



Universidad Veracruzana

**Facultad de Instrumentación Electrónica**

**Región Xalapa**

Ingeniería en Instrumentación electrónica

**“Sistema de reconocimiento de objetos en tiempo real para  
navegación en drones”**

Tesis para acreditar la Experiencia Recepcional

Presenta:

**Román Gabriel Velásquez Reyes**

Directores de Tesis:

Dr. Roberto Castañeda Sheissa

Dr. Héctor Vázquez Leal

Junio de 2022

“Lis de Veracruz: Arte, Ciencia, Luz”





Universidad Veracruzana

Facultad de Instrumentación Electrónica  
Región Xalapa

Ingeniería en Instrumentación Electrónica

*Sistema de reconocimiento de objetos en tiempo real para  
navegación en drones*

Tesis para acreditar la Experiencia recepcional

Presenta:  
Román Gabriel Velásquez Reyes

Directores de tesis:  
Dr. Roberto Castañeda Sheissa  
Dr. Héctor Vázquez Leal



Por este conducto, quien suscribe, C. Román Gabriel Velásquez Reyes, alumno del programa educativo de Ingeniería en Instrumentación Electrónica, con matrícula S17012869, manifiesta haber realizado el presente trabajo recepcional en la modalidad de Tesis para acreditar con ello la experiencia educativa Experiencia Recepcional inscrita en el periodo Febrero – Julio 2022 y que lleva por título: Sistema de reconocimiento de objetos en tiempo real para navegación en drones.

Este trabajo escrito ha sido revisado y aprobado por quienes fungieron como directores de este, los académicos Dr. Roberto Castañeda Sheissa y Dr. Héctor Vázquez Leal, quienes confirman que el trabajo tiene la calidad suficiente para ser evaluado para el fin mencionado.

Asimismo, manifiesto que el trabajo es de mi total autoría y que lo entrego en forma electrónica por los medios indicados por el académico encargado de la Experiencia Educativa Experiencia Recepcional. Lo anterior con la finalidad de cumplir con los requisitos establecidos y llevar a cabo la evaluación de la Experiencia Recepcional a través de la exposición oral, de acuerdo con lo establecido en el Estatuto de los Alumnos vigente.

Xalapa, Ver., a 20 de Junio de 2022.

“Lis de Veracruz: Arte, Ciencia, Luz”

Atentamente



Román Gabriel Velásquez Reyes

Vo. Bo.



Dr. Roberto Castañeda Sheissa

Vo. Bo.



Dr. Héctor Vázquez Leal



## **Dedicatoria**

A Dios, familia y amigos.

# Índice

Dedicatoria o agradecimientos .....	2
Índice .....	3
Resumen .....	8
I. Introducción .....	9
II. Objetivos.....	10
Objetivo General .....	10
Objetivos Específicos.....	10
III. Hipótesis .....	11
IV. Justificación de la investigación.....	11
V. Estado del Arte .....	12
1. Conceptos Básicos.....	15
1.1    Inteligencia Artificial .....	15
1.2    Machine Learning .....	15
1.2.1    Aprendizaje Supervisado.....	16
1.2.2    Aprendizaje No Supervisado.....	16
1.3    Deep Learning.....	16
1.4    Neurona.....	17
1.5 Redes Neuronales Artificiales.....	18
1.5.1    Nodo BIAS o Sesgo.....	18
1.5.2    Función de Propagación .....	18
1.6 Estructura de red neuronal .....	19
1.6.1    Capas .....	19
1.6.2    Funciones de Activación .....	20
1.6.4    Píxeles.....	22
1.6.5    Convoluciones .....	23
1.6.6    Capa Convolutiva .....	23
1.7    Detección de Objetos .....	26
1.7.1    Modelos de detección de Objetos .....	27
1.7.2    COCO (Common Objects in Context).....	27



1.7.3	Serie YOLO .....	28
1.8	YOLOv5 .....	28
1.9	Estructura del Modelo.....	30
1.10	Aportaciones y Mejoras de Datos .....	30
1.11	Conjunto de Datos.....	31
1.11.1	Conjunto de Datos Personalizado .....	31
1.12	Técnicas de Aumento de datos .....	31
1.13	Imágenes de Entrenamiento, Test (prueba) y Validación.....	32
1.14	Etiquetado .....	32
1.15	Entrenamiento .....	32
1.16	Índices de evaluación.....	33
1.16.1	Precisión en Machine Learning .....	34
1.16.2	Recall .....	34
1.16.3	IoU (Intersection over Union) .....	35
2.	Metodología.....	36
2.1	Obtención del conjunto de datos.....	36
2.2	Etiquetado .....	37
2.3	Obtención de Snapshot .....	38
2.4	Entrenamiento en Google Colab.....	40
2.5	Set Up de Hardware .....	46
2.6	Set Up de Software .....	48
2.7	Librerías-Frameworks.....	48
3.	Pruebas y resultados obtenidos.....	49
4.	Conclusiones.....	58
4.1	Recomendaciones .....	59
5.	Anexos .....	65
6.	Referencias .....	62

## Índice de Figuras

Figura 1. Modelo de Aprendizaje Supervisado. Ejemplo: Aumento de Mosaico. ....	16
Figura 2. Estructura Red Neuronal con Pesos .....	17
Figura 3. Estructura de una Red Neuronal .....	18
Figura 4. Incorporación del Sesgo a la Red.....	19
Figura 5. Capas de una Red Neuronal .....	20
Figura 6. Suma de Operaciones de Regresión Lineal y No Lineal.....	21
Figura 7. Gráfica Operacional de la Función Sigmoide. ....	21
Figura 8. Modelo de Red Convolutiva con Imagen de Entrada.....	22
Figura 9. Valores de Píxeles de 0 a 255 .....	22
Figura 10. Imagen a color conformada por 3 Canales e Imagen Blanco y Negro Conformado por 1 canal .....	23
Figura 11. Operación Convolutiva en una imagen de 5x5, con operación de Zero Padding y un Filtro de 3x3. [20].....	24
Figura 12. Operación Convolutiva con distintos valores de núcleo, Obteniendo nuevas imágenes .....	25
Figura 13. Operación MaxPooling con Stride (2,2) [20].....	25
Figura 14. Ejemplo Operación de Zero Padding en una Matriz de 4x4. [20] .....	26
Figura 15. Funcionamiento de un Detector de Objetos. ....	26
Figura 16. Evolución “Machine Learning” y “Deep Learning” (Algunos de los Algoritmos Actuales) [30]. ....	28
Figura 17. Características de los Modelos Pre Entrenados de YOLOv5.....	29
Figura 18. Rendimiento de los Distintos Modelos Evaluados con el Conjunto de Datos de COCO. [32] .....	29
Figura 19 Ejemplos Aumento de Datos YOLOv5.....	30
Figura 20 Un boceto de una representación gráfica de la métrica IoU [38].....	35
Figura 21. Esquema Básico de la Ejecución del Detector de Objetos.....	36
Figura 22. Muestras de fotografías de Cruz de 224x224 px.....	37
Figura 23. Procedimiento para generar y recopilar conjunto de datos. ....	37
Figura 24. Ventana de la herramienta de LabelImg y el archivo que genera con las anotaciones del área de interés. ....	38
Figura 25. Configuración de nuestro set de datos con Roboflow.....	39
Figura 26. Ventanas de configuración del tipo de exportación del conjunto de datos generados.....	40
Figura 27. Configuración inicial para Google Colab .....	40
Figura 28 Línea de código para clonar el repositorio de YOLOv5 .....	41
Figura 29. Líneas de código para instalar dependencia de YOLOv5 .....	41
Figura 30. Requirements.txt .....	41
Figura 31. Líneas de código del paquete PIP del conjunto de datos .....	42
Figura 32. Actualización de las Carpetas al momento .....	42
Figura 33. Archivo generado “data.yaml” .....	43
Figura 34. Línea de código “train.py” .....	43
Figura 35. Entrenamiento del detector YOLOv5s personalizado.....	44
Figura 37. Líneas de código para invocar la herramienta de Tensorboard.....	44
Figura 36. Gráficas de las métricas del entrenamiento.....	44
Figura 38. Resultados trazados de Tensorboard del conjunto de datos YOLOv5.....	45
Figura 39. Línea de código para usar el script de “detect.py” .....	45

Figura 40. Inferencia con los pesos calculados de YOLOv5s con en Imágenes de train. ....	46
Figura 41. Línea de código para exportar el modelo “best.pt” .....	46
Figura 42. Gráfica de la métrica de Map 0.5 .....	51
Figura 43. Ejecución del script de “detect.py” para imágenes .....	53
Figura 44. Capturas de pantallas de videos “Video_tamaños” y “video_parque” (ver anexo 1) .....	55
Figura 45. Figuras de distintos tamaños para la evaluación de la inferencia en la Raspberry Pi .....	56
Figura 46. Ventanas de la transmisión en tiempo real y la terminal mostrando información sobre los objetos detectados. ....	56

## **Lista de abreviaturas:**

IA: Inteligencia Artificial

CNN: Convolutional Neural Network

RNA: Redes Neuronales Artificiales

SBU: Single Board Unit

GPU: Unidad de Procesamiento Grafico

RAM: Random Access Memory

## Resumen

Conforme ha avanzado la tecnología en las últimas décadas, la inteligencia artificial (IA) participa en el día a día de los seres humanos. Actualmente, se encuentra su aplicación en el ámbito literario como en la generación de texto, traducción de idiomas; en el ámbito de la medicina, pronóstico de enfermedades y probabilidad de que respondan satisfactoriamente a tratamientos; en el ámbito de comunicación, redes sociales, formación de algoritmos en relación a preferencias, asistentes de voz como Siri, Cortana, Google Now o Alexa; en el ámbito de transporte, detección, identificación y seguimiento de objetos como automóviles de la agencia de Tesla que utiliza esta tecnología para su conducción autónoma.

Otra aplicación que utiliza la identificación y detección de objetos se aplica con los drones de búsqueda para coordinar operaciones de rescate o evaluar situaciones de emergencia. Por lo tanto, agiliza y evita el riesgo del personal de emergencias en lugares peligrosos para rescate de personas. La desventaja de esta tecnología es el precio elevado y requerimiento de capacitación para aprender a utilizarlo. Además, su uso en tiempo real puede generar confusión para distinguir entre el objeto a detectar de rescate e interferencia, siendo que el tiempo es de alta relevancia en situaciones de emergencia.

Por lo tanto, el objetivo del presente trabajo fue el desarrollo de un sistema de detección de figuras para uso en drones de rescate en tiempo real, utilizando el enfoque de Deep Learning y redes neuronales convolucionales. Se realizó un entrenamiento utilizando redes neuronales convolucionales con una red neuronal pre entrenada (Yolov5). Se evaluó el rendimiento para detectar objetos en tiempo real y adquirir una imagen completa en menos de 2 segundos, para predecir cuadros delimitadores “Bouding boxes” y probabilidad de la clase en una Raspberry pi 4, modelo B. Esto permitió sugerir la aplicación de este enfoque de IA en drones de operaciones de rescate, debido a la predicción e identificación del 90% de objetos.

La presente tesis es el primer paso para la optimización del modelo propuesto, debido a que se logró identificar figuras en distintos ambientes con una velocidad apropiada, permitiendo a los operadores tener el tiempo adecuado para decidir y actuar.

**Palabras clave:** Machine Learning, Deep learning, Redes Neuronales Convolucionales, Detección de Objetos, YOLOv5, mAP, Raspberry Pi.

## I. Introducción

Dado el aumento de eventos como terremotos, accidentes de tránsito, huracanes, inundaciones y desapariciones de personas, han provocado el diseño de equipos y técnicas que agilicen los procesos de rescate [1].

Las entidades que trabajan en estos eventos tienen una serie de protocolos y tiempos de acción, dependiendo de la magnitud del desastre. Disponen de tres tipos de grupos, los cuales llegan en distintos tiempos al lugar. Los primeros dos grupos realizan la búsqueda y rescate con recursos humanos y el último grupo se basa en el uso de drones [2].

La implementación de las nuevas tecnologías en los procesos de búsqueda y rescate, como los drones, se basan en el uso de dispositivos voladores no tripulados y cumpliendo con todos los parámetros de seguridad y permisos requeridos. Reduciendo el tiempo de búsqueda inicial por su mayor cobertura de área en sitios de difícil acceso, donde la intervención humana para el rescate es cuestionada dado su alto nivel de peligro. Resaltando la desventaja que el rastreo realizado mediante estos dispositivos solo es superficial [3].

Mediante el uso de inteligencia artificial se podría especializar este tipo de protocolo de emergencia con reconocimiento de objetos de rescate [3]. Como el uso de una figura universalmente reconocida en botiquines o personal de salud, una cruz.

El presente trabajo está enfocado en la solución de procesos de búsqueda y rescate en cualquier emergencia utilizando redes neuronales convolucionales para la detección de figuras.

## **II. Objetivos**

### **Objetivo General**

Desarrollar un modelo detector de objetos mediante librerías computacionales de Deep Learning usando una red neuronal pre entrenada “YOLOv5” e implementarlo a Raspberry Pi 4 como un sistema de control en un dron de rescate.

### **Objetivos Específicos**

1. Definir un procedimiento y metodología para crear un conjunto de datos (imágenes) que se utilizarán en el entrenamiento de la red neuronal convolucional.
2. Categorizar las imágenes de cruz y círculo, utilizando herramientas y técnicas para etiquetar y obtener las anotaciones del conjunto de datos.
3. Seleccionar los mejores parámetros de entrenamiento (Batch size, epoch e iteraciones) para obtener un modelo con excelentes métricas (mAP).
4. Comparar y analizar las distintas métricas de los modelos entrenados (precision y recall).
5. Evaluar el rendimiento del modelo de detección de objetos en la Single Board Unit Raspberry Pi 4 bajo el sistema operativo Linux (Distribución Debian Bullseye), además de utilizar el módulo Raspi Camera en modo de operación “tiempo real”.

### **III. Hipótesis**

La implementación de Redes Neuronales Convolucionales (CNN) de bajo costo utilizando una SBU (Raspberry Pi 4) para la rápida identificación de objetos, permitirá su uso en posibles situaciones de búsqueda, en particular, la detección de patrones para rescate y apoyo.

### **IV. Justificación de la investigación**

Actualmente la inteligencia artificial se aplica en diferentes ámbitos de la vida cotidiana. Posiblemente una de sus aplicaciones con mayor importancia sería en el uso de emergencias en búsqueda y rescate de personas.

Sin embargo, no existen estudios recientes sobre el uso de Deep Learning y redes neuronales convolucionales, además de su aplicación en drones para detectar rutas de acceso para el personal de rescate. Agregando que se debe considerar que estas tecnologías no son de acceso fácil de adquirir o aprender. Por lo que se optó por utilizar una SBU, Raspberry Pi 4, hardware capaz de replicar el objetivo del dron de rescate, entre sus ventajas destaca que este puede ser reprogramable y aditarle mejoras a su sistema. Implementar una red neuronal convolucional entrenada que complemente su tarea de búsqueda de objetos personalizada que se requiere detecte. De tal manera, que resulta necesario evaluar si con el uso de redes neuronales convolucionales en la detección de objetos de figuras como cruces, o círculos, permitirán el aumento en la probabilidad de identificación de personas en situaciones de búsqueda y rescate.

Por todo lo anterior, la justificación de este trabajo surgió por la necesidad de generar información acerca del uso de Deep Learning y redes neuronales convolucionales enfocado en la detección de objetos para aplicarlo en situaciones de emergencia.



## V. Estado del Arte

En la actualidad existe una conexión directa y una gran interacción y dependencia entre el progreso de la sociedad y el avance de la ciencia, convirtiéndose en uno de los factores esenciales para el bienestar de las personas y para el desarrollo social de las comunidades y países. Por otra parte, los avances y progresos de la ciencia tienen efectos directos sobre la sociedad actual, pero en especial sobre la sociedad futura; porque ha modificado y transformado la armonía entre la naturaleza, la sociedad y el hombre [4]. Con el desarrollo de la ciencia y la tecnología, ha surgido gradualmente la investigación sobre sistemas inteligentes en CNN.

En este apartado se describirán diferentes métodos de aplicación de las redes neuronales para generar diferentes trabajos relacionados a una serie de algoritmos y soluciones que han sido desarrollado hasta el momento en relación con el tema de detección de objetos y redes neuronales convolucionales.

La detección de objetos tiene una amplia gama de aplicaciones, incluida la visión de robots, la electrónica, seguridad, conducción autónoma, interacción humano-computadora y realidad aumentada. Esta tecnología comenzó a implementarse entre los años 80 y 90, con la limitación de tecnología como trabajar con procesadores poco potentes y bases de datos muy pequeñas [5].

Investigadores como Yann LeCun, quien en 1989 con el primer algoritmo de “Backpropagation” introdujo el primer diseño de una red neuronal convolucional para la detección de números escritos en cheques bancarios de manera totalmente supervisada [6].

Esto en un futuro motivaría a varios grupos de investigadores a crear grandes bases de datos a mano. Proporcionando categorías y cuadros delimitadores de millones de imágenes para cierto punto de realizar competencias donde los programas de software compiten para clasificar y detectar correctamente objetos con sus propios modelos de CNN. Donde destaca la aparición del grupo de Geoffrey Hinton en la competición ImageNet. ImageNet fue una competición que consistía en resolver un problema de clasificación de imágenes en 1000 categorías después de haber realizado un entrenamiento sobre 1.4 millones de imágenes. Se consiguió una tasa de acierto del 83.6 %, esto gracias al uso de unidades de procesamiento de gráficos (GPU) [7].

La Navegación de UAVs con Deep Learning, está relacionado con una red neuronal para que un dron pueda moverse a través de entornos desconocidos y pueda esquivar obstáculos ya que en interiores no podrá usar el GPS debido a que existen interferencias haciendo que no pueda funcionar correctamente como lo haría trabajando en exterior, implementando el framework ROS (Robot Operating System) que incluye un extenso conjunto de recursos para todas las partes de software de un robot, para evitar empezar desde cero [8].

Trabajar con GPU permite acelerar los cálculos computacionales, debido a que las CNN trabajan con tensores y estos utilizan una gran cantidad de números y su procesamiento. Para el año de 2018 se terminó de desarrollar “PyTorch,” una biblioteca de Python. Su investigación fue llevada a cabo por el departamento de investigación de IA, destacando a PyTorch como principal biblioteca para la investigación y desarrollo en el campo del machine learning, centrado en el desarrollo de redes neuronales [9].

Siendo en un futuro PyTorch usado para el desarrollo de aplicaciones como el Autopilot de Tesla y el Pyro de Uber, se ha convertido en uno de los frameworks de Deep Learning más populares del mundo [10].

Cuando se trata de sistemas de aprendizaje automático, TensorFlow y PyTorch son muy populares entre los profesionales. No hay otro método que se acerque a lo que estos dos productos de Google y Facebook tienen reservados. Estos marcos han ganado popularidad de manera constante en la cultura de la IA. Para los investigadores de aprendizaje automático, PyTorch ha sido el sistema de referencia [9].

Las citas de PyTorch en los artículos de ArXiv aumentaron un 194 % en la primera mitad de 2019, aunque la cantidad de contribuyentes al sitio aumentó en más del 50 %.

Muchas compañías en todo el mundo comenzaron varios proyectos para explorar el uso de CNN en sistemas inteligentes de transporte (ITS), por sus siglas en inglés. Tesla es la compañía que actualmente se encuentran en la fase más avanzada del desarrollo de coches autónomos. “Tesla Autopilot” es un sistema avanzado de asistencia a la conducción desarrollado por Tesla que se lanzó por primera vez en 2015 con la versión 7 de su software. Todos sus vehículos estaban equipados con ocho cámaras, doce sensores ultrasónicos y un radar orientado hacia adelante para permitir la capacidad de conducción automática. Las CNN se utilizan para detectar objetos a partir de las imágenes capturadas por estas cámaras, se procesan y realizan otras acciones, eventualmente.

Tesla utiliza Pytorch para el entrenamiento de CNN. Para el piloto automático, entrena alrededor de 48 redes que hacen 1000 predicciones diferentes y requiere 70 000 horas de GPU. Las redes se ejecutan en el propio hardware de Tesla, lo que les otorga control total de estas funciones [ 9].

En el presente proyecto se utilizó la red neuronal de “YOLOv5”, una moderna red convolucional conocida no solo por ser la sucesora de la versión 4 de YOLO, sino también por su accesibilidad sin utilizar el marco de “Darknet” [11]. YOLOv5 no trabaja en el entorno de TensorFlow, utiliza PyTorch. Siendo este más llamativo para los investigadores, aumentando su número de citas y consultas. Teniendo contribuciones a la biblioteca y desarrollo de YOLOv5 para su aplicación en distintos campos.

# **I. Conceptos Básicos**

En este apartado se realiza una exploración sobre las definiciones de la teoría que se desarrolla en esta investigación, con respecto a la Inteligencia Artificial, redes neuronales convolucionales, arquitectura YOLO y varios aspectos que sientan la base de la detección de objetos.

## **1.1 Inteligencia Artificial**

La inteligencia artificial (IA) se ha desarrollado desde hace muchas décadas, como el modelo matemático de la máquina de Turing desarrollado por el británico Alan Turing [12]. Esto inspiró a la creación de los primeros prototipos de computadoras del siglo XX, convirtiéndolo en el pionero y fundador de la rama de la IA. Posteriormente se vería aplicado en el ajedrez con la super computadora Deep Blue [13].

La IA tiene subcategorías como Big Data, robótica, sistemas de lenguaje natural, lógica difusa, sistemas expertos y machine learning o aprendizaje automático, que permite que el sistema aprenda y relacione información similar a una persona. También se incorpora su subdisciplina Deep Learning en relación con las redes neuronales convolucionales [14].

## **1.2 Machine Learning**

Traducido al español como aprendizaje automático es una disciplina del ámbito de la IA que abarca los sistemas que aprenden automáticamente, sin necesidad de programación, su conocimiento se debe a la capacidad de aprender y adaptarse, (relacionando situaciones complejas) gracias a un algoritmo que le permite predecir eventos futuros [9].

Existen dos tipos de aprendizaje en machine learning: el aprendizaje supervisado y el no supervisado.

### 1.2.1 Aprendizaje Supervisado

Este aprendizaje consiste en un conjunto de una base de datos previamente etiquetados por un ser humano con el fin de clasificar de forma directa. Es decir, el usuario se encarga de indicar al algoritmo los datos de entrada, así como las predicciones o salidas que debe dar el modelo para esos mismos datos (figura 1). Los parámetros de la red se ajustan o corrigen con distintos métodos de acuerdo con la magnitud del error [ 15].

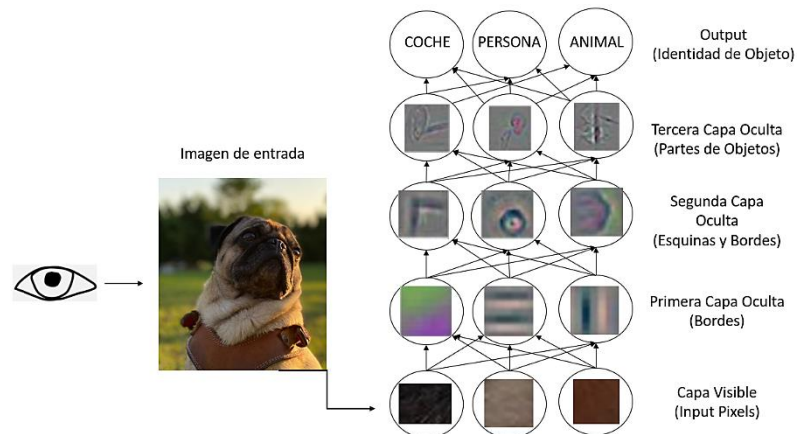


Figura 1. Modelo de Aprendizaje Supervisado. Ejemplo: Aumento de Mosaico.

### 1.2.2 Aprendizaje No Supervisado

El aprendizaje no supervisado, su propósito principal es el análisis de patrones, la síntesis de los datos observados, o bien una agrupación, sin que se tenga una etiqueta para cada clase de patrón u objetivo. Sólo ajustan su modelo predictivo tomando en cuenta los datos de entrada, sin importar los de salida. El aprendizaje supervisado se utiliza en sistemas de clasificación con características en común [14].

## 1.3 Deep Learning

El aprendizaje profundo es una rama del machine learning que se aplica fundamentalmente para el estudio de imágenes. Se define como un conjunto de técnicas y pasos a seguir formando un algoritmo para lograr que un dispositivo aprenda de la misma manera que haría el trabajo un ser humano. Este sistema de aprendizaje se inspira en el funcionamiento de las redes neuronales del cerebro humano para procesar la información y funciona con una base matemática muy compleja [15].

## 1.4 Neurona

Una neurona, o la unidad básica de procesamiento, es el elemento básico de una red neuronal similar a una neurona biológica encargada de recibir, procesar y transmitir información que llega al cerebro. Estas neuronas tienen conexiones de entrada, a través de los que reciben estímulos externos “los valores de entrada”. Con estos valores ( $X_1, X_2 \dots X_n$ ), la neurona realizará un cálculo interno y genera un valor de salida “ $Y_1$ ”. La neurona es la representación de una función matemática (ver figura 2) [16].

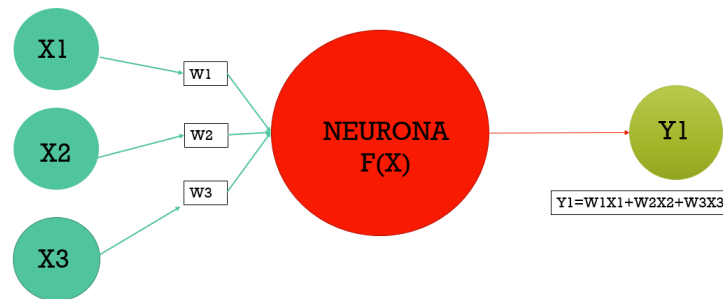


Figura 2. Estructura Red Neuronal con Pesos

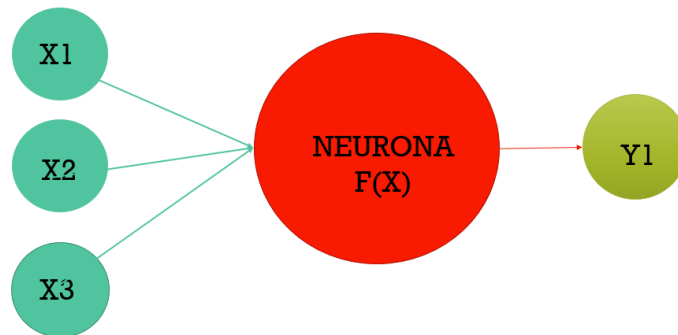
Internamente en la neurona se ejecuta un cálculo numérico, utiliza los valores de entrada para realizar una suma ponderada de ello. La ponderación de cada una de las entradas viene dada por el peso que se le asigna a cada una de las conexiones de entrada ( $W_1, W_2 \dots W_n$ ).

Cada una de estas conexiones tiene un “peso” asignado, este peso es un valor numérico asignado que representa la importancia de conexión entre las neuronas.

Las señales de entrada se multiplican por los pesos, es decir cada entrada tendrá asociado un peso correspondiente, siendo los parámetros del modelo, de esta manera cada entrada puede ser ajustada individualmente para cada  $X_1, X_2$  y  $X_3$ . Se pueden ver todas las entradas como un vector de entrada y los pesos correspondientes como los pesos del vector [17].

## 1.5 Redes Neuronales Artificiales

Las RNA son modelos computacionales, formados por unos elementos conocidos como neuronas las cuales están conectadas mediante unos valores variables llamados pesos sinápticos a otras neuronas. Mediante las redes neuronales se pueden realizar trabajos de predicción o clasificación  $Y1$  (ver figura 3) [18]. La neurona no es más que una representación de una función matemática [16].



*Figura 3. Estructura de una Red Neuronal*

### 1.5.1 Nodo BIAS o Sesgo.

Justo antes de aplicar la función de activación, cada neurona añade a la suma de productos un nuevo término constante. Cada neurona a excepción de la capa de entrada tiene un sesgo [15]. El sesgo, también conocido como BIAS en inglés, es un valor numérico que se ajusta durante la fase de aprendizaje, justo antes de aplicar la función de activación, se encarga de mover verticalmente nuestra función y se representa como otra conexión a la neurona pero que la variable siempre esta asignada a “1” y podemos manipular moviendo el parámetro de sesgo. De esta manera nuestra neurona actuaría como un modelo de regresión lineal. Varios de los problemas con relación a las redes neuronales se resuelven combinando varias neuronas y conexiones para poder conseguir modelos más complejos [15].

### 1.5.2 Función de Propagación

La función de propagación se encarga de modificar sumando y ponderando todas las entradas con sus pesos, a lo cual se le añade un sesgo, que no es más que un offset con el que el valor de la suma ponderada de partida, ajustando el resultado final. Corresponde a un factor de corrección [19].

## 1.6 Estructura de red neuronal

Podemos ver la arquitectura genérica de una RNA en la figura 4

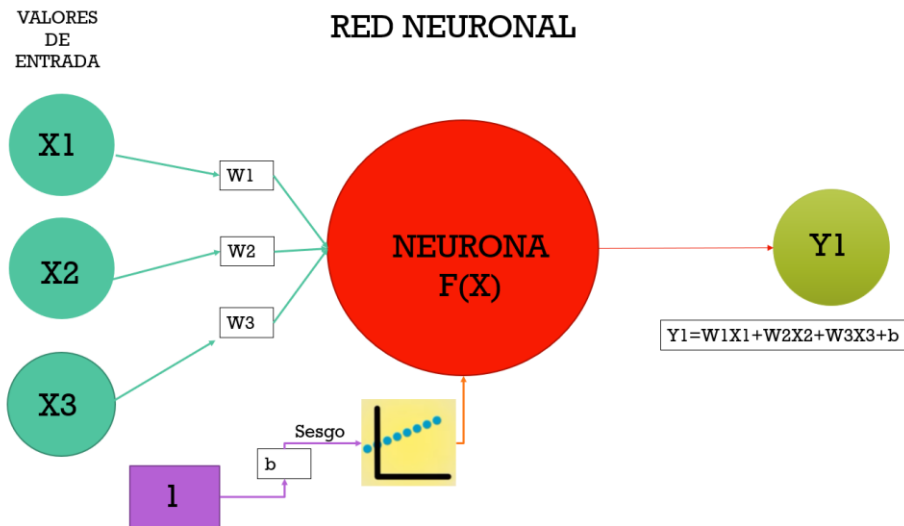


Figura 4. Incorporación del Sesgo a la Red

Cada círculo representa una neurona. Estas se organizan en columna, o mejor dicho en capas. Las neuronas que se encuentran en la misma capa recibirán la misma información de entrada de la capa anterior y los cálculos que realicen pasarán de manera secuencial. Se denominan: capas de entrada, ocultas y salida. Las neuronas de las capas tienen conexiones con un número asociado, llamado peso.

Cuando se tiene una señal a la entrada, se multiplica por el valor del peso que se asigna a esta entrada con los valores de cada neurona. Es decir, si una neurona tiene tres entradas, entonces tiene tres pesos que se pueden ajustar individualmente. Los pesos generalmente se ajustan durante la fase de aprendizaje.

Después de esto, se suman las señales de entrada modificadas. También es posible agregar adicionalmente el llamado sesgo ( $b$ ) a esta suma.

Finalmente, se debe determinar la salida real. Para ello, se aplica una función de activación o paso  $\Phi$  a la suma ponderada de los valores de entrada [20].

### 1.6.1 Capas

Una RNA básica está conformada por 3 tipos de capas. Estas pueden tener una o más neuronas, reciben la información procesada por la neurona anterior, haciendo que la red pueda aprender por conocimiento jerarquizado.



Los datos ingresan a la RNA a través de la capa de entrada y luego viajan a través de muchos niveles ocultos antes de llegar a la capa de salida. La predicción de la red se refleja a través de la capa de salida. En términos de pérdida o error, la salida de la red se compara con las etiquetas reales [21].

- Capa de Entrada

Se adquieren y procesan información de señales, datos o variables de entrada, es decir, la información proveniente de sensores o de algún sistema de procesamiento para introducirla en el sistema [16].

- Capa Oculta

Son las capas intermedias, también conocidas como capas ocultas, donde se extraen características abstractas de capas previas, tienen conexión con el sesgo o BIAS. En esta capa se obtienen los descriptores o patrones más significativos [16].

- Capa de Salida

Se obtiene con ayuda de las neuronas una clasificación o predicción por la red donde obtendremos el resultado calculado [16]. Se puede apreciar la forma y orden de las capas en la figura 5.

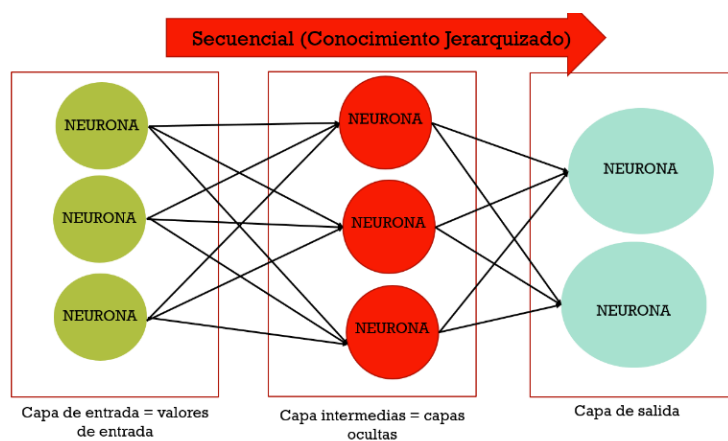


Figura 5. Capas de una Red Neuronal

### 1.6.2 Funciones de Activación

Las operaciones de redes neuronales equivalen a productos y sumas, conectando múltiples neuronas de forma secuencial, obteniendo en su salida una función de regresión lineal, es decir, se concatenan diferentes operaciones. Se puede comprobar, ya que el efecto de sumar varias operaciones de regresión lineal (sumar varias líneas rectas), equivale a realizar una sola operación, obteniendo como resultado otra línea recta (ver figura 6).

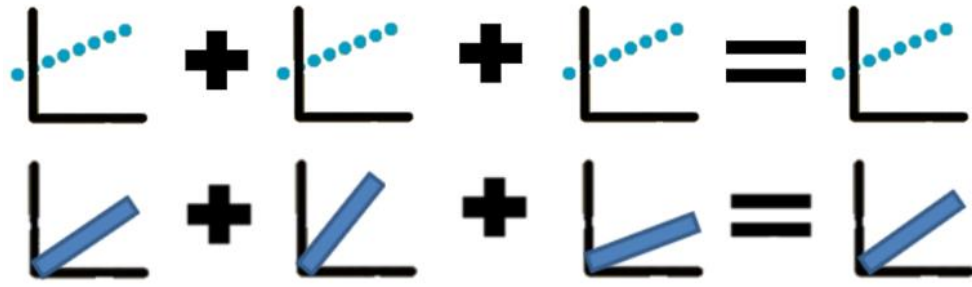


Figura 6. Suma de Operaciones de Regresión Lineal y No Lineal

Para evitar la linealidad a la salida de la red, se requiere de una función no lineal que manipule estas rectas y las distorsione. Por lo que se requiere a una operación ejecutada en la salida de las capas, se trata de las funciones de activación. Con esto se determina si el vector de entradas inicial ha producido un estímulo en la neurona, ocasionando su activación. El propósito de la función de activación es introducir no linealidad en la salida de una neurona y encadenar de forma efectiva la computación de varias de ellas [ 22].

En la figura 7, se pueden observar algunas de las funciones de activación más comunes:

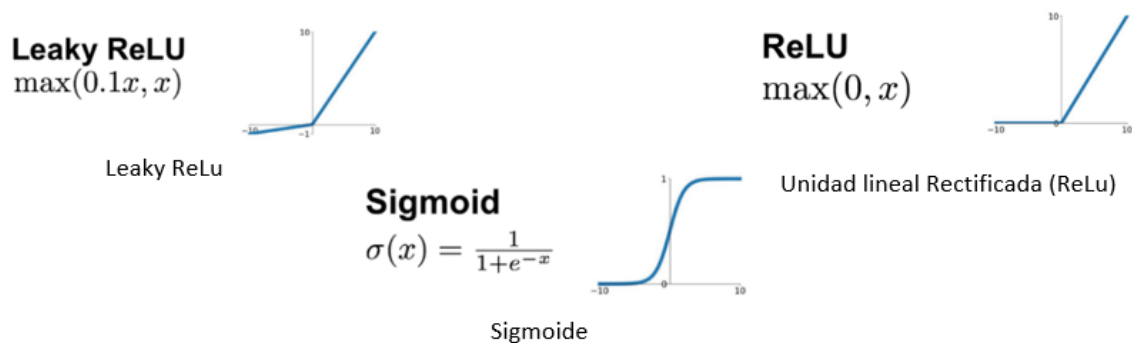


Figura 7. Gráfica Operacional de la Función Leaky ReLU, Sigmoid y ReLU [18].

### 1.6.3 Convolutional Neural Networks (Redes Neuronales Convolucionales)

Se desea aprender a clasificar y detectar imágenes, es necesario aprender a utilizar las capas de redes neuronales convolucionales. Están especialmente diseñadas para el procesamiento de datos de dos dimensiones. Una red neuronal convolucional es un tipo de red neuronal que se usa para procesar imágenes mediante aprendizaje supervisado que procesa sus capas imitando al ojo humano para identificar las características más importantes dentro de las imágenes figura 8 (se analiza cada píxel).

Para ello, las CNN están basadas en operaciones de varias capas ocultas especializadas llamadas convoluciones. Esto quiere decir que las primeras capas pueden detectar patrones sencillos como líneas, bordes, esquinas y curvas. Conforme la red se va haciendo más profunda los patrones ganan complejidad y se van especializando hasta reconocer formas complejas como objetos o animales [23].

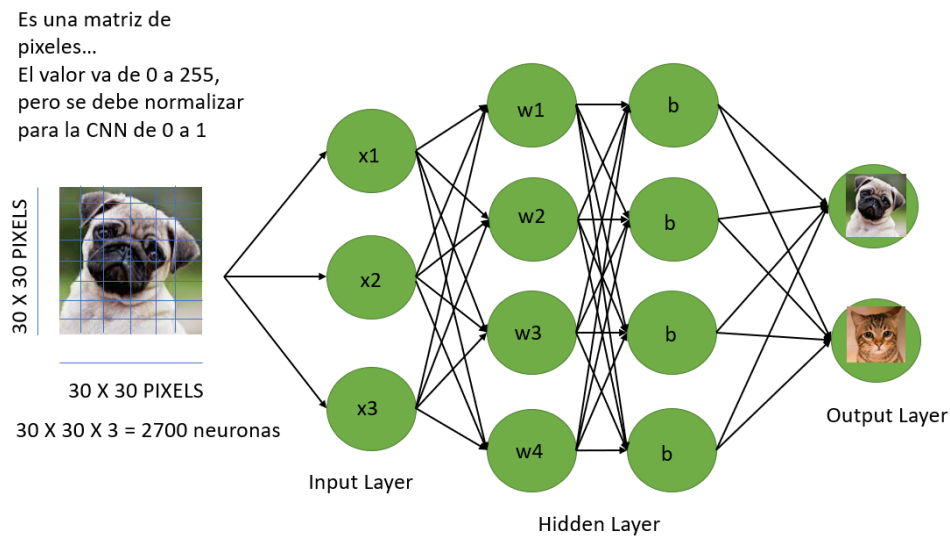


Figura 8. Modelo de Red Convolutiva con Imagen de Entrada

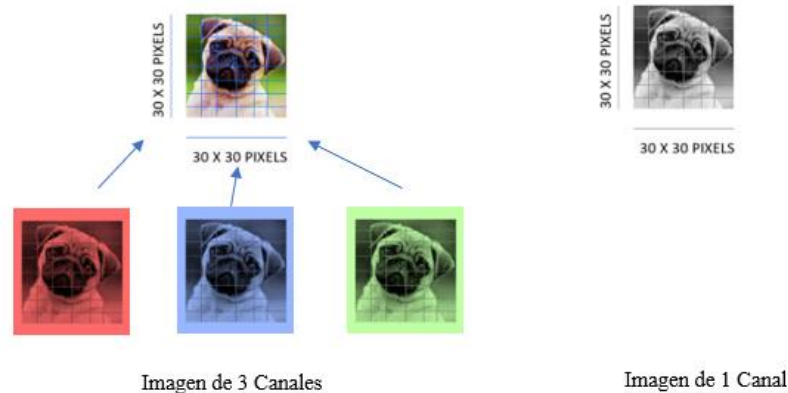
#### 1.6.4 Píxeles

Para que una red neuronal trabaje con una imagen, primero se separa en cada uno de sus píxeles y se coloca en cada neurona de la capa de entrada. Si la imagen es color blanco y negro (se dice que su canal es de 1) se le otorga un valor al píxel, que va del 0 para el negro hasta el 255 para el blanco (figura 9) [24].



Figura 9. Valores de Píxeles de 0 a 255

Cada píxel en la imagen se considera como una entrada, si esta mide 30px X 30px, se tiene un total de 900 entradas o píxeles, cada uno con un valor entre 0 a 255. Esta red trata de analizar cada píxel y encontrar características (partes del objeto) que le permitan predecir la etiqueta o grupo al que pertenece. Cuando la imagen es a color se separa en 3 canales (RGB), cada una tendrá una matriz diferente, una para cada canal, realizando las convoluciones por separado y al final se va píxel por píxel, sumando los resultados y agregando un sesgo (Figura 10). Retomando la operación inicial si nuestra imagen fuera color y mide 30px X 30px, la operación es:  $30 \times 30 \times 3 = 2700$  neuronas de entrada [24].



*Figura 10. Imagen a color conformada por 3 Canales e Imagen Blanco y Negro Conformado por 1 canal*

### 1.6.5 Convoluciones

Las computadoras no aprecian las imágenes como el ser humano, como anteriormente, lo que visualizan es una serie de píxeles representados por datos numéricos. Este proceso lo lleva a cabo realizando una operación matemática (núcleo) que se aplica generalmente sobre matrices. Para detectar características, es imposible si se revisa píxel por píxel de manera individual por lo que recurre a usar datos de los píxeles vecinos para mediante operaciones sencillas calcular un nuevo valor de numérico relacionado con todos los que lo han generado. La convolución da lugar a una imagen agrupada resultante de igual dimensión basada en la imagen original pero completamente distinta [24].

### 1.6.6 Capa Convolutiva

La capa convolutiva es el diferenciador de una red convolutiva a cualquier otra red neuronal. Se encarga de generar los mapas de características mediante filtros por los cuales aprende y detecta características como bordes, manchas y orientaciones. Cada filtro recorre varias regiones de la imagen para generar estos mapas (ver figura 11) [20].

Opera de la siguiente forma:

- Toman una imagen y evalúan píxel por píxel, evaluando también los pixeles de alrededor, colocando una matriz de 3x3 (el tamaño de la matriz puede cambiar).
- Funcionan como detector de características o filtro.
- El filtro detecta bordes o formas específicas.
- Se coloca en la parte superior izquierda de la imagen y se multiplica por el valor. Después, los resultados se suman y se escriben en la matriz de salida.
- El filtro se desliza hacia la derecha para analizar toda la imagen.
- En términos matemáticos, calcula un promedio del valor de todos los pixeles, sumando todos y dividiéndolos entre 9

0	0	0	0	0	0	0
0	60	113	56	139	85	0
0	73	121	54	84	128	0
0	131	99	70	129	127	0
0	80	57	115	69	134	0
0	104	126	123	95	130	0
0	0	0	0	0	0	0

Kernel		
0	-1	0
-1	5	-1
0	-1	0

114	328	-26	470	158
53	266	-61	-30	344
403	116	-47	295	244
108	-135	256	-128	344
314	346	279	153	421

Figura 11. Operación Convolutiva en una imagen de 5x5, con operación de Zero Paddding y un Filtro de 3x3 [20].

A la estructura o matriz que se utiliza para transformar la imagen durante la convolución se le puede denominar: filtro, matriz, kernel o máscara. Dependiendo de la dimensión y los valores del kernel, se obtienen resultados diferentes. Cada máscara representa un filtro distinto que sirve para destacar un patrón concreto dentro de la imagen. Los números dentro de las casillas del kernel, indican que importancia va a dar a los pixeles, según su posición. Al cambiar los valores del núcleo se pueden generar imágenes con desenfoques y filtros (figura 12).



Figura 12. Operación Convolutiva con distintos valores de núcleo, obteniendo nuevas imágenes

Estos son algunos hiper parámetros que controlan la capa convolutiva:

- Capa Pooling

Extraídas las características desde una capa de convolución la siguiente capa es la de Pooling. Reduce la imagen para minimizar los parámetros y el proceso de cálculo, tomando solo 1 píxel. Hay diferentes maneras de realizar esta reducción:

- Max Pooling: Se desliza un filtro de tamaño  $m \times m$  a través de la matriz de entrada, se selecciona el píxel de valor más grande y se pasa a una nueva imagen resultante. (ver figura 13)[20].
- Average Pooling: igual que en la anterior se desliza un filtro, pero el resultado es la media de todos los elementos de la matriz de entrada [20].

Para esta capa también se aplica un stride como en la capa de convolución. El número de canales no se reduce, únicamente el número de dimensiones (ver figura 13).

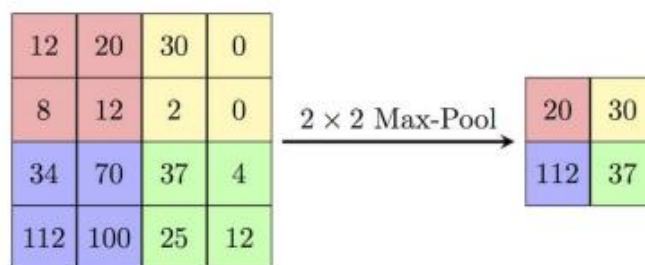


Figura 13. Operación MaxPooling con Stride (2,2) [20].

- Zero Padding (Relleno con ceros)

Las operaciones convolucionales disminuyen el tamaño de la imagen, por lo que se aplica esta función para preservar el tamaño de entrada. Para aumentar la dimensión de la entrada se empieza añadiendo una columna de ceros en la derecha o abajo donde fuera necesario, en caso de necesitar más se rellenaría con ceros en la izquierda o arriba y las características más importantes se preserven al inicio de las primeras convoluciones (ver figura 14) [16].

## 1.7 Detección de Objetos

0	0	0	0	0	0
0	35	19	25	6	0
0	13	22	16	53	0
0	4	3	7	10	0
0	9	8	1	3	0
0	0	0	0	0	0

Figura 14. Ejemplo Operación de Zero Padding en una Matriz de 4x4 [20].

Es una rama de la visión por computadora que identifica y ubica objetos en una imagen o transmisión de video. Esta técnica permite etiquetar los objetos con precisión. La detección de objetos se puede utilizar para determinar y contar objetos en una escena o para seguir su movimiento [27].

En la figura 15 se muestra un esquema del funcionamiento básico de los detectores de objetos utilizando redes neuronales convolucionales.

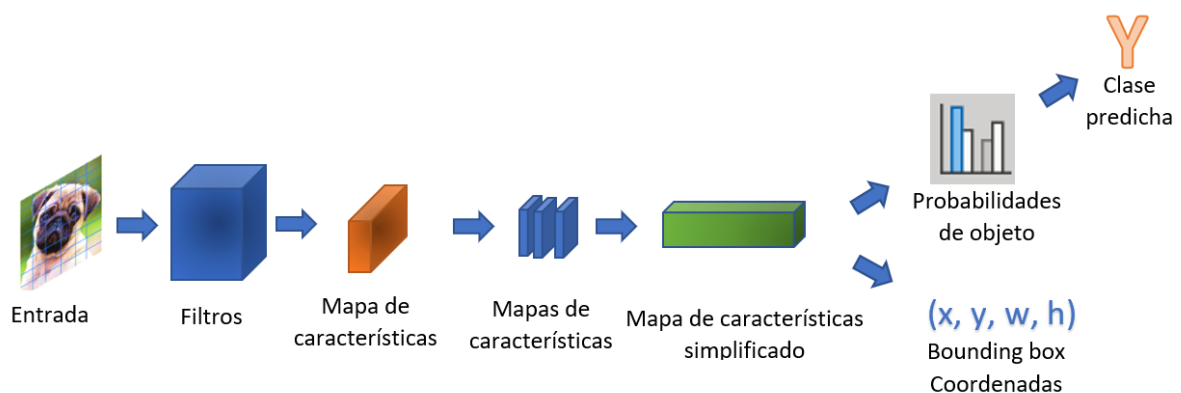


Figura 15. Funcionamiento de un Detector de Objetos.

Se utilizan los píxeles de una imagen como entrada, la cual pasa por unos filtros que genera mapas de características, conforme avance este proceso las imágenes van disminuyendo su tamaño, pero aumentando a la vez el número de mapas de características, obteniendo un mapa de características simplificado, este pasa por una capa conectada que se encarga de asignar probabilidades de que exista cierto objeto. Obteniendo una probabilidad para cada una de las clases que tenemos y recibir una clase predicha. También se agrega otra salida al mapa de características que nos ayudara a encontrar la caja que encapsula el objeto, mejor conocido como bounding box [27].

### **1.7.1 Modelos de detección de Objetos**

Existen una gran variedad de modelos de detección de objetos, que al pasar los años han ido evolucionando y facilitando este campo. El objetivo de los modelos de detección de objetos es buscar la presencia de objetos relevantes en imágenes y clasificarlas en clases. Dependiendo del hardware que se disponga para trabajar y el tipo de objeto a detectar, se necesita un modelo pre-entrenado que en la salida prediga cuadros delimitadores “bounding boxes” que rodean las celdas de la imagen, y le asigne una etiqueta a cada caja [28].

### **1.7.2 COCO (Common Objects in Context)**

Es un conjunto de datos de subtítulos, segmentación y detección de objetos a gran escala. Tiene varias características:

- Segmentación de objetos.
- Reconocimiento en contexto.
- 330K imágenes.
- 1,5 millones de instancias de objetos.
- 80 categorías de objetos.
- 91 categorías de cosas.
- Subtítulos por imagen.
- 250 000 personas con puntos clave.
- Se utiliza principalmente para entrenar y evaluar el rendimiento de los modelos de detección de objetos [29].



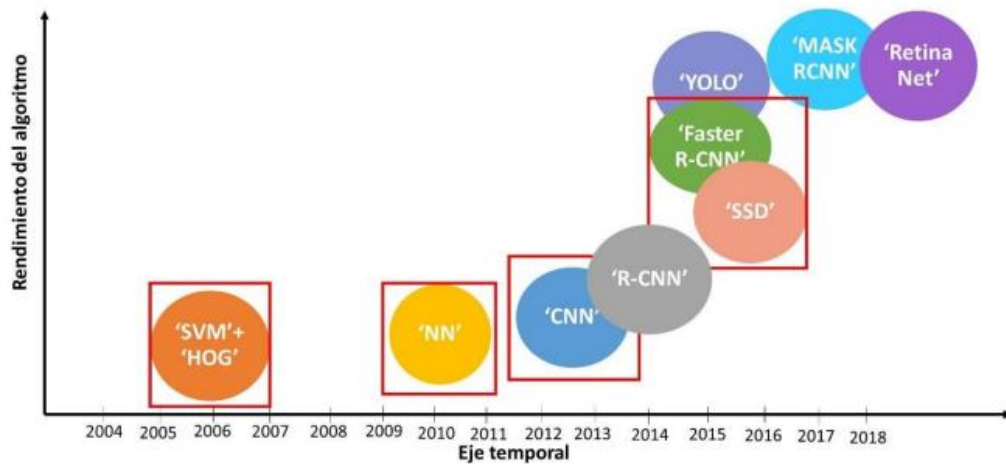


Figura 16. Evolución “Machine Learning” y “Deep Learning” (Algunos de los Algoritmos Actuales) [30].

### 1.7.3 Serie YOLO

YOLO (You Only Look Once) es una familia de modelos que ("PJ Reddie") Joseph Redmon acuñó originalmente con una publicación de 2016. Los modelos YOLO son famosos por su alto rendimiento, pero a la vez increíblemente pequeños, lo que los convierte en candidatos ideales para condiciones en tiempo real y entornos de implementación en hardware como dispositivos móviles, computadoras y drones. YOLO funciona dividiendo las imágenes en una cuadrícula donde cada celda de la cuadrícula identifica un objeto por sí misma (ver figura 16).

## 1.8 YOLOv5

Está escrita en el marco Ultralytics PyTorch, mejorando la accesibilidad para la detección de objetos en tiempo real. Tiene un modelo que simplifica su implementación hasta en dispositivos móviles e integrándose mucho más fácil siendo así casi un 90% más pequeño que YOLOv4.

El proyecto fue iniciado por Glenn Jocher bajo la organización de Ultralytics lanzó YOLOv5 con varias diferencias y mejoras. En el código oficial de lanzamiento de YOLOv5, se proporcionan 4 versiones de la red de detección de objetos: YOLOv5s, YOLOv5m, YOLOv5l y YOLOv5x (figura 17) [31]. (Nelson, 2020).

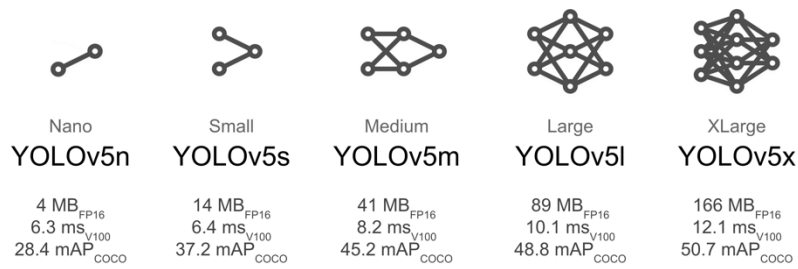


Figura 17. Características de los Modelos Pre Entrenados de YOLOv5 [19].

En la comunidad de investigación, los modelos se comparan con las mismas imágenes del conjunto de datos “Common Object in Context” (COCO) durante 300 épocas [29]. En la figura 18 se compara el rendimiento de los distintos modelos del autor de YOLOv5 e incluyendo EfficientDet un modelo de Google Brain. Se puede apreciar que las cuatro estructuras de red finales tienen sus propias ventajas y desventajas en términos de rendimiento.

Todos los modelos de YOLO están disponibles para descarga desde el repositorio de Ultralytics.

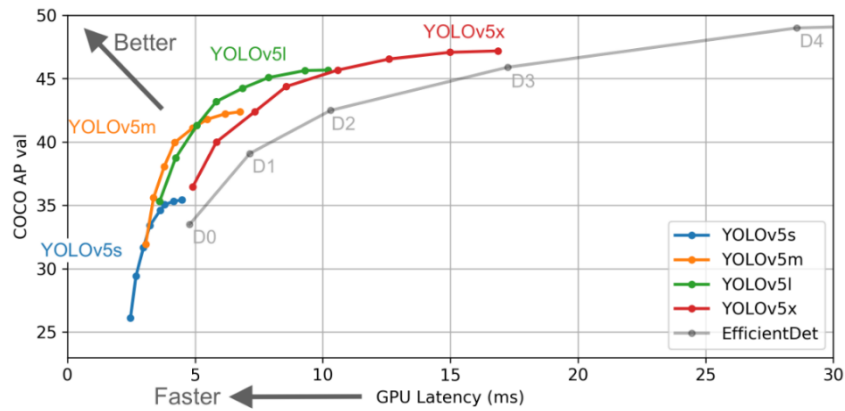


Figura 18. Rendimiento de los Distintos Modelos Evaluados con el Conjunto de Datos de COCO. [32]

## 1.9 Estructura del Modelo

Los detectores de objetos basados en CNN capas o redes troncales YOLOv5 es un detector de 2 partes. Aunque en los últimos años los detectores de objetos que se han desarrollado insertan algunas capas entre el Backbone, que se encarga de extraer las características de una imagen y Head, responsable de predecir las clases [33].

El archivo de la estructura de la red lo proporciona el repositorio de Ultralytics, está en formato “yaml”. Este archivo es nombrado como yolov5s.yaml y es editable.

### 1.10 Aportaciones y Mejoras de Datos

La efectividad del aumento de datos es expandir el conjunto de datos, de modo que el modelo tenga una mayor robustez para las imágenes obtenidas de diferentes entornos. Se han incorporado varias mejoras de métodos que utilizan ampliamente las distorsiones fotométricas y las distorsiones geométricas, anteriormente usadas en versiones de YOLO, como el caso de “Aumento de Mosaico” que se incorporó en YOLOv4. Para YOLOv5, se han aplicado actualizaciones y se incorpora de manera efectiva, incluyendo otras alternativas. (ver figura 19).

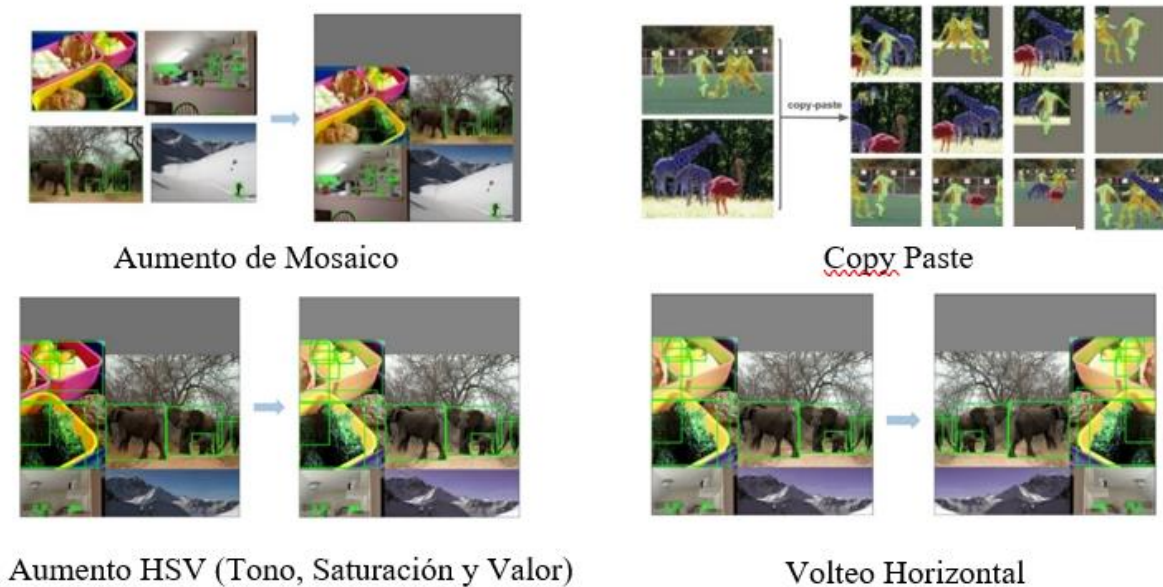


Figura 19 Ejemplos Aumento de Datos YOLOv5 [19]

## 1.11 Conjunto de Datos

Las redes neuronales convolucionales utilizan imágenes como entrada de una base de datos o data set para entrenarse, estas se analizan y se obtienen características que les permiten aprender. Uno de los problemas iniciales es tener una base de datos adecuada para el objeto a detectar, auxiliarnos de algunos repositorios con base de datos personalizados, pero existe el riesgo de tener problemas en el entrenamiento, algunos para su acceso tienen un costo monetario o simplemente son protegidos por derechos de autor.

Existen librerías, como el caso de Scikit-learn que proporciona conjuntos de data sets como “Flores de Iris” o “COCO library”, que contienen conjunto de datos de subtítulos, detección y segmentación de objetos a gran escala [29]. Disponibles para todo público que quiera experimentar con redes neuronales y evaluarlas con sus bases de datos clasificadas y etiquetadas.

### 1.11.1 Conjunto de Datos Personalizado

Como solución, al no contar con un conjunto de datos para reutilizar, se realiza un procedimiento para obtener o generarlo, el cual consiste en la recolección de imágenes para realizar un entrenamiento.

## 1.12 Técnicas de Aumento de datos

Aplicando distintos métodos y herramientas para generar nuevas imágenes a partir de un local conjunto de datos, con diversas transformaciones para la red, se entrene con imágenes distintas y con variaciones. Cambios como: rotaciones, cambio de posición, ampliaciones, etc. Estas son 3 opciones de aumento de datos para complementar el conjunto de datos:



### **1.13 Imágenes de Entrenamiento, Test (prueba) y Validación.**

Generar variantes de las imágenes del conjunto de datos, es importante ya que de ellas la red obtiene características que les permiten aprender, por lo que se recomienda tener un gran número de imágenes, tanto para entrenamiento, prueba y validación. El objetivo de repartir o separar el total de imágenes del conjunto de datos, es para evitar que el modelo generalice y sea capaz de predecir.

No existen reglas para repartir el porcentaje de imágenes para cada categoría, lo importante es que el sistema cuente con datos relevantes y procesables [35].

- Datos de entrenamiento

Con este conjunto de datos el algoritmo aprende, permite obtener los pesos de una red dada y ajusta parámetros.

- Datos de prueba

Este conjunto de datos se utiliza para evaluar la generalización del rendimiento del modelo y permite estimar el desempeño de la red en datos futuros.

- Datos de validación

Permite comparar el desempeño de distintas arquitecturas de red.

### **1.14 Etiquetado**

El programa de aprendizaje automático recibe los datos de entrada y el etiquetado correspondiente. Esto significa que los datos de aprendizaje (imágenes de nuestros objetos a detectar) deben ser etiquetados de antemano por una persona. Algunos de las mejores herramientas de etiquetado de código abierto gratuitos son “LabelImg”, “Roboflow Annotate” y “CVAT” [36].

### **1.15 Entrenamiento**

El entrenamiento consiste en minimizar la función de pérdida. Se configura con sus hiperparámetros que realizara en cada iteración y permiten mejorar el entrenamiento de la red neuronal, aprenderá dado el conjunto de datos y así obtener mejoras durante aprendizaje [30].

- Batch size (Tamaño de lote)

Este es el número de ejemplos que se introducen en la red para que entrene de cada vez. Si el número es pequeño, la red tiene en memoria poca cantidad de datos y entrena más rápido.

Sin embargo, es posible que no aprenda las características y detalles que pueden ser significativos en la predicción. Si es grande, ocurre, al contrario, es posible obtener los mejores casos en el entrenamiento, pero esto puede ser más tardado.

Sus valores pueden variar desde 16, 32, 64 y 128. Es decir, para cada Epoch, toma “n” filas del dataset respectivamente. Se recomienda usar el más grande batch-size que el hardware permita. Los tamaños de lote pequeños producen estadísticas de norma de lote deficientes y deben evitarse [37].

- Epoch (Época)

Es el número de veces que se va a pasar cada ejemplo de entrenamiento por la red.

Con un número alto de épocas, se presenta el riesgo de consumir tiempo y energía entrenándose y puede producirse un “overfitting” al igual que con un número bajo de épocas se puede provocar “underfitting” en la red, lo que provoca que no aprenda lo suficiente [37].

- Iteraciones

Es el número de lotes que se procesan durante el entrenamiento de una red neuronal [24].

## **1.16 Índices de evaluación**

El análisis de datos el cual tiene una gran repercusión en los resultados, ya que hacen de su uso para comparar el rendimiento de los modelos de detección de objetos. Una forma de evaluar los modelos es por medio de imágenes, evaluar el comportamiento del detector, prediciendo y agregando bounding boxes adecuados, pero no asegura su eficacia solo visualizando una imagen y confiar de sus resultados. Ahí es cuando entra el análisis de datos, que permite cuantificar los modelos de detección. Se utilizan precisión, recall y mAP (Precision media media) como indicadores de medición [25]. En la tabla I, se definen sus métricas.

Tabla I. Resultados de Predicciones			
Categoría	Salida	Situación	Conclusión
True Positives (TP)	Detección y clasificación correctas.	El modelo dibujo la caja del tamaño correcto en el objeto correcto.	El entrenamiento del modelo fue el adecuado permitiendo buenos resultados.
False Positives (FP)	Detección y clasificación incorrectas.	El modelo dibujo la caja en un objeto que no debía.	Posiblemente los objetos se parecen entre sí, debido al etiquetado de las clases pudieron confundir al modelo.
False Negatives (FN)	Hay un objeto, pero no detecta ni clasifica	El objeto estaba en la imagen y el modelo no dibujo la caja.	Ocurre cuando hay variantes en la imagen como iluminación, distintos ángulos o estados no se tomaron en cuenta para el entrenamiento del modelo. Este problema se soluciona aumentando el número de datos de entrenamiento.
True Negatives (TN)	No hay objeto que detectar y clasificar	El objeto no estaba en la imagen y el modelo no dibujo la caja.	No había objeto en la imagen y predijo que no existía objeto que detectar.

### 1.16.1 Precisión en Machine Learning

La precisión es una medida que indica que tan precisas son las predicciones. La relación de los casos correctos e incorrectos predichos como positivos (ver figura 20). La precisión es la fracción o porcentaje de las predicciones correctas [36].

La fórmula:

$$Accuracy = TP / (TP + FP)$$

*Ecuación 1: Precisión*

### 1.16.2 Recall

Mide los aspectos positivos, se define como la relación entre los casos positivos identificados y todos los casos positivos reales, que es la suma de los "falsos negativos" y los "verdaderos positivos" [38].

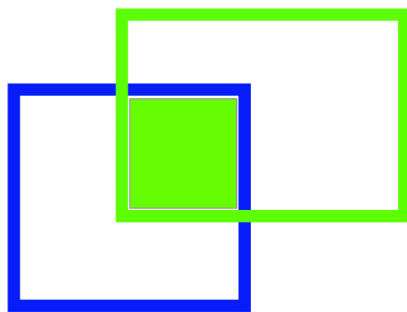
$$Recall = TP / (TP + FN)$$

*Ecuación 2: Recall*

Utilizando la precisión y el recall, se puede evaluar el rendimiento general de nuestro modelo entrenado y funciona como métrica para el rendimiento relativo y comparar con otros modelos.

### 1.16.3 IoU (Intersection over Union)

Los sistemas de detección de objetos hacen predicciones en términos de una caja limitadora y una etiqueta de clase. La Intersección sobre unión (IoU) se encarga de medir y establecer la superposición entre dos límites (figura 20). Se utiliza para medir cuanto se superpone el límite predicho con el cuadro delimitador de la verdad o realidad [38].



*Figura 20 Un boceto de una representación gráfica de la métrica IoU [38]*

- AP (Average precision)

También conocido como la precisión promedio, es la métrica que se utiliza para medir la precisión de los detectores de objetos en todos los umbrales de IoU. Calcula el valor de la precisión promedio para el valor de recuperación de 0 a 1 [26].

- mAP (average precision)

Compara el cuadro delimitador de la verdad (ground-truth) en la imagen con el cuadro detectado (detected box) y devuelve una puntuación. Cuanto mayor sea la puntuación, más preciso será el modelo en sus detecciones [26].

- mAP@ 0.5

Cuando IoU se establece en 0.5, se calcula el AP de todas las imágenes de cada categoría y luego se promedian todas las categorías [26].

- mAP@ 0.5:0.95

Representa el promedio de mAP en diferentes umbrales de IoU (de 0.5 a 0.95 en pasos de 0.05) [26].



## 2. Metodología

En el presente apartado describen los procedimientos de métodos utilizados para el desarrollo y obtención de una CNN. Posteriormente, se implementan los pesos generados de la CNN entrenada en la Raspberry Pi para evaluar su inferencia en tiempo real.

En la figura 21 se presenta un esquema básico de la ejecución del detector de objetos:

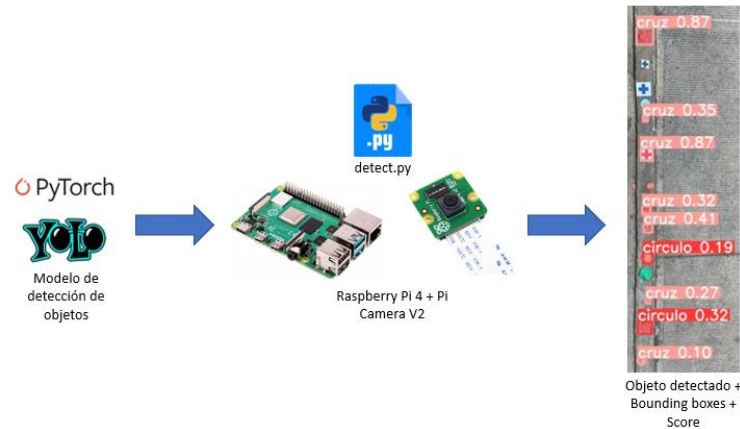


Figura 21. Esquema Básico de la Ejecución del Detector de Objetos.

### 2.1 Obtención del conjunto de datos

Para generar un conjunto de datos personalizado, se realiza el siguiente procedimiento:

- 1 Definir el número de clases: Dos clases de objetos, la etiqueta círculo representa emergencia y la etiqueta cruz, representa rescate. Ambas de color rojo.
- 2 Recolección de fotografías de figuras (cruz y circulo) de hojas de papel color rojo y distintos tamaños, tomadas por una cámara de 12 MP.
- 3 Se complementan las fotografías del conjunto de datos, utilizando técnicas de aumento de datos (ImageDataGenerator, Google Download All Images y Roboflow).
- 4 Las imágenes se agrupan en distintas carpetas con su respectiva etiqueta.
- 5 Utilizando Google Colab se implementan 2 códigos al conjunto de datos (tabla II):

**Tabla II. Códigos implementados**

Código	Descripción
Modificar tamaño y aumento de datos de las imágenes.	<ul style="list-style-type: none"> <li>• Redimensiona todas las imágenes a un tamaño de 224px x 224 px (se recomienda utilizar múltiplos de 32).</li> <li>• Genera 4 imágenes nuevas con variaciones (giros, rotaciones, zoom y etc) por cada imagen del conjunto de datos.</li> </ul>
Renombrar las imágenes.	<ul style="list-style-type: none"> <li>• Para identificar las imágenes y tener un orden, se enumeran.</li> </ul>

Se generaron en total 1000 imágenes por cada etiqueta de cruz y círculo. Todas enumeradas del 0 al 999 (círculo) y del 1000 a 2000 (cruz) (figura 22).



Figura 22. Muestras de fotografías de Cruz de 224x224 px

La figura 23 resume el procedimiento para generar y recopilar el conjunto de datos:



Figura 23. Procedimiento para generar y recopilar conjunto de datos.

## 2.2 Etiquetado

Una vez obtenido el conjunto de datos personalizados, se realizan anotaciones de cuadros delimitadores en las imágenes. Utilizando la herramienta de Python “LabelImg” se indica el área de interés (ubicación de la figura en la imagen) y etiquetar cada cuadro con la clase de objeto que el detector prediga. Para realizar las anotaciones se realizan los siguientes pasos:

- 1 Utilizando la terminal de Windows se instala “LabelImg” por medio de comandos.
- 2 Se ejecuta escribiendo en la terminal “python labelImg.py”.
- 3 Una vez en su interfaz, lo primero es agregar la ubicación de la carpeta que contiene las fotografías de las figuras, dando clic en “Open Dir”, se mostraran las imágenes listas para ser etiquetas. Se realiza el mismo procedimiento para

seleccionar la carpeta donde almacenará las anotaciones de las coordenadas del área de interés de nuestro objeto, dando clic en “Change Save Dir”.

- 4 LabelImg, trabaja distintos formatos de anotaciones, como “PascalVOC” y “YOLO”. Nota: No importa el formato a trabajar, ya que posteriormente se transformarán las anotaciones para el modelo que se requiere.
- 5 Con la herramienta “Create RectBox”, lo siguiente es hacer clic y soltar el botón izquierdo del ratón seleccionando la región de interés de la imagen.
- 6 Se asigna una etiqueta prescrita (círculo o cruz).
- 7 Se aplican los pasos 5 y 6 en todas las imágenes.
- 8 Al concluir el etiquetado, en la carpeta seleccionada del paso 3, se encontrarán las anotaciones (coordenadas de las figuras en la imagen) en formato xml o txt. Estos archivos contienen datos como: nombre de la imagen, nombre de la clase, ancho y altura de la caja y coordenadas del área de interés. Nota: una imagen puede contener varias etiquetas (ver figura 24).

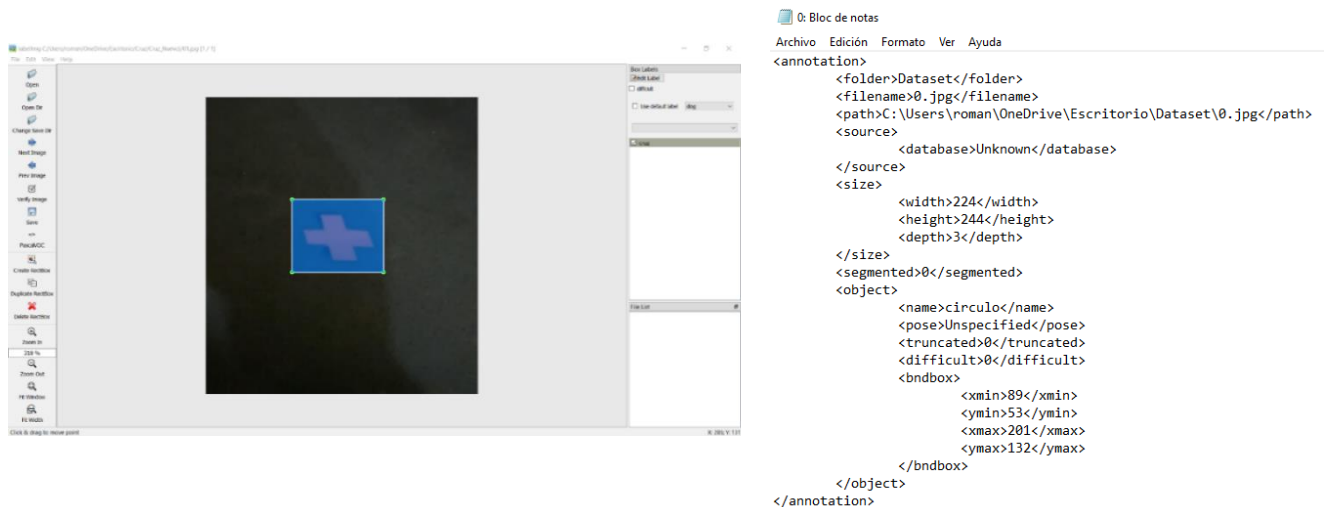


Figura 24. Ventana de la herramienta de LabelImg y el archivo que genera con las anotaciones del área de interés.

## 2.3 Obtención de Snapshot

Implementando la herramienta de Roboflow, el cual proporciona un método fácil para preparar y exportar el conjunto de datos directamente al formato de YOLOv5.

Utilizando una cuenta e ingresando a su página en un navegador. Se accede a un espacio de trabajo público donde se importan el conjunto de imágenes (los datos pueden ser privados con un costo de plan de pago).

1. Se importan las carpetas de las etiquetas y anotaciones de cada clase. Cada imagen debe tener su anotación correspondiente, en caso contrario, Roboflow aporta una

herramienta que permite etiquetar las imágenes desde su interfaz (similar a LabelImg).

2. Administrar el número de imágenes del conjunto de datos: por medio de una barra de desplazamiento, se reparte el número de imágenes de entrenamiento, validación y test. La configuración corresponde con el Split de train con 80%, test con 19.95% y valid con 0.05%. Con los porcentajes antes mencionados, se prioriza el entrenamiento con las imágenes de train.
3. Preprocessing: Utilizando esta herramienta se redimensiona las imágenes a un tamaño de 224 x 224 px a 416 x 416 px, (el tiempo de entrenamiento depende del tamaño de las imágenes, al igual que la potencia de procesamiento del computador con el que se trabaje). Nota: En caso de realizar el entrenamiento en la Raspberry Pi, se recomienda que las imágenes sean de pequeño tamaño, ya que pueden consumir mucha memoria conforme avance el entrenamiento.
4. Aumento de datos: En esta sección permite crear nuevos ejemplos de entrenamiento, para que el modelo aprenda con versiones aumentadas de cada imagen del conjunto de entrenamiento: flips, rotaciones, cambios de Hue, cambiar a escala de grises y entre otras opciones.
5. Genera un conjunto de datos con las configuraciones propuestas anteriormente.

El siguiente diagrama resume la configuración del conjunto de datos (ver figura 25)

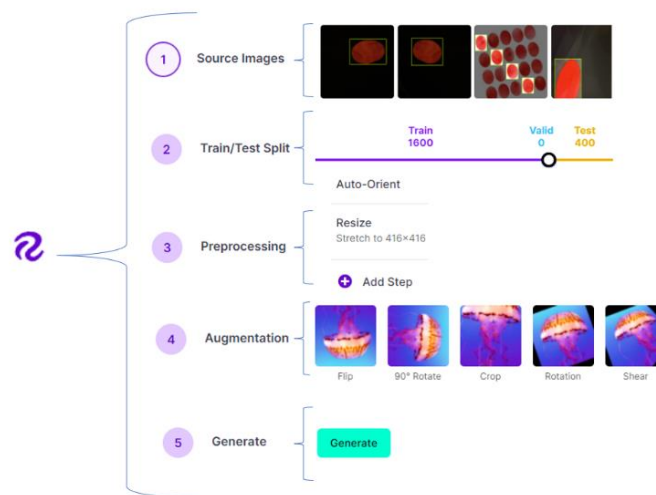


Figura 25. Configuración de nuestro set de datos con Roboflow



## 1 Instalación Dependencias de YOLOv5

- Concluida la configuración del cuaderno, se importa el repositorio YOLOv5 de GitHub escribiendo el siguiente comando (ver figura 28):

```
!git clone https://github.com/ultralytics/yolov5
```

Figura 28 Línea de código para clonar el repositorio de YOLOv5

- Ya instalada la carpeta de YOLOv5, se procede a instalar las dependencias (Incluyendo la librería de Torch) usando el siguiente comando (ver figura 29):

```
#clone YOLOv5 and
!git clone https://github.com/ultralytics/yolov5 # clone repo
%cd yolov5
%pip install -qr requirements.txt # install dependencies
%pip install -q roboflow

import torch
import os
from IPython.display import Image, clear_output # to display images
```

Figura 29. Líneas de código para instalar dependencia de YOLOv5

Los paquetes que se instalan son los siguientes (ver figura 30):

```
1 # pip install -r requirements.txt
2
3 # Base -----
4 matplotlib>=3.2.2
5 numpy>=1.18.5
6 opencv-python>=4.1.1
7 Pillow>=7.1.2
8 PyYAML>=5.3.1
9 requests>=2.23.0
10 scipy>=1.4.1 # Google Colab version
11 torch>=1.7.0
12 torchvision>=0.8.1
13 tqdm>=4.41.0
14
15 # Logging -----
16 tensorboard>=2.4.1
17 # wandb
18
19 # Plotting -----
20 pandas>=1.1.4
21 seaborn>=0.11.0
22
23 # Export -----
24 # coremltools>=4.1 # CoreML export
25 # onnx>=1.9.0 # ONNX export
26 # onnx-simplifier>=0.3.6 # ONNX simplifier
27 # scikit-learn==0.19.2 # CoreML quantization
28 # tensorflow>=2.4.1 # TFLite export
29 # tensorflowjs>=3.9.0 # TF.js export
30 # opencv-dev # OpenVINO export
31
32 # Extras -----
33 # albumentations>=1.0.3
34 # Cython # for pycocotools https://github.com/cocodataset/cocoapi/issues/172
35 # pycocotools>=2.0 # COCO mAP
36 # roboflow
37 thop # FLOPs computation
```

Figura 30. Requirements.txt [19].

## 2 Importar Conjunto de Datos

- Se instala el ambiente de Roboflow y las líneas de código que proporciono. (api\_key, nombre de usuario, proyecto y espacio de trabajo) (figura 31):

```
from roboflow import Roboflow
rf = Roboflow(model_format="yolov5", notebook="ultralytics")

upload and label your dataset, and get an API KEY here: https://app.roboflow.com/?model=yolov5&ref=ultralytics

# set up environment
os.environ["DATASET_DIRECTORY"] = "/content/datasets"

#after following the link above, recieve python code with these fields filled in
from roboflow import Roboflow

rf = Roboflow(api_key="HkMZYEI6MCJXDmQaw19g")
project = rf.workspace("roman-velasquez").project("reconocimiento-figuras")
dataset = project.version(5).download("yolov5")
```

Figura 31. Líneas de código del paquete PIP del conjunto de datos

- Al actualizar la ventana de archivos, se aditan 3 carpetas: “test”, “train” y “valid”. Estas carpetas contienen imágenes con1 sus respectivas etiquetas (figura 32).

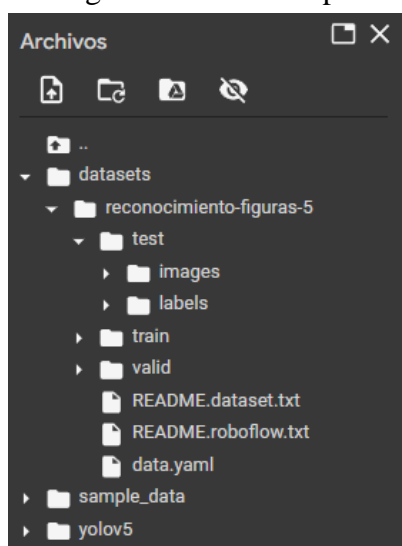
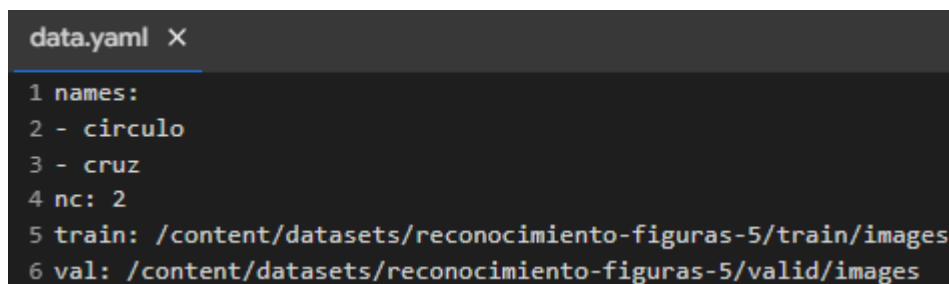


Figura 32. Actualización de las Carpetas al momento

Durante el proceso de exportación del conjunto de datos al cuaderno, se crea automáticamente un archivo llamado “data.yaml” contiene información sobre la ubicación de imágenes y etiquetas (ver figura 33).



```
data.yaml X
1 names:
2 - circulo
3 - cruz
4 nc: 2
5 train: /content/datasets/reconocimiento-figuras-5/train/images
6 val: /content/datasets/reconocimiento-figuras-5/valid/images
```

Figura 33. Archivo generado “data.yaml”

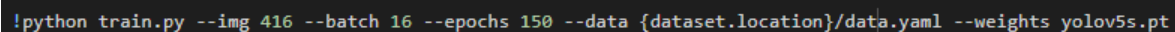
#### 4 Configuración de Parámetros de Entrenamiento

Aquí se ingresan los parámetros con los que se realizara el entrenamiento.

- **Img:** Se ingresa la resolución de las imágenes del conjunto de datos (416 px)
- **Batch o lote:** Número de muestras que se cargan en un lote durante el entrenamiento. Se utiliza el valor más grande que el hardware permita (16 y 32).
- **Epochs o épocas:** Define el número de épocas de entrenamiento (50, 100 y 150).
- **Data:** Ubicación del archivo YAML del conjunto de datos ({dataset.location}/data.yaml)
- **Weights o pesos:** Ruta de los pesos para comenzar a transferir el aprendizaje. Toma el punto de control pre-entrenado de COCO genérico (yolov5s.pt).

#### 5 Entrenamiento

Se agregan los parámetros a la línea de código de entrenamiento (ver figura 34).



```
!python train.py --img 416 --batch 16 --epochs 150 --data {dataset.location}/data.yaml --weights yolov5s.pt
```

Figura 34. Línea de código “train.py”

Se ejecuta el script de entrenamiento tres veces con distintos argumentos para evaluar y comparar su rendimiento.

El entrenamiento tomará tiempo dependiendo del hardware. Por medio de la terminal, imprimirá las métricas de mAP que consigue por cada época (ver figura 35).



```

Epoch 0/49  gpu_mem 5.38G box 0.09742 obj 0.0234 cls 0.02528 labels 171 img_size 416: 100% 25/25 [00:51<00:00, 2.07s/it]
            Class Images Labels P R mAP@.5 mAP@.5:.95: 100% 1/1 [00:01<00:00, 1.49s/it]
            all 100 162 0.385 0.42 0.254 0.0851

Epoch 1/49  gpu_mem 6.02G box 0.06618 obj 0.02224 cls 0.009503 labels 163 img_size 416: 100% 25/25 [00:49<00:00, 1.96s/it]
            Class Images Labels P R mAP@.5 mAP@.5:.95: 100% 1/1 [00:01<00:00, 1.31s/it]
            all 100 162 0.628 0.438 0.469 0.16

Epoch 2/49  gpu_mem 6.02G box 0.05714 obj 0.01861 cls 0.004376 labels 204 img_size 416: 100% 25/25 [00:48<00:00, 1.96s/it]
            Class Images Labels P R mAP@.5 mAP@.5:.95: 100% 1/1 [00:01<00:00, 1.33s/it]
            all 100 162 0.626 0.519 0.526 0.168

Epoch 3/49  gpu_mem 6.02G box 0.04866 obj 0.01696 cls 0.003192 labels 145 img_size 416: 100% 25/25 [00:48<00:00, 1.96s/it]
            Class Images Labels P R mAP@.5 mAP@.5:.95: 100% 1/1 [00:01<00:00, 1.29s/it]
            all 100 162 0.35 0.358 0.325 0.12

Epoch 4/49  gpu_mem 6.02G box 0.04213 obj 0.01623 cls 0.002686 labels 181 img_size 416: 100% 25/25 [00:48<00:00, 1.96s/it]
            Class Images Labels P R mAP@.5 mAP@.5:.95: 100% 1/1 [00:01<00:00, 1.25s/it]
            all 100 162 0.69 0.66 0.712 0.396

Epoch 5/49  gpu_mem 6.02G box 0.03796 obj 0.01535 cls 0.002478 labels 173 img_size 416: 100% 25/25 [00:49<00:00, 1.96s/it]
            Class Images Labels P R mAP@.5 mAP@.5:.95: 100% 1/1 [00:01<00:00, 1.24s/it]
            all 100 162 0.594 0.66 0.576 0.328

Epoch 49/49  gpu_mem 6.02G box 0.02012 obj 0.01124 cls 0.0001453 labels 161 img_size 416: 100% 25/25 [00:48<00:00, 1.93s/it]
            Class Images Labels P R mAP@.5 mAP@.5:.95: 100% 1/1 [00:01<00:00, 1.20s/it]
            all 100 162 0.844 0.771 0.89 0.708

50 epochs completed in 0.701 hours.
Optimizer stripped from runs/train/exp/weights/last.pt, 14.3MB
Optimizer stripped from runs/train/exp/weights/best.pt, 14.3MB

Validating runs/train/exp/weights/best.pt...
Fusing layers...
Model summary: 213 layers, 7015519 parameters, 0 gradients, 15.8 GFLOPs
            Class Images Labels P R mAP@.5 mAP@.5:.95: 100% 1/1 [00:01<00:00, 1.48s/it]
            all 100 162 0.847 0.753 0.887 0.712
            circulo 100 162 0.847 0.753 0.887 0.712

Results saved to runs/train/exp

```

Figura 35. Entrenamiento del detector YOLOv5s personalizado

## 6 Evaluación del rendimiento del detector

Ya finalizado el entrenamiento, utilizando la herramienta de Tensorboard, desplegamos las gráficas de las métricas de validación (ver figuras 36 y 37).

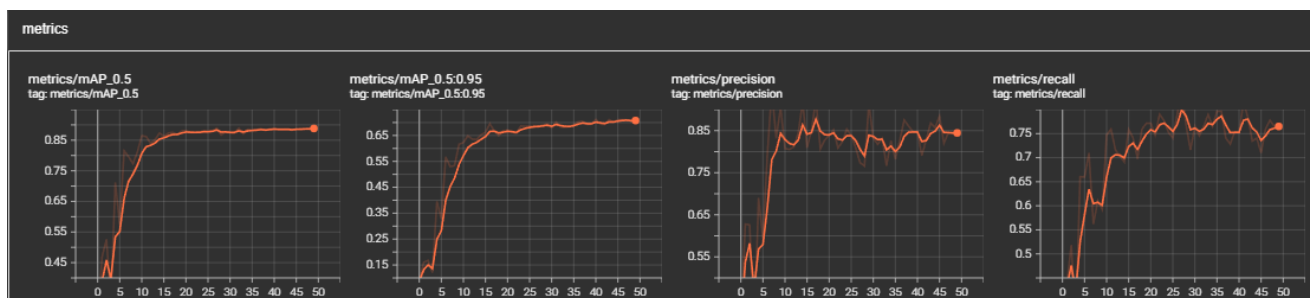


Figura 37. Gráficas de las métricas del entrenamiento

```

# Start tensorboard
# Launch after you have started training
# logs save in the folder "runs"
%load_ext tensorboard
%tensorboard --logdir runs

```

Figura 36. Líneas de código para invocar la herramienta de Tensorboard

Tensorboard permite descargar las gráficas en formato png (ver figura 38):

Indica las funciones de perdida para entrenamiento (train) y validación (val):

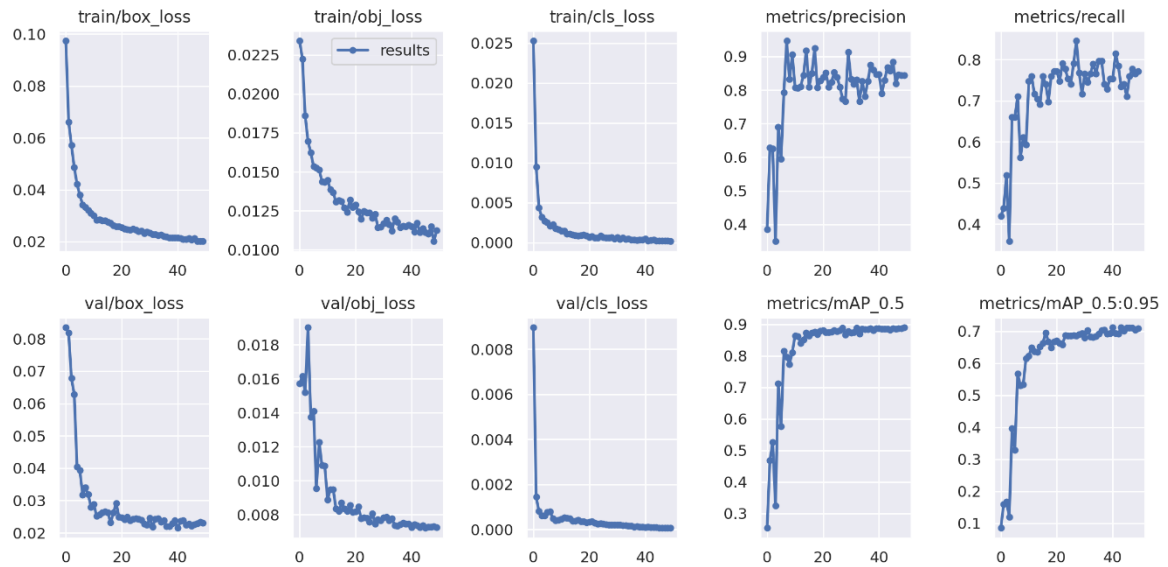


Figura 38. Resultados trazados de Tensorboard del conjunto de datos YOLOv5

- **Box\_loss:** Pérdida debida a una predicción de caja que no cubre exactamente un objeto.
- **Obj\_loss:** Pérdida debida a una predicción errónea de caja-objeto IoU
- **Classification (cls\_loss):** Pérdida debido a desviaciones de la predicción de '1' para las clases correctas y '0' para todas las demás clases del objeto en esa casilla.

Información sobre las métricas de precisión y recall:

- **Precision:** Es el porcentaje de las veces que se realizó una predicción correcta (TP/TP+FP).
- **Recall:** Es el porcentaje que cuantifica la bondad de una predicción contando los falsos negativos (TP/TP + FN).

## 7 Inferencia YOLOv5

Una manera de evaluar la red es por medio de imágenes nuevas, archivos de video, videos de YouTube, transmisión HTTP y también el puerto de cámara. Utilizando el script de “detect.py” (ver figura 39) contiene el script que ejecuta la inferencia. Se creará una nueva carpeta que contendrá los resultados en archivos, el nombre de la carpeta es “runs/detect”.

```
!python detect.py --weights runs/train/exp/weights/best.pt --img 416 --conf 0.1 --source {dataset.location}/test/images
```

Figura 39. Línea de código para usar el script de “detect.py”

Tanto en el computador y en la Raspberry Pi, se pueden realizar el proceso de inferencia. Después de ejecutar el script, se imprimirá por la terminal los resultados con los objetos detectados como se muestra en la figura 40.

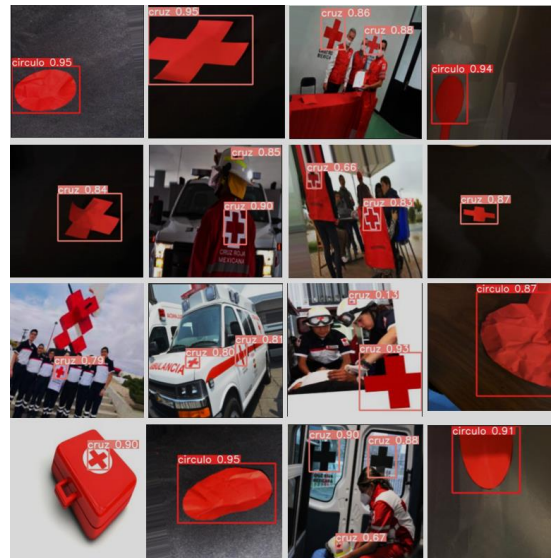


Figura 40. Inferencia con los pesos calculados de YOLOv5s con en Imágenes de train.

## 8 Exportar ponderaciones YOLOv5

A través del cuaderno de Google Colab, exportamos los pesos entrenados (ver figura 41). Al finalizar el entrenamiento el código genera 2 archivos de distintos pesos: “Last.pt” (contiene los pesos de la última época) y los mejores pesos (la red considera estos como los mejores pesos obtenidos durante el entrenamiento).





```
#export your model's weights for future use
from google.colab import files
files.download('./runs/train/exp/weights/best.pt')
```

Figura 41. Línea de código para exportar el modelo “best.pt”

## 2.5 Set Up de Hardware

Previo a los resultados se presenta el esquema del hardware utilizado. La Raspberry Pi fungió como computadora, (Single Board Computer). En la Tabla III se describen los componentes que se emplearon y sus características para el set up inicial.

Tabla III. Set Up de Hardware

Nombre del componente	Imagen	Información	Especificaciones técnicas
Vilros - Kit completo Raspberry Pi 4 con caja transparente enfriada por ventilador		Modelo más reciente de Raspberry Pi 4 ofrece incrementos innovadores en la velocidad del procesador, rendimiento multimedia, conectividad, memoria y mucho más. El rendimiento de escritorio de este modelo es comparable a los sistemas de PC x86 de nivel básico. [39]	<ul style="list-style-type: none"> <li>Procesador: Broadcom BCM2711, quad-core Cortex-A72 (ARM v8) 64-bit SoC @ 1.5GHz</li> <li>Memoria: 8GB de RAM LPDDR4</li> <li>Conectividad: 4 puertos USB (2.4 GHz y 5.0 GHz), internet, ethernet y Bluetooth 5.0</li> <li>Cuenta con 40 pines de GPIO</li> <li>2 micro puertos HDMI, 1 puerto de cámara MIPI CSI, 1 puerto para módulo de pantalla y puerto para audio stereo.</li> <li>Ranura para tarjeta Micro SD para cargar el sistema operativo y almacenamiento de datos.</li> <li>Incluye un ventilador preinstalado listo para conectar en la funda y 4 disipadores de calor.</li> </ul>
Raspberry Pi Módulo de cámara V2 – 8 megapíxeles, 1080p		La Raspberry Pi Camera v2 es un sensor de imagen de 8 megapíxeles Sony IMX219 de alta calidad diseñado a medida para Raspberry Pi, con un objetivo de enfoque fijo. Se conecta a la RB Pi por medio de la interfaz CSI. Se conecta por medio de cable de cinta para cámara [40].	<ul style="list-style-type: none"> <li>Marca: Raspberry Pi</li> <li>Resolución de captura de video: 1080p</li> <li>Fuente de alimentación: 2ª</li> </ul>
Akozon PL2303HX RS232 Convertidor USB TTL Convertidor actualizado USB, Adaptador de Serie COM/TTL Cable de Descarga STC		Convertidor USB actualizado es un cable adaptador COM/TTL para descarga STC. Permite conectarnos en la computadora y con un programa a la terminal a 115200 baudios.	<ul style="list-style-type: none"> <li>Tipo de conector: USB Tipo A</li> <li>Marca: Akozon</li> <li>Definición de Cables: <ul style="list-style-type: none"> <li>Rojo +5 V</li> <li>Negro: GND</li> <li>Verde: TXD</li> <li>Blanco: RXD</li> </ul> </li> </ul>
Samsung 32GB 95MB/s (U1) MicroSD EVO Select Memory Card with Adapter (MB-ME32GA/AM)		Tarjeta de memoria para almacenar el sistema operativo que usaremos en la Raspberry Pi.	<ul style="list-style-type: none"> <li>Marca: SAMSUNG</li> <li>Tipo de memoria: Micro SDHC</li> <li>Clase de velocidad de asociación digital segura: Clase 10</li> </ul>

## 2.6 Set Up de Software

La Tabla IV hace una descripción de los entornos, sistema operativo y herramientas de software utilizados como herramientas de aprendizaje.

Tabla IV. Set Up de Software		
Software	Descripción	Funcionamiento
Sistema Operativo Linux	Es un sistema operativo de Linux gratuito y su distribución es Debian 11 (Bullseye), optimizado para el hardware Raspberry Pi y es el SO recomendado para su uso normal. Python3 es el lenguaje que utiliza [41].	Se instala la última versión del sistema operativo basado en Linux Debian "Bullseye" por su accesible instalación en la Raspberry Pi.
LabelImg	Herramienta de anotaciones de imágenes. Las anotaciones se guardan en archivos XML, en formato PASCAL VOC, YOLO y CreateML. Escrita en Python y su interfaz gráfica usa Qt [42].	Se utiliza para etiquetar las imágenes del conjunto de datos.
Google Colab	Es un servicio de cuaderno alojado de Jupyter. Permite a cualquier usuario escribir y ejecutar código de Python utilizando el navegador [43].	Ofrece recursos informáticos como GPU y el código se ejecuta en "la nube". El entrenamiento de la red neuronal se ejecuta en el cuaderno.
Roboflow	Es una herramienta de conversión universal para conjuntos de datos de visión artificial [32].	Transforma el formato de anotación del conjunto de datos personalizados a cualquier otro formato de anotación (YOLOv5).

## 2.7 Librerías-Frameworks

Se utiliza el lenguaje de programación python3, uno de los lenguajes prácticos para IA y el uso de una serie de librerías que permiten el uso de frameworks como pytorch. La Tabla V contiene una breve descripción de los paquetes que recomienda el desarrollador de YOLOv5 (tabla V).

Tabla V. Librerías-Frameworks		
Librería	Descripción	Función
Matplotlib	Es una biblioteca para generar gráficos a partir de una lista de datos [44].	Permite visualizar el desarrollo del entrenamiento mediante graficas.
Numpy	Este paquete trabaja con arreglos numéricos y estructuras de datos que proporciona un objeto de matriz multidimensional [45].	Realiza las operaciones matriciales obtenidas de las convoluciones
Opencv	Es una librería de código abierto dedicada para manipulación de imágenes y aplicaciones de visión por computadora [46].	Proporciona líneas de código para acceder a la cámara de la Raspberry Pi.
Pillow	Permite tener compatibilidad con formatos de archivo de imágenes, soporte para abrir, manipular, procesamiento y guardar formatos de archivo de imagen diferentes [47].	Se utiliza para la manipulación de archivo de imágenes.
PyYAML	Es un framework que proporciona utilidades para analizar y manipular archivos YAML (Yet Another Markup Language) [48].	Esta librería incorpora los archivos generados en el etiquetado del conjunto de datos.
Scipy	Esta librería realiza cálculos avanzados en Python: optimización, álgebra lineal e integración de la red neuronal [45].	Complementa la librería de Numpy para realizar operaciones avanzadas.
PyTorch	Es una biblioteca de "Deep learning framework" que trabaja con tensores optimizada para el aprendizaje profundo mediante GPU y CPU, utiliza el cálculo de tensor (como NumPy) con fuerte aceleración de GPU y trabaja con redes neuronales profundas [49].	YOLOv5 está construido en este ecosistema. Realiza las tareas de aprendizaje automático.

TensorBoard	Esta librería permite visualizar a las métricas de rendimiento en cualquier momento después de entrenar un modelo, como: métricas, pérdida, predicciones de validación a lo largo de las imágenes y precisión promedio (mAP) [50].	Se visualizan las métricas del modelo entrenado por medio de graficas.
-------------	--	--

### 3. Pruebas y resultados obtenidos

De acuerdo con la tabla VI, se realizaron 3 entrenamientos con distintos parámetros para analizar las gráficas y determinar si existe mejora en las métricas.

**Tabla VI. Configuración de distintos entrenamientos**

Entrenamiento	Tamaño de imagen de entrada (px)	Epochs	Batch Size	Tiempo de entrenamiento (Horas)
1	416	150	32	2.375
2	416	100	32	1.461
3	416	50	64	0.701

La librería de YOLOv5 recomienda empezar el entrenamiento con 300 épocas, para buscar el sobre ajuste, obtener su gráfica e interceptar el número de época en la cual sucede y a partir de ahí comenzar a realizar los entrenamientos necesarios y descartar un elevado número de épocas. Un entrenamiento no tan estricto puede empezar a entrenar con 150 épocas y 32 de Batch. El hardware va a determinar los entrenamientos, y se verá reflejado en una tabla el tiempo que le tomará al modelo realizar el proceso. Es importante que, al momento de la ejecución, se mantenga el usuario al pendiente del cuaderno de Google Colab debido a que, si no detecta algún cambio o interacción, puede pausar el modelo perdiendo todo el progreso. Cheng et al., 2022, mostraron las curvas de precisión de entrenamiento y precisión de validación de diferentes métodos. Encontrando que uno de sus métodos converge más rápido que otros métodos (100% después de unas 30 épocas y luego se estabiliza). Esto significa que el algoritmo propuesto consume menos recursos informáticos y tiene una convergencia rápida para capturar la capacidad de reconocimiento.

En la tabla VII se muestra el rendimiento de los 3 entrenamientos para analizar con detalle sus métricas:



**Tabla VII. Metrics/train/val de los 3 entrenamientos**

Entrenamiento	Tabla metrics/train/val				
1	<p>Metrics for training 1 (100 epochs):</p> <ul style="list-style-type: none"> <li><b>train/box_loss:</b> Decreases from ~0.08 to ~0.02.</li> <li><b>train/obj_loss:</b> Decreases from ~0.0225 to ~0.0100.</li> <li><b>train/cls_loss:</b> Decreases from ~0.020 to ~0.000.</li> <li><b>metrics/precision:</b> Fluctuates between 0.6 and 0.9.</li> <li><b>metrics/recall:</b> Fluctuates between 0.5 and 0.8.</li> <li><b>val/box_loss:</b> Decreases from ~0.05 to ~0.02.</li> <li><b>val/obj_loss:</b> Decreases from ~0.012 to ~0.006.</li> <li><b>val/cls_loss:</b> Decreases from ~0.003 to ~0.000.</li> <li><b>metrics/mAP_0.5:</b> Increases from ~0.4 to ~0.9.</li> <li><b>metrics/mAP_0.5:0.95:</b> Increases from ~0.2 to ~0.7.</li> </ul>				
2	<p>Metrics for training 2 (100 epochs):</p> <ul style="list-style-type: none"> <li><b>train/box_loss:</b> Decreases from ~0.08 to ~0.02.</li> <li><b>train/obj_loss:</b> Decreases from ~0.0225 to ~0.0100.</li> <li><b>train/cls_loss:</b> Decreases from ~0.020 to ~0.000.</li> <li><b>metrics/precision:</b> Fluctuates between 0.5 and 0.9.</li> <li><b>metrics/recall:</b> Fluctuates between 0.4 and 0.8.</li> <li><b>val/box_loss:</b> Decreases from ~0.07 to ~0.02.</li> <li><b>val/obj_loss:</b> Decreases from ~0.016 to ~0.008.</li> <li><b>val/cls_loss:</b> Decreases from ~0.004 to ~0.000.</li> <li><b>metrics/mAP_0.5:</b> Increases from ~0.4 to ~0.9.</li> <li><b>metrics/mAP_0.5:0.95:</b> Increases from ~0.2 to ~0.7.</li> </ul>				
3	<p>Metrics for training 3 (40 epochs):</p> <ul style="list-style-type: none"> <li><b>train/box_loss:</b> Decreases from ~0.10 to ~0.02.</li> <li><b>train/obj_loss:</b> Decreases from ~0.0225 to ~0.0100.</li> <li><b>train/cls_loss:</b> Decreases from ~0.025 to ~0.000.</li> <li><b>metrics/precision:</b> Fluctuates between 0.4 and 0.9.</li> <li><b>metrics/recall:</b> Fluctuates between 0.4 and 0.8.</li> <li><b>val/box_loss:</b> Decreases from ~0.08 to ~0.02.</li> <li><b>val/obj_loss:</b> Decreases from ~0.018 to ~0.008.</li> <li><b>val/cls_loss:</b> Decreases from ~0.008 to ~0.000.</li> <li><b>metrics/mAP_0.5:</b> Increases from ~0.3 to ~0.9.</li> <li><b>metrics/mAP_0.5:0.95:</b> Increases from ~0.1 to ~0.7.</li> </ul>				

Utilizando la herramienta TensorBoard, se puede desplazar el mouse en las gráficas para obtener los valores de las métricas en cierto punto (ver figura 42).

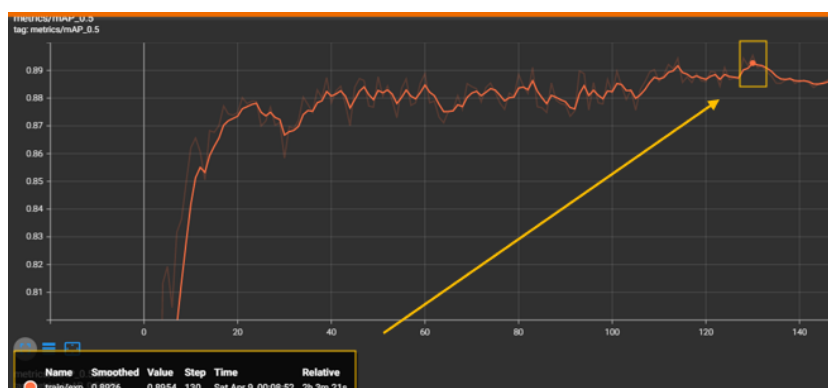


Figura 42. Gráfica de la métrica de Map 0.5

Utilizando esta herramienta se puede buscar en las gráficas de los tres entrenamientos las métricas más destacables. En la Tabla VIII, se muestran capturas de pantalla de los mejores valores obtenidos durante el entrenamiento, además muestra el tiempo y época en el que ocurrió. La mejor métrica es la que se acerca más al valor de uno.

Tabla VIII. Mejores Métricas De los 3 Entrenamientos

Entrenamiento	metrics/mAP_0.5	metrics/mAP_0.5:0.95	metrics/precision	metrics/recall
1	Value Step Time Relative 0.8954 130 Sat Apr 9, 00:08:52 2h 3m 21s	Value Step Time Relative 0.7178 132 Sat Apr 9, 00:10:47 2h 5m 15s	Value Step Time Relative 0.9573 149 Sat Apr 9, 00:27:01 2h 21m 30s	Value Step Time Relative 0.8148 132 Sat Apr 9, 00:10:47 2h 5m 15s
2	Value Step Time Relative 0.8954 94 Sat Apr 9, 15:50:34 1h 22m 18s	Value Step Time Relative 0.7073 91 Sat Apr 9, 15:47:56 1h 19m 40s	Value Step Time Relative 0.9465 15 Sat Apr 9, 14:41:29 13m 13s	Value Step Time Relative 0.8058 84 Sat Apr 9, 15:41:49 1h 13m 33s
3	Value Step Time Relative 0.8904 49 Sat Apr 9, 16:58:04 41m 10s	Value Step Time Relative 0.7085 49 Sat Apr 9, 16:58:04 41m 10s	Value Step Time Relative 0.925 17 Sat Apr 9, 16:31:18 14m 23s	Value Step Time Relative 0.8455 27 Sat Apr 9, 16:39:43 22m 49s

Por lo tanto, se puede establecer que:

- 1 **metrics/mAP\_0.5:** El valor máximo de métricas de “mAP 0.5” lo obtuvieron los entrenamientos 1 y 2 tiendo el mismo valor de 0.8954 y el entrenamiento 3 con un valor muy cercano de 0.8904. Tomando en cuenta el tiempo, el entrenamiento 3, obtiene su valor al finalizarlo, acercándose al mismo valor que los demás entrenamientos en menos tiempo y épocas.
- 2 **metrics/mAP\_0.5:0.95:** En cuanto métricas de “mAP 0.5:0.95”, continúa destacando el entrenamiento 3 (0.7085), comparando con el entrenamiento 1 (0.7178) quien obtuvo el máximo valor, pero lo hace casi con el triple de épocas y tiempo.
- 3 **metrics/precisión:** El valor de precisión al inicio tiene muchas variaciones, con valores mínimos de 0.7 y máximos de 0.9. Se ajusta conforme avance el



entrenamiento, hasta tener sus valores mínimos a partir de 0.8 de precisión. Se obtienen máximos resultados de 0.9 para antes de la época 20 (entrenamiento 2 y 3). El entrenamiento 1 obtuvo el mayor valor de precisión (0.9573), pero sus límites no logran estabilizarse, teniendo los mismos mínimos y máximos.

- 4 metrics/recall:** Ninguno de los 3 entrenamientos logra un valor de 9 y sus gráficas varían valores desde 0.7 a 0.85. El entrenamiento 3 (0.8455) logra en menos tiempo el máximo valor de esta métrica, pero también tiende a tener valores mínimos en 0.7.

Shi et al., 2022, realizó un método de reconocimiento de desechos de desgaste basado en la red YOLOv5s. Para este estudio se utilizaron 800 imágenes y 23 tipos de desechos de desgaste, alrededor de 5000 desechos de desgaste para entrenar y probar el modelo. El valor de mAP general de cada tipo de residuo de desgaste fue de 40.5%, y la tasa de precisión general y de recuperación alcanzaron 0.4 y 0.5, respectivamente. Concordando con nuestro trabajo, el método reconocimiento basado en el modelo YOLOv5s muestra ventajas en términos del efecto de reconocimiento de objetos, velocidad de operación y tamaño de los archivos de peso.

Xikun et al., 2021, realizó la aplicación de un algoritmo de atención con YOLOv4 para la detección de defectos metálicos. Los resultados que obtuvieron fueron con un mAP de 0.86, cumpliendo con los requisitos de precisión mínimos para la detección de defectos en superficies metálicas. En el presente estudio, los 3 entrenamientos obtuvieron mayores valores de mAP (<0.8904).

Los resultados mostrados en la tabla VIII permitió decidir que entrenamiento puede ser más eficaz para la detección de objetos. Al finalizar el script, obtendremos dos pesos. “last.pt” (los pesos del último entrenamiento) y “best.pt” (los mejores pesos con las mejores métricas). Por lo que el script del entrenamiento nos da la facilidad de poder trabajar con los mejores pesos y evitamos buscar en todas las épocas sus mejores métricas.

## Resultados de Inferencia

Después de analizar las métricas de los entrenamientos, se llevó a cabo su evaluación. Utilizando la inferencia, que consiste en ejecutar el script de “detect.py” en imágenes y videos con el archivo “best.pt” (los mejores pesos).

Se tomaron nuevas fotografías en distintos escenarios y distancias, en las que se colocaron figuras de papel de cruces y círculos de distintos tamaños.

Es preferible que al ejecutar el script sea con el mismo tamaño de la imagen con la que se realizó el entrenamiento. En caso de que la imagen de entrada sea de otro tamaño, se debe indicar al script el tamaño de imagen que ejecute. Este valor también va en múltiplos de 32, por lo que, si la imagen de entrada no es múltiplo, la red automáticamente cambiará el tamaño de entrada con el número más cercano.

Se ejecutaron los 3 entrenamientos con las mismas imágenes (Figura 43).



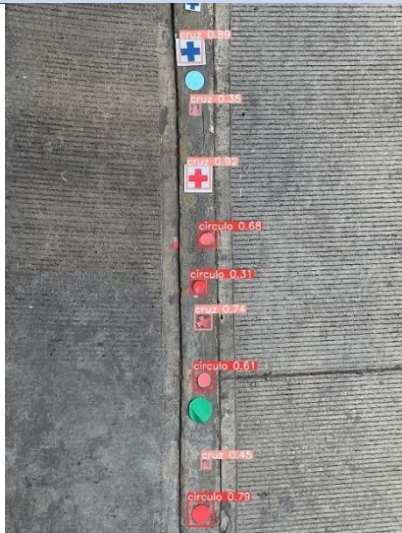
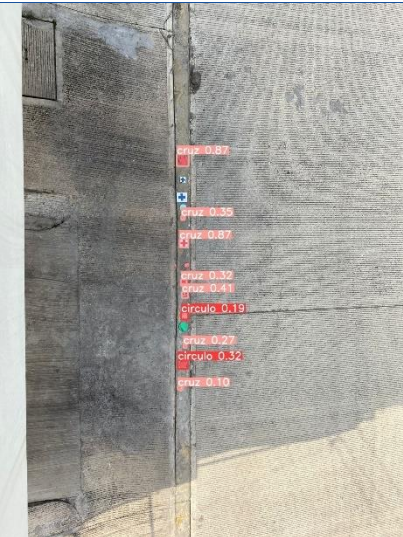

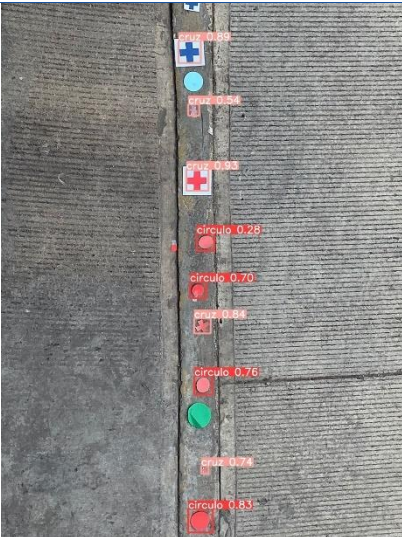
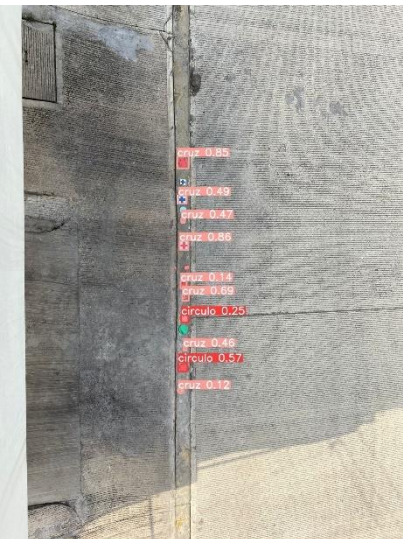

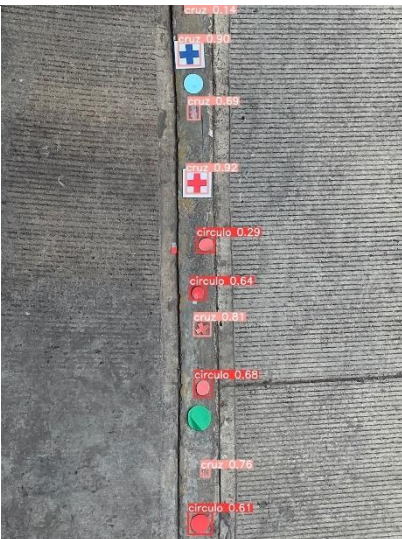
```
!python detect.py --weights /content/peso/best.pt --img 832 --conf 0.1 --source /content/peso/foto_casa_zoom_2.jpeg
```

*Figura 43. Ejecución del script de “detect.py” para imágenes*

- Imagen 1 (Foto\_casa): Foto con una distancia a las figuras, cerca de los 4 metros.
- Imagen 2 (foto\_casa\_zoom): Misma distancia aplicando zoom de 2.15x en la cámara.
- Imagen 3 (foto\_casa\_zoom\_2): Misma distancia aplicando zoom de 2.88x en la cámara.

Los resultados se muestran en la tabla IX.

Tabla IX. Comparación de distintas imágenes con los 3 entrenamientos

Entrenamiento	Imagen 1(foto_casa)	Imagen 2(foto_casa_zoom)	Imagen 3(foto_casa_zoom_2)
1			
2			
3			



En la mayoría de las imágenes se logró con éxito la detección de objetos. Los bounding box funcionaron adecuadamente, son pocos los casos de confusión de los objetos y con variedad de porcentaje en la predicción. Esto se debe a que el parámetro con el que se ejecutó el script “--conf” (confianza) se estableció en 0.1. Por lo que mostrará los objetos que tengan un 10% de predicción en el objeto. Si este parámetro cambia a 0.8, solo detectará o mostrará en las imágenes los objetos que tengan una predicción igual o mayor al 80%. La predicción osciló entre 10% (el valor mínimo de confianza) y el 92%.

El uso de diferentes imágenes para las pruebas garantiza que la evaluación de los modelos no esté sesgada por el uso de conjuntos de prueba y entrenamiento similares [54].

El aumento de datos puede mejorar el tamaño y la calidad de un conjunto de datos de entrenamiento de modo que se puedan construir mejores modelos de redes profundas [55].

Para los 3 entrenamientos fue posible detectar las figuras a una distancia cerca de los 4 metros, con un valor de certeza mayor al 80%. Debido a que se determinó el valor de confianza en 10%, detectó las figuras más pequeñas. Cuando se aplicó zoom a las figuras, el porcentaje de predicción aumentó.

El script “detect.py” también permite aplicar inferencia a videos, por lo que también se evaluaron los 3 entrenamientos en distintos videos pregrabados. Los archivos (imágenes, videos y pesos) se encuentran disponibles en la nube, para realizar un mejor análisis (ver Anexo 1).

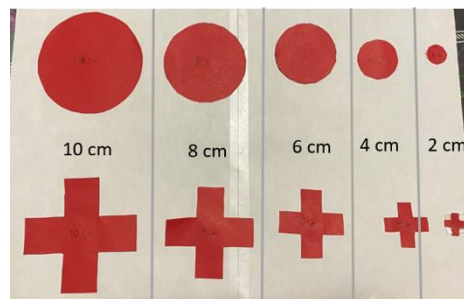


Figura 44. Capturas de pantallas de videos “Video\_tamaños” y “video\_parque” (ver anexo 1)

## Inferencia en la Raspberry Pi

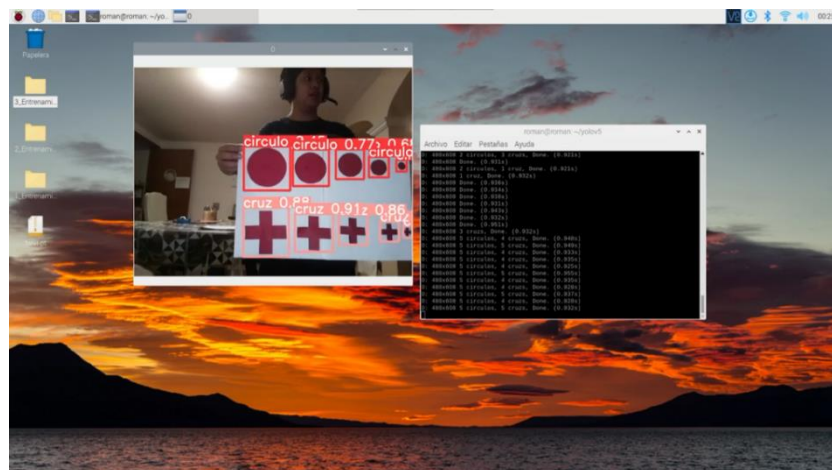
Después de exportar los pesos obtenidos del entrenamiento, se pudo utilizar en la Raspberry Pi, solo se instaló las dependencias de YOLOv5 como se realizó para el entrenamiento en Google Colab y ejecutar el script de “detect.py” con dirección a la cámara del módulo Raspi Cam.

1. Para su evaluación, por medio de una memoria, se añadieron los pesos de los 3 entrenamientos, a la Raspberry Pi y se ejecuta el script.
2. En una hoja blanca, se colocaron las figuras de cruces y círculos de distintos tamaños como en la figura 45.



*Figura 45. Figuras de distintos tamaños para la evaluación de la inferencia en la Raspberry Pi*

3. Se ejecuto el código de detección de objetos por medio de la terminal (ver figura 46).



*Figura 46. Ventanas de la transmisión en tiempo real y la terminal mostrando información sobre los objetos detectados.*

Los resultados de las métricas que se obtuvieron al analizar las gráficas de los 3 entrenamientos permiten visualizar como trabaja el detector en práctica. Se evaluó la inferencia en la Raspberry Pi, detectando objetos en tiempo real con la Raspi Cam.

- **Inferencia los pesos del entrenamiento 1:** Este modelo logró detectar a mayor distancia (4 metros) la figura de cruz más grande (10 cm). Utilizando los bounding box, detecta mejor las figuras de cruz en todas las distancias. Cerca de la cámara (medio metro) ronda entre 90% y 0.69%. 79% fue el máximo que se obtuvo a una distancia de 4 metros.
- **Inferencia los pesos del entrenamiento 2:** Con este modelo se logró detectar todas las figuras estáticas (Al menos por 5 segundos). Las mejores predicciones se lograron con la etiqueta de cruz (0.89% - 0.81%) aun variando la distancia. Para la etiqueta de círculo se dificultaba su detección, solo en distancias cercanas a la cámara (1 metro a ½ metro). Se obtuvieron porcentajes de predicción muy pequeños (0.63% - 0.43%).
- **Inferencia los pesos del entrenamiento 3:** La detección de objetos fue más rápida, detectando con facilidad las figuras de ambas etiquetas. Para los 3 modelos se obtuvieron mejores predicciones para las cruces que para los círculos. A partir de los 3 metros de distancia de la cámara, no logró detectar ninguna figura. Este modelo trabaja mejor a cortas distancias.

Los videos de las 3 inferencias se encuentran en el anexo (2)

## 4. Conclusiones

La evaluación de imágenes por parte de expertos humanos todavía se ve afectada por los problemas de alto costo y subjetividad. Actualmente, la solución más prometedora a los problemas es el aprendizaje automático, mediante redes neuronales convolucionales ha tenido éxito en muchas aplicaciones similares, como el reconocimiento de patrones.

Se definió un procedimiento y metodología para la creación de conjuntos de datos personalizado de imágenes que se utilizaron en el entrenamiento de la red neuronal convolucional.

Se utilizaron distintas técnicas de aumento de datos que permitieron mejorar el porcentaje de mAP ( $<0.8904$ ) y precisión ( $<0.925$ ).

Se categorizaron las imágenes utilizando la herramienta de Python “LabelImg”, con etiquetas personalizables de una manera muy eficiente y accesible.

Se realizaron 3 entrenamientos con distintos parámetros para analizar su comportamiento y determinar su mejora en las métricas. El mejor entrenamiento logro un mAP 0.5 de 89.7% y mAP 0.5:0.95 de 0.712.

Existe un mayor porcentaje de predicción para la etiqueta de cruces debido a las imágenes y anotaciones que se utilizaron para el entrenamiento. Los factores que pueden afectar la predicción son la iluminación, ángulos de la imagen o eventos/situaciones que no se tomaron en cuenta para el entrenamiento del modelo.

La propuesta de la metodología obtuvo resultados alentadores y la factibilidad de llevarlo a cabo con drones.

## 4.1 Recomendaciones

Se obtuvieron resultados satisfactorios, cumpliendo los objetivos planteados inicialmente. Trabajar con CNN implica emplear muchos conceptos y actividades como: aplicar métodos de convoluciones, técnicas de rendimiento, funciones de activación y obtención de métricas. La alternativa que ofrece YOLOv5, en conjunto con PyTorch, le permite al usuario desarrollar su propio detector de objetos con etiquetas personalizadas de una manera accesible al nivel de un experto en CNN. Haciendo posible discutir sobre sus ventajas y desventajas de iniciar con esta propuesta de red neuronal.

### Conjunto de datos personalizados:

- El tamaño de las imágenes de entrada puede variar y se sugiere trabajar con imágenes de grandes píxeles. Su entrenamiento es posible, pero requiere más poder de GPU y RAM. Otra opción es redimensionar el tamaño de las imágenes, YOLOv5s, permite trabajar con imágenes de pequeñas dimensiones, permitiendo continuar con un buen rendimiento. Se recomienda utilizar un margen de 224x224 px.
- La red neuronal fue entrenada con 1000 imágenes para cada etiqueta (se reparten en carpetas de train, valid y test). Las imágenes con las que se entrenó la CNN no eran las indicadas para ser evaluadas con los videos e imágenes con los que realizaron las inferencias, por lo que con dificultad se detectaron los objetos. Este problema se soluciona implementando herramientas de aumentando datos, las cuales proporcionaran imágenes nuevas con diferente iluminación, distintos ángulos o estados que tendrá en cuenta la CNN en el entrenamiento.
- La precisión del modelo puede mejorar, aumentando el número de épocas, pero después de un cierto período puede alcanzar un umbral, por lo que el valor debe determinarse en consecuencia.
- La precisión se reduce con la tasa de “Falsos Positivos” ya que el modelo identifica falsamente los objetos. Esto se debe a que las imágenes que se utilizaron en el entrenamiento no definen con claridad las características del objeto a detectar.

### YOLOv5:

- Utilizar este modelo es intuitivo en la programación. Es la ventaja que ofrecen las redes neuronales pre entrenadas, te evitan comenzar a programar desde cero. Se implemento el modelo pequeño de YOLOv5s, ya que permite implementarlo en



hardware que requiera poco consumo de recursos (RAM y GPU), como los dispositivos móviles o la Raspberry Pi.

- Los modelos que proporciona YOLO se pueden modificar, requiriendo un mayor nivel de conocimientos, ya que se trata de interactuar con la estructura de métodos sintetizados que permiten la correcta operación de las CNN.
- Los pesos que se obtienen del entrenamiento de YOLO se pueden exportar a distintos formatos y ambientes, incluidos los de Tensorflow y Tensorflow Lite.
- Actualmente la librería de YOLO sigue recibiendo constantes actualizaciones debido a que es de acceso público para que investigadores y usuarios puedan colaborar y aportar tanto investigaciones, citas y mejoras para el modelo.

### **Raspberry Pi:**

Se recomienda que, al trabajar con CNN, sea por medio de GPU (coprocesador dedicado al procesamiento de imágenes y operaciones de coma flotante) e integrar al sistema una tarjeta de video. Debido a que al implementar Deep Learning, presenta una gran carga computacional ya que las imágenes del entrenamiento, las interpreta como matrices numéricas. Por la razón anterior, se decidió utilizar el modelo más pequeño (YOLOv5s). La Raspberry Pi 4 con 8GB de GPU logró con dificultad una tasa fluida de FPS aun trabajando el modelo más pequeño.

Incorporar una tarjeta de video hace que se eleve el precio de la inversión y se deba implementar técnicas y configuraciones adicionales para incorporar los drivers correspondientes de su funcionamiento.

Feng et al., 2022, concluyeron en su estudio que un tamaño de cuadro más grande no aumenta el uso de la GPU, pero aumenta en gran medida el uso de la CPU y la memoria. Por lo tanto, si es necesario usar GPU para realizar muchas tareas además de la inferencia, minimizar el tamaño de los datos de entrada es un método importante para mantener la función de manera estable.

Se implementaron 3 métodos como alternativas de solución:

1. Comprimir el modelo, transformándolo a Tensorflow lite y utilizar los pesos en la Raspberry Pi.
2. Editando el script de “detect.py” el parámetro de la resolución de la imagen que va a recibir en tiempo real.

3. Aumentando la velocidad del GPU (Overclock). Este método es el menos recomendado ya que si el hardware de la Raspberry Pi no cuenta con disipadores de calor y ventiladores puede provocar un sobrecalentamiento en el computador, disminuyendo su tiempo de vida o hasta quemarlo.

## 5. Referencias

1. Manual de Protección Civil ante Casos de Emergencia, Contingencias y Desastres del STCONAPRA. (2017).
2. Vallejo Avilan Johann Sebastian. (n.d.). PRISMA Sistema de rastreo para desastres naturales.
3. Pérez Orozco Bernardo. (2018). Inteligencia artificial. INCYTU.
4. Ortiz Pabón, Efrain Gestión de tecnología e innovación- Teoría, proceso y práctica / Efrain Ortiz Pabón, Nofal Nagles García. -- 2 ed. -- Bogotá : Universidad EAN, 2013. -- (Libro de investigación) 404 p. ISBN: 978-958-756-255-2 1. Gestión tecnológica 2. Innovaciones tecnológicas I. Nagles García, Nofal.
5. Díaz De León Luis, J., Juan, C., Chimal, E., & Santiago, M. D. F. (2014). "Realidad aumentada para el control de robots móviles" "Maestría en Ciencias de la Computación."
6. Pusiol D. Pablo. (2014). Redes Convolucionales en Comprensión de Escenas.
7. Ávila, R. H., Pérez-Perdomo, E., Orozco, D., Sánchez, L. M., & Jvsoft, V. (2018). Deep Learning. Una revisión Image Recognition Library for framework JCLAL View project Human Resource Control Management View project. <https://doi.org/10.13140/RG.2.2.26893.84961>
8. Hidalgo Jorge Martín. (2019). Navegación de UAVs con Deep Learning.
9. Martínez, J., Tutor, M., & Silla Jiménez, F. (2017). Iniciación al Entorno de Deep Learning Torch.
10. Mazzola, I., Tutorizado, O., & López Valverde, F. (n.d.). Aplicación de la Inteligencia Artificial Geoespacial a la epidemiología medio ambiental.
11. Benjumea, A., Teeti, I., Cuzzolin, F., & Bradley, A. (2021). YOLO-Z: Improving small object detection in YOLOv5 for autonomous vehicles.
12. Barchini, G. E. (2008). Las Máquinas de Turing Como Modelo General de la Computación. ¿Hacia un Cambio de Paradigma?
13. Campbell, M., Hoane, A. J., & Hsu, F.-H. (2002). Deep Blue. In Artificial Intelligence (Vol. 134).
14. Delmau Pablo Alberto. (2020). Empleo de Inteligencia Artificial para el desarrollo de las Funciones Operacionales.
15. Navarro, B. G. (2015). Implementación de Técnicas de Deep Learning Implementation of Deep Learning Techniques.
16. García Sánchez, E., Tomás, J., Nalvaiz, A., & Lozano, L. O. (2019). Introducción a las redes neuronales de convolución. Aplicación a la visión por ordenador.
17. Milton, B. W., Aroquipa, M., Optar, P., Título, E. L., & De, P. (2020). DETECCIÓN DE OBJETOS EN IMÁGENES DE HERRAJE INSTALADO EN EL TENDIDO DE FIBRA ÓPTICA DEL PROYECTO REGIONAL DE INSTALACIÓN DE BANDA ANCHA.
18. Roberto, E., Guzmán, S., & Guzmán, G. (2020). Tutorial: Entrenamiento de la Red Neuronal Convolucional YOLO para objetos propios.
19. Muñiz Luis. (2021). Viabilidad y rendimiento de YOLOv5 en Raspberry Pi 4 modelo B.
20. Solsona, A. V. (2018). Facial Expression Detection using Convolutional Neural Networks.
21. Debnath, T., Reza, Md. M., Rahman, A., Beheshti, A., Band, S. S., & Alinejad-Rokny, H. (2022). Four-layer ConvNet to facial emotion recognition with minimal epochs and the significance of data diversity. Scientific Reports, 12(1), 6991. <https://doi.org/10.1038/s41598-022-11173-0>
22. Arnal, M. G. (2018). Estudio y aplicación de las redes neuronales convolucionales 3D.
23. Abellán Marcos. (2021). Implementación y aceleración de algoritmos de IA sobre Raspberry Pi.
24. David, A. :, & Arroyo, E. (2019). Visualizando neuronas en Redes Neuronales Convolucionales.
25. Murthy, C. B., Hashmi, M. F., Bokde, N. D., & Geem, Z. W. (2020). Investigations of object detection in images/videos using various deep learning techniques and embedded platforms-A comprehensive review. In Applied Sciences (Switzerland) (Vol. 10, Issue 9). MDPI AG. <https://doi.org/10.3390/app10093280>
26. Petton, E. (2022). Object detection: train YOLOv5 on a custom dataset. [Disponible en]: <https://blog.ovhcloud.com/object-detection-train-yolov5-on-a-custom-dataset/>. Accesado por última vez 20 de Abril del 2022.
27. Zhang, A., Lipton, Z. C., Li, M., & Smola, A. J. (2021). Dive into Deep Learning Release 0.17.0.
28. Arriola Ignacio. (2018). Detección de objetos basada en Deep Learning y aplicada a vehículos autónomos.

29. Lin, T.-Y., Maire, M., Belongie, S., Bourdev, L., Girshick, R., Hays, J., Perona, P., Ramanan, D., Zitnick, C. L., & Dollár, P. (2014). Microsoft COCO: Common Objects in Context. <http://arxiv.org/abs/1405.0312>
30. Mendiña Clara. (2018). Desarrollo de un sistema de detección de personas en ambientes de interior usando cámaras ojo de pez en plano cenital y algoritmos basados en Deep Learning.
31. Nelson J. (2020). YOLOv5 is Here: State-of-the-Art Object Detection at 140 FPS [Disponible en]: <https://blog.roboflow.com/yolov5-is-here/>. Accesado por última vez 18 de Abril del 2022.
32. Beera, S., Chembeti, M., Hrithik, N., & Chitturi, A. (2021). The Yolo V5 Based Smart Cellphone Detector. In Volatiles & Essent. Oils (Vol. 8, Issue 6).
33. Zhu, X., Lyu, S., Wang, X., & Zhao, Q. (2021). TPH-YOLOv5: Improved YOLOv5 Based on Transformer Prediction Head for Object Detection on Drone-captured Scenarios.
34. Wang, Y., Bashir, S. M. A., Khan, M., Ullah, Q., Wang, R., Song, Y., Guo, Z., & Niu, Y. (2021). Remote Sensing Image Super-resolution and Object Detection: Benchmark and State of the Art. <https://doi.org/10.1016/j.eswa.2022.116793>
35. Verdebout, B. (2020). Machine learning : from idea to reality. [Disponible en]: <https://blog.ovhcloud.com/machine-learning-from-idea-to-reality/>. Accesado por última vez 3 de Abril del 2022.
36. Klein, B. (2021). Machine Learning with Python Tutorial. [www.python-course.eu](http://www.python-course.eu)
37. Casas, J. (2020). Tuneando los hiperparámetros de una red neuronal LSTM para obtener un aprendizaje más eficiente. [Disponible en]: <https://es.linkedin.com/pulse/tuneando-los-hiperpar%C3%A1metros-de-una-red-neuronal-lstm-casas-gonzalez/>. Accesado por última vez 3 de Abril del 2022.
38. Solawetz J. (2020) What is Mean Average Precision (mAP) in Object Detection? [Disponible en]: <https://blog.roboflow.com/mean-average-precision/>. Accesado por última vez 5 de Mayo del 2022.
39. Raspberry Pi 4 Computer Model B. (n.d.). [www.raspberrypi.org](http://www.raspberrypi.org)
40. Alasdair, A. (2022). Raspberry Pi Documentation. [Disponible en]: [https://github.com/raspberrypi/documentation/commits/develop/documentation/asciidoc/accessories/camera/camera\\_hardware.adoc](https://github.com/raspberrypi/documentation/commits/develop/documentation/asciidoc/accessories/camera/camera_hardware.adoc). Accesado por última vez 17 de Abril del 2022.
41. Alasdair, A. (2021). Rpi-os-introduction.adoc. [Disponible en]: <https://github.com/raspberrypi/documentation/commits/develop/documentation/asciidoc/compute-rs/os/rpi-os-introduction.adoc>. Accesado por última vez 17 de Abril del 2022.
42. Tzatalin D. (2020) Labellmg [Disponible en]: <https://github.com/tzatalin/labellmg>.
43. Baume -2021, G. L. (n.d.). Breve introducción a Google Colab. <https://colab.research.google.com/>
44. J. D. Hunter, "Matplotlib: A 2D Graphics Environment," in Computing in Science & Engineering, vol. 9, no. 3, pp. 90-95, May-June 2007, doi: 10.1109/MCSE.2007.55.
45. NumPy Reference Written by the NumPy community. (2022).
46. Cavallaro, A. (2005). IMAGE ANALYSIS AND COMPUTER VISION FOR UNDERGRADUATES.
47. Murray, A. (2022). Python-pillow docs. [Disponible en]: <https://github.com/python-pillow/Pillow/blob/5d070222d21138d2ead002fd33fd5adcb708941/docs/index.rst>. Accesado por última vez 18 de Abril del 2022.
48. YANG Yahui. (2012). Impact Data-exchange based on XML. IEEE.
49. Caña, J. (2022). Pytorch-Tensors and Dynamic neural networks in Python with strong GPU acceleration. [Disponible en]: <https://github.com/pytorch/pytorch>. Accesado por última vez 18 de Abril del 2022.
50. Vogelsang, D. C., & Erickson, B. J. (2020). Magician's Corner: 6. TensorFlow and TensorBoard. Radiology: Artificial Intelligence, 2(3), e200012. <https://doi.org/10.1148/ryai.2020200012>
51. Cheng, G., Chen, X. & Gong, J. Deep Convolutional Network with Pixel-Aware Attention for Smoke Recognition. Fire Technol (2022).
52. Shi, X., Cui, C., He, S., Xie, X., Sun, Y. and Qin, C. (2022), "Research on recognition method of wear debris based on YOLO V5S network", Industrial Lubrication and Tribology, Vol. 74 No. 5, pp. 488-497. <https://doi.org/10.1108/ILT-08-2021-0334>
53. X. Xikun, L. Changjiang and X. Meng (2021). "Application of attention YOLOV 4 algorithm in metal defect detection," 2021 IEEE International Conference on Emergency Science and Information Technology (ICESIT), 2021, pp. 465-468, doi: 10.1109/ICESIT53460.2021.9696808.
54. Chang, H., Borghesani, P., Peng, Z. (2020). Automated assessment of gear wear mechanism and severity using mould images and convolutional neural networks. Tribology International, Volume 147. 106280

55. Shorten C, Khoshgoftaar T,M. (2019) A survey on image data augmentation for deep learning. Journal of Big Data 6(1):1–48
56. Feng, H., Gaoze, M., Shida Z., Peichang., Tao, Y. 2022. "Benchmark Analysis of YOLO Performance on Edge Intelligence Devices" Cryptography 6, no. 2: 16.<https://doi.org/10.3390/cryptography6020016>

## **6. Anexos**

### **Anexo 1. Entrenamiento de Inferencias.**

Enlace: [https://uvmx-](https://uvmx-my.sharepoint.com/:f/g/personal/zs17012869_estudiantes_uv_mx/Ep3CKyGJyjpJrI29U2smPfcByOP34sGU7Mt4piSl8VtFsA?e=5hnWK1)

[my.sharepoint.com/:f/g/personal/zs17012869\\_estudiantes\\_uv\\_mx/Ep3CKyGJyjpJrI29U2smPfcByOP34sGU7Mt4piSl8VtFsA?e=5hnWK1](https://uvmx-my.sharepoint.com/:f/g/personal/zs17012869_estudiantes_uv_mx/Ep3CKyGJyjpJrI29U2smPfcByOP34sGU7Mt4piSl8VtFsA?e=5hnWK1)

### **Anexo 2. Inferencia en Raspberry Pi.**

Enlace: [https://uvmx-](https://uvmx-my.sharepoint.com/:f/g/personal/zs17012869_estudiantes_uv_mx/EjJn4Jm-EMJBqZJI_9AQDOwBBE2K8txZTxDfaGYG76rpCQ?e=WSQcjk)

[my.sharepoint.com/:f/g/personal/zs17012869\\_estudiantes\\_uv\\_mx/EjJn4Jm-EMJBqZJI\\_9AQDOwBBE2K8txZTxDfaGYG76rpCQ?e=WSQcjk](https://uvmx-my.sharepoint.com/:f/g/personal/zs17012869_estudiantes_uv_mx/EjJn4Jm-EMJBqZJI_9AQDOwBBE2K8txZTxDfaGYG76rpCQ?e=WSQcjk)



“Lis de Veracruz: Arte, Ciencia, Luz”

**[www.uv.mx](http://www.uv.mx)**