**MODULO DETECTOR DE PLACAS**

**Necesidad**: Se requiere crear un Script que permita detectar, identificar y exportar la lectura de las placas de los vehículos.

**Alcance**: Se debe permitir conectarse a una cámara IP donde se hará toda la toma del video.

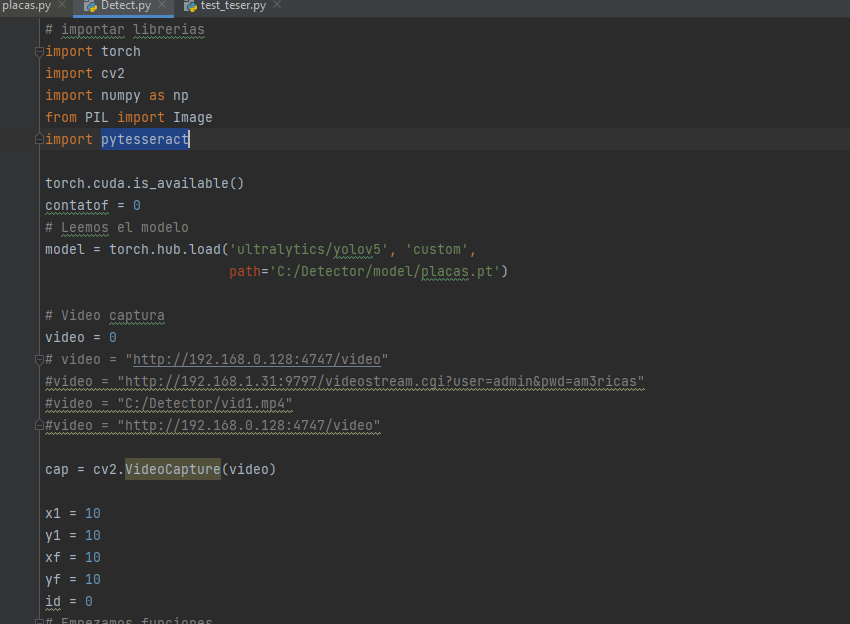
**Desarrollo**: Se realizó uso de python 3.8 y las librerías siguientes:  
**Torch**: Permite hacer el uso de detectar objetos usando visión artificial mediante **Yolov5**, el cual a **Yolov5** se le pasa un modelo creado de objetos identificados, el cual permite posteriormente saber que debe distinguir.

**Cv2**: **Opencv** como **Yolov5** permite hacer uso de visión artificial, la diferencia es que con **Opencv** podemos hacerle tratamiento directo a esa imagen, en este caso un video, alterando su tamaño, color, brillo o formato de salida de imagen a binarizado.

**Pill from Image**; Lo que permite hacer esta librería es convertir un arreglo de imágenes o videos, en una sola pero cumpliendo con cierto criterios.

**Pytesseract:** Esta librería nos permite usar una funcionalidad llamada OCR(Optical Character Recognition) el cual nos permite reconoce texto de imágenes que se les pase en el momento pero para hacer uso de el, debe estar instalado con sus componentes.

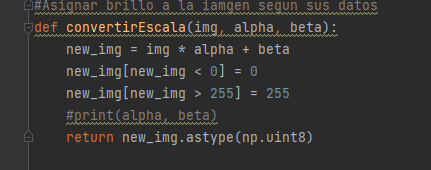
Codigo:



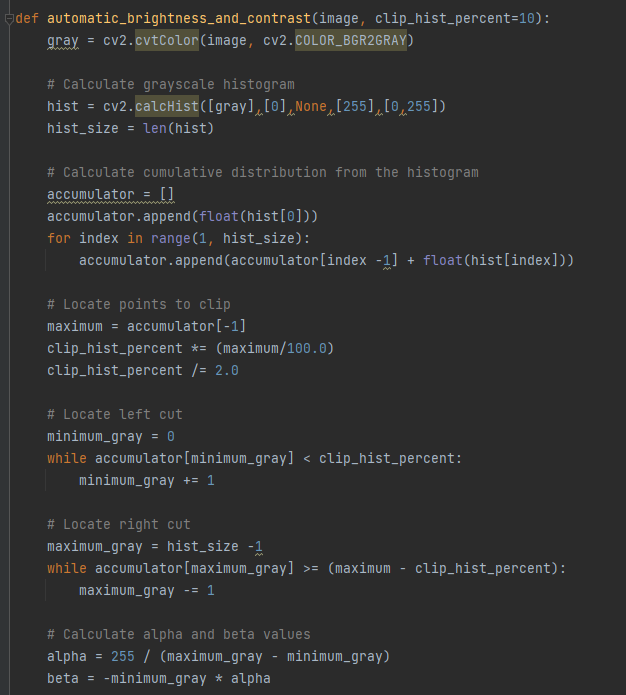
Primeramente instalamos librerías que nos permitiera hacer usa de CUDA que es usar la tarjeta grafica para mejorar la ejecución.  
Después de importar las librerías y hacer uso de CUDA, llamamos nuestro modelo creado que se creo y descargó en la ruta en el disco C:/Detector.

También tenemos nuestra variable donde va la URL de donde tomaremos el video.

El marco o información del video quedara guardado en la variable cap.

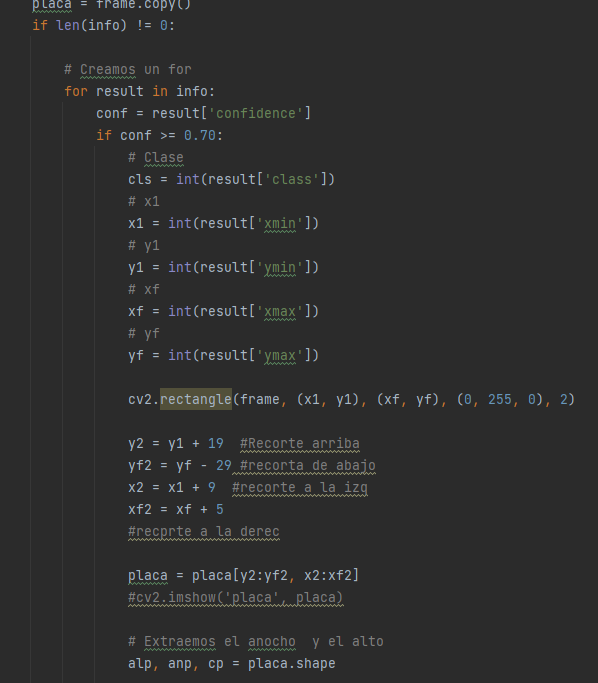


Encontramos la primera función el cual permite adecuar el nivel de brillo de la imagen, pero los valores a operar se reciben más adelante.

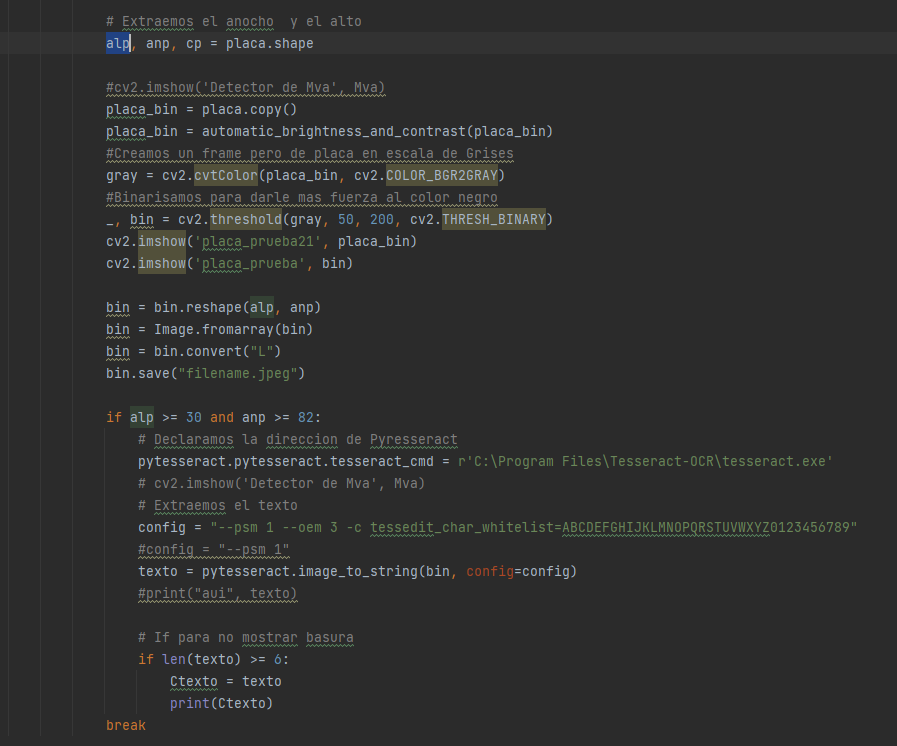


Esta siguiente función nos permite recorrer cada capa del video pero convertido a escala de gris posteriormente recorre cada capa en donde nos dirá que valor de la capa tiene según su pixel. Al final tenemos el valor de alpha y beta, eso valores son los que nos permiten modificar el brillo de la imagen.

Ahora después de saber el valor de brillo y la imagen modificada, se procede a recorrer el video, sabemos que nuestro video viene en el arreglo en la variable llamada “cap”, debemos extraer el **frame** del video pero también su valor bolean(variable “ret”) . Ahora le pasamos el ‘video’ o frame a nuestro modelo para que empiece a realizar la validación y si encuentra coincidencias se hará lo siguiente.  
En nuestra variable “info” extraerá las coordenadas donde se encuentra la coincidencia de video con nuestro modelo.



Si la variable tiene información es decir, hay una coincidencia en el modelo, recorreremos el arreglo y extraeremos la clase a la cual corresponde nuestro modelo y las coordenadas del objeto, posteriormente con “Rectangle” señalamos donde se realiza la detección. A las coordenada de recorte le ajustamos manualmente un poco de longitud para que el recorte quede un poco más ajustado. Cuando llamamos a “placa[y:yf, x2:xf2]” estamos realizando el recorte del video en esa posición especifica.

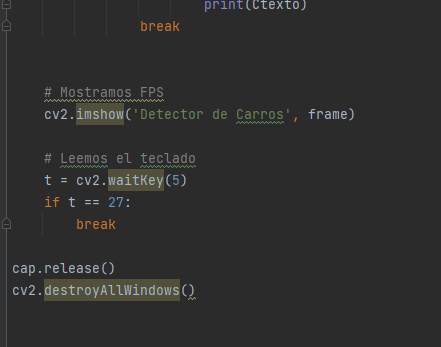


A la variable “placa” es decir el recorte del video donde debe estar la placa, se le extrae en alto y el ancho de la misma.

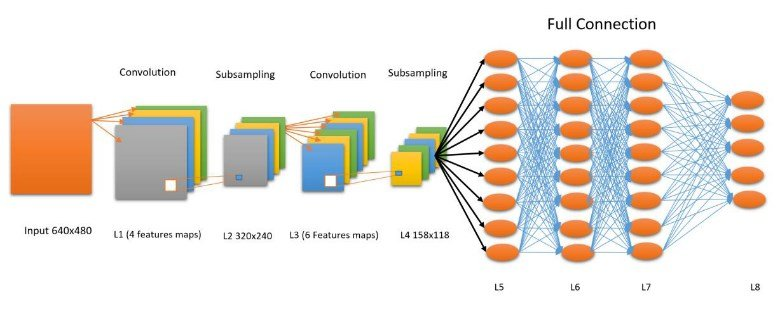
Ahora se procede a realizar una copia del frame llamado “placa” y se asigna a otra variable llamada “placa\_bin” luego todo frame se pasa a la función donde ajustamos el brillo y el contraste. Ese frame mejorado se le hace una conversión a escalas de gris y posteriormente se binariza la imagen, que en pocas palabras es dejar solo un valor de imagen 1 o 0 pero respetando el color de la imagen, después convertimos el arreglo o video en una imagen mediante bin este proceso ayuda a eliminar o “limpiar” la imagen para tener una bien legible.

Hacemos una verificación del tamaño del frame o por así decirlo saber si lo que se ha identificado tiene

el tamaño de una placa, siguiendo con el proceso usamos Pytesseract donde le pasamos la imagen del video convertido y la configuración de detección de caracteres y por ultimo se almacena en una variable, se verifica que sea mayor o igual a 6 es decir su longitud.



con esto ultimo, mostramos nuestro video o frame de nuestra cámara y si deseamos salir, se crea un condicional que espera la tecla “Esc” y finalmente destruye toda las sesiones.

Nota:  
  
**Redes neuronales convolucionales.  
**

Su función básicamente lo que permite hacer es identificar objetos según las imágenes creadas igual que nuestro cerebro, la diferencia es que se limita el proceso por la cantidad de neuronas, esto quiere decir que pueden haber 100 neuronas y 20 verifican la esquina de la placa, a su vez esas 20 le preguntan a las otras 80 que están detectando y si el resultando es el correcto significa que todas las neuronas dijeron que eso que detectaron es lo que necesitaban. Siempre las neuronas se comunican entre ellas para un resultado final.