Project report

Semantic Segmentation using Fully Convolutional Networks

Suhas Chowdary J (suhaschowdaryj@email.arizona.edu)

ELECTRICAL AND COMPUTER ENGINEERING

ECE 533

Digital Image Processing

Spring-2018

Instructor: Prof. Jeff Rodriguez

May 01, 2018

THE UNIVERSITY OF ARIZONA®

Arizona's First University.

College of Engineering

ABSTRACT

Semantic segmentation is the pixel level classification which associates each pixel to a predefined class label. Deep convolutional neural networks have shown exceptional performance in computer vision tasks such as object detection, image segmentation and image classification. However, traditional neural networks cannot perform well on semantic segmentation as they cannot preserve spatial information of the image. Fully Convolutional networks, a variant of neural networks with introduction of techniques like deconvolution, skip models and upsampling can improve the performance of semantic segmentation. In this project, we built two Fully Convolutional Network architectures (FCN-32s and FCN-8s) for segmentation task. We used transfer learning on VGG net architecture and fine-tuned the model with various hyper parameters. To improve efficiency of the model and to produce fine grain level, we coupled the fine-tuned models with skip models and used techniques like up-sampling and atrous convolution for dense prediction. The models are trained and validated on PASCAL VOC 2012 data using Keras, an open source neural network library with TensorFlow as backend and Python. With FCN-32s architecture, we achieved a pixel accuracy of 58% and Intersection over Union (IoU) of 52%. Whereas Using FCN-8s model, we obtained a pixel accuracy rate of over 63.6% and IoU of 57%.

1. Introduction

Convolutional neural networks are a class of neural networks which are developed specifically for images. They explicitly assume that the inputs are images and leverage properties of images to boost the performance. Extensive use of Convolutional neural networks (CNN) has been started since Alexnet [11] won the ImageNet large Scale Visual Recognition Challenge (ILSVRC) by a large margin in 2012. The Alexnet has introduced use of new techniques like Rectified linear units (ReLu) as activation functions and new data augmentation techniques. The architecture has been modified in numerous ways which includes introduction of pooling layers, strides and changing filter size from 11x11 to 3x3 and 5x5 to improve receptive field capacity and detain noisy background. VGGnet [8], GoogleNet [12] and resnet [13] architectures have pushed performance of image classification to human level error. CNN's are also advancing in retaining spatial locations of the pixels which helps in problems like object detection etc. The next step is making fine inference at each pixel, which requires dense predictions as in semantic segmentation. However, conventional convolutional neural networks cannot perform on pixel level classification due to short comings like loss of spatial information and computational expenses.

Semantic segmentation involves capturing fine pixel level classification while retaining locations. So, it involves tradeoff between spatial features and semantics. Higher level features define the nature of the image (identifies the content of the image like cat, book etc.) while deep features retain the location of the image (where an object is located in image). Fully convolutional networks with the help of skip architectures can classify at pixel level while retaining spatial features of the image. Skip architecture can combine shallow information with deep coarse information.

Fully convolutional networks (FCN) [1] are convolutional neural networks without fully connected layers. The traditional CNNs consists of fully connected layer which throws away the spatial information of the image and only retains the global information which helps in classification which are replaced by deconv layers in fully convolutional networks. The main difference between FCN and CNN is that FCNs are learning filters from end to end. The decision-making layers of CNNs are replaced by filters in CNNs. In semantic segmentation, a fully convolutional network trained pixel to pixel can perform better than state of art methods. A FCN architecture can take arbitrary size inputs and predicts dense outputs using local spatial details.

In this project, we have implemented two network models for semantic segmentation. First network model is built using FCN32 which indicates fully convolutional networks with stride length of 32. Stride is the number of skips a filter make while performing convolution. It controls the convolution of the filters with the input image and retains the properties of the image. We have developed second model using FCN8 architecture. It takes a stride of 8 along with skip architecture which combines pool3 and pool4 layers to retain high level semantic details and improve precision. The semantic segmentation has been done on PASCAL VOC2012 dataset. The dataset consists of around 9993 images with pixel-level segmentation. The images are classified with 20 objects like aero plane, cat, bus etc. as shown in fig 1.

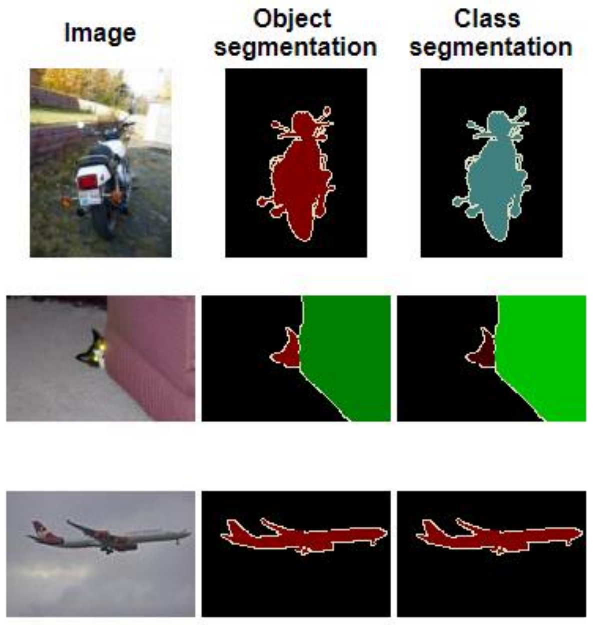

Figure 1. Sample images from PASCAL VOC2012 dataset

The rest of the report is organized as follows: Section 2 reviews related work on semantic segmentation, FCNs and various convolutional layers. The following sections explain the implemented FCN models, experimental setup and model training. In the final section, we report the performance of the models on PASCAL VOC2012 data set.

2. Related work

Our model is implemented based on fully convolutional networks and transfer learning [6, 7]. Transfer learning is method which involves learning a new task based on the transfer of knowledge acquired from an already learned task. It is reuse of a pretrained model for a new task. Transfer learning was demonstrated in classification, segmentation and detection tasks based on instance learning [9]. The model extracted from transfer learning acts as a feature extractor for the second model. As training the neural network from scratch require massive amounts of memory and computational resources, transfer learning can be used to decrease the computational expenses of the networks. For this project, we extracted the transfer learning model developed based on VGGnet architecture [8]. Then we fine tune the model on PASCAL VOC2012 data and couple with fully convolutional networks.

Patch wise training [13] is used initially for semantic segmentation tasks which can reproduce any distribution of the inputs. But the efficiency degrades when the overlap of the objects in the image increases. With intervene of fully convolutional networks, the segmentation is made possible even with high overlap between objects. To our knowledge, an end-end FCNs are first developed by Jonathan Long et al. [1]. The initial idea of FCN is used in classic LeNet [5] which identifies multiple handwritten digits. However, it is confined to 1D strings and used Viterbi decoding to recognize digits. The idea of capturing spatial information is extended from 1D to 2D strings by mapping detection scores by Wolf and Platt [10]. However, these two ideas couldn't capture fine grain level information from the image. Fully convolutional networks overcome this problem by the score maps (coarse) obtained by backward propagation.

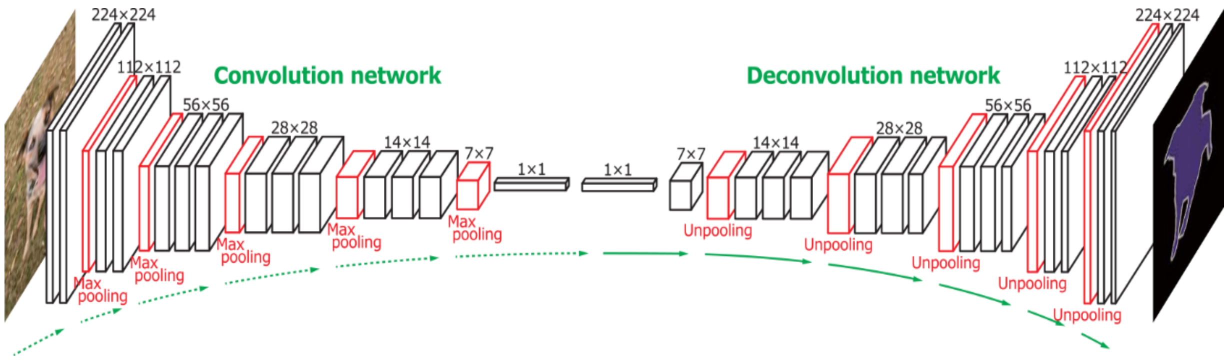

Figure 2. Fully Convolutional Networks architecture with pooling and convolution layers.

3. Fully convolutional networks

3.1 Model architecture

We have implemented two models based on fully convolutional networks for semantic segmentation. The first model involves FCN with stride 32 whereas in the second model, we tried to improve the performance by using stride 8 and introducing skip models. The fully convolutional networks are built using CNNs with two parts - convolution and deconvolution networks. For the project, we have used VGGnet architecture for the convolution network. The VGGnet architecture consists of 16 layers with 14 convolution layers and 2 fully connected layers. The architecture utilizes 3x3 filters for convolution which decrease the noise in the background significantly and creates high receptive field. Our model utilizes 13 convolution layers along with maxpool layers and rectified linear units (ReLU). The VGGnet consists of 2 fully connected regions which throws away spatial information and are instrumental in classification task. However, for semantic segmentation, the spatial information should be preserved. So, the 2 fully connected layers are discarded and replaced with 1x1 convolution layers. Then we feed the output of the convolution part to deconvolution part which consists of deconvolutional layers and unpooling layers. The convolution part generates multi-dimensional features from the input which acts as a feature extractor for the architecture. This is fed into deconvolution network which acts as a shape generator and results in segmentation of the image. The resultant obtained from the deconvolution part is a probability map, which estimates the pixel probability of classes.

3.1.1. Convolution network

The images are resized into dimensions of size 224x224 and fed into convolution network. The input is convolved with 64 filters of size 3x3. The convolved images are activated with Rectified linear units which introduces non-linearity into the systems and prevents the network from saturating. Then we use maxpool layers which reduces the dimensions to 128x128. The maxpool layers shrinks multiple activations to a single pixel which retains the robust pixel of the window. These three functions are considered a block and we repeat these block functions four times by varying number of filters and continuously decreasing the dimension size. The final layer of the convolution consists of size 7x7 and 512 stacked layers. We used dropout ratio of 0.5 during back propagation to prevent overfitting.

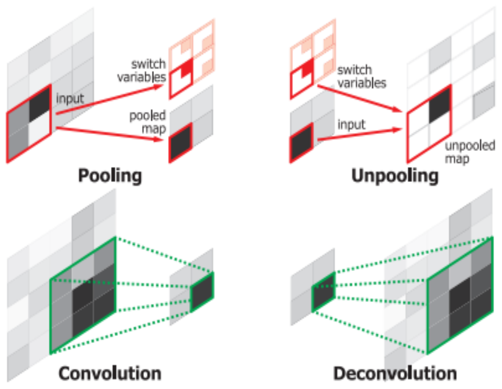

Figure 3. Simple illustration of pooling and convolution.

3.1.2. Deconvolution network

The output of the convolution layer is fed into deconvolution layer as shown in fig 2. The deconvolution network operates in the opposite way of convolution network. It consists of deconvolutional layers and unpooling layers. Maxpooling layers which are used in convolution network retains only robust activations and filters out noisy activations by choosing the only strong activations in a receptive field. However, in this process, it losses the spatial information of the pixels which cannot be used in semantic segmentation. To overcome this problem, we use unpooling layers in deconvolution networks. The unpooling network performs reverse operation by tracking the locations of maxpool activations and at reconstruct the image by pooling the activations to the original location. This helps to retain the spatial information of the images. The unpooling layer results in a sparse activation map.

The deconvolution layer converts these sparse layers to dense activation maps by performing reverse operations of a convolution layer. The convolution layer performs many to one mapping where various activations within a filter are connected to a single activation function. Whereas the deconvolution layer performs one to many mapping where one activation is mapped to multiple outputs as shown in fig 3. This enlarges the input from unpooling layers and converts sparse activation to a dense activation map. To retain the dimensions of the images, we crop the resultant map to the size of the input. The unpooling layers combined with deconvolutional layers reconstructs the input image while retaining the spatial information and performs semantic segmentation.

3.2 FCN-32 and FCN-8s architectures

While fully convolutional network described in the previous section works fine in segmenting, still the result is coarse. So, to improve the performance, we implemented FCN32 model. In FCN32, at each convolution, the filter takes a stride of 32. It indicates that while performing convolution, the filter will skip 32 subsequent values between each operation. Also, while performing deconvolution, we used upsampling to transform coarse pixels to dense network. The upsampling of a sample is given by

$$y_{ij} = \sum_{\alpha,\beta=0}^{1} \left|1-\alpha-\left\{\frac{i}{f}\right\}\right| \left|1-\beta-\left\{\frac{i}{j}\right\}\right| x_{\lfloor \frac{i}{f} \rfloor+\alpha,\ \lfloor \frac{j}{f} \rfloor+\beta}$$

Where {.} is fractional part and f denotes up sampling factor.

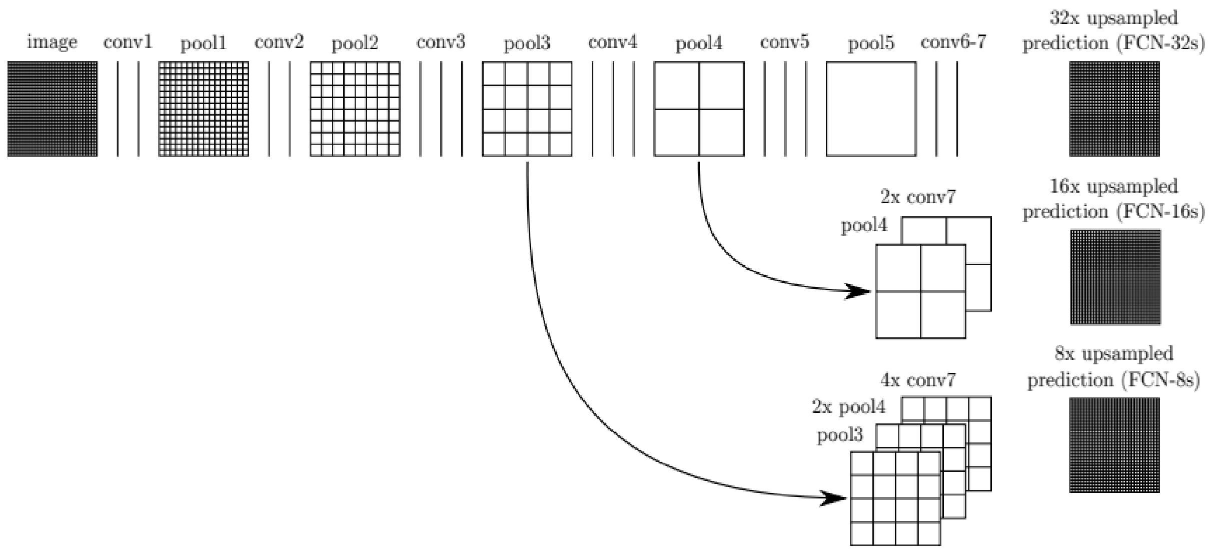
image
conv1
pool1
conv2
pool2
conv3
pool3
conv4
pool4
conv5
pool5
conv6-7
32x upsampled prediction (FCN-32s)
2x conv7
pool4
16x upsampled prediction (FCN-16s)
4x conv7
2x pool4
pool3
8x upsampled prediction (FCN-8s)

Figure 4. Implemented FCN-32s and FCN-8s architectures along with skip models.

To further improve the efficiency of the system and to increase the fine level details, we implemented FCN8 model which takes a stride of 8 steps. While following the hierarchical structure of FCN32, information loss occurs at different layers of the network. To prevent this loss, in FCN8, we use skip architectures. As shown in fig 4, we combine the predictions from pool4 layer and first layer of the deconvolution network. The output is fed into second deconvolution layer and then we combine pool3 layer. They retain semantic information while improving the precision of the prediction. Then we upsample the resultant to match the dimensions of the input image.

4. Model implementation

4.1. System requirements

We have used Keras, an open source neural network library to train and test the model. We used TensorFlow as a backend to Keras. We used python as programming language. The models require at least 16GB ram and GPU to train the model due to complexity of neural network architectures. We have trained the model for over 4 hours for each architecture.

4.2. Training and testing

We have trained our model on PASCAL VOC2012 dataset which contains 9993 images. We have used 5500 images for training and 2000 images for validation. The remaining images are used to test the output. Initially we have used transfer learning to acquire the weights from the VGGnet architecture pre-trained on the data set. Each convolution layer is treated with a ReLu layer, max pooling layer and a local normal response layer. We have used Rectified linear unit (ReLu) as activation function which helps in increasing the speed of convergence of the stochastic gradient descent. The resultant is treated with a Max pooling layer. It is used to reduce the spatial dimensions of the model which results in decrease of number of parameters and computation involved in the model. Local response normalization layer (LRN) has been used which helps in contrasting high frequency features with that of low frequency features. This results in eliminating the noise in the background. Drop out layers with drop out ratio 0.5 are used to prevent overfitting. The sample outputs for FCN32 and FCN8 are shown in fig 5 and fig 6 respectively.

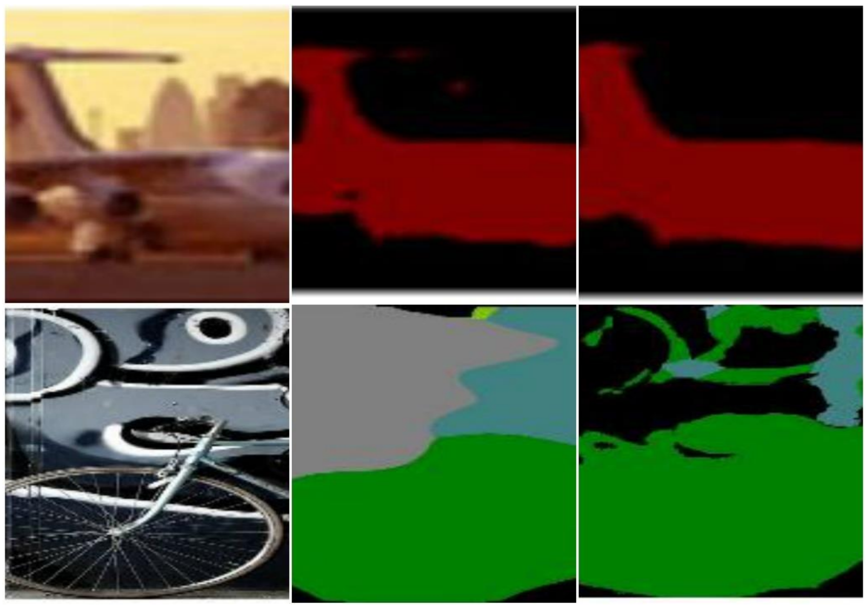

Figure 5. Input image, predicted FCN-32s and FCN-8s respectively.

4.3. Adam optimizer:

A mini batch of size 25 is fed into the network. We used Adam optimizer to train the network. Adam optimizer is a combination of Adaptive gradient algorithm (Adagrad) and Root Mean Square propagation algorithm (RMSprop). Adagrad algorithm uses adaptive learning technique which weighs heavily the infrequent parameters and vice versa which makes it robust while training on sparse data. RMSprop updates weights based on exponential decay of gradients and it performs well on noisy data. Adam optimizer take leverage of both these algorithms which makes it robust to both coarse and noisy data. For FCN32, we used learning rate of 0.001 whereas for FCN8, a learning rate of 0.01 is used which were identified by cross validation.

5. Results

We have tested the models on over 2000 images. To test the performance and accuracy of the models, we have used three metrics: Pixel accuracy, mean accuracy and Intersection over union. Pixel accuracy calculates number of correctly classified pixels to over all pixels. Mean accuracy gives the number of correctly classified pixels per each class. Intersection over Union computes the area of overlap of a predicted class to original class.

Pixel accuracy: $\sum n_{ii} / \sum t_i$

Mean accuracy: $(1/n_{class}) \sum n_{ii} / \sum t_i$

Intersection over Union: $\frac{True\ positive}{True\ positive + False\ positive + False\ negative}$

Where n_{ij} represents pixels predicted as class i where the pixels belong to class j. t_i represents total number of pixels.

With FCN-32s, we have achieved pixel accuracy of 58.8% and IoU of 52.5%. There is high Intersection of different classes while segmenting the image and subsequently it affected the performance of the model. While FCN-8s uses the skip architecture, it tries to prevent information loss between layers and also improve grouping of similar pixel classes in the image. As a result, the pixel accuracy increased to 63.6% and the IoU increased to 57.8%.

Table 1. Performance evaluation of FCN-32s and FCN-8s models

Metrics	FCN-32s	FCN-8s
Pixel accuracy	58.8	63.6
Mean accuracy	55.2	61.2
IoU	52.5	57.8

6. Conclusion

Semantic segmentation is one of the active areas of research in the field of image processing. It requires pixel level classification and ability to distinguish pixels of different objects makes it a challenging task. With Fully convolutional networks and skip models, pixel level classification is made possible. In this project we implemented FCN-32s and FCN-8s models for semantic segmentation. In these models, we used techniques like deconvolution, upsampling and skip architectures to improve the segmentation quality and to preserve spatial information. We have tested the models on PASCAL VOC2012 data set and the results were reported. Even though we have achieved significant results, the efficiency of the models can be improved a lot. In future, we plan to combine Conditional random fields with FCN-8s model to improve the performance of the models.

References

[1] Long, Jonathan, Evan Shelhamer, and Trevor Darrell. "Fully convolutional networks for semantic segmentation." *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2015.

[2] Data set: http://host.robots.ox.ac.uk/pascal/VOC/voc2012/index.html

[3] D. C. Ciresan, A. Giusti, L. M. Gambardella, and J. Schmidhuber. Deep neural networks segment neuronal membranes in electron microscopy images. In NIPS, pages 2852–2860, 2012.

[4] J. Carreira, R. Caseiro, J. Batista, and C. Sminchisescu. Semantic segmentation with second-order pooling. In ECCV, 2012. 9

[5] Matan, Ofer, et al. "Multi-digit recognition using a space displacement neural network." *Advances in neural information processing systems*. 1992.

[6] Raina, Rajat, et al. "Self-taught learning: transfer learning from unlabeled data." *Proceedings of the 24th international conference on Machine learning*. ACM, 2007.

[7] Bengio, Yoshua. "Deep learning of representations for unsupervised and transfer learning." *Proceedings of ICML Workshop on Unsupervised and Transfer Learning*. 2012.

[8] Simonyan, Karen, and Andrew Zisserman. "Very deep convolutional networks for large-scale image recognition." *arXiv preprint arXiv:1409.1556* (2014).

[9] R. Girshick, J. Donahue, T. Darrell, and J. Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In Computer Vision and Pattern Recognition, 2014.

[10] R. Wolf and J. C. Platt. Postal address block location using a convolutional locator network. Advances in Neural Information Processing Systems, pages 745–745, 1994.

[11] Krizhevsky, Alex, Ilya Sutskever, and Geoffrey E. Hinton. "Imagenet classification with deep convolutional neural networks." *Advances in neural information processing systems*. 2012.

[12] Szegedy, Christian, et al. "Going deeper with convolutions." Cvpr, 2015.

[13] Dong, Chao, et al. "Learning a deep convolutional network for image super-resolution." *European Conference on Computer Vision*. Springer, Cham, 2014.