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

FINAL PROJECT - IMAGE SEGMENTATION

COMPARISON BETWEEN *FCN* AND *DEEPLAB* MODELS

PROJECT

In this project, we're going to look at two Deep Learning based models for Semantic Segmentation. Fully Convolutional network and DeeplabV3. We'll start by reviewing these models, the structure and how they're implemented. We'll use these models ,which are already pre-trained, to test them in a Dataset created by us, which combines images of the objects that the models can classify. There are 20 categories supported by the models such as background, aeroplane, bicycle, bird, etc.
In order to test the model, we've chosen 5 images from aeroplanes, birds, botes, cars, dogs, horses, humans, sheeps.
After testing the models with the Dataset, we'll make a comparison between them by looking at the Inference time on CPU and GPU, the size of the model, and GPU memory used while inference.

FCN

The first model we've tried on the dataset is FCN model, Fully Convolutional Network. This model, along with the DEEPLAB one ,which we'll explain later, have been trained on a subset of COCO Train 2017 dataset which corresponds to the PASCAL VOC dataset.

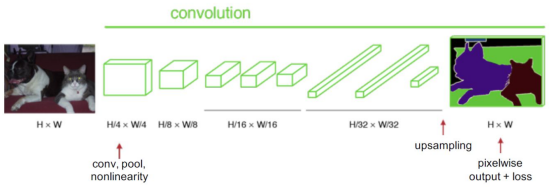
FCN can execute tasks for semantic segmentation, which is the purpose of our project, nevertheless segmentation is much more difficult task than classification and detection tasks. Semantic segmentation aims to classify the object class for each pixel with an image. That means there is a label for each pixel.

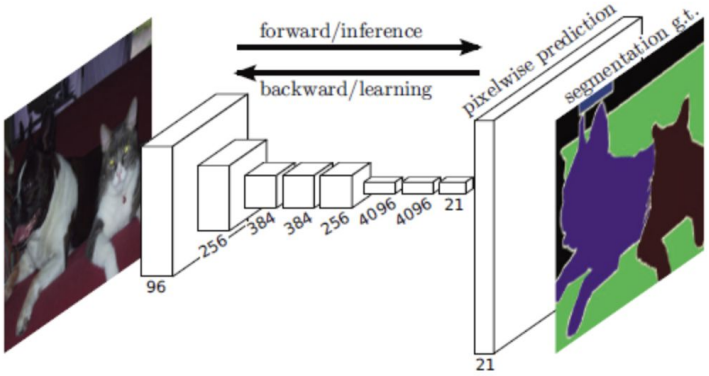
A fully convolutional network (FCN) uses a convolutional neural network to transform image pixels to pixel categories. Unlike the convolutional neural networks previously introduced, an FCN transforms the height and width of the intermediate layer feature map back to the size of input image through the transposed convolution layer, so that the predictions have a one-to-one correspondence with input image in spatial dimension (height and width). Given a position on the spatial dimension, the output of the channel dimension will be a category prediction of the pixel corresponding to the location.

<u>Model architecture:</u>

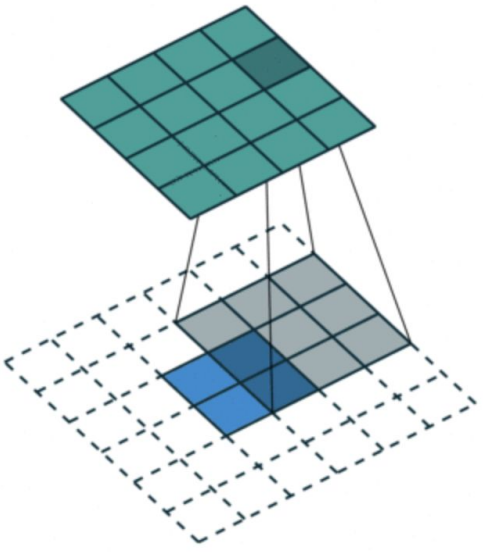
The fully convolutional network first uses the convolutional neural network to extract images features, which includes having conv, pool filters (nonlinearity). Then,while Image Classification would use Fully Connected layers, FCN turns FC layers into 1x1 convolutional layers so now it transforms the number of channels into the number of categories.

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





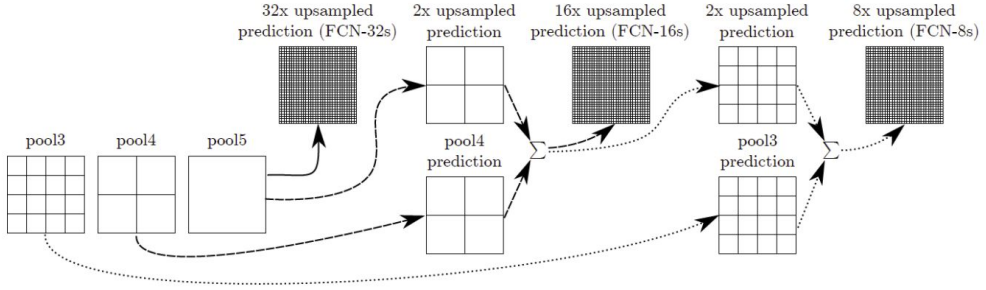
Finally,we upsample the output by transforming the height and width of the feature map to the size of the input image by using the transposed convolution layer.
The process of upsampling is also called fractional stride convolution when fractional stride is used.



Upsampling via Deconvolution

After going through the convolutions below, the output size is small, then the 32x upsampling is done to make the output have the same size of the input image. But we combine FCN 16 and FCN 8s after FCN-32 upsampled is done, if we only due FCN-32 it would make the

output labe map rough due to the fact that is too deep and spatial location is lost when going deeper.
In FCN-16s the output from pool5 is 2x upsampled and fused with pool4 and perform 16x upsampling.
In FCN-8s it does similar operations, the output from FCN16 is 2x upsampled and fused with pool3 so we now have an 8x upsampled prediction.

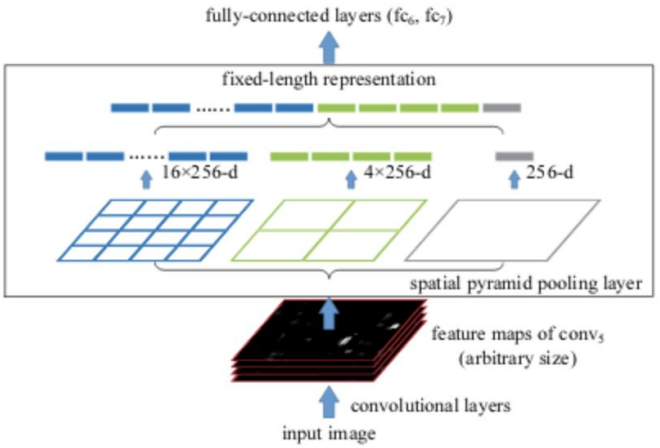


DEEPLAB
The model which we have compared the pretrained FCN architecture with is called DeepLab. This other model is based on an architecture composed by an encoder and a decoder, generally. The encoder and decoder will carry out their respective tasks, in case of the encoder, we expect to implement some kind of Convolutional Neural Network to reduce the dimensionality of the input that we are providing to the model at the same time that we are extracting the meaningful features of the input image. In case of the decoder, we expect the reconstruction of the output from the significant information taken by the encoder, this process will make use of some classification function to match the corresponding labels with every pixel on the input, apart from the upsampling task.

As we can see, the working is pretty similar to the one seen on the FCN model. However, the DeepLab model includes some fundamental building blocks that make it more efficient in terms of the computational complexity.

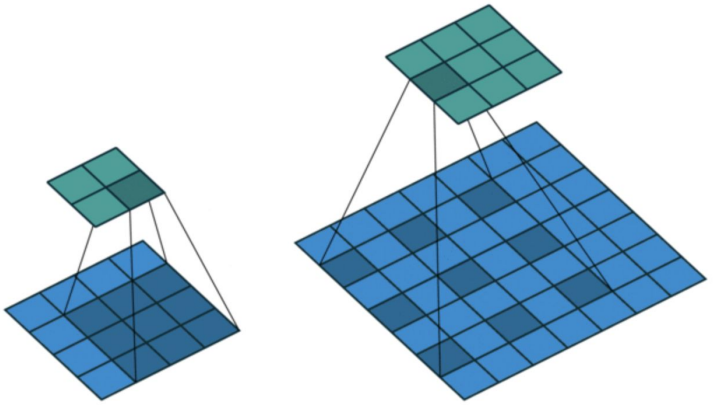
Firstly, there is a problem related with the size of the inputs. Of course, we can design a model which is trained on a dataset including only images with the same size, but this wouldn't be optimal for future implementations on test datasets where the images have some random factor with respect to their size. Because of that, we wish that our model is capable of adapting to different dimensions on inputs, and that is achieved by training the model on different scales of the images as the input.
DeepLab uses Spatial Pyramid Pooling as a pooling layer set among the last convolutional layer and the Fully Connected layer. This DeepLab pooling layer will convert the output of the previous layer or feature map, which is basically formed by the relevant information extracted by the filters during the convolutional process, into a vector with fixed dimensions (FxB, where F is the number of filters in the previous layer and B is constant that indicates a number of bins, the feature maps from the convolutional layer will be divided into a constant number of bins with size proportional to the input size). The output vector of the SPP will be

the input of the FC layer and since the number of bins is fixed as mentioned, the dimensions of that vector will not be variant. Next, you have a graphical representation of the Spatial Pyramid Pooling:



Notice that the vector which is sent to the next layer is a concatenation of the output vectors of every SPP process applied on each of the feature maps.

Nonetheless, the vector which is introduced on the FC layer can have large dimensions and so the dimensionality problem wouldn't be solved yet, in this case, the computational complexity could have suffered an increment. This is the reason why the DeepLab model implements a dilation parameter (*Atrous Convolutions*) which provides information about the space between the positions in a kernel. So, for example, using a dilation parameter of value 1 would be equivalent to accomplishing the convolution as usual, without modification in the kernel. The following representation shows the intuition behind this process:



As you see, applying such parameter causes the growth of the field where the filter will be implemented, because of that, in the left example the position of the filter covers a 3x3 filed. The thing is that the dilation will not mean the use of more parameters.
The dilation will be used during the SPP process, to avoid the previously mentioned problem.

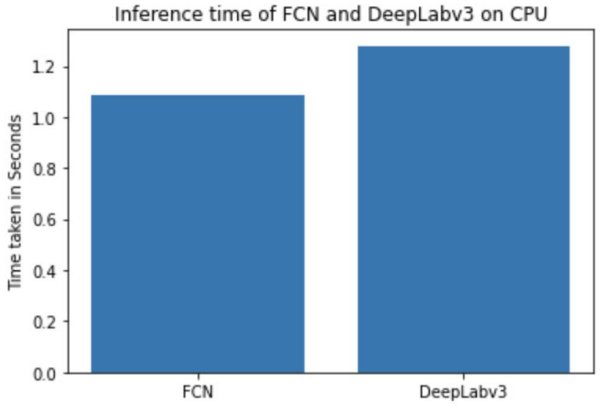
Pablo Paseiro NIA: 205247
Jorge Bermejo NIA: 206993
Raúl Cabrera NIA: 205918

For the procedure of this project we have concretely tested the model *DeepLabv3*. As stated before, the model is based on an architecture composed by an encoder and a decoder, but the tasks made by both blocks are distributed in a different way, the *DeepLab v3* model uses Depthwise Separable Convolution, which is based on two steps. The input generally will be an RGB image, it implies that the input has 3 channels. If the filters used to convolve have no padding (SAME) and a stride of 1, the model firstly computes the convolution separately for each of the 3 channels and then, stacks the three outputs [*Depthwise Convolution*]. Secondly, to increase the number of channels, as with MAX Pooling, a convolution with the same number of 1x1 filters as the fixed number of bins will be computed upon the previous step output, so the Spatial Pyramid Pooling is applied [*Pointwise Convolution*]. The final result of the Depthwise Separable Convolution will be a HxWxBins output, which in comparison to the usual convolution, this process will have supposed less computations.

The model *DeepLab v3* uses an architecture based on ResNet 101 and it is been pre-trained on the dataset MS COCO. That is why the two previous models can be compared perfectly to check their efficiencies with respect to the other.

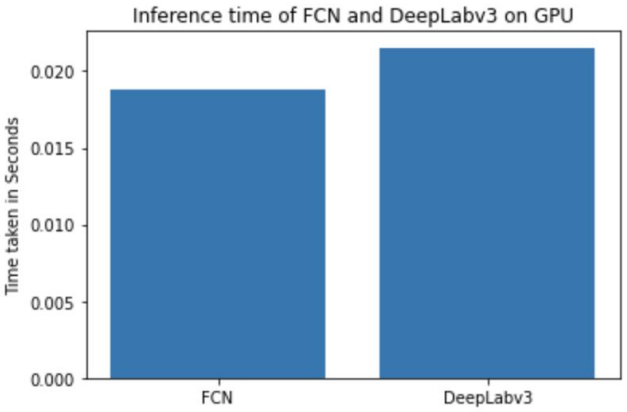
COMPARISON BETWEEN MODELS:

Inference time:




```
The Average Inference time on FCN is:     0.98s
The Average Inference time on DeepLab is: 1.17s
```

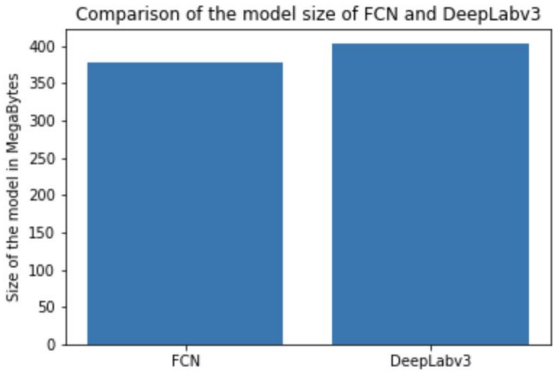
Pablo Paseiro NIA: 205247
Jorge Bermejo NIA: 206993
Raúl Cabrera NIA: 205918




```
The Average Inference time on FCN is:     0.019s
The Average Inference time on DeepLab is: 0.022s
```

We can see that DeepLab model is slightly slower than FCN with a difference of 0.19s in CPU and 0.003s in GPU

Model size:




```
Size of the FCN model with Resnet101 backbone is:       378.16 MB
Size of the DeepLabv3 model with Resnet101 backbone is: 403.67 MB
```

Model size refers to the weights file for the model. We can see that DeepLab is slightly bigger model than FCN.

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

REFERENCES

[1]https://www.learnopencv.com/pytorch-for-beginners-semantic-segmentation-using-torchvision/
[2]https://kharshit.github.io/blog/2019/08/09/quick-intro-to-semantic-segmentation
[3].https://towardsdatascience.com/review-fcn-semantic-segmentation-eb8c9b50d2d1
[4]https://d2l.ai/chapter_computer-vision/fcn.html
[5]https://towardsdatascience.com/implementing-a-fully-convolutional-network-fcn-in-tensorflow-2-3c46fb61de3b#:~:text=FCN%20is%20a%20network%20that,connected%20layers%20(Dense%20layers).
[6]https://towardsdatascience.com/applied-deep-learning-part-4-convolutional-neural-networks-584bc134c1e2