



MAESTRÍA EN SISTEMAS EMBEBIDOS

MEMORIA DEL TRABAJO FINAL

Cámara IoT para detección facial con conectividad Wi-Fi

Autor:

Esp. Ing. Mauricio Barroso Benavides

Director:

Mg. Ing. Gonzalo Sanchez (FF.AA, FIUBA)

Jurados:

Mg. Ing. Edgardo Torrelli (FIUBA)

Mg. Lic. Leopoldo Zimperz (FIUBA)

Mg. Ing. Sebastián Guarino (FIUBA)

*Este trabajo fue realizado en la ciudad de Tupiza,
entre junio de 2021 y diciembre de 2022.*

Resumen

Esta memoria describe el proceso de desarrollo de un dispositivo electrónico compuesto principalmente por un módulo de procesamiento con conectividad Wi-Fi y una cámara, que puede capturar imágenes para procesarlas mediante algoritmos de Inteligencia Artificial y así detectar rostros humanos. Los datos generados por el dispositivo son transmitidos hacia servidores en la nube encargados de procesar, almacenar y facilitar su visualización para los usuarios finales. La principal aplicación de este trabajo es generar información sobre la presencia de personas para, por ejemplo, activar otros dispositivos como bocinas o mecanismos de cierre/apertura de puertas.

En la realización del presente trabajo se utilizaron conocimientos adquiridos a lo largo de la carrera como desarrollo de firmware, visión artificial, diseño de hardware, sistemas distribuidos, gestión de proyectos y gestión de tecnología.

Agradecimientos

A Gonzalo Sanchez, director de este trabajo, por sus valiosos consejos y criterios que se ven reflejados a lo largo de este desarrollo.

A los profesores de la Maestria en Sistemas embebidos, por contribuir en mi formacion academica con sus conocimientos y experiencias.

Índice general

Resumen	I
1. Introducción general	3
1.1. Inteligencia Artificial, Aprendizaje Automático y Aprendizaje Profundo	3
1.2. Redes neuronales convolucionales	6
1.2.1. Capa de convoluciones	7
1.2.2. Capa de <i>pooling</i>	7
1.2.3. Capa <i>fully-connected</i>	8
1.3. Vision artificial	8
1.4. Servicios en la nube	9
1.5. Motivación	10
1.6. Estado del arte	11
1.7. Objetivos y alcance	11
1.8. Requerimientos	12
2. Introducción específica	13
2.1. Diagrama general del sistema	13
2.1.1. Placa de desarrollo ESP32-S3-DevKitC-1	13
2.1.2. Sensor de movimiento PIR	15
Sensor IRA-S230ST01	15
Amplificador operacional TVL8544	16
2.1.3. Cámara OV2640	17
2.2. MTCNN (<i>Multi-Task Cascaded Convolutional Networks</i> , Redes Convolucionales en Cascada Multitarea)	18
2.2.1. P-Net	19
2.2.2. R-Net	20
2.2.3. O-Net	20
2.2.4. Tareas de MTCNN	20
2.3. TensorFlow	21
2.4. AWS (<i>Amazon Web Services</i> , Servicios Web de Amazon)	22
2.4.1. IoT Core	22
2.4.2. Amazon Timestream	23
2.5. Grafana	23
3. Diseño e implementación	25
3.1. Análisis del software	25
4. Ensayos y resultados	27
4.1. Pruebas funcionales del hardware	27
5. Conclusiones	29
5.1. Conclusiones generales	29

5.2. Próximos pasos	29
-------------------------------	----

Índice de figuras

1.1.	Diferencias entre AI, ML y DL	3
1.2.	Aprendizaje supervisado.	4
1.3.	Aprendizaje no supervisado.	5
1.4.	Aprendizaje reforzado.	5
1.5.	Arquitectura de una red neuronal artificial.	6
1.6.	Nodo de una red neuronal artificial.	6
1.7.	CNN para clasificar dígitos escritos a mano.	7
1.8.	Convolución de una entrada de 3x3 con un <i>kernel</i> de 2x2 y <i>stride</i> de 1.	7
1.9.	Tipos de <i>pooling</i>	8
1.10.	Componentes de un sistema de visión artificial y un sistema de visión humana.	8
1.11.	Imagen procesada por un sistema de detección facial.	9
1.12.	Capacidades de SaaS, IaaS y PaaS ¹	10
1.13.	Diagrama en bloques del sistema propuesto por	11
2.1.	Diagrama en bloques del sistema.	13
2.2.	Componentes del ESP32-S3-DevKitC-1.	14
2.3.	Diagrama en bloques del sensor de movimiento PIR.	15
2.4.	Fotografia del sensor IRA-S230ST01.	15
2.5.	Fotografia del TLV8544 en un encapsulado TSSOP-14.	16
2.6.	Diagrama en bloques del modulo ESP-LyraP-CAM.	17
2.7.	<i>Pipeline de MTCNN</i>	19
2.8.	<i>Arquitectura de P-Net</i>	19
2.9.	<i>Arquitectura de R-Net</i>	20
2.10.	<i>Arquitectura de O-Net</i>	20
2.11.	Diagrama de conexion entre dispositivos IoT y AWS.	22

Índice de tablas

2.1. ESP32-S3-DevKitC-1 especificaciones	14
2.2. IRA-S230ST01 especificaciones	16
2.3. TLV8544 especificaciones	17
2.4. OV2640 especificaciones	18

Este trabajo se lo dedico a mi familia, eternas gracias por su apoyo incondicional en cada etapa de mi vida. Ustedes son la luz que guia mi camino

-

Capítulo 1

Introducción general

En este capítulo se presentan conceptos básicos sobre las tecnologías y técnicas que fueron utilizadas en el desarrollo del trabajo. Se abordan nociones sobre inteligencia artificial, aprendizaje automático, aprendizaje profundo, redes neuronales convolucionales, visión artificial y servicios en la nube. También se citan trabajos anteriores que inspiraron a este, las motivaciones para llevarlo a cabo junto a sus objetivos y alcances.

1.1. Inteligencia Artificial, Aprendizaje Automático y Aprendizaje Profundo

AI (*Artificial Intelligence*, Inteligencia Artificial), ML (*Machine Learning*, Aprendizaje Automático) y DL (*Deep Learning*, Aprendizaje Profundo), son términos muy utilizados hoy en día en el mundo del desarrollo tecnológico [ref]. Aunque estos términos son muy parecidos, entre ellos existen dependencias que pueden ser visualizadas con ayuda de la figura 1.1.

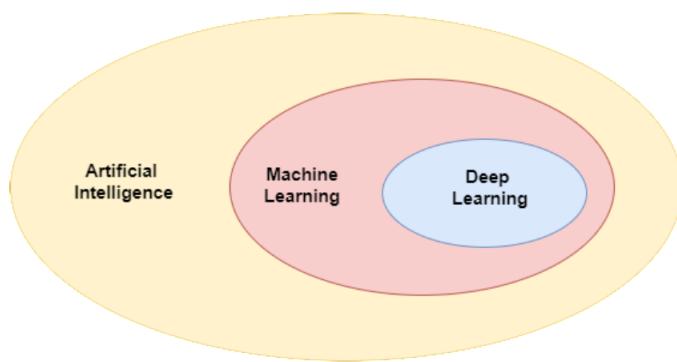


FIGURA 1.1. Diferencias entre AI, ML y DL.

AI es un área de la computación que permite a los sistemas computacionales imitar la inteligencia humana para entender su entorno y tomar acciones que maximicen sus posibilidades de lograr sus objetivos [ref]. Sus aplicaciones más importante se encuentran en la áreas de comercio, educación, robótica, salud, agricultura, automotriz y finanzas[<https://www.simplilearn.com/tutorials/artificial-intelligence-tutorial/artificial-intelligence-applications>]. Todos los sistemas de inteligencia artificial reales e hipotéticos pueden ser clasificados en alguno de los siguientes tipos:

- ANI (*Artificial Narrow Intelligence, Inteligencia Artificial Estrecha*): también conocida como inteligencia artificial débil, su objetivo es llevar a cabo un solo tipo de tarea. Estos sistemas no poseen conciencia y no son manejados por sentimientos como lo haría un humano. Algunos ejemplos de ANI son los *chatbots* o los automóviles autónomos.
- AGI (*Artificial General Intelligence, Inteligencia Artificial General*): también conocida como inteligencia artificial fuerte, es un concepto en el que las máquinas exhiben inteligencia humana. Estos sistemas tendrían la capacidad de aprender, entender y actuar de tal manera que sería indistinguible a un humano. AIG actualmente no existe, pero es utilizado en industrias como el cine
- ASI (*Artificial Super Intelligence, Super Inteligencia Artificial*): ASI también forma parte de la inteligencia artificial fuerte. Se le considera muy poderosa por ser capaz de volverse consciente y autónoma. No sólo replica el comportamiento humano, sino que lo supera. Puede pensar mejor y tener más habilidades. Sin embargo, esta tecnología aún está en desarrollo.

ML es un subconjunto de AI que utiliza algoritmos de aprendizaje estadísticos para construir sistemas con la habilidad de aprender automáticamente y mejorar a partir de experiencias previas sin ser explícitamente programados para esto. Muchos de los servicios de recomendación utilizados por empresas como Netflix, YouTube o Spotify, utilizan ML para adaptarse a un usuario en particular y ofrecer una mejor experiencia más personalizada. Estos algoritmos pueden ser clasificados de la siguiente manera:

- Aprendizaje supervisado: se refiere al aprendizaje modelando a partir de un conjunto de datos, mejor conocidos como *dataset*, cuyas respuestas son conocidas con antelación y están asociadas a una etiqueta o *label*. Por ejemplo, el *dataset* pueden ser muchas fotografías de gatos y el *label* asociado el nombre de este animal. De esta manera el modelo es entrenado para generar predicciones de datos nuevos. En la figura 1.2 se representa gráficamente el aprendizaje supervisado.

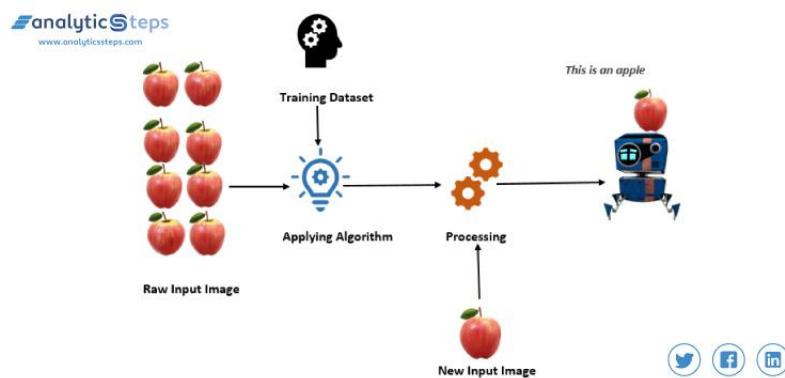


FIGURA 1.2. Aprendizaje supervisado.

- Aprendizaje no supervisado: es utilizado cuando los datos utilizados para el aprendizaje no tienen *labels*. Su objetivo principal es aprender acerca de los datos e inferir patrones sin ningún tipo de referencia sobre las respuestas esperadas. Es mayormente utilizado como parte del análisis exploratorio

de datos [<https://towardsdatascience.com/understanding-the-difference-between-ai-ml-and-dl-cceb63252a6c>]. En la figura 1.3 se representa gráficamente el aprendizaje no supervisado.

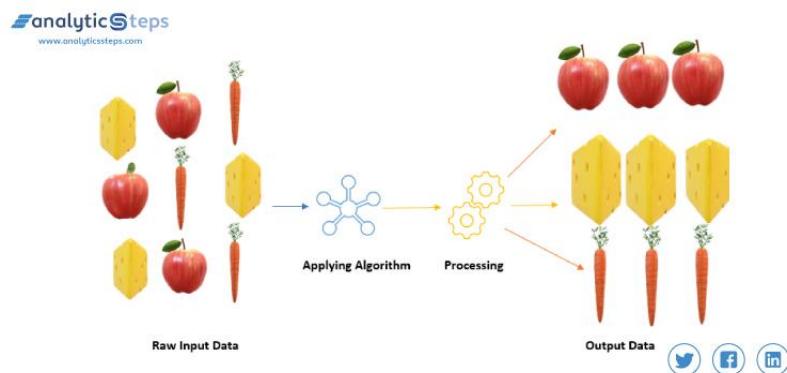


FIGURA 1.3. Aprendizaje no supervisado.

- Aprendizaje reforzado: es el aprendizaje mediante la interacción continua con el entorno con el método de prueba y error, y utiliza continuamente la retroalimentación de sus acciones y experiencias previas. Este tipo de aprendizaje utiliza recompensas si se realizan acciones correctas y penalizaciones si son incorrectas. En la figura 1.4 se representa gráficamente el aprendizaje reforzado.

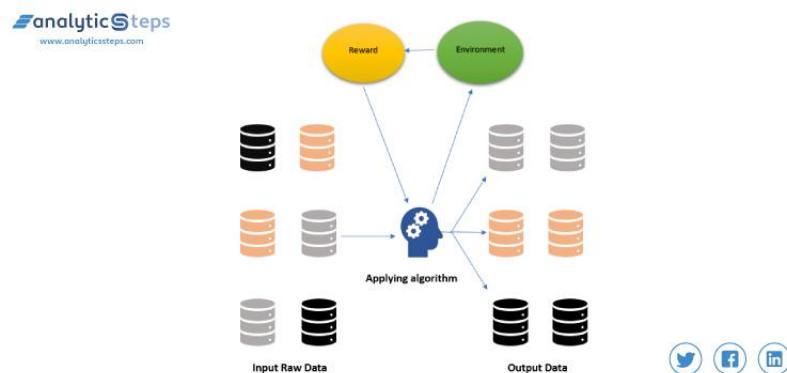


FIGURA 1.4. Aprendizaje reforzado.

DL es una técnica de ML que está inspirada en la forma en la que el cerebro humano filtra información. Cómo DL procesa información de manera similar al cerebro humano sus aplicaciones son tareas que un humano generalmente realiza, como distinguir entre un peatón o un poste de luz en el caso de automóviles autónomos. El componente principal de DL son las redes neuronales artificiales, que son capas de nodos interconectados, donde existen una capa de entrada, una o varias capas ocultas y una capa de salida. En la figura 1.5 se puede observar la arquitectura de una red neuronal artificial.

Cada uno de los nodos de las capas ocultas y de salida, tienen como entrada la salida de los nodos anteriores multiplicadas por unos términos denominados pesos o *weights* y que sumados junto a otro término llamado sesgo o *bias* pasan por una función de activación no lineal para generar su salida. En la figura 1.6 se visualiza un nodo de las capas ocultas o de salida.

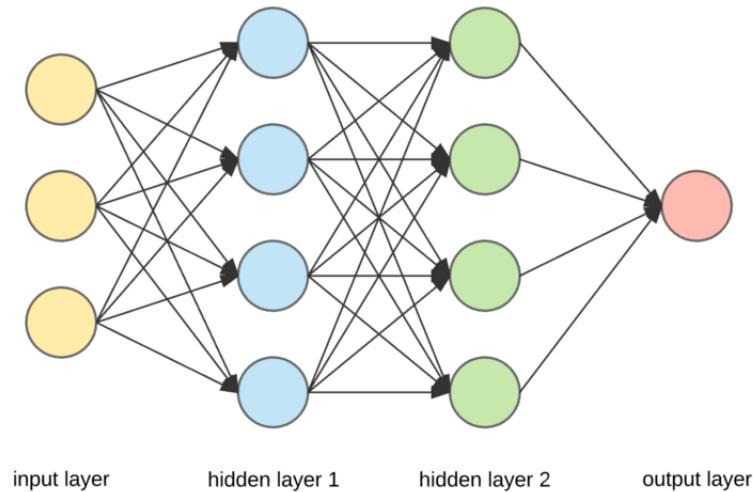


FIGURA 1.5. Arquitectura de una red neuronal artificial.

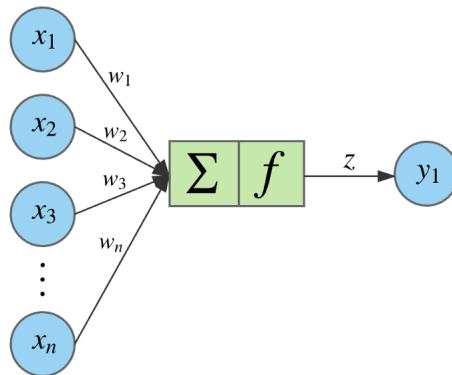


FIGURA 1.6. Nodo de una red neuronal artificial.

1.2. Redes neuronales convolucionales

También conocidas como CNN (*Convolutional Neural Networks*, Redes Neuronales Convolucionales) o ConvNet, son un algoritmo de DL que están orientadas a recibir como entrada una, asignarle *weights* y *biases* entrenables a varios aspectos/objetos en la imagen para poder diferenciarlas unas de otras. Su uso reduce el pre procesamiento de las imágenes de entrada con respecto a otros modelos de clasificación, ya que los filtros necesarios son incorporados en su arquitectura y tienen la habilidad de ser entrenados.

Computacionalmente una imagen puede ser muy difícil de procesar, esto depende del espacio de colores donde se encuentra [ref] y las dimensiones que posee. Por ejemplo una imagen RGB (*Red Green Blue*, Rojo Verde Azul) y de dimensiones 1920x1080 pixeles, tiene un tamaño de 6220800 bytes. El objetivo principal de las CNN es reducir la dimensionalidad de las imágenes de entrada, de tal forma que sean más fáciles de procesar y no pierdan sus características o *features* principales que son críticas para obtener una buena predicción.

La arquitectura de una CNN es independiente del tipo de aplicación, donde las capas que lo componen son elegidas en función de los objetivos que se persiguen. En la figura 1.7 se puede observar la arquitectura de una CNN para clasificar dígitos escritos a mano.

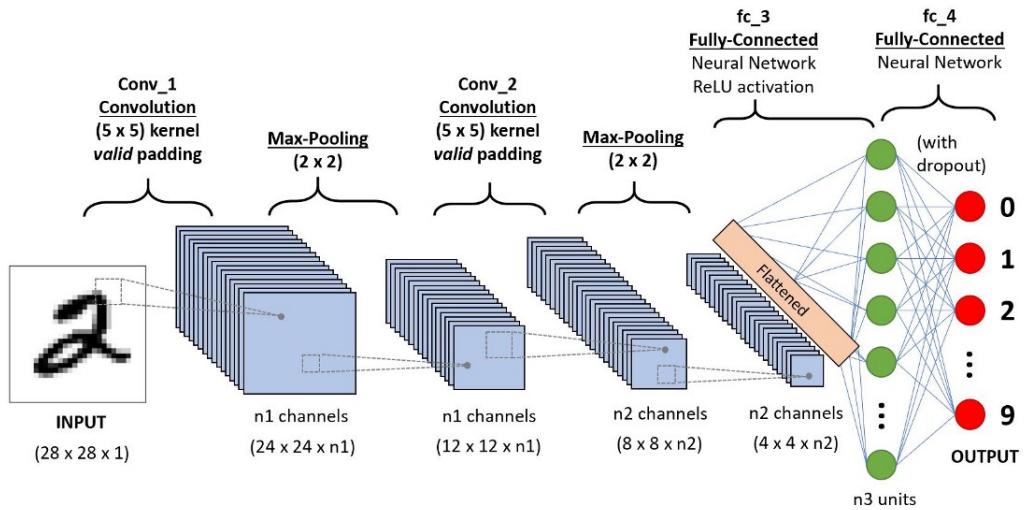


FIGURA 1.7. CNN para clasificar dígitos escritos a mano.

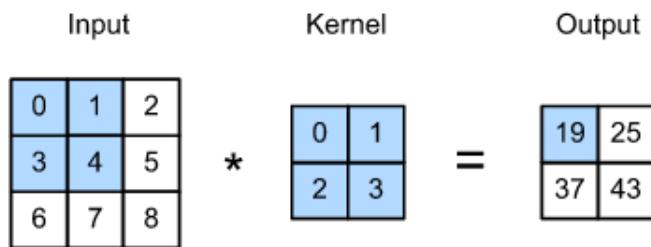
En la arquitectura de la figura 1.7 se pueden observar tres capas principales para construir una CNN: capa de convoluciones, capa de *pooling* y capa *fully-connected*.

1.2.1. Capa de convoluciones

Esta capa es la encargada de aplicar la operación de convolución sobre las imágenes de entrada para encontrar patrones que más adelante permitirán clasificarlas. La convolución de una imagen con un *kernel* no es más que la aplicación del operador punto entre ambos. Este tipo de capas se definen por:

- El número de los *kernels* o filtros que se aplican a la imagen, que es el número de matrices por las que se van a convolucionar las imágenes de entrada.
- El tamaño de los *kernels*, donde casi siempre tienen dimensiones cuadradas e impares como 3x3 o 5x5.
- El *stride* o paso, se refiere a la forma en como el *kernel* recorre la imagen.

En la figura 1.8 se puede observar como un

FIGURA 1.8. Convolución de una entrada de 3x3 con un *kernel* de 2x2 y *stride* de 1.

1.2.2. Capa de *pooling*

Similar a la capa de convoluciones, tiene el objetivo de reducir la dimensionalidad de los *features* obtenidos mediante las convoluciones aplicadas en la capa

anterior, para reducir el poder computacional requerido en un principio. Existen dos tipos de dos tipos: *max pooling* y *Average pooling*. El primero retorna el valor máximo de una porción de la imagen cubierta por el *kernel* y el segundo el valor promedio o *average*. *Max pooling* también funciona como supresor de ruido al mismo tiempo que reduce la dimensionalidad. Mientras que *average pooling* solo sirve para reducir la dimensionalidad. En la figura 1.9 se pueden observar estos tipos de *pooling*.

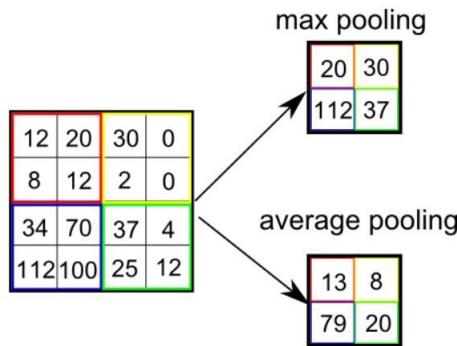


FIGURA 1.9. Tipos de *pooling*.

1.2.3. Capa *fully-connected*

También conocida como capa lineal o FC (*Fully Connected*, Totalmente Conectada), es simplemente una red neuronal artificial como la mostrada en la sección anterior y se utiliza después de que las capas de convolución y *pooling* desglosan los *features* más importantes presentes en la imagen de entrada la CNN. La capa FC brinda las probabilidades finales para cada *label* esperado.

1.3. Vision artificial

La visión artificial o *computer vision* es un campo científico interdisciplinario que se encarga de cómo los sistemas computacionales pueden obtener un entendimiento de alto nivel de imágenes y videos digitales, para comprender y automatizar tareas como lo haría un sistema de visión humana. Las tareas que ejecuta un sistema de visión artificial son de adquisición, procesamiento, análisis y entendimiento de imágenes. En la figura 1.10 se pueden apreciar los componentes de un sistema de visión artificial y un sistema de visión humano.

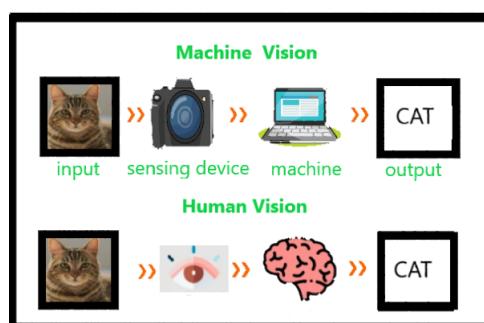


FIGURA 1.10. Componentes de un sistema de visión artificial y un sistema de visión humano.

Uno de los campos de estudio más importantes de la visión artificial es la detección facial. La detección facial puede ser considerada como un caso particular de la detección de objetos y tiene los objetivos de detectar y localizar todos los rostros humanos contenidos en una imagen digital. En la figura 1.11 se puede observar una imagen procesada por un sistema de detección facial.

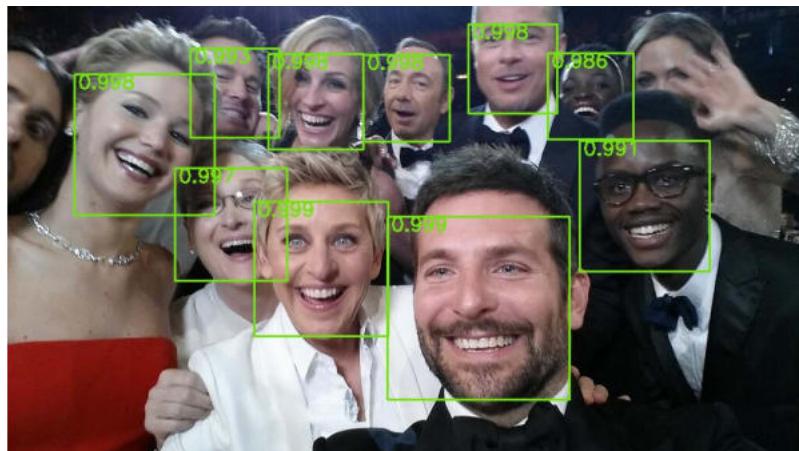


FIGURA 1.11. Imagen procesada por un sistema de detección facial.

Hoy en día, muchos dispositivos comerciales y profesionales como smartphones, tablets y robots, utilizan la detección facial como primer paso para otro tipo de aplicaciones más complejas, entre las que destacan: reconocimiento facial, computación afectiva y grabación de vídeo inteligente.

1.4. Servicios en la nube

El término servicios en la nube hace referencia a un amplio rango de servicios ofrecidos bajo demanda a compañías y usuarios a través de internet. Estos servicios están diseñados para proveer de una manera fácil y asequible acceso a aplicaciones y recursos, sin la necesidad de una infraestructura o hardware propios.

Los servicios en la nube son administrados totalmente por proveedores de computación en la nube [ref]. Estos se encuentran disponibles para los usuarios desde los servidores de los proveedores, por lo que no es necesario que una empresa aloje aplicaciones en sus propios servidores.

De manera general, existen tres tipos básicos de servicios en la nube [ref]:

- SaaS (*Software as a Service*, Software como un Servicio): en este servicio el proveedor sólo proporciona el software o aplicaciones en la nube mediante internet. Los clientes tienen acceso a través de APIs (*Application Programming Interface*, Interfaz de Programación de Aplicaciones) o a través de la web, que les permite interactuar de manera sencilla, sin la necesidad de gestionar, instalar ni actualizar el software.
- IaaS (*Infrastructure as a Service*, Infraestructura como un Servicio): este servicio implica la contratación de una infraestructura de hardware a un tercero, donde varios cliente comparten los recursos de una máquina física. El proveedor proporciona a sus clientes el acceso a los recursos computacionales

necesarios para almacenar o ejecutar tareas que pueden incluir servidores, redes, *backup, firewalls*, entre otros.

- PaaS (*Platform as a Service*, Plataforma como un Servicio): es un servicio que se encuentra conceptualmente entre SaaS e IaaS al eliminar la parte física de la infraestructura y ofrece una plataforma donde los cliente pueden crear, desarrollar, gestionar y distribuir sus aplicaciones. El proveedor es el encargado de la gestión y mantenimiento de la plataforma y permite que los clientes se dediquen exclusivamente al desarrollo.

En la figura 1.12 se puede observar las diferencias entre estos servicios en función de los elementos que pueden gestionar.

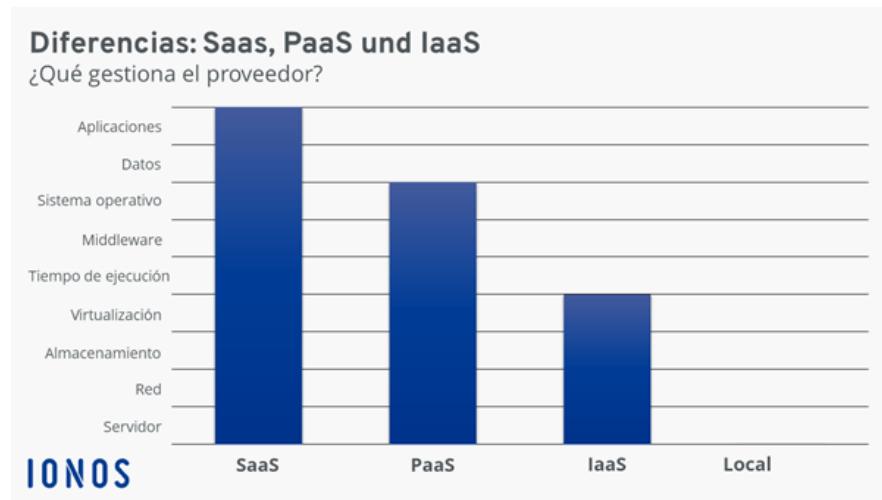


FIGURA 1.12. Capacidades de SaaS, IaaS y PaaS¹.

1.5. Motivación

Gracias a la amplia gama de plataformas de hardware y la disponibilidad de bibliotecas de código abierto para implementar AI, ML y DL, además de la difusión de información en foros y sitios web especializados, es posible desarrollar sistemas de visión artificial personalizados para distintos tipos de arquitecturas [nwengineeringllc.com/article/computer-vision-in-embedded-systems-and-ai-platforms.php].

Estas bibliotecas de código para implementar visión artificial no suelen ser aptas para cualquier dispositivo, ya que son, en la mayoría de los casos, muy grandes en tamaño y requieren de una capacidad de procesamiento lo suficientemente grande para ejecutarse en tiempo real. Pero los constantes esfuerzos de las empresas para ofrecer *frameworks* optimizados que reducen el tamaño y poder de procesamiento necesarios para ejecutar estos algoritmos, la integración de aceleradores de hardware para redes neuronales y DSP (*Digital Signal Processor, Procesador Digital de Señales*) en SoCs actuales (*System on a Chip, Sistema en un Chip*) y el estudio de nuevas y mejoradas arquitecturas para visión artificial basadas en

¹Imagen tomada de: <https://expansionba.com.ar/2020/05/23/medidas-para-amortiguar-el-costo-energetico-en-pymes/>

CNN, permiten el desarrollo de dispositivos embebidos con la capacidad de ejecutar modelos de visión artificial. Algunas de sus aplicaciones más importantes son: industria 4.0, seguridad, vehículos autónomos y robótica.

La motivación principal de este trabajo fue desarrollar un sistema embebido de bajo costo económico, bajo consumo energético y de código abierto, que integre algoritmos de DL para visión artificial enfocado a la tarea de detección facial.

Una motivación adicional fue integrar otra tecnología actual como es IoT (*Internet of Things*, Internet de las Cosas), para trabajar en conjunto con los algoritmos de visión artificial. Así las aplicaciones que se pueden obtener son más versátiles a la hora de su implementación en entornos urbanos.

1.6. Estado del arte

Como antecedente existe un trabajo denominado ‘A New IoT Combined Face Detection of People by Using Computer Vision for Security Application’... El *paper* donde se describe su trabajo presenta el desarrollo de un dispositivo electrónico que tiene como componentes principales una Raspberry Pi 3, un sensor PIR (Passive Infra Red, Pasivo Infrarrojo) y una cámara. Su objetivo principal es detectar personas con ayuda del sensor de movimiento PIR, fotografiarlas y aplicar el algoritmo de detección facial Haar Cascade [ref], para posteriormente guardar una imagen del rostro detectado y visualizarla en un teléfono móvil con ayuda de la aplicación Telegram. En la figura 1.13 se puede observar el diagrama en bloques del sistema descrito anteriormente.

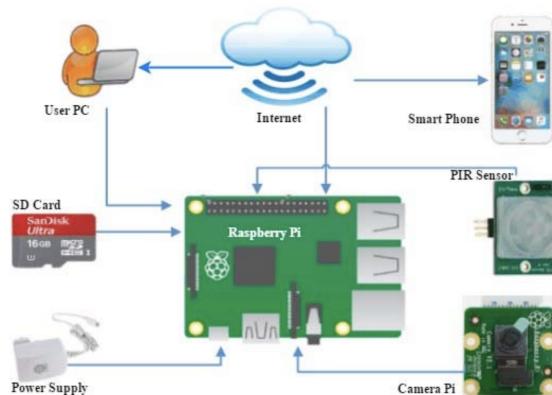


FIGURA 1.13. Diagrama en bloques del sistema propuesto por ...

1.7. Objetivos y alcance

El objetivo principal de este trabajo fue desarrollar un sistema embebido con la capacidad de ejecutar modelos de AI para detectar y localizar rostros humanos de imágenes digitales capturadas por su cámara.

El alcance de este trabajo incluyó:

- Construcción de un prototipo de pruebas
- Desarrollo e implementación de los modelos de AI necesarios
- Implementación de los servicios en la nube necesarios

1.8. Requerimientos

Los requerimientos planteados para este trabajo fueron:

1. Requerimientos funcionales

- a)* El sistema debe detectar y contar todos los rostros existentes de las imágenes obtenidas por su cámara con ayuda de las técnicas de procesamiento de imágenes pyramid image y sliding window, y modelos de DL que alcancen una precisión de al menos 80 %.
- b)* El sistema debe conectarse a una red Wi-Fi existente a través de algún mecanismo de aprovisionamiento de credenciales de red.
- c)* El sistema debe establecer comunicación con los servidores de AWS.
- d)* El sistema debe ser alimentado mediante dos baterías AA de litio.
- e)* El sistema debe poseer mecanismos de seguridad implementados tanto en hardware como en firmware para evitar su manipulación incorrecta.
- f)* El sistema debe funcionar solamente si se detecta movimiento en el sector donde se encuentra instalada.

2. Requerimientos no funcionales

- a)* El sistema debe tener un costo de desarrollo igual o menor a US\$200.
- b)* El sistema debe tener documentación adecuada sobre su uso y desarrollo.

Capítulo 2

Introducción específica

Este capítulo expone una descripción detallada del sistema, del hardware utilizado y las herramientas de software necesarias en el desarrollo del trabajo. Se abarcan la descripción del sistema y sus componentes, los *frameworks* y modelos utilizados para detección facial y las herramientas utilizadas en la web.

2.1. Diagrama general del sistema

El sistema desarrollado en este trabajo consta de varios componentes de hardware que interconectados entre sí son capaces de cubrir. En la figura 2.1 se muestra el diagrama en bloques del sistema.

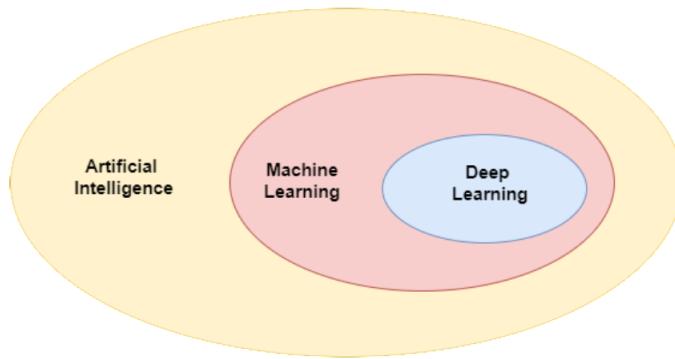


FIGURA 2.1. Diagrama en bloques del sistema.

explicar el funcionamiento...

2.1.1. Placa de desarrollo ESP32-S3-DevKitC-1

El componente central del sistema es la tarjeta de desarrollo ESP32-S3-DevKitC-1-N8R8 de la empresa Espressif. Tiene como componente central el módulo ESP32-S3-WROOM-1-N8R8 y varios otros componentes que simplifican el proceso de desarrollo de aplicaciones para IoT. En la figura 2.2 se observa una fotografía de la tarjeta con el detalle de sus componentes.

El módulo μ -WROOM-1-N8R8 es un potente módulo MCU (*Microcontroller Unit*, Unidad de Microcontrolador) de doble núcleo que incorpora Wi-Fi y BLE (*Bluetooth Low Energy*, Bluetooth de Baja Energía) y tiene un amplio conjunto de periféricos. Sus especificaciones técnicas más relevantes se detallan en la tabla 2.1.

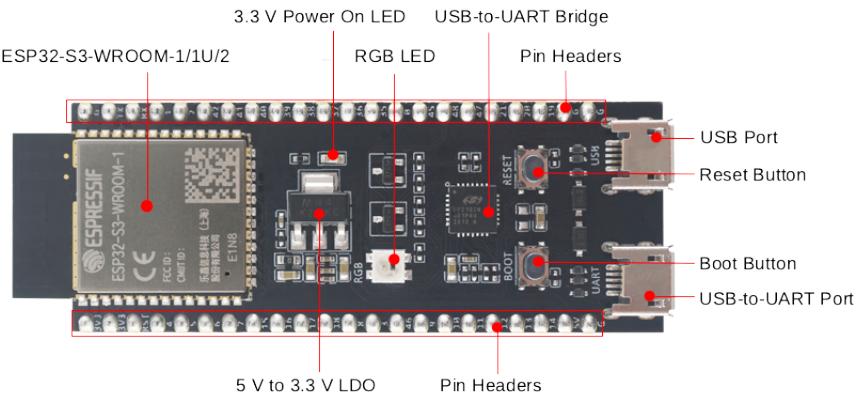


FIGURA 2.2. Componentes del ESP32-S3-DevKitC-1.

TABLA 2.1. Tabla de especificaciones del ESP32-S3-DevKitC-1

Característica	Descripción
SoC embebido	ESP32-S3R8
Procesador	Xtensa LX7 doble nucleo de 32 bits
Frecuencia	Hasta 240 MHz
ROM	384 KB
SRAM	512 KB
Pines	41
Flash	8 MB
PSRAM	8 MB
Tipo de antena	PCB
Wi-Fi	802.11 b/g/n hasta 150 Mbps
Bluetooth	Bluetooth 5 y Bluetooth mesh
Perifericos	GPIO, I2C, SPI, interfaz LCD, interfaz de camara, UART, I2S, USB, PWM, ADC, sensor tactil, sensor de temperatura, timer y watchdogs
Rango de temeperatura	-40 °Ca 65 °C

En el mercado existen muchos fabricantes que ofrecen tarjetas de desarrollo de características técnicas que podrían haber sido utilizadas para el desarrollo de este trabajo. Sin ir muy lejos, Espressif, fabricante de la ESP32-S3-DevKitC-1-N8R8, tiene toda una familia de modulos y tarjetas muy similares entre si. La elección de esta tarjeta en particular responde a los siguientes criterios:

- Costo: Espressif ofrece en todos sus SoCs, modulos y tarjetas, un costo muy contenido por la gran cantidad de características ofrecidas.
- Redes neuronales: la serie de SoCs ESP32-S3 ofrece soporte para instrucciones vectoriales, que acelera las tareas de computación de redes neuronales. Esta fue la característica más importante al momento de la elección de esta tarjeta.
- Memoria: como el trabajo implicaba el uso de una cámara y por tanto el manejo de *buffers* de memoria de gran tamaño para manipular las imágenes

obtenidas, la cantidad de memoria externa que ofrecia esta tarjeta la hizo optima para la aplicacion.

2.1.2. Sensor de movimiento PIR

Un sensor de movimiento PIR basa su funcionamiento al detectar diferencias en la energia IR (*Infrared, Infrarojo*) en el campo de vision del sensor. Debido a que la senal de salida generada por el sensor es muy pequena, es necesario aplicar etapas de amplificacion y filtrado para elevar el nivel de tension de la senal de salida y al mismo tiempo filtrar el ruido que puede generar eventos falsos positivos. Esta salida analogica luego se debe convertir en una senal digital mediante la operacion de comparacion de ventanas y se puede utilizar, por ejemplo, como una interrupcion en un MCU. En la figura 2.3 se muestra el diagrama en bloques del sensor de movimiento PIR.

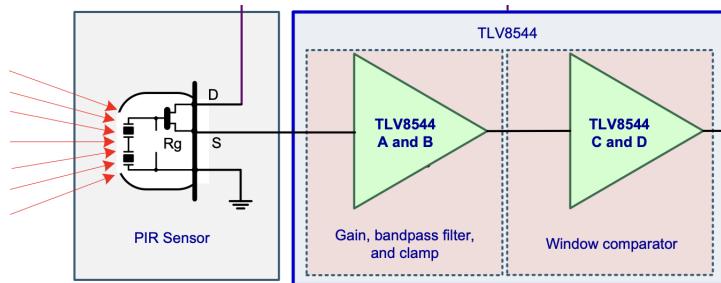


FIGURA 2.3. Diagrama en bloques del sensor de movimiento PIR.

Sensor IRA-S230ST01

El IRA-S230ST01 es un sensor PIR fabricado por la empresa Murata. Posee una alta sensibilidad y un rendimiento confiable gracias a la tecnologia ceramica y la tecnica de IC (*Integrated Circuit, Circuito Integrado*) hibrida de Murata. Tiene ademas una sensibilidad mejorada a la interferencia de RF (*Radio Frequency, Radiofrecuencia*). Sus aplicaciones mas comunes incluyen sistemas de seguridad, aparatos de iluminacion, electrodomesticos, entre otros. En la figura 2.4 se puede observar una fotografia del IRA-S230ST01.



FIGURA 2.4. Fotografia del sensor IRA-S230ST01.

En la tabla 2.2 se detallan sus caracteristicas tecnicas mas importantes.

La elección del IRA-S230ST01 como sensor PIR responde a los siguientes criterios:

TABLA 2.2. Tabla de especificaciones del IRA-S230ST01

Caracteristica	Descripcion
Rango de temperatura	-40 °C a 70 °C
SNR	40 dB
Campo de vision	theta1=theta2=45 grados
Electrodo	(2.0x1.0mm)x2
Responsividad	4.6 mV
Filtro optico	5 μm paso alto
Fuente de alimentacion	2 V a 15 V

- Marca: Murata es una marca muy reconocida en el mundo de los semiconductores y ofrece productos de muy alta calidad.
- Documentacion: el IRA-S230ST01 cuenta con documentación muy clara sobre sus características técnicas.

Amplificador operacional TTVL8544

El TLV8544 es un amplificador operacional cuadruple de ultra bajo consumo energético de la empresa Texas Instruments, de costo optimizado para aplicaciones de sensado en equipos inalámbricos y cableados de bajo consumo. El diseño del TLV8544 minimiza el consumo energético en dispositivos como sensores de movimiento para sistemas de seguridad, donde el tiempo de vida de la batería que los alimenta es crítico. Su uso más común es en configuraciones de amplificadores de transimpedancia con resistencias de *feedback* en el orden de los Mega ohms. Adicionalmente, tiene protección contra EMI (*Electromagnetic Interference*, Interferencia Electromagnética) que reduce la sensibilidad a las señales de RF no deseadas de fuentes como teléfonos móviles, Wi-Fi y transmisores de radio. En la figura 2.5 se puede observar una fotografía del TLV8544 en un encapsulado TSSOP-14.

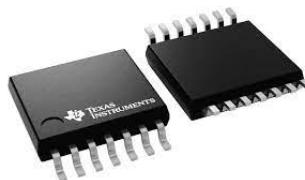


FIGURA 2.5. Fotografía del TLV8544 en un encapsulado TSSOP-14.

Las características técnicas más importantes del TLV8455 se presentan en la tabla 2.3.

Desde hace muchos años los amplificadores operacionales son un dispositivo muy utilizado y en el mercado existen muchas empresas que los fabrican y muchos modelos existentes. Estos son los criterios de elección del TLV8544 para el presente trabajo:

- Aplicación: por sus características técnicas, el TLV8544 está diseñado para ser parte de las etapas de amplificación y filtrado en el diseño de un sensor de movimiento PIR.

TABLA 2.3. Tabla de especificaciones del TLV8544

Caracteristica	Descripcion
Numero de canales	4
Fuente de alimentacion	1.7 V a 3.6 V
Corriente de salida por canal	30 mA
Corriente de operacion	500 nA
CMMR (<i>Common Mode Rejection Ratio</i> , Relacion de Rechazo del Modo Comun)	90 dB
Rango de temperatura	-40 °C a 125 °C
Corriente de polarizacion	100 fA
Ancho de banda de ganancia	8 kHz

- Documentacion: Texas Instruments, ademas del correspondiente *datasheet* del TLV8544, ofrece varios documentos tecnicos con ejemplos de diseño para el TLV8544.
- Costo: es un dispositivo de precio muy razonable por todas las características que ofrece.
- Consumo energetico: con sus 500 nA de corriente de funcionamiento por canal, el TLV8544 es una opción ideal para aplicaciones que requieran el uso de baterias.

2.1.3. Cámara OV2640

Otro de los componentes principales del sistema es la cámara, que permite obtener imágenes en un formato digital que posteriormente deben ser procesadas por los algoritmos de DL. Para este trabajo se utilizó el módulo ESP-LyraP-CAM. Este módulo integra un CCM (*Compact Camera Module*, Módulo de Cámara Compacto) con un sensor OV2640 en conjunto con dos reguladores de tensión para su correcto funcionamiento. En la figura 2.6 se puede observar unas fotografías del módulo ESP-LyraP-CAM y sus componentes.

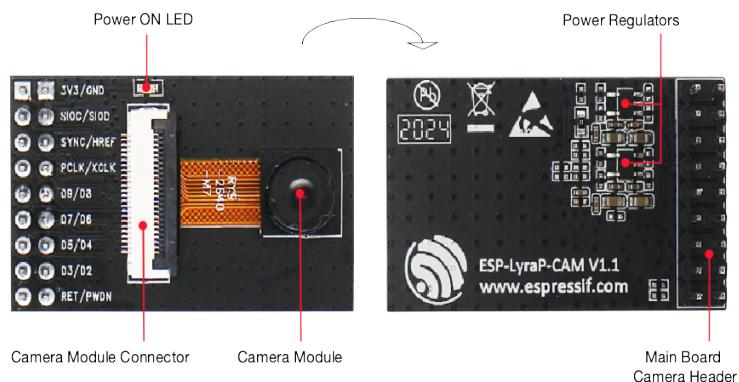


FIGURA 2.6. Diagrama en bloques del módulo ESP-LyraP-CAM.

El OV2640 de la empresa OmniVision es un sensor CMOS (*Complementary Metal-Oxide-Semiconductor*, Semiconductor de Oxido de Metal Complementario) de 2 MP, cuenta con una interfaz de comunicación compatible con DVP (*Digital Video*

Port, Puerto de Video Digital), soporta codificacion JPEG (Joint Photographic Experts Group, Grupo Unido de Expertos en Fotografia) y es de bajo consumo energetico. En la tabla 2.4 se muestran las caracteristicas tecnicas mas importantes del OV2640.

TABLA 2.4. Tabla de especificaciones del OV2640

Caracteristica	Descripcion
Tamano de matriz	1600x1200 (UXGA) Core: 1.3 V ± 5 %
Fuente de alimentacion	Analog 2.5 - 3.0 V I/O: 1.7 V - 3.3 V
Consumo energetico	Free running: 125 mW Standby: 600 µA
Formato de imagen del sensor	1/4"
Tasa de transferencia maxima	1600×1200 a 15 fps SVGA a 30 fps CIF a 60 fps
Sensibilidad	0.6 / Lux-sec
SNR	40 dB
Rango dinamico	50 dB
Tamano de pixel	x2.2x2.2 µm
Formato de salida	YUV/RGB/MJPEG

Los criterios para utilizar este modulo como camara del sistema son los siguientes:

- Costo: los modulos con el sensor OV2640 tienen un costo muy reducido en comparacion con otros disponibles en el mercado.
- Bajo consumo energetico: como se mostro en la tabal 2.4 el consumo energetico del modulo en modo *standby* es lo suficientemente bajo como para funcionar alimentado por baterias.
- Disponibilidad de codigo: al ser un modulo que ya lleva mucho tiempo en el mercado existen muchas bibliotecas de codigo para utilizarlo, lo que simplifica en gran medida el tiempo de desarrollo de *firmware*.

2.2. MTCNN (*Multi-Task Cascaded Convolutional Networks, Redes Convolucionales en Cascada Multitarea*)

MTCNN es un *framework* basado el el *paper* Joint Face Detection and Alignment using Multi-task Cascaded Convolutional Networks” by Zhang, Zhang and Zhi-feng, esta desarrollado para integrar las tareas de deteccion facial y alineamiento facial con ayuda de CNNS en cascada mediante aprendizaje multitarea. El proceso consta de tres etapas de CNNs que puede detectar rostros humanos, sus posiciones y las posiciones de sus rasgos faciales (nariz, ojos y boca). En la figura 2.7 se muestra el *pipeline* utilizado en MTCNN.

El *pipeline* de la figura 2.7 indica que las imagenes de entrada deben ser reducidas de tamano en varias escalas para formar un piramide de imagenes, que serviran para alimentar las siguientes tres etapas de MTCNN

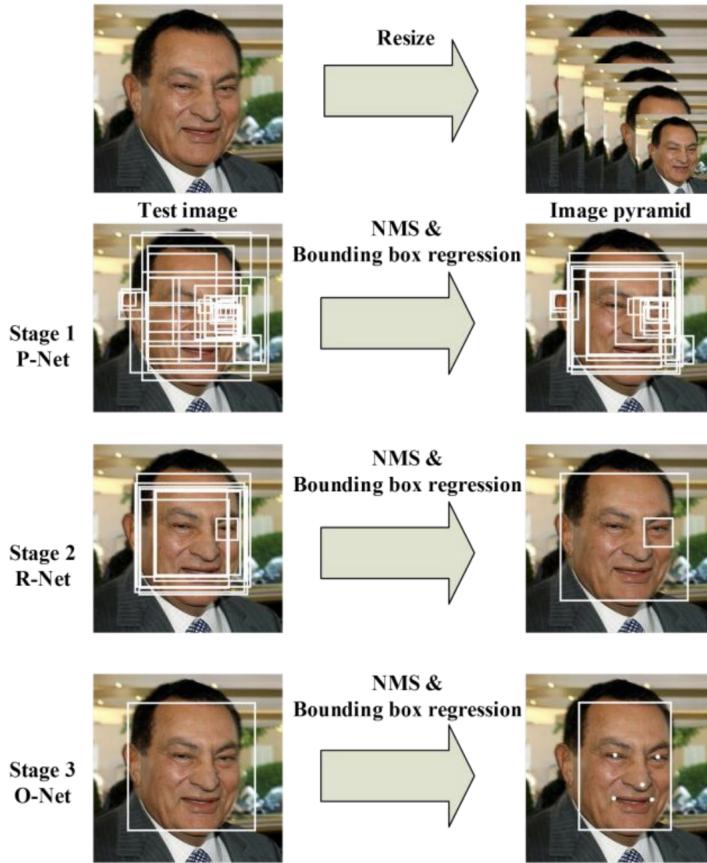


FIGURA 2.7. Pipeline de MTCNN.

2.2.1. P-Net

Tambien conocida como *Proposal Network* o red de propuestas, esta etapa esta compuesta de una FCN (*Fully Convolutional Network*, Red Totalmente Convolucional). La diferencia entre una FCN y una CNN es que la FCN no utiliza una capa FC como parte de su arquitectura. Tiene la funcion de obtener ventanas candidatas y sus vectores de regresion de *bounding box*. La regresion de *bounding box* es una tecnica para predecir la localizacion de un cuadro delimitador en el que se encuentra el objeto que quiere ser detectado, en este caso rostros humanos. Una vez que se obtienen estos vectores, se realizan una operacion conocida como NMS (*Non Max Supression*, Supresion no Maxima) para combinar las regiones sobrepuertas entre si. Finalmente las ventanas candidatas resultantes pasan a la siguiente etapa. En la figura 2.8 se muestra la arquitectura de P-Net.

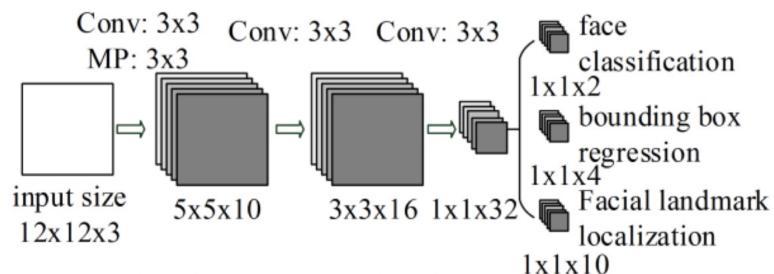


FIGURA 2.8. Arquitectura de P-Net.

2.2.2. R-Net

Esta es una CNN denominada *Refine Network* o red de refinamiento. Los candidatos provenientes de P-Net son la entrada de esta red. La arquitectura de R-Net reduce aun mas el numero de candidatos, realiza la calibracion con regresion de *bounding box* y emplea NMS para fusionar candidatos superpuestos. Para cada candidato de entrada, R-Net obtiene la probabilidad de si es un rostro o no, un vector de 4 elementos que es el *bounding box* y un vector de 10 elementos que representan la localizacion de rasgos faciales. En la figura 2.9 se muestra la arquitectura de R-Net.

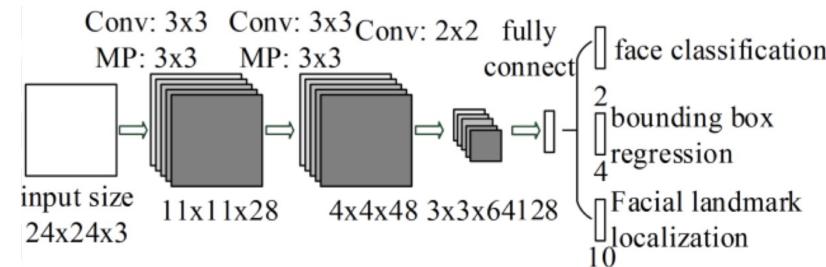


FIGURA 2.9. Arquitectura de R-Net.

2.2.3. O-Net

Conocida como *Output Network* o red de salida, es muy similar a R-Net, pero esta enfocada a describir el rostro con mas detalle y generar las cinco localizaciones para ojos, boca y nariz. Su arquitectura se muestra en la figura 2.10.

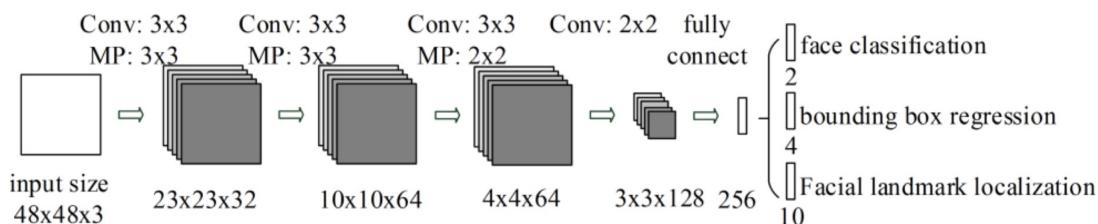


FIGURA 2.10. Arquitectura de O-Net.

2.2.4. Tareas de MTCNN

Como se explico en los puntos anteriores, MTCNN se compone de tres etapas que filtran ventanas candidatas y con la ayuda de NMS y calibracion con los vectores de regresion de *bounding box*, se detectan rostros y sus rasgos. Entonces, el proposito de MTCNN es cumplir con las siguientes tareas:

- Clasificacion rostro/no rostro: este es un problema de clasificacion binaria que utiliza una funcion de perdida de entropia cruzada.
- Regresion de *bounding box*: el objetivo de aprendizaje es un problema de regresion. Para cada candidato, se calcula el *offset* entre el candidato y el *ground truth* [ref] mas cercano. La funcion de perdida Euclidiana es utilizada para esta tarea.
- Localizacion de rasgos faciales: es formulada como un problema de regresion en el que la funcion de perdida es la distancia Euclidiana.

2.3. TensorFlow

TensorFlow es una plataforma de código abierto para ML. Tiene un ecosistema completo y flexible de herramientas, bibliotecas y recursos comunitarios que permite a los desarrolladores crear e implementar fácilmente aplicaciones basadas en ML.

Fue originalmente desarrollado por investigadores e ingenieros que trabajaban en el equipo Google Brain dentro de la organización de investigación de inteligencia artificial de Google y la versión inicial fue lanzada en 2015 bajo la licencia Apache License 2.0 [<https://github.com/tensorflow/tensorflow>].

TensorFlow proporciona APIs estables y oficiales para Python y C++, aunque también existen APIs para otros lenguajes de programación que no están garantizadas de manera oficial.

Sus características principales son:

- Autodiferenciación: es el proceso de cálculo automático del vector gradiente de un modelo respecto a cada uno de sus parámetros.
- Ejecución ansiosa: significa que las operaciones se evalúan de manera inmediata en lugar de agregarse a un gráfico computacional que se ejecuta más tarde.
- Distribuido: TensorFlow proporciona una API para distribuir el cálculo en múltiples dispositivos tanto para ejecución ansiosa como para gráficos computacionales.
- Perdidas: TensorFlow proporciona un conjunto de funciones de pérdida, también conocidas como funciones de costo.
- Métricas: TensorFlow brinda acceso a un API de métricas de uso común que se utilizan para evaluar el rendimiento de los modelos de ML.
- Optimizadores: TensorFlow ofrece un conjunto de optimizadores para entrenar redes neuronales, algunos son ADAM, ADAGRAD y SGD (*Stochastic Gradient Descent*, Descenso de Gradiente Estocástico).

Para el desarrollo de aplicaciones de ML existen varias otras bibliotecas, algunas de las más populares son: PyTorch, Caffe Computer Vision Library, DeepLearning, Neuroph, OpenNN, Theano, Torch y MXNet. Los criterios de elección de TensorFlow en este trabajo sobre las anteriores bibliotecas citadas fueron:

- Experiencia: este fue el criterio más fuerte en elección de TensorFlow como *framework* para el desarrollo de modelos de ML. El autor de este trabajo ya poseía experiencia trabajando con TensorFlow.
- Documentación: TensorFlow tiene mucha documentación oficial sobre su API y una gran variedad de tutoriales de uso.
- Herramientas para cuantización: TensorFlow cuenta con herramientas de cuantización de datos para optimizar el tamaño y tiempos de ejecución de modelos de ML.

2.4. AWS (*Amazon Web Services, Servicios Web de Amazon*)

AWS es una plataforma de *cloud computing* provista por Amazon que incluye una combinación de IaaS, PaaS y SaaS. Los servicios de AWS pueden ofrecer herramientas de poder computo, almacenamiento de datos y servicios de entrega de contenido.

AWS está dividido en distintos tipos de servicios que pueden ser configurados según las necesidades de cada usuario. Estos servicios puede dividirse en las siguientes categorías: computación, almacenamiento, bases de datos, administración de datos, migración, redes, herramientas de desarrollo, monitoreo, administración de *big data*, analíticas, AI, desarrollo móvil, mensajería y notificaciones.

De todos la extensa cantidad de servicios que ofrece AWS, para este trabajo se necesitaron solo los siguientes: IoT Core y Amazon Timestream.

2.4.1. IoT Core

Proporciona los servicios en la nube necesarios para conectar dispositivos IoT entre si y a los servicios a otros servicios de AWS. En la figura 2.11 se puede observar un diagrama de interconexión de dispositivos IoT y los servicios de AWS mediante IoT Core.

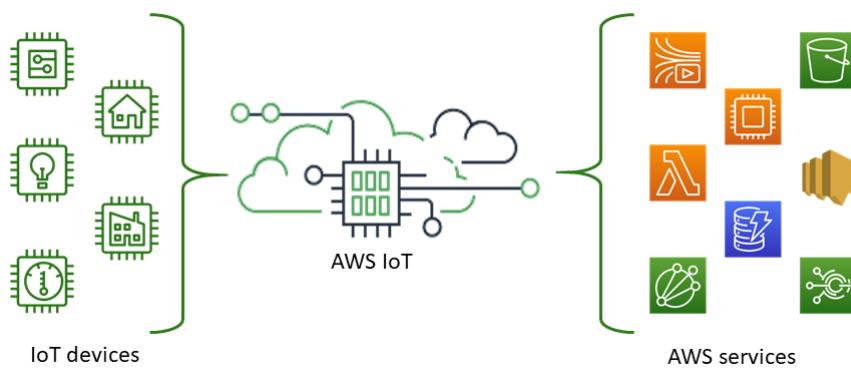


FIGURA 2.11. Diagrama de conexión entre dispositivos IoT y AWS.

IoT Core permite seleccionar las tecnologías más adecuadas y actualizadas para interconectar dispositivos IoT. Los protocolos de comunicación soportados son: MQTT (*Message Queuing and Telemetry Transport*, Cola de Mensajes y Transporte de Telemetría), MQTT sobre WSS (*Websocket Secure*, Websocket Seguro), HTTPS (*Hypertext Transfer Protocol Secure*, Protocolo de Transferencia de Hipertexto Seguro) y LoRaWAN.

El *broker* de IoT Core admite dispositivos y clientes que utilizan MQTT y MQTT sobre WSS para publicar y suscribirse a algún topico. También es compatible con dispositivos y clientes que utilizan HTTPS para publicar mensajes.

2.4.2. Amazon Timestream

Amazon Timestream es una base de datos de series temporales rápida, escalable y totalmente administrada, que facilita el almacenamiento y el análisis de billones de datos de series temporales al día. Timestream ahorra tiempos y costos con su capacidad de administrar los ciclos de vida de los datos de series temporales, donde mantiene los datos recientes en la memoria y mueve los datos históricos a un nivel de almacenamiento optimizado según las políticas definidas previamente por el usuario. El motor de consultas de Timestream permite acceder y analizar datos recientes e históricos al mismo tiempo. No necesita servidor y su tamaño se acomoda automáticamente para ajustar la capacidad y el rendimiento requeridos.

Los beneficios más notables que ofrece Amazon Timestream son:

- Sin servidor con escalado automático: a medida que cambian las necesidades de la aplicación, Timestream escala automáticamente para ajustar la capacidad.
- Administración de los ciclos de vida de los datos: ofrece niveles de almacenamiento, con un almacenamiento de memoria para datos recientes y un almacenamiento magnético para datos históricos. Timestream automatiza el proceso de transferencia entre ambos almacenamientos.
- Acceso simplificado a los datos: el motor de consultas de Timestream permite acceder a los datos de forma transparente, sin la necesidad de especificar el nivel de almacenamiento.
- Diseñado para series temporales: puede analizar datos de series de tiempo con SQL, con funciones integradas de series de tiempo para suavizar, aproximar e interpolar.
- Siempre cifrado: garantiza que los datos de series de tiempo siempre estén cifrados. Timestream permite especificar una clave administrada para encriptar datos en el almacenamiento magnético.

2.5. Grafana

Es una aplicación web multiplataforma de análisis y visualización interactiva. Proporciona tablas, gráficos y alertas a través de la web cuando se conecta a alguna fuente de datos compatible. Los usuarios pueden crear *dashboards* de monitoreo de datos complejos con ayuda de generadores de consultas interactivos.

Como herramienta de visualización, Grafana es muy popular gracias a las siguientes características:

- Se conecta a muchas fuentes de datos populares como Graphite, Prometheus, InfluxDB, Elasticsearch, MySQL, PostgreSQL, entre otros.
- Es de código abierto y distribuida bajo la licencia AGPL-3.0, que permite desarrollar complementos desde cero para integrarla con otras fuentes de datos.
- Ayuda a estudiar, analizar y monitorear datos durante un período de tiempo configurable por el usuario.

- Puede ser implementado localmente por organizaciones que quieran mantener sus datos confidenciales sin acceso a internet.
- Se pueden configurar alertas que se envian por otros medios de comunicación bajo ciertas condiciones pre establecidas.

Capítulo 3

Diseño e implementación

3.1. Análisis del software

La idea de esta sección es resaltar los problemas encontrados, los criterios utilizados y la justificación de las decisiones que se hayan tomado.

Se puede agregar código o pseudocódigo dentro de un entorno lstlisting con el siguiente código:

```
\begin{lstlisting}[caption= "un epígrafe descriptivo"]
las líneas de código irían aquí...
\end{lstlisting}
```

A modo de ejemplo:

```

1 #define MAX_SENSOR_NUMBER 3
2 #define MAX_ALARM_NUMBER 6
3 #define MAX_ACTUATOR_NUMBER 6
4
5 uint32_t sensorValue[MAX_SENSOR_NUMBER];
6 FunctionalState alarmControl[MAX_ALARM_NUMBER]; //ENABLE or DISABLE
7 state_t alarmState[MAX_ALARM_NUMBER]; //ON or OFF
8 state_t actuatorState[MAX_ACTUATOR_NUMBER]; //ON or OFF
9
10 void vControl() {
11     initGlobalVariables();
12     period = 500 ms;
13
14     while(1) {
15         ticks = xTaskGetTickCount();
16         updateSensors();
17         updateAlarms();
18         controlActuators();
19
20         vTaskDelayUntil(&ticks, period);
21     }
22 }
```

CÓDIGO 3.1. Pseudocódigo del lazo principal de control.

Capítulo 4

Ensayos y resultados

4.1. Pruebas funcionales del hardware

La idea de esta sección es explicar cómo se hicieron los ensayos, qué resultados se obtuvieron y analizarlos.

Capítulo 5

Conclusiones

5.1. Conclusiones generales

La idea de esta sección es resaltar cuáles son los principales aportes del trabajo realizado y cómo se podría continuar. Debe ser especialmente breve y concisa. Es buena idea usar un listado para enumerar los logros obtenidos.

Algunas preguntas que pueden servir para completar este capítulo:

- ¿Cuál es el grado de cumplimiento de los requerimientos?
- ¿Cuán fielmente se pudo seguir la planificación original (cronograma incluido)?
- ¿Se manifestó algunos de los riesgos identificados en la planificación? ¿Fue efectivo el plan de mitigación? ¿Se debió aplicar alguna otra acción no contemplada previamente?
- Si se debieron hacer modificaciones a lo planificado ¿Cuáles fueron las causas y los efectos?
- ¿Qué técnicas resultaron útiles para el desarrollo del proyecto y cuáles no tanto?

5.2. Próximos pasos

Acá se indica cómo se podría continuar el trabajo más adelante.