UNIVERSIDAD ESTATAL PENÍNSULA DE SANTA ELENA

FACULTAD DE SISTEMAS Y TELECOMUNICACIONES

CARRERA DE INF/TI

EXAMEN DE CARÁCTER COMPLEXIVO

Componente Práctico, previo a la obtención del Título de:

INGENIERO EN TECNOLOGÍAS DE LA INFORMACIÓN

"APLICACIÓN DE UN MODELO DE RECONOCIMIENTO DE OBJETOS UTILIZANDO YOLO (YOU ONLY LOOK ONCE)"

AUTOR

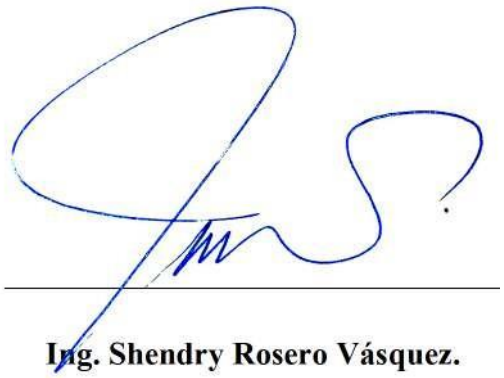
DORYS LISSETE MOREIRA RAMOS

LA LIBERTAD – ECUADOR
2021

APROBACIÓN DEL TUTOR

En mi calidad de tutor del trabajo de componente práctico del examen de carácter complexivo: "APLICACIÓN DE UN MODELO DE RECONOCIMIENTO DE OBJETOS UTILIZANDO YOLO (YOU ONLY LOOK ONCE)", elaborado por la Srta. Moreira Ramos Dorys Lissete, de la carrera de Tecnología de la Información de la Universidad Estatal Península de Santa Elena, me permito declarar que luego de haber orientado, estudiado y revisado, la apruebo en todas sus partes.

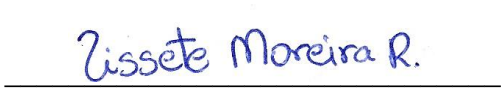
La libertad, 8 de marzo del 2021.



Ing. Shendry Rosero Vásquez.

DECLARACIÓN

El contenido del presente componente práctico del examen de carácter complexivo es de mi responsabilidad; el patrimonio intelectual del mismo pertenece a la Universidad Estatal Península de Santa Elena.



Dorys Lissete Moreira Ramos

AGRADECIMIENTO

En primer lugar, quiero agradecer a mis padres: Francisco Moreira y Ninfa Ramos por su apoyo incondicional en todo momento, por enseñarme la importancia del estudio, ser mi inspiración y motivación para cumplir mis objetivos en la vida.

A mis hermanos y hermanas, en especial a Tnlga. Tatiana Moreira quien fue mi inspiración y guía desde pequeña para ser una profesional.

A la Universidad Estatal Península de Santa Elena por permitirme estudiar en su establecimiento, brindando calidad de estudio a través de sus profesionales docentes que han sido parte de mi carrera universitaria brindándome sus conocimiento y sabiduría en cada semestre transcurrido.

A mi tutor Ing. Shendry Rosero, que me brindó su apoyo y motivación como docente dentro de las aulas de clases, sobre todo en esta última etapa de mi carrera universitaria.

Dorys Moreira

DEDICATORIA

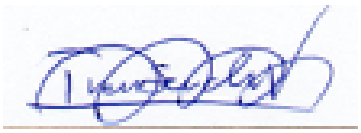
Dedico este logro a mis padres, a mis hermanos, a todas las personas que estuvieron a mi lado a lo largo de esta etapa, quienes han creído en mí dándome fuerzas para culminar con éxito mi carrera universitaria.

Dorys Moreira

TRIBUNAL DE GRADO

Ing. Samuel Bustos Gaibor, Mgt.

DIRECTOR DE LA CARRERA DE TECNOLOGÍAS DE LA INFORMACIÓN

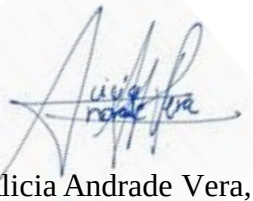


Ing. Ivan Sanchez Vera, Mgt.

DOCENTE ESPECIALISTA

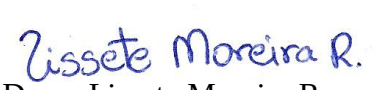
Ing. Shendry Rosero Vasquez, Ms.CC.

DOCENTE TUTOR



Ing. Alicia Andrade Vera, Mgt.

DOCENTE GUÍA UIC



Dorys Lissete Moreira Ramos

Estudiante

RESUMEN

El presente trabajo de investigación propone la implementación de la librería de reconocimiento de objetos llamado Yolo. Se evaluó la efectividad para la detección y clasificación de objetos en fotografías. Se usará la metodología de investigación de tipo exploratorio, para la búsqueda de información y trabajos relacionados con el reconocimiento de objetos a través de distintos modelos usados en la actualidad, con el objetivo de escoger Yolo.

De la literatura investigada se determinó que los algoritmos de reconocimiento de objetos a lo largo de su evolución y actualización han ayudado a realizar diferentes propuestas de innovación en distintas áreas donde se las aplica, pero no existe estudios de una evaluación entre los diferentes modelos para verificar la confiabilidad en el mejor reconocedor de objetos.

El proyecto propuesto se divide en tres secciones, la sección uno contiene la investigación de los modelos de reconocimiento de objetos, planteando objetivos y determinando las etapas que se ejecutaran para la implementación, la sección dos contiene el marco teórico y la metodología del proyecto, la sección tres presenta el experimento ejecutado para obtener los resultados y realizar la comparativa con Keras y TensorFlow.

Este proyecto brindará información oportuna que servirá para futuros proyectos donde se requiera el reconocimiento de objetos.

ABSTRACT

This research work proposes the implementation of the object recognition library called Yolo. The effectiveness for detecting and classifying objects in photographs was evaluated. The exploratory research methodology will be used, for the search for information and works related to the recognition of objects through different models used today, with the aim of choosing Yolo.

The researched literature found that object recognition algorithms throughout their evolution and updating have helped to make different innovation proposals in different areas where they are applied, but there are no studies of an evaluation between the different models to verify reliability in the best object recognizer.

The proposed project is divided into three sections, section one contains the research of object recognition models, setting objectives and determining the stages that will be executed for implementation, section two contains the theoretical framework and methodology of the project, section three presents the experiment executed to obtain the results and make the comparison with Keras and TensorFlow.

This project will provide timely information that will serve future projects where object recognition is required.

Índice

Índice de Gráficos

Índice de Figuras

Índice de Tablas

Capítulo 1

1. Fundamentación

1.1. Antecedentes

El reconocimiento de objetos es una tecnología de visión artificial que se utiliza para reconocer objetos en imágenes o videos. El reconocer objetos es un avance clave en los algoritmos de aprendizaje profundo y aprendizaje automático. Cuando el ser humano observa fotos o videos, podemos detectar rápidamente personas, objetos, ubicaciones y detalles visuales. El objetivo es enseñar a las computadoras a realizar lo que es natural para los humanos: obtener un cierto grado de comprensión del contenido de las imágenes [1].

En la actualidad existen diversas librerías orientadas a la identificación de objetos mediante Maching Learning y visión por computador, la literatura analizada hasta el momento no presenta un análisis de la confiabilidad de los métodos de detección de objetos basados en YOLO, que tan fiable es cada modelo cuando se enfrentan a un ejercicio simple, es decir tener una base de comparación de la confiabilidad de estos modelos basados en experimentos sencillos con instrumento de identificación de objetos en fotografías, objetos comunes dentro de la habitación de un hogar.

Existen modelos que permiten reconocer los objetos a través de imágenes, video o cámara en tiempo real, pero no existe una evaluación entre los algoritmos o librerías que detecte los objetos mediante modelos basados en Yolo, keras y tensorflow que permita determinar qué tan confiable es la identificación o implementación dentro del ambiente del hogar.

En la Universidad Zaragoza se desarrolló un reconocedor de objetos en Android para la aplicación de asistencia cuyos objetivos objetivos la creación de una base de datos de objetos que se ven a diario en un supermercado, se estudió y evaluó distintos algoritmos para realizar la tarea de reconocer objetos en un local determinado utilizando el sistema operativo Android, eligiendo que algoritmo sería el más confiable para poder implementar y aplicarlo en su práctica [2].

La Universidad de la Rioja (UNIR), en la Escuela Superior de Ingeniería y Tecnología se propuso el desarrollo e implementación IoT de un sistema de reconocimiento de

imágenes a nivel industrial, el objetivo del proyecto era diseñar un reconocedor de objetos en la Api "Object Detection" de TensorFlow con el fin de aplicar a nivel industrial, la mejora que propuso en comparación de otros modelos fue la detección de objetos en un mercado, aumentando su eficacia con técnicas de datos [3].

En la Universidad Estatal Península de Santa Elena, la Facultad de Sistemas y Telecomunicaciones propuso la implementación del sistema de registro automático de las placas vehiculares, utilizando reconocimiento óptico de caracteres y visión artificial, en la garita 1 de la universidad estatal península de santa elena, cuyos objetivos fueron el análisis de las cámaras, implementación del sistema de registro automáticos de la entrada y salida de vehículos mediante la visión artificial [4].

1.2. Descripción del proyecto

Se propone la implementación de un modelo de reconocimiento de objetos utilizando Yolo, mediante el análisis de estadísticas para medir la confiabilidad del modelo implementado, para obtener indicadores de confiabilidad en base a la comparación con Keras y TensorFlow, en la implementación se ejecutarán las siguientes etapas:

- Levantamiento de información: En esta etapa se realizará la investigación de los programas requeridos para la implementación de modelo, las versiones y compatibilidad que tienen cada uno de los programas a instalar.
- Instalación de entorno de trabajo: Descarga de software para el entorno de trabajo, instalación de librerías, pruebas de compatibilidad de versiones.
- Implementación: Se implementará el modelo de detector de objetos Yolo.
- Pruebas: Realizar pruebas de objetos elegidos de la habitación de un hogar, probando la cámara de un ordenador, y la efectividad de Yolo.

Para la implementación del modelo se usará las siguientes herramientas:

- **Anaconda Cloud 3.0:** Anaconda Cloud es la asistencia de gestión de paquetes de Anaconda. La nube agiliza el proceso de buscar, ingreso, el almacenamiento y el intercambio de cuadernos, entornos y paquetes conda y PyPI públicos. La nube también le facilita la actualización de los paquetes de software y los entornos que utiliza de manera oportuna. La nube alberga miles de paquetes, cuadernos, proyectos y entornos útiles de Python que se pueden utilizar para diversas

aplicaciones. Puede buscar, descargar e instalar paquetes públicos sin iniciar sesión o incluso sin una cuenta en la nube [5].

- **Pyhton 3.6:** Python es un lenguaje de scripting orientado a objetos, independiente de la plataforma, que puede ejecutar cualquier tipo de programa, desde aplicaciones de Windows hasta servidores web e incluso páginas web. Es un lenguaje interpretado, lo que quiere decir que puede ejecutar sin compilar el código fuente, por lo que ofrece ventajas como velocidad de desarrollo y desventajas como menor velocidad [6].
- **Jupyter:** Jupyter es un software de código libre sin fines de lucro, creado del proyecto IPython en el año 2014, su creación fue destinada para el respaldo de los datos interactivo y la computación científica en lenguajes de programación [7].
- **Yolo:** You only look once (YOLO), sistema de detector de objetos en tiempo real, aplica una red neuronal adaptándose a la imagen, la red divide la imagen en partes y pronostica cuadros de limitaciones y probabilidad de cada región, procesa imágenes a 30FPS (Fotogramas por segundo) [8].

Este proyecto contribuirá a la línea de investigación de Tecnologías y Gestión de la información relacionada al tema de gestión de desarrollo de software, debido a que la propuesta se refiere al estudio de los modelos de reconocimiento de objetos basados en herramientas de desarrollo de código libre [9].

1.3. Objetivos del proyecto

1.3.1. Objetivo general

Implementar un modelo de reconocimiento de objetos utilizando Yolo para el análisis de efectividad de reconocer objetos en fotografías.

1.3.2. Objetivos específicos

- Analizar diferentes modelos de reconocimiento de objetos utilizando en Yolo, Keras y TensorFlow.
- Instalar un ambiente de desarrollo basados en herramientas open source que trabajen con Python.
- Evaluar la confiabilidad del reconocimiento de objetos del modelo Yolo.

1.4. Justificación

El ser humano observa una imagen e inmediatamente reconocen qué objetos hay en la imagen, su ubicación y cómo interactúan. El sistema visual humano es rápido y preciso, por lo que facilita realizar tareas complejas. Los algoritmos de detección de objetos permitirán a las computadoras conducir automóviles sin sensores especiales y facilitará que los equipos auxiliares transmitan en tiempo real información a los usuarios humanos desde la escena y desbloquear el potencial utilizado en sistemas robóticos sensibles de propósito amplio [10].

El modelo Yolo (You only look once) que significa "Solo mira una vez" proporcionará un enfoque diferente de los estudios de reconocimiento de objetos, la herramienta innovadora está siendo implementada en distintos reconocedores a nivel de hardware sin una debida evaluación previa, por lo tanto se realizará etapas donde se verificará su efectividad, la librería Yolo está diseñado para que detecte y clasifique objetos en fotografías, videos, o cámara en tiempo real, su ventaja sobre otros modelos se basa en sus clasificadores, se observa la imagen completa en la prueba, de esta manera provee mejores medidas de precisión en resultados finales,

Con la elaboración de este proyecto se brindará a la comunidad información del modelo Yolo, las pruebas a realizarse brindaran estadísticas oportunas que servirán para futuros proyectos donde se requiera analizar objetos, fotografías o videos en tiempo real.

El tema propuesto está alineado a los objetivos del Plan Nacional de Desarrollo, específicamente al siguiente eje:

Eje 2.- Economía al servicio de la sociedad.

Objetivo 5.- Impulsar la productividad y competitividad para el crecimiento económico sostenible de manera redistributiva y solidaria.

Política 5.6.- Promover la investigación, la formación, la capacitación, el desarrollo y la transferencia tecnológica, la innovación y el emprendimiento, la protección de la propiedad intelectual, para impulsar el cambio de la matriz productiva mediante la vinculación entre el sector público, productivo y las universidades [11].

1.5. Alcance del proyecto

Se implementará un modelo de reconocimiento de objetos utilizando Yolo.

1. Levantamiento de información.
   1.1. Recopilar información de los modelos de reconocimiento de objetos.
   1.2. Analizar librerías para la instalación y ejecución de Yolo.

2. Instalación de entorno de trabajo.
   2.1. Descarga de software requerido para el entorno de trabajo.
   2.2. Instalación de librerías.
   2.3. Comprobar la compatibilidad de versiones.

3. Implementación.
   3.1. Se implementa el modelo Yolo.

4. Pruebas.
   4.1. Pruebas de Imágenes.
   4.2. Pruebas realizadas con objetos elegidos de la habitación del hogar.

En la implementación del modelo detector de objetos, solo se instalará y realizarán las pruebas con el modelo Yolo.

La comparación con los modelos Keras y TensorFlow se realizará en base a experimentos y proyectos previos.

Capítulo 2

2. Marco teórico y metodología del proyecto

2.1. Marco teórico

2.1.1. Red neuronal

Método de computación que durante el aprendizaje simula el funcionamiento neuronal, está formada por elementos interconectados recibiendo información para elaborar una señal que se transmita hasta que genere una respuesta, como resultado a la aplicación en sistemas puede ser entrenada para aprender partiendo de una gran cantidad de muestras para predecir patrones, tendencias y descubrir la relación entre datos [12].

2.1.2. Redes Neuronales Artificiales

Las redes neuronales Artificial (RNA) están inspiradas en la biología [13], es un modelo basado en el cerebro humano y su funcionamiento, formado por la unión de nodos (neuronas artificiales) conectados que transmiten señales entre sí desde el ingreso hasta generar una salida, con el objetivo de aprender automáticamente con modificaciones que se realiza a sí mismo, logrando ordenes que no podría realizar mediante la programación básica basada en reglas [14].

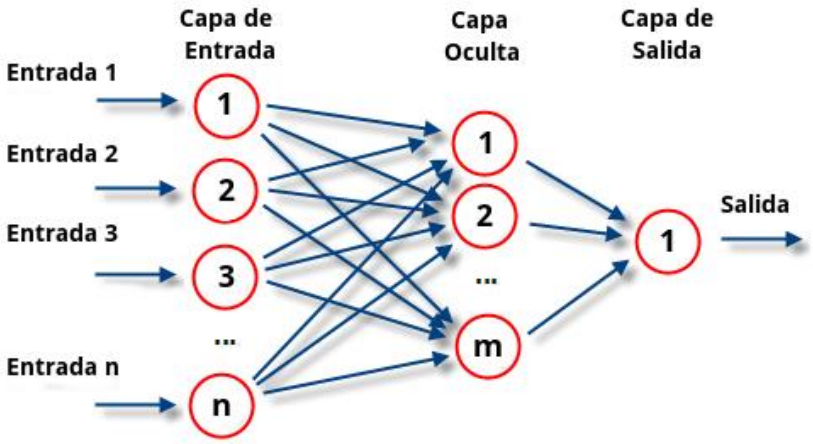


Gráfico 1 Funcionamiento de las Redes Neuronales. *[14]*

2.1.3. Redes Neuronales Convolucionales (CNN)

Las redes convolucionales son ejemplos de un diseño de red neuronal especializado que incluye conocimiento que se basa en invariancias de formas bidimensionales usando patrones de conexión local y con restricciones en los pesos, presentan una arquitectura multicapa, en cada capa hay un número determinado de convoluciones con funciones de activación no lineal, empleando unidad lineal rectificada (ReLU) o función tangente hiperbólica (tanh) para la obtención de resultados [15].

La CNN es un tipo de red neuronal artificial con aprendizaje supervisado que procesa sus capas imitando al córtex visual del ojo humano que usa para la identificación de características en las distintas entradas que pueda identificar objetos y "ver", para realizar este procedimiento tiene varias capas ocultas especializada y con una jerarquía, de tal forma que las primeras capas detectan líneas, curvas y van especializándose hasta que llegue a las capas más profundas que reconozcan formas complejas como rostro o silueta de un objeto [16].

El objetivo es extraer la mayor cantidad de características de una imagen para clasificar y detectar objetos que se encuentran dentro, los parámetros y filtros que se aprenden en estas capas, se ajustan y optimizan junto a componentes de clasificación para hacer mínimo el error de clasificación total. Este modelo ha logrado llevar un rápido desarrollo del campo de detección de objetos, por esto se ha adaptado a varios algoritmos avanzados y actuales en el campo de detección de objetos con redes neuronales convolucionales como Yolo [17].

2.1.4. Reconocedor de objetos

Su objetivo es identificar elementos en una escena pertenecientes a varias categorías, como por ejemplo mesa, silla, cama, etc, en base a entrenamiento de redes neuronales. Para el ser humano esto resulta algo cotidiano, sin embargo, para la implementación con éxito en un computador se hace complejo, por sus diversas técnicas que se han propuesto a lo largo del tiempo para el modelado del reconocedor. Entre los más usados se ubican quienes basan sus características en objetos propios, apariencia, conocimiento de ellos o modelos geométricos, estas técnicas requieren de un equipo de cómputo con gran capacidad de datos para el entrenamiento [18].

2.1.5. Librería Yolo

Aplica una red neuronal única a la imagen por completo, que divide en regiones, pronóstica cuadros delimitadores y probabilidad por cada región. El modelo basa sus ventajas en mirar completamente el gráfico en el momento que se realiza la prueba, por esta razón se tiene el contexto global [8].

El modelo usa Deep learning y CNN en la detección de objetos, y se caracteriza por "Ver" una sola vez la imagen, lo que acelera el resultado final, cuando se hace la división de una cuadricula SxS (a) en cada celda predice N posibles "cuadros delimitadores", calculando el nivel de probabilidad (b), realizando el cálculo SxSxN en gran parte con un nivel de incertidumbre bajo. Cuando se obtiene las predicciones se eliminan las cajas que están por debajo del límite, y las que restan pasan por "supresión no máxima", que sirve para eliminar los posibles elementos que se detectan como duplicados, de esta manera queda el más exacto (c) [19].

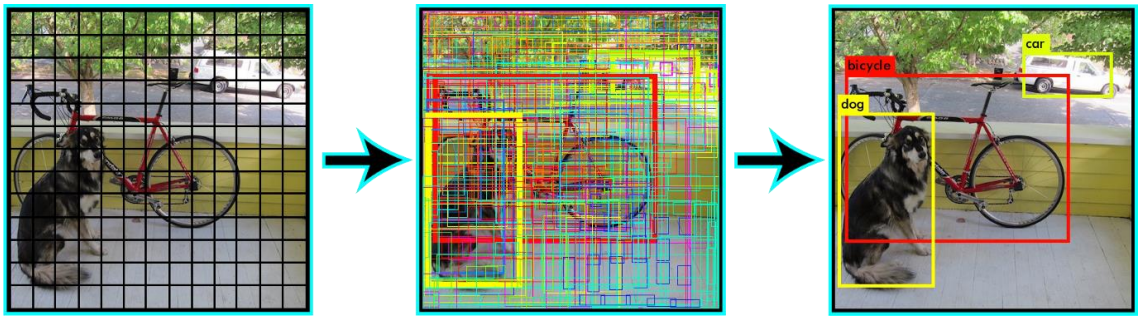


Gráfico 2 Funcionamiento Yolo V2. (a) división de una cuadricula SxS, (b) nivel de probabilidad, (c) Resultado final. [19]

2.1.6. Implementaciones

En la actualidad existen dos formas de implementación, las dos técnicas permiten el entrenamiento para la detección de nuevas clases.

- Darknet: Es la versión oficial de Yolo, creada por los autores del algoritmo, escrito en C con CUDA (Arquitectura Unificada de Dispositivos de Cómputo) que soporta GPU (Unidad de procesamiento gráfico), es un framework para redes neuronales [20].
- Darkflow: Repositorio de código libre para el aprendizaje automático, permitiendo construir y entrenar redes neuronales, es el puerto de unión entre Darknet y TensorFlow, su implementación se creó con el fin de mejorar el proceso en el manejo de imágenes para optimizar el rendimiento en base a la velocidad [21].

2.1.7. Versiones

- Versión 1 You Only Look Once: Unified, Real-Time Object Detection Primer Sistema para la detección de objetos en el conjunto de datos Pascal VOC 2012, detecta 20 clases de elementos:
  - Persona.
  - Pájaro, gato, vaca, perro, caballo, oveja.
  - Avión, bicicleta, barco, autobús, coche, moto, tren.
  - Botella, silla, mesa de comedor, planta en maceta, sofá, tv/ monitor.
- Versión 2: Mejoramiento del entrenamiento y aumento de rendimiento con un modelo convolucional, pero se sigue usando la imagen completa se

ajusta los valores obtenidos en los cuadros que delimitan en lugar de que se prediga el ancho y alto, aun se predice coordenadas “x” y “y”.

- Versión 3: Mejoramiento del entrenamiento y aumento de rendimiento esto incluye predecir múltiples escalas, y un clasificador de columna vertebral.

2.1.8. Keras

Keras es una librería de alto nivel de aprendizaje en red neuronal de código abierto escrito en Python por François Chollet. Ingeniero de Google y construido sobre Tensorflow 2.0, funciona sobre Theano, CNTK o TensorFlow, permite definir y entrenar modelos de redes neuronal, la API maneja las formas de crear modelos, definir capas o configurar múltiples modelos de entrada y salida, compila modelos con funciones de perdida y optimizador, proceso de entrenamiento con funciones de ajustes, en la actualidad es usado en proyectos del sector IA [22].

2.1.9. TensorFlow

Desarrollado por el equipo de Brain de Google, es una herramienta de aprendizaje profundo muy popular en Deep Learning, librería de computación numérica de nivel bajo de Pyhton, que ejecuta de manera eficiente y rápida gráficos de flujos, puede usarse para Deep Learning y otras aplicaciones de cálculo científico [23].

2.2. Metodología del proyecto

2.2.1. Metodología de la investigación

Para el proyecto propuesto se usará la metodología de investigación de tipo exploratorio, se realizará la búsqueda de información y trabajos relacionados con el reconocimiento de objetos a través de distintos modelos usados en la actualidad, con el objetivo de escoger el modelo “You Only Look Once” y realizar la comparación con dos modelos más, con esta investigación se busca ofrecer con valores estadísticos la confiabilidad de Yolo.

El estudio diagnóstico ayudará a conocer los procesos que lleva el modelo Yolo para el reconocimiento de objetos en fotografías.

Se usará como variable de investigación: Implementar un reconocedor de objetos basado en librería Yolo (You Only Look Once).

2.2.2. Técnicas de recolección de información

La técnica que se usará para la recolección de información será mediante paginas oficiales de internet, experimentos y proyectos previos con el fin de recolectar información necesaria para la implementación.

2.2.3. Metodología de desarrollo

Para la implementación del reconocedor de objetos se ejecutarán las siguientes etapas:

Etapa de investigación. - Se realizará la búsqueda de información de los modelos de reconocimiento de objetos, dando como prioridad el modelo Yolo.

Etapa de preparación de entorno de trabajo. – Se instalarán los software, componentes y librerías necesarias para la implementación de Yolo.

Etapa de implementación. – Se ejecutará el modelo Yolo para la detección de objetos en fotografías.

Etapa de pruebas. – Se realizarán las pruebas necesarias para la verificación de la confiabilidad del modelo Yolo.

2.3. Resultados esperados

- Obtener información cuantitativa de la confiabilidad de los modelos de reconocimiento de objetos
- Implementar un identificador de objetos basados en el modelo Yolo.
- Analizar los resultados de las pruebas y obtener indicadores de confiabilidad en base a la comparación con Keras y TensorFlow.

Capítulo 3

3. Propuesta

3.1 Yolo

Proyecto original realizado por Joseph Redmon de la Universidad de Washington en el año 2016, creo un ambiente personalizado llamado Darknet “Sitio Web correspondiente a redes neuronales de código abierto en C”, en colaboración con Santosh Divvala, Ali Farhadi del Instituto Allen de IA y Ross Girshick Investigador de IA de Facebook [24]

lanzaron las versiones YOLOv2 YOLO9000: Mejor, más rápido, más fuerte (2017) y YOLOv3: Una mejora incremental (2018) [8].

Redmon anunció su retiro de la investigación en visión por computadora en febrero del 2020, en abril del mismo año Alexey Bochkovskiy, Chien-Yao Wang y Hong-Yuan Mark Liao presentaron YOLOv4, Glenn Jocher anunció una nueva y actual versión "YoloV5", haciendo una extensión natural de la implementación de YOLOv3 [25].

3.2 YOLO + COCO

Coco es un conjunto de datos de detección de Microsoft con 80 categorías de objetos [26], en unión con Yolo, se crea un modelo previamente entrenado, que reconocerá los siguientes elementos:

Person	*Bicycle*	*traffic light*	*bird*	*frisbee*
Backpack	*Car*	*fire hydrant*	*cat*	*skis*
Umbrela	*Motorbike*	*stop sign*	*dog*	*snowboard*
Handbag	*aeroplane*	*parking meter*	*horse*	*sports ball*
Tie	*bus*	*bench*	*sheep*	*kite*
Suitcase	*train*		*cow*	*baseball bat*
	truck		*elephant*	*baseball glove*
	boat		*bear*	*skateboard*
			zebra	*surfboard*
			giraffe	*tennis racket*

bottle	*banana*	*chair*	*tvmonitor*	*microwave*	*book*
wine	*apple*	*sofa*	*laptop*	*oven*	*clock*
glass	*sandwich*	*pottedplant*	*mouse*	*toaster*	*vase*
cup	*orange*	*bed*	*remote*	*sink*	*scissors*
fork	*broccoli*	*diningtable*	*keyboard*	*refrigerator*	*teddy bear*
knife	*carrot*	*toilet*	*cell phone*		*hair drier*
spoon	*hot dog*				*toothbrush*
bowl	*pizza*				
	donut				
	cake				

Tabla 1. Categoría de objetos Coco
Elaboración: Propia

3.3 Desarrollo

A continuación, se detalla las etapas mencionadas para la implementación.

3.3.1 Levantamiento de información

En esta etapa se procede al análisis de tres versiones del modelo, la primera versión se escoge por ser la principal y pionera en su desarrollo, la versión tres es la última oficial y versión cinco siendo la más actual y rápida creada que representa la investigación de código abierto de Ultralytics en base a las versiones oficiales.

3.3.1.1 Yolo V1 (Original)

Primera red de detección de objetos que combinó cuadros delimitadores con etiquetas para identificar clases en una red diferenciable de extremo a extremo. Sistema para detectar objetos en el conjunto de datos Pascal VOC 2012 "Proporciona conjuntos de datos de imágenes estandarizados para el reconocimiento de clases de objetos", detecta 20 elementos [27].

3.3.1.2 Yolo V3 (Una mejora incremental)

En su última versión oficial es rápido y preciso, basada en modelos anteriores agregando una puntuación de objetividad a la predicción del cuadro delimitador, capas de red troncal y realiza predicciones en tres niveles separados de granularidad para la mejora en detectar objetos más pequeños [8].

3.3.1.3 YoloV5

En su página oficial Ultralytics [28], Glenn Jocher publico la primera versión oficial de YOLOv5, basado en el último modelo publicado por los creadores originales, una gran contribución en esta actualización es traducir el marco de investigación Darknet al marco de Pytorch [29].

El modelo pasa los datos de entrenamiento a través de un cargador de datos, que aumenta los datos en línea, el cargador realizar tres tipos de aumentos: escala, ajuste de espacio y color y aumento de mosaico, este último es el más novedoso porque combina cuatro imágenes en cuatro mosaicos de proposición aleatoria [30].

Ventajas

- Fácil instalación: YOLOv5 requiere crear un ambiente de entorno para la descarga de bibliotecas de python ligeras.

- Rápido de Entrenamiento - El modelo YOLOv5 entrenan de forma extremadamente rápida que ayuda a reducir los costos de experimentación a medida que construye su modelo.
- Puertos de inferencia que funcionan: Puede inferir con YOLOv5 en imágenes individuales, imágenes por lotes, feeds de video o puertos de cámara web.
- Diseño intuitivo: El diseño de la carpeta de archivos es intuitivo y fácil de navegar durante el desarrollo.
- Fácil traducción a dispositivos móviles: Puede traducir fácilmente YOLOv5 de PyTorch a IOS.

3.3.2 Instalación de entorno de trabajo

3.3.2.1 Características de hardware

A continuación, se detalla las características del ordenador:

- Windows 10 Home
- Procesador Intel(R) Celeron(R) CPU N2840 @2.16Hz
- Memoria RAM 4.00 GB
- Sistema operativo de 64 bits, procesador x64

3.3.2.2 Detección de objetos en imágenes usando Pytorch y YoloV3

- **Crear y activar entorno en anaconda**

  conda create -n YoloEntorno anaconda python=3.6

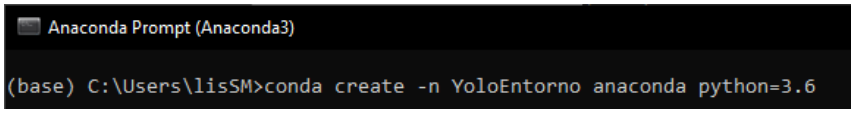


Figura 1. Creación de entorno de trabajo

Elaboración: Propia

  conda activate YoloEntorno

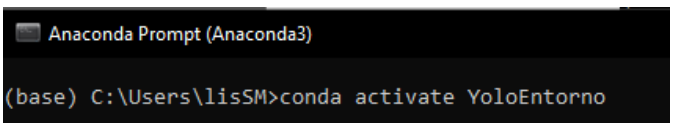


Figura 2. Activación de entorno de trabajo
Elaboración: Propia

- **Instalación de librerías**

pip install opencv-python

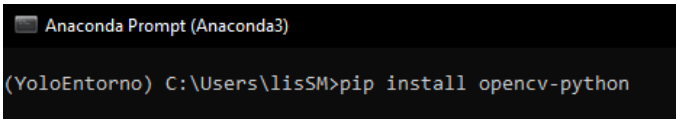


Figura 3. Instalación de librería opencv-python
Elaboración: Propia

conda install pytorch==1.1.0 torchvision==0.3.0 cudatoolkit=10.0 -c pytorch

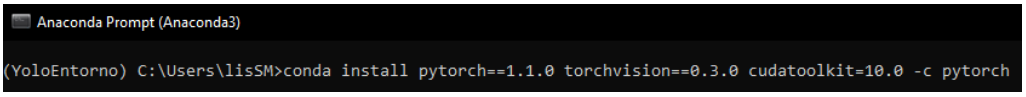


Figura 4. Instalación de librerías pytorch, torchvision, cudatoolkit
Elaboración: Propia

pip install matplotlib

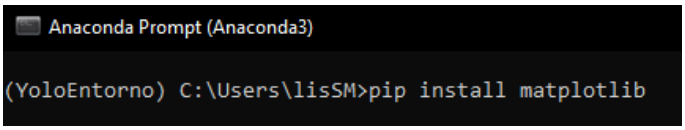


Figura 5. instalación de librería matplotlib
Elaboración: Propia

pip install tensorboard

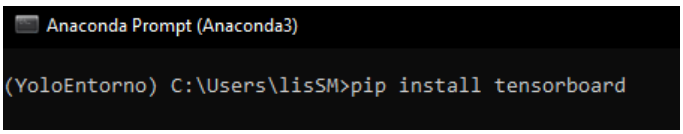


Figura 6. Instalación de libreria tensorboard
Elaboración: Propia

pip install terminaltables

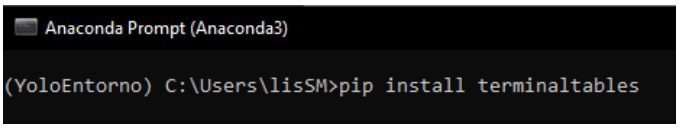


Figura 7. Instalación de librería terminaltables
Elaboración: Propia

pip install pillow

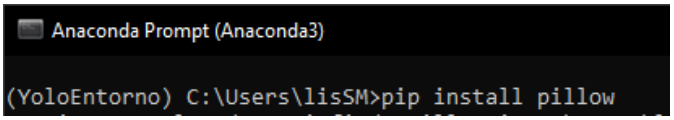


Figura 8. Instalación de libreria pillow
Elaboración: Propia

pip install tqdm

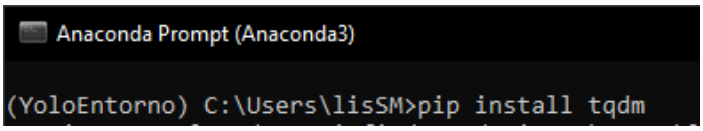


Figura 9. Instalación de librería tqdm
Elaboración: Propia

- **Crear carpeta destino y clonar el repositorio de trabajo**

  git clone https://github.com/DavidReveloLuna/YoloV3_video.git

Figura 10. Creación de carpeta destina y clonación de repositorio
Elaboración: Propia

- **Descargar archivos del entrenamiento de la red y copiar en la carpeta weights**

  Descarga weights for vanilla YOLOv3: https://pjreddie.com/media/files/yolov3.weights

  Descarga weights for tiny YOLOv3: https://pjreddie.com/media/files/yolov3-tiny.weights

  Descarga weights for backbone network: https://pjreddie.com/media/files/darknet53.conv.74

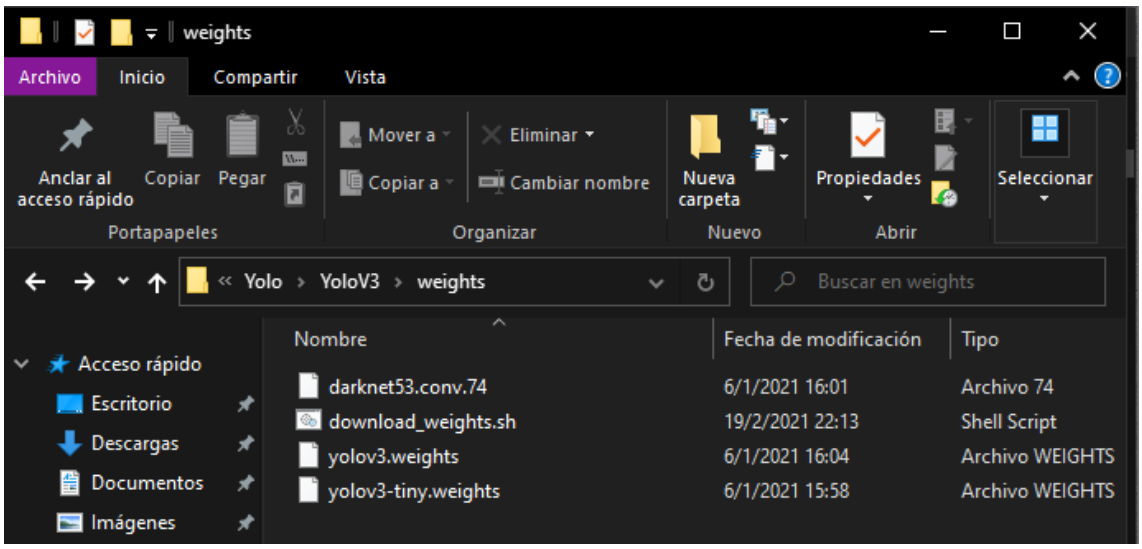


Figura 11. Archivos del entrenamiento de la red
Elaboración: Propia

- **Crear carpeta de salida de los resultados**

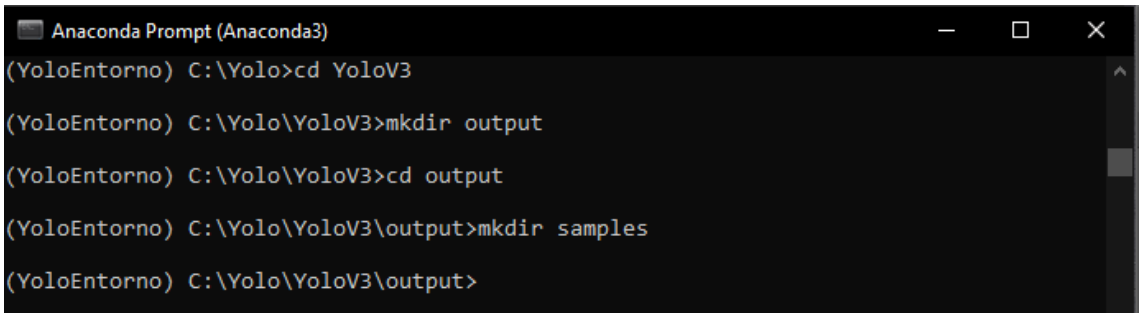


Figura 12. Creación de carpeta de salida de los resultados
Elaboración: Propia

3.3.2.3 YOLOv5 en PyTorch

- **Crear y activar entorno en anaconda**

  conda create -n YoloV anaconda python=3.8

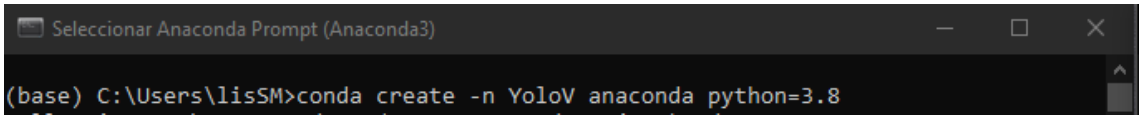


Figura 13. Creación de entorno de trabajo
Elaboración: Propia

  conda activate YoloV

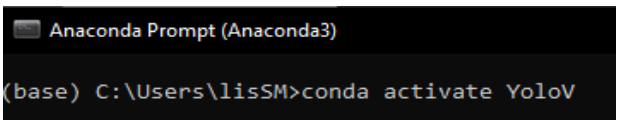


Figura 14. Activación de entorno de trabajo
Elaboración: Propia

- **Crear carpeta destino y clonar el repositorio de trabajo**

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

Figura 15 Creación de carpeta destina y clonación de repositorio

Elaboración: Propia

- **Entrar en la carpeta destino y ejecutar**

  conda install torchvision -c pytorch

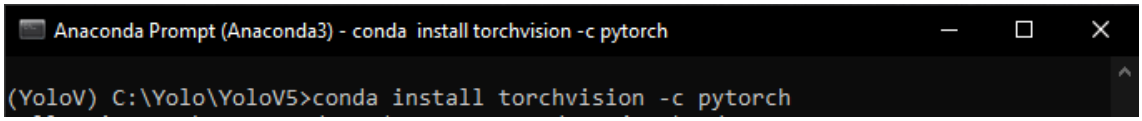


Figura 16. instalación de librería torchvision

Elaboración: Propia

conda install -c conda-forge pycocotools

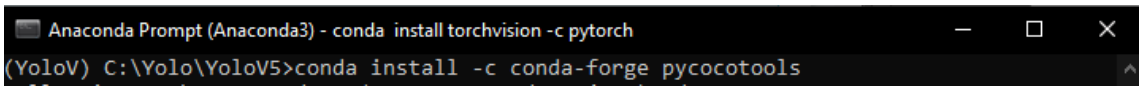


Figura 17. instalación de librería pycocotools
Elaboración: Propia

pip install -r requirements.txt

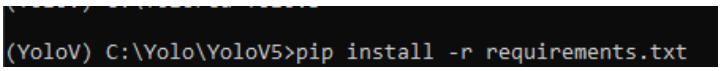


Figura 18. Instalación de requerimientos para la implementación
Elaboración: Propia

- **Descargar archivos del entrenamiento de la red y copiar en la carpeta weights**

Model	size	AP^{val}	AP^{test}	AP_{50}	$Speed_{V100}$	FPS_{V100}		params	GFLOPS
YOLOv5s	640	36.8	36.8	55.6	2.2ms	455		7.3M	17.0
YOLOv5m	640	44.5	44.5	63.1	2.9ms	345		21.4M	51.3
YOLOv5l	640	48.1	48.1	66.4	3.8ms	264		47.0M	115.4
YOLOv5x	640	50.1	50.1	68.7	6.0ms	167		87.7M	218.8
YOLOv5x + TTA	832	51.9	51.9	69.6	24.9ms	40		87.7M	1005.3

Figura 19. Archivos de descargas YoloV5 [31]

Sitio de descarga: https://github.com/ultralytics/yolov5/releases

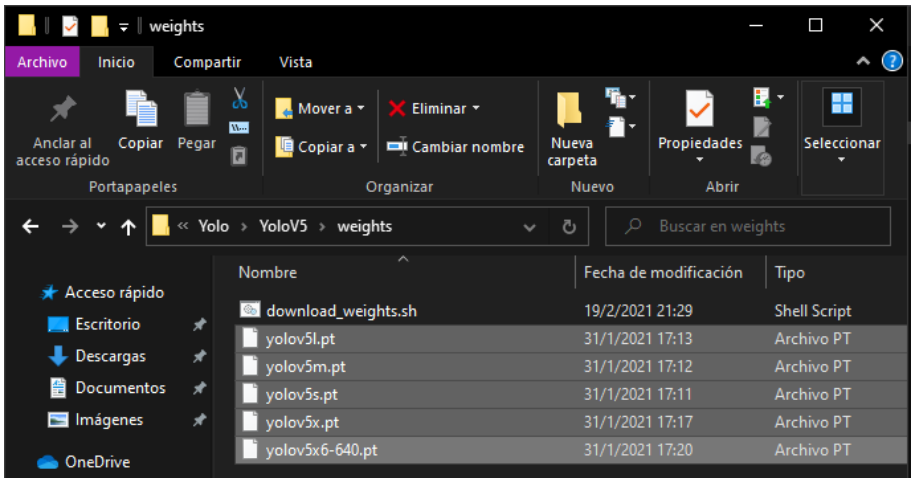


Figura 20. Archivos del entrenamiento de la red
Elaboración: Propia

3.3.3 Implementación

3.3.1.1 Pruebas YOLOv3 con fotografías de una habitación

Cantidad de imágenes: 20 fotografías

Tiempo: 04:21:12

Proceso

Con el comando "python detect.py --image_folder data/samples/", el programa entra a la carpeta donde se encuentran las imágenes que se usaran para la ejecución, detecta individualmente cada uno y compara si lo que reconoce está dentro del listado de coco, mostrando el tiempo que tarda en la detección.

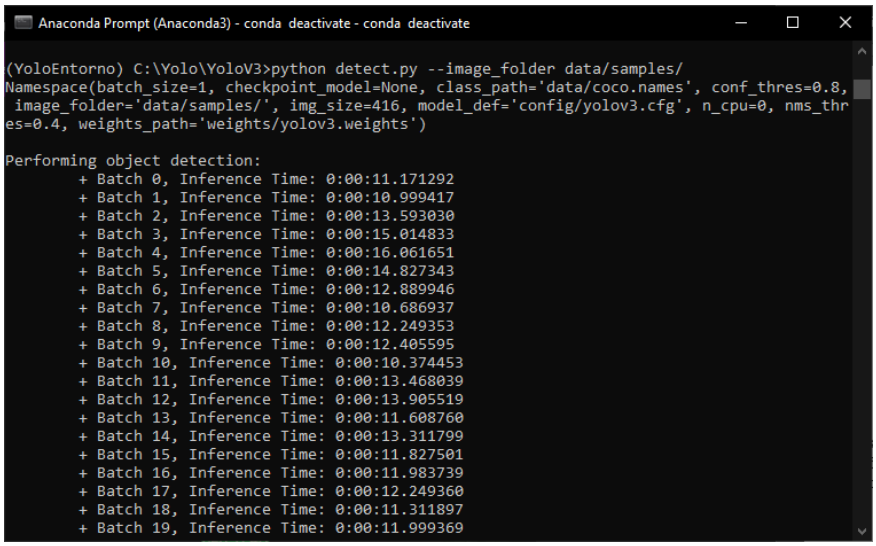


Figura 21 Detección de objetos
Elaboración: Propia

El siguiente proceso automático que realiza es guardar las imágenes con los resultados en formato jpg en la carpeta destino, mostrando los objetos detectados en cada una de las fotografías.

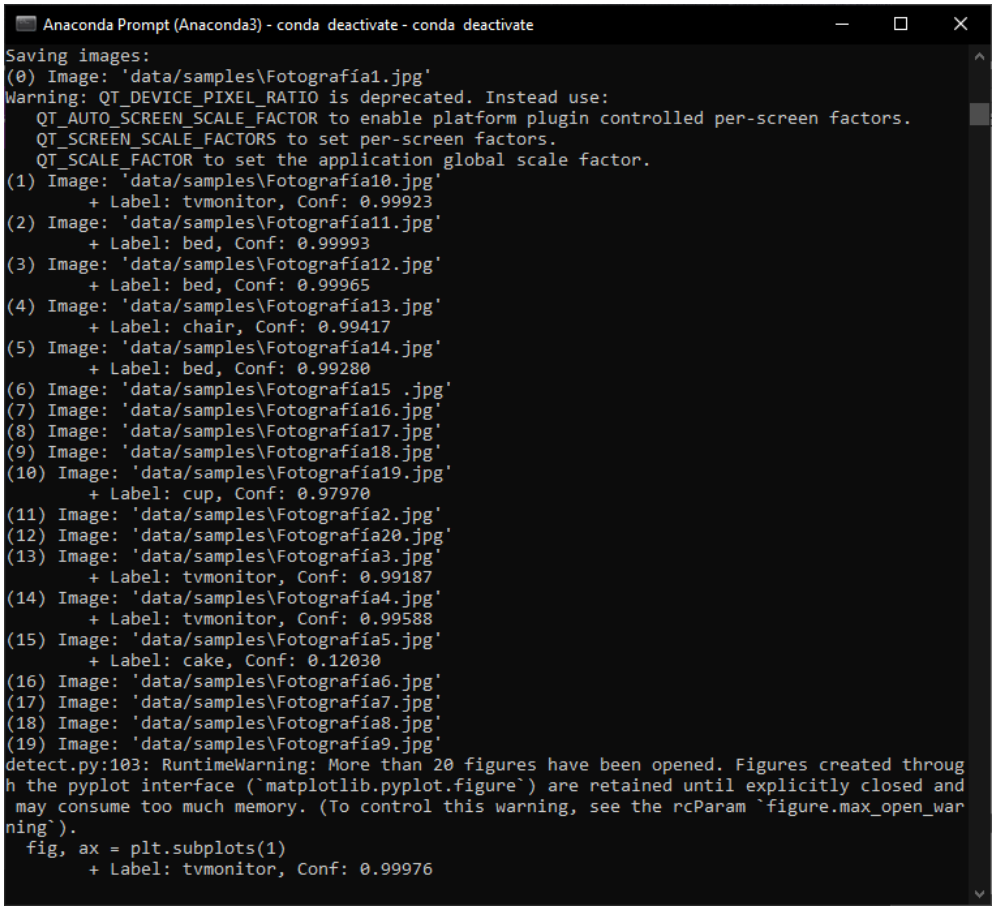


Figura 22 Proceso automático de guardar resultados

Elaboración: Propia

3.3.1.2 Pruebas YOLOv5 con fotografías de una habitación

Cantidad de imágenes: 20 fotografías

Tiempo en segundos: 00:55:59

Proceso

Con el comando "python detect.py --source data/images/", el programa entra a la carpeta donde se encuentran las imágenes que se usarán para la ejecución, cada imagen la compara individualmente con una base de datos que se encuentra en un repositorio en la web y la guarda con formato jpg en una carpeta de salida que se crea automáticamente en el análisis, mostrando en consola los nombres de los objetos y el tiempo que tarda en reconocerlos.

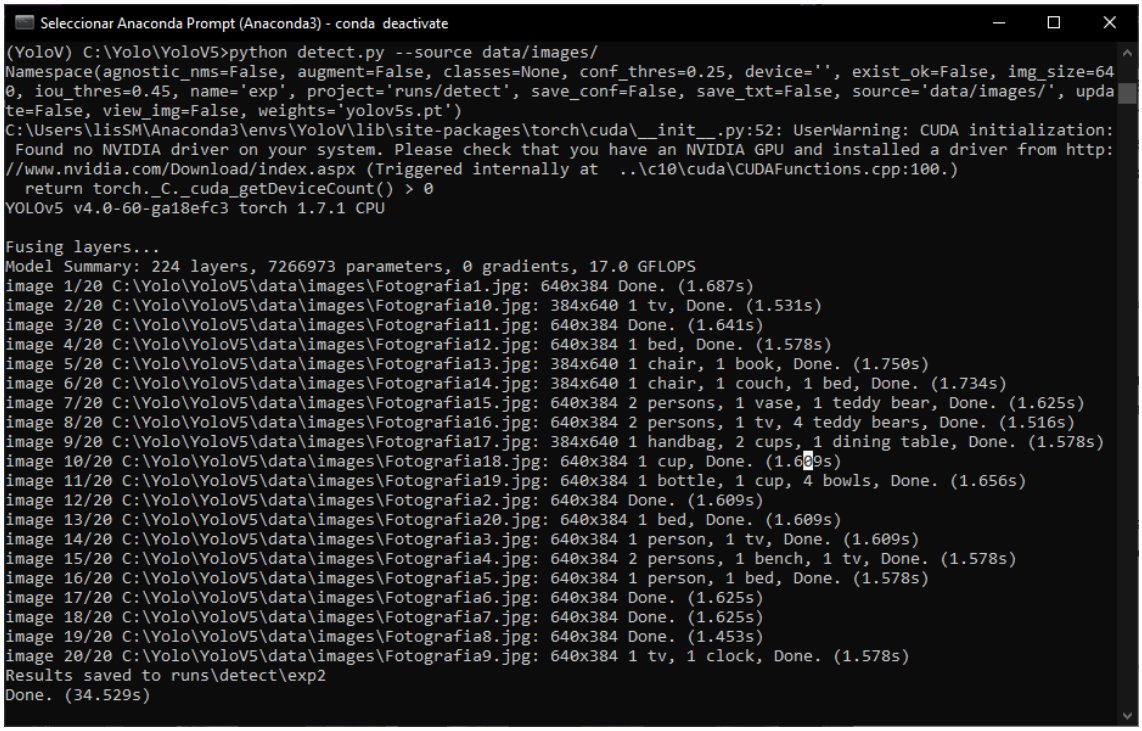


Figura 23 Detección de objetos y guardado automático de resultados.
Elaboración: Propia

3.3.4 Pruebas

Resultados de la ejecución de las pruebas en las versiones tres y cinco, se escoge 10 objetos en dos ángulos diferentes y da un total de 22 fotografías de la habitación de un hogar.

3.3.4.1 Resultados Yolo V3

En las imágenes generadas automáticamente se visualizan las pruebas donde un marco cuadrado encierra el objeto que reconoce y le asigna el nombre de acuerdo a las características y base de datos.

3.3.4.1.1 Muestras positivas

Se muestran los artículos que proporcionan resultados en las imágenes generadas por el programa.

Figura 24 tv
Elaboración: Propia

Figura 25 Tv
Elaboración: Propia

Figura 26 Cama (Bed)
Elaboración: Propia

Figura 27 Cama (Bed)
Elaboración: Propia

3.3.4.1.2 Muestras negativas

Se muestran los artículos que no proporcionan resultados o imágenes con datos erróneos.

Figura 28 Puerta (Door)
Elaboración: Propia

Figura 29 Ventana (Windows)
Elaboración: Propia

Figura 30 Espejo (Mirror)
Elaboración: Propia

Figura 31 Ropero (closet)
Elaboración: Propia

Figura 32 cómoda de ropa (dresser)
Elaboración: Propia

3.3.4.2 Resultados Yolo V5

En las imágenes generadas automáticamente se visualizan las pruebas donde un marco cuadrado encierra el objeto que reconoce, le asigna el nombre de acuerdo a las características, proporciona un porcentaje de efectividad de reconocimiento.

3.3.4.2.1 Muestras positivas

Se muestran los artículos que proporcionan resultados en las imágenes generadas por el programa.

Figura 33 Tv
Elaboración: Propia

Figura 34 Tv
Elaboración: Propia

Figura 35 Cama (Bed)
Elaboración: Propia

Figura 36 Cama (Bed)
Elaboración: Propia

Figura 37 Ropero (closet)
Elaboración: Propia

Figura 38 cómoda de ropa (dresser)
Elaboración: Propia

3.3.4.2.2 Muestras negativas

Se muestran los artículos que no proporcionan resultados o imágenes con datos erróneos.

Figura 39 Puerta (Door)
Elaboración: Propia

Figura 40 Ventana (Windows)
Elaboración: Propia

Figura 41 Espejo (Mirror)
Elaboración: Propia

3.3.4.3 Comparación de V3 y V5

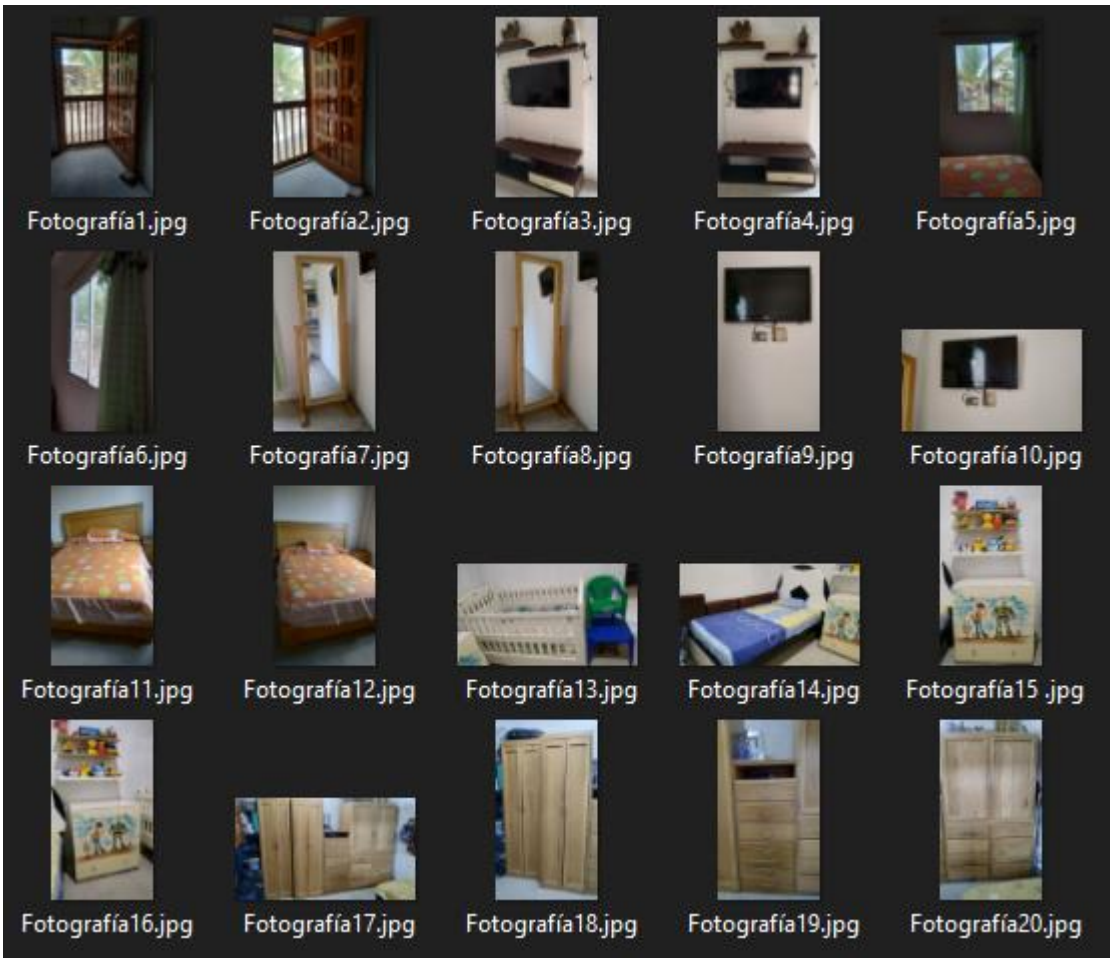


Figura 42 Fotografías
Elaboración: Propia

Nombre de Imagen	V3 (t= 04:21:12)	V5 (t= 00:55:59)	Resultado
F1	-	-	En las dos versiones de evaluación, se visualiza una puerta, pero el programa no la detecta porque no consta en el conjunto de datos de coco.
F2	-	-	

F3	*tvmonitor*	*1 person, 1 tv*	V3 detecta solo el tv, mientras que la v5, detecta primero una persona, pero es una escultura que se asemeja a la figura humana y un tv.
F4	*tvmonitor*	*2 persons, 1 bench, 1 tv*	Es similar a la fotografía 3 sin embargo desde otra posición (de frente), dando el mismo resultado anterior de la v3 y aumenta más objetos en la v5.
F5	*cake*	*1 person, 1 bed*	V3 con resultados no compatibles con la fotografía y v5 solo *bed* está correcto.
F6	-	-	Se visualiza una ventana, pero el programa no la detecta porque no consta en el conjunto de datos de coco.
F7	-	-	En las dos versiones de evaluación, se visualiza un espejo, sin embargo, el programa no la detecta porque no consta en el conjunto de datos de coco.
F8	-	-	
F9	-	*1 tv, 1 clock*	Sin datos en la v3, y con un resultado erróneo en la v5, no existe *clock* en la fotografía.
F10	*tvmonitor*	*1 tv*	Resultados correctos en las dos versiones de evaluación.
F11	*bed*	-	Resultado correcto en la v3 y sin datos en la v5.

F12	*bed*	*1 bed*	Resultados correctos en las dos versiones de evaluación.
F13	*chair*	*1 chair, 1 book*	Resultado no correcto en la v5, no existe un *book* en la fotografía.
F14	*bed*	*1 Chair, 1 couch, 1 bed*	V3 y v5 con resultados correctos, mostrando más objetos en la v5.
F15	-	*2 persons, 1 vase, 1 teddy bear*	Se obtienen resultados solo en la v5, de doce objetos solo existe fallo en dos.
F16	-	*2 persons, 1 tv, 4 teddy bears*	
F17	-	*1 handbag, 2 cups, 1 dinning table*	En la v3 solo da un resultado correcto en la fotografía 19, mientras que en la v5 de doce resultados solo siete son correctos.
F18	-	*1 cup*	
F19	*cup*	*1 bottle, 1 cup, 4 bowls*	
F20	-	*1 bed*	

Tabla 2. Tabla comparativa YoloV3 - YoloV5
Fuente: Elaboración propia

3.4 Datos estadísticos de porcentaje de efectividad

Versión 3		
Resultados Positivos	Resultados Negativos	Resultados Nulos
8	1	11

Tabla 3. Tabla estadística de datos versión 3
Fuente: Elaboración propia

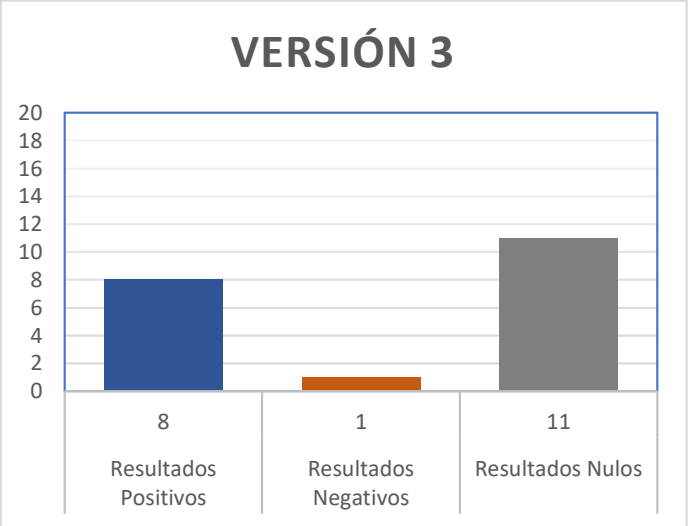


Gráfico 3. Gráfico estadístico de datos versión 3
Fuente: Elaboración propia

Versión 5		
Resultados Positivos	Resultados Negativos	Resultados Nulos
12	2	6

Tabla 4. Tabla estadística de datos versión 5
Fuente: Elaboración propia

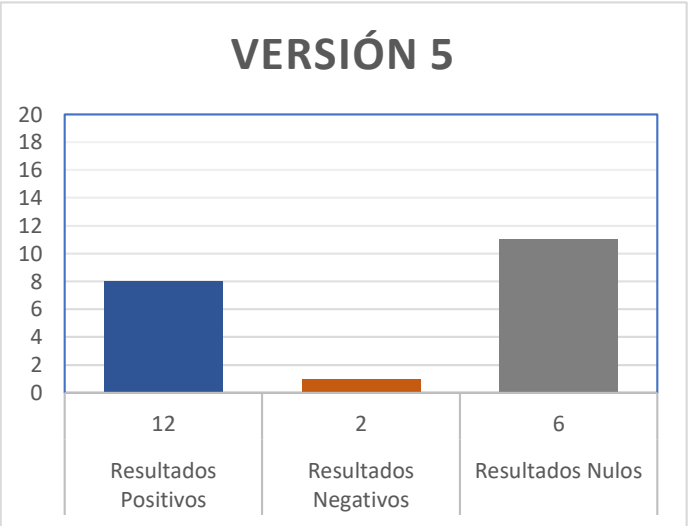


Gráfico 4. Gráfico estadístico de datos versión 5
Fuente: Elaboración propia

	Resultados Positivos	Resultados Negativos
Versión 3	8	12
Versión 5	12	8

Tabla 5. Tabla estadística comparativa
Fuente: Elaboración propia

Gráfico 5.Gráfico estadístico comparativo
Fuente: Elaboración propia

3.5 Comparación de Yolo con Keras y TensorFlow

Características	Yolo	Keras	TensorFlow
Definición	Librería Yolo, diseñado para la detección y clasificación de objetos en fotografías, videos, o cámara en tiempo real [8].	Librería de redes neuronales de alto nivel para Python [22].	Librería de código abierto de extremo a extremo para el aprendizaje automático [23].
Compatibilidad	• Python 3.6 – 3.8. • Windows 7 o posterior. • macOS 10.12.6 (Sierra) o posterior.	• Python 3.5–3.8. • Ubuntu 16.04 o posterior. • Windows 7 o posterior. • macOS 10.12.6 (Sierra) o posterior.	• Python 3.5–3.8. • Ubuntu 16.04 o posterior. • Windows 7 o posterior. macOS 10.12.6 (Sierra) o posterior.

Lenguaje de programación	C	Python	C++
Implementación	Rápida y fácil de entender.	Rápida y fácil de entender.	Rápida y fácil de entender.
Plataformas	• Linux • Mac OS • Windows • Navegador web compatible con .js	• iOS con CoreML • Android con Tensorflow Android, • Navegador web compatible con .js • Motor en la nube • Raspberry Pi	• Linux • Mac OS • Windows • Android • Raspberry Pi

Tabla 6. Tabla comparativa Yolo, Keras y TensorFlow
Elaboración: Propia

Conclusiones

- El estudio de tiempos de ejecución en las dos versiones evaluadas, garantiza la efectividad de los resultados obtenidos, así como la confiabilidad de los datos que emite.
- Se determino que, en una prueba de 20 fotografías, la versión 3 tiene un 40% de efectividad en el reconocimiento de los elementos, mientras que la versiones 5 tiene un alto índice de respuestas positivas con el 70% de confiabilidad.
- La versión 5 usa una red neuronal convolucional que posee una arquitectura que implementa una gran cantidad de filtros pequeños en sus capas de detecciones de los elementos, permitiendo obtener mejores resultados en la evaluación del modelo.

Recomendaciones

- Después del análisis de tiempos de procesamiento relacionados con el entrenamiento de redes neuronales y la calidad de los resultados finales, se recomienda el uso de una GPU, debido a que las imágenes son de calidad alta y necesita un procesamiento óptimo para reducir y optimizar la detección y su tiempo de ejecución.

- Para mejorar la confiabilidad del reconocimiento y el tiempo de respuesta en la versión tres, se recomienda el estudio de métodos de programación o la utilización de algoritmo de aprendizaje profundos.
- Si se requiere que el programa identifique objetos que no está en la base de datos de Coco, se deberá entrenar el modelo Yolo aumentando la base de datos local, insertando diferentes imágenes de los elementos, proporcionando información relevante para la detección.

Bibliografía

[1] MathWorks, «MathWorks,» [En línea]. Available: https://la.mathworks.com/solutions/image-video-processing/object-recognition.html.

[2] A. M. Ferrer, «Reconocimiento de objetos en Android para,» Zaragoza, 2016.

[3] D. S. Cabreros, «Desarrollo e implementación IoT de un sistema de reconocimiento de imágenes a nivel industrial,» Valladolid, 2019.

[4] T. G. E. E. Rodríguez Yagual Cristhian Antonio, «Implementación del sistema de registro automático de las placas vehiculares utilizando reconocimiento óptico de caracteres y visión artificial, en la garita 1 de la Universidad Estatal Penisula de Santa Elena,» Santa Elena, 2013.

[5] Anaconda, «Anaconda Cloud,» 2020. [En línea]. Available: https://docs.anaconda.com/anaconda-cloud/#anaconda-cloud.

[6] Miguel Angel Alvarez, «desarrolloweb,» 19 Noviembre 2003. [En línea]. Available: https://desarrolloweb.com/articulos/1325.php.

[7] Project Jupyter, «Jupyter,» [En línea]. Available: https://jupyter.org/.

[8] J. a. F. A. Redmon, «YOLOv3: An Incremental Improvement,» arXiv, 2018. [En línea]. Available: https://pjreddie.com/darknet/yolo/.

[9] Facultad de Sistemas y Telecomunicaciones, «FACSISTEL,» 2019. [En línea]. Available: http://facsistel.upse.edu.ec/index.php?option=com_content&view=article&id=58&Itemid=463.

[10] S. D. R. G. A. F. Joseph Redmon, «You Only Look Once: Unified, Real-Time Object Detection,» Washington, 2015.

[11] Secretaría Nacional de Planificación y Desarrollo - Senplades 2017, «Plan Nacional de Desarrollo 2017-2021. Toda una Vida,» Quito, 2017.

[12] J. E. Caparros, «Redes neuronales: concepto, fundamentos y aplicaciones en el labomtorio clinico,» San José, 1994.

[13] R. C. Almeida, «Inteligencia Artificial».

[14] ATRIA Innovation, 22 Octubre 2019. [En línea]. Available: https://www.atriainnovation.com/que-son-las-redes-neuronales-y-sus-funciones/#:~:text=Las%20redes%20neuronales%20artificiales%20son,entrada%20hasta%20generar%20una%20salida..

[15] M. A. L. Pacheco, «Identificación de sistemas no lineales con redes neuronales convolucionales,» Ciudad de México, 2017.

[16] J. I. Bagnato, «Aprende Machine Learning,» 29 Noviembre 2018. [En línea]. Available: https://www.aprendemachinelearning.com/modelos-de-deteccion-de-objetos/.

[17] C. D. J. Á. F. Manlio Massiris, «Detección de equipos de protección personal mediante red neuronal convolucional yolo,» Coruña, 2018.

[18] J. M. F.-A. M. J. G.-J. Jose-Raul Ruiz-Sarmiento, «Tutorial para el reconocimiento de objetos basado en caracter´isticas empleando herramientas python,» Málaga.

[19] E. A, «Medium,» 12 mayo 2018. [En línea]. Available: https://medium.com/@enriqueav/detecci%C3%B3n-de-objetos-con-yolo-implementaciones-y-como-usarlas-c73ca2489246.

[20] J. Redmon, «Darknet: Redes neuronales de código abierto en C,» 2013-2016. [En línea]. Available: http://pjreddie.com/darknet/.

[21] C. H. A. Guerrero, «Sistema de estimación de número de personas en tiempo real durante misiones de reconocimiento del ejército ecuatoriano utilizando vehículos aéreos no tripulados multirotor,» Sangolqui, 2018.

[22] Keras, «Keras,» [En línea]. Available: https://keras.io/. [Último acceso: 19 Febrero 2021].

[23] Tensorflow, «Tensorflow,» [En línea]. Available: https://www.tensorflow.org/. [Último acceso: 19 02 2021].

[24] S. D. R. G. A. F. Joseph Redmon, «You Only Look Once: Unified, Real-Time Object Detection,» pp. 779-788, 2016.

[25] J. S. Joseph Nelson, «Roboflow,» 12 06 2020. [En línea]. Available: https://blog.roboflow.com/yolov4-versus-yolov5/. [Último acceso: 19 02 2021].

[26] Coco Common Objects in Context, [En línea]. Available: https://cocodataset.org/#explore. [Último acceso: 01 Febrero 2021].

[27] L. v. G. C. W. J. W. A. Z. Mark Everingham, «The PASCAL Visual Object Classes (VOC) Challenge,» *International Journal of Computer Vision,* vol. 88, nº 2, pp. 303-338, 2010.

[28] Ultralytics LLC, «Ultralytics,» [En línea]. Available: https://www.ultralytics.com/.

[29] J. Glenn, «Github,» 05 Enero 2021. [En línea]. Available: https://github.com/ultralytics/yolov5.

[30] J. Solawetz, «Roboflow,» 29 Junio 2020. [En línea]. Available: https://blog.roboflow.com/yolov5-improvements-and-evaluation/.

[31] Glenn-jocher, «Github,» 04 01 2021. [En línea]. Available: https://github.com/ultralytics/yolov5/. [Último acceso: 19 02 2021].