**Historial y Manual de Usuario:**

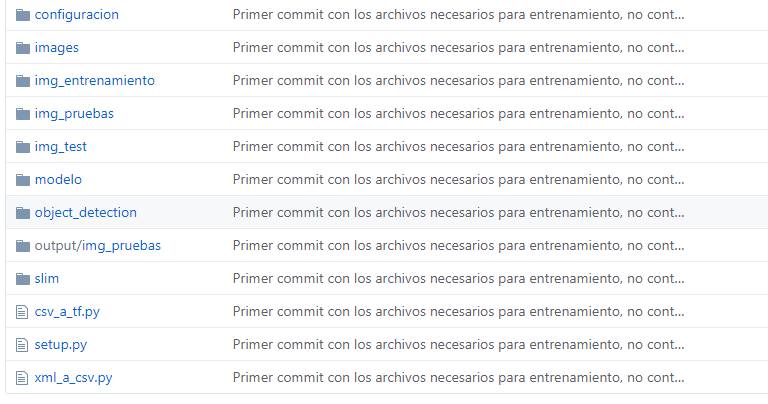
**Detección de Objetos (Aguacate) usando TensorFlow:**

El funcionamiento se trata de que el sistema al ingresarle una imagen debe de reconocer si es un aguacate hass o no, para eso se tiene un ambiente de aprendizaje para la red neuronal.

Se utilizó un código el cual bajamos de un repositorio de github que este tenía como especialidad reconocer tigres, nosotros lo remodelamos para que reconociera los aguacates has con un data set de aguacates hass.

Link del repositorio:

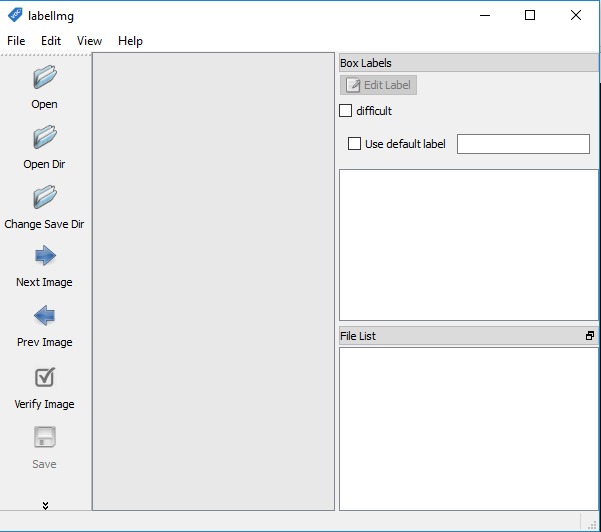
<https://github.com/puigalex/deteccion_objetos>

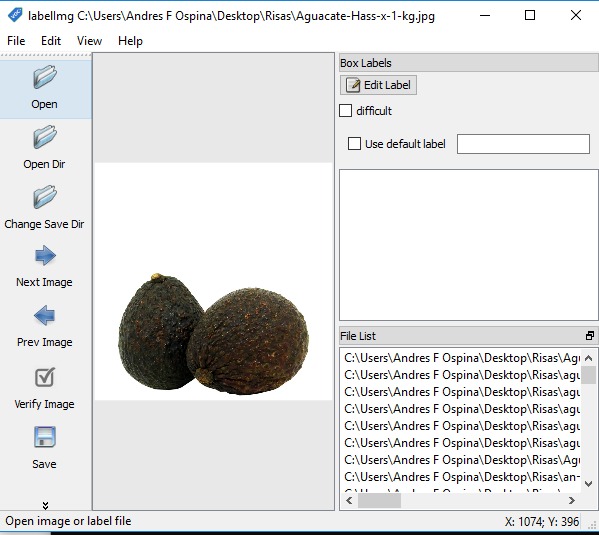


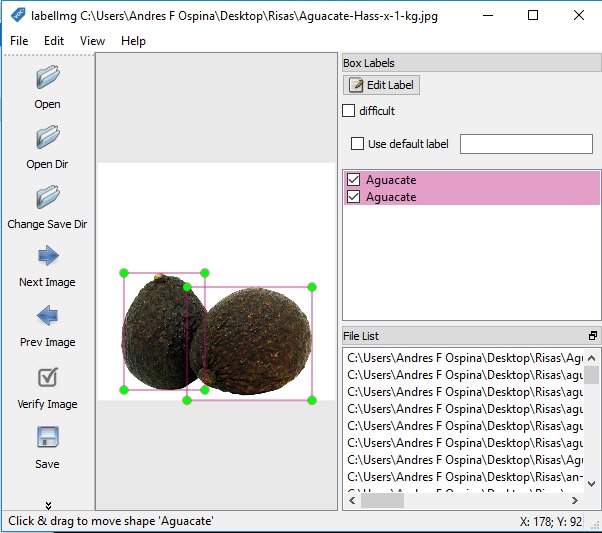
Carpetas del repositorio, las carpetas de imágenes acá están llenas con fotos de tigres y el programa solo reconoce a este animal, además hace faltan 2 carpetas con lo que serían archivos de entrenamiento.

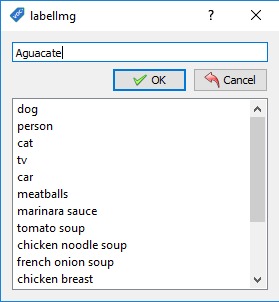
**Etiquetado de imágenes.**

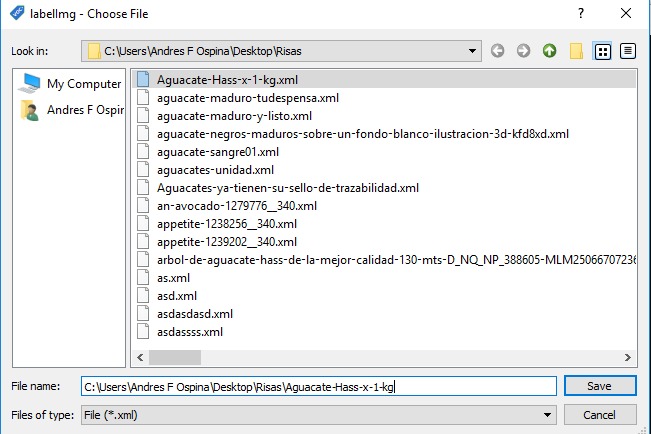
Este es el trabajo más complejo ya que se debe bajar un data set con las imágenes de los aguates en nuestro caso se tomaron 210 imágenes para ser etiquetadas. Para esto utilizamos una herramienta de etiquetado de imágenes (Labeling) la cual toma como base una imagen y nosotros debemos crear una etiqueta donde se debe seleccionar el lugar donde está ubicado el objeto(aguacate) y darle un nombre a la clase (Esta clase de la etiqueta debe llamarse igual que en los archivos de configuración del archivo de detección\_objetos en nuestro caso se llama **Aguacates**). Ya las imágenes etiquetadas se le da a guardar y esto generara un archivo .XML. el proceso se debe hacer para cada una de las imágenes.











Se muestra la interfaz de cómo funciona esta aplicación para el etiquetado de imágenes.

Url de descarga: [https://tzutalin.github.io/labelImg/](https://www.youtube.com/redirect?redir_token=2sl6iL7FEAc7By7C-c8ju-xaRg18MTU2MzMzMzU0OEAxNTYzMjQ3MTQ4&q=https%3A%2F%2Ftzutalin.github.io%2FlabelImg%2F&event=video_description&v=SJRP0IRfPj0)

**Alimentar el entorno:**

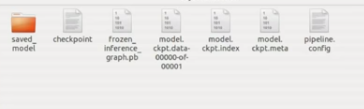
Ahora con las imágenes etiquetadas procedemos a copiar y pegar tanto las imágenes como los archivos .XML a la carpeta detección\_objetos/imágenes, ahora debemos dar el 85% del data set a la carpeta imag\_entrenamiento y el otro 15% a imag\_test.

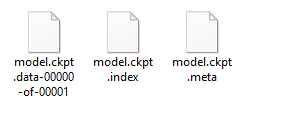
Ahora necesitaremos la ayuda de un modelo preentrenado y para eso utilizaremos faster\_rcnn que es un modelo que hace una detección de objetos mucho más rápida y para ello utilizarmos la versión faster\_rcnn101\_resnet101\_coco, descargamos este entorno pereentrenado y es un .zip ahora se descomprime y vamos a proceder a copiar 3 archivos que se necesitan.

(Entorno preeentrenado en github)

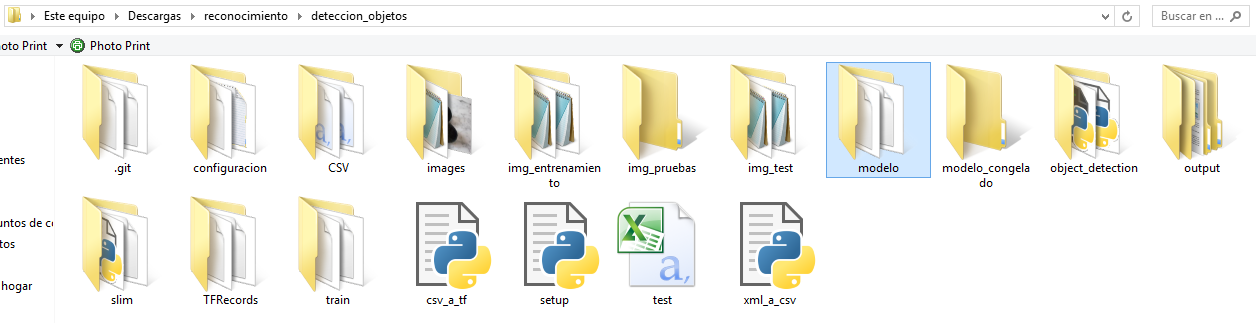
Link de descarga: <https://github.com/tensorflow/models/blob/master/research/object_detection/g3doc/detection_model_zoo.md>

Carpetas a copiar (las tres que dicen modelo)



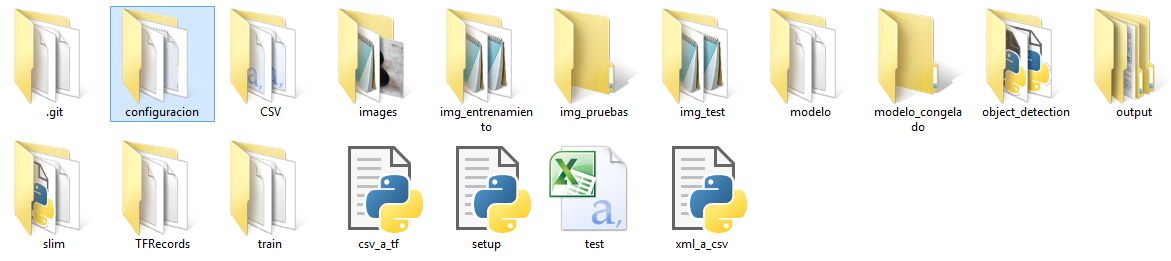


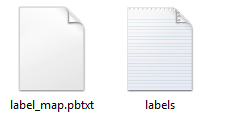
Se deben copiar en la carpeta modelo la cual está ubicada en detección\_objetos



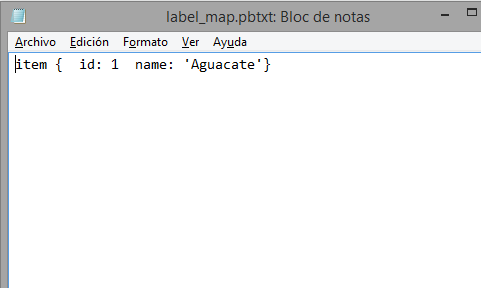
Debemos modificar unas cuantas cosas:

Entrar a la carpeta configuración:

  
Luego encontraremos 2 archivos:

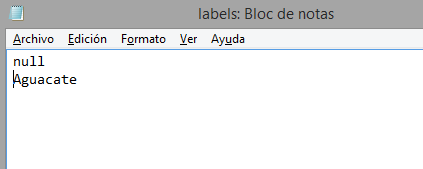


Ahora modificaremos label\_map primero

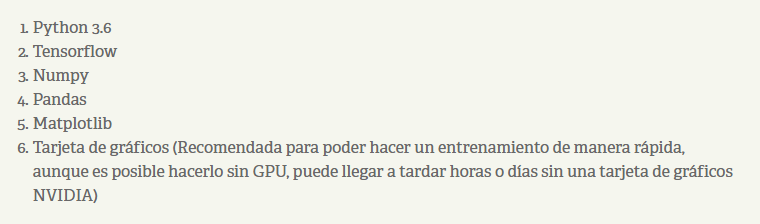


Debemos crear un ID para la clase con el correspondiente nombre a la etiqueta ya dada

Paso siguiente abrir label para anunciar las clases y colocar un dato null por la razón para cuando se ingrese una imagen que no corresponda este no la clasifique



Ahora tenemos los requerimientos para poder correr este código:



Antes de poner a aprender a nuestro sistemas se debe deben crear 2 carpetas TFRecords y Train las cuales nos servirán para la conversión de los archivos .xml a .csv y .record.

**Comandos para el sistema:**

El primer comando me sirve para la creación de un PATH en Python ya que si no se utiliza genera un error con una librería creada en object\_detection.

Para Linux: **export PYTHONPATH=$PYTHONPATH:`pwd`:`pwd`/slim**

Para Windows: **set PYTHONPATH=$PYTHONPATH:`pwd`:`pwd`models/slim**

Ahora se coloca a transformar los archivos:

**python xml\_a\_csv.py --inputs=img\_test --output=test**

**python xml\_a\_csv.py --inputs=img\_entrenamiento --output=entrenamiento**

**python csv\_a\_tf.py --csv\_input=CSV/test.csv --output\_path=TFRecords/test.record --images=images**

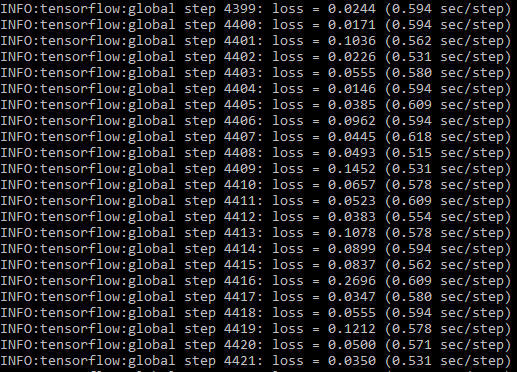
**python csv\_a\_tf.py --csv\_input=CSV/entrenamiento.csv --output\_path=TFRecords/entrenamiento.record --images=images**

observasion: cuando se trabaja con tensorflow siempre se utiliza Python 3.X, en caso tal que tenga ambas versiones de Python (2 y 3.X) se agrega un 3 a cada comando quedando como python3 …

**Entrenamiento:**

Es la parte más pesada ya que exige mucho rendimiento de CPU y GPU para que el sistema aprenda. Este sistema se diseñó para aprender en 20000 pasos y para reconocer un objeto entonces para poder ejecutar este entrenamiento se debe dar el siguiente comando:

**python object\_detection/train.py --logtostderr --train\_dir=train --pipeline\_config\_path=modelo/faster\_rcnn\_resnet101\_coco.config**



La imagen anterior representa el sistema en entrenamiento ya después de 4400 pasos

**El checkpoint:**

Un punto de segundo plano son los checkpoints lo cual me permite después de un cierto tiempo hacer un reentreno con más información partiendo de lo que ya se tiene y para eso se debe utilizar este comando:

**python object\_detection/export\_inference\_graph.py --input\_type image\_tensor --pipeline\_config\_path modelo/faster\_rcnn\_resnet101\_coco.config  --trained\_checkpoint\_prefix train/model.ckpt-684 --output\_directory modelo\_congelado**

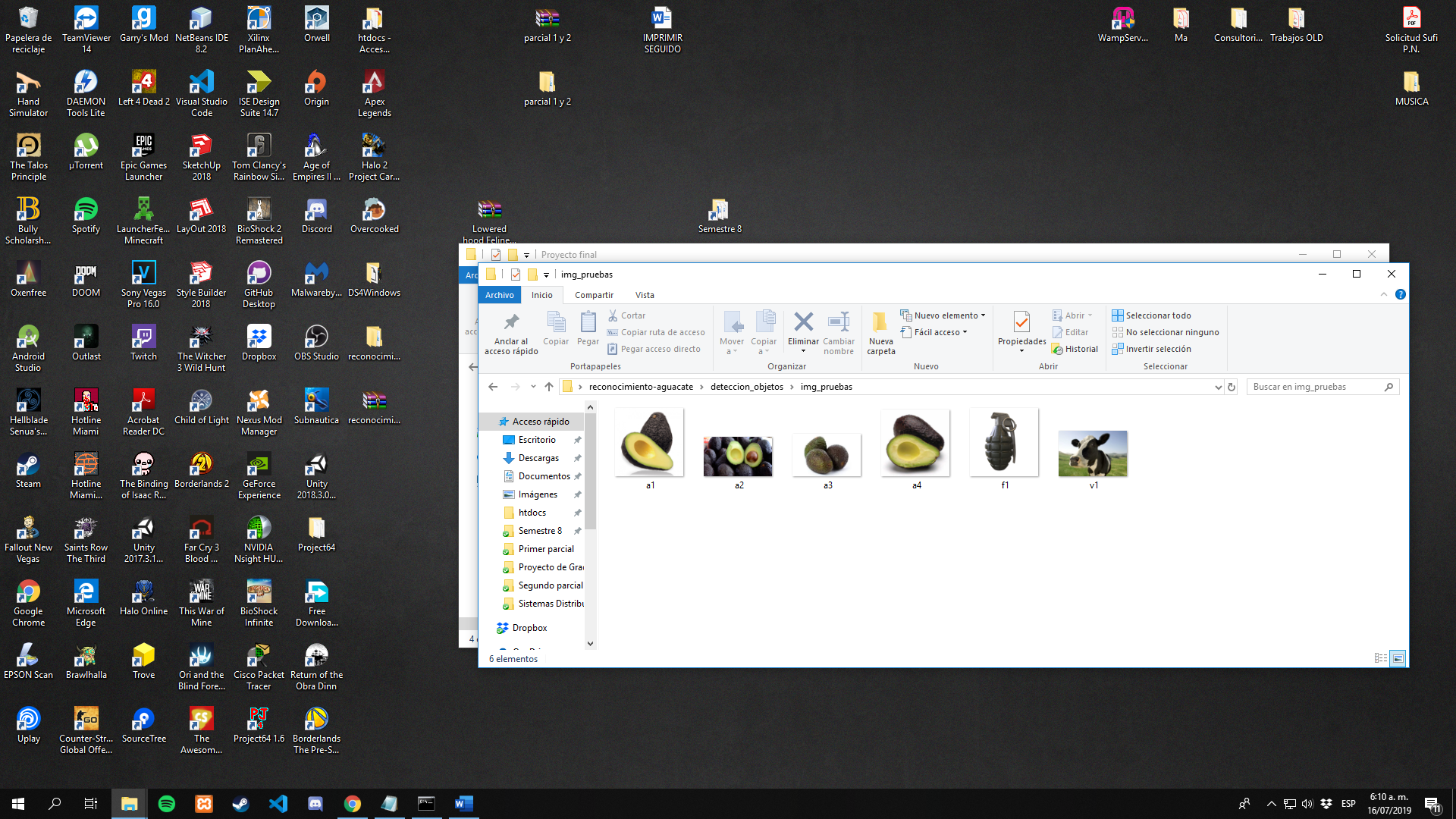
**Ejecución para la predicción:**

Este último paso es de cuidado porque debemos tener las imágenes cargadas en la carpeta img\_prueba ubicada en la carpeta detección\_objetos y de ahí obtendremos las predicciones en la carpeta de outputs. Este es el comando a ejecutar:

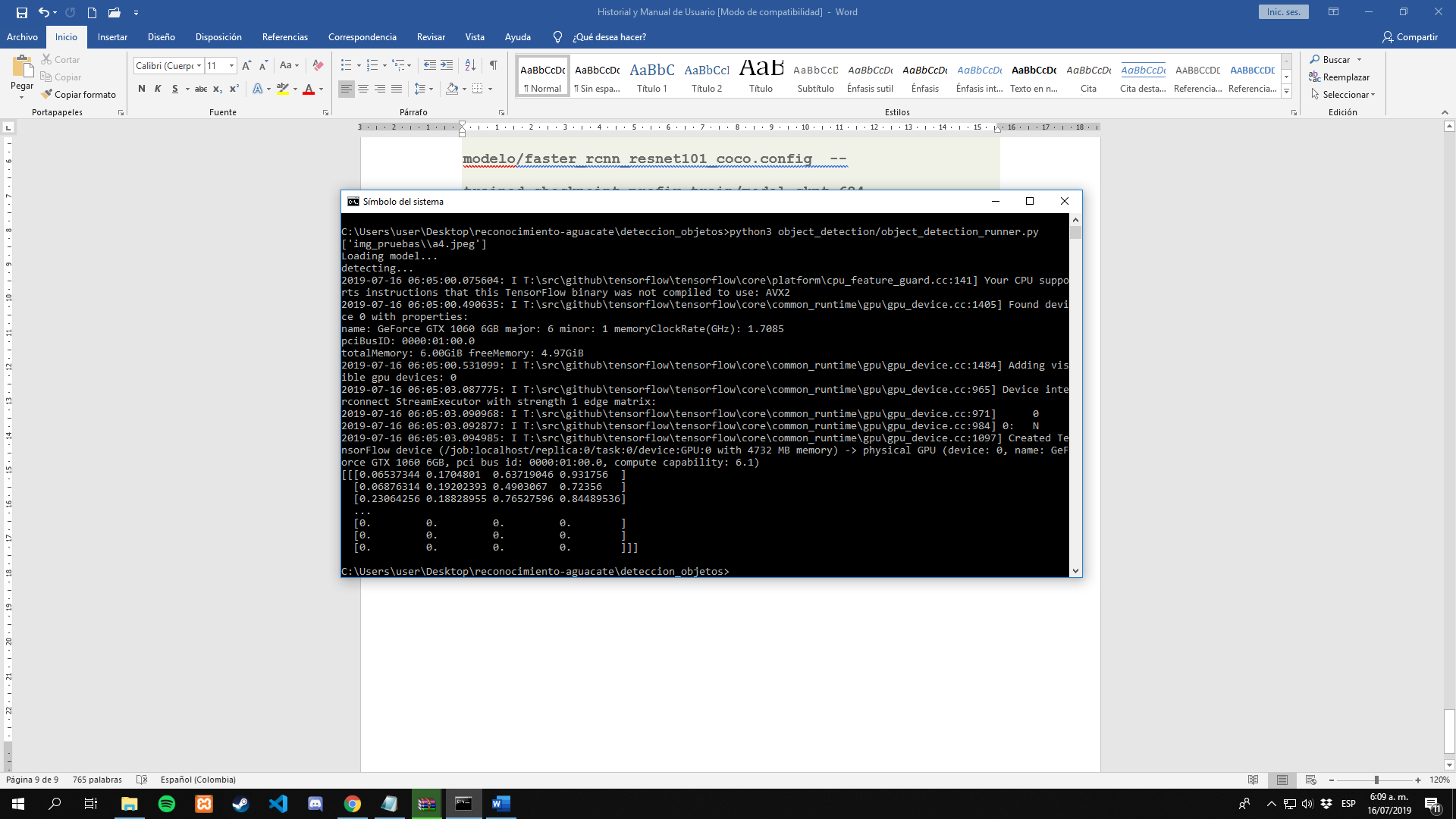
**python object\_detection/object\_detection\_runner.py**

**Resultados:**

Siendo las imagénes de pruebas:



Y los resultados:



Se evidencia en los primeros resultados que clasifica de forma correcta un aguacate hass maduro (Resultado diferente de 0), mientras que en las de abajo se puede ver que no es un aguacate (Granada y vaca). Demostrando que el modelo y la predicción son correctos.

En este Link podemos encontrar la red entrenada con los aguates:

<https://drive.google.com/open?id=1f0pc1cGZd82Hb4Kf_DaFNxEObg_2nXfR>