Image Segmentation

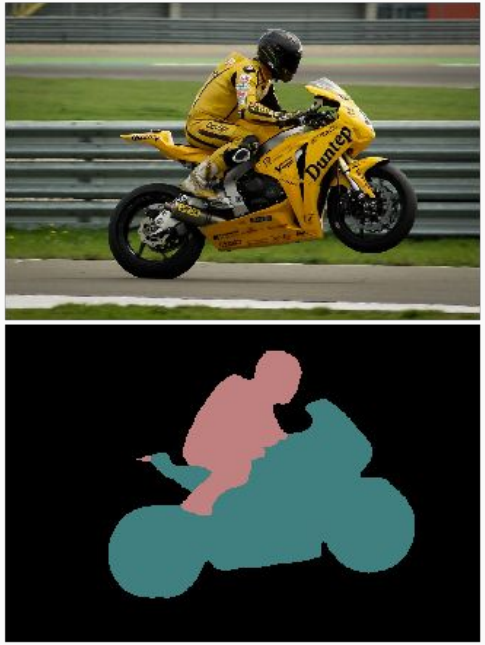
Pablo Paseiro NIA: 205247
Jorge Bermejo NIA: 206993
Raúl Cabrera NIA: 205918

Index

- Semantic Segmentation
- Motivation
- FCN
- Implementation
- Results
- Next Steps

Semantic Segmentation

- Semantic Segmentation is an image analysis task in which we classify each pixel in the image into a class
- Each pixel in the image is classified to its respective class (object in image)
- For example, the person is one class, the motorbike is another and the third is the background



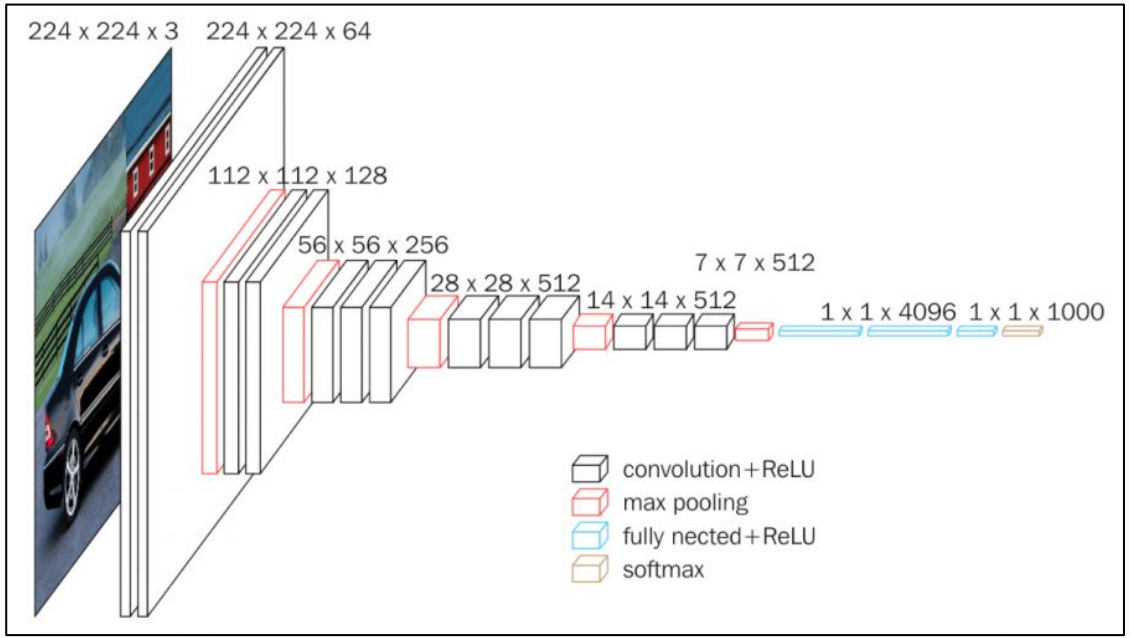
Motivation

We consider that the project is very relevant in the current times due to its several applications.

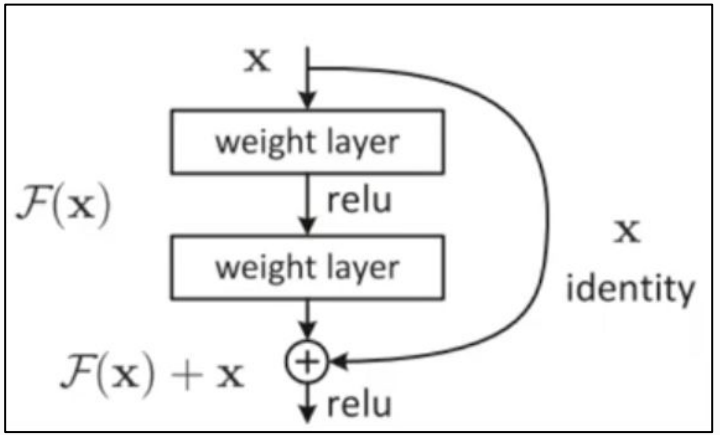
We find the Image Segmentation in fields such as:

- Autonomous Driving
- Facial Segmentation
- Indoor Object Segmentation (Augmented Reality and Virtual Reality)

VGG-16 & ResNet



Skip Connection



FCN

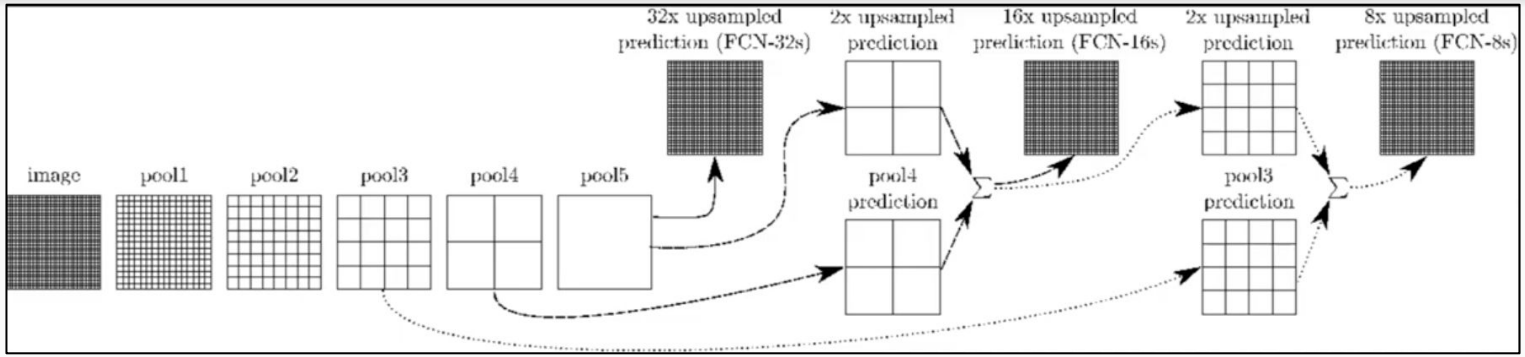
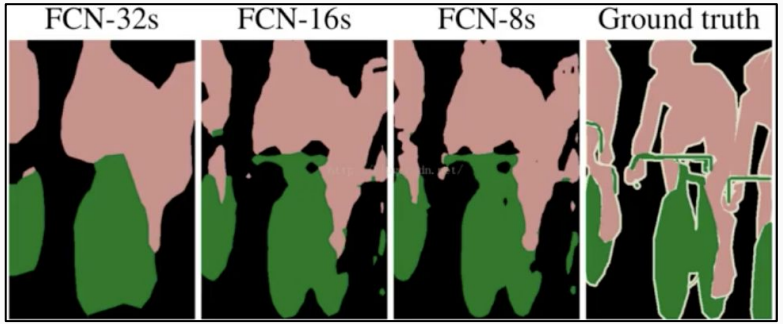
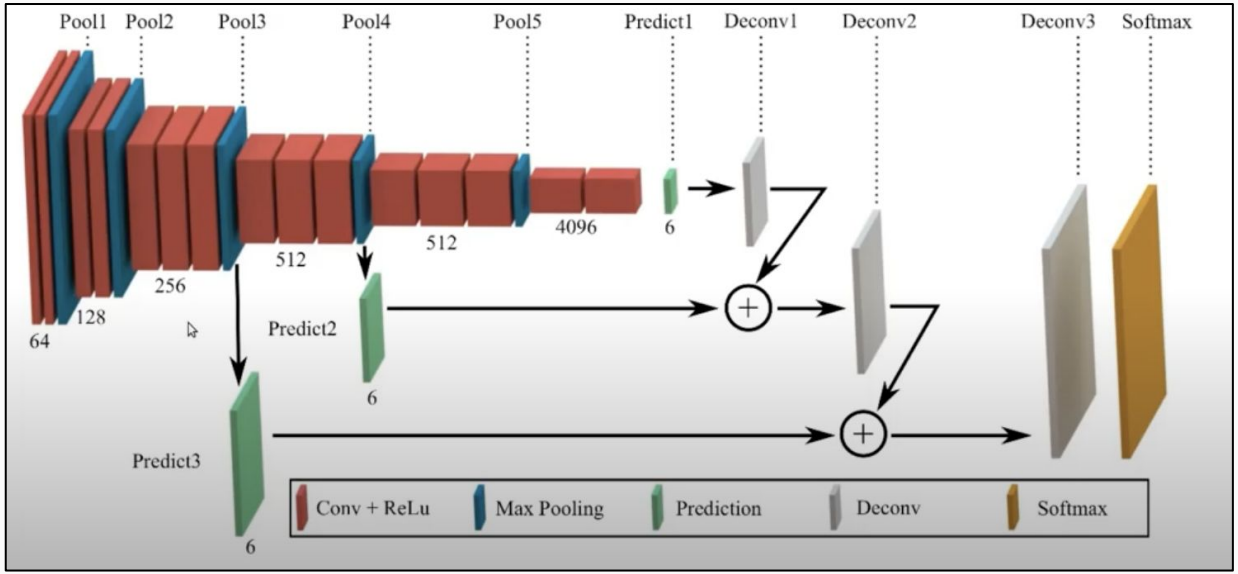


image
pool1
pool2
pool3
pool4
pool5
32x upsampled prediction (FCN-32s)
2x upsampled prediction
pool4 prediction
16x upsampled prediction (FCN-16s)
2x upsampled prediction
pool3 prediction
8x upsampled prediction (FCN-8s)


FCN-32s
FCN-16s
FCN-8s
Ground truth

FCN-8s


Pool1
Pool2
Pool3
Pool4
Pool5
Predict1
Deconv1
Deconv2
Deconv3
Softmax
64
128
256
512
512
4096
6
Predict2
6
Predict3
6
Conv + ReLu
Max Pooling
Prediction
Deconv
Softmax

Implementation

Semantic Segmentation using torchvision

We’re going to implement FCN model pretrained on a subset of COCO Train 2017 dataset which correspond to the PASCAL VOC dataset

Implementation : input and Output

These models expect a 3-channel image (RGB) which is normalized with imagenet mean and standard dev.

Input dimension: `[Ni x Ci x Hi x Wi]`

Ni-> Batch size
Ci -> Number of channels
Hi -> Height of the image
Wi -> Width of the image

Output dimension: `[No x Co x Ho x Wo]`

No -> Batch size
Co -> Number of classes of the Dataset
Ho -> Height of the image
Wo -> Width of the image

Implementation : FCN with Resnet-101 backbone

The application of the model is carried out in Torchvision

The structure of the implementation will be divided into:

- Loading process of the pre-trained model and the image from the chosen dataset
- The pre-processed of the image
- The implementation of the model upon the image
- The decode of the output

Loading Process

The pre-trained model is an implementation of the architecture of ResNet-101:

- ResNet is a CNN that is 101 layers deep. The network has been trained on a wide range of images so it has learned several feature representations

FCN-ResNet101 is a construction of a Fully-Convolutional Network model based on the ResNet-101 structure trained on 20 categories from the dataset:

```
fcn = models.segmentation.fcn_resnet101(pretrained=True).eval()
```

The image is loaded from the URL by using PIL

Processed of the image

We need to pre-process and normalize the image to the right format of the model, so we need to carry the following steps:

1. Resize the image to (256 x 256) → `T.resize(256)`
2. CenterCrop it to (224 x 224) → `T.CenterCrop(224)`
3. Convert it to Tensor (scale values to [0,1]) → `T.ToTensor()`
4. Normalize with Imagenet specific values → `T.Normalize(mean,std)`
5. Unsqueeze image to `[1 x C x H x W]` from `[C x H x W]` → `trf().unsqueeze`

Implementation of the FCN model

1. Pass the image through the model and get the out key `out=fcn(inp)['out']`

   Out key: Final output of the model with shape **`[1 x 21 x H x W]`**

2. Make this 21 channeled output into 2d image, each pixel of that image corresponds to a class :

   We take a max index for each pixel position, which represents the class:

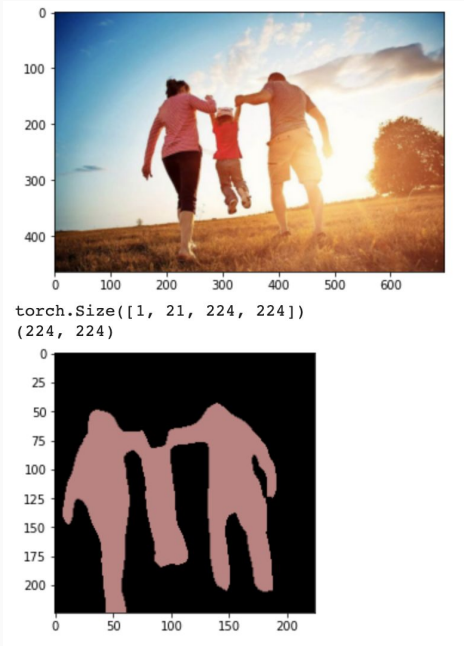
```
om = torch.argmax(out.squeeze(), dim=0).detach().cpu().numpy()
```

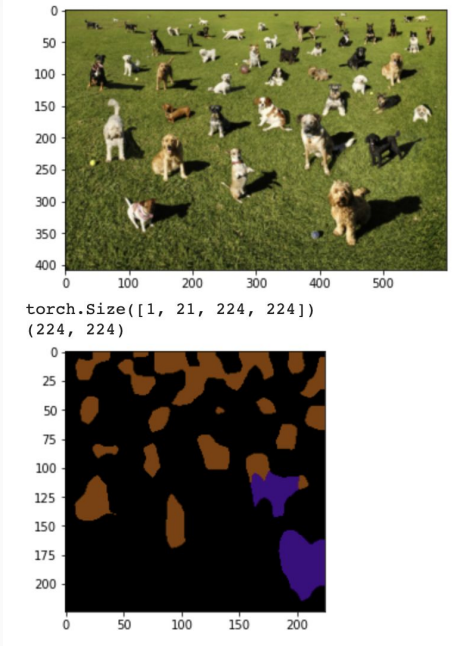
Implementation of the FCN model II

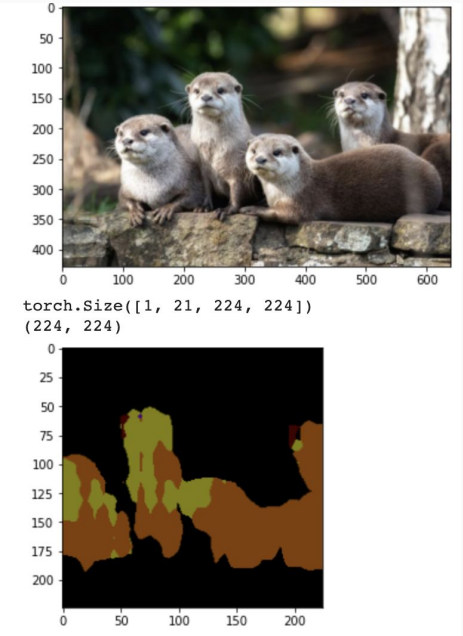
Convert 2D image to an RGB image:

1. Store color for each class in a vector according to the index (label)
2. Create empty matrices for all 3 channels (RGB)
3. Loop over each class color and get position in the image where the index = label
4. Assign in that position a color value for 3 channels
5. Stack the 3 channels to form a RGB image

Implementation : Final results







Next Steps

Having implemented the pre-trained model of the FCN in the seen case, the objective of our project is:

- To compare the model with the DeepLab architecture
- To replicate the behaviour of the model FCN and to understand its training process
- Implement model Res-Net 101 with a Big Dataset, train this Dataset with DeepLab architecture and compare the accuracy and training time

THANK YOU!

Image Segmentation

Pablo Paseiro NIA: 205247
Jorge Bermejo NIA: 206993
Raúl Cabrera NIA: 205918