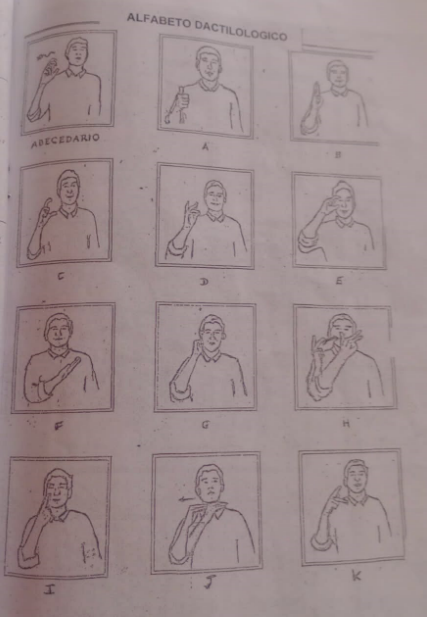
Desarrollo del proyecto Detección de Señas en LSA

# Generación del Dataset

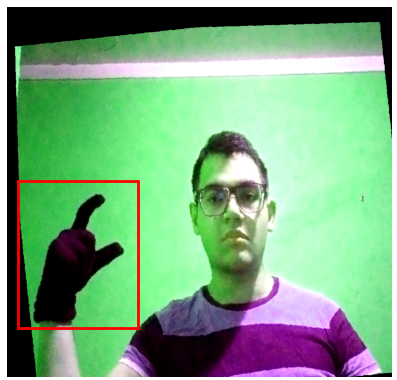
* **Captura de las Imágenes**: Para el armado del dataset, fue necesario sacar imágenes propias de cada seña en el alfabeto del Lenguaje de señas Argentino.

Para el mismo se capturaron 217 imágenes propias, realizando cada seña en un promedio de 8 veces, salvo aquellas que contienen movimiento, las cuales se han realizado en las diferentes posiciones que puede ocupar la seña.

Debido a la cantidad de variantes que pueden haber en el mismo LSA, se ha utilizado como base las siguientes imágenes:

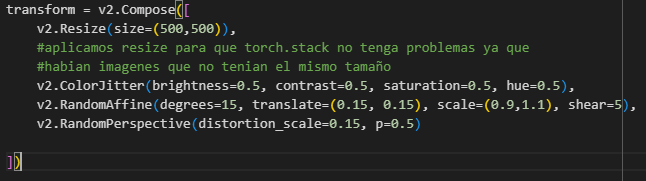


Finalmente, se utilizó la herramienta [VGG Image Annotator](https://www.robots.ox.ac.uk/~vgg/software/via/via_demo.html) para definir las bounding box, y obtener el archivo csv, con el nombre del archivo, y las coordenadas de la bounding box.



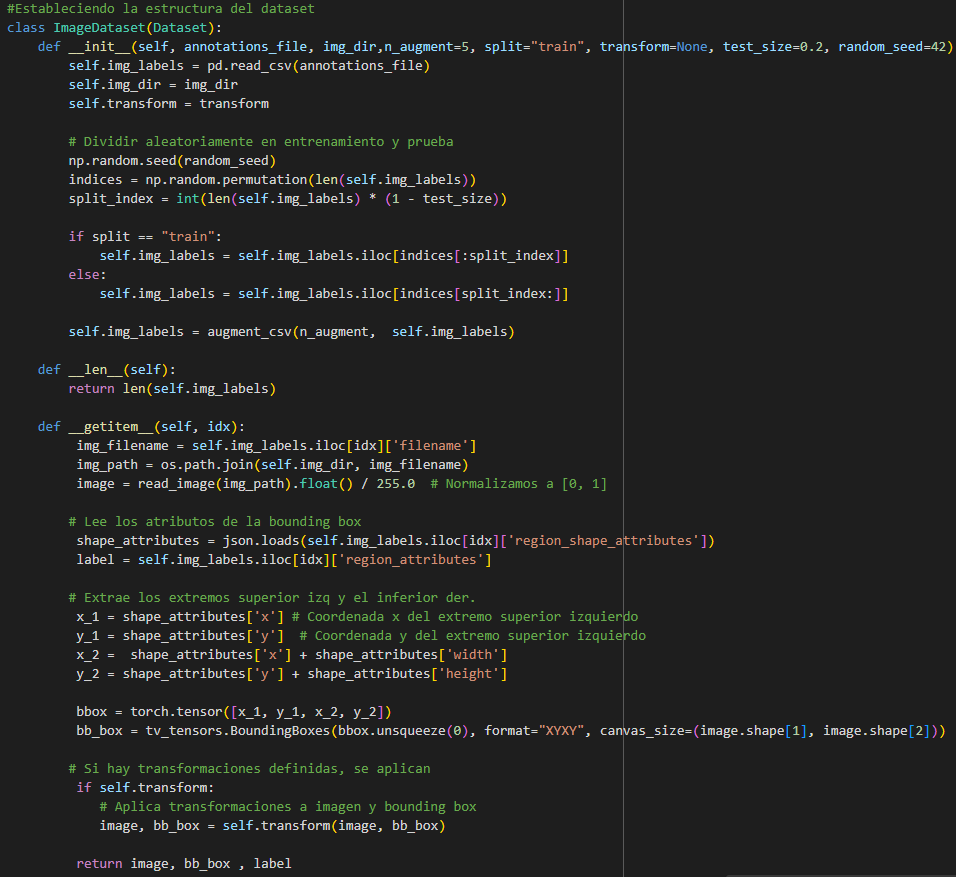
* **Augmentation**: Debido al reducido tamaño del dataset, se han realizado transformaciones, como redimension, cambios de color, de rotación, perspectiva, posición, para obtener más ejemplos

Se utiliza ***torchvision transforms v2***, ya que permite aplicar la misma transformación a las bounding box de las mismas.



* **Armado del Dataset:** Se creó el dataset, con un split de 60% entrenamiento, 20% validación, 20% prueba. El mismo carga el archivo csv, y repite sus filas **n\_augment** cantidad de veces.

En **get\_item**, se obtiene la imagen ya procesada, se define su bounding box y se le aplica una transformación aleatoria. Devuelve finalmente la imagen transformada, la bbox transformada, y su label

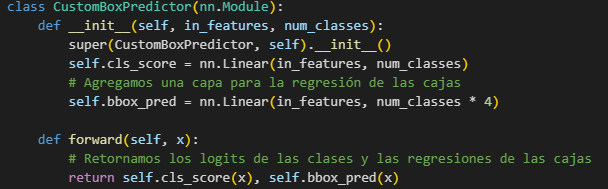


# Entrenamiento del modelo

Se realizaron pruebas con los modelos SSD y Faster RCNN.

Los mismos son modelos que se descargaron por medio de pytorch, con sus pesos preentrenados, y luego fueron finetunneados, el SSD redefiniendo las dimensiones de su matriz de pesos y bias para corresponder a las clases que necesitamos.

En FasterRCNN reemplazando su capa final con una capa personalizada, CustomBoxPredictor, que predice la clase y los desplazamientos de las cajas identificadas.



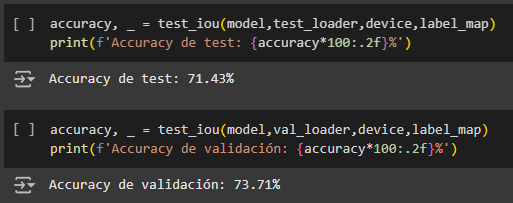
En cada epoch durante el proceso de entrenamiento, se envía un batch de imágenes, se realiza el entrenamiento, utilizando SGD como optimizador, y finalmente se calcula el accuracy del dataset de test.

Se realiza una comparación en cada epoch con el mejor accuracy, y si es más alto, se guardan sus pesos en el archivo ***best\_[modelo].pt.*** También, al final de cada epoch se guardan en un archivo ssd las métricas de train\_loss y test\_accuracy.

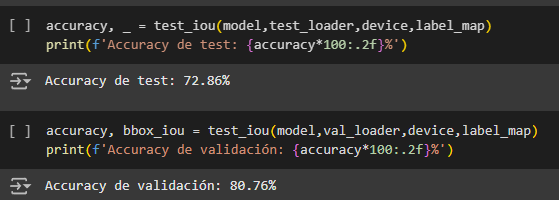
Al necesitar realizar más entrenamiento, se define el parámetro “nuevo” como False, y se busca el archivo con las métricas, para recuperar el mejor accuracy y continuar con el entrenamiento.

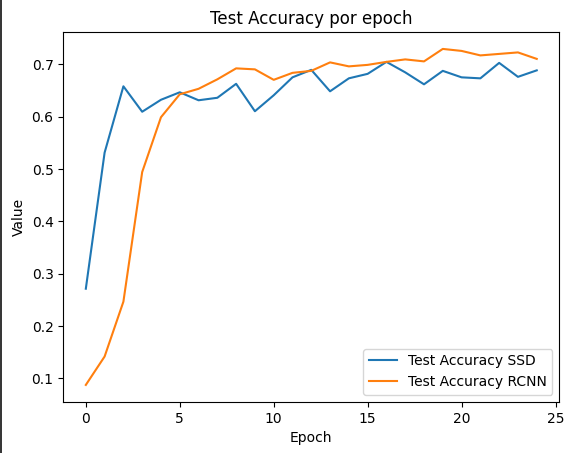
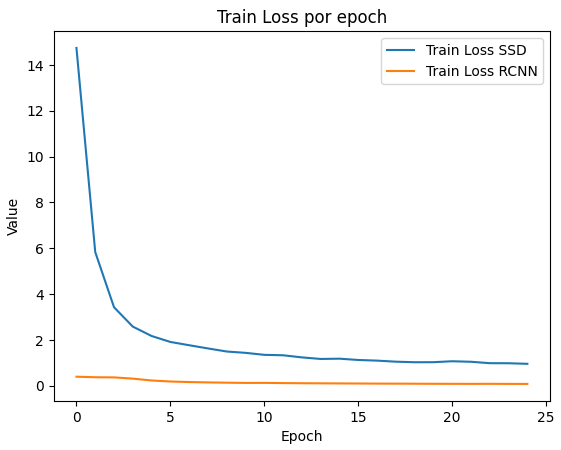
Terminado el entrenamiento con 25 epochs, se realizó una comparación entre los resultados de los modelos, utilizando la misma función de accuracy con los conjuntos de validación. Finalmente, se eligió el modelo FasterRCNN.

**SSD:**

****

**FasterRCNN:**

****

****