**UNIVERSIDAD POLITÉCNICA ESTATAL DEL CARCHI**



**FACULTAD DE INDUSTRIAS AGROPECUARIAS Y CIENCIAS AMBIENTALES**

**CARRERA DE COMPUTACIÓN**

**MÉTODOS NUMÉRICOS**

**PROYECTO FINAL**

**Tema**: “Algoritmos que permitan reconocer objetos en movimiento en tiempo real a través de un sistema de cómputo numérico.”

**AUTORES:**

Guerrero Byron

Obando Jessica

**DOCENTE:**

Msc. Samuel Lascano

**CURSO:** 4° “A-M”

Junio – septiembre 2020

**ÍNDICE**

[I. PROBLEMA 5](#_Toc49974765)

[1.1. PLANTEAMIENTO DEL PROBLEMA 5](#_Toc49974766)

[1.2. JUSTIFICACIÓN 6](#_Toc49974767)

[1.3. OBJETIVOS 7](#_Toc49974768)

[1.3.1. Objetivo general 7](#_Toc49974769)

[1.3.2. Objetivos específicos 7](#_Toc49974770)

[II. FUNDAMENTACIÓN TEÓRICA 8](#_Toc49974771)

[2.1. ANTECEDENTES INVESTIGATIVOS 8](#_Toc49974772)

[2.2. MARCO TEÓRICO 9](#_Toc49974773)

[III. CONCLUSIONES 22](#_Toc49974774)

[3.1. CONCLUSIONES 22](#_Toc49974775)

[3.2. RECOMENDACIONES 22](#_Toc49974776)

[IV. BIBLIOGRAFÍA 23](#_Toc49974777)

[V. ANEXOS 24](#_Toc49974778)

[5.1. EVIDENCIAS 24](#_Toc49974779)

[Figura 1: Entorno de trabajo ANACONDA 10](#_Toc50313634)

[Figura 2. Creación del ambiente 10](#_Toc50313635)

[Figura 3. Descarga de complementos 11](#_Toc50313636)

[Figura 4. Instalación de librerías 11](#_Toc50313637)

[Figura 5. Activación del entorno creado 12](#_Toc50313638)

[Figura 6. Instalación de librerías y paquetes faltantes 12](#_Toc50313639)

[Figura 7. Instalación de numpy 12](#_Toc50313640)

[Figura 8. Instalación del torch 13](#_Toc50313641)

[Figura 9. Instalación de paquetes necesarios 13](#_Toc50313642)

[Figura 10. Verificación de paquetes necesarios 14](#_Toc50313643)

[Figura 11. Instalar Tensorboard 14](#_Toc50313644)

[Figura 12. Instalar terminaltables 14](#_Toc50313645)

[Figura 13. Instalación de pillow 15](#_Toc50313646)

[Figura 14. Instalación de tqdm 15](#_Toc50313647)

[Figura 15. Clonar el directorio 15](#_Toc50313648)

[Figura 16. Instalación de la carpeta en el disco 16](#_Toc50313649)

[Figura 17. Carpeta creada en el repositorio 16](#_Toc50313650)

[Figura 18. Descarga de pesos de entrenamiento de la red 17](#_Toc50313651)

[Figura 19. Crear nueva carpeta output 17](#_Toc50313652)

[Figura 20. Crear carpeta samples 18](#_Toc50313653)

[Figura 21. Comprobar si el proyecto se ejecuta correctamente 18](#_Toc50313654)

[Figura 22. Procesamiento de imágenes 18](#_Toc50313655)

[Figura 23. Paso de acceso a la cámara 19](#_Toc50313656)

[Figura 24. Ingreso al prompt 19](#_Toc50313657)

[Figura 25. Activación del proyecto 19](#_Toc50313658)

[Figura 26. Dependencias de Sandbox 20](#_Toc50313659)

[Figura 27. Implementación de clases en Sandbox 21](#_Toc50313660)

[Figura 28. Vista general de Sandbox 21](#_Toc50313661)

[Figura 29. Reconocimiento de objetos mediante el proyecto desarrollado 24](#_Toc50313662)

[Figura 30. Reconocimiento de personas 24](#_Toc50313663)

# **PROBLEMA**

## PLANTEAMIENTO DEL PROBLEMA

El desarrollo de tecnologías abre nuevos e interesantes canales tanto para la provisión de servicios a la sociedad como para mejorar la calidad y oportunidad de la información a la que los usuarios pueden acceder.

En este sentido la implementación de nueva tecnología representa un gran reto para el desarrollo de nuevas propuestas en el ámbito virtual con apoyo de empresas que fomentan la creación de espacios libres en donde los programadores pueden generar códigos y base de datos para proyectos en el ámbito de inteligencia artificial.

A mediados del año 1963, Lawrence Roberts en sus aportes académicos plantea la forma de obtener datos de 3D a partir de vistas 2D de sólidos (Lawrence, 1963). Por este aporte a continuación es reconocido como el primer promotor de la visión artificial. Con esta información varios investigadores del Instituto Tecnológico de Massachusetts, usan los aportes de Lawrence para estudios con vistas simples en solidos geométricos. Dando paso en un corto tiempo a estudiar imágenes capturadas del mundo real. Esto sería el inicio de la Visión Artificial.

Este hecho hace que parte de los desarrollos tecnológicos más avanzados estén encaminados a la mejora de la calidad de vida de las personas. El desarrollo de sistemas de software dotados de un alto nivel de movilidad e inteligencia permitirá evolucionar dentro del campo tecnológico, sin embargo, se ha visto que tienen grandes limitaciones de uso en entornos domésticos dado que necesitan de grandes y complejos sistemas de control que hacen muy difícil su aprendizaje

Actualmente, en cualquier web medianamente moderna, todo lo que hacemos queda registrado. Cada link que visitamos, cada movimiento de ratón, cuánto nos desplazamos y cuánto estamos viendo qué cosas. Todo ello con el objetivo, al menos teóricamente, de ofrecer al usuario la mejor experiencia, de permitirle encontrar antes lo que desea y de que, al final, eso repercuta positivamente para la compañía, tanto social como económicamente. Es fácil ver que el manejo de toda esta información es simplemente imposible para cualquier humano y los algoritmos tradicionales tienen numerosos problemas o son ineficientes. Para ello ha surgido un nuevo campo en la computación que emula sistemas biológicos. En concreto, se busca la resolución de problemas complejos a través de sistemas de computación basados en el cerebro humano. A estos sistemas se les llama Redes Neuronales.

## JUSTIFICACIÓN

Las soluciones de ingeniería basadas en el uso de visión artificial, constituyen una tendencia tecnológica que se encuentra en pleno auge. La implementación de sistemas basados en algoritmos de visión permite obtener este tipo de datos de forma precisa mediante la combinación de técnicas adecuadas para cada ambiente operativo. (Miller, Mohan, Eichel, & Mishra, 2015)

El proyecto que se resume a continuación tiene como objetivo principal el desarrollo de un modelo de clasificación de objetos que determine en tiempo real el análisis y respuesta de que objetos se muestran o encuentran en un entorno determinado. Para ello, se utiliza una variedad de repositorios y librerías necearías para su ejecución dentro de dispositivos electrónicos con el objetivo de determinar el nivel de uso que se les dará y el beneficio que aportaran por medio de la inteligencia artificial. La detección de primer plano es una de las grandes áreas de trabajo en el ámbito de la visión por computador (computer vision). Muchas de sus aplicaciones están estrechamente relacionadas con la automatización y optimización de tareas rutinarias y de larga duración. A modo de ejemplo:

* **Control de Tráfico:** conteo de vehículos, para detectar donde hay más tráfico, evitar colisiones como lo hacen los autos autónomos, seguimiento de vehículos.
* **Seguridad:** detectar la presencia de objetos, personas o animales en espacios físicos, reducción de riesgos en espacios industriales, etc…
* **Reconstrucción 3D:** para la generación del segmentado y elaboración de los planos de reconstrucción es necesario utilizar el detectar
* **Detección y reconocimiento de gestos:** si bien es reciente, el crecimiento de esta aplicación es notorio. La detección de gestos humanos para la interacción con máquinas es uno de los puntos de mayor destaque.

## OBJETIVOS

### Objetivo general

Implementar una recopilación de datos de manera visual, que resuelva mediante redes neuronales, para que devuelva información sobre un entorno específico para lo cual será un detector de objetos en tiempo real teniendo en cuenta el entorno que se encuentre.

Se intentará que la interacción con la web sea lo más rápida e intuitiva posible, así como que las recomendaciones sean lo suficientemente precisas.

### Objetivos específicos

* Analizar un código de seguimiento de objetos.
* Determinar la cantidad de objetos que puede procesar.
* Instalar un código de visión artificial en nuestros ordenadores usando ANACONDA y repositorios YOLO para examinar su funcionalidad

# **FUNDAMENTACIÓN TEÓRICA**

## ANTECEDENTES INVESTIGATIVOS

**VISION ARTIFICIAL**

Posee un extenso grupo de técnicas y métodos para la obtención, creación y análisis de imágenes en tiempo real por medio de un computador. Su complejidad es muy extensa ser inversa inherente su información. Queriendo decir que una imagen bidimensional, se quiere obtener su característica tridimensional en una escena capturada. (Szeliski, 2010)

La visión artificial es una de las ramas de soporte de la robótica. Como por ejemplo los brazos robóticos para recoger y ubicar objetos en una empresa.

**Aplicaciones de la visión artificial**:

* Reconocimiento y detección de rostros.
* Interpretación de expresiones faciales.
* Referencia visual para objetos en movimiento.
* Análisis en la medicina, mediante el procesamiento y diagnóstico.
* Reconocimiento de texto.
* Biometría (de rostros o dinámica).
* Seguimiento de objetos.

**Entrenamiento de imágenes**

Previo a la implementación de cualquier aplicación de visión artificial, es necesario someter a la imagen capturada a un entrenamiento que facilite la ejecución del o los algoritmos seleccionados. Se lista a continuación algunas de las tareas de entrenamiento de imágenes comunes:

* **Redimensionamiento:** comprende una operación básica de cambio de dimensiones del archivo de imagen a procesar. Lo cual resulta crítico para aplicaciones de tiempo real.
* **Reducción de ruido:** consiste en la eliminación de características indeseadas en la imagen capturada, debido a condiciones propias del ambiente de trabajo. **Ecualización de histograma:** creación de una nueva imagen con el mismo valor de intensidad a píxeles que posean valores congruentes entre sí (umbrales).
* **Extracción de características:** ciertos algoritmos requieren trabajar con una serie de parámetros propios de los pixeles de la imagen. Dichos parámetros se conocen como características.
* Características de nivel 1 y 2: consisten en datos de intensidad de color, contraste, saturación, entre otros; de un pixel en específico.
* Características de nivel 3: las características de nivel 3 son el resultado de aplicar un algoritmo de visión específico a las características de nivel 1 y 2.
* Vectores de características: contenedores de característica de píxeles. Los mismos son creados con el objeto de reducir tiempo de cómputo, o presentar datos de entrada compatibles con ciertos algoritmos.

## 2.2. MARCO TEÓRICO

**Instalación de librerías y algoritmo para crear un detector de objetos en tiempo real en nuestro computador**

1. Utilizamos un entorno de trabajo llamado ANACONDA instalado con anterioridad para poder usar un CMD llamado Anaconda Prompt el cual nos servirá para el objetivo planteado.

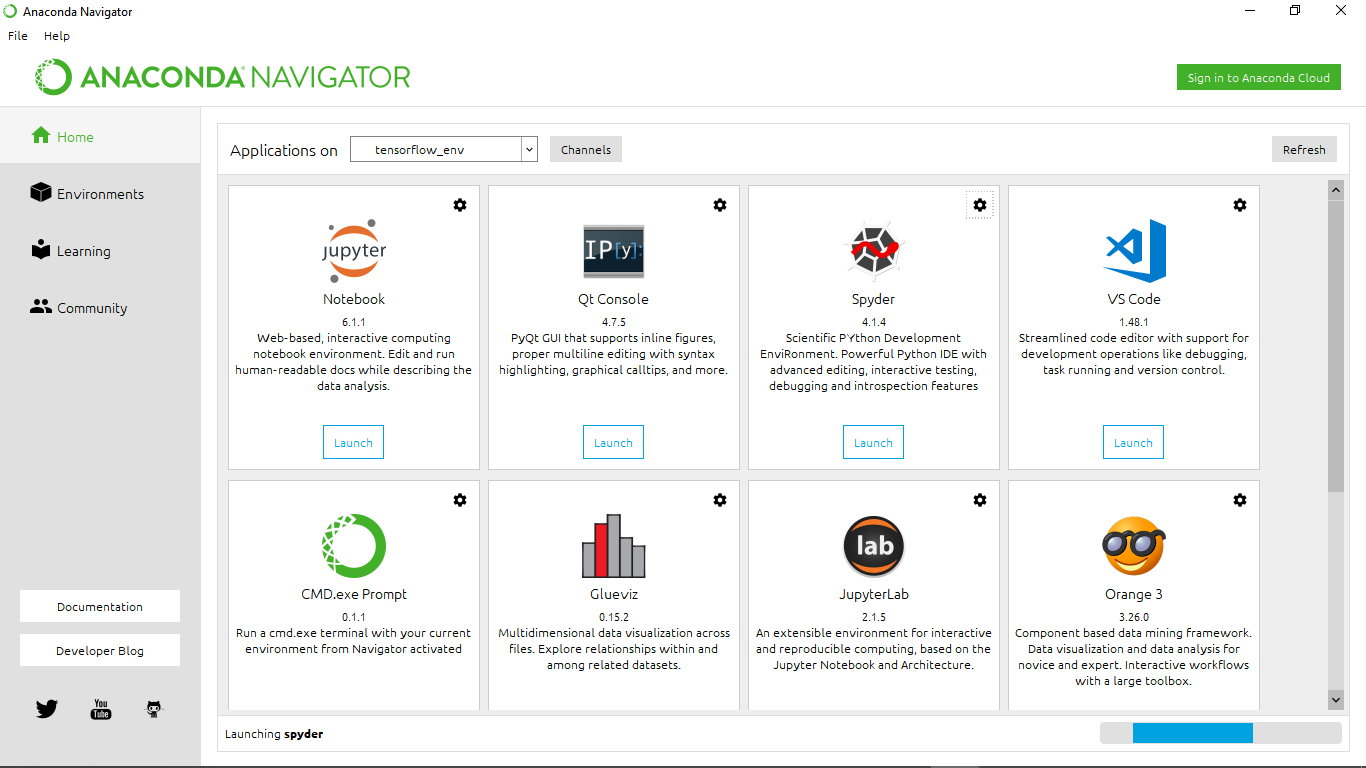


Figura 1: Entorno de trabajo ANACONDA

1. Abrimos la ventana de comandos Anaconda Prompt

Creamos un ambiente llamado **detector** por medio del cmd anaconda prompt escribiendo el siguiente comando: **conda create –n detector anaconda python=3.6**

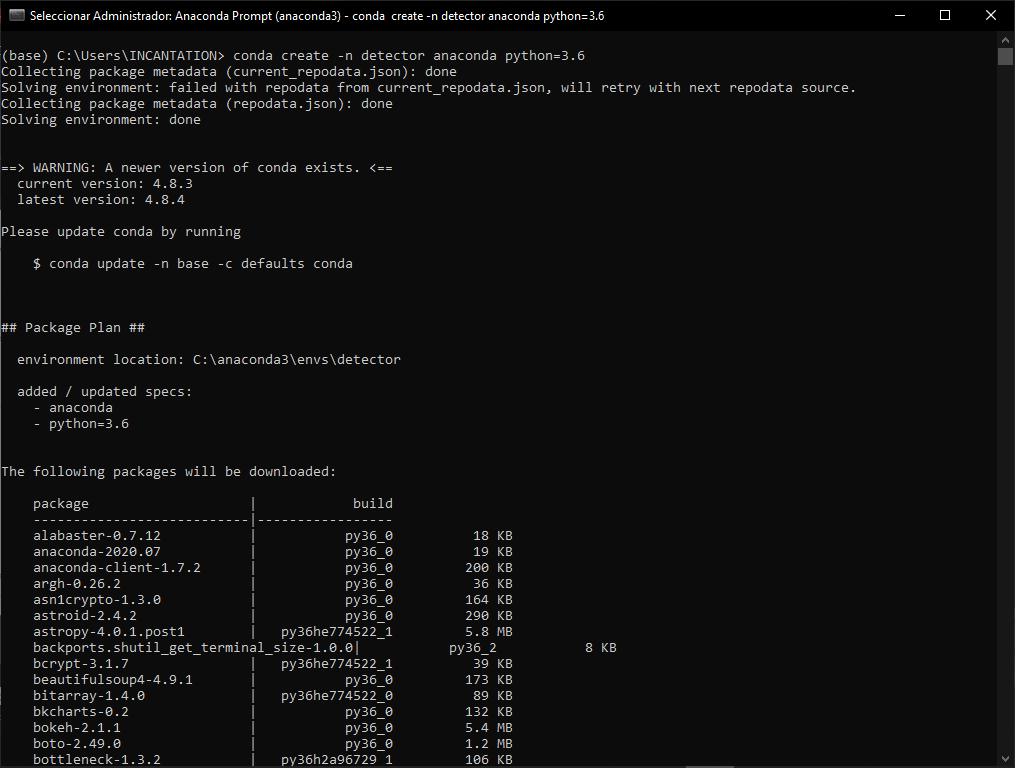


Figura 2. Creación del ambiente

Enseguida empezara a descargar los complementos para poder implementar el detector de objetos

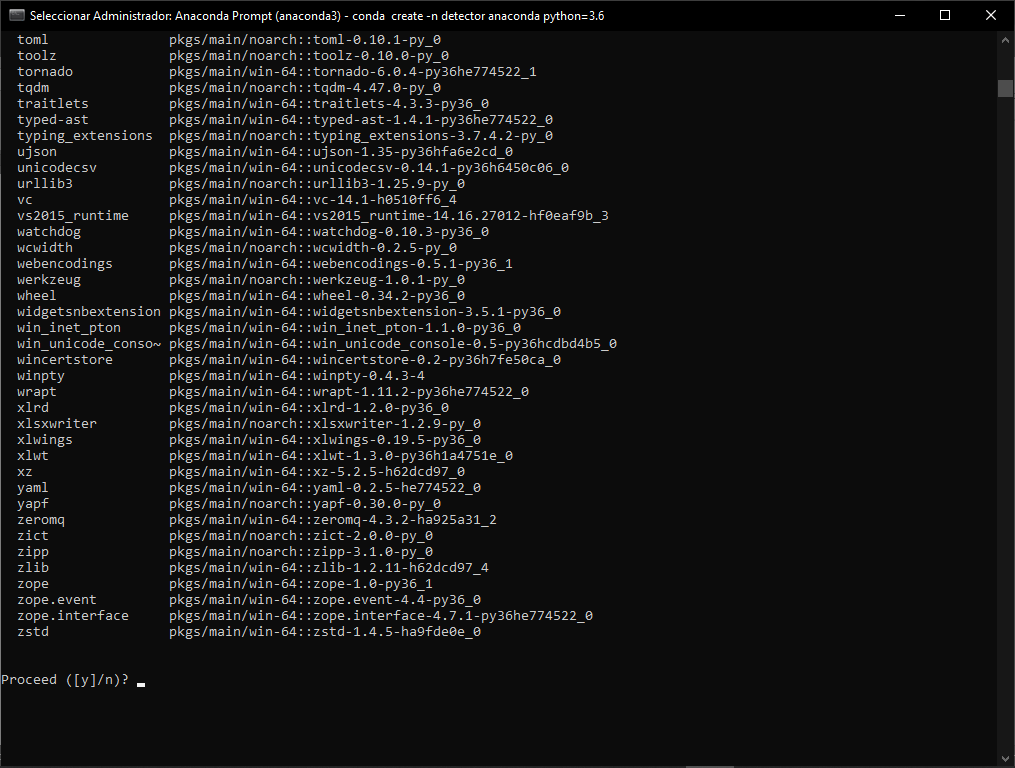


Figura 3. Descarga de complementos

Le damos en **[y]** para que se descarguen e instalen las librerías

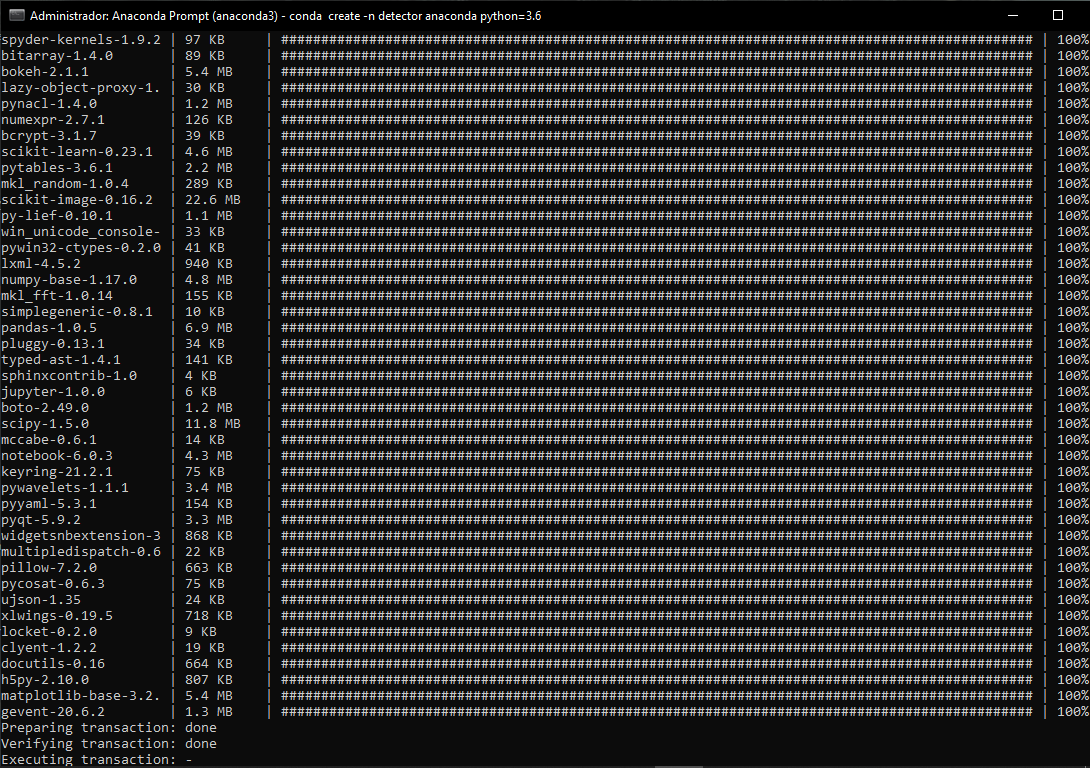


Figura 4. Instalación de librerías

Activamos el entorno que creamos llamado detector con el siguiente comando: **conda actívate detector**

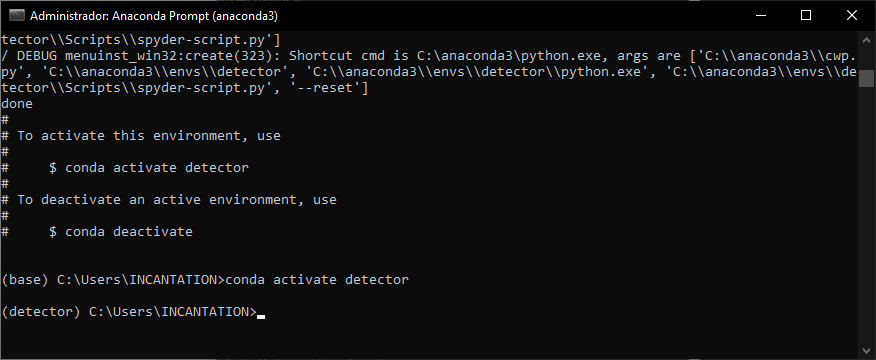


Figura 5. Activación del entorno creado

Ahora vamos a instalar las librerías y paquetes faltantes que no vienen instaladas por defecto: **pip install opencv-python**

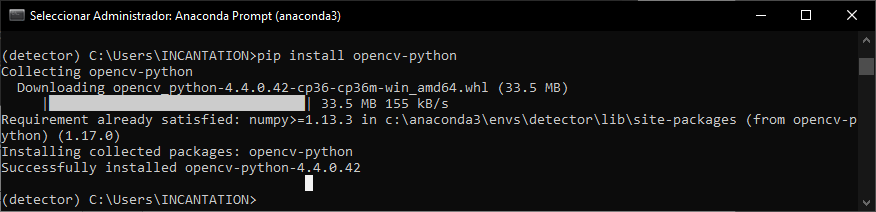


Figura 6. Instalación de librerías y paquetes faltantes

**pip install numpy**

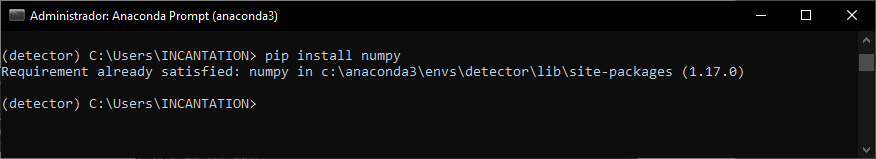


Figura 7. Instalación de numpy

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

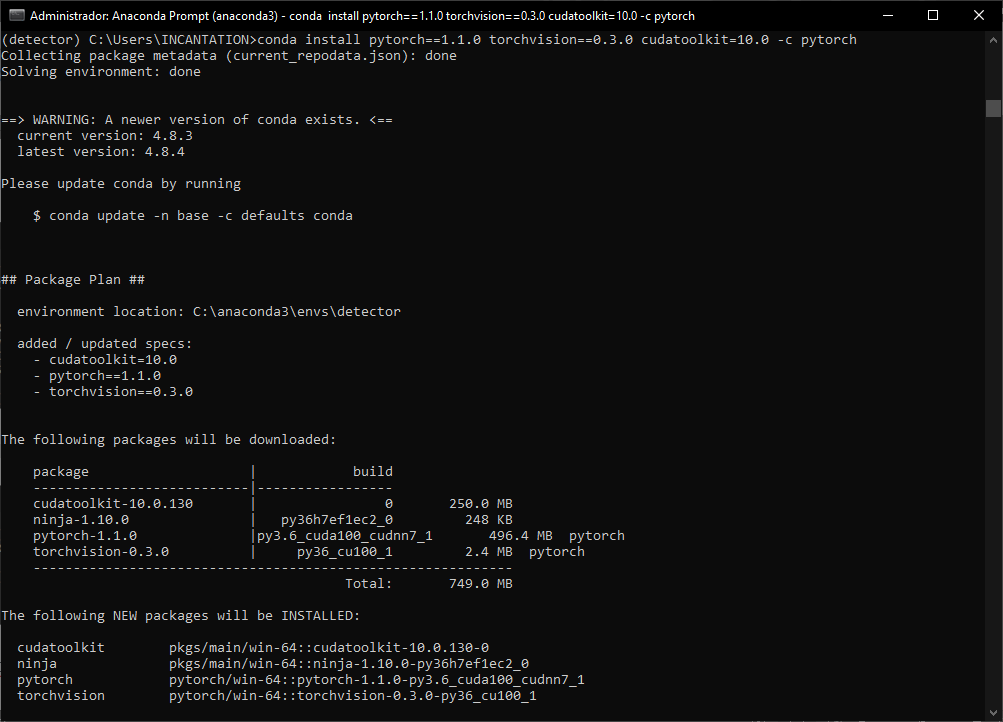


Figura 8. Instalación del torch

Escribimos **[y]** y damos enter para que se instale paquetes necesarios

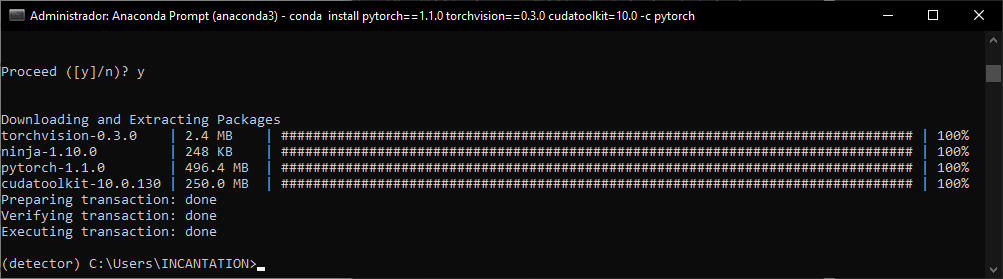


Figura 9. Instalación de paquetes necesarios

Instalamos otros paquetes ya pueda que estén instalados, pero lo verificamos: **pip install matplotlib**

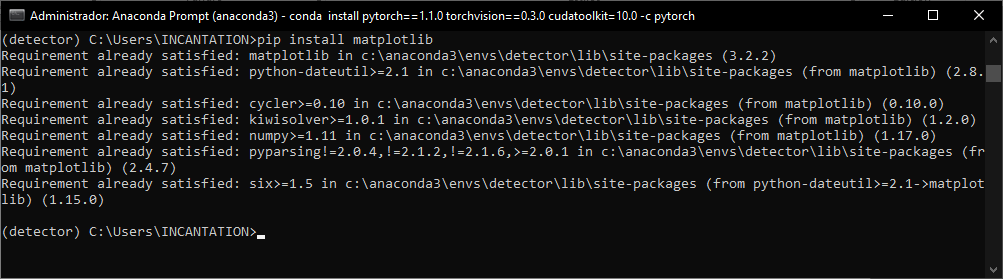


Figura 10. Verificación de paquetes necesarios

**pip install tensorboard**

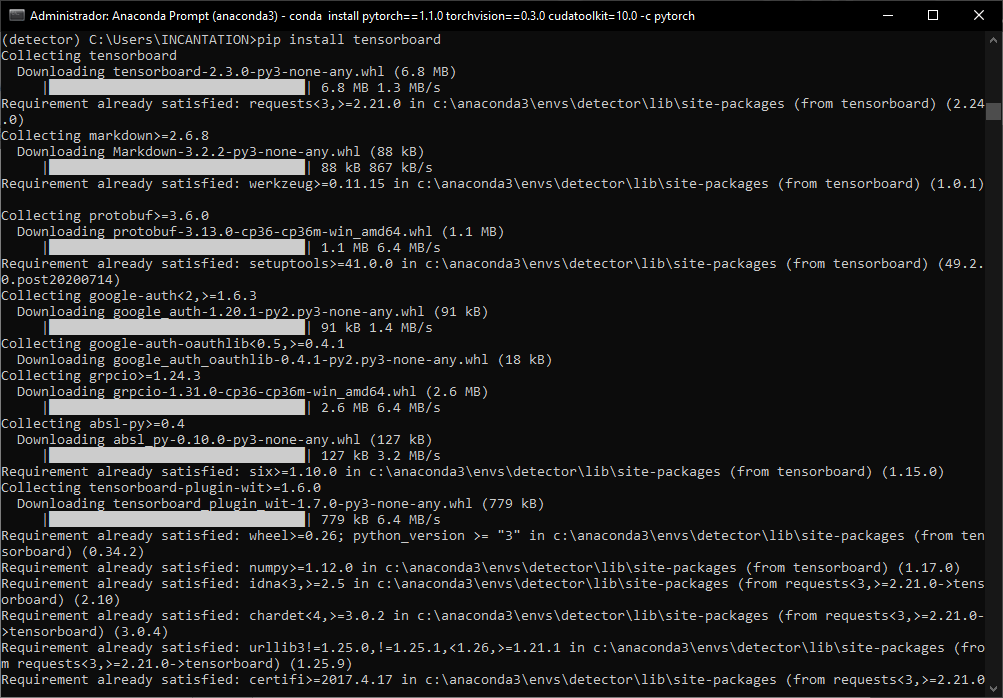


Figura 11. Instalar Tensorboard

**pip install terminaltables**

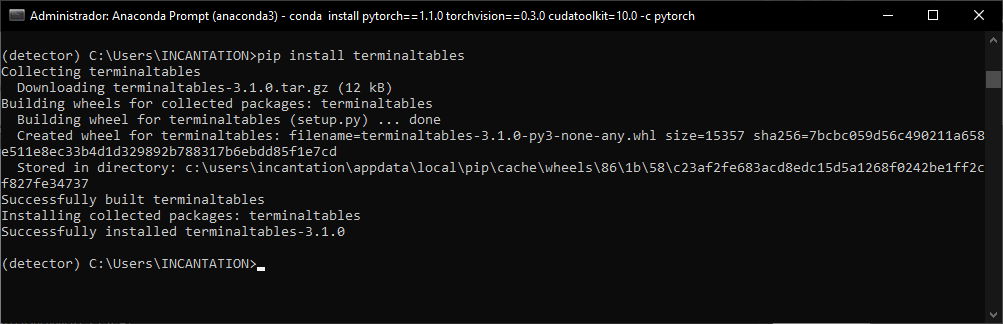


Figura 12. Instalar terminaltables

**pip install pillow**

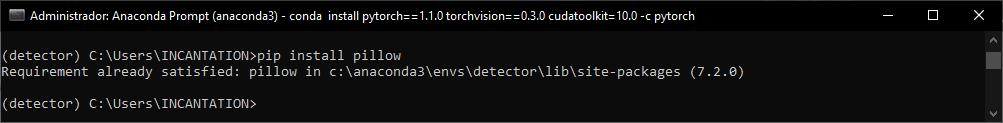


Figura 13. Instalación de pillow

**pip install tqdm**

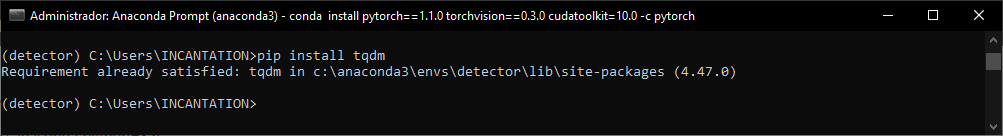


Figura 14. Instalación de tqdm

1. Agregamos el directorio para guardar un repositorio, escogemos el disco para crear una carpeta en este caso elegimos el disco “D:”, llamamos a la carpeta yolovideo y adentro de esta carpeta creamos otra carpeta llamada Yolo para poder guardar el clon del repositorio con imágenes ya entrenadas anteriormente ya creadas estas carpetas dentro del directorio D procedemos a escribir los siguientes comandos, y en el cuarto paso agregamos la dirección del repositorio a clonar

1) (detector) C:\Users\INCANTATION>**d:**

2) (detector) D:\>**cd "yolovideo"**

3) (detector) D:\yolovideo>**cd Yolo**

4) (detector) D:\yolovideo\yolo>**git clone**

**https://github.com/DavidReveloLuna/YoloV3\_video.git**

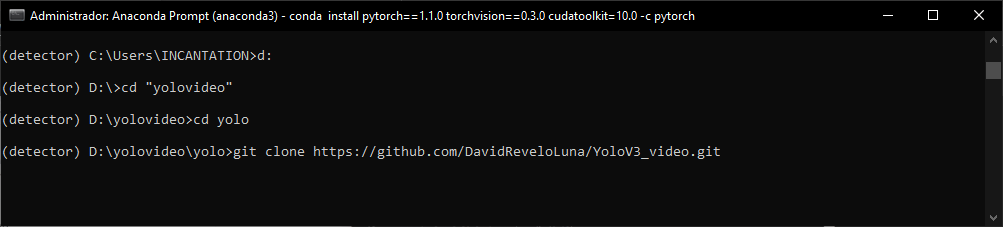


Figura 15. Clonar el directorio

Esperamos que se instale en nuestra carpeta creada en el disco

D:\yolovideo\yolo\**YoloV3\_video**

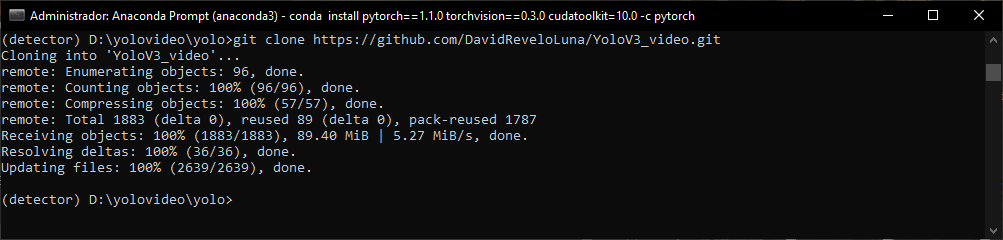


Figura 16. Instalación de la carpeta en el disco

Ahora ingresamos el nombre de la carpeta que se instaló del repositorio **YoloV3\_video**

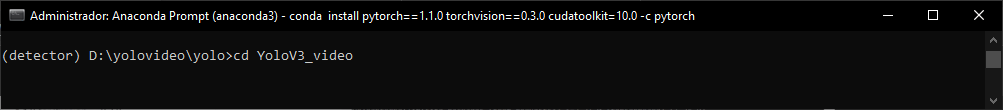


Figura 17. Carpeta creada en el repositorio

1. Descargamos los pesos del entrenamiento de la red por medio del navegador los siguientes links, al descargarse los copiamos y pegamos dentro de la carpeta **weights**

**Download weights for vanilla YOLOv3**

<https://pjreddie.com/media/files/yolov3.weights>

**Download weights for tiny YOLOv3**

<https://pjreddie.com/media/files/yolov3-tiny.weights>

**Download weights for backbone network**

<https://pjreddie.com/media/files/darknet53.conv.74>

Al descargarse los pegamos dentro de la carpeta **weights**

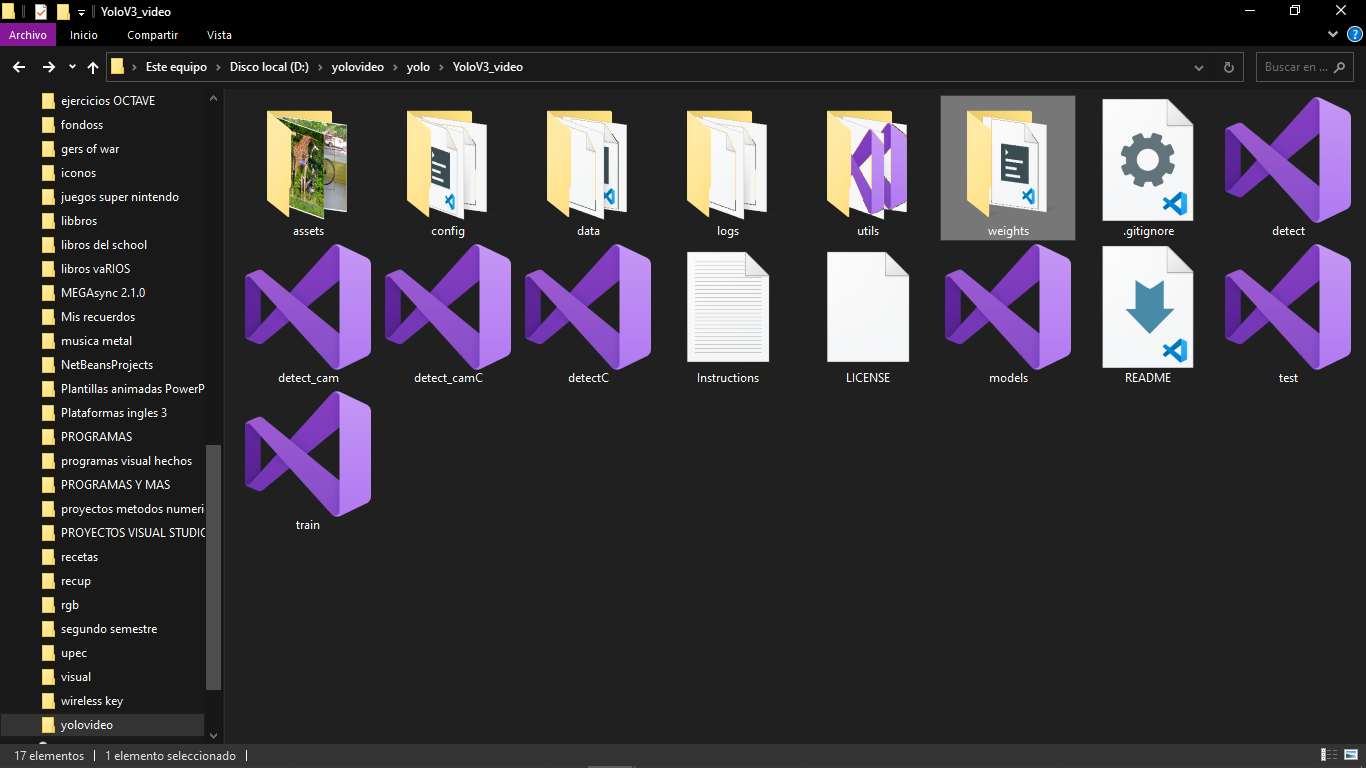


Figura 18. Descarga de pesos de entrenamiento de la red

Dentro de la carpeta YoloV3\_video creamos otra carpeta que se llame **output**

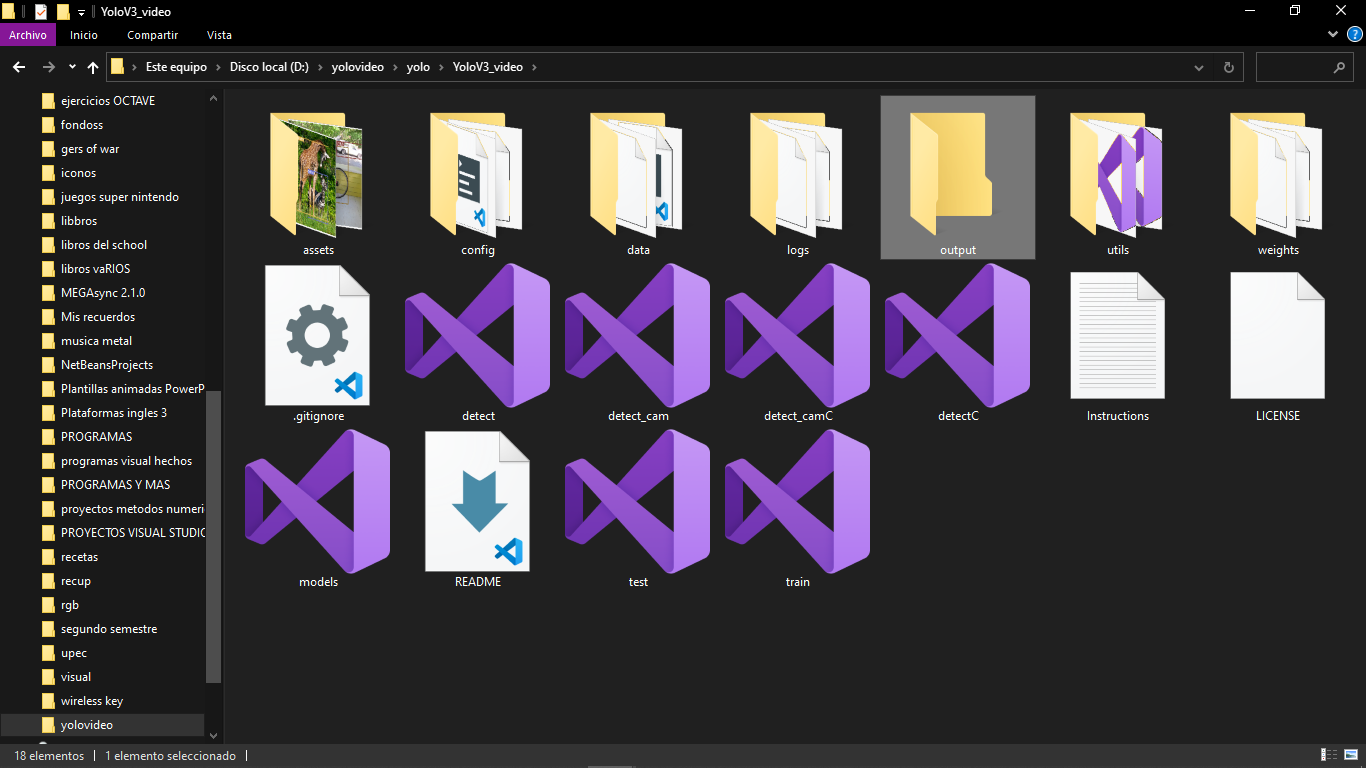


Figura 19. Crear nueva carpeta output

Y dentro de la carpeta output agregamos otra carpeta llamada **samples**

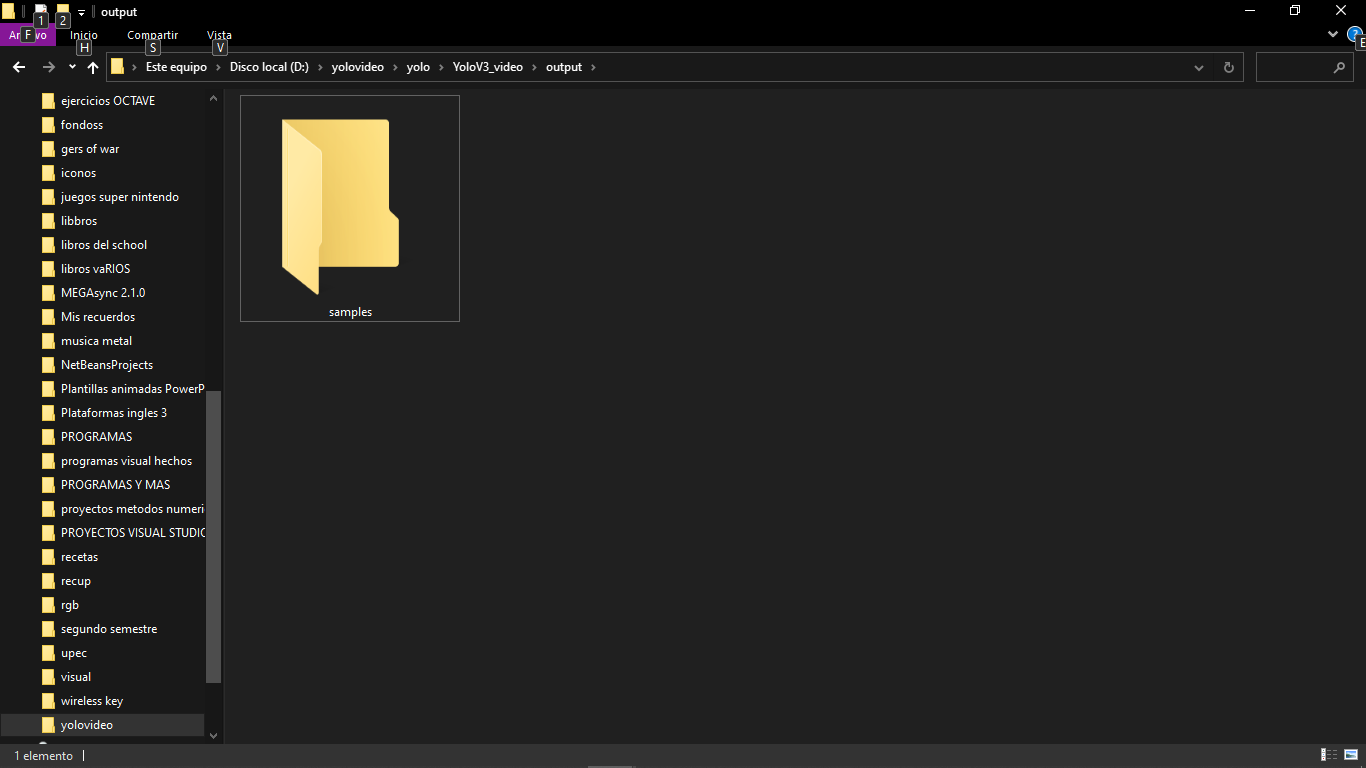


Figura 20. Crear carpeta samples

Ahora probamos si el proyecto se ejecuta **python detect.py --image\_folder data/samples/**

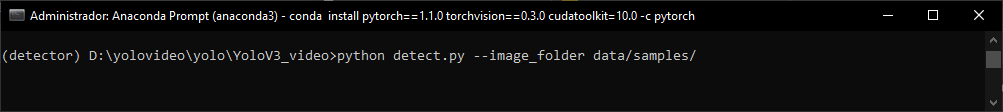


Figura 21. Comprobar si el proyecto se ejecuta correctamente

De esta manera empieza a procesar las imágenes de ejemplo

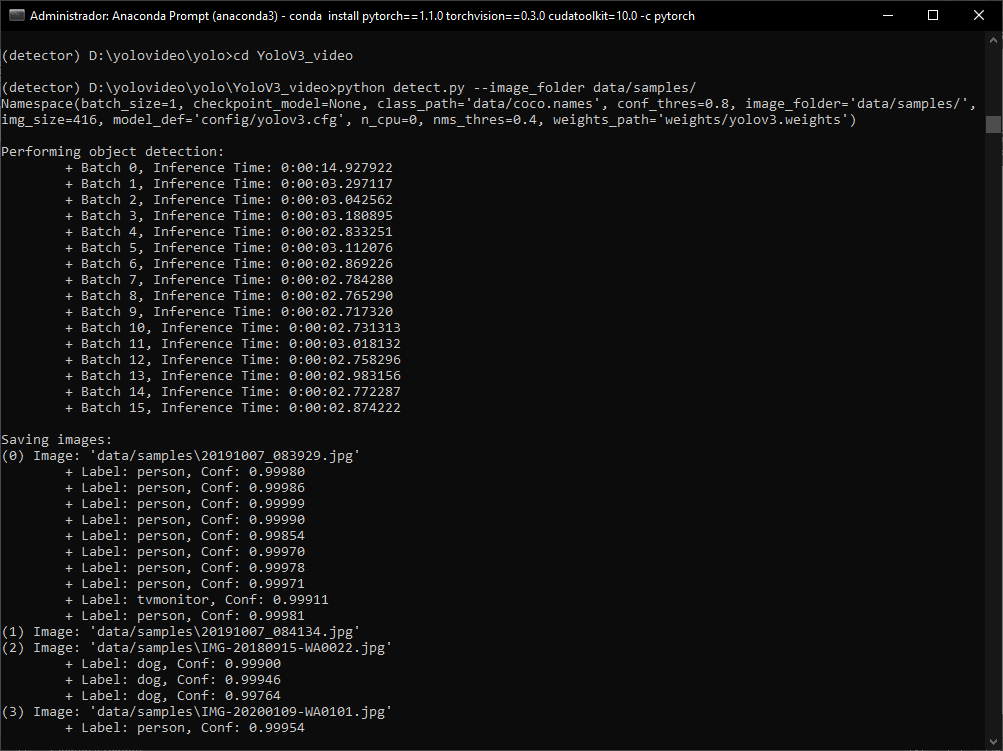


Figura 22. Procesamiento de imágenes

Ahora ingresamos **python detect\_cam.py** para acceder a nuestra cámara de video

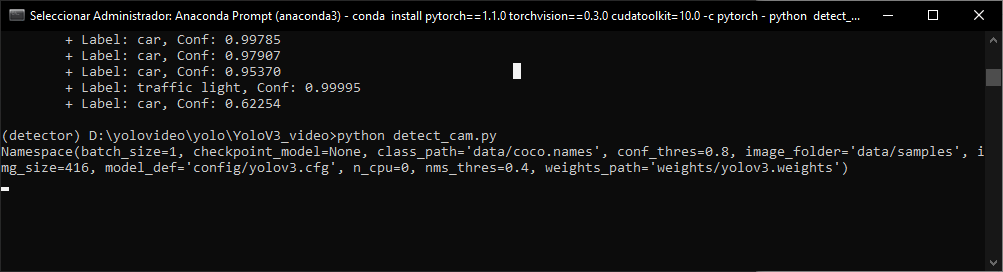


Figura 23. Paso de acceso a la cámara

1. Para activar el proyecto ingresamos por prompt (detector) escribimos lo que está en negrita



Figura 24. Ingreso al prompt

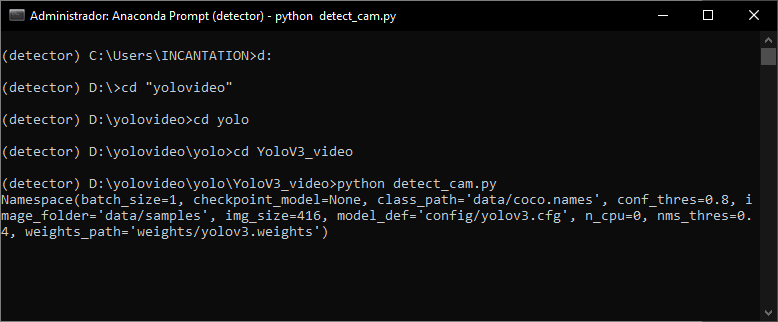


Figura 25. Activación del proyecto

1) (detector) C:\Users\INCANTATION>**d:**

2) (detector) D:\>**cd "yolovideo"**

3) (detector) D:\yolovideo>**cd Yolo**

4) (detector) D:\yolovideo\yolo>**cd YoloV3\_video**

5) (detector) D:\yolovideo\yolo\YoloV3\_video>**python detect\_cam.py**

6) damos clic en q para cerrar la detección

Al iniciar el programa puede detectar todos los objetos que están en su entorno de esta manera ira analizando dentro del repositorio de imágenes descargadas de las librerías YOLO entrenadas con anterioridad para poder predecir que objetos son exactamente.

**SANDBOX**

Cabe recalcar también que para nuestro proyecto hicimos uso de una herramienta de edición de desarrollo web online como es Sandbox, este ha sido muy debutado a la hora de edición online puesto que es muy cómodo y sencillo de usar, además que perite previsualizar el URL del proyecto tras cada línea de código creada. (Cabot, 2018) asegura que “una funcionabilidad muy útil de Sandbox es que puedes compartir el URL del proyecto y únicamente el usuario con ese URL puede acceder a él y modificarlo”. En nuestro caso:

Así se ve nuestro código en Sandbox, aquí contamos con la vista a las dependencias con las cuales se llama a los repositorios de las imágenes ya entrenadas.

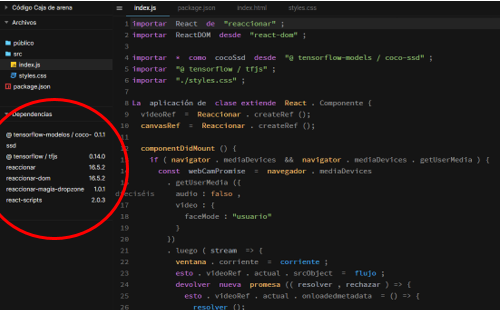


Figura 26. Dependencias de Sandbox

Además, también contamos con las clases que se emplean para que funcione el entorno de la detección de objetos.

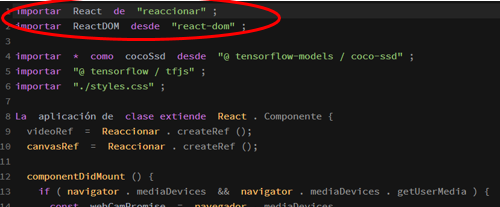


Figura 27. Implementación de clases en Sandbox

Y así sería la vista general de Sandbox. https://codesandbox.io/embed/z364noozrm?code mirror=1

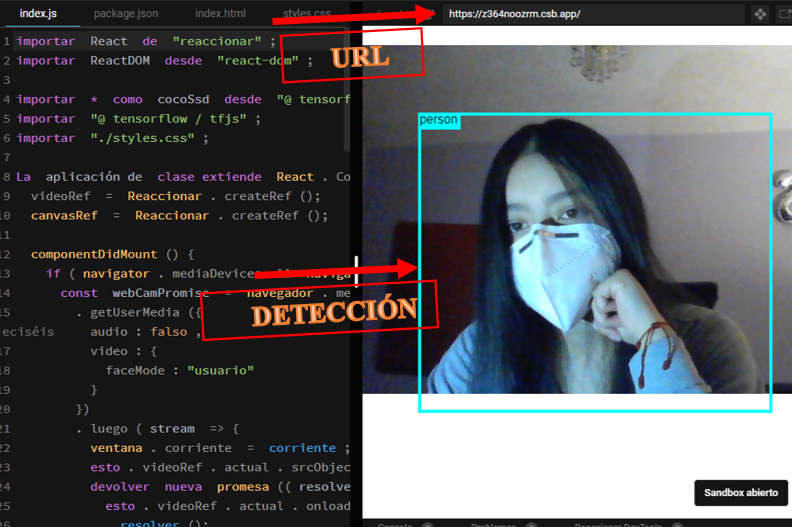


Figura 28. Vista general de Sandbox

# **CONCLUSIONES**

## CONCLUSIONES

* Gracias a la suite de código abierto Anaconda y los repositorios Yolo, se logró desarrollar exitosamente un algoritmo que reconozca objetos en tiempo real.
* Conseguimos instalar correctamente un código de visión artificial en nuestro ordenador con excelente funcionalidad.
* Se implementó datos visuales que ayudaron a que el proyecto de ejecute de manera muy rápida e intuitiva.

## RECOMENDACIONES

Como recomendaciones se tiene principalmente el uso y escritura adecuada de los comandos a usar ya sea para la instalación de paquetes necesarios o verificación de los mismos, así como también para el ingreso a la ejecución del proyecto a través de la cámara.

# **BIBLIOGRAFÍA**

Lawrence, G. (1963). *MIT Libraries*. Obtenido de https://dspace.mit.edu/handle/1721.1/11589

Miller, N., Mohan, T., Eichel, J., & Mishra, A. (2015). Obtenido de https://ieeexplore.ieee.org/document/7158929

Pérez, J. (Enero de 2019). Obtenido de https://repositorio.uta.edu.ec/bitstream/123456789/29182/1/Tesis\_t1535masc.pdf

Szeliski, R. (03 de Septiembre de 2010). Obtenido de http://szeliski.org/Book/drafts/SzeliskiBook\_20100903\_draft.pdf

# **ANEXOS**

## EVIDENCIAS



Figura 29. Reconocimiento de objetos mediante el proyecto desarrollado

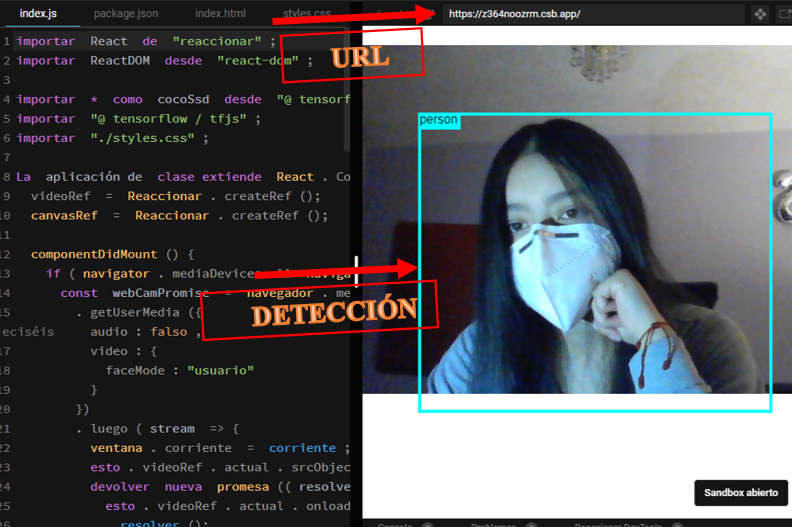


Figura 30. Reconocimiento de personas