinguir entre las dos categorías y luego; se puede evaluar su rendimiento en un conjunto de datos de prueba para medir su precisión.

Los modelos: Modelo Secuencial, Modelo Red Neuronal Convolucional (CNN) y Modelo Red Neuronal Convolucional con DropOut, son tres tipos de arquitecturas de redes neuronales que se utilizan para la clasificación de imágenes; cada uno de ellos tiene sus ventajas y desventajas que se describen a continuación:

1.1. Modelos considerados

1.1.1. Modelo Secuencial Denso

Este modelo secuencial es el más simple y básico de los tres, consiste en una serie de capas lineales que procesan la imagen de entrada y producen una salida; con las probabilidades de pertenencia a cada clase. Este modelo es fácil de implementar y entender, pero tiene algunas limitaciones. Por ejemplo, no puede capturar las relaciones espaciales entre los píxeles de la imagen, lo que puede afectar al rendimiento en problemas complejos, además es propenso al sobreajuste, es decir; a memorizar los datos de entrenamiento y no generalizar bien a nuevos ejemplos.

1.1.2. Modelo Red Neuronal Convolucional

El modelo CNN es una mejora del modelo secuencial, ya que incorpora capas convolucionales; que aplican filtros sobre la imagen de entrada para extraer características relevantes. Estas capas permiten al modelo aprender patrones visuales de diferentes niveles de abstracción, lo que mejora la capacidad de clasificación.

Además de ello, el modelo CNN reduce el número de parámetros a entrenar; lo que reduce el riesgo de sobreajuste y el tiempo de cómputo. Sin embargo, el modelo CNN también tiene algunos inconvenientes. Por ejemplo, requiere más datos y recursos para entrenarse correctamente, lo que puede ser un problema en algunos

escenarios, adicionalmente a ello; puede ser difícil interpretar los resultados del modelo y entender qué ha aprendido cada capa.

1.1.3. Modelo de Aprendizaje Profundo MoViNet

MoViNet es un tipo de modelo de aprendizaje profundo, diseñado para comprender y procesar vídeos; el nombre "MoViNet" proviene de "Motion and Vídeo Network" (Red de Movimiento y Vídeo, en español).

Los modelos MoViNet están diseñados específicamente para tareas relacionadas con el análisis de vídeo, como el reconocimiento de acciones, la detección de objetos y el seguimiento en vídeo; estos modelos se basan en redes neuronales convolucionales (CNN) y están diseñados para capturar las características espaciales y temporales de los vídeos. Se destacan en tareas que requieren comprender el movimiento y el contexto de objetos y acciones, como videovigilancia; conducción autónoma y análisis de contenido de vídeo.

Los mencionados modelos son parte de los avances continuos en los campos de la visión por computadora y el aprendizaje profundo, y demuestran mejoras significativas en precisión y eficiencia con respecto a modelos anteriores cuando se implementan tareas relacionadas con vídeo.

En resumen, los modelos Secuencial, CNN y MoViNet son tres opciones válidas para la clasificación de imágenes, considerando que la clasificación se realizará en frames para determinar si las imágenes son delictivas o no; sin embargo, cada uno de estos modelos tiene sus ventajas y desventajas.

El modelo Secuencial es el más sencillo y fácil de utilizar, pero también es el menos potente y más susceptible al sobreajuste. Por otro lado, el modelo CNN es más avanzado y eficaz en términos de rendimiento; pero su complejidad aumenta debido al incremento en el número de capas.

El modelo MoViNet, basado en redes convolucionales 3D, va un paso más allá al considerar no solo las dos dimensiones de ancho y largo de la imagen, sino también el factor temporal; lo que permite un análisis más completo de las acciones en vídeos. Este modelo, se posiciona como la opción más avanzada y eficaz en términos de rendimiento; aunque es importante destacar que requiere mayores recursos computacionales.

En última instancia, la elección del modelo dependerá de las necesidades específicas del proyecto, los recursos disponibles y el equilibrio entre la simplicidad y el rendimiento que se busca alcanzar.

1.2. Librerías y Capas

1.2.1. Librerías

- TensorFlow: Es una biblioteca de aprendizaje automático y de redes neuronales, en particular se utilizó la sub biblioteca keras de TensorFlow; que proporciona una API de alto nivel para construir y entrenar modelos de redes neuronales.
- keras: Es una biblioteca de código abierto para la programación de redes neuronales en Python. Fue desarrollada inicialmente por François Chollet y se ha integrado en TensorFlow, una popular plataforma de aprendizaje automático; como su interfaz de alto nivel. Keras proporciona una API simple y consistente para diseñar, entrenar y evaluar modelos de redes neuronales profundas.
- TensorFlow. keras. layers: Es una sub biblioteca de TensorFlow/Keras que proporciona clases para crear diferentes tipos de capas de redes neuronales, como capas densas, capas de convolución, capas de agrupamiento, etc.
- TensorFlow.keras.models: Es una sub biblioteca de TensorFlow/Keras que proporciona la clase Sequential, utilizada para crear una instancia de un modelo de red neuronal secuencial.
- TensorFlow.keras.preprocessing.image: Es una sub biblioteca de TensorFlow/Keras que proporciona la clase ImageDataGenerator, que se utiliza para cargar y pre procesar imágenes para su uso en el entrenamiento de modelos de redes neuronales
- tqdm: Es una librería de Python que se utiliza para agregar barras de progreso (progress bars) a bucles y tareas iterativas, el nombre "tqdm" proviene de la frase en inglés "taqaddum"; que significa "progreso" en árabe.

- OpenCV (Open Source Computer Vision Library): Es una librería de código abierto ampliamente utilizada para el procesamiento de imágenes y visión por computadora. Fue desarrollada originalmente por Intel y ahora es mantenida por una comunidad de código abierto

1.2.2. Capas

- Flatten: Capa de aplanamiento, que se utiliza para convertir la entrada de una imagen de dos dimensiones; a un vector de una dimensión antes de pasarla a una capa densa.
- Dense: Capa densa completamente conectada, que se utiliza para realizar operaciones de multiplicación de matrices; entre las entradas y los pesos de la capa.
- Dropout: Capa que se utiliza para evitar el sobreajuste en el modelo, descartando aleatoriamente algunas unidades de la capa; durante el entrenamiento.
- Conv2D: Capa de convolución bidimensional, que se utiliza para extraer características de la imagen.
- Conv2Plus1D: Capa de red neuronal convolucionales tridimensionales (3D CNN) y se basa en el concepto de separar las convoluciones espaciales y temporales para un procesamiento más eficiente de los datos de vídeo.
- MaxPooling2D: Capa de agrupamiento que se utiliza para reducir la dimensionalidad de las características extraídas por la capa de convolución.

1.3. Dataset

Un dataset es un conjunto de datos que se utiliza para el entrenamiento de inteligencia artificial en clasificación de imágenes, un dataset puede contener miles o millones de imágenes etiquetadas con diferentes categorías, como animales, plantas, personas, objetos, etc.

La utilidad de un dataset es que permite a la inteligencia artificial aprender a reconocer y diferenciar las características visuales de cada categoría, así como a asignar una etiqueta a una imagen nueva que no haya visto antes. Un dataset bien diseñado y

equilibrado es esencial para el éxito de un sistema de clasificación de imágenes, ya que influye en la precisión y la generalización del modelo.

1.4. Preparación de Dataset

1.4.1. Obtención de dataset

Addon “"Download All Images"”

"Download All Images" es una extensión para el navegador Google Chrome, que permite descargar todas las imágenes de una página web en un solo clic.

Una vez que se ha instalado la extensión, aparecerá un botón "Download All Images" en la barra de herramientas del navegador. Al hacer clic en este botón, todas las imágenes cargadas de manera temporal en la página web se descargarán

Es importante tener en cuenta que, esta extensión solo descarga imágenes que se pueden cargar en la página web actual y que están disponibles públicamente; no se pueden descargar imágenes que se encuentran detrás de una protección con contraseña o que son propiedad de otra persona sin su permiso.

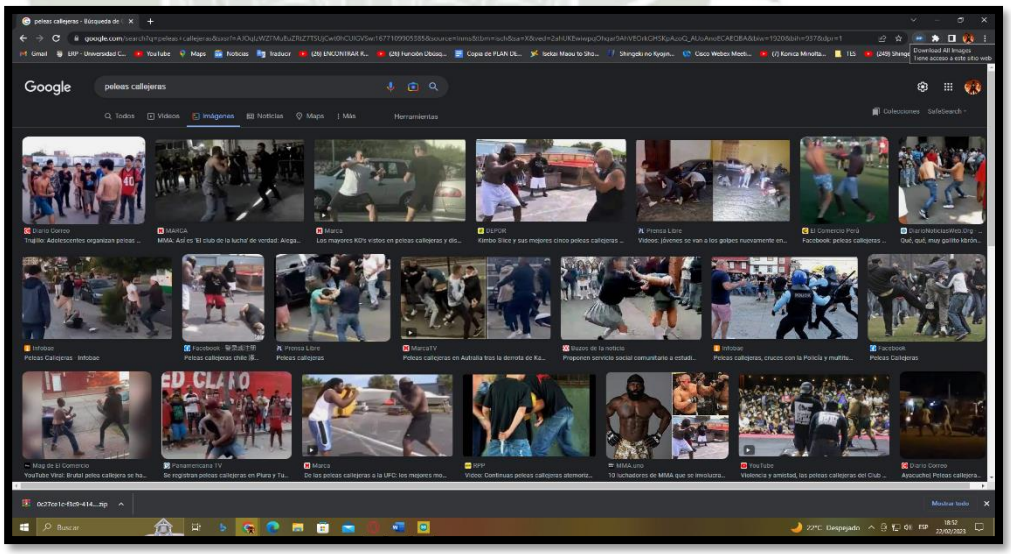


Ilustración 22: Descarga de dataset "Actos Delictivos",
Fuente: Propia

Dataset de acceso gratuito

Un dataset de acceso gratuito es un conjunto de datos que está disponible en línea y que puede ser descargado y utilizado sin costo alguno. Estos conjuntos de datos son una herramienta muy valiosa para entrenar algoritmos de aprendizaje automático, como redes neuronales; que pueden aprender a realizar tareas específicas a partir de ejemplos proporcionados en el dataset.

En este caso, se están utilizando algunos datasets de acceso gratuito de la página https://dasci.es/es/transferencia/open-data/deteccion-de-armas/ para entrenar redes neuronales en la tarea de detección de armas.

Uno de los datasets que se está utilizando se llama "Detección de pistolas" y contiene 3000 imágenes de armas de fuego cortas con un fondo rico en contexto.

Estas imágenes, han sido seleccionadas de internet y contienen una o más pistolas en situaciones diversas; incluyendo contextos de videovigilancia.

El otro dataset que se está utilizando se llama "Detección de armas blancas" y contiene 2078 imágenes donde al menos aparece un cuchillo, estas imágenes también han sido seleccionadas de internet y algunas pertenecen a fotogramas; extraídos de vídeos de YouTube o de videovigilancia.

Es importante destacar que estos datasets son de acceso gratuito, lo que significa que cualquier persona puede descargarlos y utilizarlos; para entrenar sus propios algoritmos de detección de armas sin incurrir en costos adicionales. Además, estos datasets proporcionan un conjunto de datos de alta calidad y diversidad, que pueden ser utilizados para entrenar algoritmos precisos y robustos; en la detección de armas en imágenes y vídeos.

Ilustración 23: Dataset "Detecciòn de pistolas",
Fuente: https://dasci.es/es/transferencia/open-data/deteccion-de-armas/

1.4.2. Preparación de Dataset

ImageDataGenerator

ImageDataGenerator es una clase en la biblioteca Keras de Python, que se utiliza para generar lotes de imágenes aumentadas en tiempo real; durante el entrenamiento de un modelo de aprendizaje profundo. La idea detrás del aumento de datos es aumentar la cantidad de datos de entrenamiento; al realizar transformaciones aleatorias en las imágenes existentes; lo que hace que el modelo sea más robusto y generalice mejor.

El ImageDataGenerator toma un conjunto de imágenes de entrenamiento y aplica transformaciones aleatorias a cada imagen, como rotaciones, ampliaciones, cambios de brillo, zoom y volteos horizontales y verticales. Estas transformaciones aumentan la cantidad de datos de entrenamiento, disponibles para el modelo sin la necesidad de recopilar y etiquetar más datos.

ImageDataGenerator también proporciona la capacidad de pre procesar las imágenes, lo que es importante para garantizar que todas las imágenes tengan las mismas dimensiones y sean adecuadas para la entrada del modelo. Además, ImageDataGenerator también puede ser utilizado para la validación y prueba, para aplicar el mismo pre procesamiento y aumentar las imágenes durante la validación y prueba; ayudando a evaluar el modelo de manera más precisa.

Expansión de dataset con ImageDataGenerator

```python
#aumentar datos con ImageDataGenerator
from tensorflow.keras.preprocessing.image import ImageDataGenerator
import numpy as np

#crear dataset generador
datagen= ImageDataGenerator(
    rescale=1. / 255,
    rotation_range = 30,
    width_shift_range=30 ,
    shear_range = 15,
    zoom_range=[0.5,1.5],
    validation_split=0.2 #20% para pruebas lo que tambien sera 80% para entrenar
)

#generadores para dataset
#entrenamiento
data_gen_entrenamiento= datagen.flow_from_directory('/content/dataset/',target_size=(224,224),
                                                    batch_size=32, shuffle=True, subset='training')
#pruebas
data_gen_pruebas= datagen.flow_from_directory('/content/dataset/',target_size=(224,224),
                                                    batch_size=32, shuffle=True, subset='validation')
#imprimir 10 imagenes preparadas
for imagen, etiqueta in data_gen_entrenamiento:
  for i in range(10):
    plt.subplot(2,5,i+1)
    plt.xticks([])
    plt.yticks([])
    plt.imshow(imagen[i])
  break
  plt.show()
```

Ilustración 24: Código Phyton , Expansión de Dataset ,
Fuente: Propia

El código desarrollado utiliza ImageDataGenerator de Keras, para aumentar los datos de imágenes en un conjunto de datos y prepararlos para el entrenamiento y evaluación de un modelo de aprendizaje profundo; a continuación, se realiza la descripción detallada del código:

1. Primero, importa la clase ImageDataGenerator de la biblioteca tensorflow.keras.preprocessing.image y la biblioteca numpy para trabajar con matrices y arreglos.
2. Luego, se crea un objeto datagen de ImageDataGenerator, que se utiliza para generar los lotes de imágenes aumentadas; datagen se configura con diferentes transformaciones de imágenes, como rotación, cambio de ancho, cambio de inclinación, aumento y validación de datos.
3. Posteriormente a ello, se definen dos generadores de datos, uno para entrenamiento (data_gen_entrenamiento) y otro para pruebas (data_gen_pruebas); utilizando la función flow_from_directory.

   La función flow_from_directory toma la ruta del directorio de imágenes, el tamaño objetivo de las imágenes, el tamaño del lote y la división de

entrenamiento/prueba. Los argumentos shuffle y subset, se utilizan para especificar si las imágenes se deben barajar después de cada época de entrenamiento y si se deben utilizar para entrenamiento o prueba.

4. Por último, se utiliza un bucle para imprimir las primeras 10 imágenes generadas, utilizando el generador de datos de entrenamiento; las imágenes se muestran en una cuadrícula de 2 filas y 5 columnas, la función plt.imshow se utiliza para mostrar cada imagen.

Obtención de frames de vídeoclips

```python
def frames_from_video_file(video_path, n_frames, output_size=(224, 224), frame_step=15):
  # Leer cada fotograma del video
  result = []
  src = cv2.VideoCapture(str(video_path))

  video_length = src.get(cv2.CAP_PROP_FRAME_COUNT)

  need_length = 1 + (n_frames - 1) * frame_step

  if need_length > video_length:
    start = 0
  else:
    max_start = video_length - need_length
    start = random.randint(0, max_start + 1)

  src.set(cv2.CAP_PROP_POS_FRAMES, start)
  # ret es un booleano que indica si la lectura fue exitosa, frame es la imagen en sí
  ret, frame = src.read()
  result.append(format_frames(frame, output_size))

  for _ in range(n_frames - 1):
    for _ in range(frame_step):
      ret, frame = src.read()
    if ret:
      frame = format_frames(frame, output_size)
      result.append(frame)
    else:
      result.append(np.zeros_like(result[0]))
  src.release()
  result = np.array(result)[..., [2, 1, 0]]

  return result

```

Ilustración 25: Código Python, Obtención de frames,
Fuente: Propia

Para la obtención de los Frame se usaron las librerías OpenCV y tqmp que sirven para el manejo de vídeos, con esto se pueden acceder al vídeoclip en formato MP4, las funciones se encargan de obtener 10 frames de cada vídeoclip y guardar cada una con el nombre del archivo más un número correlativo; también se encarga de cambiar el tamaño a 244*244 píxeles que es el tamaño predeterminado por tensorflow lo que asegura el mejor rendimiento para el modelo MoViNet.

1.4.3. División de Dataset en datos de entrenamiento y pruebas

La división de un conjunto de datos en datos de entrenamiento, pruebas y validación es una técnica fundamental en el aprendizaje automático y la ciencia de datos.

Esta división se utiliza para evaluar y validar el rendimiento de un modelo de machine learning, antes de implementarlo en un entorno de producción; a continuación, se explica brevemente qué significa cada uno de estos conjuntos de datos:

- Datos de entrenamiento (Training Data): Este conjunto de datos se utiliza para entrenar el modelo de machine learning, los algoritmos de aprendizaje automático aprenden a partir de estos datos y ajustan sus parámetros para hacer predicciones precisas.
- Datos de prueba (Testing Data): Este conjunto de datos se utiliza para evaluar el rendimiento del modelo después de que ha sido entrenado. El modelo hace predicciones en función de estos datos, y luego se comparan estas predicciones con los valores reales conocidos.
- Datos de validación (Validation Data): Este conjunto se utiliza para ajustar los hiperparámetros del modelo y realizar la selección del mejor modelo.

```python
def divisiones(ruta_archivo_zip, num_clases, divisiones, directorio_descarga):
  with zipfile.ZipFile(ruta_archivo_zip, 'r') as zip:
    archivos = zip.namelist()

    archivos_por_clase = get_files_per_class(archivos)

    clases = list(archivos_por_clase.keys())[:num_clases]

    for clase in clases:
      random.shuffle(archivos_por_clase[clase])

    # Utiliza solo la cantidad de clases que deseas en el diccionario
    archivos_por_clase = {x: archivos_por_clase[x] for x in clases}

    directorios = {}
    for nombre_division, cantidad_archivos in divisiones.items():
      print(nombre_division, ":")
      directorio_division = directorio_descarga / nombre_division
      archivos_division, archivos_por_clase = split_class_lists(archivos_por_clase, cantidad_archivos)
      download_from_zip(ruta_archivo_zip, directorio_division, archivos_division)
      directorios[nombre_division] = directorio_division

  return directorios

ruta_archivo_zip = '/content/DATAVIDEOS.zip'
directorio_descarga = pathlib.Path('/content/Datasetv/')
num_clases = 2  # Número de clases que deseas incluir en el subconjunto
divisiones = {"entrenamiento": 80, "validacion": 10, "prueba": 10}  # Número de archivos por división

subset_paths = divisiones(ruta_archivo_zip, num_clases, divisiones, directorio_descarga)
```

Ilustración 26: Código Python, División de dataset,
Fuente: Propia

1.5. Entrenamiento de Modelo Denso

La red consiste en una capa de aplanamiento, seguida de dos capas densas (totalmente conectadas) con activación ReLU y una capa densa final con activación softmax.

```
from tensorflow.keras.layers import Dense, Flatten
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Conv2D, MaxPooling2D, Flatten, Dense, Dropout
# Crear el modelo
model = Sequential()
model.add(Flatten(input_shape=(224,224,3))) # Aplanar las imagenes para poder ser procesadas
model.add(Dense(128, activation='relu')) # Capa densa con 128 unidades y función de activación relu
model.add(Dense(64, activation='relu')) # Capa densa con 64 unidades y función de activación relu
model.add(Dense(2, activation='softmax')) # Capa densa con 2 unidades y función de activación softmax

# Compilar el modelo
model.compile(optimizer='adam', loss='categorical_crossentropy', metrics=['accuracy'])

# Entrenar el modelo
history = model.fit(data_gen_entrenamiento, epochs=50, validation_data=data_gen_pruebas,callbacks=[tdenso])
```

Ilustración 27: Código Phyton , Entrenamiento Modelo Denso, fuente propia

A continuación, se explica las líneas del código en detalle:

1. Crea un nuevo modelo secuencial vacío.
2. Esta línea define la capa de aplanamiento, que convierte la imagen de entrada en un vector unidimensional. La entrada debe tener una forma de (224, 224, 3), lo que significa que las imágenes de entrada deben tener una resolución de 224x224 y tener tres canales de color (rojo, verde y azul).
3. Esta línea agrega una capa densa con 128 unidades y activación ReLU, la capa densa es completamente conectada; lo que significa que cada neurona en la capa anterior está conectada a cada neurona en la capa actual. La función de activación ReLU, es una función no lineal que se usa comúnmente en redes neuronales.
4. Esta línea agrega otra capa densa con 64 unidades y activación ReLU.
5. Esta línea agrega una capa densa final con dos unidades y activación softmax, que es comúnmente utilizada en la capa de salida de una red neuronal; para producir una distribución de probabilidad sobre varias clases. En este caso, hay dos clases: una para cada categoría de imágenes.
6. Esta línea compila el modelo, se usa el optimizador "Adam"; que es un método popular para optimizar las redes neuronales. La función de pérdida se establece en "categorical_crossentropy", que es una función de pérdida comúnmente

utilizada para problemas de clasificación; el modelo se evaluará en términos de precisión (accuracy).

7. Esta línea entrena el modelo en los datos de entrenamiento durante 50 épocas, se utilizan los datos de prueba como conjunto de validación; también se utiliza una función de callback para monitorear el progreso del entrenamiento.

1.6. Entrenamiento de Modelo Red Neuronal Convolucional

Este código desarrollado crea un modelo de red neuronal convolucional (CNN) para la clasificación de imágenes, la red tiene varias capas que procesan los datos de entrada y generan una salida; que corresponde a una de dos clases (lo que se conoce como un problema de clasificación binaria).

```
# Crear el modelo
model1 = Sequential()
model1.add(Conv2D(32, (3,3), activation='relu', input_shape=(224,224,3))) # Capa convocional con 32 filtros y tamaño de filtro 3x3
model1.add(MaxPooling2D((2,2))) # Capa de pooling con tamaño de ventana 2x2
model1.add(Conv2D(64, (3,3), activation='relu')) # Capa convocional con 64 filtros y tamaño de filtro 3x3
model1.add(MaxPooling2D((2,2))) # Capa de pooling con tamaño de ventana 2x2
model1.add(Flatten()) # Aplanar las imagenes para poder ser procesadas
model1.add(Dense(128, activation='relu')) # Capa densa con 128 unidades y función de activación relu
model1.add(Dense(2, activation='softmax')) # Capa densa con 2 unidades y función de activación softmax

# Compilar el modelo
model1.compile(optimizer='adam', loss='categorical_crossentropy', metrics=['accuracy'])

# Entrenar el modelo
history = model1.fit(data_gen_entrenamiento, epochs=50, validation_data=data_gen_pruebas,callbacks=[tconv])
```

Ilustración 28: Código Python, Modelo Convolucional, fuente propia

A continuación, se explica las líneas de código en detalle:

1. Crea una instancia de la clase Sequential de Keras, que es un contenedor para organizar capas de red neuronal.
2. Agrega una capa convolucional 2D a la red con 32 filtros, un tamaño de kernel de 3x3; la función de activación 'relu' y una forma de entrada de imágenes de tamaño 224x224 con 3 canales de color.
3. Agrega una capa de pooling (de sub muestreo) que reduce el tamaño espacial de la representación de entrada, en este caso; se usa una ventana de pooling de tamaño 2x2.
4. Agrega otra capa convolucional 2D a la red con 64 filtros y un tamaño de kernel de 3x3.

5. Agrega otra capa de pooling para reducir el tamaño espacial de la representación de entrada aún más.
6. Agrega una capa de aplanamiento, que transforma la representación de entrada tridimensional en un vector unidimensional; que puede ser procesado por capas densas.
7. Agrega una capa densa con 128 neuronas y la función de activación 'relu', esta capa toma el vector aplanado como entrada.
8. Agrega una capa densa final con 2 neuronas y la función de activación 'softmax', esta capa produce una salida de probabilidad para cada una de las dos clases.
9. Compila el modelo y define la función de pérdida, el optimizador y las métricas a utilizar durante el entrenamiento.
10. Entrena el modelo en los datos de entrenamiento generados por un generador de imágenes data_gen_entrenamiento, el modelo se entrena durante 50 épocas y se utiliza un conjunto de datos de prueba (data_gen_pruebas); para evaluar el rendimiento del modelo después de cada época. El argumento callbacks, se utiliza para pasar una lista de objetos de devolución de llamada; que se llamarán al final de cada época.

1.7. Entrenamiento de Modelo MoViNet

Este código desarrollado descarga el modelo MoViNet como una capa keras en versión streaming, el cual permite la recepción de frames para clasificar mientras recibe un Frame nuevo, este modelo está diseñado con redes convolucionales 3D las cuales permite manejar 3 dimensiones que serían el ancho de frame, el alto de frame y los frames de un vídeo esta sección de código fue adaptada del código de modo de uso proporcionado por TensorFlow "MoViNet; para el reconocimiento de acción de transmisión" (TensorFlow, 2023)

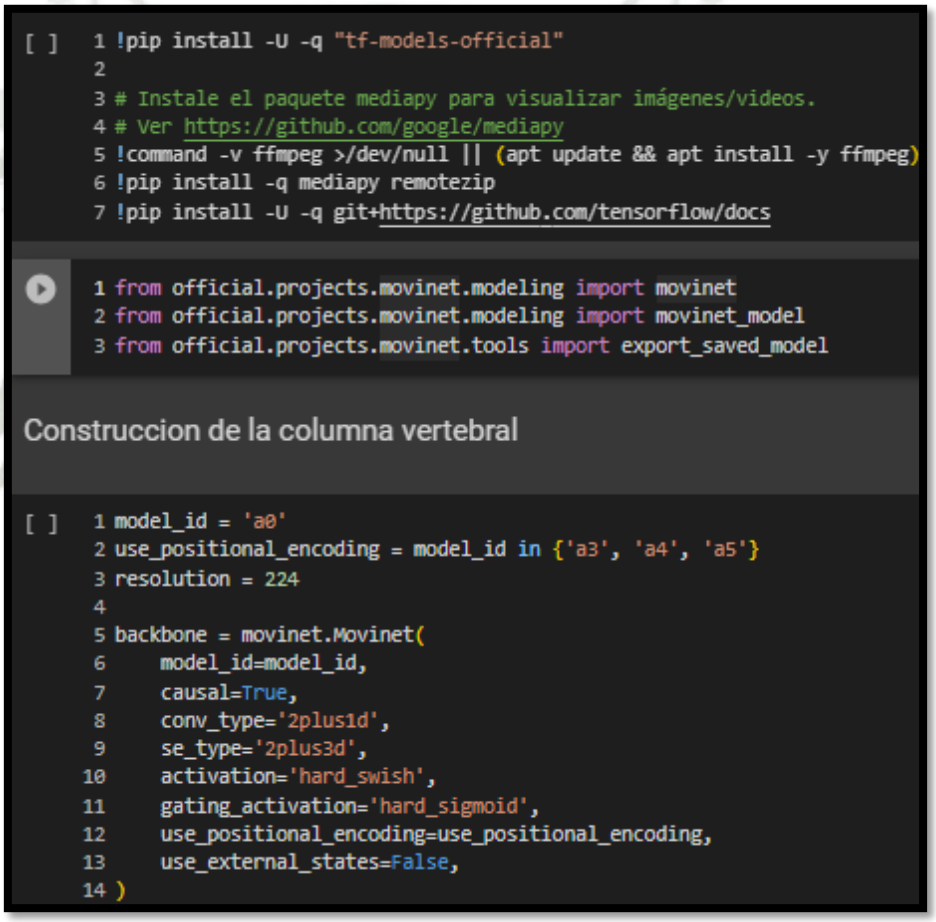


Ilustración 29: Código Python, Construcción de columna vertebral, fuente propia

A continuación, se describe el código:

- !pip install -U -q "tf-models-official": Instala o actualiza en silencio el paquete "tf-models-official", que contiene modelos y utilidades oficiales de TensorFlow; para tareas de aprendizaje automático.

- !command -v ffmpeg >/dev/null || (apt update && apt install -y ffmpeg): Comprueba si el comando ffmpeg está disponible en el sistema, si no lo está; actualiza las fuentes del paquete APT y luego instala ffmpeg para trabajar con archivos multimedia.
- !pip install -q mediapy remotezip: Instala en silencio los paquetes mediapy y remotezip, que se utilizan para visualizar imágenes/vídeos y trabajar con archivos ZIP remotos; respectivamente.
- !pip install -U -q git+https://github.com/tensorflow/docs: Instala o actualiza en silencio, la versión más reciente de la documentación de TensorFlow desde su repositorio en GitHub.
- A partir de aquí, el código importa módulos y funciones específicas relacionadas con el proyecto "MoViNet" en TensorFlow. En particular, se construye un objeto backbone de tipo movinet.Movinet con varias configuraciones, como el tipo de modelo, si se utiliza codificación posicional, resolución de entrada y activaciones

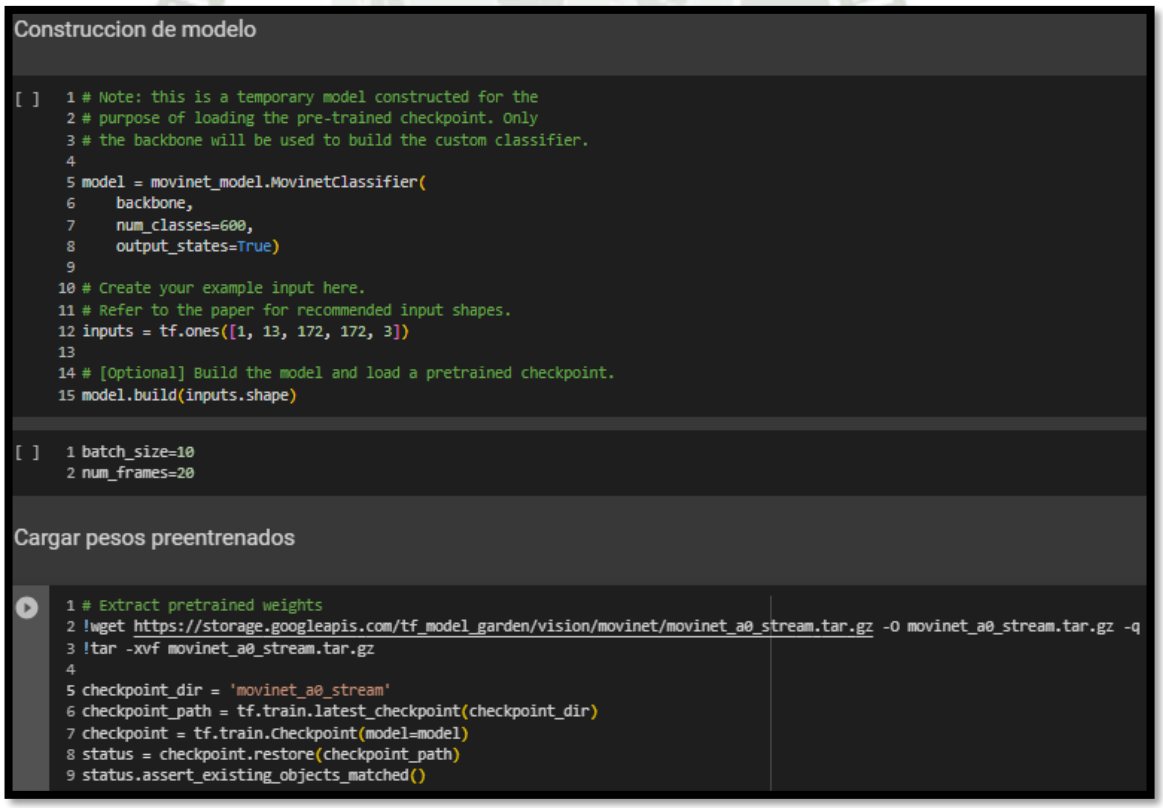


Ilustración 30: Código Python, Construcción del modelo para entrenamiento, fuente propia

A continuación, se describe el código:

1. model = movinet_model.MovinetClassifier: Se crea un objeto de modelo llamado model utilizando la clase movinet_model.MovinetClassifier, este modelo toma como entrada la columna vertebral (backbone); previamente definida y se configura para clasificar datos en 600 clases. También se especifica que se deben generar salidas intermedias del modelo (output_states=True), lo que significa que se obtendrán las representaciones intermedias del modelo además de la salida final.
2. inputs = tf.ones([1, 13, 172, 172, 3]): Se crea un tensor de ejemplo (inputs), que representa un único lote de datos con una forma de (1, 13, 172, 172, 3); este tensor sirve como entrada de ejemplo para construir el modelo.
3. model.build(inputs.shape): Se construye el modelo utilizando la forma del tensor de entrada de ejemplo inputs, esto configura las dimensiones internas del modelo en función de la entrada proporcionada.
4. batch_size=10 y num_frames=10: Estas variables se definen con valores específicos de tamaño de lote y número de fotogramas
5. !wget : Se descargan archivos de pesos pre entrenados de un servidor remoto, los pesos pre entrenados se almacenan en un archivo llamado movinet_a0_stream.tar.gz; que se extrae a un directorio llamado movinet_a0_stream. Estos pesos pre entrenados se utilizarán para inicializar el modelo model.
6. checkpoint_path = tf.train.latest_checkpoint(checkpoint_dir): Se obtiene la ruta del último punto de control (checkpoint), guardado en el directorio donde se extrajeron los pesos pre entrenados.
7. checkpoint = tf.train.Checkpoint(model=model): Se crea un objeto de punto de control (checkpoint), que contiene el modelo model que se construyó previamente.
8. status = checkpoint.restore(checkpoint_path): Se restauran los pesos pre entrenados en el modelo model, desde el punto de control obtenido en el paso anterior.
9. status.assert_existing_objects_matched(): Se verifica que los objetos existentes en el punto de control coincidan con los objetos del modelo

cargado, esto ayuda a garantizar que la restauración de pesos se haya realizado correctamente.

constructor del modelo

```
[ ]  1 def build_classifier(batch_size, num_frames, resolution, backbone, num_classes):
     2   """Builds a classifier on top of a backbone model."""
     3   model = movinet_model.MovinetClassifier(
     4       backbone=backbone,
     5       num_classes=num_classes)
     6   model.build([batch_size, num_frames, resolution, resolution, 3])
     7
     8   return model
     9
    10 # Construct loss, optimizer and compile the model
    11 with distribution_strategy.scope():
    12   model = build_classifier(batch_size, num_frames, resolution, backbone, 2)
    13   loss_obj = tf.keras.losses.SparseCategoricalCrossentropy(from_logits=True)
    14   optimizer = tf.keras.optimizers.Adam(learning_rate = 0.001)
    15   model.compile(loss=loss_obj, optimizer=optimizer, metrics=['accuracy'])
```

Train the model

```
1 results = model.fit(train_ds,
2                     validation_data=val_ds,
3                     epochs=4,
4                     validation_freq=1,
5                     verbose=1,
6                     callbacks=[cp_callback])
```

Ilustración 31: Código Python, Construcción del modelo para entrenamiento, fuente propia

A continuación, se describe el código:

1. def build_classifier: Se define una función llamada build_classifier que toma varios argumentos, como el tamaño del lote (batch_size), el número de fotogramas (num_frames), la resolución de entrada (resolution), la columna vertebral del modelo (backbone), y el número de clases (num_classes). Esta función construye un modelo de clasificación basado en la columna vertebral proporcionada y devuelve el modelo construido.
2. model = build_classifier: Se llama a la función build_classifier para construir el modelo de clasificación. Se utiliza la estrategia de distribución (distribution_strategy.scope()), para especificar cómo se realizará la distribución de entrenamiento; lo que puede ser útil en entornos de entrenamiento distribuido.

3. loss_obj = tf.keras.losses.SparseCategoricalCrossentropy: Se define un objeto de función de pérdida (loss_obj) que utiliza la entropía cruzada categórica escasa como función de pérdida para la clasificación. La opción from_logits=True indica que se espera que las salidas del modelo sean logaritmos no normalizados antes de aplicar la función de pérdida.
4. optimizer = tf.keras.optimizers.Adam: Se define un optimizador Adam con una tasa de aprendizaje de 0.001, para ajustar los pesos del modelo durante el entrenamiento.
5. model.compile: Se compila el modelo con la función de pérdida, el optimizador y la métrica de precisión ('accuracy') especificados; esto configura el modelo para el entrenamiento.
6. results = model.fit: Se inicia el entrenamiento del modelo utilizando el método fit, se proporciona el conjunto de datos de entrenamiento (train_ds) y el conjunto de datos de validación (val_ds) como entrada. El modelo se entrena durante 4 épocas (epochs=4), con validación después de cada época (validation_freq=1). Además, se utiliza el callback cp_callback que no se muestra en el código proporcionado, pero probablemente se utilice para guardar puntos de control; durante el entrenamiento.

1.8. Resultados de entrenamiento

1.8.1. Métricas Usadas

Epoch_Accuracy

En TensorBoard, "epoch_accuracy" se refiere a la métrica de precisión (accuracy en inglés), calculada durante el entrenamiento de un modelo de aprendizaje automático; en una sola época (epoch en inglés).

La precisión es una medida comúnmente utilizada para evaluar la calidad del modelo de aprendizaje automático en la tarea de clasificación. Se calcula dividiendo el número de predicciones correctas realizadas por el modelo, sobre el conjunto de datos de prueba; por el número total de ejemplos en el conjunto de datos de prueba.

Durante el entrenamiento del modelo, se puede calcular la precisión en cada época para evaluar cómo está mejorando el modelo; en la tarea de clasificación a medida que se ajustan los pesos y sesgos de las capas de la red neuronal. Esta métrica se puede visualizar en TensorBoard como "epoch_accuracy", en una gráfica que muestra la precisión en función del número de épocas de entrenamiento.

Epoch_loss

En TensorBoard, "epoch_loss" se refiere a la métrica de pérdida (loss en inglés), calculada durante el entrenamiento de un modelo de aprendizaje automático en una sola época (epoch en inglés).

La pérdida es una medida de qué tan malo es el modelo en la tarea que está tratando de realizar. Durante el entrenamiento, el objetivo es minimizar la pérdida del modelo, ajustando los pesos y sesgos de las capas de la red neuronal. Por otro lado, la pérdida se calcula mediante una función de costo; que compara las predicciones del modelo con las etiquetas verdaderas de los datos de entrenamiento.

Durante el entrenamiento del modelo, se puede calcular la pérdida en cada época para evaluar cómo está mejorando el modelo en la tarea de clasificación a medida que se ajustan los pesos y sesgos de las capas de la red neuronal.

Esta métrica se puede visualizar en TensorBoard como "epoch_loss" en una gráfica que muestra la pérdida en función del número de épocas de entrenamiento. Una disminución en la pérdida a lo largo del tiempo indica que el modelo está mejorando su capacidad para realizar la tarea de clasificación.

1.8.2. Modelo denso

Epoc_Acuracy

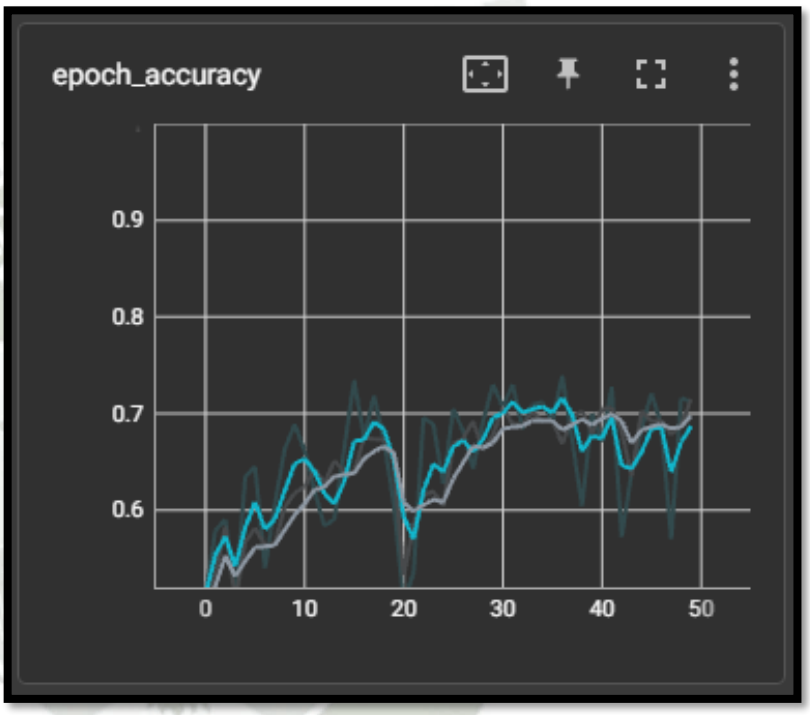


Ilustración 32: Grafico TensorBoard, Modelo Denso,
Fuente: Propia

De acuerdo a la gráfica, la capacidad del modelo para dar predicciones correctas es muy inestable a medida que se incrementan las épocas, obteniendo un valor máximo de 0.69 o 69% en los datos de entrenamiento y de 0.70 o 70% en los datos de prueba para la cantidad de aciertos en la época 50

Epoc_loss

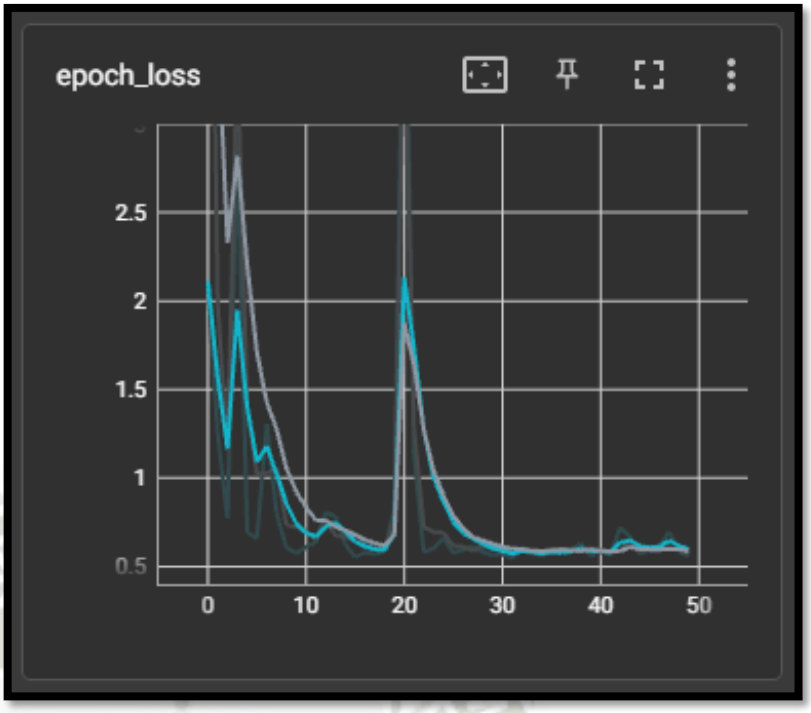


Ilustración 33: Grafico TensorBoard, Modelo Denso,
Fuente: Propia

Según se aprecia en la gráfica, la capacidad del modelo para ejecutar la tarea de clasificación de acción delictiva o no delictiva a medida que se incrementan las épocas alcanza la estabilidad en un valor, siendo muy inestable en las épocas cercanas a 20 y teniendo un valor final de 0.5690 o 56.90% en los datos de entrenamiento y de 0.5696 o 56.96% en los datos de prueba para la perdida en la época 50.

1.8.3. Modelo Red Neuronal Convolucional

Epoc_Acuracy

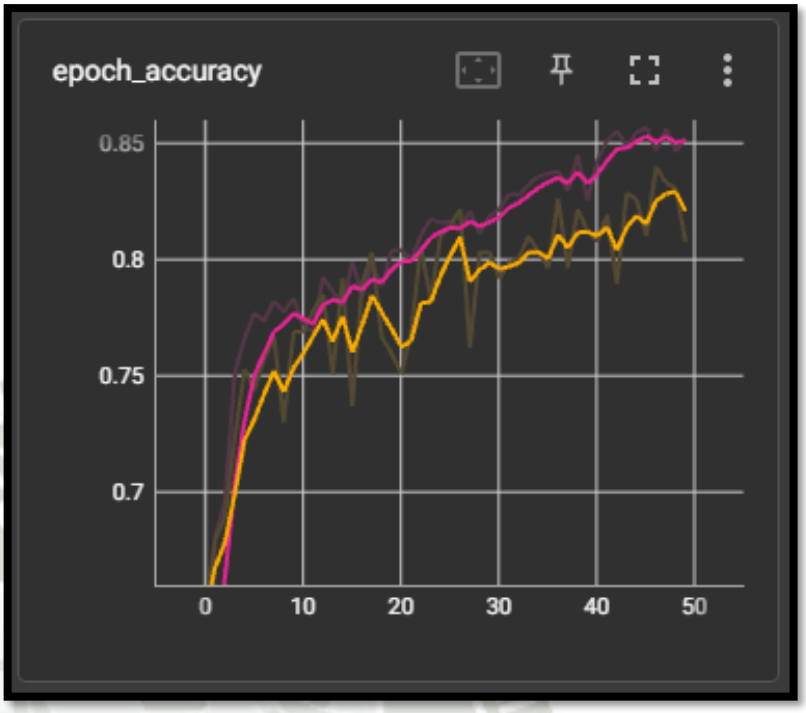


Ilustración 34: Grafico TensorBoard, Modelo Convolucional,
Fuente: Propia

Como se puede observar en la gráfica, la capacidad del modelo para dar predicciones va creciendo a medida que se incrementan las épocas; obteniendo un valor máximo de 0.85 o 85% de acierto para datos de entrenamiento y 0.82% o 82% para datos de prueba en la época 50.

Epoc_loss

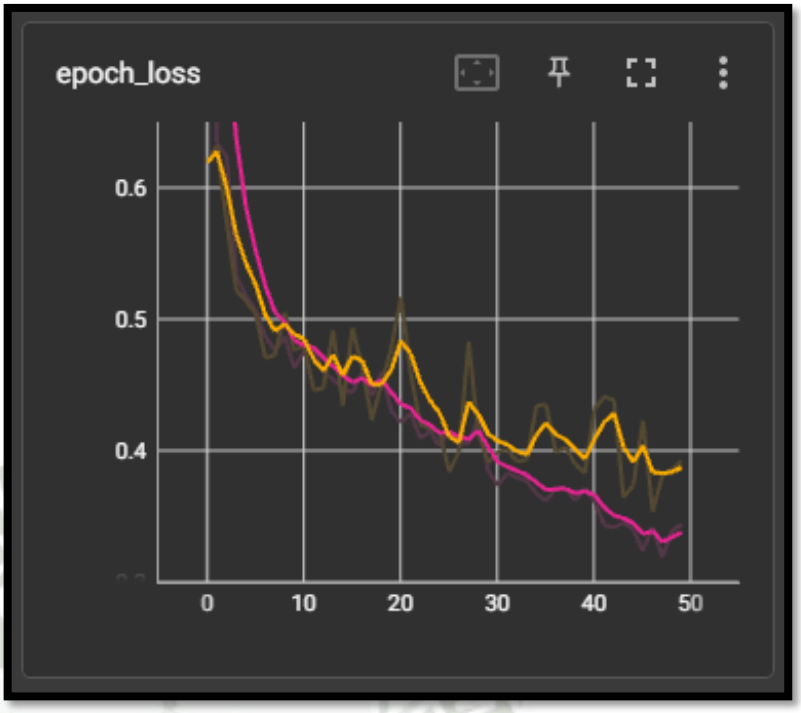


Ilustración 35: Grafico TensorBoard, Modelo Convolucional,
Fuente: Propia

Como se puede observar en la gráfica, la capacidad del modelo para ejecutar la tarea de clasificación de acción delictiva o no delictiva, reduce su pérdida a medida que se incrementan las épocas; teniendo un valor final de 0.34 o 34% para los datos de validación y de 0.39 o 39% para los datos de prueba en la época 50.

1.8.4. Modelo MoViNet

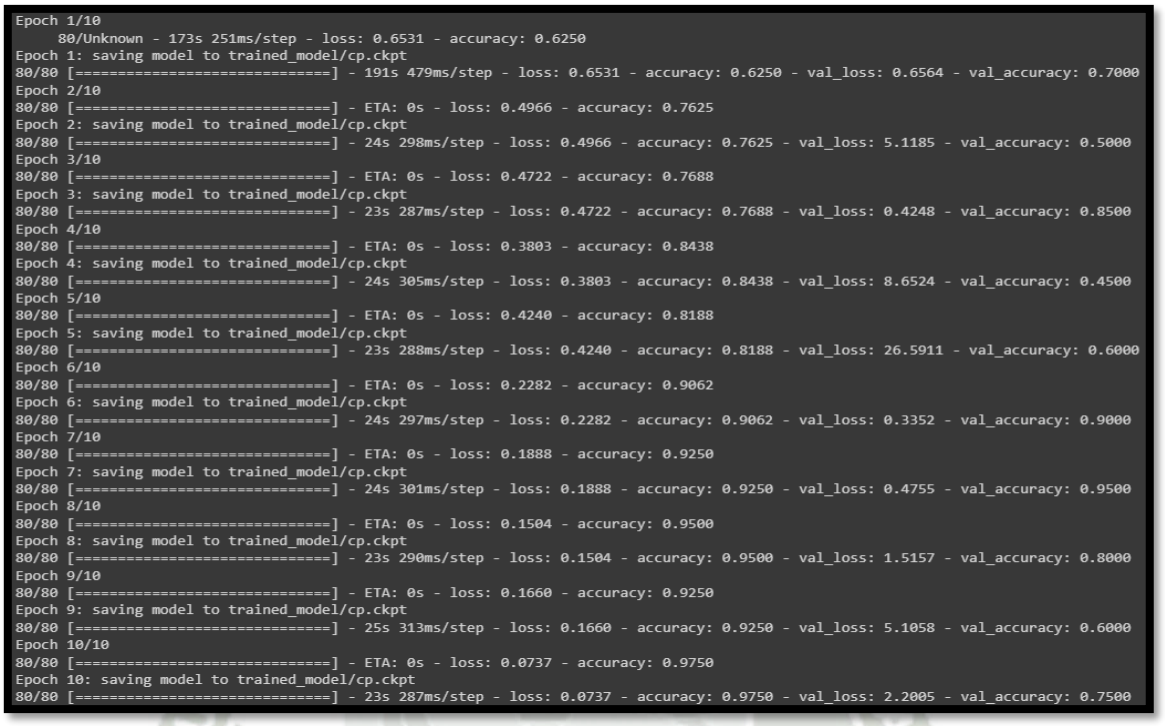


Ilustración 36: Captura de pantalla, Modelo MoViNet entrenamientos por épocas, Fuente: Propia

Como podemos observar al realizar el entrenamiento del modelo MoViNet en la primera época de entrenamiento obtenemos un valor alto como el 70% y luego con la siguiente cae a 50% en la tasa de acierto, pero en el valor de pérdida obtenemos números altos esto se debe a que el modelo no está generalizando bien aún los datos nuevos entregados por el conjunto de validación, posteriormente este valor va decreciendo y aumentando lo que significa que el modelo se está ajustando hasta la época 10 en esta época obtenemos:

- **80/80 [==============================]:** Esta parte del registro muestra el progreso del proceso de entrenamiento. Los números 80/80 representan el número de lotes procesados durante esta época. En este caso, parece que hay 80 lotes en el conjunto de datos de entrenamiento y todos se han procesado en esta época.

- **Pérdida 0.0737:** Es la pérdida de entrenamiento, que es una medida de cuán bien está funcionando el modelo en los datos de entrenamiento. El objetivo

suele ser minimizar este valor, en este caso, la pérdida de entrenamiento es 0.0737.

- **Exactitud 0.9750:** Se refiere a la exactitud de entrenamiento, que es una medida de cuántos de los ejemplos de entrenamiento están siendo clasificados correctamente por el modelo. En este caso, la exactitud de entrenamiento es del 97.50%, lo cual es bastante alto.

- **val_pérdida 2.2005:** Esta es la pérdida de validación, que es una medida de cuán bien está funcionando el modelo en un conjunto de datos de validación separado que no ha visto durante el entrenamiento. La pérdida de validación es 2.2005, lo que indica cuán bien generaliza el modelo a datos no vistos. Una pérdida de validación más baja generalmente es mejor.

- **val_exactitud 0.7500:** Es la exactitud de validación, que es una medida de cuán bien el modelo está clasificando ejemplos en el conjunto de datos de validación. En este caso, la exactitud de validación es del 75.00%.

1.9. Comparación de modelos

1.9.1. Tabla Comparativa

Modelo	Comportamiento de grafica	Acierto en Época Final		Perdida en Época Final	
Denso	Inestable Creciente Decreciente	0.69/69%	0.70/70%	5.69	5.6
Convolucional	Creciente Decreciente	0.85/85%	0.82/82%	3.9	3.4
MoViNet	No Observable	0.97/97%	0.75/75%	0.07	2.20

Tabla 3 Tabla comparativa de modelos entrenados

1.9.2. Resultado de comparación.

El primer modelo es un modelo Denso, que tiene un comportamiento inestable en términos de su gráfica; su precisión en la época 50 es de 0.69 o 69% y su pérdida es de 0.5690 o 56.90%.

El segundo modelo es un modelo Convolucional, que tiene una gráfica que aumenta y luego disminuye, este modelo tiene un mejor rendimiento que el modelo Denso, con una precisión de 0.85 o 85% y una pérdida de 0.39 o 39% en la época 50.

En las pruebas realizadas al modelo mediante la entrega de imágenes para su clasificación, observamos que de 10 imágenes clasificadas acertó las 10 indicándonos un 100% de detecciones acertadas, pero al realizarse su uso para la clasificación de acciones sobre distintos frames de un video de una acción positiva y luego negativa; observamos que la cantidad de falsos positivos incrementa entre 6 a 9 falso positivos de cada 10 frames analizados siendo el 80% de detecciones erradas.

El tercer modelo es un modelo MoViNet, no tiene gráfica debido a que los valores de entrenamiento se realizan por paquetes de 80 vídeos y 10 frames de cada uno de ellos; pero obtenemos los valores de acierto tanto por el grupo de entrenamiento y por el grupo de validación.

El mencionado modelo tiene una precisión de 0.75 o 75% y una pérdida de 2.20 en la época 10, esto hace que este modelo sea mejor en comparación con el modelo convolucional 2D, debido a la capacidad que tiene para trabajar sobre la dimensión temporal, esto nos permite analizar más frames para determinar la acción realizada; en las pruebas realizadas de 10 videos clasificados se clasificaron 7 correctamente indicándonos un acierto del 70 % y 30% de detección de falsos positivos.

En general, parece que el modelo MoViNet tiene un mejor rendimiento que el modelo Denso, y agregar la capacidad de análisis temporal; lo que nos permite analizar varios frames en comparación con el modelo Convolucional.

2. Programación del módulo prototipo disuasorio ESP32-CAM

2.1. Librerías Utilizadas

- La librería <WiFi>, permite conectarse a una red inalámbrica Wi-Fi y enviar y recibir datos a través de ella. Proporciona funciones para configurar y conectar el módulo Wi-Fi a una red, así como para enviar y recibir datos a través de ella.

- La librería <AsyncTCP.h>, proporciona una implementación asíncrona del protocolo de control de transmisión (TCP) para Arduino. Esta librería permite la comunicación asíncrona con dispositivos remotos, lo que significa que se pueden enviar y recibir datos; sin tener que esperar a que se completen las operaciones anteriores.
- La librería <ESPAsyncWebServer.h>, es una biblioteca para crear servidores web en el ESP32; de forma asíncrona. Permite crear y gestionar rutas para procesar solicitudes HTTP y enviar respuestas, es útil para construir aplicaciones web complejas y para controlar el ESP32 desde una interfaz web.
- La librería ArduinoWebsockets.h, proporciona una forma de comunicarse con otros dispositivos a través del protocolo WebSocket. WebSocket es un protocolo de comunicación bidireccional, que se ejecuta sobre TCP y se utiliza para intercambiar datos en tiempo real entre clientes y servidores.
  Esta librería permite al ESP32 establecer una conexión WebSocket y enviar y recibir datos en tiempo real.
- Las librerías "soc/soc.h" y "soc/rtc_cntl_reg.h", se utilizan para solucionar problemas de compatibilidad entre diferentes versiones de la biblioteca ESP32 y el código de Arduino. La inclusión de estas librerías, permite que el código se ejecute correctamente en diferentes versiones de hardware y software.

2.2. Módulo de cámara del ESP32-CAM

En esta sección del código describiremos los pasos realizados para la configuración e inicio de la cámara, para posteriormente capturar imágenes con ella; lo cual nos permitirá configurar esta tarjeta como una cámara de videovigilancia IP.

2.2.1. Configuración de la cámara

```
//************************************************************************
//configuracion de camara
typedef struct {
        camera_fb_t * fb;
        size_t index;
} camera_frame_t;

#define PART_BOUNDARY "123456789000000000000987654321"
static const char* STREAM_CONTENT_TYPE = "multipart/x-mixed-replace;boundary=" PART_BOUNDARY;
static const char* STREAM_BOUNDARY = "\r\n--" PART_BOUNDARY "\r\n";
static const char* STREAM_PART = "Content-Type: %s\r\nContent-Length: %u\r\n\r\n";
static const char * JPG_CONTENT_TYPE = "image/jpeg";


// pines de camara
#define PWDN_GPIO_NUM     32
#define RESET_GPIO_NUM    -1
#define XCLK_GPIO_NUM      0
#define SIOD_GPIO_NUM     26
#define SIOC_GPIO_NUM     27

#define Y9_GPIO_NUM       35
#define Y8_GPIO_NUM       34
#define Y7_GPIO_NUM       39
#define Y6_GPIO_NUM       36
#define Y5_GPIO_NUM       21
#define Y4_GPIO_NUM       19
#define Y3_GPIO_NUM       18
#define Y2_GPIO_NUM        5
#define VSYNC_GPIO_NUM    25
#define HREF_GPIO_NUM     23
#define PCLK_GPIO_NUM     22
camera_fb_t * fb = NULL;
```

Ilustración 37: Código Arduino, Pines de cámara onvif,
Fuente: propia

La cámara se configura para capturar imágenes y transmitirlas en formato JPEG a través de un flujo multipartido (multipart/x-mixed-replace). Para llevar a cabo esta tarea, hemos definido algunas variables que serán utilizadas al transmitir el flujo de frames; al servidor web de nuestra cámara IP.

Es necesario definir los pines de la cámara, ya que la tarjeta ESP32-CAM puede venir con diferentes modelos de cámaras, en nuestro caso, utilizamos el modelo OV2640. Los pines internos conectados al puerto de la cámara se obtuvieron del datasheet, proporcionado por el fabricante de la tarjeta ESP32-CAM (AI-Tinker, 2017).

2.2.2. Función initCamara ()

Esta función se encarga de inicializar la cámara utilizando los pines previamente definidos en el código. En esta sección, se configuró el tamaño del fotograma de captura de la cámara, la calidad de la imagen y la cantidad de fotogramas que se enviarán por solicitud. Además, esta función proporcionará un mensaje de error en caso de que la cámara no se inicie correctamente.

```
  //init with high specs to pre-allocate larger buffers
  if(psramFound()){
    config.frame_size = FRAMESIZE_SVGA;
    config.jpeg_quality = 10;
    config.fb_count = 2;
  } else {
    config.frame_size = FRAMESIZE_SVGA;
    config.jpeg_quality = 12;
    config.fb_count = 1;
  }
  // camera init
  esp_err_t err = esp_camera_init(&config);
  if (err != ESP_OK) {
    Serial.printf("Camera init failed with error 0x%x", err);
    return;
  }

}
```

Ilustración 38: Código Arduino, inicio de cámara,
Fuente: propia

2.3. Servidor Web Asíncrono

2.3.1. Configuraciones Iniciales

Para el desarrollo de este proyecto se configuró el puerto 80, como puerto de acceso de las peticiones HTTP y para establecer una conexión WebSocket a través de la biblioteca " ESPAsyncWebServer.h".

```
//IP STATICA
IPAddress local_IP(192, 168, 137, 200);
IPAddress gateway(192, 168, 137, 1);
IPAddress subnet(255, 255, 0, 0);
//CREDENCIALES WIFI
const char* ssid = "LAPRENE";
const char* password = "Akamegakill1234";
//*****************************************************************************
// CONFIGURACIONES INICIALES SERVIDOR WEB
//*****************************************************************************

// Servidor  AsyncWebServer en puerto http 80
AsyncWebServer server(80);
AsyncWebSocket wc("/wc");
```

Ilustración 39: Código Arduino, Configuraciones Iniciales,
Fuente: Propia

Las tres primeras líneas configuran la dirección IP estática, a través de la cual se accederá a la cámara web ESP32-CAM y donde se ubicará el servidor de vídeo y WebSocket. Las siguientes líneas contienen las credenciales de la IP que nos permitirán conectarnos a la red WiFi.

En la siguiente línea, "AsyncWebServer server(80);", se crea una instancia de la clase "AsyncWebServer"; que se utilizará para manejar las solicitudes HTTP entrantes. Esta instancia se denomina "server" y está configurada para escuchar en el puerto 80, que es el puerto estándar para las solicitudes HTTP.

También, en este mismo puerto se ubicará el servidor WebSocket encargado de recibir la clasificación, determinando si es delictiva o no. Dicho WebSocket responderá a la dirección ws://hostnameip/wc.

2.3.2. WebSockets

2.3.3. Función AsyncJpegStreamResponse

Esta función implementa una respuesta asincrónica personalizada para transmitir imágenes JPEG capturadas por la cámara ONVIF y enviarlas a través del servidor web. La clase AsyncJpegStreamResponse maneja la lógica para obtener y enviar los datos de la imagen JPEG al cliente de manera eficiente

```
class AsyncJpegStreamResponse: public AsyncAbstractResponse {
    private:
        camera_frame_t _frame;
        size_t _index;
        size_t _jpg_buf_len;
        uint8_t * _jpg_buf;
        uint64_t lastAsyncRequest;
    public:
        AsyncJpegStreamResponse(){
            _callback = nullptr;
            _code = 200;
            _contentLength = 0;
            _contentType = STREAM_CONTENT_TYPE;
            _sendContentLength = false;
            _chunked = true;
            _index = 0;
            _jpg_buf_len = 0;
            _jpg_buf = NULL;
            lastAsyncRequest = 0;
            memset(&_frame, 0, sizeof(camera_frame_t));
        }
        ~AsyncJpegStreamResponse(){
            if(_frame.fb){
                if(_frame.fb->format != PIXFORMAT_JPEG){
                    free(_jpg_buf);
                }
                esp_camera_fb_return(_frame.fb);
            }
        }
```

Ilustración 40: Código Arduino, fragmento de función AsynJpegStreamResponse ,
Fuente:Propia

2.3.4. Inicio de servidor Web

Una vez que se hayan realizado las configuraciones iniciales del prototipo del módulo disuasorio ESP32-CAM y el inicio de los módulos de cámara y disuasorios, se inicia el servidor definiendo las rutas de solicitudes HTTP:

```
//servidor web
server.on("/jpg", HTTP_GET, [] (AsyncWebServerRequest *request) {
   serveJpg(request);
 });
server.on("/streaming", HTTP_GET, [](AsyncWebServerRequest *request){
   serverstreaming(request);
 });
 server.on("/", HTTP_GET, serveIndex);
 server.begin();
 wc.onEvent(websoketclasificador);
 server.addHandler(&wc);
```

Ilustración 41: Código Arduino, Rutas de HTTP,
Fuente propia

Rutas HTTP:

- La primera ruta "/jpg" está configurada para manejar solicitudes HTTP GET y llama a la función serveJpg(request), cuando se recibe una solicitud en esta ruta.
- La segunda ruta "/streaming" también está configurada para manejar solicitudes HTTP GET y llama a la función serverstreaming(request), cuando se recibe una solicitud en esta ruta.
- La tercera ruta "/" está configurada para manejar solicitudes HTTP GET y llama a la función serveIndex, cuando se recibe una solicitud en la raíz del servidor.

Comandos de inicio

- Se inicia el servidor web utilizando server.begin().
- Se configura un manejador de eventos WebSocket con wc.onEvent(websoketclasificador). Esto indica que cuando se produzca un evento WebSocket, se llamará a la función websoketclasificador para manejarlo.
- Se agrega el manejador de WebSocket al servidor web con server.addHandler(&wc). Esto conecta el servidor WebSocket al

servidor web, permitiendo la comunicación bidireccional con clientes a través de WebSocket.

3. Servidor TensorFlow Serving para el modelo clasificador

3.1. Ejecución de TensorFlow Serving con Docker

Para poder ejecutar un servidor de TensorFlow Serving se requiere de un sistema Linux, debido a que Tensorflow Serving está basado en Linux; para realizar las pruebas del prototipo. Para el desarrollo de este trabajo de investigación se recurrió a un aplicativo de contenedores como Docker que nos permitió crear un entorno aislado, portable, gestionable, escalable y compatible con Windows.

3.1.1. Comandos de instalación y pruebas de funcionamiento

Los siguientes comandos fueron ejecutados en el terminal de comandos de Windows, este sistema operativo cuenta ya con la aplicación Docker y Git; también con una carpeta donde estarán los modelos para este trabajo estará en la ubicación "D:\mentrenados"

- Descarga de imagen TensorFlow Serving:
  - docker pull tensorflow/serving
- Clonación de modelos de prueba:
  - cd D:
  - cd / mentrenados
  - git clone https://github.com/tensorflow/serving
- Creación de variable demo
  - Set-Variable -Name "TESTDATA" -Value "$(pwd)/serving/tensorflow_serving/servables/tensorflow/testdata"
- Ejecución de Tensorflow Serving
  - docker run -t --rm -p 8501:8501 -v "$TESTDATA/saved_model_half_plus_two_cpu:/models/half_plus_two" -e MODEL_NAME=half_plus_two tensorflow/serving
- Respuesta del modelo prueba
  - curl -d '{"instances": [1.0, 2.0, 5.0]}' \
    -X POST http://localhost:8501/v1/models/half_plus_two:predict

Si la ejecución de los comandos fue correcta, el servidor responderá con la respuesta "Returns => {"predictions": [2.5, 3.0, 4.5]}" que sería el resultado de predicción del modelo demo, lo cual nos mostraría que la instalación fue correcta; es preciso indicar que los comandos anteriores fueron modificados para su uso en Windows, ya que estos se encuentran dentro de la guía de TensorFlow para la instalación de TensorFlow Serving con docker (TensorFlow, 2021)

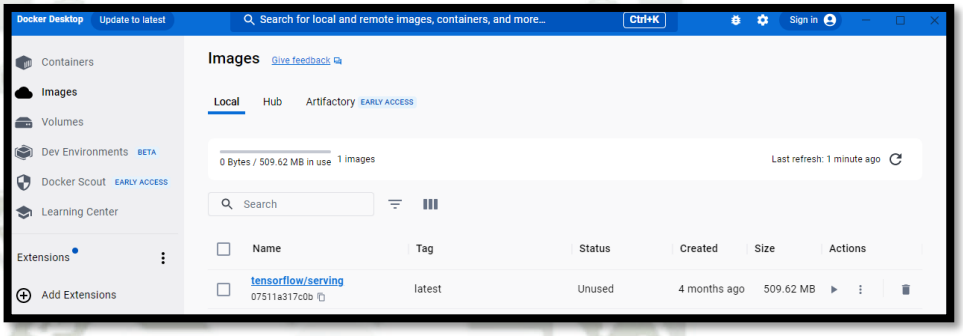


Ilustración 42: Captura de pantalla, Tensorflow Serving en docker,
Fuente: Propia

3.2. Publicación de modelo de clasificación de acciones delictivas

Luego de entrenado y probado el modelo de clasificación MoViNet este será publicado con TensorFlow serving para que pueda recibir solicitudes de clasificación en forma de tensores de 5 dimensiones: tamaño_lote, núm_frames, altura_píxeles, ancho_píxeles, Canal de colores.

3.2.1. Comandos para la publicación del modelo

Los siguientes comandos fueron ejecutados en el terminal de comandos de Windows, este sistema operativo cuenta ya con la aplicación Docker y la imagen de TensorFlow Serving también con una carpeta donde estará el modelo ya entrenado para este trabajo, la carpeta donde está el modelo es: "D:\mentrenados\MODELOV5".

- Creación de variable de ubicación del modelo:
  - Set-Variable -Name "MODMovi" -Value "D:\mentrenados\MODELOV5"
- Cargar de modelo a Tensorflow Serving

- docker run -t --rm -p 8501:8501 -v "$MODMovi/MoViNet:/models/MoViNet" -e MODEL_NAME=MoViNet tensorflow/serving

Con estos comandos el modelo entrenado para este trabajo se encontrará publicado y en ejecución a la espera de solicitudes de clasificación en la dirección: http://localhost:8501/v1/models/MoViNet:predict

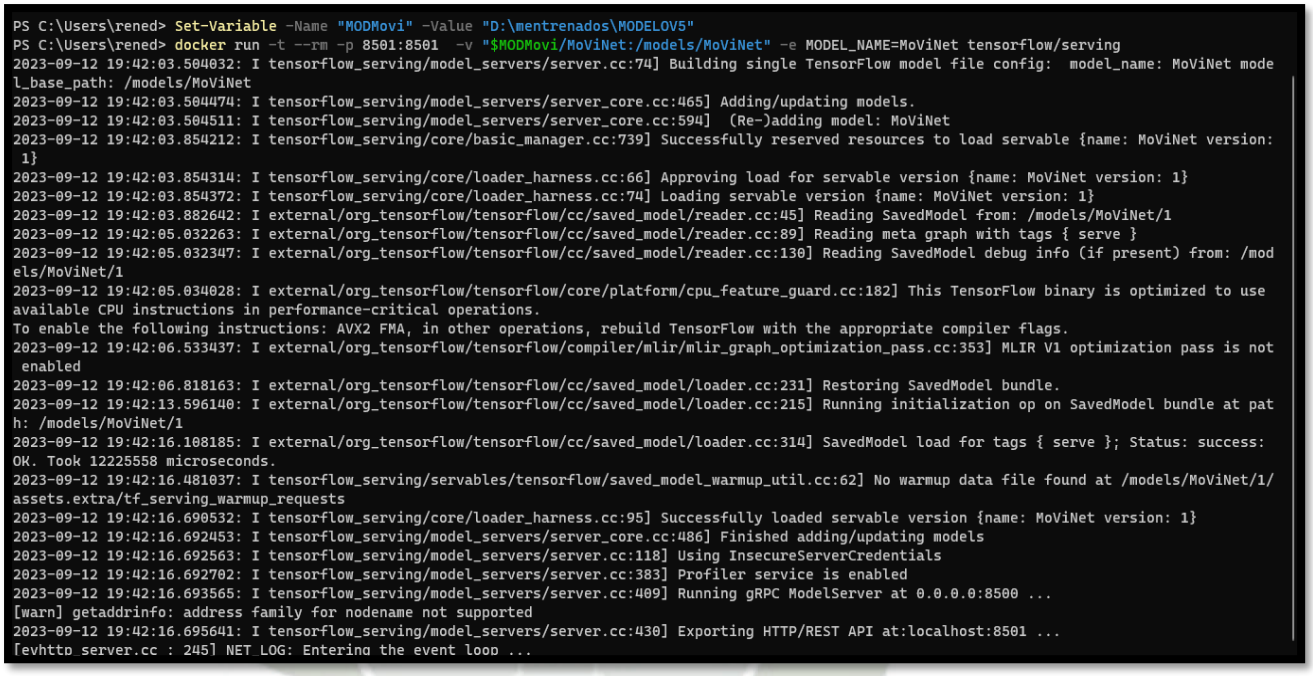


Ilustración 43: Captura de Pantalla, Terminal de Windows publicación de modelo, Fuente propia

4. Interfaz prototipo del sistema clasificador

Se codificó esta interfaz en HTML y JavaScript con el propósito de facilitar su utilización en cualquier navegador web. Esto posibilita un acceso más sencillo por parte de los usuarios, ya que solo se requiere un navegador web para llevar a cabo el control, visualización y clasificación de cámaras IP.

La funcionalidad de esta interfaz incluye la configuración de hasta cuatro cámaras, su conexión, la visualización de las imágenes de las cámaras y la realización de clasificaciones, que inicialmente están configuradas como "No Delictivas".

El código ha sido diseñado de manera que sea compatible con TensorFlow.js, lo que nos permite generar los tensores que serán enviados al servidor TensorFlow Serving

para llevar a cabo la clasificación mediante el modelo previamente publicado. Crea una interfaz de usuario para la visualización y clasificación de cámaras IP.

4.1. Estructura de interfaz web

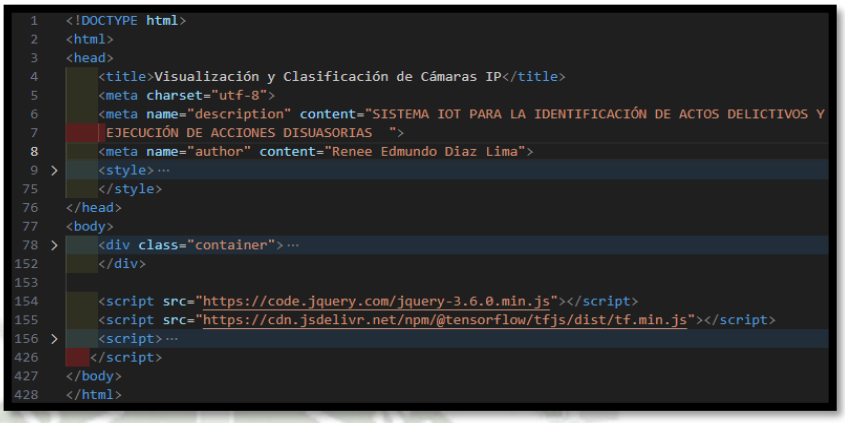


Ilustración 44: Captura de Pantalla, visual estudio code sección HTML, Fuente propia

- Metadatos y título: El código define el título de la página y algunos metadatos como el autor y una breve descripción.
- Estilos CSS: Se incluyen estilos CSS en la sección <style> para dar formato y diseño a la página. Estos estilos definen la apariencia de elementos como la barra lateral, las cámaras y los botones.
- Estructura del cuerpo de la página: La página tiene una estructura principal con dos áreas principales: una barra lateral y un contenedor para las cámaras.
- Barra lateral: La barra lateral contiene opciones de configuración para las cámaras. Cada cámara tiene un campo de entrada de texto, un botón "Conectar" y un botón "Mostrar Webcam" (solo para la cámara 1). También hay botones para iniciar y detener la clasificación.
- Resultados de predicción: La barra lateral muestra los resultados de la clasificación para cada cámara. Cada cámara tiene una sección que muestra el nombre de la cámara y una predicción (inicialmente establecida en "No Delictivo").
- Contenedor de cámaras: En el contenedor de cámaras, se muestran las vistas de las cámaras. Cada cámara tiene su propio cuadro con el nombre de la cámara.
- Scripts externos: Se cargan dos scripts externos al final del documento. Uno es jQuery y el otro es TensorFlow.js.

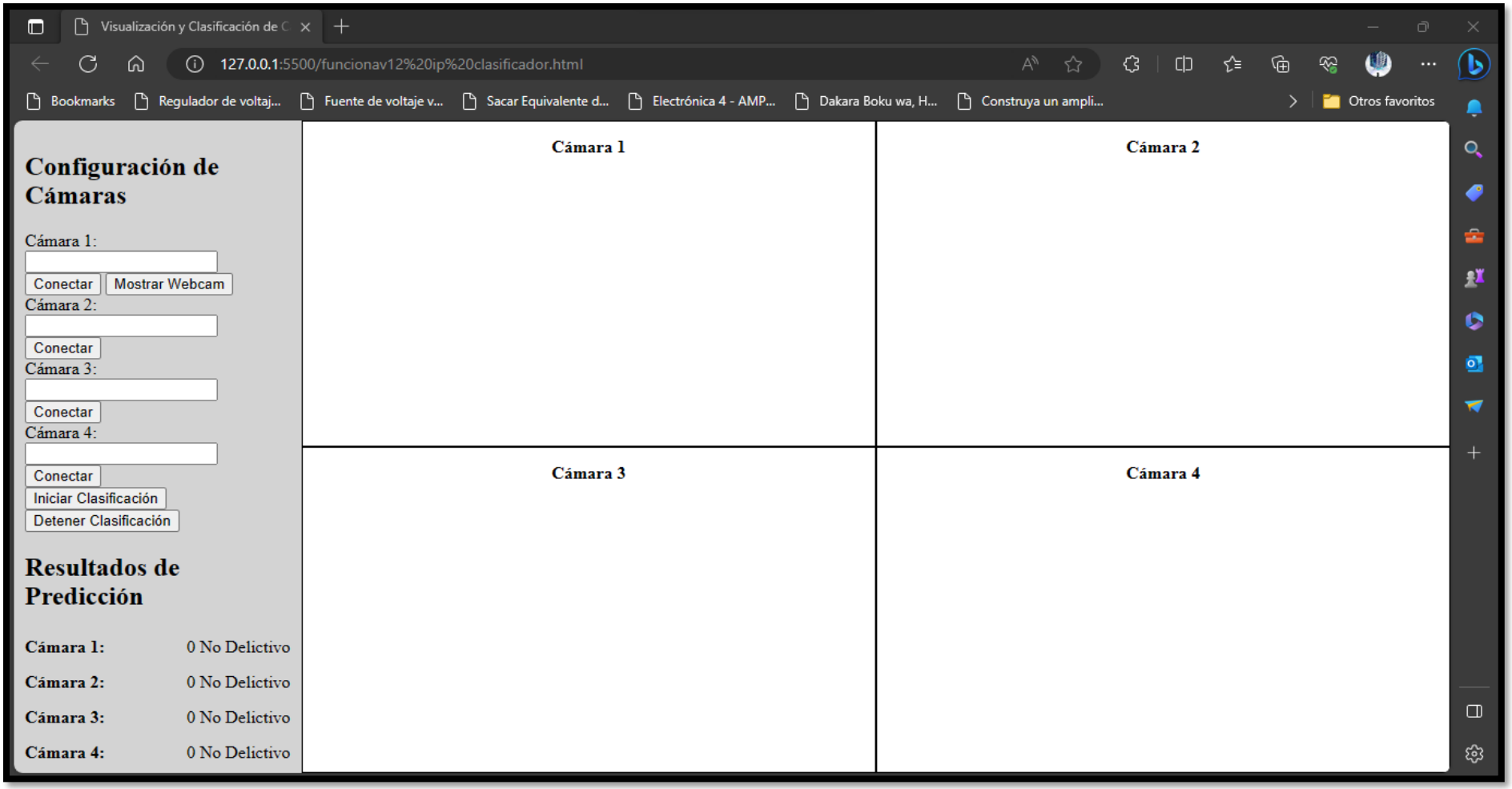


Ilustración 45: Captura de Pantalla, Estructura Pagina WEB,

Fuente propia

4.2. Script de conexión con las cámaras IP

La función script connectCamera(cameraId) se encarga de conectar una cámara IP y mostrar su vista en la página web

```
//-----------------------------------------------------------------------------------------------------------
//Conexion de las camaras
//-----------------------------------------------------------------------------------------------------------
    function connectCamera(cameraId) {
        var cameraInput = document.getElementById(cameraId);
        var cameraUrl = cameraInput.value;

        if (cameraUrl !== "") {
            var cameraView = document.getElementById("cameraView" + cameraId.slice(-1));
            cameraView.innerHTML = "<img src='" + cameraUrl + "' alt='Cámara " + cameraId.slice(-1) + "'>";
            var ipAddress = cameraUrl.match(/\/\/(.*?)\//)[1];
            camara=cameraId;
            clasificador(ipAddress);
        }
    }
```

Ilustración 46: Código HTML-JavaScript, script de conexión de datos, Fuente propia

Explicación del código:

- Recibe un parámetro cameraId, que es el identificador de la cámara (por ejemplo, "camera1", "camera2", etc.).
- Obtiene el elemento de entrada de texto correspondiente a la cámara mediante document.getElementById(cameraId).
- Obtiene la URL de la cámara IP ingresada en el campo de entrada de texto mediante cameraInput.value.
- Comprueba si la URL no está vacía. Si la URL no está vacía, continúa con los siguientes pasos.
- Obtiene el elemento de la vista de la cámara correspondiente mediante document.getElementById("cameraView" + cameraId.slice(-1)). El cameraId.slice(-1) se utiliza para extraer el número de la cámara del identificador y así construir el ID del elemento de la vista de la cámara.
- Actualiza el contenido del elemento de la vista de la cámara con una etiqueta <img> que muestra la imagen de la cámara utilizando la URL obtenida anteriormente. También incluye un atributo "alt" con el número de la cámara.
- Extrae la dirección IP de la cámara de la URL utilizando una expresión regular y la asigna a la variable ipAddress.

- Establece una variable global llamada camara con el valor de cameraId. Esto parece estar relacionado con el seguimiento de la cámara seleccionada.
- Llama a la función clasificador(ipAddress) pasando la dirección IP como argumento. Esto sugiere que se está utilizando algún tipo de clasificador o proceso relacionado con la dirección IP de la cámara.

4.3. Script del Modelo clasificador

4.3.1. Función classifyAllCameras

Esta función se encargará de capturar los frames de cada cámara para posteriormente procesarlas y enviarlas a la función de clasificación de imágenes el cual luego de ser ejecutado entregará el resultado de predicción, el que es enviado a la función updatePrediction; que actualizará los resultados en la interfaz.

```
function classifyAllCameras() {
        var cameraViews = document.getElementsByClassName("camera");

        //camara 1
        var cameraView = cameraViews[0];
        var imageElement = cameraView.querySelector("img, video")
        if (imageElement) {
            classifyImage(imageElement)
                .then(function(predictions) {
                    console.log("predicciones para la cámara " + "1" + ": " + predictions);
                    updatePredictionResults(1, predictions);
                })
                .catch(function(error) {
                    console.error("Error al clasificar la imagen: ", error);
                });
        }
        //------------------------------------------------------------
```

Ilustración 47: Código HTML-JavaScript, script de obtención de frames , Fuente propia

Explicación del código:

- La primera línea selecciona todos los elementos HTML con la clase "camera" y los almacena en la variable cameraViews.
- Luego, selecciona el primer elemento de la lista cameraViews y lo almacena en la variable cameraView.
- Busca un elemento img o vídeo dentro del elemento cameraView y lo almacena en la variable imageElement.

- Si se encuentra un elemento img o vídeo, se llama a la función classifyImage pasando el elemento imageElement como argumento. Esta función devuelve una promesa que se resuelve con predicciones.
- Cuando la promesa se resuelve exitosamente, se muestra un mensaje en la consola que indica las predicciones para la cámara número 1 y se llama a la función updatePredictionResults con el número 1 y las predicciones como argumentos.
- Si la promesa se rechaza (por ejemplo, si hay un error en la clasificación de la imagen), se muestra un mensaje de error en la consola.
- Estas líneas de código serán repetidas para cada cámara con el cambio del número de cámara clasificado esto se realiza para que la función updatePrediction que espera dos valores reciba el valor de numero de cámara clasificada y el valor de la predicción.

4.3.2. Función classifyImage

Esta función recibe los frames capturados por la función anterior escalarla y enviarlas al servidor TensorFlow Serving, luego el resultado de este modelo es enviado a la función updatePrediction; que actualizará los resultados en la interfaz.

```
async function classifyImage(image) {
console.log(image);
image.crossOrigin = "Anonymous";
var canvas = document.createElement('canvas');
var context = canvas.getContext('2d');
canvas.width = 224;
canvas.height = 224;
context.drawImage(image, 0, 0, 224, 224);
const imageData = context.getImageData(0, 0, 224, 224);
console.log(imageData);
const tensor = preprocessImage(imageData);
await sendTensorToServer(tensor);

// Esperar el siguiente cuadro para continuar la clasificación
await new Promise((resolve) => requestAnimationFrame(resolve));
//retorno
return await sendTensorToServer(tensor);
}
```

Ilustración 48: Código HTML-JavaScript, script de clasificación de imágenes, Fuente propia

Explicación del código:

- Luego de definida la función, la primera línea registra la imagen en la consola para depuración.
- Establece la propiedad crossOrigin de la imagen en "Anonymous", lo que permite cargar imágenes desde otros orígenes.
- Crea un nuevo elemento de lienzo (canvas) con un contexto 2D.
- Establece el tamaño del lienzo en 224x224 píxeles.
- Dibuja la imagen en el lienzo, redimensionándola a 224x224 píxeles.
- Obtiene los datos de imagen (píxeles) del lienzo en formato ImageData.
- Llama a la función preprocessImage para realizar algún procesamiento adicional en los datos de imagen. Esto probablemente incluye normalización y ajustes necesarios para el modelo de clasificación.
- Llama a la función sendTensorToServer para enviar los datos de imagen procesados al servidor para su clasificación.
- Espera el siguiente cuadro (frame) de animación utilizando requestAnimationFrame. Lo que puede ser útil si se está procesando una secuencia de imágenes en un vídeo.
- Finalmente, vuelve a llamar a sendTensorToServer para asegurarse de que se haya enviado correctamente la información al servidor y luego retorna el resultado.

4.3.3. Función asíncrona sendTensorToServer

Esta función envía el tensor al servidor TensorFlow Serving. Los tensores se generan a partir de los frames capturados por las cámaras y son clasificados por el modelo, el cual nos proporciona una respuesta en forma de dos números que estarán en el rango de 0 a 1. Estos números indicarán si la situación es potencialmente delictiva o no.

```
async function sendTensorToServer(tensor) {
const url = "http://localhost:8501/v1/models/MoviNet:predict";
const data = {
instances: tensor.arraySync(),
};

try {
const response = await fetch(url, {
    method: "POST",
    body: JSON.stringify(data),
});

if (!response.ok) {
    throw new Error("Error en la clasificación del video.");
}

const result = await response.json();
console.log("Resultado de la clasificación: ", result.predictions[0]); // Acceder al array de predicciones
return result.predictions[0]; // Devolver el array de predicciones
} catch (err) {
console.error("Error en la solicitud al servidor TensorFlow Serving: ", err);
throw err;
}
}
```

Ilustración 49: Código HTML-JavaScript, script de envió de Tensor, Fuente propia

Explicación del código:

- Luego de definida la función, la primera línea define una URL que apunta al servidor TensorFlow Serving. En este caso, la URL es "http://localhost:8501/v1/models/MoviNet:predict".
- Prepara los datos que se enviarán al servidor en un formato adecuado. En este caso, los datos se estructuran como un objeto JavaScript con una propiedad "instances" que contiene los datos del tensor en formato de matriz (tensor.arraySync()).
- Utiliza la función fetch para realizar una solicitud POST al servidor con la URL y los datos preparados.

- Verifica si la respuesta del servidor es exitosa (status code 200). Si no lo es, lanza un error indicando que ha habido un problema en la clasificación.
- Si la respuesta es exitosa, convierte la respuesta en un objeto JSON y registra el resultado en la consola. Luego, devuelve la primera predicción del resultado.
- En caso de que ocurra un error en la solicitud al servidor, captura ese error, lo registra en la consola y lo vuelve a lanzar para que pueda ser manejado más arriba en la cadena de llamadas.

4.4. Script de envió de datos ESP32

```
function clasificador(iptemp){
    if(conectado==0)
        {
        ip=iptemp
        camaraws=camara;
        console.log("ws:"+camaraws);
        var gateway = "ws://"+ip+"/wc";
        clasiw = new WebSocket(gateway);

        clasiw.onopen = onOpen;
        clasiw.onclose = onClose;
        }
    }

    function onOpen(event) {
    console.log('Connection opened');
    conectado=1;
    }

    function onClose(event) {
    console.log('Connection closed');
    }

    function send2(){
        console.log("camws"+camaraws)
        var camenv="camera"+camara
        console.log("cam"+camenv)
        if(camaraws==camenv)
        {
            if(tipo!=tempt)
            {
            var mensaje = tipo + ",:Clasificacion";
            clasiw.send(mensaje);
            console.log('enviando dato clasificacion');
            tempt=tipo;
            }
        }
    }
```

Ilustración 50: Código HTML-JavaScript, Envió de datos, Fuente propia

Explicación del código:

- La función clasificadora toma un argumento iptemp. Si la variable conectada es igual a 0, asigna el valor de iptemp a la variable ip, y asigna el valor de cámara a la variable camaraws. Luego, construye una URL de WebSocket (gateway) utilizando la dirección IP (ip) y establece una conexión WebSocket (clasiw) con esa URL. También configura funciones de devolución de llamada (onOpen y onClose) para manejar eventos de apertura y cierre de la conexión.
- La función onOpen se ejecuta cuando la conexión WebSocket se abre con éxito. Registra un mensaje en la consola y establece la variable conectada en 1 para indicar que la conexión está abierta.
- La función onClose se ejecuta cuando la conexión WebSocket se cierra. Registra un mensaje en la consola.
- La función send2 verifica si camaraws es igual a una cadena construida como "camera" seguida del valor de cámara. Si son iguales y tipo no es igual a tempt, construye un mensaje de texto que contiene tipo y “: Clasificación" y lo envía a través de la conexión WebSocket (clasiw). Luego, registra un mensaje en la consola y actualiza tempt con el valor de tipo.

5. Modelo 3D para el prototipo de módulo disuasorio ESP32-CAM

He diseñado el modelo 3D con la capacidad de alojar todos los componentes del módulo prototipo. La estructura incluye aberturas destinadas a permitir la conexión externa de una lámpara eléctrica al control de relé; así como la conexión del cargador de batería a través del puerto USB.

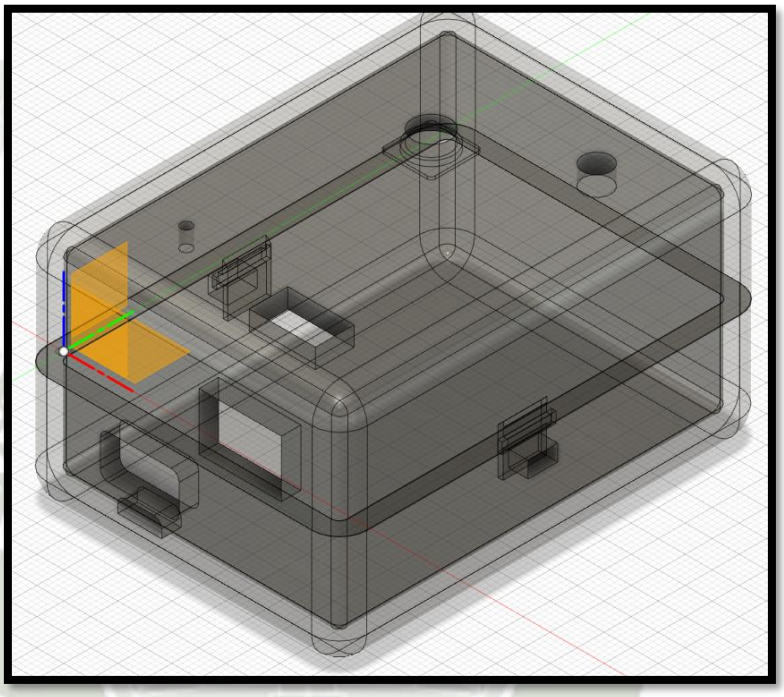


Ilustración 51: Estructura 3D, Estructura para el prototipo de módulo disuasorio, Fuente Propia

CAPÍTULO V. FUNCIONAMIENTO Y RESULTADOS FINALES DEL SISTEMA

1. Módulo prototipo disuasorio Ensamblado

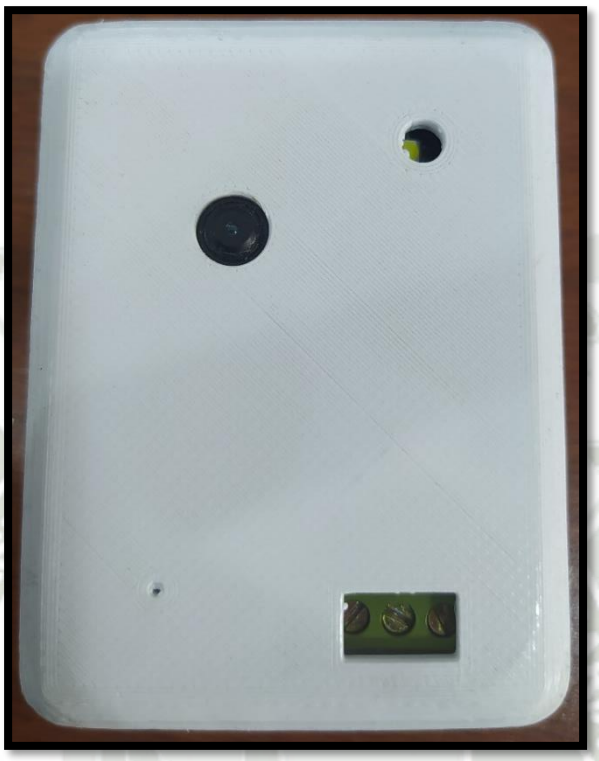


Ilustración 52: Foto, Prototipo ensamblado ESP32,
Fuente Propia

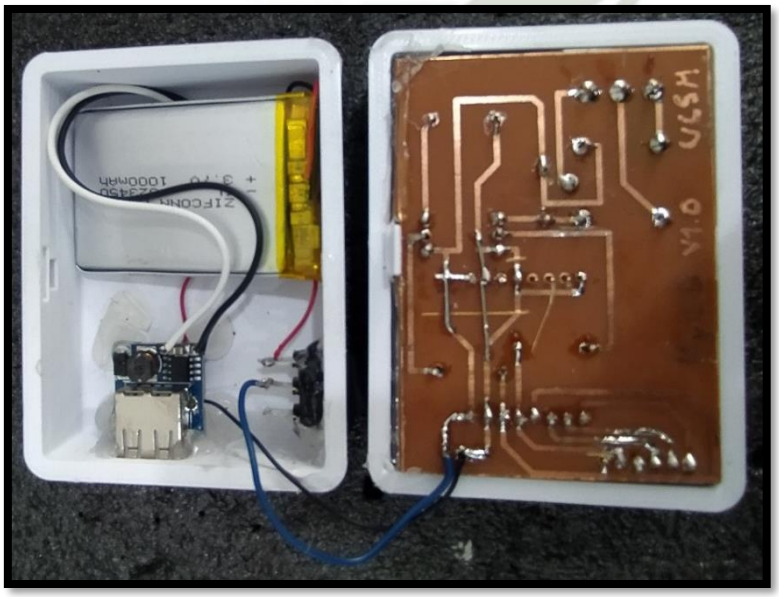


Ilustración 53: Foto, Sección interna prototipo,
Fuente Propia

2. Carga de código Arduino módulo disuasorio

2.1. Conexión ESP32-ARDUINO

2.1.1. Conexión física entre ESP32-CAM y PC

Debido a que el ESP32-CAM no cuenta con un puerto USB, se tiene que usar un converso de TTL a USB; con el objetivo de generar un puerto COM en la computadora y poder subir el programa para este proyecto usamos su programador que es ESP32-CAM-MB.

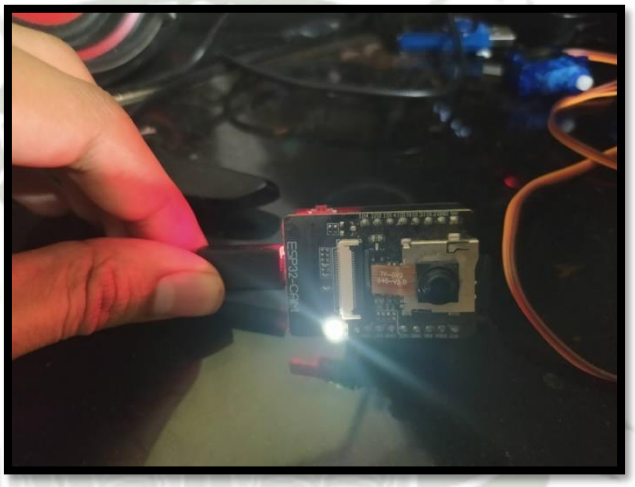


Ilustración 54: Foto, Conexión física ESP32,
Fuente Propia

2.1.2. Conexión con puerto

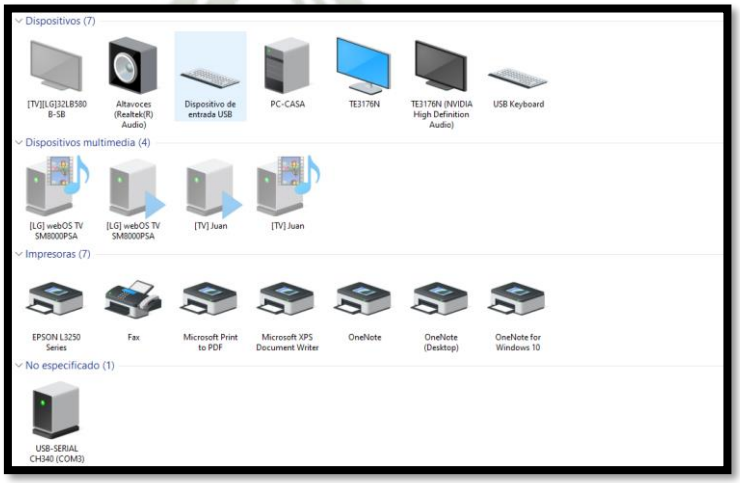


Ilustración 55: Captura de pantalla, Conexión con puerto,
Fuente Propia

Como se puede observar no se necesita ninguna configuración adicional, gracias a la instalación del programa Arduino IDE el driver CH340; se instala y asigna un puerto COM en este caso COM3.

2.2. Datos Recibidos por el puerto serial

2.2.1. Puerto Serial

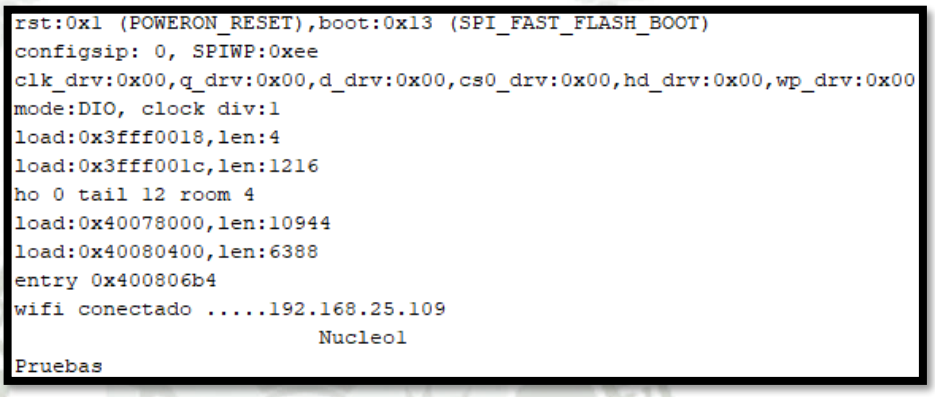


Ilustración 56: Captura de pantalla, Puerto serial Arduino, Fuente Propia

Como se puede observar en los resultados del puerto serial la inicialización de los módulos wifi e inicialización, al entregarnos una dirección ip; también observamos la inicialización del nucleo1 y la palabra “prueba” se imprime como confirmación de inicio correcto del ESP32-CAM.

2.2.2. Datos obtenidos

- WIFI módulo activado
- IP de servidor WEB: 192.168.25.109
- Inicio correcto de la cámara OVI
- Inicio correcto de Núcleo 1
- Programa ESP32-CAM inicializado correctamente

3. Modelo Clasificado en servidor TensorFlow Serving

3.1.1. Carpeta de Modelo:

La versión del modelo MoViNet es el modelo desarrollado para este trabajo, el cual será almacenado en la carpeta "TESISMODELO". Desde dicha ubicación, el modelo será copiado a la imagen del servidor TensorFlow Serving.

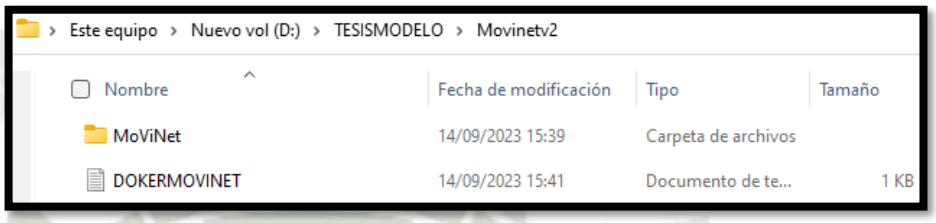


Ilustración 57: Captura de pantalla, Carpeta del modelo clasificador, Fuente Propia

3.1.2. Comandos para la publicación del modelo

- Creación de variable de ubicación del modelo:
  - Set-Variable -Name "MODMovi" -Value "D:\mentrenados\MODELOV5"
- Cargar de modelo a Tensorflow Serving
  - docker run -t --rm -p 8501:8501 -v "$MODMovi/MoViNet:/models/MoViNet" -e MODEL_NAME=MoViNet tensorflow/serving

Con estos comandos el modelo entrenado para este trabajo se encontrará publicado y en ejecución a la espera de solicitudes de clasificación en la dirección: http://localhost:8501/v1/models/MoViNet:predict

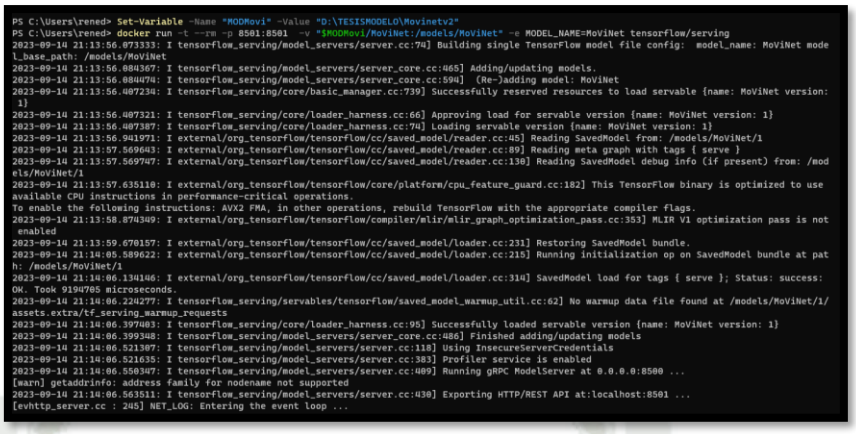


Ilustración 58: Captura de pantalla, Terminal de Windows para cargar modelo, Fuente Propia

3.1.3. Modelo publicado

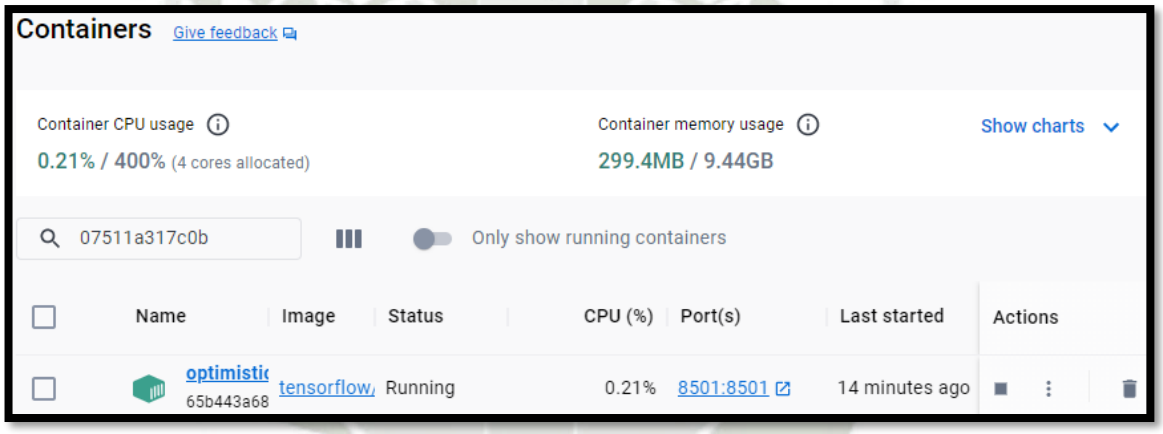


Ilustración 59: Captura de pantalla, Conteiner Tensorflow Serving publicando el modelo, Fuente Propia

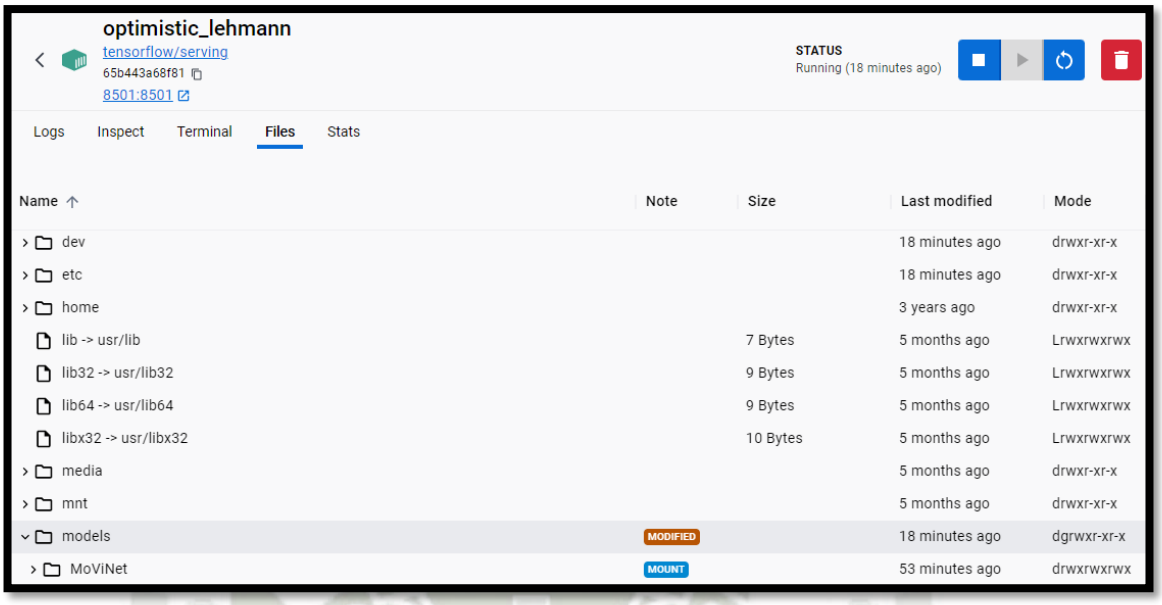


Ilustración 60: Captura de pantalla, Conteiner TensorFlow modelo cargado,

Fuente Propia

Con esto, el modelo se encuentra publicado y listo para recibir solicitudes de clasificación. Como se puede observar, TensorFlow Serving dispone de una carpeta denominada "models", en esta carpeta es posible agregar modelos adicionales para llevar a cabo distintos tipos de clasificaciones. La cantidad de solicitudes que pueden ser procesadas, depende de los recursos computacionales disponibles en el servidor; en el cual se está ejecutando la imagen.

4. Página Web Index.html

4.1.1. Acceso a la IP desde navegador

Para acceder a la página "index.html", debemos utilizar la dirección IP local, ya que el servidor Live Server de Visual Studio Code no crea una máquina virtual; sino que genera un servidor local.

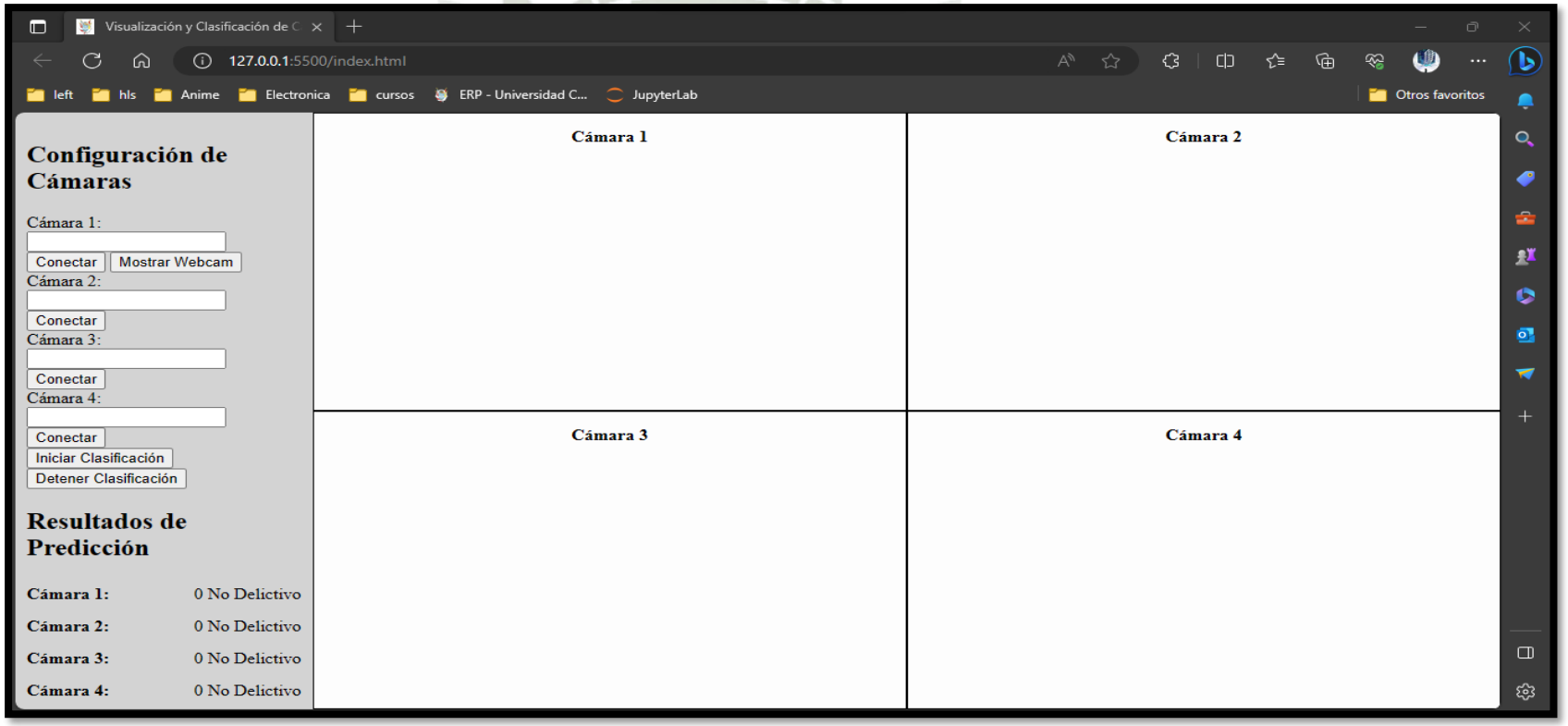


Ilustración 61: Captura de Pantalla, Pagina WEB Index.html,
Fuente Propia

4.1.2. Conexión de las cámaras

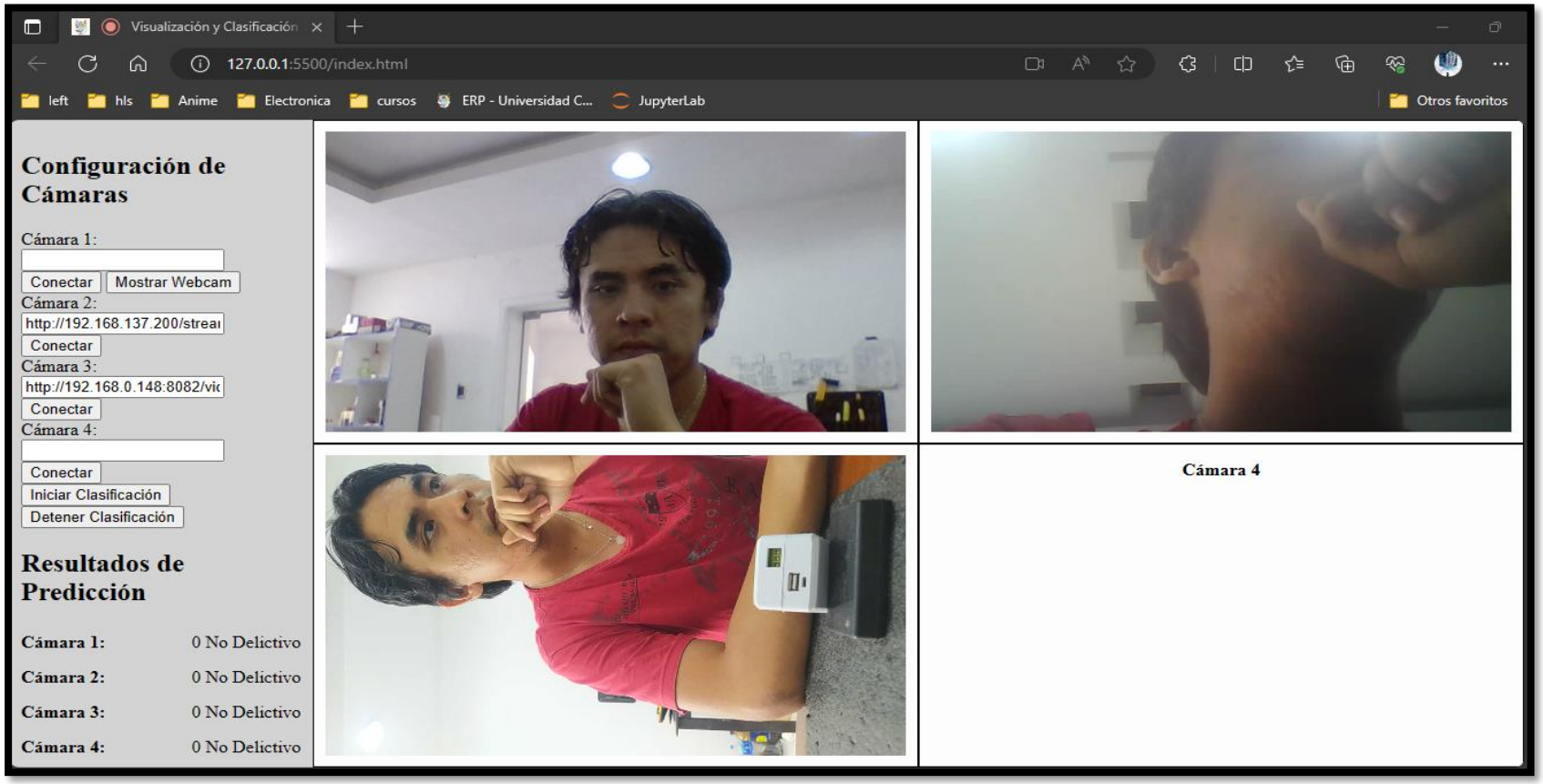


Ilustración 62: Captura de Pantalla, Registros Red,
Fuente Propia

4.1.3. Registros de red

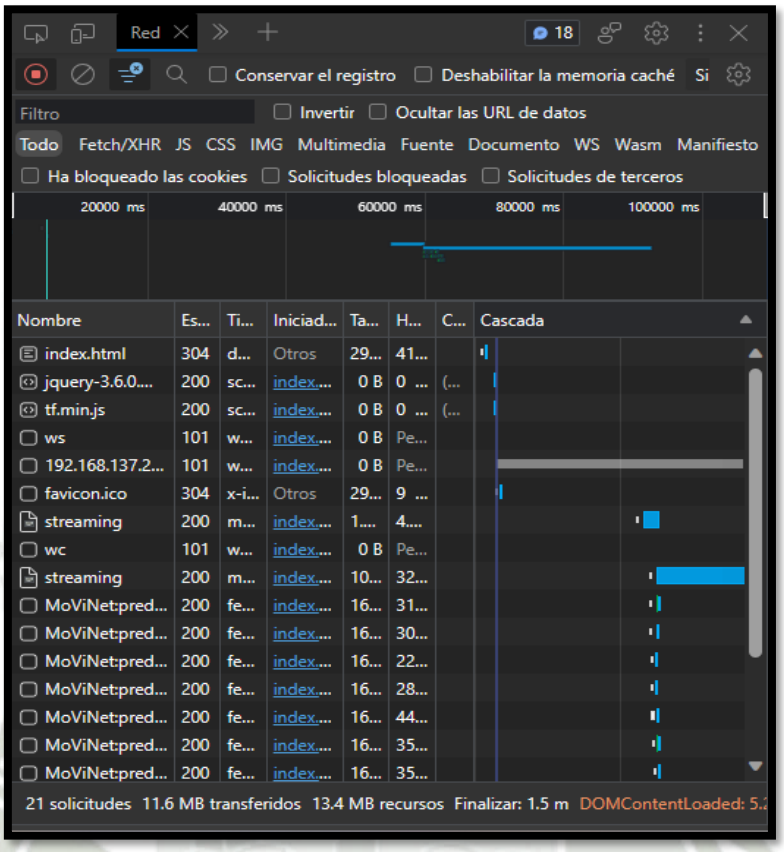


Ilustración 63: Captura de Pantalla, Registros Red, Fuente Propia

Como se puede observar, la carga de la página INDEX.HTML provoca la carga de recursos adicionales solicitados por la misma página y sus diversos componentes; que incluyen lo siguiente:

- Favicon.ico: Se trata de la imagen que se muestra en la esquina superior de la pestaña del navegador, en este caso; el logotipo de la universidad.
- Streaming: Esta corresponde a la conexión establecida con la cámara ESP32-CAM. Es importante destacar que la transmisión de datos se mantiene constante, debido a que el ESP32 responde continuamente con el envío constante de fotogramas para su visualización.
- WebSocket (WC): Esta se utiliza para llevar a cabo una conexión de prueba destinada a enviar el estado de clasificación al módulo disuasorio. Es esencial señalar que este componente no opera de manera continua, sino que

se activa únicamente cuando se produce un cambio en los estados de clasificación.

- MoViNet Predict: Este módulo se encarga del envío de los fotogramas de la cámara al servidor para su posterior clasificación. Las respuestas son recibidas después del envío, y en el caso de que la respuesta indique una actividad delictiva, se establece una conexión WebSocket (WC) con la cámara del módulo disuasorio para transmitir los datos y ejecutar las acciones disuasorias correspondientes.

5. Clasificación de Acciones Delictivas y no Delictivas

5.1.1. Acciones Delictivas Webcam

Ilustración 64: Captura de Pantalla, Clasificación de acción delictiva WEBCAM,

Fuente Propia

5.1.2. Acción delictiva módulo disuasorio ESP32CAM e IP CAM

Ilustración 65: Captura de Pantalla, Clasificación de acción delictiva ESP32-CAM y IP CAM,
Fuente Propia

5.1.3. Acciones no Delictivas

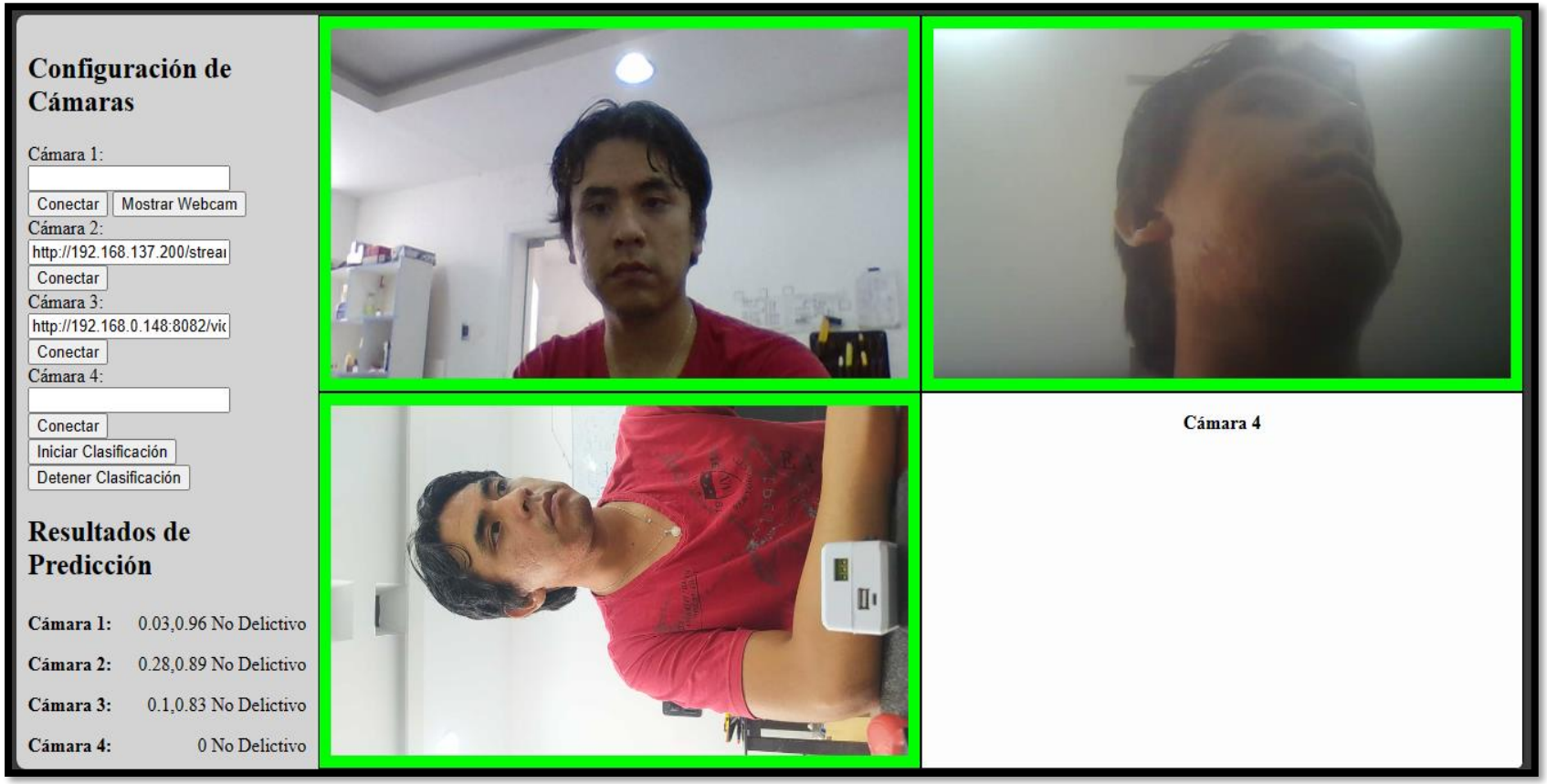


Ilustración 66: Captura de Pantalla, Clasificación de acción no delictiva todas las cámaras, Fuente Propia

6. Especificaciones y requerimientos del prototipo

Las siguientes especificaciones, han sido elaboradas con el propósito de asegurar la correcta ejecución del sistema prototipo; que permite trabajar con un máximo de 4 cámaras. Es importante destacar que, para esta configuración se asume que tanto los servidores web como el cliente; están alojados en el mismo dispositivo computador. En el caso de que se desee implementar el sistema de manera separada, los requerimientos computacionales variarán significativamente respecto a los que se describen a continuación; y dependerán directamente de la cantidad de clientes y cámaras que se requiera clasificar.

6.1.1. Hardware

Dispositivo Servidor-Cliente

- **Sistema Operativo:** Windows 10 o superior
- **Memoria Ram:** 16 Gbytes
- **Procesador:** Intel Core I5
- **Memoria Disco:** 1 Tbytes
- Virtualización de procesador
- Compatibilidad con Híper V
- Tarjeta WiFi

Cámaras de Vigilancia

- **Cantidad Máxima**: 4 distribuidores de videovigilancia por servidor web y manejo de interfaz.
- **Resolución de vídeo**: 480 p – 720
- **Modos de visión:** Nocturna y Diurna
- **Distancia máxima de visión**: Recomendable 20 m
- **Capacidad de movimiento:** 355 ° Pan 90 ° Tilt
- **Resistencia a exteriores:** IP66
- Lámparas Led
- Sistemas de comunicación: WiFi

Fuente de Alimentación

- **Voltaje de entrada:** 220 – 240 AC
- **Frecuencia:** 60 Hz
- **Voltaje de salida:** 5v DC
- **Corriente de salida:** 1.5 Amperios

Router WiFi

- **Cobertura de doble banda:** La banda de 2.4 GHz y la banda de 5 GHz.
- **Velocidades de Wi-Fi:** 300 Mbps en la banda de 2.4 GHz
- **Antenas ajustables:** Si debido a que permite el ajuste de zonas de cobertura
- **Amplificador de señal (opcional):** Depende de la distancia entre los dispositivos y el Router
- **Seguridad:** WPA2 o WPA3
- **Número de puertos Ethernet:** Depende de la cantidad de dispositivos consumidores conectados a través de cable ethernet

6.1.2. Software

Navegador web

- Soporte para HTML 5
- Soporte para WebGL 1.0 o superior.
- Soporte para WebAssembly.
- Soporte para Typed Arrays.
- Soporte para ECMAScript 6 o superior.
- Soporte para SharedArrayBuffer.
- Addon necesario Moesif CORS

Docker

- Soporte para TensorFlow Serving
- Virtualización Habilitada
- Complemento de Híper V

CONCLUSIONES

1. La selección del algoritmo de Aprendizaje automático que permita identificar actos delictivos se logró mediante la evaluación de los resultados donde destaca la eficacia del Modelo MoViNet. Se emplearon clips de videos con 10 frames para MoViNet e imágenes provenientes de los mismos clips para el modelo convolucional. La tasa de falsos positivos, alcanzo un 80% con el modelo convolucional y un 30% con la combinación de redes 3D y recurrentes de MoViNet.
2. Las falsas clasificaciones de acciones delictivas señaladas por el prototipo, se deben a un conjunto de datos pequeños; en comparación con los utilizados para el entrenamiento de otras redes neuronales de uso similar. Es preciso indicar que una red neuronal, puede ser reentrenada de forma continua; para mejorar su capacidad de detección.
3. El uso de redes neuronales convolucionales 3D, ha demostrado ofrecer resultados superiores; en comparación con las redes neuronales convolucionales 2D. Esto se debe a la incorporación de una dimensión adicional a las que ya son utilizadas en las redes 2D, que sería el ancho, lo que permite trabajar con múltiples fotogramas; que representan breves intervalos de tiempo.
4. El modelo MoViNet para la detección de actos delictivos de robo y asalto se logró empleando conjuntos de datos gratuitos, algunos extraídos de videos de YouTube para su entrenamiento. Se utilizaron clips de videos para MoViNet e imágenes para los demás modelos. Las métricas utilizadas fueron la pérdida (loss) y la precisión (accuracy). Para el modelo denso, se obtuvieron resultados de 5.69 en pérdida y 0.69 en precisión. En el caso del modelo convolucional, los resultados fueron de 3.4 en pérdida y 0.82 en precisión. Para MoViNet, se registró una precisión del 0.75 y una pérdida de 2.20. Estos resultados subrayan la efectividad de MoViNet en la identificación de actos delictivos.
5. El diseño del código fuente para la identificación del acto delictivo y la implementación de medidas disuasorias se ha logrado a través del uso de un microprocesador ESP32-CAM. Este módulo prototipo captura imágenes y las envía a un modelo de clasificación. En caso de que el modelo identifique un acto delictivo, el módulo prototipo activa un relé (donde iría conectado la lampara y sirena) y un zumbador como medidas disuasorias.

6. Para el desarrollo del módulo en la sección de software, es decir en la programación de la placa ESP-32, se empleó la librería ESPAsyncWebServer. Esta elección se basó en el reconocimiento de que, a pesar de la mayor complejidad inherente a la implementación de un servidor asíncrono; sus ventajas son significativas. Estas ventajas incluyen una mayor eficiencia en la gestión de recursos, menor latencia y una mayor escalabilidad.
7. El diseño e implementación de la estructura prototipo del módulo disuasorio, realizado a través de Fusión 360, ha logrado integrar de manera compacta y segura todos los componentes electrónicos. Aunque el modelo actual carece de protecciones IP, es esencial señalar que el modelo final deberá incorporar estas salvaguardas. Esto se debe a que, al estar situado en exteriores, se espera que el módulo disuasorio pueda resistir condiciones climáticas adversas.
8. Dado el porcentaje de aciertos del prototipo en la detección de acciones delictivas o no delictivas, es del orden del 70 %; podemos afirmar que se ha cumplido el objetivo de identificación de acciones delictivas.

RECOMENDACIONES

1. Se recomienda la aplicación de la técnica de aprendizaje transferido (transfer learning) al modelo existente, con el objetivo de incrementar significativamente la eficiencia y precisión en la identificación de acciones delictivas y su distinción de aquellas que no lo son. Con ello se lograría que el modelo futuro pueda beneficiarse de los patrones y características aprendidas de los datos usados para este trabajo.
2. Se recomienda explorar alternativas a la red WiFi, en lugares de alta concurrencia; debido a posibles problemas de interferencia y saturación de canales. Asimismo, es necesario tener en cuenta la seguridad de la red WiFi utilizada para garantizar la protección de los datos transmitidos en el sistema IoT.
3. Es importante señalar la información considerada en la investigación se basa en información disponible a partir de septiembre de 2021 y es posible que se hayan generado desarrollos adicionales en el modelo MoViNet o investigaciones relacionadas desde entonces. Si está interesado en la información más reciente, se recomienda que consultar artículos de investigación y recursos recientes en las áreas de visión por computadora y aprendizaje profundo.

REFERENCIAS

López Bonilla, A. D., & Espinoza Quezada , F. D. (2022). UNIVERSIDAD DE AZUAY. *Estudio técnico de un nuevo servicio de seguridad ciudadana*. Cuenca, Ecuador. Recuperado el 20 de 11 de 2022, de https://dspace.uazuay.edu.ec/bitstream/datos/11893/1/17420.pdf

ABDATUM. (22 de Noviembre de 2021). *Redes neuronales recurrentes | Qué son las RNN*. Recuperado el 05 de Setiembre de 2023, de Abdatum: https://abdatum.com/tecnologia/redes-neuronales-recurrentes

AI-Tinker. (2017). *ESP32-CAM module*. Recuperado el 16 de Mayo de 2023, de loboris: https://loboris.eu/ESP32/ESP32-CAM%20Product%20Specification.pdf

Anonimo. (s.f.). *Altium*. Obtenido de Cómo Diseñar Circuitos de Alimentación MicroSD sin Desestabilizar el Suministro de Voltaje a Bordo: https://resources.altium.com/es/p/how-to-design-microsd-power-circuits-without-destabilizing-on-board-voltage-supply

Anonimo. (s.f.). *electrocrea*. Obtenido de SG90 180º Servo motor: https://electrocrea.com/products/servo-mini-sg90

Anonimo. (s.f.). *naylampmechatronics*. Obtenido de RELAY 5VDC SRD-05VDC-SL-C: https://naylampmechatronics.com/drivers/263-relay-5vdc-srd-05vdc-sl-c.html

Arango, M. B. (21 de Abril de 2022). *RPP*. Obtenido de https://rpp.pe/peru/actualidad/comisarias-una-radiografia-de-sus-principales-deficiencias-informe-noticia-1399241?ref=rpp

AWS. (s.f.). *¿Qué es IoT? - Explicación del Internet de las cosas*. Obtenido de AWS Amazon: https://aws.amazon.com/es/what-is/iot/

Barrientos Avendaño, E., Pérez Yáñez, J. M., & Rico-Bautista, D. (s.f.). Dispositivo de seguridad IoT soportado por sensores para el monitoreo de bicicletas. Ocaña, Colombia. Recuperado el 20 de 11 de 2022, de https://www.fesc.edu.co/Revistas/OJS/index.php/mundofesc/article/view/830/672

Brunero, G. (4 de Mayo de 2020). *Servidores web sincrónicos y asíncronos*. Recuperado el 27 de Julio de 2022, de Medium: https://medium.com/@giuliano.brunero/servidores-web-sincr%C3%B3nicos-y-as%C3%ADncronos-19c2e4286f7c

Calderón Ñaccha, G. L., Santillán Paredes, J. F., & Masias Donayre, Y. M. (s.f.). Plan de negocios para implementar un sistema de detección y alertas. Lima, Peru. Recuperado el 20 de 11 de 2022, de https://repositorio.esan.edu.pe/bitstream/handle/20.500.12640/3096/2022_MADTI_19-1_04_TI.pdf?sequence=1&isAllowed=y

Dan Kondratyuk, L. Y., Zhang, L., Mingxing, B., Brown, M., & Gong, B. (2021 de Abril de 2018). *MoViNets: redes de video móviles para un reconocimiento de video eficiente.* Recuperado el 05 de Mayo de 2023, de arxiv: https://arxiv.org/abs/2103.11511

Docker. (2022). *CLI Cheat Sheet*. Recuperado el 20 de Enero de 2023, de Docker: https://docs.docker.com/get-started/docker_cheatsheet.pdf

Du, T., Heng, W., Lorenzo, T., Jamie, R., Yann, L., & Manohar, P. (12 de Abril de 2018). *A Closer Look at Spatiotemporal Convolutions for Action Recognition.* Recuperado el 07 de Setiembre de 2023, de Arxiv.: https://arxiv.org/abs/1711.11248v3

Espressif Systems. (2017). *ESP32*. Recuperado el 25 de Mayo de 2023, de Espressif Overview: https://www.espressif.com/en/products/socs/esp32

Espressif Systems. (2019). *ESP32 SeriesDatasheetIncluding.* Obtenido de Alldatasheet: https://pdf1.alldatasheet.com/datasheet-pdf/view/1148023/ESPRESSIF/ESP32.html

FOSHAN. (s.f.). *alldatasheet*. Obtenido de TIP41C: https://pdf1.alldatasheet.com/datasheet-pdf/view/962262/FOSHAN/TIP41C.html

geeksforgeek. (10 de Enero de 2023). *Redes residuales (ResNet) – Deep Learning.* Recuperado el 05 de Setiembre de 2023, de geeksforgeeks: https://www.geeksforgeeks.org/residual-networks-resnet-deep-learning/

González Vidal, J. C. (2015). Desarrollo de un servidor web asíncrono con Arduino. Cartagena , Colombia. Obtenido de https://repositorio.upct.es/bitstream/handle/10317/5246/tfg772.pdf?sequence=1&isAllowed=y

Google. (27 de Agosto de 2022). *Estudio detallado del AA: Regresión lineal*. Obtenido de Google developers: https://developers.google.com/machine-learning/crash-course/descending-into-ml/linear-regression?hl=es-419

Google. (15 de Julio de 2023). *Reducción de la pérdida: Tasa de aprendizaje*. Obtenido de Google developers: https://developers.google.com/machine-learning/crash-course/reducing-loss/learning-rate?hl=es-419

Gunther Gridling, B. W. (2007). *Introduction to Microcontrollers.* Vienna: Vienna University of Technology.

herramientasingenieria. (2021). *Duración teórica de la carga de una batería*. Recuperado el 07 de Setiembre de 2023, de herramientasingenieria: https://www.herramientasingenieria.com/onlinecalc/spa/duracion-baterias/duracion-baterias.html

Hewlett Packard. (2023). *Contenedores*. Recuperado el 10 de Setiembre de 2023, de Hewlett Packard: https://www.hpe.com/es/es/what-is/containers.html

HUARAYA APAZA, L. (2015). UNIVERSIDAD NACIONAL TECNOLÓGICA DE LIMA SUR. *SISTEMA WEB PARA MEJORAR EL CONTROL DE OPERACIONES DE LA SEGURIDAD CIUDADANA EN LA MUNICIPALIDAD DE SANTIAGO DE SURCO*. Lima, Peru. Recuperado el 20 de 11 de 2022, de https://repositorio.untels.edu.pe/jspui/bitstream/123456789/728/1/HUARAYA%20APAZA%2c%20LUCIA.pdf

IBM. (s.f.). *¿Qué son las redes neuronales recurrentes?* Recuperado el 05 de Mayo de 2023, de IBM CLOUD: https://www.ibm.com/es-es/topics/recurrent-neural-networks

Instituto Nacional de Estadística e Informática. (2017). *Percepción de Inseguridad.* Recuperado el 25 de Mayo de 2022, de INEI: https://www.inei.gob.pe/media/MenuRecursivo/publicaciones_digitales/Est/Lib1519/cap04.pdf

Instituto Nacional de Estadística e Informática. (Octubre de 2021). *Estadistica de Criminalidad.* Recuperado el 25 de Mayo de 2022, de INEI: https://www.inei.gob.pe/media/MenuRecursivo/boletines/estadisticas_de_criminalidad_seguridad_ciudadana_abr-jun2021.pdf

Kaiming, H., Xiangyu , Z., Shaoqing, R., & Jian, S. (10 de Diciembre de 2015). *Deep Residual Learning for Image Recognition.* Recuperado el 05 de Setiembre de 2023, de ArXiv: https://arxiv.org/abs/1512.03385

Kazimipour, B. (2021). *¿Cuál es la diferencia entre una red neuronal convolucional y una red neuronal normal?* Recuperado el 06 de Setiembre de 2023, de QA Stack: https://qastack.mx/ai/5546/what-is-the-difference-between-a-convolutional-neural-network-and-a-regular-neur

Kravets, I. (25 de Marzo de 2022). *ESPAsyncWebServer*. Recuperado el 15 de Agosto de 2023, de Github: https://github.com/me-no-dev/ESPAsyncWebServer

Machaca Arcana, L. A. (2019). *Reconocimiento de eventos anomalos en videos.* Recuperado el 13 de Octubre de 2023, de Repositorio de UNSA: https://repositorio.unsa.edu.pe/server/api/core/bitstreams/457c1ecd-8327-4212-acf2-1a31463c6f9c/content

Microsoft. (10 de Octubre de 2021). *¿Qué es Docker?* Recuperado el 05 de Mayo de 2023, de Learn microsoft: https://learn.microsoft.com/es-es/dotnet/architecture/microservices/container-docker-introduction/docker-defined

Microsoft. (s.f.). *Microsoft Learn*. Obtenido de The WebSocket API: https://learn.microsoft.com/en-us/previous-versions/windows/internet-explorer/ie-developer/dev-guides/hh673567(v=vs.85)?redirectedfrom=MSDN

Quiñonez Muñoz, O. (2019). *Internet de las Cosas (IoT).* Ibukku LLC. Recuperado el 16 de Mayo de 2023, de https://books.google.com.pe/books?id=vnnEDwAAQBAJ&pg=PT2&hl=es&source=gbs_toc_r&cad=2#v=onepage&q&f=false

Redhat. (20 de Enero de 2023). *¿Qué es Docker y cómo funciona?* Recuperado el 25 de Mayo de 2023, de Redhat: https://www.redhat.com/es/topics/containers/what-is-docker

Sánchez , Laura. (18 de Mayo de 2023). *Efecto disuasorio: el mejor aliado frente a los robos*. Recuperado el 15 de Julio de 2023, de asesoralarmas.com: https://www.asesoralarmas.com/blog/efecto-disuasorio-mejor-aliado-frente-los-robos/

Saravia, A. R. (2019). *ESP32 NODE MCU.* Obtenido de https://www.microelectronicash.com/: https://www.microelectronicash.com/downloads/ESP32_MANUAL.pdf

Securitas. (2020). *Cómo implementar medidas disuasorias en mi hogar o negocio*. Recuperado el 28 de Julio de 2022, de SecuritasDirect: https://www.securitasdirect.es/blog/medidas-disuasorias/

Spiegato. (2023). *¿Qué es una red neuronal convolucional?* Recuperado el 05 de Setiembre de 2023, de Spiegato: https://spiegato.com/es/que-es-una-red-neuronal-convolucional

TensorFlow. (2020). *Train and serve a TensorFlow model with TensorFlow Serving.* Recuperado el 15 de Setiembre de 2023, de Colab: https://colab.research.google.com/github/tensorflow/tfx/blob/master/docs/tutorials/serving/rest_simple.ipynb?hl=es-419

TensorFlow. (28 de Enero de 2021). *Arquitectura TensorFlow Serving.* Recuperado el 15 de Setiembre de 2023, de TensorFlow: https://www.tensorflow.org/tfx/serving/architecture?hl=es-419

TensorFlow. (07 de Julio de 2021). *Publicación de TensorFlow con Docker*. Recuperado el 18 de Marzo de 2023, de TensorFlow: https://www.tensorflow.org/tfx/serving/docker?hl=es-419

TensorFlow. (27 de Mayo de 2023). *Cargar datos de vídeo*. Recuperado el 02 de Junio de 2023, de TensorFlow: https://www.tensorflow.org/tutorials/load_data/video

TensorFlow. (08 de Agosto de 2023). *MoViNet para el reconocimiento de acción de transmisión*. Recuperado el 01 de Mayo de 2023, de TensorFlow Hub: https://www.tensorflow.org/hub/tutorials/movinet#the_streaming_model

TensorFlow. (27 de Julio de 2023). *Video classification with a 3D convolutional neural network.* Recuperado el 25 de Agosto de 2023, de TensorFlow: https://www.tensorflow.org/tutorials/video/video_classification

Unitelectronics. (s.f.). *TP4056.* Recuperado el 07 de Setiembre de 2023, de uelectronics: https://uelectronics.com/producto/tp4056-con-proteccion-dual-microusb-tipo-c-cargador-de-baterias-li-ion-li-po/

Universidad de Chile. (2008). *Cómo funciona la Web.* Santiago: Gráfica LOM.

Valdéz Peréz, F., & Pallás Areny, R. (2007). *Microcontroladores Fundamentos y Aplicaciones con PIC.* Marcombo.

Verizon. (21 de Febrero de 2023). *Streaming*. Obtenido de Verizon español: https://espanol.verizon.com/articles/internet-essentials/streaming-definition/

Wikipedia. (30 de Agosto de 2021). *Redes neuronales recurrentes.* Recuperado el 05 de Setiembre de 2023, de Wikipedia: https://es.wikipedia.org/wiki/Redes_neuronales_recurrentes

Wikipedia. (2023). *SSID*. Obtenido de Wikipedia, la enciclopedia libre: https://es.wikipedia.org/wiki/SSID

Zamudio, R. (16 de Octubre de 2016). *Qué significa la cantidad de mAh (miliamperios) en la duración de las baterías.* Recuperado el 07 de Setiembre de 2023, de Qore: https://www.qore.com/noticias/51574/Que-significa-la-cantidad-de-mAh-miliamperios-en-la-duracion-de-las-baterias

GLOSARIO

A

accuracy
La exactitud es la medida de cuán cerca está una medición del valor verdadero o real. La precisión es la medida de cuán cerca están entre sí varias mediciones del mismo objeto. En otras palabras, la precisión es una descripción de los errores aleatorios, una medida de la variabilidad estadística, 79, 86, 149, 150, *Véase*

asíncrono
Un servidor web asíncrono es capaz de atender a varios clientes de forma simultánea sin detener todas las demás operaciones mientras espera que vuelva la llamada del servicio web, 27, 49

C

callback
En programación informática, una retrollamada o devolución de llamada (en inglés callback), también denominada «posllamada» (o «call-after» en inglés), es una función ejecutable que se usa como argumento de otra función. De esta forma, al llamar a la función externa, esta ejecutará la función de retrollamada, 79

CNN
Una red neuronal convolucional (CNN o ConvNet) es una arquitectura de red para Deep Learning que aprende directamente a partir de datos. Son particularmente útiles para identificar patrones en imágenes con el fin de reconocer objetos, clases y categorías. Estas redes son particularmente útiles para encontrar patrones en imágenes para reconocer objetos, caras y escenas, 36, 37, 45, 68, 79

D

Dataset
Un dataset es un conjunto de datos que se utiliza para el entrenamiento de inteligencia artificial en clasificación de imágene, 71, 72, 73, 75

Disuasorio
Que disuade o tiene la capacidad de hacer que alguien desista de una acción o decisión., 10

E

época
En una red neuronal, una época es un término que se refiere a una iteración completa de todo el conjunto de datos de entrenamiento a través de la red neuronal1. En otras palabras, una época es un término que se utiliza para describir el número de veces que se ha pasado por todo el conjunto de datos durante el entrenamiento, 76, 80, 86, 87, 88, 90, 92, 93

esp32-cam
El ESP32-CAM es un dispositivo multifuncional. Además de la conectividad Wifi y Bluetooth que viene integrada de fábrica, también cuenta con pines GPIO. Se han añadido dos opciones adicionales
una pequeña cámara de video y una conexión para una tarjeta MicroSD, donde se pueden almacenar fotos o videos, 5, 49

H

hiperparametros
Los hiperparámetros se utilizan para controlar el algoritmo de aprendizaje en el aprendizaje automático y pueden considerarse como los ajustes de un algoritmo de aprendizaje automático que se ajustan para optimizar su rendimiento, 37

I

IOT
El Internet de las cosas (IoT) describe la red de objetos físicos ("cosas") que llevan incorporados sensores, software y otras tecnologías con el fin de conectarse e intercambiar datos con otros dispositivos y sistemas a través de Internet, 1, 2, 5, 9, 49

M

Machine Learning
El aprendizaje automático es una rama de la inteligencia artificial que se basa en el uso de datos y modelos matemáticos para crear programas informáticos que pueden aprender, mejorar y predecir sin instrucciones directa, 30, 44, 68

O

overfitting
El overfitting es un fenómeno en el aprendizaje automático que ocurre cuando un modelo estadístico se ajusta exactamente a sus datos de entrenamiento. Cuando esto sucede, el modelo no puede funcionar con precisión contra datos no vistos, lo que derrota su propósito, 37

R

Red neuronal convolucional
Una red neuronal convolucional es un tipo de red neuronal artificial donde las neuronas artificiales corresponden a campos receptivos de una manera muy similar a las neuronas en la corteza visual primaria (V1) de un cerebro biológico, 36

ReLU
La función de activación ReLU (Rectified Linear Unit) es una función no lineal que producirá la entrada directamente si es positiva, de lo contrario, producirá cero. Es la función de activación más comúnmente utilizada en redes neuronales, especialmente en redes neuronales convolucionales (CNN) y perceptrones multicapa, 78

RNN
Las redes neuronales recurrentes (RNN) son un tipo de redes neuronales que pueden procesar secuencias de datos, como texto, audio o vídeo. A diferencia de las redes neuronales convencionales, las RNN tienen conexiones entre las unidades ocultas que forman un ciclo, lo que les permite mantener una memoria de los estados anteriores, 38

S

softmax
La función softmax es una función que convierte un vector de K valores reales en un vector de K valores reales que suman 1. Los valores de entrada pueden ser positivos, negativos, cero o mayores que uno, pero la función softmax los transforma en valores entre 0 y 1, de modo que se puedan interpretar como probabilidades., 78, 80, 149, 150

streaming
La transmisión, o "streaming" , es la transmisión continua y reproducción de contenido multimedia, como audio o video, a través de Internet, 81, 98

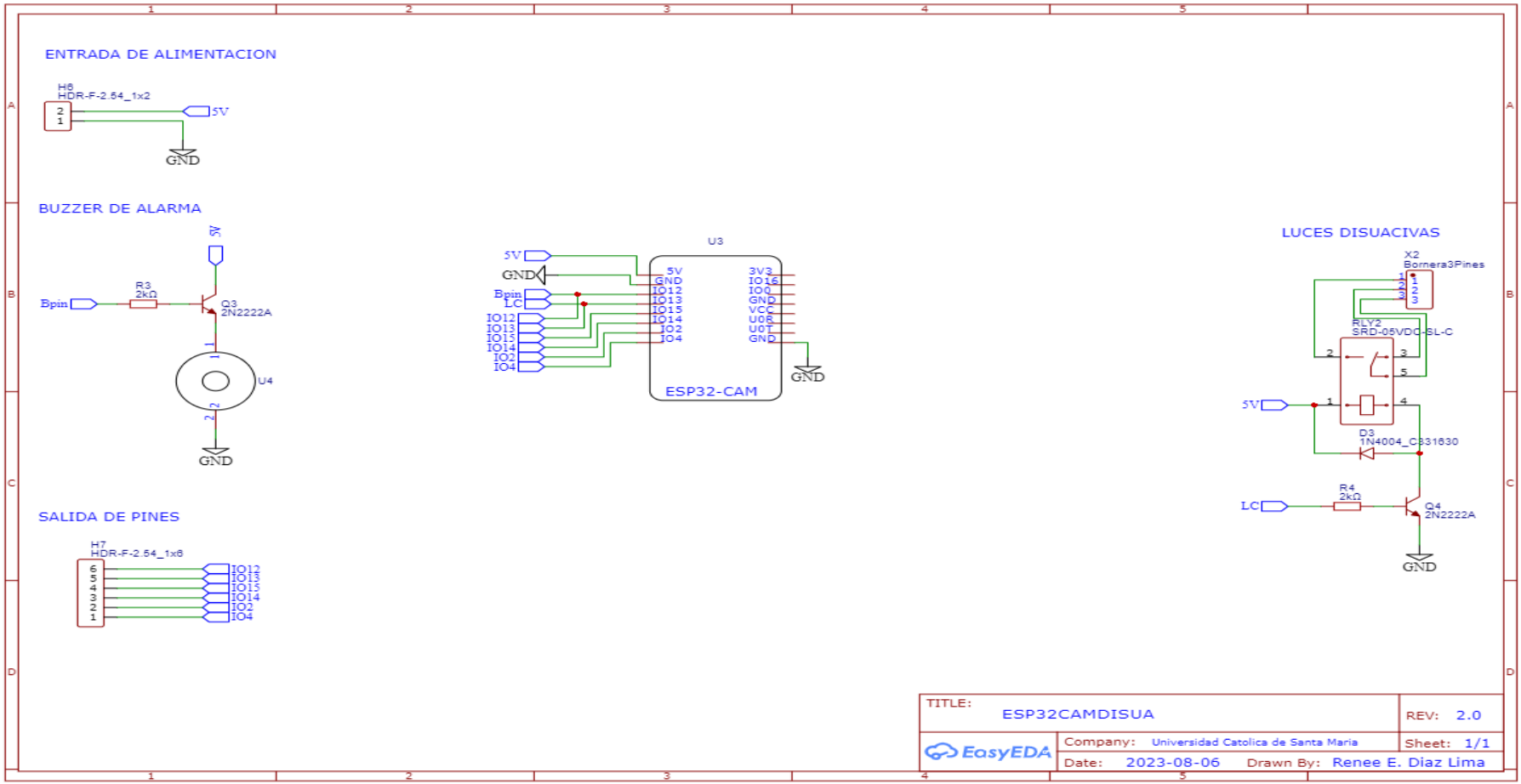
submuestreo
es una tecnica que se utiliza para reducir la dimensionalidad espacial de la representación de características, 36, 79

W

WebSocket
Un WebSocket es un protocolo de comunicación bidireccional y full-duplex que permite la comunicación en tiempo real entre un cliente y un servidor a través de una conexión persistente, 27, 94, 96, 97, 98, 99

ANEXOS

Anexo 1: Diagrama Electrónico



Anexo 2: Código Arduino para módulo disuasorio ESP32-CAM

```
// LIBRERÍAS USADAS
//WEB
#include "esp_camera.h"
#include <Arduino.h>
#include <WiFi.h>
#include <AsyncTCP.h>
#include <ESPAsyncWebServer.h>
#include "soc/soc.h"
#include "soc/rtc_cntl_reg.h"
//*******************************************************************************
//CREDENCIALES WIFI
//*******************************************************************************
//IP STATICA
IPAddress local_IP(192, 168, 137, 200);
IPAddress gateway(192, 168, 137, 1);
IPAddress subnet(255, 255, 255, 0);
//CREDENCIALES WIFI
const char* ssid = "LAPRENE";
const char* password = "Akamegakill1234";
//*******************************************************************************
// CONFIGURACIONES INICIALES SERVIDOR WEB
//*******************************************************************************

// Servidor  AsyncWebServer en puerto http 80
AsyncWebServer server(80);
AsyncWebSocket wc("/wc");



//*****************************************************************
//Variables Globales
//*****************************************************************
int tipo=0;
int ledPin=04;
int Bpin=12;
int Rpin=13;

//*******************************************************************************
// INICIO DE MÓDULO
//*******************************************************************************
//*****************************************************************
//INICIO WIFI
```

```
void initWiFi() {
  WiFi.mode(WIFI_STA);
  WiFi.config(ip, gateway, subnet);
  WiFi.begin(ssid, password);
  Serial.print("wifi conectado ..");
  while (WiFi.status() != WL_CONNECTED) {
    Serial.print('.');
    delay(1000);
  }
  Serial.println(WiFi.localIP());
}
//*****************************************************************
//configuracion de cámara
typedef struct {
        camera_fb_t * fb;
        size_t index;
} camera_frame_t;

#define PART_BOUNDARY "123456789000000000000987654321"
static const char* STREAM_CONTENT_TYPE = "multipart/x-mixed-replace;boundary="
PART_BOUNDARY;
static const char* STREAM_BOUNDARY = "\r\n--" PART_BOUNDARY "\r\n";
static const char* STREAM_PART = "Content-Type: %s\r\nContent-Length: %u\r\n\r\n";
static const char * JPG_CONTENT_TYPE = "image/jpeg";


// pines de cámara
#define PWDN_GPIO_NUM     32
#define RESET_GPIO_NUM    -1
#define XCLK_GPIO_NUM      0
#define SIOD_GPIO_NUM     26
#define SIOC_GPIO_NUM     27

#define Y9_GPIO_NUM       35
#define Y8_GPIO_NUM       34
#define Y7_GPIO_NUM       39
#define Y6_GPIO_NUM       36
#define Y5_GPIO_NUM       21
#define Y4_GPIO_NUM       19
#define Y3_GPIO_NUM       18
#define Y2_GPIO_NUM        5
#define VSYNC_GPIO_NUM    25
#define HREF_GPIO_NUM     23
#define PCLK_GPIO_NUM     22
```

```
camera_fb_t * fb = NULL;

////////////////////////////////////////////////
//inicio de cámara
void initCámara()
{
  camera_config_t config;
  config.ledc_channel = LEDC_CHANNEL_0;
  config.ledc_timer = LEDC_TIMER_0;
  config.pin_d0 = Y2_GPIO_NUM;
  config.pin_d1 = Y3_GPIO_NUM;
  config.pin_d2 = Y4_GPIO_NUM;
  config.pin_d3 = Y5_GPIO_NUM;
  config.pin_d4 = Y6_GPIO_NUM;
  config.pin_d5 = Y7_GPIO_NUM;
  config.pin_d6 = Y8_GPIO_NUM;
  config.pin_d7 = Y9_GPIO_NUM;
  config.pin_xclk = XCLK_GPIO_NUM;
  config.pin_pclk = PCLK_GPIO_NUM;
  config.pin_vsync = VSYNC_GPIO_NUM;
  config.pin_href = HREF_GPIO_NUM;
  config.pin_sscb_sda = SIOD_GPIO_NUM;
  config.pin_sscb_scl = SIOC_GPIO_NUM;
  config.pin_pwdn = PWDN_GPIO_NUM;
  config.pin_reset = RESET_GPIO_NUM;
  config.xclk_freq_hz = 20000000;
  config.pixel_format = PIXFORMAT_JPEG;
  //init with high specs to pre-allocate larger buffers
  if(psramFound()){
    config.frame_size = FRAMESIZE_SVGA;
    config.jpeg_quality = 10;
    config.fb_count = 2;
  } else {
    config.frame_size = FRAMESIZE_SVGA;
    config.jpeg_quality = 12;
    config.fb_count = 1;
  }
  // camera init
  esp_err_t err = esp_camera_init(&config);
  if (err != ESP_OK) {
    Serial.printf("Camera init failed with error 0x%x", err);
    return;
  }
```

```
}

////////////////////////////////////////////////////////
///serverstreaming
//  response->addHeader("Access-Control-Allow-Origin", "*");
// Esto permite el acceso desde cualquier origen
// response->addHeader("Cache-Control", "no-store");
////////////////////////////////////////////////////////
class AsyncJpegStreamResponse: public AsyncAbstractResponse {
    private:
        camera_frame_t _frame;
        size_t _index;
        size_t _jpg_buf_len;
        uint8_t * _jpg_buf;
        uint64_t lastAsyncRequest;
    public:
        AsyncJpegStreamResponse(){
            _callback = nullptr;
            _code = 200;
            _contentLength = 0;
            _contentType = STREAM_CONTENT_TYPE;
            _sendContentLength = false;
            _chunked = true;
            _index = 0;
            _jpg_buf_len = 0;
            _jpg_buf = NULL;
            lastAsyncRequest = 0;
            memset(&_frame, 0, sizeof(camera_frame_t));
        }
        ~AsyncJpegStreamResponse(){
            if(_frame.fb){
                if(_frame.fb->format != PIXFORMAT_JPEG){
                    free(_jpg_buf);
                }
                esp_camera_fb_return(_frame.fb);
            }
        }
        bool _sourceValid() const {
            return true;
        }
        virtual size_t _fillBuffer(uint8_t *buf, size_t maxLen) override {
            size_t ret = _content(buf, maxLen, _index);
            if(ret != RESPONSE_TRY_AGAIN){
                _index += ret;
```

```
            }
            return ret;
        }
        size_t _content(uint8_t *buffer, size_t maxLen, size_t index){
            if(!_frame.fb || _frame.index == _jpg_buf_len){
                if(index && _frame.fb){
                    uint64_t end = (uint64_t)micros();
                    int fp = (end - lastAsyncRequest) / 1000;
                    log_printf("Size: %uKB, Time: %ums (%.1ffps)\n", _jpg_buf_len/1024,
fp);
                    lastAsyncRequest = end;
                    if(_frame.fb->format != PIXFORMAT_JPEG){
                        free(_jpg_buf);
                    }
                    esp_camera_fb_return(_frame.fb);
                    _frame.fb = NULL;
                    _jpg_buf_len = 0;
                    _jpg_buf = NULL;
                }
                if(maxLen < (strlen(STREAM_BOUNDARY) + strlen(STREAM_PART) +
strlen(JPG_CONTENT_TYPE) + 8)){
                    //log_w("Not enough space for headers");
                    return RESPONSE_TRY_AGAIN;
                }
                //get frame
                _frame.index = 0;

                _frame.fb = esp_camera_fb_get();
                if (_frame.fb == NULL) {
                    log_e("Camera frame failed");
                    return 0;
                }

                if(_frame.fb->format != PIXFORMAT_JPEG){
                    unsigned long st = millis();
                    bool jpeg_converted = frame2jpg(_frame.fb, 80, &_jpg_buf,
&_jpg_buf_len);
                    if(!jpeg_converted){
                        log_e("JPEG compression failed");
                        esp_camera_fb_return(_frame.fb);
                        _frame.fb = NULL;
                        _jpg_buf_len = 0;
                        _jpg_buf = NULL;
                        return 0;
```

```
                    }
                    log_i("JPEG: %lums, %uB", millis() - st, _jpg_buf_len);
                } else {
                    _jpg_buf_len = _frame.fb->len;
                    _jpg_buf = _frame.fb->buf;
                }

                //send boundary
                size_t blen = 0;
                if(index){
                    blen = strlen(STREAM_BOUNDARY);
                    memcpy(buffer, STREAM_BOUNDARY, blen);
                    buffer += blen;
                }
                //send header
                size_t hlen = sprintf((char *)buffer, STREAM_PART, JPG_CONTENT_TYPE,
_jpg_buf_len);
                buffer += hlen;
                //send frame
                hlen = maxLen - hlen - blen;
                if(hlen > _jpg_buf_len){
                    maxLen -= hlen - _jpg_buf_len;
                    hlen = _jpg_buf_len;
                }
                memcpy(buffer, _jpg_buf, hlen);
                _frame.index += hlen;
                return maxLen;
            }

            size_t available = _jpg_buf_len - _frame.index;
            if(maxLen > available){
                maxLen = available;
            }
            memcpy(buffer, _jpg_buf+_frame.index, maxLen);
            _frame.index += maxLen;

            return maxLen;
        }
};

void serverstreaming(AsyncWebServerRequest *request) {
  AsyncJpegStreamResponse *response = new AsyncJpegStreamResponse();
    if(!response){
        request->send(501);
```

```
        return;
    }
    response->addHeader("Access-Control-Allow-Origin", "*");
    request->send(response);
}


////////////////////////////////////////////////////////////
///serverjpg
////////////////////////////////////////////////////////////
void serveJpg(AsyncWebServerRequest *request) {
  // Capturar un frame
  camera_fb_t * frame = esp_camera_fb_get();
  if (!frame) {
    Serial.println("Error al capturar el frame");
    request->send(503, "", "");
    return;
  }

  // Enviar la imagen al cliente
  request->send_P(200, "image/jpeg", reinterpret_cast<uint8_t*>(frame->buf), frame->len);

  // Liberar el frame
  esp_camera_fb_return(frame);
}

//************************************************************************************
// Archivos HTML, CSS y JavaScript incrustados
//************************************************************************************
const char index_html[] PROGMEM = R"rawliteral(
<!DOCTYPE html>
<html>
<head>
  <title>Video en tiempo real</title>
  <style>
    body {
      display: flex;
      justify-content: center;
      align-items: center;
      height: 100vh;
      margin: 0;
    }
    img {
      width: 640px;
```

```
      height: 480px;
    }
  </style>
</head>
<body>
  <img id="video" src="">
  <script>
    // Obtener el nombre del host de la ventana actual
    var hostname = window.location.hostname;

    // Construir la URL del flujo de video usando el nombre del host
    var videoStreamUrl = "http://" + hostname + "/streaming";

    // Establecer el atributo src del elemento img en la URL del flujo de video
    document.getElementById("video").src = videoStreamUrl;
  </script>
</body>
</html>
)rawliteral";
 void serveIndex(AsyncWebServerRequest *request) {
  request->send(200, "text/html", index_html);
}
//*****************************************************************
//websokets
//*****************************************************************
//clasificador
void websoketclasificador(AsyncWebSocket *server, AsyncWebSocketClient *client,
AwsEventType type, void *arg, uint8_t *data, size_t len) {
 switch (type) {
    case WS_EVT_CONNECT:
      Serial.println("Cliente conectado clasificador");
      break;
    case WS_EVT_DISCONNECT:
      Serial.println("Cliente desconectado clasificador");
      break;
    case WS_EVT_DATA:
      if (len > 0) {
        String message = String((char*) data);
        int commaIndex = message.indexOf(",");
        if (commaIndex >= 0) {
          tipo = message.substring(0, commaIndex).toInt();
          Serial.print("tipo: ");
          Serial.print(tipo);
          Serial.print(", ");
```

```
          Serial.println(message);
        }
      }
      break;
  }
}

//*****************************************************************************
//Uso de nucleos ESP32
//*****************************************************************************
//-----------------------------------
//Nucleo1
//-----------------------------------
TaskHandle_t nucleo1;
//loop nucleo 1
void loopn1(void*parameter)
{
//seccion que solo se ejecutara una vez
//+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
++++++++++++++++++++++++++
Serial.println("\t\t\tNucleo"+String(xPortGetCoreID()));
delay(500);
Serial.println("Pruebas");
delay(500);
digitalWrite(Bpin, 1);
digitalWrite(Rpin, 1);
delay(1000);
digitalWrite(Bpin, 0);
digitalWrite(Rpin, 0);
delay(1000);

//+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
++++++++++++++++++++++++++

for(;;)
{
//loop
infinito//////////////////////////////////////////////////////////////////////////////
//////////////////////
if(tipo==1)
{
 delay(1000);
 digitalWrite(Bpin, 1);
 digitalWrite(Rpin, 1);
```

```
}
else
{
  delay(500);
  digitalWrite(Bpin, 0);
  digitalWrite(Rpin, 0);
}

//final loop
infinito////////////////////////////////////////////////////////////////////////////////
///////////////
}
}

void setup() {
  WRITE_PERI_REG(RTC_CNTL_BROWN_OUT_REG, 0);
  Serial.begin(115200);
  pinMode(ledPin, OUTPUT);
  pinMode(Bpin, OUTPUT);
  pinMode(Rpin, OUTPUT);
  initWiFi();
  initCámara();
  digitalWrite(ledPin, 1);
  delay(1000);
  digitalWrite(ledPin, 0);
  delay(1000);
  digitalWrite(ledPin, 1);
  delay(1000);
  digitalWrite(ledPin, 0);
  delay(1000);
////////////////////////////////////////////////////
//servidor web
 server.on("/jpg", HTTP_GET, [] (AsyncWebServerRequest *request) {
    serveJpg(request);
  });
 server.on("/streaming", HTTP_GET, [](AsyncWebServerRequest *request){
    serverstreaming(request);
  });
  server.on("/", HTTP_GET, serveIndex);
  server.begin();
  wc.onEvent(websoketclasificador);
  server.addHandler(&wc);
  ////////////////////////////////////////////////////
  ///Ejecucion de las tareas en el Nucleo 1
```

```
  xTaskCreatePinnedToCore(loopn1,"N1",16384,NULL,1,&nucleo1,1);
}
void loop() {}
```

Anexo 3: Código Python para el entrenamiento de modelos Denso y Convolucional

```
#creacion de carpetas para entrenamiento
!mkdir Anoviolentas
!mkdir Aviolentas
#Entrar a las carpetas y desconprimir
%cd Anoviolentas
!unzip AccionesnoViolentas.zip
%cd ..

%cd Aviolentas
!unzip AccionesViolentas.zip
%cd ..
#borrar archivos zip
!rm -rf /content/Anoviolentas/AccionesnoViolentas.zip
!rm -rf /content/Aviolentas/AccionesViolentas.zip
#mostrar algunas imagenes
import os
import matplotlib.pyplot as plt
import matplotlib.image as mpimg
plt.figure(figsize=(15,15))
carpeta = '/content/Anoviolentas'
imagenes= os.listdir(carpeta)
for i, nombreimg in enumerate(imagenes[:25]):
 plt.subplot(5,5,i+1)
 imagen=mpimg.imread(carpeta+'/'+nombreimg)
 plt.imshow(imagen)
#crear carpeta dataset y subcarpetas
!mkdir dataset
!mkdir dataset/Anoviolentas
!mkdir dataset/Aviolentas
#copiar imagenes y limitarlas apra que todas tengan el mismo numero
#numero minimo 1103
import shutil
Cfuente='/content/Anoviolentas'
Cdestino='/content/dataset/Anoviolentas'
```

```
imagenes=os.listdir(Cfuente)

for i, nombreimg in enumerate(imagenes):
  if i<1103:#numero minimo 828
    shutil.copy(Cfuente+'/'+nombreimg,Cdestino+'/'+nombreimg)
Cfuente='/content/Aviolentas'
Cdestino='/content/dataset/Aviolentas'

imagenes=os.listdir(Cfuente)

for i, nombreimg in enumerate(imagenes):
  if i<1103:#numero minimo 828
    shutil.copy(Cfuente+'/'+nombreimg,Cdestino+'/'+nombreimg)
#aumentar datos con ImageDataGenerator
from tensorflow.keras.preprocessing.image import ImageDataGenerator
import numpy as np

#crear dataset generador
datagen= ImageDataGenerator(
    rescale=1. / 255,
    rotation_range = 30,
    width_shift_range=30 ,
    shear_range = 15,
    zoom_range=[0.5,1.5],
    validation_split=0.2 #20% para pruebas lo que tambien sera 80% para entrenar
)

#generadores para dataset
#entrenamiento
data_gen_entrenamiento=
datagen.flow_from_directory('/content/dataset/',target_size=(224,224),
                                                batch_size=32, shuffle=True,
subset='training')
#pruebas
data_gen_pruebas=
datagen.flow_from_directory('/content/dataset/',target_size=(224,224),
                                                batch_size=32, shuffle=True,
subset='validation')
#imprimir 10 imagenes preparadas
for imagen, etiqueta in data_gen_entrenamiento:
  for i in range(10):
    plt.subplot(2,5,i+1)
    plt.xticks([])
    plt.yticks([])
```

```
    plt.imshow(imagen[i])
  break
  plt.show()
from tensorflow.keras.callbacks import TensorBoard
tdenso= TensorBoard(log_dir='logs/denso')
tconv= TensorBoard(log_dir='logs/convolucional')
tcond= TensorBoard(log_dir='logs/convolucionaldrop')
# Modelo denso
from tensorflow.keras.layers import Dense, Flatten
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Conv2D, MaxPooling2D, Flatten, Dense, Dropout
# Crear el modelo
model = Sequential()
model.add(Flatten(input_shape=(224,224,3))) # Aplanar las imagenes para poder ser
procesadas
model.add(Dense(128, activation='relu')) # Capa densa con 128 unidades y función de
activación relu
model.add(Dense(64, activation='relu')) # Capa densa con 64 unidades y función de
activación relu
model.add(Dense(2, activation='softmax')) # Capa densa con 2 unidades y función de
activación softmax

# Compilar el modelo
model.compile(optimizer='adam', loss='categorical_crossentropy',
metrics=['accuracy'])

# Entrenar el modelo
history = model.fit(data_gen_entrenamiento, epochs=50,
validation_data=data_gen_pruebas,callbacks=[tdenso])
# Modelo denso convolucional

# Crear el modelo
model1 = Sequential()
model1.add(Conv2D(32, (3,3), activation='relu', input_shape=(224,224,3))) # Capa
convocional con 32 filtros y tamaño de filtro 3x3
model1.add(MaxPooling2D((2,2))) # Capa de pooling con tamaño de ventana 2x2
model1.add(Conv2D(64, (3,3), activation='relu')) # Capa convocional con 64 filtros
y tamaño de filtro 3x3
model1.add(MaxPooling2D((2,2))) # Capa de pooling con tamaño de ventana 2x2
model1.add(Flatten()) # Aplanar las imagenes para poder ser procesadas
model1.add(Dense(128, activation='relu')) # Capa densa con 128 unidades y función
de activación relu
model1.add(Dense(2, activation='softmax')) # Capa densa con 2 unidades y función de
activación softmax
```

```
# Compilar el modelo
model1.compile(optimizer='adam', loss='categorical_crossentropy',
metrics=['accuracy'])

# Entrenar el modelo
history = model1.fit(data_gen_entrenamiento, epochs=50,
validation_data=data_gen_pruebas,callbacks=[tconv])
# Modelo denso convolucional con dropout
# Crear el modelo
model2 = Sequential()
model2.add(Conv2D(32, (3,3), activation='relu', input_shape=(224,224,3))) # Capa
convocional con 32 filtros y tamaño de filtro 3x3
model2.add(MaxPooling2D((2,2))) # Capa de pooling con tamaño de ventana 2x2
model2.add(Conv2D(64, (3,3), activation='relu')) # Capa convocional con 64 filtros
y tamaño de filtro 3x3
model2.add(MaxPooling2D((2,2))) # Capa de pooling con tamaño de ventana 2x2
model2.add(Flatten()) # Aplanar las imagenes para poder ser procesadas
model2.add(Dense(128, activation='relu')) # Capa densa con 128 unidades y función
de activación relu
model2.add(Dropout(0.5)) # Capa de descarte con tasa de descarte 0.5
model2.add(Dense(2, activation='softmax')) # Capa densa con 2 unidades y función de
activación softmax

# Compilar el modelo
model2.compile(optimizer='adam', loss='categorical_crossentropy',
metrics=['accuracy'])

# Entrenar el modelo
history = model2.fit(data_gen_entrenamiento, epochs=50,
validation_data=data_gen_pruebas,callbacks=[tcond])
%load_ext tensorboard
%tensorboard --logdir logs
#guardamos los modelos
model.save('modelodenso.h5')
model1.save('modeloconvolucional.h5')
model2.save('modeloconvolucionaldrop.h5')
#modelo guardado para js
!tensorflowjs_converter --input_format keras modeloconvolucional.h5 carpeta_salida
```

Anexo 4: Código Python para el entrenamiento de modelo Movinet

```
# -*- coding: utf-8 -*-
"""Movinet TESIS2023.ipynb

Automatically generated by Colaboratory.

Original file is located at
    https://colab.research.google.com/drive/1o6_Qg2hsUFRSgEy2llPI0DOLD_t24I7A

# Cargar Archivos Google Drive
"""

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

import shutil

# Montar Google Drive
drive.mount('/content/drive',force_remount=True)

# Ruta del archivo en Google Drive
ruta_archivo_drive = '/content/drive/MyDrive/TESIS2023/DATAVIDEOS.zip'

# Ruta de la carpeta en Google Colab
ruta_carpeta_colab = '/content/'

# Copiar el archivo a la carpeta en Google Colab
shutil.copy(ruta_archivo_drive, ruta_carpeta_colab)

"""# Crear dataset de videos"""

# The way this tutorial uses the `TimeDistributed` layer requires TF>=2.10
!pip install -U "tensorflow>=2.10.0"
```

```
!pip install remotezip tqdm opencv-python
!pip install -q git+https://github.com/tensorflow/docs

import tqdm
import random
import pathlib
import itertools
import collections

import os
import cv2
import numpy as np
import remotezip as rz

import tensorflow as tf

# Some modules to display an animation using imageio.
import imageio
from IPython import display
from urllib import request
from tensorflow_docs.vis import embed

URL = '/content/DATAVIDEOS.zip'

"""## Archivos en el ZIP"""

import zipfile

zip_path = '/content/DATAVIDEOS.zip'

# Obtener la lista de archivos dentro del archivo ZIP
with zipfile.ZipFile(zip_path, 'r') as zip_file:
    files = zip_file.namelist()

# Filtrar los archivos para incluir solo los que terminan con '.mp4'
avi_files = [f for f in files if f.endswith('.mp4')]

# Obtener los primeros 10 archivos (si hay suficientes)
first_10_files = avi_files[:10]

# Imprimir los primeros 400 archivos
print(first_10_files)

def get_class(fname):
```

```
    """Recupera el nombre de la clase dada una nombre de archivo.

    Args:
        fname: Nombre del archivo en el conjunto de datos UCF101.

    Returns:
        Clase a la que pertenece el archivo.
    """
    return fname.split('-')[0]

import collections

def get_files_per_class(files):
    """Recupera los archivos que pertenecen a cada clase.

    Args:
        files: Lista de archivos en el conjunto de datos.

    Returns:
        Diccionario de nombres de clase (clave) y archivos (valores).
    """
    files_for_class = collections.defaultdict(list)
    for fname in files:
        class_name = get_class(fname)
        files_for_class[class_name].append(fname)
    return files_for_class

NUM_CLASSES = 2
FILES_PER_CLASS = 100

files_for_class = get_files_per_class(files)
classes = list(files_for_class.keys())

print('Num classes:', len(classes))
print('Num videos for class[0]:', len(files_for_class[classes[0]]))

def select_subset_of_classes(files_for_class, classes, files_per_class):
    """Crea un diccionario con el nombre de clase y un subconjunto de los archivos de esa
clase.

    Args:
        files_for_class: Diccionario de nombres de clase (clave) y archivos (valores).
        classes: Lista de clases.
        files_per_class: Número de archivos por clase de interés.
```

```
    Returns:
        Diccionario con la clase como clave y una lista del número especificado de archivos
de video en esa clase.
    """
    files_subset = dict()

    for class_name in classes:
        class_files = files_for_class[class_name]
        files_subset[class_name] = class_files[:files_per_class]

    return files_subset

files_subset = select_subset_of_classes(files_for_class, classes[:NUM_CLASSES],
FILES_PER_CLASS)
list(files_subset.keys())

def archivozip(ruta_archivo_zip, directorio_destino, nombres_archivos):
  """desconprime el contenido del archivo zip desde la ruta local proporcionada.

    Args:
      ruta_archivo_zip: Ruta local del archivo ZIP que contiene los datos.
      directorio_destino: Directorio donde se descargarán los datos.
      nombres_archivos: Nombres de los archivos a descargar.
  """
  with zipfile.ZipFile(ruta_archivo_zip, 'r') as zip:
    for nombre_archivo in tqdm.tqdm(nombres_archivos):
      class_name = get_class(nombre_archivo)
      zip.extract(nombre_archivo, str(directorio_destino / class_name))
      archivo_extraido = directorio_destino / class_name / nombre_archivo

      nombre_archivo = pathlib.Path(nombre_archivo).parts[-1]
      archivo_salida = directorio_destino / class_name / nombre_archivo
      archivo_extraido.rename(archivo_salida)

def split_class_lists(archivos_por_clase, cantidad):
  """Devuelve la lista de archivos que pertenecen a un subconjunto de datos, así como el
resto de
    archivos que necesitan ser descargados.

    Args:
      archivos_por_clase: Archivos que pertenecen a una clase particular de datos.
      cantidad: Número de archivos a descargar.
```

```
    Returns:
      Archivos que pertenecen al subconjunto de datos y un diccionario con el resto de
archivos que necesitan ser descargados.
  """
  archivos_division = []
  resto = {}
  for clase in archivos_por_clase:
    archivos_division.extend(archivos_por_clase[clase][:cantidad])
    resto[clase] = archivos_por_clase[clase][cantidad:]
  return archivos_division, resto


def division(ruta_archivo_zip, num_clases, divisiones, directorio_descarga):
  """Archivo dividido en partes, como entrenamiento, validación y prueba.

    Args:
      ruta_archivo_zip: Ruta local del archivo ZIP que contiene los datos.
      num_clases: Número de etiquetas.
      divisiones: Diccionario que especifica la división (clave) de los datos (valor es el
número de archivos por división),
                como entrenamiento, validación, prueba, etc.
      directorio_descarga: Directorio donde se descargarán los datos.

    Return:
      Mapeo de los directorios que contienen las secciones del subconjunto de datos.
  """
  with zipfile.ZipFile(ruta_archivo_zip, 'r') as zip:
    archivos = zip.namelist()

    archivos_por_clase = get_files_per_class(archivos)

    clases = list(archivos_por_clase.keys())[:num_clases]

    for clase in clases:
      random.shuffle(archivos_por_clase[clase])

    # Utiliza solo la cantidad de clases que deseas en el diccionario
    archivos_por_clase = {x: archivos_por_clase[x] for x in clases}

    directorios = {}
    for nombre_division, cantidad_archivos in divisiones.items():
      print(nombre_division, ":")
      directorio_division = directorio_descarga / nombre_division
```

```
      archivos_division, archivos_por_clase = split_class_lists(archivos_por_clase,
cantidad_archivos)
      archivozip(ruta_archivo_zip, directorio_division, archivos_division)
      directorios[nombre_division] = directorio_division

  return directorios


def get_class(nombre_archivo):
  """Obtiene el nombre de la clase a partir del nombre del archivo.

  Args:
    nombre_archivo: Nombre del archivo.

  Returns:
    Nombre de la clase.
  """
  # Implementa tu lógica para extraer el nombre de la clase a partir del nombre del archivo
  # Puedes usar patrones de coincidencia, expresiones regulares, etc.
  # En esta implementación de ejemplo, se utiliza una lógica de extracción simple asumiendo
que el nombre de la clase está antes del guion bajo ('_')
  return nombre_archivo.split('-')[0]


def get_files_per_class(archivos):
  """Agrupa los archivos por clase.

  Args:
    archivos: Lista de nombres de archivos.

  Returns:
    Diccionario que mapea el nombre de la clase a una lista de archivos pertenecientes a
esa clase.
  """
  archivos_por_clase = {}
  for archivo in archivos:
    clase = get_class(archivo)
    if clase not in archivos_por_clase:
      archivos_por_clase[clase] = []
    archivos_por_clase[clase].append(archivo)
  return archivos_por_clase


ruta_archivo_zip = '/content/DATAVIDEOS.zip'
```

```
directorio_descarga = pathlib.Path('/content/Datasetv/')
num_clases = 2  # Número de clases que deseas incluir en el subconjunto
divisiones = {"entrenamiento": 80, "validacion": 10, "prueba": 10}  # Número de archivos
por división

subset_paths = division(ruta_archivo_zip, num_clases, divisiones, directorio_descarga)

video_count_train = len(list(directorio_descarga.glob('entrenamiento/*/*.mp4')))
video_count_val = len(list(directorio_descarga.glob('validacion/*/*.mp4')))
video_count_test = len(list(directorio_descarga.glob('prueba/*/*.mp4')))
video_total = video_count_train + video_count_val + video_count_test
print(f"Total videos: {video_total}")

"""# Fotogramas a partir de videos"""

def format_frames(frame, output_size):
  """
    Rellene y cambie el tamaño de una imagen de un vídeo.

    Argumentos:
      marco: Imagen que necesita cambiarse de tamaño y rellenarse.
      Output_size: tamaño de píxel de la imagen del cuadro de salida.

    Devolver:
      Marco formateado con relleno del tamaño de salida especificado.
  """
  frame = tf.image.convert_image_dtype(frame, tf.float32)
  frame = tf.image.resize_with_pad(frame, *output_size)
  return frame

def frames_from_video_file(video_path, n_frames, output_size = (224,224), frame_step = 15):
  """
    Crea fotogramas a partir de cada archivo de vídeo presente para cada categoría.

    Argumentos:
      video_path: ruta del archivo al vídeo.
      n_frames: Número de fotogramas que se crearán por archivo de vídeo.
      Output_size: tamaño de píxel de la imagen del cuadro de salida.

    Devolver:
      Una matriz NumPy de marcos con la forma de (n_frames, alto, ancho, canales).
  """
  # Read each video frame by frame
  result = []
```

```
  src = cv2.VideoCapture(str(video_path))

  video_length = src.get(cv2.CAP_PROP_FRAME_COUNT)

  need_length = 1 + (n_frames - 1) * frame_step

  if need_length > video_length:
    start = 0
  else:
    max_start = video_length - need_length
    start = random.randint(0, max_start + 1)

  src.set(cv2.CAP_PROP_POS_FRAMES, start)
  # ret is a boolean indicating whether read was successful, frame is the image itself
  ret, frame = src.read()
  result.append(format_frames(frame, output_size))

  for _ in range(n_frames - 1):
    for _ in range(frame_step):
      ret, frame = src.read()
    if ret:
      frame = format_frames(frame, output_size)
      result.append(frame)
    else:
      result.append(np.zeros_like(result[0]))
  src.release()
  result = np.array(result)[..., [2, 1, 0]]

  return result

"""## Gif de videoclip con los fotogramas

"""

def to_gif(images):
  converted_images = np.clip(images * 255, 0, 255).astype(np.uint8)
  imageio.mimsave('./animation.gif', converted_images, duration=0.1)
  return embed.embed_file('./animation.gif')

# Suponiendo que 'subset_paths' y 'validacion' están definidos
video1 = frames_from_video_file(next(subset_paths['validacion'].glob('*/*.mp4')), 10)
video1.shape

to_gif(video1)
```

```
"""# Generacion de tensores"""

class FrameGenerator:
  def __init__(self, path, n_frames, training = False):
    """ Devuelve un conjunto de fotogramas con su etiqueta asociada.

        Args:
        path: Rutas de archivos de video.
        n_frames: Número de fotogramas.
        training: Booleano para determinar si se está creando el conjunto de datos de
entrenamiento.
    """
    self.path = path
    self.n_frames = n_frames
    self.training = training
    self.class_names = sorted(set(p.name for p in self.path.iterdir() if p.is_dir()))
    self.class_ids_for_name = dict((name, idx) for idx, name in
enumerate(self.class_names))

  def get_files_and_class_names(self):
    video_paths = list(self.path.glob('*/*.mp4'))
    classes = [p.parent.name for p in video_paths]
    return video_paths, classes

  def __call__(self):
    video_paths, classes = self.get_files_and_class_names()

    pairs = list(zip(video_paths, classes))

    if self.training:
      random.shuffle(pairs)

    for path, name in pairs:
      video_frames = frames_from_video_file(path, self.n_frames)
      label = self.class_ids_for_name[name] # Encode labels
      yield video_frames, label

"""## Prueba del generador"""

fg = FrameGenerator(subset_paths['validacion'], 10, training=True)

frames, label = next(fg())
```

```
print(f"Forma: {frames.shape}")
print(f"Etiqueta: {label}")

"""# Conjuntos de entrenamiento,prueba y validacion"""

# Create the training set
output_signature = (tf.TensorSpec(shape = (None, None, None, 3), dtype = tf.float32),
                    tf.TensorSpec(shape = (), dtype = tf.int16))
train_ds = tf.data.Dataset.from_generator(FrameGenerator(subset_paths['entrenamiento'], 10,
training=True),
                                          output_signature = output_signature)

"""ver el barajamiento de los datos de entrenamiento"""

for frames, labels in train_ds.take(10):
  print(labels)

# Create the validation set
val_ds = tf.data.Dataset.from_generator(FrameGenerator(subset_paths['validacion'], 10),
                                        output_signature = output_signature)

train_frames, train_labels = next(iter(train_ds))
print(f'Forma del conjunto de entrenamiento de fotogramas: {train_frames.shape}')
print(f'Forma de las etiquetas de entrenamiento: {train_labels.shape}')

val_frames, val_labels = next(iter(val_ds))
print(f'Forma del conjunto de validación de fotogramas: {val_frames.shape}')
print(f'Forma de las etiquetas de validación: {val_labels.shape}')

# Create the pruebas set
test_ds = tf.data.Dataset.from_generator(FrameGenerator(subset_paths['prueba'], 10),
                                        output_signature = output_signature)

test_frames, test_labels = next(iter(test_ds))
print(f'Forma del conjunto de validación de fotogramas: {test_frames.shape}')
print(f'Forma de las etiquetas de validación: {test_labels.shape}')

"""Optimizacion para el entrenamiento


*   **Dataset.cache:**mantiene los conjuntos de cuadros en la memoria después de que se cargan fuera del disco durante la primera época. Esta función garantiza que el conjunto de datos no se convierta en un cuello de botella mientras entrena su modelo. Si su conjunto de
```

```
datos es demasiado grande para caber en la memoria, también puede usar este método para
crear un caché en disco de rendimiento.
*   **Dataset.prefetch:**superpone el preprocesamiento de datos y la ejecución del modelo
durante el entrenamiento. Hacer referencia a Mejor rendimiento con el tf.data para más
detalles.

"""

AUTOTUNE = tf.data.AUTOTUNE

train_ds = train_ds.cache().shuffle(1000).prefetch(buffer_size = AUTOTUNE)
val_ds = val_ds.cache().shuffle(1000).prefetch(buffer_size = AUTOTUNE)

train_ds = train_ds.batch(2)
val_ds = val_ds.batch(2)

train_frames, train_labels = next(iter(train_ds))
print(f'Forma del conjunto de entrenamiento de fotogramas: {train_frames.shape}')
print(f'Forma de las etiquetas de entrenamiento: {train_labels.shape}')

val_frames, val_labels = next(iter(val_ds))
print(f'Forma del conjunto de validación de fotogramas: {val_frames.shape}')
print(f'Forma de las etiquetas de validación: {val_labels.shape}')

test_ds = test_ds.cache().shuffle(1000).prefetch(buffer_size = AUTOTUNE)

test_ds = test_ds.batch(2)

test_frames, test_labels = next(iter(test_ds))
print(f'Forma del conjunto de entrenamiento de fotogramas: {test_frames.shape}')
print(f'Forma de las etiquetas de entrenamiento: {test_labels.shape}')

"""# Modelo entrenamiento"""

pip cache purge

pip install --upgrade pip setuptools

!pip install -U tf-models-official

!pip install -U -q "tf-models-official"
from official.projects.movinet.modeling import movinet
from official.projects.movinet.modeling import movinet_model
from official.projects.movinet.tools import export_saved_model
```

```
from official.projects.movinet.modeling import movinet
from official.projects.movinet.modeling import movinet_model
from official.projects.movinet.tools import export_saved_model

"""### Construct the backbone with proper parameters"""

model_id = 'a2'
use_positional_encoding = model_id in {'a3', 'a4', 'a5'}
resolution = 224

backbone = movinet.Movinet(
    model_id=model_id,
    causal=True,
    conv_type='2plus1d',
    se_type='2plus3d',
    activation='hard_swish',
    gating_activation='hard_sigmoid',
    use_positional_encoding=use_positional_encoding,
    use_external_states=False,
)

import gc
gc.collect()

"""### Construct the model"""

# Note: this is a temporary model constructed for the
# purpose of loading the pre-trained checkpoint. Only
# the backbone will be used to build the custom classifier.

model = movinet_model.MovinetClassifier(
    backbone,
    num_classes=600,
    output_states=True)

# Create your example input here.
# Refer to the paper for recommended input shapes.
# inputs = tf.ones([1, 13, 172, 172, 3])
inputs = tf.ones([1, 10, 224, 224, 3])
# [Optional] Build the model and load a pretrained checkpoint.
model.build(inputs.shape)

batch_size=2
```

```
num_frames=10

"""### Load the pretrained weights"""

# Descargar y extraer los pesos preentrenados
!wget
https://storage.googleapis.com/tf_model_garden/vision/movinet/movinet_a2_stream.tar.gz -O
movinet_a2_stream.tar.gz -q
!tar -xvf movinet_a2_stream.tar.gz

# Suponiendo que has definido o importado 'model' de manera apropiada

# Ruta al directorio de puntos de control
checkpoint_dir = 'movinet_a2_stream'
checkpoint_path = tf.train.latest_checkpoint(checkpoint_dir)

# Verifica si se encontró un punto de control válido
if checkpoint_path:
    checkpoint = tf.train.Checkpoint(model=model)
    status = checkpoint.restore(checkpoint_path)
    status.assert_existing_objects_matched()
    print("Punto de control restaurado exitosamente.")
else:
    print("No se encontró ningún punto de control en la ruta especificada.")

"""### Set up the distribution strategy"""

# Detect hardware
try:
  tpu_resolver = tf.distribute.cluster_resolver.TPUClusterResolver() # TPU detection
except ValueError:
  tpu_resolver = None
  gpus = tf.config.experimental.list_logical_devices("GPU")

# Select appropriate distribution strategy
if tpu_resolver:
  tf.config.experimental_connect_to_cluster(tpu_resolver)
  tf.tpu.experimental.initialize_tpu_system(tpu_resolver)
  distribution_strategy = tf.distribute.experimental.TPUStrategy(tpu_resolver)
  print('Running on TPU ', tpu_resolver.cluster_spec().as_dict()['worker'])
elif len(gpus) > 1:
  distribution_strategy = tf.distribute.MirroredStrategy([gpu.name for gpu in gpus])
  print('Running on multiple GPUs ', [gpu.name for gpu in gpus])
elif len(gpus) == 1:
```

```
  distribution_strategy = tf.distribute.get_strategy() # default strategy that works on CPU
and single GPU
  print('Running on single GPU ', gpus[0].name)
else:
  distribution_strategy = tf.distribute.get_strategy() # default strategy that works on CPU
and single GPU
  print('Running on CPU')

print("Number of accelerators: ", distribution_strategy.num_replicas_in_sync)

"""### Construct custom classifier with required number of classes"""

def build_classifier(batch_size, num_frames, resolution, backbone, num_classes):
  """Builds a classifier on top of a backbone model."""
  model = movinet_model.MovinetClassifier(
      backbone=backbone,
      num_classes=num_classes)
  model.build([batch_size, num_frames, resolution, resolution, 3])

  return model

# Construct loss, optimizer and compile the model
with distribution_strategy.scope():
  model = build_classifier(batch_size, num_frames, resolution, backbone, 2)
  loss_obj = tf.keras.losses.SparseCategoricalCrossentropy(from_logits=True)
  optimizer = tf.keras.optimizers.Adam(learning_rate = 0.001)
  model.compile(loss=loss_obj, optimizer=optimizer, metrics=['accuracy'])

"""## Create a callback for storing the checkpoints"""

checkpoint_path = "trained_model/cp.ckpt"
checkpoint_dir = os.path.dirname(checkpoint_path)

# Create a callback that saves the model's weights
cp_callback = tf.keras.callbacks.ModelCheckpoint(filepath=checkpoint_path,
                                                 save_weights_only=True,
                                                 verbose=1)

"""## Train the model"""

historial = model.fit(train_ds,
                      validation_data=val_ds,
                      epochs=10,
                      validation_freq=1,
```

```
                      verbose=1,
                      callbacks=[cp_callback])

import matplotlib.pyplot as plt
def plot_history(history):
  """
    Plotting training and validation learning curves.

    Args:
      history: model history with all the metric measures
  """
  fig, (ax1, ax2) = plt.subplots(2)

  fig.set_size_inches(18.5, 10.5)

  # Plot loss
  ax1.set_title('Loss')
  ax1.plot(history.history['loss'], label = 'train')
  ax1.plot(history.history['val_loss'], label = 'test')
  ax1.set_ylabel('Loss')

  # Determine upper bound of y-axis
  max_loss = max(history.history['loss'] + history.history['val_loss'])

  ax1.set_ylim([0, np.ceil(max_loss)])
  ax1.set_xlabel('Epoch')
  ax1.legend(['Train', 'Validation'])

  # Plot accuracy
  ax2.set_title('Accuracy')
  ax2.plot(history.history['accuracy'],  label = 'train')
  ax2.plot(history.history['val_accuracy'], label = 'test')
  ax2.set_ylabel('Accuracy')
  ax2.set_ylim([0, 1])
  ax2.set_xlabel('Epoch')
  ax2.legend(['Train', 'Validation'])

  plt.show()

plot_history(historial)

"""## Evaluate the model"""

model.evaluate(test_ds, return_dict=True)
```

```
"""# Pruebas del modelo

"""

import cv2
import numpy as np

def video_to_tensor(video_path, num_frames, target_height, target_width):
    cap = cv2.VideoCapture(video_path)

    frames = []
    while True:
        ret, frame = cap.read()
        if not ret:
            break
        frames.append(frame)
        if len(frames) >= num_frames:
            break

    cap.release()

    # Resize frames to the target height and width
    resized_frames = [cv2.resize(frame, (target_width, target_height)) for frame in frames]

    # Normalize the frames (scale pixel values to [0, 1])
    normalized_frames = [frame.astype(np.float32) / 255.0 for frame in resized_frames]

    # Convert frames list to a numpy array
    tensor = np.array(normalized_frames)

    return tensor

# Example usage:
video_path = "/content/Datasetv/entrenamiento/NOvideo/NOvideo-01-1.mp4"
num_frames = 5
target_height = 224
target_width = 224
tensor2 = video_to_tensor(video_path, num_frames, target_height, target_width)

print("Tensor shape:", tensor2.shape)
 model.predict(tensor2)
```

Anexo 5: Código HTML página Web

```
<!DOCTYPE html>
<html>
<head>
    <title>Visualización y Clasificación de Cámaras IP</title>
    <meta charset="utf-8">
     <meta name="description" content="SISTEMA IOT PARA LA IDENTIFICACIÓN DE ACTOS DELICTIVOS Y
     EJECUCIÓN DE ACCIONES DISUASORIAS  ">
     <meta name="author" content="Renee Edmundo Diaz Lima">
    <style>
        * {
            box-sizing: border-box;
          }

        body, html {
            height: 100%;
            margin: 0;
            padding: 0;
        }

        .container {
            display: flex;
            height: 100%;
        }
```

```
24.
25.         .sidebar {
26.             width: 20%;
27.             background-color: lightgray;
28.             padding: 10px;
29.         }
30.
31.         .cameras {
32.             width: 80%;
33.             display: flex;
34.             flex-wrap: wrap;
35.             align-content: flex-start;
36.         }
37.
38.         .camera {
39.             width: 50%;
40.             height: 50%;
41.             padding: 10px;
42.             border: 1px solid black;
43.             background-color: rgb(253, 253, 253);
44.             box-sizing: border-box;
45.             display: flex;
46.             flex-direction: column;
47.             justify-content: space-between;
48.         }
49.
50.         .camera img,
51.         .camera video {
52.             width: 100%;
53.             height: 100%;
54.             object-fit: cover;
55.         }
56.
57.         .camera-name {
58.             margin: 5px 0;
59.             text-align: center;
60.             font-weight: bold;
61.         }
62.
63.         .classify-button {
64.             margin-top: 10px;
65.         }
66.         .prediction-row {
67.             display: flex;
68.             justify-content: space-between;
69.             align-items: center;
70.             margin-bottom: 5px;
```

```
        }

        .camera-name {
            font-weight: bold;
        }

    </style>
</head>
<body>
    <div class="container">
        <div class="sidebar">
            <h2>Configuración de Cámaras</h2>
            <div>
                <label for="camera1">Cámara 1:</label>
                <input type="text" id="camera1">
                <button onclick="connectCamera('camera1')">Conectar</button>
                <button onclick="showWebcam()">Mostrar Webcam</button>
            </div>
            <div>
                <label for="camera2">Cámara 2:</label>
                <input type="text" id="camera2">
                <button onclick="connectCamera('camera2')">Conectar</button>
            </div>
            <div>
                <label for="camera3">Cámara 3:</label>
                <input type="text" id="camera3">
                <button onclick="connectCamera('camera3')">Conectar</button>
            </div>
            <div>
                <label for="camera4">Cámara 4:</label>
                <input type="text" id="camera4">
                <button onclick="connectCamera('camera4')">Conectar</button>
            </div>
            <div>
                <button onclick="startClassification()">Iniciar Clasificación</button>
                <button onclick="stopClassification()">Detener Clasificación</button>
            </div>
            <h2>Resultados de Predicción</h2>
            <div id="predictionResults">
                <div class="prediction-row" id="predictionRow1">
                    <div class="camera-name">Cámara 1:</div>
                    <div class="predictions" id="predictions1">
                        <span class="prediction-value">0</span>
                        <span class="Accion-value">No Delictivo</span>
                    </div>
                </div>
                <div class="prediction-row" id="predictionRow2">
```

```
                    <div class="camera-name">Cámara 2:</div>
                    <div class="predictions" id="predictions2">
                        <span class="prediction-value">0</span>
                        <span class="Accion-value">No Delictivo</span>
                    </div>
                </div>
                <div class="prediction-row" id="predictionRow3">
                    <div class="camera-name">Cámara 3:</div>
                    <div class="predictions" id="predictions3">
                        <span class="prediction-value">0</span>
                        <span class="Accion-value">No Delictivo</span>
                    </div>
                </div>
                <div class="prediction-row" id="predictionRow4">
                    <div class="camera-name">Cámara 4:</div>
                    <div class="predictions" id="predictions4">
                        <span class="prediction-value">0</span>
                        <span class="Accion-value">No Delictivo</span>
                    </div>
                </div>
            </div>
        </div>
        <div class="cameras">
            <div class="camera" id="cameraView1">
                <div class="camera-name">Cámara 1</div>
            </div>
            <div class="camera" id="cameraView2">
                <div class="camera-name">Cámara 2</div>
            </div>
            <div class="camera" id="cameraView3">
                <div class="camera-name">Cámara 3</div>
            </div>
            <div class="camera" id="cameraView4">
                <div class="camera-name">Cámara 4</div>
            </div>
        </div>
    </div>

    <script src="https://code.jquery.com/jquery-3.6.0.min.js"></script>
    <script src="https://cdn.jsdelivr.net/npm/@tensorflow/tfjs/dist/tf.min.js"></script>
    <script>
//-----------------------------------------------------------------------------------------
-----------------
//Variables Globales
//-----------------------------------------------------------------------------------------
-----------------
    let canvas;
```

```
163.     let context;
164.     var tipo=0;
165.     var c=0;
166.     var c1=0;
167.     let cámara;
168.
169. //-------------------------------------------------------------------------------------------
-----------------
170. //Conexion de las cámaras
171. //-------------------------------------------------------------------------------------------
-----------------
172.         function connectCamera(cameraId) {
173.             var cameraInput = document.getElementById(cameraId);
174.             var cameraUrl = cameraInput.value;
175.
176.             if (cameraUrl !== "") {
177.                 var cameraView = document.getElementById("cameraView" + cameraId.slice(-
1));
178.                 cameraView.innerHTML = "<img src='" + cameraUrl + "' alt='Cámara " +
cameraId.slice(-1) + "'>";
179.                 var ipAddress = cameraUrl.match(/\/\/(.*?)\//)[1];
180.                 cámara=cameraId;
181.                 clasificador(ipAddress);
182.             }
183.         }
184.
185.         function showWebcam() {
186.             var cameraView = document.getElementById("cameraView1");
187.             navigator.mediaDevices.getUserMedia({ video: true })
188.                 .then(function(stream) {
189.                     var videoElement = document.createElement("video");
190.                     videoElement.srcObject = stream;
191.                     videoElement.autoplay = true;
192.                     videoElement.muted = true;
193.                     cameraView.innerHTML = "";
194.                     cameraView.appendChild(videoElement);
195.                 })
196.                 .catch(function(error) {
197.                     console.error("Error al acceder a la webcam: ", error);
198.                 });
199.         }
200. //-------------------------------------------------------------------------------------
201. //Intervalos de Prediccion y resultados
202. //-------------------------------------------------------------------------------------
203.         var classificationInterval;  // Variable para almacenar el intervalo de
clasificación
204.
```

```
205.         function startClassification() {
206.             classificationInterval = setInterval(classifyAllCameras, 1000);  // Clasificar
cada 900 milisegundos
207.         }
208.
209.         function stopClassification() {
210.             clearInterval(classificationInterval);  // Detener la clasificación al limpiar
el intervalo
211.         }
212.
213.         function updatePredictionResults(cameraNumber, predictionAverage,colorcam) {
214.             cámara=cameraNumber;
215.             var predictionRow = document.getElementById("predictionRow" + cameraNumber);
216.             //console.log("predictionRow:"+predictionRow);
217.             var predictionsElement = document.getElementById("predictions" + cameraNumber);
218.             //console.log("predictionsElement:"+predictionsElement);
219.             var predictionValueElement = predictionsElement.querySelector(".prediction-
value"); // Encuentra el elemento para el valor de predicción
220.             //console.log("predictionValueElement:"+predictionValueElement);
221.             var AccionvalueElement = predictionsElement.querySelector(".Accion-value");
222.
223.             var predictionValue = [0, 0];
224.             if (predictionAverage) {
225.                 predictionValue = [parseFloat(predictionAverage[0]),
parseFloat(predictionAverage[1])];
226.             }
227.
228.             // Actualiza el contenido del valor de predicción
229.             predictionValueElement.textContent = predictionValue.map(element =>
Number(element.toFixed(2)));
230.
231.             // Categorizar la imagen
232.             var categoriatext="No Clasificados";
233.             if (predictionAverage[0]<predictionAverage[1]) {
234.                 console.log("Categoría: Acción no Delictiva");
235.                 categoriatext="No Delictivo";
236.                 tipo=0;
237.                 send2();
238.                 colorcam.style.backgroundColor = "#00ff00";
239.             } else {
240.                 console.log("Categoría: Acción Delictiva");
241.                 categoriatext="Delictivo";
242.                 tipo=1;
243.                 send2();
244.                 colorcam.style.backgroundColor = "#ff0000";
245.             }
246.             // Actualiza el contenido del valor de categoria
```

```
247.             AccionvalueElement.textContent = categoriatext;
248.
249.         }
250.
251. //-------------------------------------------------------------------------------
252. //Modelo Clasificador
253. //-------------------------------------------------------------------------------
254.         function classifyAllCameras() {
255.                     var cameraViews = document.getElementsByClassName("camera");
256.
257.                     //cámara 1
258.                     var cameraView = cameraViews[0];
259.                     var imageElement = cameraView.querySelector("img, video")
260.                     if (imageElement) {
261.                             classifyImage(imageElement)
262.                                 .then(function(predictions) {
263.                                     console.log("predicciones para la cámara " + "1" + ": "
+ predictions);
264.                                     updatePredictionResults(1, predictions,cameraViews[0]);
265.                                 })
266.                                 .catch(function(error) {
267.                                     console.error("Error al clasificar la imagen: ",
error);
268.                                 });
269.                         }
270.                     //-------------------------------------------------------------
271.
272.                     //cámara 2
273.                     cameraView = cameraViews[1];
274.                     imageElement = cameraView.querySelector("img, video")
275.                     if (imageElement) {
276.                             classifyImage(imageElement)
277.                                 .then(function(predictions) {
278.                                     console.log("predicciones para la cámara " + "2" + ": "
+ predictions);
279.                                     updatePredictionResults(2, predictions,cameraViews[1]);
280.                                 })
281.                                 .catch(function(error) {
282.                                     console.error("Error al clasificar la imagen: ",
error,);
283.                                 });
284.                         }
285.                     //-------------------------------------------------------------
286.
287.                     //cámara 3
288.                     cameraView = cameraViews[2];
289.                     imageElement = cameraView.querySelector("img, video")
```

```
290.                     if (imageElement) {
291.                             classifyImage(imageElement)
292.                                 .then(function(predictions) {
293.                                     console.log("predicciones para la cámara " + "3" + ": "
+ predictions);
294.                                     updatePredictionResults(3, predictions,cameraViews[2]);
295.                                 })
296.                                 .catch(function(error) {
297.                                     console.error("Error al clasificar la imagen: ",
error);
298.                                 });
299.                         }
300.                     //------------------------------------------------------------
301.
302.                     //cámara4
303.                     cameraView = cameraViews[3];
304.                     imageElement = cameraView.querySelector("img, video")
305.                     if (imageElement) {
306.                             classifyImage(imageElement)
307.                                 .then(function(predictions) {
308.                                     console.log("predicciones para la cámara " + "4" + ": "
+ predictions);
309.                                     updatePredictionResults(4, predictions,
cameraViews[3]);
310.                                 })
311.                                 .catch(function(error) {
312.                                     console.error("Error al clasificar la imagen: ",
error);
313.                                 });
314.                         }
315.
316.                     //------------------------------------------------------------
317.                 }
318.
319.         async function classifyImage(image) {
320.         console.log(image);
321.         image.crossOrigin = "Anonymous";
322.         var canvas = document.createElement('canvas');
323.         var context = canvas.getContext('2d');
324.         canvas.width = 224;
325.         canvas.height = 224;
326.         context.drawImage(image, 0, 0, 224, 224);
327.         const imageData = context.getImageData(0, 0, 224, 224);
328.         console.log(imageData);
329.         const tensor = preprocessImage(imageData);
330.         await sendTensorToServer(tensor);
331.
```

```
332.          // Esperar el siguiente cuadro para continuar la clasificación
333.          await new Promise((resolve) => requestAnimationFrame(resolve));
334.          //retorno
335.          return await sendTensorToServer(tensor);
336.          }
337.
338.          function preprocessImage(imageData) {
339.          const tensor = tf.browser.fromPixels(imageData).toFloat();
340.          // Normalizar el tensor para que los valores estén en el rango [0, 1]
341.          const normalizedTensor = tensor.div(255.0);
342.          // Expandir el tensor para agregar las dimensiones de batch y num_frames
343.          const expandedTensor = normalizedTensor.expandDims(0).expandDims(1);
344.          return expandedTensor;
345.          }
346.
347.          async function sendTensorToServer(tensor) {
348.          const url = "http://localhost:8501/v1/models/EFNet:predict";
349.          const data = {
350.          instances: tensor.arraySync(),
351.          };
352.
353.          try {
354.          const response = await fetch(url, {
355.              method: "POST",
356.              body: JSON.stringify(data),
357.          });
358.
359.          if (!response.ok) {
360.              throw new Error("Error en la clasificación del video.");
361.          }
362.
363.          const result = await response.json();
364.          console.log("Resultado de la clasificación: ", result.predictions[0]); // Acceder
al array de predicciones
365.          return result.predictions[0]; // Devolver el array de predicciones
366.          } catch (err) {
367.          console.error("Error en la solicitud al servidor TensorFlow Serving: ", err);
368.          throw err;
369.          }
370.          }


371. //---------------------------------------------------------------------------------
372. //seccion de envio de informacion a ESP32---Clasificador
373. //---------------------------------------------------------------------------------
374. let ip;
375. let cámaraws;
```

```
let conectado=0;
var clasiw;
var tempt=0;

    function clasificador(iptemp){
        if(conectado==0)
            {
            ip=iptemp
            cámaraws=cámara;
            console.log("ws:"+cámaraws);
            var gateway = "ws://"+ip+"/wc";
            clasiw = new WebSocket(gateway);

            clasiw.onopen = onOpen;
            clasiw.onclose = onClose;
            }
        }

        function onOpen(event) {
        console.log('Connection opened');
        conectado=1;
        }

        function onClose(event) {
        console.log('Connection closed');
        }

        function send2(){
            console.log("camws"+cámaraws)
            var camenv="camera"+cámara
            console.log("cam"+camenv)
            if(cámaraws==camenv)
            {
                if(tipo!=tempt)
                {
                var mensaje = tipo + ",:Clasificacion";
                clasiw.send(mensaje);
                console.log('enviando dato clasificacion');
                tempt=tipo;
                }
            }
        }

        // Observador de cambios (change watcher) para detectar cambios en la variable
"clasificacion"
        Object.defineProperty(window, 'clasificacion', {
        get: function() {
```

```
422.             return this._clasificacion;
423.         },
424.         set: function(value) {
425.             this._clasificacion = value;
426.             send2();
427.         }
428.         });
429.
430.     </script>
431. </body>
432. </html>
433.
```

Anexo 6: ESP32 Hoja técnica

ESP32 Series
Datasheet

Including:

ESP32-D0WD

ESP32-D0WDQ6

ESP32-D2WD

ESP32-S0WD

Version 3.0
Espressif Systems

www.espressif.com

1. Overview

ESP32 is a single 2.4 GHz Wi-Fi-and-Bluetooth combo chip designed with the TSMC ultra-low-power 40 nm technology. It is designed to achieve the best power and RF performance, showing robustness, versatility and reliability in a wide variety of applications and power scenarios.

The ESP32 series of chips includes ESP32-D0WDQ6, ESP32-D0WD, ESP32-D2WD, and ESP32-S0WD. For details on part numbers and ordering information, please refer to Part Number and Ordering Information.

1.1 Featured Solutions

1.1.1 Ultra-Low-Power Solution

ESP32 is designed for mobile, wearable electronics, and Internet-of-Things (IoT) applications. It features all the state-of-the-art characteristics of low-power chips, including fine-grained clock gating, multiple power modes, and dynamic power scaling. For instance, in a low-power IoT sensor hub application scenario, ESP32 is woken up periodically and only when a specified condition is detected. Low-duty cycle is used to minimize the amount of energy that the chip expends. The output of the power amplifier is also adjustable, thus contributing to an optimal trade-off between communication range, data rate and power consumption.

Note:
For more information, refer to Section 3.7 RTC and Low-Power Management.

1.1.2 Complete Integration Solution

ESP32 is a highly-integrated solution for Wi-Fi-and-Bluetooth IoT applications, with around 20 external components. ESP32 integrates an antenna switch, RF balun, power amplifier, low-noise receive amplifier, filters, and power management modules. As such, the entire solution occupies minimal Printed Circuit Board (PCB) area.

ESP32 uses CMOS for single-chip fully-integrated radio and baseband, while also integrating advanced calibration circuitries that allow the solution to remove external circuit imperfections or adjust to changes in external conditions. As such, the mass production of ESP32 solutions does not require expensive and specialized Wi-Fi testing equipment.

1.2 Wi-Fi Key Features

- 802.11 b/g/n
- 802.11 n (2.4 GHz), up to 150 Mbps
- WMM
- TX/RX A-MPDU, RX A-MSDU
- Immediate Block ACK
- Defragmentation
- Automatic Beacon monitoring (hardware TSF)
- 4 × virtual Wi-Fi interfaces

- Simultaneous support for Infrastructure Station, SoftAP, and Promiscuous modes
  Note that when ESP32 is in Station mode, performing a scan, the SoftAP channel will be changed.
- Antenna diversity

Note:
For more information, please refer to Section 3.5 Wi-Fi.

1.3 BT Key Features

- Compliant with Bluetooth v4.2 BR/EDR and BLE specifications
- Class-1, class-2 and class-3 transmitter without external power amplifier
- Enhanced Power Control
- +12 dBm transmitting power
- NZIF receiver with –97 dBm BLE sensitivity
- Adaptive Frequency Hopping (AFH)
- Standard HCI based on SDIO/SPI/UART
- High-speed UART HCI, up to 4 Mbps
- Bluetooth 4.2 BR/EDR BLE dual mode controller
- Synchronous Connection-Oriented/Extended (SCO/eSCO)
- CVSD and SBC for audio codec
- Bluetooth Piconet and Scatternet
- Multi-connections in Classic BT and BLE
- Simultaneous advertising and scanning

1.4 MCU and Advanced Features

1.4.1 CPU and Memory

- Xtensa® single-/dual-core 32-bit LX6 microprocessor(s), up to 600 MIPS (200 MIPS for ESP32-S0WD, 400 MIPS for ESP32-D2WD)
- 448 KB ROM
- 520 KB SRAM
- 16 KB SRAM in RTC
- QSPI supports multiple flash/SRAM chips

1.4.2 Clocks and Timers

- Internal 8 MHz oscillator with calibration
- Internal RC oscillator with calibration
- External 2 MHz ~ 60 MHz crystal oscillator (40 MHz only for Wi-Fi/BT functionality)
- External 32 kHz crystal oscillator for RTC with calibration
- Two timer groups, including 2 × 64-bit timers and 1 × main watchdog in each group
- One RTC timer
- RTC watchdog

1.4.3 Advanced Peripheral Interfaces

- 34 × programmable GPIOs
- 12-bit SAR ADC up to 18 channels
- 2 × 8-bit DAC
- 10 × touch sensors
- 4 × SPI
- 2 × I²S
- 2 × I²C
- 3 × UART
- 1 host (SD/eMMC/SDIO)
- 1 slave (SDIO/SPI)
- Ethernet MAC interface with dedicated DMA and IEEE 1588 support
- CAN 2.0
- IR (TX/RX)
- Motor PWM
- LED PWM up to 16 channels
- Hall sensor

1.4.4 Security

- Secure boot
- Flash encryption
- 1024-bit OTP, up to 768-bit for customers
- Cryptographic hardware acceleration:
  - AES
  - Hash (SHA-2)
  - RSA
  - ECC

  - Random Number Generator (RNG)

1.5 Applications (A Non-exhaustive List)

- Generic Low-power IoT Sensor Hub
- Generic Low-power IoT Data Loggers
- Cameras for Video Streaming
- Over-the-top (OTT) Devices
- Speech Recognition
- Image Recognition
- Mesh Network
- Home Automation
  - Light control
  - Smart plugs
  - Smart door locks
- Smart Building
  - Smart lighting
  - Energy monitoring
- Industrial Automation
  - Industrial wireless control
  - Industrial robotics
- Smart Agriculture
  - Smart greenhouses
  - Smart irrigation
  - Agriculture robotics
- Audio Applications
  - Internet music players
  - Live streaming devices
  - Internet radio players
  - Audio headsets
- Health Care Applications
  - Health monitoring
  - Baby monitors
- Wi-Fi-enabled Toys
  - Remote control toys
  - Proximity sensing toys
  - Educational toys
- Wearable Electronics
  - Smart watches
  - Smart bracelets
- Retail & Catering Applications
  - POS machines
  - Service robots

1.6 Block Diagram

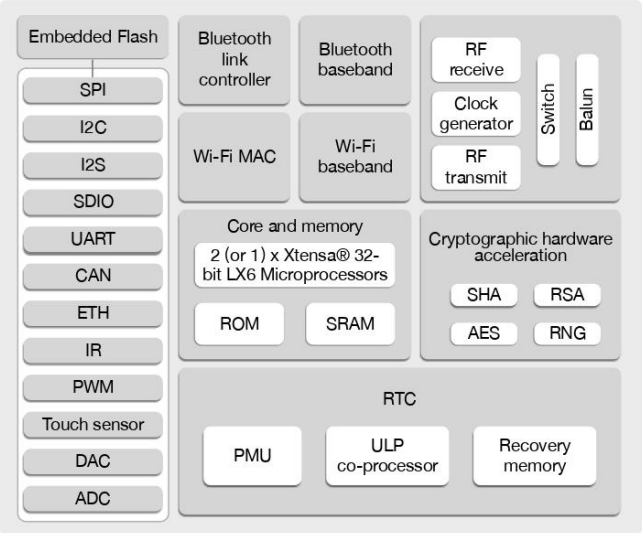


Figure 1: Functional Block Diagram

Note:
Products in the ESP32 series differ from each other in terms of their support for embedded flash and the number of CPUs they have. For details, please refer to Part Number and Ordering Information.

5. Electrical Characteristics

5.1 Absolute Maximum Ratings

Stresses beyond the absolute maximum ratings listed in the table below may cause permanent damage to the device. These are stress ratings only, and do not refer to the functional operation of the device that should follow the recommended operating conditions.

Table 11: Absolute Maximum Ratings

Symbol	Parameter	Min	Max	Unit
VDDA, VDD3P3, VDD3P3_RTC, VDD3P3_CPU, VDD_SDIO	Voltage applied to power supply pins per power domain	–0.3	3.6	V
I_{output} *	Cumulative IO output current	-	1,200	mA
T_{store}	Storage temperature	–40	150	°C

* The chip worked properly after a 24-hour test in ambient temperature at 25 °C, and the IOs in three domains (VDD3P3_RTC, VDD3P3_CPU, VDD_SDIO) output high logic level to ground.

5.2 Recommended Operating Conditions

Table 12: Recommended Operating Conditions

Symbol	Parameter	Min	Typical	Max	Unit
VDDA, VDD3P3_RTC [1] VDD3P3, VDD_SDIO (3.3 V mode) [2]	Voltage applied to power supply pins per power domain	2.3	3.3	3.6	V
VDD3P3_CPU	Voltage applied to power supply pin	1.8	3.3	3.6	V
I_{VDD}	Current delivered by external power supply	0.5	-	-	A
T [3]	Operating temperature	–40	-	125	°C

1. When writing eFuse, VDD3P3_RTC should be at least 3.3 V.
2. - VDD_SDIO works as the power supply for the related IO, and also for an external device. Please refer to the Appendix IO_MUX of this datasheet for more details.
   - VDD_SDIO can be sourced internally by the ESP32 from the VDD3P3_RTC power domain:
     - When VDD_SDIO operates at 3.3 V, it is driven directly by VDD3P3_RTC through a 6 Ω resistor, therefore, there will be some voltage drop from VDD3P3_RTC.
     - When VDD_SDIO operates at 1.8 V, it can be generated from ESP32's internal LDO. The maximum current this LDO can offer is 40 mA, and the output voltage range is 1.65 V ~ 2.0 V.
   - VDD_SDIO can also be driven by an external power supply.
   - Please refer to Power Scheme, section 2.3, for more information.
3. The operating temperature of ESP32-D2WD ranges from –40 °C ~ 105 °C, due to the flash embedded in it. The other chips in this series have no ebedded flash, so their range of operating temperatures is –40 °C ~ 125 °C.

5.3 DC Characteristics (3.3 V, 25 °C)

Table 13: DC Characteristics (3.3 V, 25 °C)

Symbol	Parameter		Min	Typ	Max	Unit
C_{IN}	Pin capacitance		-	2	-	pF
V_{IH}	High-level input voltage		0.75×VDD[1]	-	VDD[1]+0.3	V
V_{IL}	Low-level input voltage		–0.3	-	0.25×VDD[1]	V
I_{IH}	High-level input current		-	-	50	nA
I_{IL}	Low-level input current		-	-	50	nA
V_{OH}	High-level output voltage		0.8×VDD[1]	-	-	V
V_{OL}	Low-level output voltage		-	-	0.1×VDD[1]	V
I_{OH}	High-level source current (VDD[1] = 3.3 V, V_{OH} >= 2.64 V, output drive strength set to the maximum)	VDD3P3_CPU power domain [1, 2]	-	40	-	mA
		VDD3P3_RTC power domain [1, 2]	-	40	-	mA
		VDD_SDIO power domain [1, 3]	-	20	-	mA
I_{OL}	Low-level sink current (VDD[1] = 3.3 V, V_{OL} = 0.495 V, output drive strength set to the maximum)		-	28	-	mA
R_{PU}	Pull-up resistor		-	45	-	kΩ
R_{PD}	Pull-down resistor		-	45	-	kΩ
V_{IL_nRST}	Low-level input voltage of CHIP_PU to power off the chip		-	-	0.6	V

Notes:

1. Please see Table IO_MUX for IO's power domain. VDD is the I/O voltage for a particular power domain of pins.
2. For VDD3P3_CPU and VDD3P3_RTC power domain, per-pin current sourced in the same domain is gradually reduced from around 40 mA to around 29 mA, V_{OH}>=2.64 V, as the number of current-source pins increases.
3. For VDD_SDIO power domain, per-pin current sourced in the same domain is gradually reduced from around 30 mA to around 10 mA, V_{OH}>=2.64 V, as the number of current-source pins increases.

5.4 Reliability Qualifications

Table 14: Reliability Qualifications

Reliability tests	Standards	Test conditions	Result
Electro-Static Discharge Sensitivity (ESD), Charge Device Mode (CDM) [1]	JEDEC EIA/JESD22-C101	±500 V, all pins	Pass
Electro-Static Discharge Sensitivity (ESD), Human Body Mode (HBM) [2]	JEDEC EIA/JESD22-A114	±1500 V, all pins	Pass
Latch-up (Over-current test)	JEDEC STANDARD NO.78	±50 mA ~ ±200 mA, room temperature, test for IO	Pass
Latch-up (Over-voltage test)	JEDEC STANDARD NO.78	1.5 × Vmax, room temperature, test for V_{supply}	Pass
Moisture Sensitivity Level (MSL)	J-STD-020, MSL 3	30 °C, 60% RH, 192 hours, IR × 3 @260 °C	Pass

1. JEDEC document JEP157 states that 250 V CDM allows safe manufacturing with a standard ESD control process.
2. JEDEC document JEP155 states that 500 V HBM allows safe manufacturing with a standard ESD control process.

5.5 RF Power-Consumption Specifications

The power consumption measurements are taken with a 3.3 V supply at 25 °C of ambient temperature at the RF port. All transmitters' measurements are based on a 50% duty cycle.

Table 15: RF Power-Consumption Specifications

Mode	Min	Typ	Max	Unit
Transmit 802.11b, DSSS 1 Mbps, POUT = +19.5 dBm	-	240	-	mA
Transmit 802.11b, OFDM 54 Mbps, POUT = +16 dBm	-	190	-	mA
Transmit 802.11g, OFDM MCS7, POUT = +14 dBm	-	180	-	mA
Receive 802.11b/g/n	-	95 ~ 100	-	mA
Transmit BT/BLE, POUT = 0 dBm	-	130	-	mA
Receive BT/BLE	-	95 ~ 100	-	mA

5.6 Wi-Fi Radio

Table 16: Wi-Fi Radio Characteristics

Description	Min	Typical	Max	Unit
Input frequency	2412	-	2484	MHz
Output impedance*	-	*	-	Ω
Tx power				
Output power of PA for 72.2 Mbps	13	14	15	dBm
Output power of PA for 11b mode	19.5	20	20.5	dBm
Sensitivity				
DSSS, 1 Mbps	-	–98	-	dBm
CCK, 11 Mbps	-	–91	-	dBm
OFDM, 6 Mbps	-	–93	-	dBm
OFDM, 54 Mbps	-	–75	-	dBm
HT20, MCS0	-	–93	-	dBm
HT20, MCS7	-	–73	-	dBm
HT40, MCS0	-	–90	-	dBm
HT40, MCS7	-	–70	-	dBm
MCS32	-	–89	-	dBm
Adjacent channel rejection				
OFDM, 6 Mbps	-	37	-	dB
OFDM, 54 Mbps	-	21	-	dB
HT20, MCS0	-	37	-	dB
HT20, MCS7	-	20	-	dB

*The typical value of ESP32's Wi-Fi radio output impedance is different in chips of different QFN packages. For ESP32 chips with a QFN 6x6 package (ESP32-D0WDQ6), the value is 30+j10 Ω. For ESP32 chips with a QFN 5x5 package (ESP32-D0WD, ESP32-D2WD, ESP32-S0WD), the value is 35+j10 Ω.

5.7 Bluetooth Radio

5.7.1 Receiver – Basic Data Rate

Table 17: Receiver Characteristics – Basic Data Rate

Parameter	Conditions	Min	Typ	Max	Unit
Sensitivity @0.1% BER	-	-	–94	-	dBm
Maximum received signal @0.1% BER	-	0	-	-	dBm
Co-channel C/I	-	-	+7	-	dB
Adjacent channel selectivity C/I	F = F0 + 1 MHz	-	-	–6	dB
	F = F0 – 1 MHz	-	-	–6	dB
	F = F0 + 2 MHz	-	-	–25	dB
	F = F0 – 2 MHz	-	-	–33	dB
	F = F0 + 3 MHz	-	-	–25	dB
	F = F0 – 3 MHz	-	-	–45	dB
Out-of-band blocking performance	30 MHz ~ 2000 MHz	–10	-	-	dBm
	2000 MHz ~ 2400 MHz	–27	-	-	dBm
	2500 MHz ~ 3000 MHz	–27	-	-	dBm
	3000 MHz ~ 12.5 GHz	–10	-	-	dBm
Intermodulation	-	–36	-	-	dBm

5.7.2 Transmitter – Basic Data Rate

Table 18: Transmitter Characteristics – Basic Data Rate

Parameter	Conditions	Min	Typ	Max	Unit
RF transmit power	-	-	0	-	dBm
Gain control step	-	-	3	-	dBm
RF power control range	-	–12	-	+9	dBm
+20 dB bandwidth	-	-	0.9	-	MHz
Adjacent channel transmit power	F = F0 ± 2 MHz	-	–47	-	dBm
	F = F0 ± 3 MHz	-	–55	-	dBm
	F = F0 ± > 3 MHz	-	–60	-	dBm
$\Delta f1_{avg}$	-	-	-	155	kHz
$\Delta f2_{max}$	-	133.7	-	-	kHz
$\Delta f2_{avg}/\Delta f1_{avg}$	-	-	0.92	-	-
ICFT	-	-	–7	-	kHz
Drift rate	-	-	0.7	-	kHz/50 μs
Drift (DH1)	-	-	6	-	kHz
Drift (DH5)	-	-	6	-	kHz

5.7.3 Receiver – Enhanced Data Rate

Table 19: Receiver Characteristics – Enhanced Data Rate

Parameter	Conditions	Min	Typ	Max	Unit
	π/4 DQPSK				
Sensitivity @0.01% BER	-	-	–90	-	dBm
Maximum received signal @0.01% BER	-	-	0	-	dBm
Co-channel C/I	-	-	11	-	dB
Adjacent channel selectivity C/I	F = F0 + 1 MHz	-	–7	-	dB
	F = F0 – 1 MHz	-	–7	-	dB
	F = F0 + 2 MHz	-	–25	-	dB
	F = F0 – 2 MHz	-	–35	-	dB
	F = F0 + 3 MHz	-	–25	-	dB
	F = F0 – 3 MHz	-	–45	-	dB
	8DPSK				
Sensitivity @0.01% BER	-	-	–84	-	dBm
Maximum received signal @0.01% BER	-	-	–5	-	dBm
C/I c-channel	-	-	18	-	dB
Adjacent channel selectivity C/I	F = F0 + 1 MHz	-	2	-	dB
	F = F0 – 1 MHz	-	2	-	dB
	F = F0 + 2 MHz	-	–25	-	dB
	F = F0 – 2 MHz	-	–25	-	dB
	F = F0 + 3 MHz	-	–25	-	dB
	F = F0 – 3 MHz	-	–38	-	dB

5.7.4 Transmitter – Enhanced Data Rate

Table 20: Transmitter Characteristics – Enhanced Data Rate

Parameter	Conditions	Min	Typ	Max	Unit
RF transmit power	-	-	0	-	dBm
Gain control step	-	-	3	-	dBm
RF power control range	-	–12	-	+9	dBm
π/4 DQPSK max w0	-	-	–0.72	-	kHz
π/4 DQPSK max wi	-	-	–6	-	kHz
π/4 DQPSK max |wi + w0|	-	-	–7.42	-	kHz
8DPSK max w0	-	-	0.7	-	kHz
8DPSK max wi	-	-	–9.6	-	kHz
8DPSK max |wi + w0|	-	-	–10	-	kHz
π/4 DQPSK modulation accuracy	RMS DEVM	-	4.28	-	%
	99% DEVM	-	100	-	%
	Peak DEVM	-	13.3	-	%
8 DPSK modulation accuracy	RMS DEVM	-	5.8	-	%
	99% DEVM	-	100	-	%
	Peak DEVM	-	14	-	%
In-band spurious emissions	F = F0 ± 1 MHz	-	–46	-	dBm

Parameter	Conditions	Min	Typ	Max	Unit
	F = F0 ± 2 MHz	-	–40	-	dBm
	F = F0 ± 3 MHz	-	–46	-	dBm
	F = F0 +/– > 3 MHz	-	-	–53	dBm
EDR differential phase coding	-	-	100	-	%

5.8 Bluetooth LE Radio

5.8.1 Receiver

Table 21: Receiver Characteristics – BLE

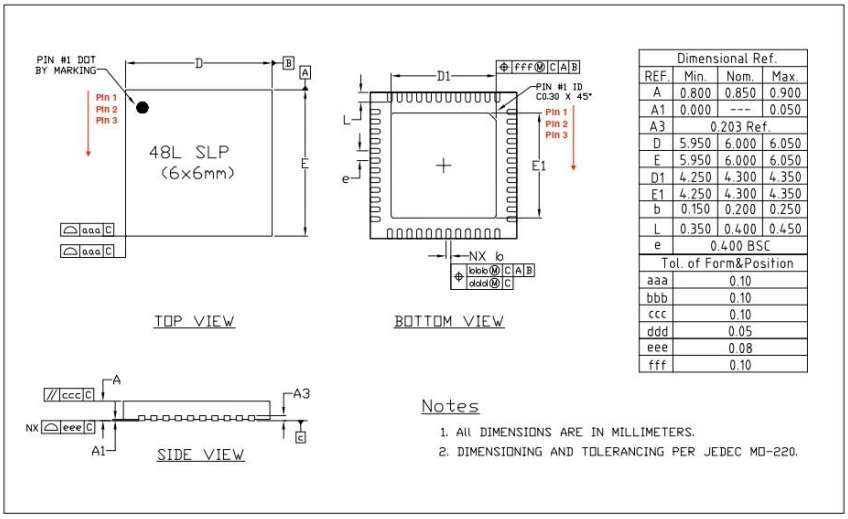
Parameter	Conditions	Min	Typ	Max	Unit
Sensitivity @30.8% PER	-	-	–97	-	dBm
Maximum received signal @30.8% PER	-	0	-	-	dBm
Co-channel C/I	-	-	+10	-	dB
Adjacent channel selectivity C/I	F = F0 + 1 MHz	-	–5	-	dB
	F = F0 – 1 MHz	-	–5	-	dB
	F = F0 + 2 MHz	-	–25	-	dB
	F = F0 – 2 MHz	-	–35	-	dB
	F = F0 + 3 MHz	-	–25	-	dB
	F = F0 – 3 MHz	-	–45	-	dB
Out-of-band blocking performance	30 MHz ~ 2000 MHz	–10	-	-	dBm
	2000 MHz ~ 2400 MHz	–27	-	-	dBm
	2500 MHz ~ 3000 MHz	–27	-	-	dBm
	3000 MHz ~ 12.5 GHz	–10	-	-	dBm
Intermodulation	-	–36	-	-	dBm

5.8.2 Transmitter

Table 22: Transmitter Characteristics – BLE

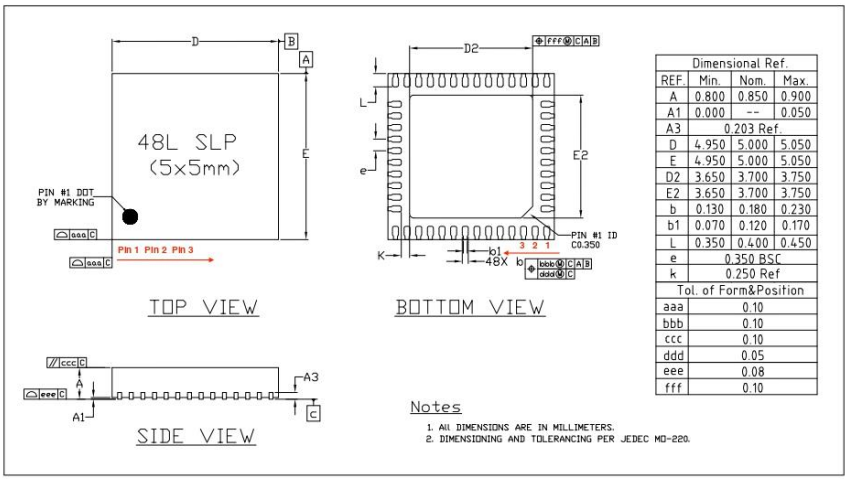
Parameter	Conditions	Min	Typ	Max	Unit
RF transmit power	-	-	0	-	dBm
Gain control step	-	-	3	-	dBm
RF power control range	-	–12	-	+9	dBm
Adjacent channel transmit power	F = F0 ± 2 MHz	-	–52	-	dBm
	F = F0 ± 3 MHz	-	–58	-	dBm
	F = F0 ± > 3 MHz	-	–60	-	dBm
$\Delta f1_{avg}$	-	-	-	265	kHz
$\Delta f2_{max}$	-	247	-	-	kHz
$\Delta f2_{avg}/\Delta f1_{avg}$	-	-	–0.92	-	-
ICFT	-	-	–10	-	kHz
Drift rate	-	-	0.7	-	kHz/50 μs
Drift	-	-	2	-	kHz

6. Package Information



Dimensional Ref.			
REF.	Min.	Nom.	Max.
A	0.800	0.850	0.900
A1	0.000	---	0.050
A3	0.203 Ref.		
D	5.950	6.000	6.050
E	5.950	6.000	6.050
D1	4.250	4.300	4.350
E1	4.250	4.300	4.350
b	0.150	0.200	0.250
L	0.350	0.400	0.450
e	0.400 BSC		
Tol. of Form&Position			
aaa	0.10		
bbb	0.10		
ccc	0.10		
ddd	0.05		
eee	0.08		
fff	0.10		

Figure 8: QFN48 (6x6 mm) Package



Dimensional Ref.			
REF.	Min.	Nom.	Max.
A	0.800	0.850	0.900
A1	0.000	--	0.050
A3	0.203 Ref.		
D	4.950	5.000	5.050
E	4.950	5.000	5.050
D2	3.650	3.700	3.750
E2	3.650	3.700	3.750
b	0.130	0.180	0.230
b1	0.070	0.120	0.170
L	0.350	0.400	0.450
e	0.350 BSC		
k	0.250 Ref		
Tol. of Form&Position			
aaa	0.10		
bbb	0.10		
ccc	0.10		
ddd	0.05		
eee	0.08		
fff	0.10		

Figure 9: QFN48 (5x5 mm) Package

Note:
The pins of the chip are numbered in an anti-clockwise direction from Pin 1 in the top view.