Road Boundary Detection with SEGNET

Balamanikanta C
IMT2016051

Puneeth Sarma K A V
IMT2016018

Soumith Kumar D
IMT2016110

Sri Sumanth Y
IMT2016081

October 20,2019

ABSTRACT

In this paper, we summarize SegNet a semantic segmentation model and propose a smaller version of segnet to work in a low data and low computational power environment. We discuss the training procedure, optimisation strategies used, and results on a small subset of sityscapes dataset.

1 Introduction

With the strides made in objection detection from 2012, one of the other major tasks in computer vision was segmentation (semantic and instance). Semantic segmentation is classifying each pixel in the image into one of the predefined classes. It can be thought of as pixel-level classification. First model for this task used convolutional layers to learn features followed by fully connected layers that predict the class probabilities for all the pixels at once. There was a problem with this method. If the network is deep, convolutional layers downsample the image. Upsampling was done using various classic techniques like the nearest neighbor, the bed of nails, etc. The details might not be fine when this is upsampled. Deep networks work better and using shallow networks is not very semantic. Subsampling actually allows the model to look at larger sections of the image to make the prediction. Hence using deep networks is necessary. To solve this problem, Fully Convolutional Networks (FCNs) were proposed. These networks use convolutional layers to upsample i.e learning the upsampling process. SegNet is one of the FCN models. There are no fully connected layers in these models. FCNs give better performance. In this paper, we talk about segnet and we propose a smaller segnet model, to train on the KittiSeg dataset for road classification. We talk about the model, training the model, and results. Since ground truth for KittiSeg is not available, we took a small sample set from the cityscapes dataset for evaluation. We also present a few outputs of our model.

2 Related work

2.1 SEGNET

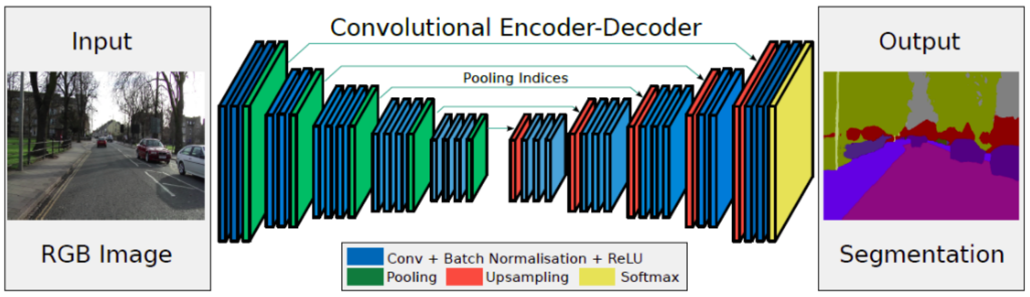

Figure 1: SegNet: Encoder Decoder architecture

This architecture is used for semantic pixel-wise image segmentation i.e. labelling each pixel of an image belonging to some class(road, vehicle etc).

The basic outline is, it takes an RGB image as an input and outputs a segmented image where each pixel of the same color belongs to the same class. As you can is in the Figure 1, Segnet architecture contains an encoder and decoder.

2.1.1 ENCODER

The encoder comprises of three types of layers:

- Convolution - To extract local features by using filters at each stage.
- Pooling - To downsample the feature map and propagate spacial invariant features to the deeper layers. It is used to reduce the number of parameters in the network.
- Batch Normalization - It normalizes the training data to accelerate learning. It allows each layer of network to learn by itself, a little bit more independent of the other layers.

The encoding network consists of 13 convolutional layers that correspond to the first 13 convolution layers in the VGG16 network. So we can initialize the training process from weights trained for classification on large datasets.There are no fully connected layers in the encoder to retain higher resolution feature maps at the output of the encoder. This reduces the number of parameters from 134M to 17.4M.

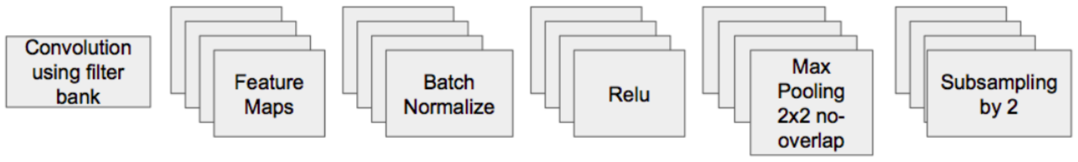

Figure 2: Encoder architecture

As shown in the image above, each encoder in the encoder network performs convolution with a filter bank to get the set of feature maps. These are batch normalized and then RELU is applied. Following that, a max pooling with a 2*2 window and stride 2(non-overlapping) is done and the resulting output is subsampled by a factor of 2. Max pooling is used to achieve translation invariance over small spatial shifts in the image. After Subsampling, it leads to each pixel governing a larger input context. The max-pooling and the subsampling methods improve the classification accuracies but reduce the feature map size. It also leads to the lossy image representation with blurred boundaries which is not ideal for segmentation purposes. It is desired that output image size is the same as input image size. To achieve this, we use the decoder network in the Segnet architecture.

2.1.2 DECODER

The Encoder obtained information about what objects are and approximately where there are. But we need to know the exact pixel where it comes from. To do this we use the decoder network. For each 13 encoder layers, there is a corresponding decoder layer. These convolutional layers add geometric details to the feature map to map up the loss encountered during the pooling layers in the encoder. The important step in the decoder part is the upsampling. During upsampling, a 1*1 pixel is converted to a 2*2 image. Here, only one pixel in the 2*2 window takes the previous value of that pixel. We can randomly assign it to anyone of the four pixels. This results in a small error in the beginning, but by the end of the last upsampling layer, the error will be large. To tackle this, in the encoder part at each pooling layer the index of the maximum of 2*2 pooling window is stored and is used in the upsampling layer in the decoder network as shown in the Figure 3.

After the final decoder layer, the output is fed to a softmax classifier to get the final prediction.

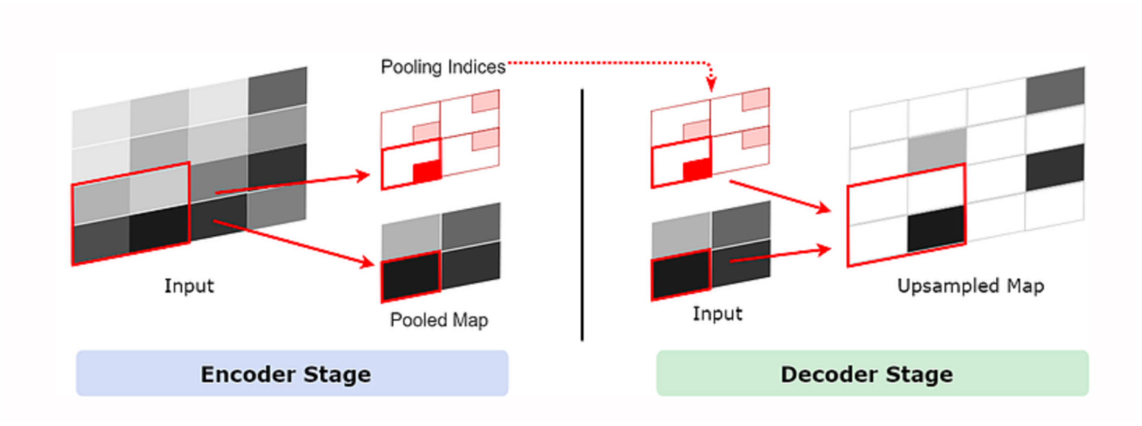

Figure 3: Pooling and Upsampling

3 Our Model

The Segnet model which was already defined is too big for our small dataset. As we know that deep neural networks trained on small dataset overfits the model. This has the effect of the model learning the statistical noise in the training data, which results in poor performance when the model is evaluated on new data, e.g. a test dataset Generalization error occurs. This also decreases the number of trainable parameters for our new model.

Our proposed idea to this is to remove some Convolution layers and their activation functions from the original model. At each filter level of the model we have removed some of these from the neural networks that extract deeper features from the data.

In the encoder network of the segnet architecture, at each pooling layer the index of the maximum of 2*2 pooling window is stored and is used in the upsampling layer (Bed of nails) in the decoder network. But the keras function *MaxPooling* has no mechanism to store the index value where the maximum element is present.

So we have implemented these two layers in our model.

- MaxPoolingWithArgmax Layer
- MaxUnpooling Layer

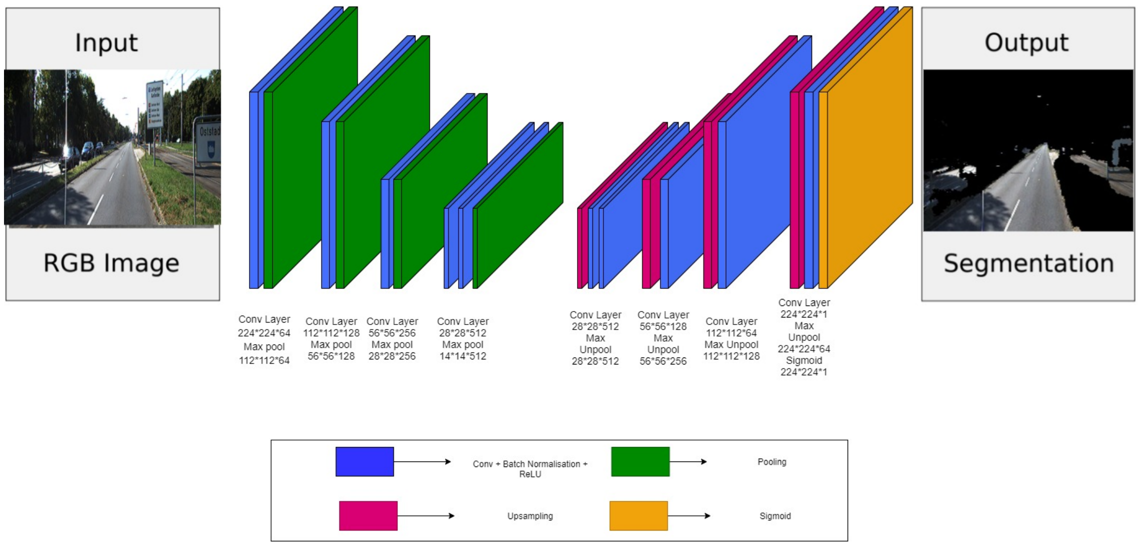
Input
RGB Image
Output
Segmentation
Conv Layer 224*224*64 Max pool 112*112*64
Conv Layer 112*112*128 Max pool 56*56*128
Conv Layer 56*56*256 Max pool 28*28*256
Conv Layer 28*28*512 Max pool 14*14*512
Conv Layer 28*28*512 Max Unpool 28*28*512
Conv Layer 56*56*128 Max Unpool 56*56*256
Conv Layer 112*112*64 Max Unpool 112*112*128
Conv Layer 224*224*1 Max Unpool 224*224*64 Sigmoid 224*224*1
Conv + Batch Normalisation + ReLU
Pooling
Upsampling
Sigmoid

Figure 4: Our Proposed Model's Architecture

4 Training

4.1 Training procedure

The Data from the KITTY data set is used for training, the train images in the data set are used as train data.In the Data set the ground truth image contains segmented masked images for multiple classes we can't use those images as ground truth so ground truth for train data is created. For generating ground truth of the train data, the ground truth images of the Data set are used.The image was segmented into two parts one is the road and the remaining image is taken as other part and the masks were generated.The resultant images with the masks are the ground truth for our train data

A data generator is created, It sends the train data and ground truth to the model to train according to the batch size. We used a batch size of **16**. The model was run using ADAM as the optimizer. The train data is included with flipped images of the original data, To add the variance to the train data. The data is given as input to the encoder part of the model. The output of the encoder is given as input to the decoder. The output of the decoder is sent to the sigmoid activation layer to obtain the final output.

The binary cross-entropy loss function is used to train the model. The loss converges by the 10^{th} epoch.

4.2 Optimisation Strategies

- Unlike segnet, which used SGD optimiser, we have used adam optimiser[5] which is known to converge faster.

- The batch norm layers in every block after the convolutional layers in both encoder and decoder networks helps in faster convergence [4].

- We have used a smaller model to avoid over fitting. For transfer learning, we tried to extract corresponding layer weights from VGG to initialize, but since we are using a smaller model we are not sure how to do this. So we have avoided this.

5 Results

Since ground truth for the kitti dataset testing images was not available, our plan was to split the training set 70 - 30. But, since the total number of images available in training set is only 200, this seemed like a bad idea. So we have used collected 200 images (Aachen city images) from Cityscapes dataset for evaluation. This will also show the generalising abilities of our model in new conditions.

Epoch No.	Training Accuracy	Training Loss
10	72	66

	Testing Accuracy	Testing Loss
	66	34

5.1 Input and output images:

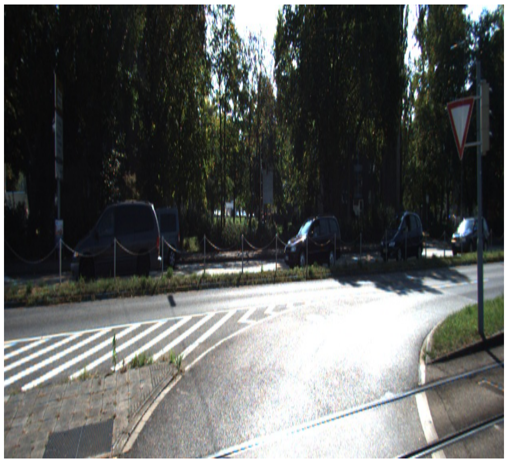

Figure 5: Input 1

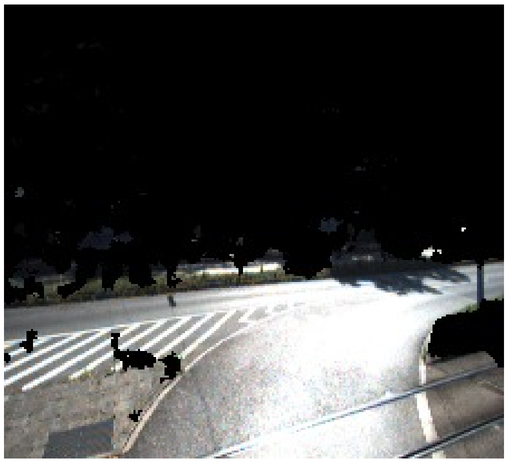

Figure 6: Output 1

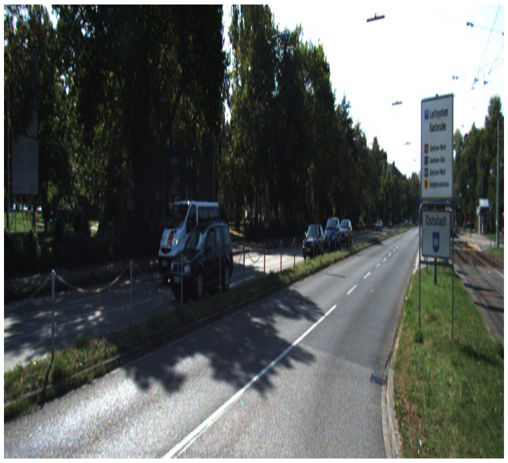

Figure 7: Input 2

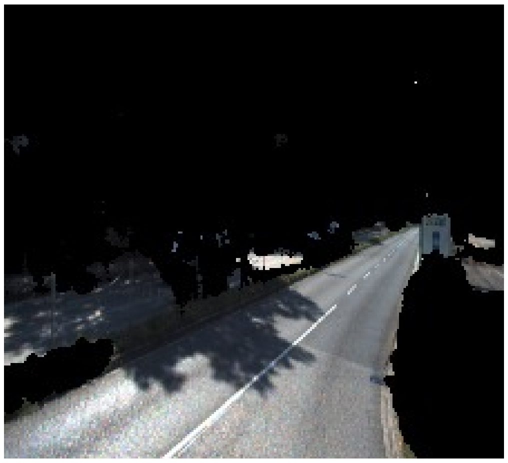

Figure 8: Output 2

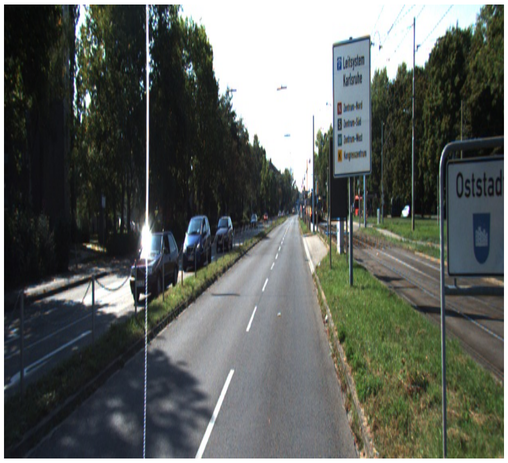

Figure 9: Input 3

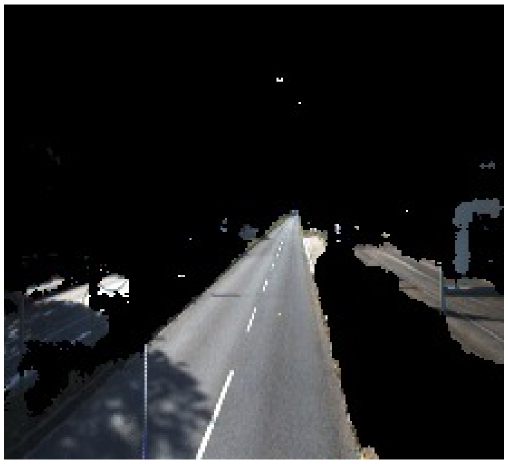

Figure 10: Output 3

References

[1] Vijay Badrinarayanan, Alex Kendall, Roberto Cipolla. SegNet: A Deep Convolutional Encoder-Decoder Architecture for Image Segmentation.

[2] Olaf Ronneberger, Philipp Fischer, Thomas Brox. U-Net: Convolutional Networks for Biomedical Image Segmentation.

[3] George Papandreou, Liang-Chieh Chen, Kevin Murphy, Alan L. Yuille. Weakly- and Semi-Supervised Learning of a DCNN for Semantic Image Segmentation.

[4] Sergey Ioffe and Christian Szegedy. Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift.

[5] Diederik P. Kingma and Jimmy Ba. Adam: A Method for Stochastic Optimization.

[6] Jonathan Long and Evan Shelhamer and Trevor Darrell Fully Convolutional Networks for Semantic Segmentation