Urban Scene Image Segmentation Based On Fully Convolutional Neural Network

Chung-Han Yang
Arizona State University
cyang122@asu.edu

Han-Zhang Wang
Arizona State University
hwang356@asu.edu

Jia-tong Zhang
Arizona State University
jzhan404@asu.edu

Abstract

This article describes the EEE598 course project of phase 2. This project includes data processing, image segmentation and deep learning. This project involves a large image data set containing more than 20000 images in many different locations. Although the traditional methods of image processing may offer promising results in certain scenarios, but to give better result on all data, multiple deep learning algorithms are being implemented and examined.

Keyword: *Data Processing, Image Segmentation, Semantic Segmentation, Deep Learning, AlexNet, VGG11*

1. Introduction

Image segmentation is a important application in computer vision. In this project, we implement the image semantic segmentation for urban scene. The goal of semantic image segmentation is to label each pixel of an image with a corresponding class of what is being represented. Because we're predicting for every pixel in the image, this task is commonly referred to as dense prediction. Semantic segmentation is useful in many technical field like autonomous vehicles, medical image diagnostic and so on. We use deep learning to implement this task and have two models lead to different result and accuracy. Most neural network models are combined by convolutional layers and fully connected layers. To implement semantic segmentation, we used deep convolutional neural network with minor modifications[3]. Note that CNN is a very popular and primary method to solve image classification problem. However in this project, the fully connected layers are being discarded and instead used deconvolutional layers[4].

With the development of autonomous vehicles, image segmentation has more importance. Urban scene traffic segmentation can help vehicles to classify many objects. In this paper, we classify each pixel to a certain class, and compute the pixel accuracy to evaluate the performance of network. The example of urban scene Image segmentation is shown in Figure 2.

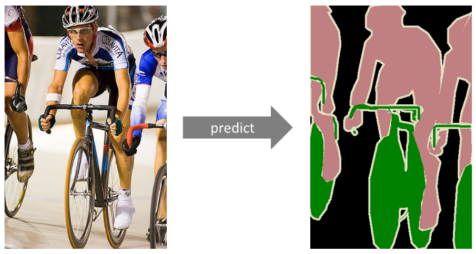

Figure 1. Example of semantic segmentation[2]

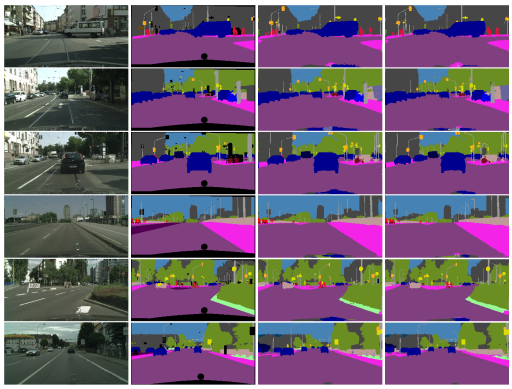

Figure 2. Example of urban segmentation[2]

2. Semantic Segmentation and Fully Convolutional Neural Convolutional Network

We just discussed that the goal of semantic segmentation is to label each pixel of an image with a corresponding class of what is being represented, that is to said we want to take a RGB image $width \times height \times 3$ and output a segmentation map $width \times height \times 1$ where each pixel contains a class label represented as an integer. Many technique help us to do that like one-hot encoding. If we

want a good result, we must have perfect ground truth label. The dimension of the label must be the same as the original image. The example of semantic label are shown in the Figure 3:

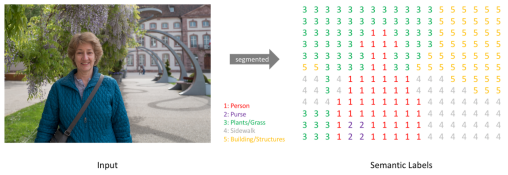

Figure 3. Example of semantic label

There are several neural network model that help us to do semantic segmentation like AlexNet[1], VGG11, VGG16, GoogLeNet, ResNet and so on. A general semantic segmentation can be thought of as an encoder network followed by a decoder network. The encoder layer is usually a pre-trained classification network like AlexNet/VGG followed by a decoder network and the task of a decoder network is to semantically project the discriminative features (lower resolution) learned by the encoder onto the pixel space (higher resolution) to get a dense classification. Unlike typical classification where the end result of the neural network is only important thing, semantic segmentation not only requires discrimination at pixel level but also a mechanism to project the discriminative features learned at different stages of the encoder onto the pixel space. In this project, we focus on fully convolutional network(FCN)[2] method. The FCN learns the mapping the mapping from pixel to pixel. The FCN network pipeline is an extension of the classical CNN. The main idea is to make the classical CNN take as input arbitrary-sized images. The restriction of CNNs to accept and produce labels only for specific sized inputs comes from the fully-connected layers which are fixed. Contrary to them, FCNs only have convolutional and pooling layers which give them the ability to make predictions on arbitrary-sized inputs. The architecture of FCN is shown in Figure 4.

One issue in this specific FCN is that by propagating through several alternated convolutional and pooling layers, the resolution of the output feature maps is down sampled. Therefore, the direct predictions of FCN are typically in low resolution, resulting in relatively fuzzy object boundaries.

3. Models

3.1. AlexNet

AlexNet is the first deep convolutional neural network that competed in the ImageNet Large Scale Visual Recog-

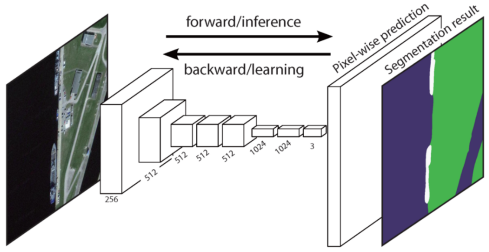

Figure 4. FCN architecture

nition Challenge in 2012. This neural network gained top-5 error of 15.3%. Alexnet is designed by Alex Krizhevsky, llya Sutskever, and Geoffrey E.Hinton[1].The input of AlexNet is an 227×227 RGB image. In the Original AlexNet, to be compatible with the fully connected layers, the size of input image should be 227×227. But in our Fully convolution network, the input size only depends on the memory of your GPU. The size of our train image is 1024×2048. To obtain a better batch-size, we resize the Image to 512×1024. The example is shown in Figure 5.

Figure 5. Resize implemention

3.2. AlexNet Network Structure

Alexnet has **5 Convolution Layers** and **3 Fully Connected Layers**.The **Convolution Layers** will extract the feature of input image and the **Fully Connected Layers** will integrate the feature to infer the output. The structure of Alexnet is shown in figure 6. Every convolution layer is followed by one RELU layer and one Max-pooling layer. RELU layer contribute to the non-linearity of the network, and Max-pooling layer can shrink the size of image to capture the most important feature. The following three fully connected layers will classify the image into certain class. The last layers is called softmax layer. This layer compute the probability of every output.

3.3. VGG11

VGG was first introduced by Simonyan and Zisserman in their 2014 paper[**?**], **Very Deep Convolutional Networks for Large Scale Image Recognition**.VGGNet uses 3×3 convolution kernels with increasing depth to scan every

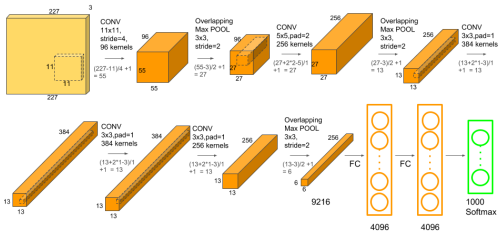

Figure 6. AlexNet Structure

small part of input image. The VGG11 network has 8 convolution layers and 3 fully connected layers. The structure of VGG11 is shown in figure 7.

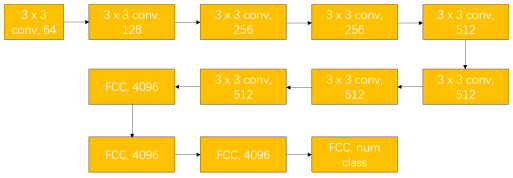

Figure 7. VGG11 Structure

4. Model construction and train method

4.1. Model construction

In this project, we used the models mentioned above, the detail of construction will be list here. In the AlexNet model, we use a pretrained model from pytorch and get rid off the last fully connected layer, deconvolutional layers instead. So there are in total of 5 convolutional layers and 3 deconvolutional layers, the dataset sent to the neural network are all used to train the deconvolutional layers. There are pooling layers inserted after convolutional layers to down size the outputs. In the VGG11 model, a pretrained model from pytorcg is also being used. Original structure of VGG11 is in total of 8 convolutional layers with 3 fully connected layers. We keep the convolutional layers and drop off the fully connected layers and use deconvolutional layers instead.

4.2. Train method

In this project, one of the most difficult problem is the unbalanced data of each class. For example, in one urban traffic image, road and building will occupy most place. Thus, more than 80% of the pixel is road and building. If network is not constrained, it will classify every pixel either road or building. The consequence of unbalanced data is the network can obtain accuracy more than 80%, but the output image is not segmentation. To solve this problem, we implement weight rescaling method to compute the size of every class and change the weight of each class. Thus, small class will have larger weight. The unbalanced data problem can be solved.

For the choice of optimization algorithm, we choose Adam. Adam was presented by Diederik Kingma from OpenAI and Jimmy Ba from the University of Toronto. Adam takes different approach to optimize the network compared to classical stochastic gradient descent. Classical stochastic gradient descent method will maintain a certain learning rate to optimize network. Adam utilize first moment(the mean) and second moment(the uncentered variance) to estimate learning rate. The parameter of learning rate in Adam only influence the initial update. When implement different method on MNIST, Adam has the best performance.

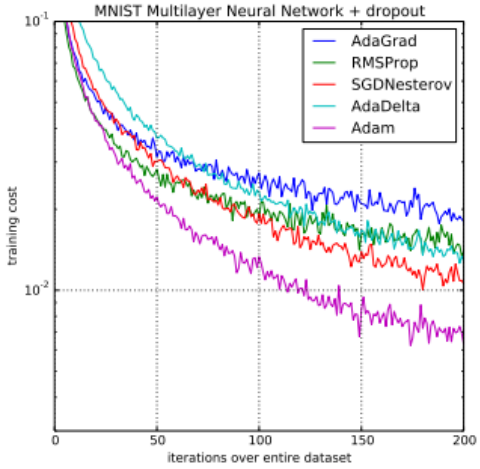

Figure 8. Performance of optimization algorithm

5. Dataset

The dataset being used in this project is called the Cityscapes Dataset. It's a new large scale dataset that contains a diverse set of stereo video sequences recorded in street scenes from 50 different cities, with high quality pixel level annotations of 5000 frames in addition to a larger set of 20000 weakly annotated frames. The dataset is thus an order of magnitude larger than similar previous attempts. The reason behind why this particular dataset is being chosen is first, this dataset works really well with assessing the performance of vision algorithms for two major tasks of semantic urban scene understanding: pixel-level and instance-level semantic labeling. Secondly, this dataset also supports research that aims to exploit large volumes of annotated data, e.g. for training deep neural networks.

We have preprocess all the ground truth label from typical PNG file in to labeled files usually a 2D matrix, we

ID	NAME	ID	NAME
0	ROAD	10	SKY
1	SIDEWALK	11	PERSON
2	BUILDING	12	RIDER
3	WALL	13	CAR
4	FENCE	14	TRUCK
5	POLE	15	BUS
6	TRAFFIC LIGHT	16	TRAIN
7	TRAFFIC SIGN	17	MOTORCYCLE
8	VEGETATION	18	BICYCLE
9	TERRAIN	19	BACKGROUND

Table 1. Class table

implemented it by matlab. The following shows our in total of 20 classes

6. Validation Process

In this project we ran 5 epochs, batch size is 5, the validation set we used containing 500 images, after each 500 iterations we do validation process to compute pixel accuracy and loss. The way used to compute loss is negative log likelihood loss(NLL loss), shown in the following:

$$loss(x, class) = -weight[class] \times x[class]$$

Where the weight is being manually re-scaled to each class. The pixel accuracy is given in the following:

$$\frac{\sum_i n_{ii}}{\sum_i t_i}$$

Where n_{ii} is the number of pixels of class i but predicted to belong to class i,t_i is the total number of pixels belong to class i.

7. Platform

Since the availability of the powerful Graphics Processing Units (GPUs) will be a crucial key to train a deep convolutional neural network. We use the compute engine with one NVIDIA GTX 1070ti GPU. Note that the memory size of 1070ti GPU is 8GB, so normally it has ability to ac- commodate most of implementation in deep convolutional architectures. The training speed is also adjust the banch critical factor to the success in deep learning. To build networks, we choose to use Pytorch as our library due to the coding convince, so we sacrificed the tract abilities for tensorflow with TensorBoard which is able to visualize and track the training progress in terms of accuracy and loss. It took a lot of work to debug when there is something wrong while training.

8. Simulation And Result

The dataset contains 19998 images for both training and testing. Both Alexnet and VGG11 were trained in all image sets. From Fig.9 until Fig.14, two sets of original image with testing result from Alexnet and VGG11 are shown and compared. In general, the VGG11 has better result compare to Alexnet.

Figure 9. Original data 1

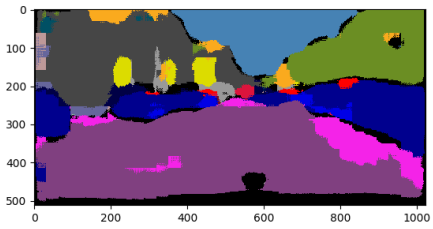

Figure 10. Alexnet prediction result for data 1

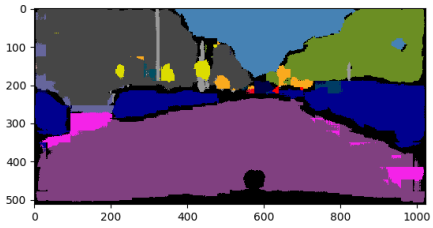

Figure 11. VGG11 prediction result for data 1

Since the pixel accuracy evaluation is being used in this project as mentioned above, a set of more detailed Accuracy vs. Validation Time plot is being provided below. And

Figure 12. Original data 2

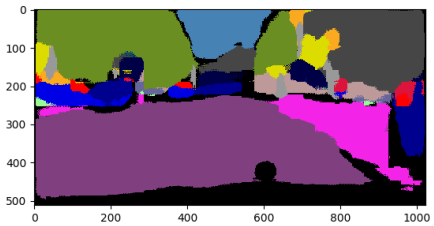

Figure 13. Alexnet prediction result for data 2

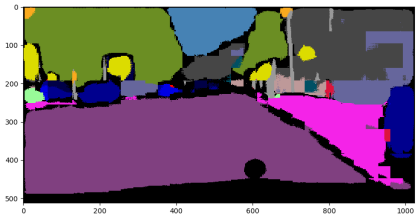

Figure 14. VGG11 prediction result for data 2

the exact numbers are also provided in the following table 2. However, even with the promising overall accuracy of the final prediction output, there are still minor problems that are very noticeable within the prediction results. One big issue is the weight distribution of different class labels are still not optimized yet, so many small classes like fence, pole or person are still experiencing a hard time to be detected and often being recognized as other random surrounding objects.

From the Accuracy graphs shown in Fig.15 and Fig.17 for Alexnet and VGG11 accordingly, it is very clear that the overall results of VGG11 is better than Alexnet. The overall advantage of using VGG11 in terms of accuracy is approximately 5 percent.

Loss of AlexNet	0.7156499028205872
Acc of AlexNet	0.826844736922711
Loss of VGG11	0.43420249223709106
Acc of VGG11	0.888389542357948

Table 2. Overall evaluation of Deep Learning method

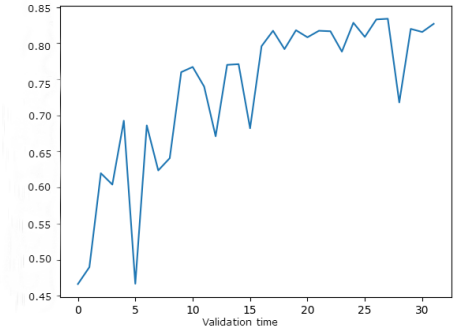

Figure 15. Accuracy plot during training for Alexnet

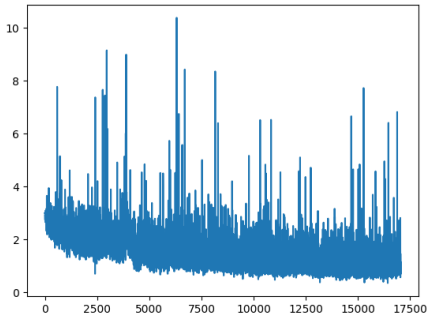

Figure 16. Loss function plot during training for Alexnet

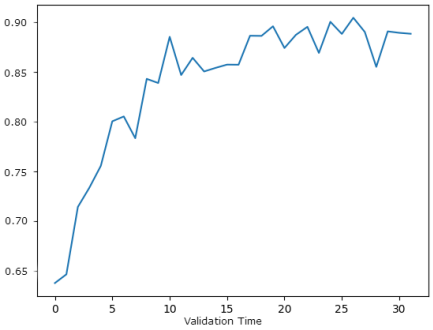

Figure 17. Accuracy plot during training for VGG11

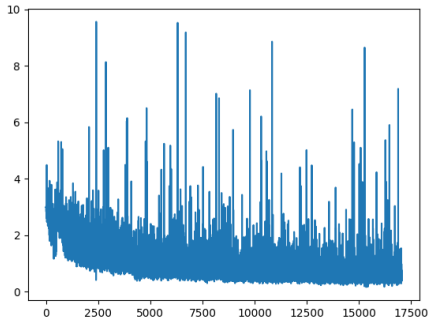

Figure 18. Loss function plot during training for VGG11

9. Future Work

Due to the limitation of time and resource, we spent much time on training process and did not able to find the proper value of parameters. Hence, we hope to complete the following works in the future:

- Use a more powerful GPU to deal with larger scale of data to make the result more precise even train our own classification model.
- Adapt the system into real-time which can really use to solve the real technical problem.
- Implement this project with newer and more efficient models such as RNN and LSTM.
- Make more detail classification like enlarge the number of class.
- Use different dataset to train the model make it more flexible.
- Make the use of skip-connection which will further increase the final result.

10. Conclusion

Implemented AlexNet and VGG-11 with Pytorch to train image to realize image semantic segmentation with Adam optimization algorithm and use weight rescaling method to converge the network and solve unbalanced data problem. Showed the comparison result between AlexNet and VGG-11 and observe the fact that VGG-11 performed better than the AlexNet. Alexnet obtain 82% pixel accuracy and VGG-11 obtain 88% accuracy.

It is known that AlexNet and VGG are similar deep learning model, but VGG has more convolution layers and is deeper than AlexNet. This often leads to better and higher feature extractions, which makes it a very if not most popular model. Also from this project, it is clear that more data pass into the training phase will increase the final result with fine turned parameters. Also with some simple testing, the skip connection is also a very efficient way to improve the segmentation detail because of fusing information that has been loss during pooling operation. In the future, we will try several methods mentioned above to modify our results.

11. Contribution

- Han-Zhang Wang:Training data preprocessing, build training model for training by pytorch, dataset check.
- Jia-Tong Zhang:Training data preprocessing, training model research, dataset check.
- Chung Han Yang:Training data preprocessing, build training model by tensorflow, dataset check.

References

[1] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems*, page 2012.

[2] J. Long, E. Shelhamer, and T. Darrell. Fully convolutional networks for semantic segmentation. In *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3431–3440, June 2015.

[3] L. B. Y. LeCun, P. Haffner and Y. Bengio. Object recognition with gradient-based learning.

[4] M. D. Zeiler and R. Fergus. Visualizing and understanding convolutional networks. In *European conference on computer vision*, 2014.