Urban Scene Segmentation for Autonomous Vehicles

Hsiao-Chen Huang
UC San Diego
hsh030@eng.ucsd.edu

Eddie Tseng
UC San Diego
edtseng@eng.ucsd.edu

Ping-Chun Chiang
UC San Diego
p3chiang@eng.ucsd.edu

Chih-Yen Lin
UC San Diego
chl886@eng.ucsd.edu

Abstract

For the past few decades, full automation has become an ultimate goal that people want to achieve, and the development of autonomous vehicles (AVs) is one of the important progress in our lives. Understanding the surrounding and making decisions are crucial concepts for AVs. We deeply believe that the computer vision techniques can help us solve these problems. The first challenge that stuck out to us is how to distinguish roads, people and cars, or even traffic signs. Hence, we aim to implement semantic segmentation in this project.

Our main idea is to use fully convolutional networks to perform pixel-wised classification in urban scene images. Though there have been much work in image segmentation, we would like to implement and evaluate the performance of several contemporary models such as AlexNet and VGG net into fully convolutional networks.

1. Introduction

Scene segmentation can be defined as a problem of end-to-end classification and it is also an important task in the field of computer vision. While similar to normal object recognition task, the goal of scene segmentation is not to find a bounding box of recognized objects but to label each pixel according to which object or class it belongs to. Scene segmentation can come in many different settings. When the requirement is only to label each pixel according to class, it can be described by other names such as pixel-level labeling or **pixel-wise classification**. In this project, we try to make dense predictions, which can label each pixel with the class of its enclosing object. The prediction provides not only the classes but also additional information regarding the spatial location of those classes. Furthermore, we can also apply object detection or tracking based on the results.

Moreover, scene segmentation plays a very important role in the context of autonomous driving. The use case often requires captioning on a video stream, but this is not our focus in this project and we will evaluate our system in an offline setting. In this experiment we started with **fully convolutional network(FCN)** which was first proposed by Long *et al.* [7]. We use **pixel accuracy**(we will mention in 4.1) as the benchmark to measure our performance because it is easy to compute and provides an intuitive approximation of human perception of the performance. The best results we achieved are 89% pixel accuracy on training data and 88% on testing data.

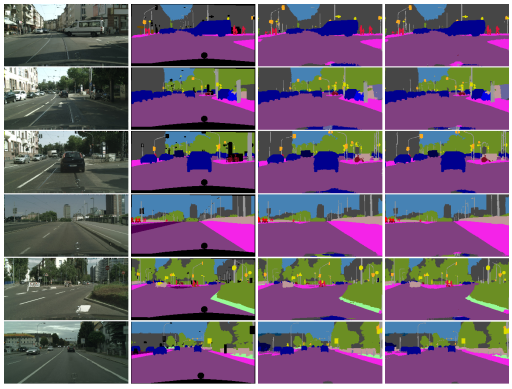

Figure 1: Example of urban scene segmentation [2]

2. Related Work

2.1. Prediction

Given a set of input images, $\mathcal{X}$ that contains N samples and each sample $\mathbf{x}_n$ is a 256×512 image.

$$\{\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_N\} \in \mathcal{X} \tag{1}$$

The corresponding annotation $\mathcal{Y}$ is represent as

$$\{\mathbf{y}_1, \mathbf{y}_2, ..., \mathbf{y}_N\} \in \mathcal{Y} \tag{2}$$

Each annotation of a sample $\mathbf{y}_n$ contains $c + 1$..., which c is the number of class and $c + 1$ is the background. The algorithm we applied is One-Hot encoding and we will talk about this later.

$$\mathbf{y}_n = \{(\mathbf{c}_0, \mathbf{c}_1, ..., \mathbf{c}_c, \mathbf{c}_{c+1}) | \mathbf{c}_i \in [0, 1]\}, \forall \mathbf{n} \tag{3}$$

In order to find corresponding position based on neural network, we define a decision function (Eq. 4) which would learn to minimize the structural risk:

$$\mathbf{g} : \mathcal{X} \rightarrow \mathcal{Y} \tag{4}$$

We use **sigmoid function** (Eq. 5) as our activation function, so that all the outputs will lie in between 0 and 1.

$$\mathbf{S}\left(x\right) = \frac{1}{1 + e^{-x}} \tag{5}$$

Evaluate **cross entropy** as loss function:

$$L[\mathbf{y}, \hat{\mathbf{y}}] = -\sum_{i} \mathbf{y}_i \log_2 \left(\hat{\mathbf{y}}_i\right) \tag{6}$$

2.2. Convolutional Neural Network

Deep Convolutional Neural Network (**CNN**) is a popular and primary method to solve image classification problem. It is first proposed by **LeCun** *et al.* [4]. Convnets are built with **convolutional** layers, **pooling** layers, and **fully-connected** layers. The convolutional and pooling layers are used to extract features by filtering and extending the depth of feature dimension. Each convolutional layer is made up of a set of neurons that we can consider it as a transfer mapping by a filter. Each neuron connects to a small region of neurons in the previous layer with different weights.The set of filters have small width and height, and depth is the feature dimension. In fully-connected layers, the output will be reduced to a single vector of probability scores along the depth dimension.

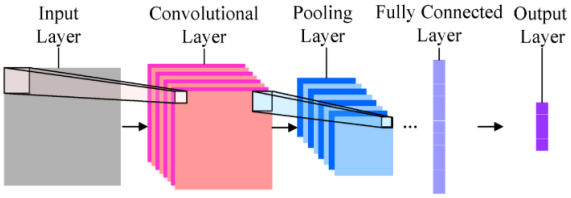

Figure 2: The architecture of CNN[6]

However, appending a fully-connected layer enables the network to learn by using global information where the spatial arrangement of the input falls away and it is harmful for dense classification tasks such as image segmentation. Hence, we mention a proper method in section 2.3 to achieve our goal.

2.3. Fully Convolutional Network

Fully convolutional indicates that the neural network is composed of convolutional layers without any fully connected layers. The main difference between CNNs and FCNs is that the fully convolutional networks is learning filters everywhere. A fully convolutional network can learn representations and make decisions based on local spatial input. After passing through the convolutional operations the same as CNN, hooking a bunch of deconvolutional layers, and then running gradient descent allows learning functions which couldn't be learned with a pure convolutional network, for instance for semantic segmentation[5]. In this project, data for each layer in a convolutional net is a three-dimensional array of size $h \times w \times d$, where h and w are the spatial dimensions, and d is the feature dimension [7]. Inputs are images with pixel size 256×512, and 3 color channels.

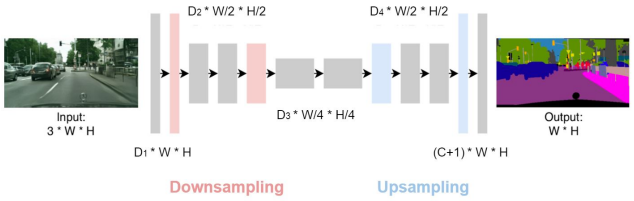

Figure 3: Design network as a bunch of convolutional layers with downsampling and upsampling inside the network

2.3.1 1x1 Convolutional Layer

Traditional convolutional networks cannot manage different image sizes since fully-connected layers, while fully convolutional networks can manage different image sizes because it only contains convolutional layers. Basically 1x1 convolutions serve several purposes:

- They perform dimensional reduction in a fast manner. With a 1x1 convolution, we can rather cheaply reduce the number of channels without losing information.
- They are used in depthwise separable convolutions to factor the convolutional layer along the depth axis, reducing computation and reducing memory requirements and often slightly improving accuracy.
- One argument is that they allow for improvements in accuracy because it means that cross-channel information and spatial information can be analyzed independently by the model.

2.3.2 Deconvolution

Deconvolution is a network with stacked layers, which can be considered as a reverse of convolution. Instead of convolving a filter mask with an image to get a set of activation as in a convolutional neural net, we are trying to infer the activation that when convolved with the filter mask, would

yield the image. The learned filters in deconvolutional layers correspond to bases to reconstruct shape of our input image. Hence, they are used to capture different level of shape details to expand the image back to its original size [9].

2.3.3 Skip Connection

Skip architecture as the name suggests skips some layer in the neural network and feeds the output of one layer as the input to the next layer as well as some other layer. Different layers in convolutional operation provide us with different levels of information. We lose some resolution during the convolutional and pooling operations. It contains wider perceptual field in lower layers, and less details in higher layers. Skip connection is a method to solve the above problem, which is to add back some resolution by fusing the output from the previous layer. It is able to help lower level information to reach top level and recover the details.

3. Experiments

3.1. Dataset

In this project, The model will be trained and evaluated on **Cityscapes dataset**. This large-scale dataset 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. This website has the tremendous database, but we only use **leftImg8bit trainextra** and **gtCoarse** as our training set which contains 19998 coarse annotations and training images.

ID	Name	Color(R, G, B)	ID	Name	Color(R, G, B)
0	road	(128, 54, 128)	10	sky	(128, 54, 128)
1	sidewalk	(244, 35, 232)	11	person	(220, 20, 60)
2	building	(70, 70, 70)	12	rider	(255, 0, 0)
3	wall	(102, 102, 156)	13	car	(0, 0, 142)
4	fence	(190, 153, 153)	14	truck	(0, 0, 70)
5	pole	(153, 153, 153)	15	bus	(0, 60, 100)
6	traffic light	(250, 170, 30)	16	train	(0, 80, 100)
7	traffic sign	(220, 220, 0)	17	motorcycle	(0, 0, 230)
8	vegetation	(107, 142, 35)	18	bicycle	(119, 11, 32)
9	terrain	(152, 251, 152)	19	background	(0, 0, 0)

Figure 4: The color map of each object

3.1.1 Dataset exploration

The training dataset provides **png** images, where pixel values encode labels. In order to compare the prediction with true value, we make some data processing tasks with the label images. First, there are fifty files with different urban scene images, we extract all the images and resize them to 256×512, and the number of channels is 20 because we decide to segment 19 classes and the last channel is background. Each channel represents one class, and we use **One-Hot** skill to give the pixel value as one when it corresponds to the certain object.

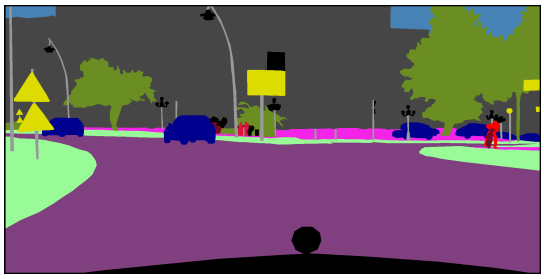

Figure 5: Example of ground truth image

The goal of One-Hot encoding is to classify an pixel into one among K classes, and we choose $K = 20$. However, in our case, we need to calculate background when we implement network. Hence, the actual number of classes is 19. The reason we use One-Hot encoding is that it transforms categorical features to a format that works better with classification and regression algorithms. This will help us improve the results when calculating the last layer of neural network.

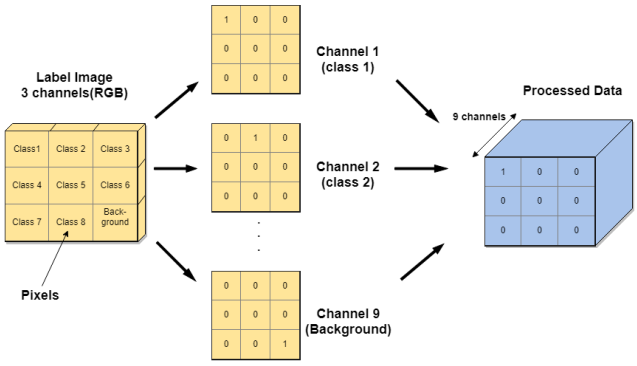

Figure 6: Example of One-Hot encoding(8 classes)

3.2. 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® **1080ti** GPU. Note that the memory size of 1080ti GPU is 11GB, so normally it has ability to accommodate most of implementation in deep convolutional architectures. The training speed is also adjust the banch critical factor to the success in deep learning. However, with

limit resource and four networks we want to compare with, we also use the computer without GPU. For our tremendous labels' data, 11 GB is still not enough. Therefore, we have to turn down the batch size, such as the maximum of batch size is 2 when we train VGG 19. Note that the processing power of 1080ti is 11.3 TFLOPS in double precision, so it is really an ideal speed up method to our expectation of training deep networks.

To build networks, we choose to use **Tensorflow** [1] as our library due to the following tractabilities. First, by using **TensorBoard**, we are able to visualize and track our training progress in terms of accuracy and loss. Secondly, by coding in Tensorflow, we get a chance to understand the details in customizing estimators and loss functions.

3.3. Models

3.3.1 AlexNet

AlexNet was first presented by Krizhevsky *et al.*[3]. Convnets are built with 8 layers, which contains **5 convolutional** layers, **3 pooling** layers, and **3 1 x 1 convolutional** layers. And it uses ReLU operation in each convolutional layer to model and add non-linearity. Also, it uses dropout to overcome the overfitting problem. With these benefits, we try to modify and apply it to this project. The architecture of AlexNet with skip connection we used are shown in Fig. 7.

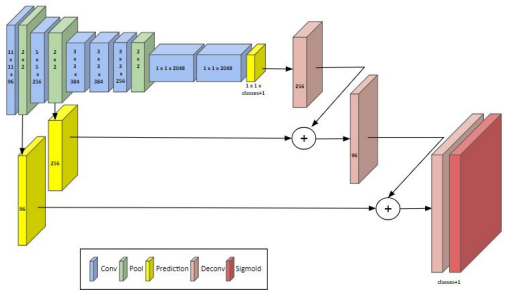

Figure 7: AlexNet with skip connection

3.3.2 VGG19

VGGNet was first presented by Simonyan *et al.*[8]. Convnets are built with **16 convolutional** layers, **5 pooling** layers, and **3 1 x 1 convolutional** layers. The architecture of VGGNet with skip connection we used are shown in Fig. 8.

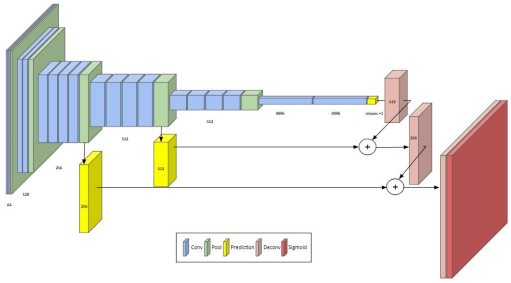

Figure 8: VGGNet with skip connection

4. Result

4.1. Evaluation Matrix

Here, we mention two methods, pixel accuracy and mean accuracy, to measure the performance of our results. Pixel accuracy presents the total accuracy by comparing each pixel in the predict image and ground truth image. In mean accuracy, we consider each pixel accuracy in each class, then take an average of those values. However, we just compute the pixel accuracy so far in this project. We hope to apply the mean accuracy in our future work.

Let n_{ij} be the number of pixels of class i predicted to belong to class j, and $t_i = \sum_j n_{ij}$ be the total number of pixels of class i. n_{cl} is considered as number of classes. The formula of pixel accuracy and mean accuracy are:

- **Pixel accuracy** : $\dfrac{\sum_i n_{ii}}{\sum_i t_i}$
- **Mean accuracy** : $\dfrac{1}{n_{cl}} \sum_i \dfrac{n_{ii}}{t_i}$

In addition, the background pixels will be ignored in evaluation part, which means the background, the last channel in label image we mention in 3.1.1, does not belong to the class i.

4.2. Comparison

The dataset contains 18,000 training and 1,998 testing images. AlexNets were trained in 13k steps and VGG was trained in 93k steps(skip) vs 41k steps(non-skip).

We compare the results of AlexNet and VGG, which are shown in Fig.9 and Fig.10. As our experiments, VGG19 with skip connection has the best accuracy (89%) and VGG19 without skip connection has the worst accuracy (59%).

In Fig.11, we show the prediction results. The first row is the result of AlexNet nonskip model, the second row is the result of AlexNet skip model, and the third row is the result of VGG skip model. By comparing the first and second row, we can see the segmenting border of AlexNet without skip

connection is blurry, but more clear with skip connection. By comparing the second and third row, we c an predict more detail objects in the deeper network, VGG19.

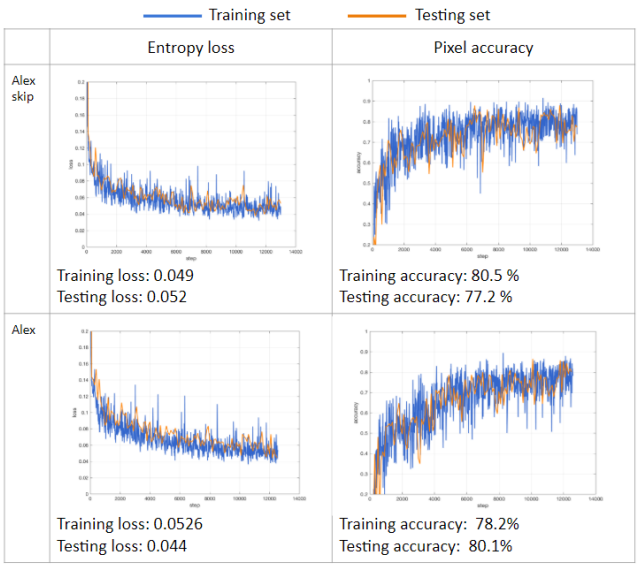

Figure 9: AlexNet comparison

Figure 10: VGG19 comparison

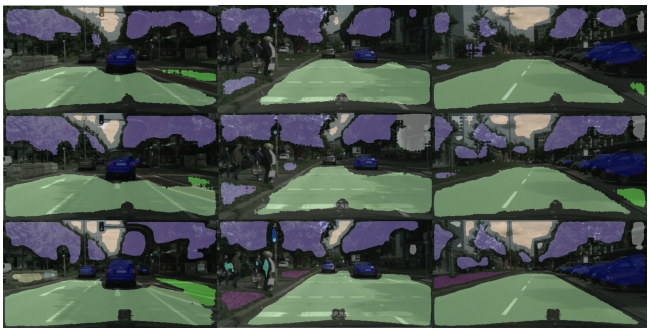

Figure 11: Prediction result.

5. Discussion

Skip Connection: The accuracy of the model with skip connection should be higher and improve the segmentation detail because of fusing information we loss during pooling operation. Also, the results shown in Fig.11 meet the expectation.

AlexNet vs. VGG net: VGG is similar to AlexNet, but more filters. Thus, VGG can extract higher features. That is why it is currently the most popular model, deep and simple.

Batch Size: Due to the limited GPU and RAM size, we trained mini-batch gradient descent with batch size equal to 5 for AlexNet and 2 for VGG. Therefore, this is the reason for large fluctuations.

Learning Rate: In this project we use $learningrate = 0.0001$. For the future work, we will tune learning rate to optimize the training speed. As the prediction results shown in 11, green indicates road, blue indicates cars, etc..

6. 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:

- Upgrade hardware by increasing the efficiency and memory of GPU
- Achieve real-time segmentation
- Implement different models (RNN, LSTM)
- Tune parameters to find the best performance
- Try more classes or use different dataset
- Calculate class mean accuracy which consider the performance of each object

References

[1] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard, et al. Tensorflow: A system for large-scale machine learning. In *OSDI*, volume 16, pages 265–283, 2016.

[2] M. Cordts, M. Omran, S. Ramos, T. Rehfeld, M. Enzweiler, R. Benenson, U. Franke, S. Roth, and B. Schiele. The cityscapes dataset for semantic urban scene understanding. *CoRR*, abs/1604.01685, 2016.

[3] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

[4] Y. LeCun, P. Haffner, L. Bottou, and Y. Bengio. Object recognition with gradient-based learning. In *Shape, Contour and Grouping in Computer Vision*, pages 319–, London, UK, UK, 1999. Springer-Verlag.

[5] H. Noh, S. Hong, and B. Han. Learning deconvolution network for semantic segmentation. In *Proceedings of the 2015 IEEE International Conference on Computer Vision (ICCV)*, ICCV '15, pages 1520–1528, Washington, DC, USA, 2015. IEEE Computer Society.

[6] M. Peng, C. Wang, T. Chen, and G. Liu. Nirfacenet: A convolutional neural network for near-infrared face identification. *Information*, 7(4):61, 2016.

[7] E. Shelhamer, J. Long, and T. Darrell. Fully convolutional networks for semantic segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 39(4):640–651, April 2017.

[8] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[9] M. D. Zeiler and R. Fergus. Visualizing and understanding convolutional networks. In *European conference on computer vision*, pages 818–833. Springer, 2014.