**INSTITUTO TECNÓLOGICO DE LA PAZ**

**DIVISIÓN DE ESTUDIOS PROFESIONALES**

**Reporte semanal de residencias Semana 04**

**“Reconocimiento y seguimiento de objetos en entornos controlados”**

**CARRERA**

**Ingeniería en Sistemas Computacionales**

**PRESENTA**

**17310793 – Eloy Antonio Clemente Rosas**

La Paz, Baja California Sur, México, 04 de Abril del 2021

# Introducción

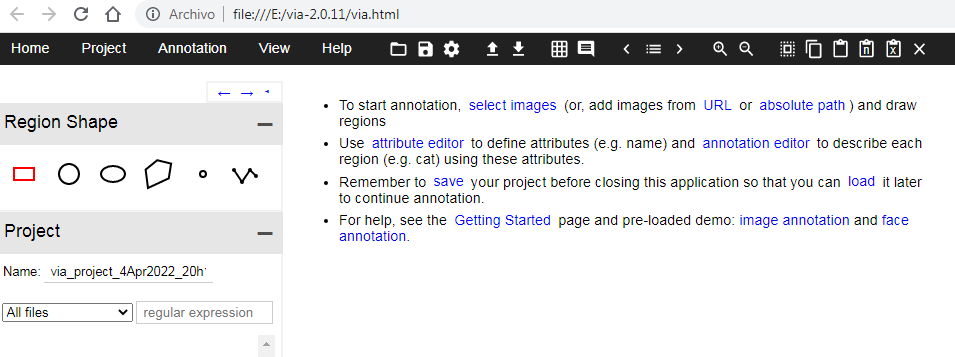
El presente documento contiene las actividades, y aprendizajes realizados durante la décima semana en mi residencia profesional. Esta semana sirvió para automatizar muchos procesos, para segmentar, y entrenar la red neuronal, se está teniendo cada vez un mejor resultado respecto a los objetivos, el segmentar con bordes fue una gran mejora al reconocimiento de la red.

# Desarrollo de actividades

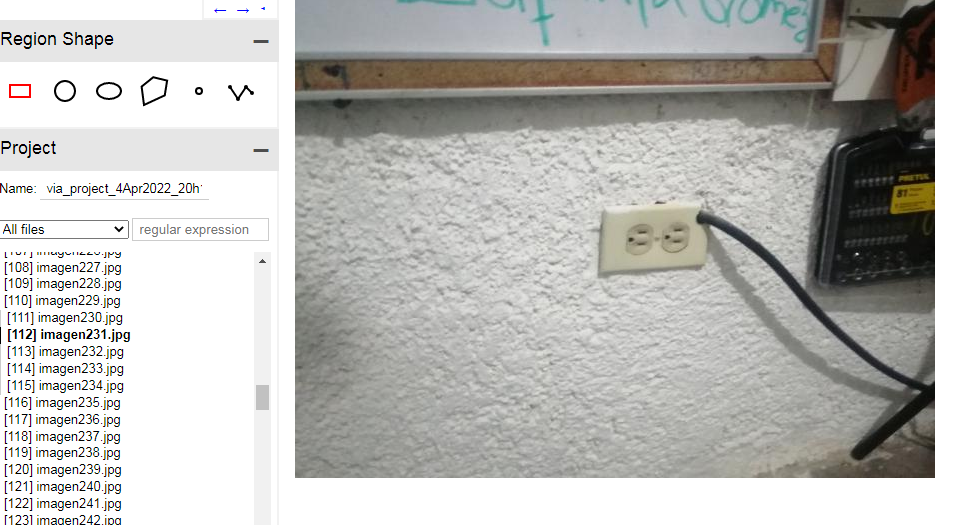
Esta semana realice una segmentación de todas las imágenes de nuevo, porque faltaron imágenes por segmentar sobre algunos objetos con los que la red tenía confusión, para esto use este script llamado VIAS que se encuentra en el siguiente repositorio:

<https://github.com/EloyFtw/Residencia-profesional/tree/main/Programas/04-04-2022/vias>

El entorno de trabajo es el siguiente:



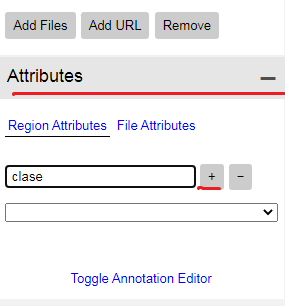
Se cargan las imágenes desde “select images”, y es posible navegar entre imágenes, por ejemplo esta imagen:



La forma de segmentar es seleccionar una forma, en esta caso polygon, y es importante seleccionar por con todo y bordes:

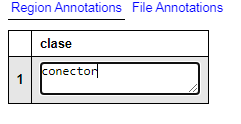


Ahora se debe seleccionar estas opciones la primera vez que se carga una imagen y se segmenta:



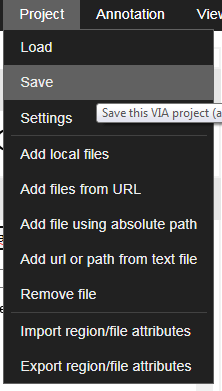
Esto para indicar a que va a pertenecer a clase la anotación de la región.

Por cada región que se seleccione se debe indicar el nombre de clase como se ve a continuación.

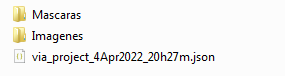


Después de segmentar todas las imágenes a las cuales se va a crear una máscara.

Para esto se guarda de la siguiente manera el proyecto:



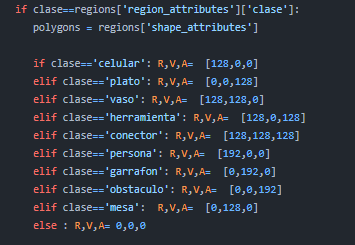
Se va a generar un archivo Json que debe guardarse en la ruta donde estén las carpetas de imágenes.



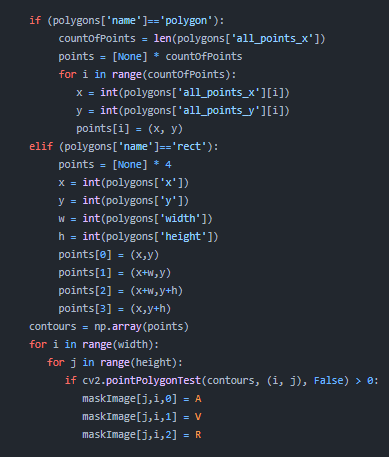
Ahora para realizar las máscaras se usa el script de Python que nació a partir de la idea del script que proporciono en donde cada sección esta vez del Json que guarda las regiones de cada imagen sea coloreado el código está en la siguiente liga:

<https://github.com/EloyFtw/Residencia-profesional/blob/main/Programas/04-04-2022/Crear%20Mascara.py>

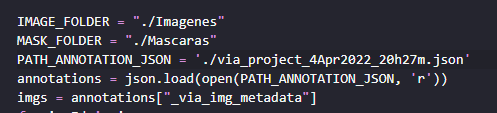
Donde en la primera región del código se definen los colores para cada objeto:



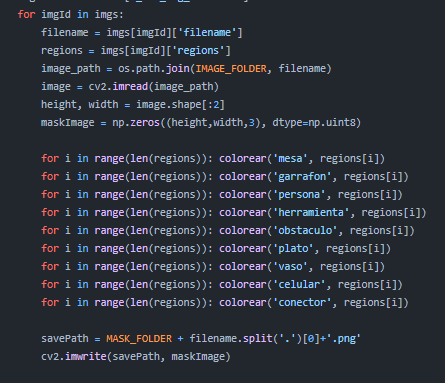
Donde “name” es el nombre del objeto a dibujar, esto así para tener un orden entre el coloreado de los objetos y poder dibujar los objetos que van encima, por ultimo.



Con la siguiente sección de código se carga el directorio de imágenes, el archivo Json con las regiones guardadas y el directorio destino de las mascaras.

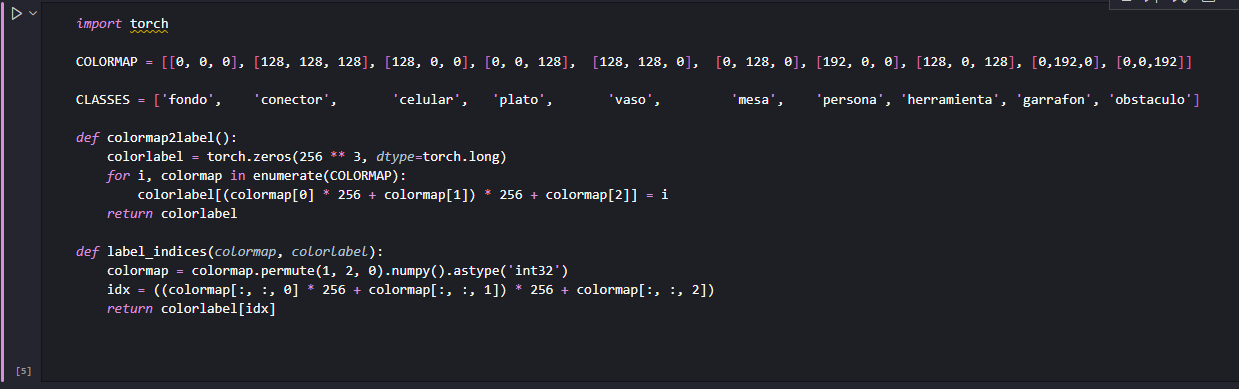


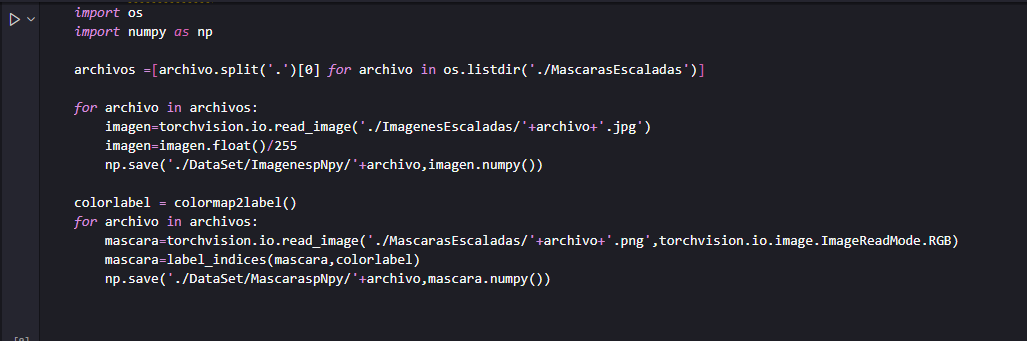
Por último se procesan las imágenes dando orden a los objetos a los que se les va a colorear cada región y se guarda la máscara en formato png.



Ahora deben procesarse las imágenes para convertirse en arreglo Numpy.

Primero se definen los colores nuevos, para generar el índice al cual va a pertenecer cada color.

Se convierte de jpg o png a Numpy las máscaras e imágenes.



En esta sección de código solo se modificó donde se indica el margen de error para que se convierta en el color que más se aproxima en los índices.

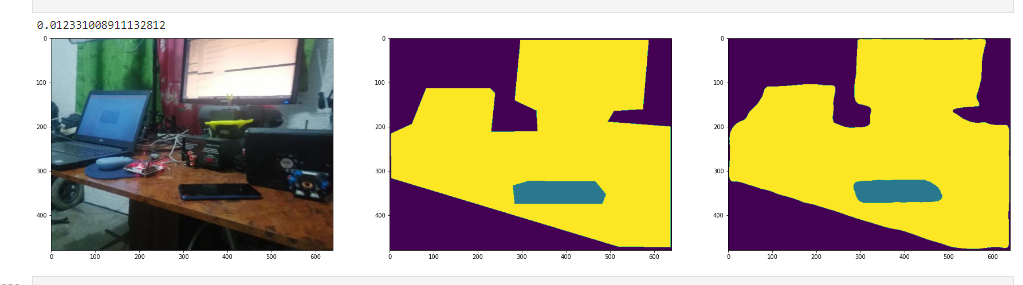


Es para escalar las imágenes. Realice dos entrenamientos, una solo con 640x480 y otro con 320x240.

El código para el entrenamiento con imágenes de 640x480 está en la siguiente liga a 100 épocas:

<https://github.com/EloyFtw/Residencia-profesional/blob/main/Programas/04-04-2022/PyTorch%20Segmentaci%C3%B3n%20Sem%C3%A1ntica.ipynb>

# Conclusiones

Como se puede apreciar la red sufrió de un sobre-entrenamiento aun así los resultados son sorprendentes teniendo muy buenos resultados pero algunos detalles surgieron logrando identificar en su mayoría la mesa de manera correcta.

El modelo entrenado con 320x240. No se guardó en mi memoria pero los resultados fueron bastante similares solo con la diferencia enorme de tiempo en entrenamiento.

Pero tengo cargados los modelos por si necesita hacer pruebas:

Link del modelo de 320x240.

<https://drive.google.com/file/d/1Mr3Sitr14AzYzrY4WE1eWapCU-5KMM0X/view?usp=sharing>

Esta versión tiene nombre diferente porque guardaba la mejor versión de cada época del entrenamiento quedándose con la mejor versión y evitando un poco el sobre entrenamiento. Lastimosamente no guarde el código pero puedo volver a configurar la red para que haga esa función.

Link del modelo de 640x480.

<https://drive.google.com/file/d/1nKjvuuqCBaUDvaMhYYZJjqkZdh4uFDO8/view?usp=sharing>

Se cargan con el código del modelo, hasta el final está el ejemplo.

Al parecer faltaron imágenes de cerca de cada objeto, además de una mejor segmentación porque algunos objetos confunden a la red aun.