Semantic Segmentation

FROM THE SCRATCH

Ramraj Segur Mahadevaraja | 6/27/2018

Introduction

What is Semantic Segmentation?

Semantic segmentation describes the process of associating each pixel of an image with a class label, (such as road, sky or car).

Why Semantic Segmentation?

It is widely used for robot vision and understanding. It is a hot research topic now and they are being studied to deploy on the Autonomous cars to better make them understand the scene / image they are seeing.

Challenge in Semantic Segmentation

The most challenging task in deploying a neural network generally is to minimize the lag between the processing the input data and providing the output.

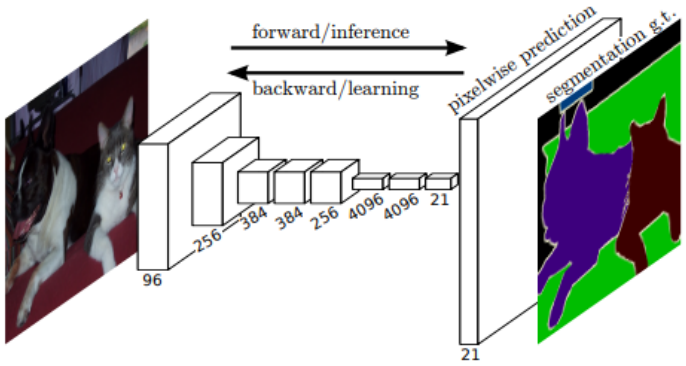
Even the real-time implementation of a sparse prediction task is a challenging one, implementation of the semantic segmentation which is a dense prediction task is a hard one. To thoroughly understand the recent research publications on this topic a literature review was made.

Literature Review:

Research papers have discussed a lot of network architectures and the following architectures are more interesting and can tackle the challenges possessed for semantic segmentation.

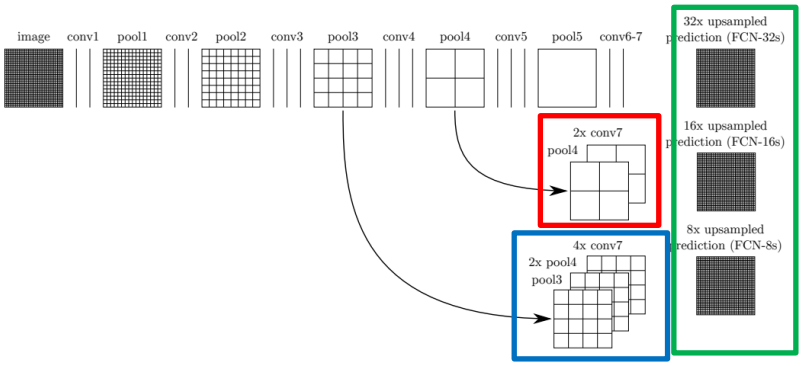
- Fully Convolutional Neural Network
- SegNet
- ICNet – Image Cascade Network

Fully Convolutional Network (FCN):

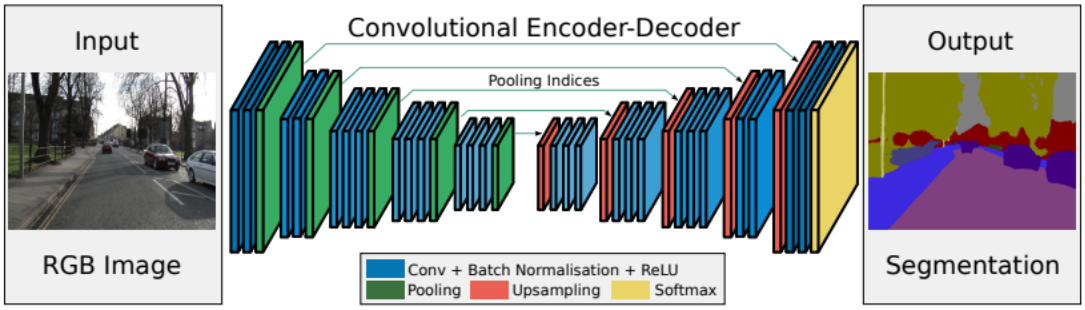


This network takes only the convolutional layers from VGGNet and then add the transposed convolution to the selected convolution layers and resize and merge them to predict the final dense image mask. The diagram below actually depicts that.

The output of the last convolution layer (conv7) is resized (twice the original) and then merged with the pool4 layer (indicated by red box). The conv7 is increased by 4 of its normal size and the pool4 is doubled and then merged with the pool3 output (indicated by blue box). The output of all of them are combined and then trained with the labels (indicated by green box).



SegNet:



The image describes the structure represented in the SegNet research paper. It is a encoded and decoder network. Encoder network is simply a Convolutional layer like the VGGNet with the fully connected layers at the end removed. Then this Convolutional Layer network is connected to the deconvolutional layers (i.e., Transposed Convolution). This series of transposed convolution layers are considered as the decoder layer which will upsample the input it gets with the memorized max-pooling layers from the respective convolutional layers. Then the output of the series of transposed convolutions is then compared with the label (mask) for training.

Image Cascade Network (ICNet):

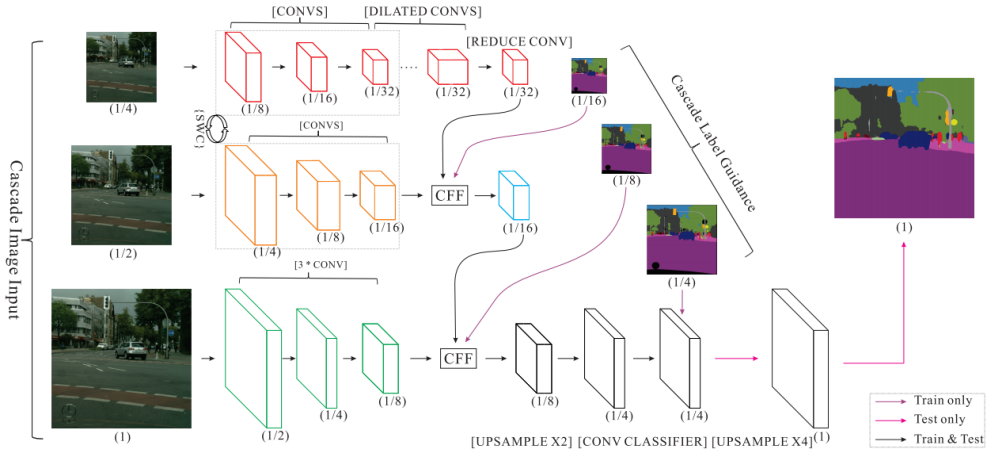
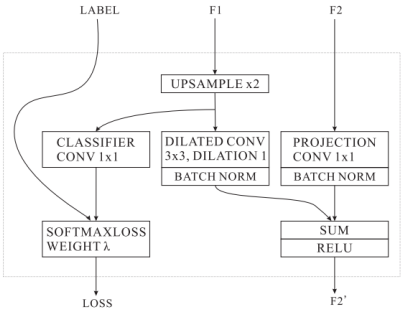
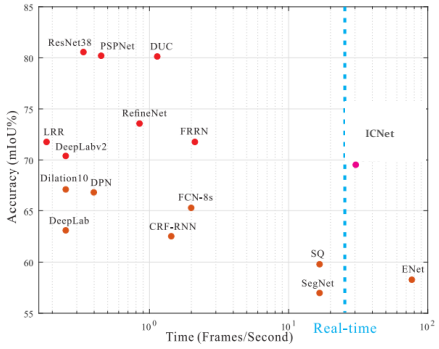


Image Cascade network eliminates the use of transposed convolution layers instead it is just employs resizing. The input images to the network is supplied in three different sizes (1/2, 1/4, 1/8) as noted the diagram representing the network architecture. The masks are also resized appropriately to fit the output of the trainable layers. Here the method simply employs the downsizing through the convolutional layers and train them at each branch. The output of one branch is connected to the next branch by using the following architecture.

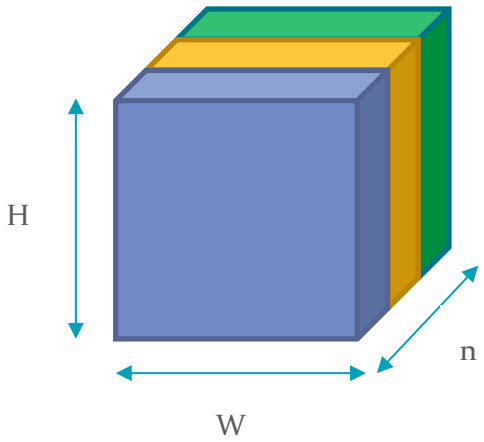




Then the output of each branch is then trained with appropriate mask labels. Then the performance of the network is compared with the previous networks. Because of the exceptional performance of the ICNet as observed in the graph shown above it was selected for the project.

Network:

The labels are represented in the following format. The mask is form of an array whose length and width are same as the length and width of the input image and the depth of the mask depends on the number of classes that need to be segmented for each pixel.



Here in this case the 'n' is considered as 3. First channel is responsible for the pixels which represent the objects of no interest (no car and no road). Second channel is responsible for the pixels which represent the road surfaces. Third channel is responsible for the pixels represent the car surfaces.

Custom Cost Function:

The cost function provides more weightage to the classes of cars as they are having small presence in the masks.

$$\text{Weights} = 1 - \frac{Number\ of\ the\ pixels\ belong\ to\ the\ class}{Total\ number\ of\ pixels\ in\ the\ image}$$

From the formula we can understand that the weights will be inversely proportional to the number of pixels representing the class.

Hence the car pixels will have more weight compared to the road pixels and the other area pixels.

Image Data Acquisition from CARLA:

In order to train the neural network, the data is acquired from the CARLA simulator which comprises both the masks and the input images in RGB spectrum.

Outcome plots:

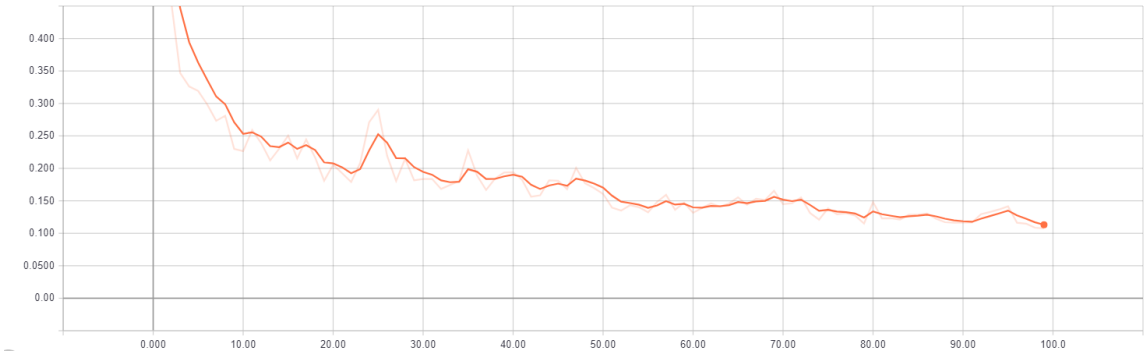


Figure 1: Training loss - Top Layer

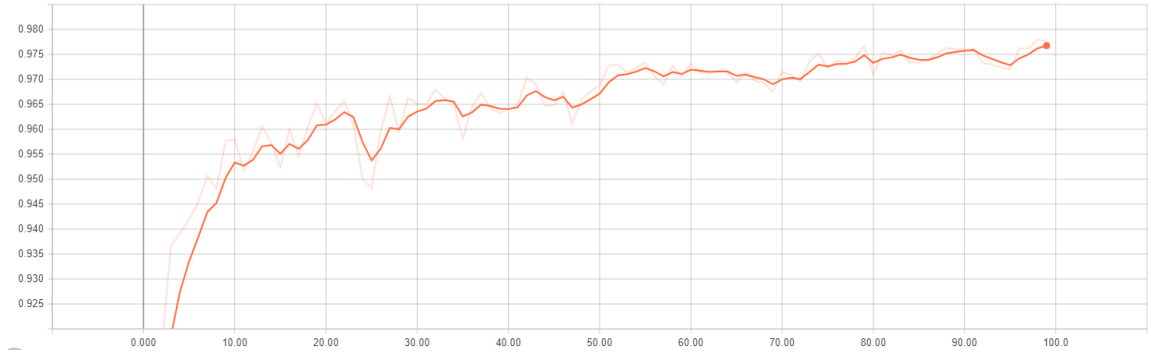


Figure 2: Training accuracy - Top Layer

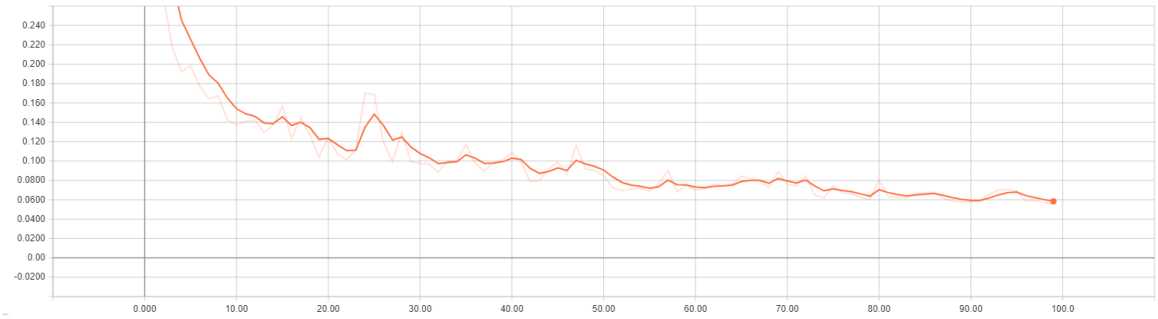


Figure 3: Training loss - Middle Layer

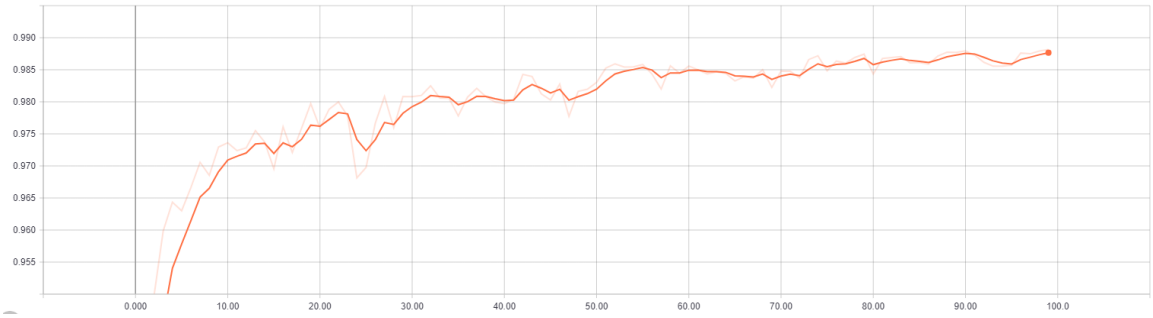


Figure 4: Training accuracy - Middle Layer

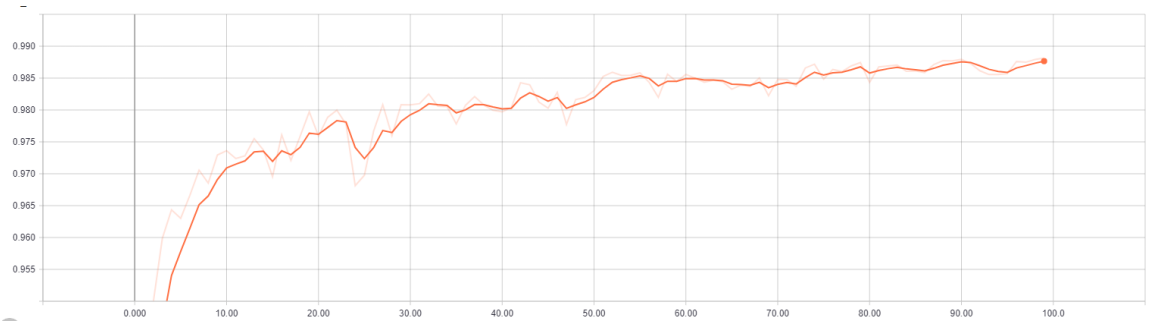


Figure 5: Training loss - Low Layer

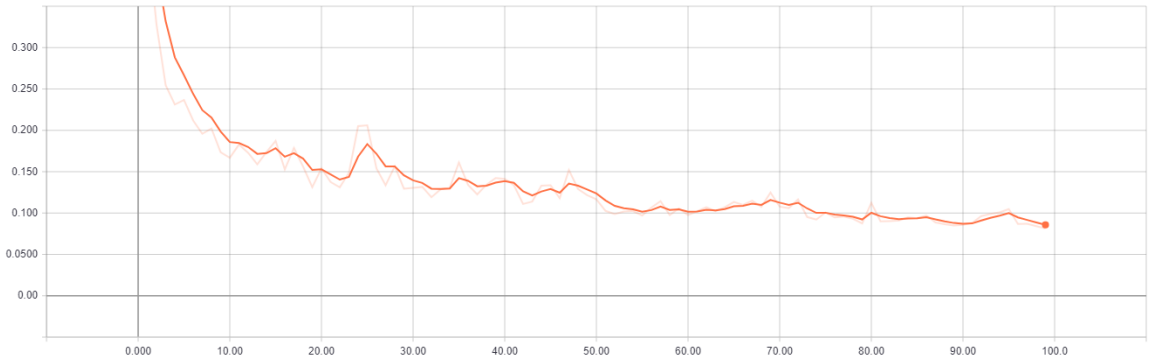


Figure 6: Training accuracy - Low Layer

Knowledge enrichment:

From this project I have got good understanding about the concept of semantic segmentation. The project has provided challenging environments of creating custom cost functions and the creation of layers in the Keras framework and setting up the virtual machine instance in google cloud.