|  |
| --- |
| **PA3 Report: Convolutional Neural Network Implementation for Semantic Segmentation on the Cityscapes Dataset** |

**Hao XiangHuimin Zeng**

Computer Science & Engineering Computer Science & Engineering

UC San Diego UC San Diego

La Jolla, CA 92093 La Jolla, CA 92093

*haxiang@ucsd.edu zenghuimin@yahoo.com*

**Xingyi YangZhenrui Yue**

Computer Science & Engineering Computer Science & Engineering

UC San Diego UC San Diego

La Jolla, CA 92093 La Jolla, CA 92093

*x3yang@ucsd.edu yuezrhb@gmail.com*

**Abstract**

This document is the report of the first programming assignment (PA3) in CSE 253 Neural Networks/Pattern Recognition at UC San Diego in Winter 2020 by Huimin Zeng, XXX, XXX and Zhenrui Yue. The report is based on the requirements of the assignment and contains sections for the programming problems as well as the individual contributions of each team member in this group.

In this report, we first described the provided dataset: Cityscapes Dataset, which contains street view images from over 50 cities in Europe with labeling data from 30 classes for each pixel, such as road, person, vehicle etc. Then, we implemented a baseline convolutional neural network (CNN) of five convolutional layers and five deconvolutional layers with perform the sematic segmentation task. We utilized softmax function for each pixel as the output layer and cross-entropy loss to be the objective function, and finished the data loader, performance indicator and the training function. The baseline model achieved XXX accuracy on this dataset after XXX epochs of training, later, we created a few more advanced CNN models with different network structure to achieve a better segmentation performance, these models were trained, tested and their performances were presented in this report visually. We tested these models on transformed images, imbalanced images and applied optimizations on the network structures to find the most suitable CNN networks for this assignment. One of our models with XXX achieved the best overall accuracy and has XXX in XXX as well as XXX in XXX. In the last section, individual contributions to this programming assignment was briefly described.

**1 Introduction**

Cityscapes Dataset is an urban visual understanding dataset based on over 50 cities in Europe. With over 20,000 images of urban traffic and corresponding pixel labelling, each label giving one of the 30 classes of possible objects like road and vehicle. With this dataset, we would like to create a convolutional neural network to perform the task of semantic segmentation, which requires the model to recognize the image in pixel level, giving labels to all pixels and hence tell all possible different objects in the image.

For this purpose, it is necessary to train convolutional neural networks (CNN) with Xavier weight initialization and batch normalization, CNNs are a certain type of neural networks commonly used for image processing and computer vision, typical structures of CNN usually contain convolutional layers, pooling layers and fully connected layers. Convolutional layers are usually used for incrementing receptive fields and extracting visual features, pooling layers could be either increasing sizes of receptive fields and strengthen the model invariance against translation and resizing, while fully connected networks would process the information from previous layers and perform classification task using softmax function.

The training process of a CNN model would require weights initialization and input normalization for better generalization, in our case, we adopted Xavier weight initialization and batch normalization. Xavier weight initialization is a specific method of weight initialization, this method ensures optimal initial weights so that vanishing gradients and saturated neurons could be avoided. The variance of each layer is expressed as:

This considers the input number and the output number so that the variances of each weight matrix are comparable, this could significantly speed up the training period. Batch normalization is an input preprocessing method, it normalizes the input in every batch with the following equation:

Where represents the batch mean, the standard deviation of the batch and a very small value in case of 0 batch standard deviation in denominator. Applying batch normalization to each batch, the normalized input could be transformed into data with zero mean and unit variance. With normalized data, the network weights no more depend on the range of the input data, and the model training process become less reliable on the initialization of the network weights, with the model itself more generalized and capable of predicting on equally preprocessed data.

**2 Related Work**

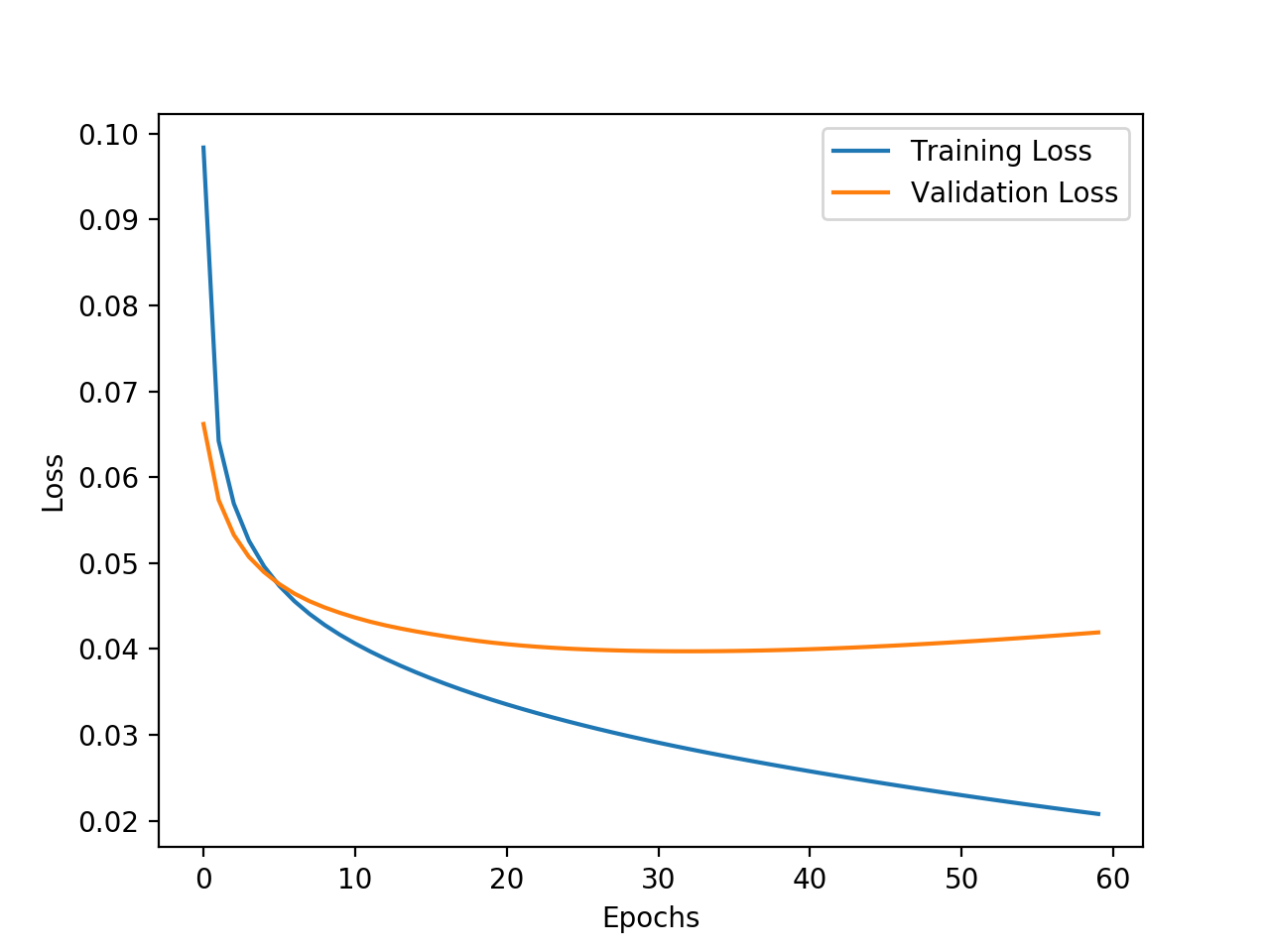
We implemented a neural network with three layers for this multiclass classification task. Specifically speaking, the first layer of the network is namely the input layer, with 784 inputs representing all pixel values in one image. The second layer is the hidden layer with 100 perceptron units, fully connected to the first layer, the third layer is the output layer with 10 units, with each of them outputting a value for the probability of each class. The second and the third layer is also fully connected, with the sigmoid function as the default activation function to project outputs of each layer into a probability space, i.e. (0,1).

**3 Methods**

After implementing and tuning the neural network for image recognition, we tried to add more useful features to the network, so that the deep learning model could converge faster or reach a higher accuracy value without overfitting. We first introduced an L2 penalty term into the loss function in order to constraint the weights, then used different activation functions and observed the changes of the model’s performance. Last but not least, we also experimented modified and different network structures to search for an ideal neural network model for this task.

**3.1 Baseline Method**

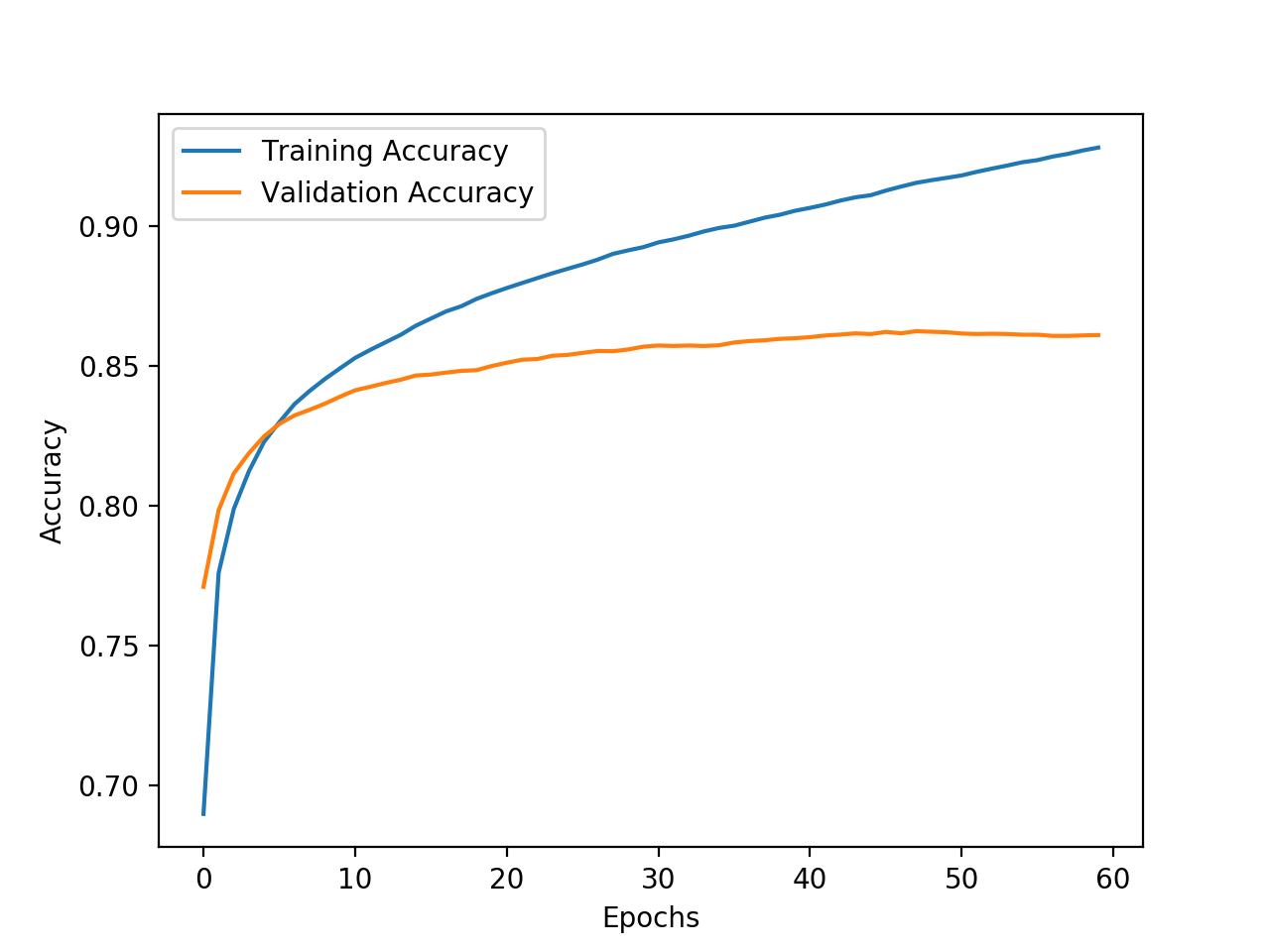
In the following charts, we tried out four different penalty terms, from 0.0001, 0.001 to 0.01 and 0.1. The results are visually represented below, here the left side represents lower penalty terms and the right side higher.

图片包含 屏幕截图

描述已自动生成图片包含 屏幕截图

描述已自动生成图片包含 屏幕截图

描述已自动生成

图片包含 屏幕截图

描述已自动生成图片包含 屏幕截图, 文字, 地图

描述已自动生成图片包含 屏幕截图

描述已自动生成

Figure 3: Loss and Accuracy Values of different Penalty Term in 60 Epochs

**3.2 Experimental Model 1**

So far, we have been using sigmoid function as our activation function in the neural network, sigmoid function is a very common activation function in the field of deep learning. However, there are other options which might work better in our case, such as the tanh and the ReLU (Rectified Linear Unit) activation function, three activation functions and their plots are posted below:

图片包含 屏幕截图

描述已自动生成图片包含 屏幕截图

描述已自动生成图片包含 屏幕截图

描述已自动生成

图片包含 屏幕截图

描述已自动生成图片包含 屏幕截图

描述已自动生成图片包含 文字, 地图, 屏幕截图

描述已自动生成

Figure 4: Performances of Activation Functions ReLU, Sigmoid, Tanh

**3.3 Experimental Model 2**

We have experimented with different regularization configuration and activation functions, now we explore the possibilities of the neural network itself and test the model performance with a modified structure. Originally, we have a three-layer neural network with one input layer, one hidden layer of 100 perceptron units and one output layer. In this section, we will first reduce or double the hidden units (200 in total) and test its performance. Then, another layer of hidden units would be added to the original model, however with same number of hidden units (50 hidden units each layer). The performances of the modified model would be tested and compared to the first model. After experimenting the neural network, we plotted the performances of each modified network below.

图片包含 屏幕截图

描述已自动生成图片包含 屏幕截图

描述已自动生成图片包含 屏幕截图

描述已自动生成

图片包含 屏幕截图

描述已自动生成图片包含 屏幕截图

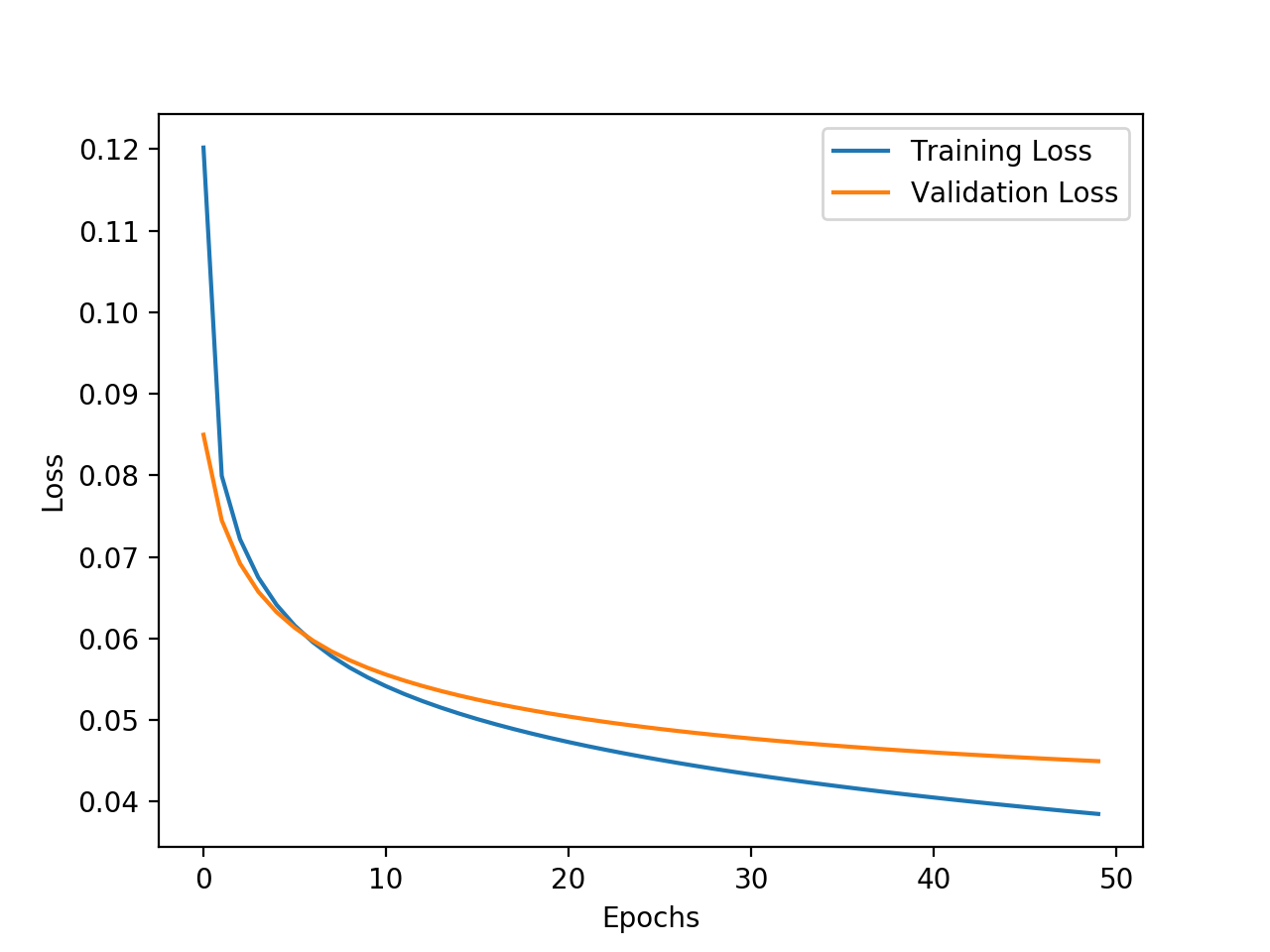
描述已自动生成图片包含 屏幕截图

描述已自动生成

Figure 5: Performances of Modified Neural Networks, Left: Original Neural Network, Middle: Network with half Hidden Units, Right: Network with doubled (200) Hidden Units

**3.4 UNet**

图片包含 屏幕截图

描述已自动生成

图片包含 屏幕截图

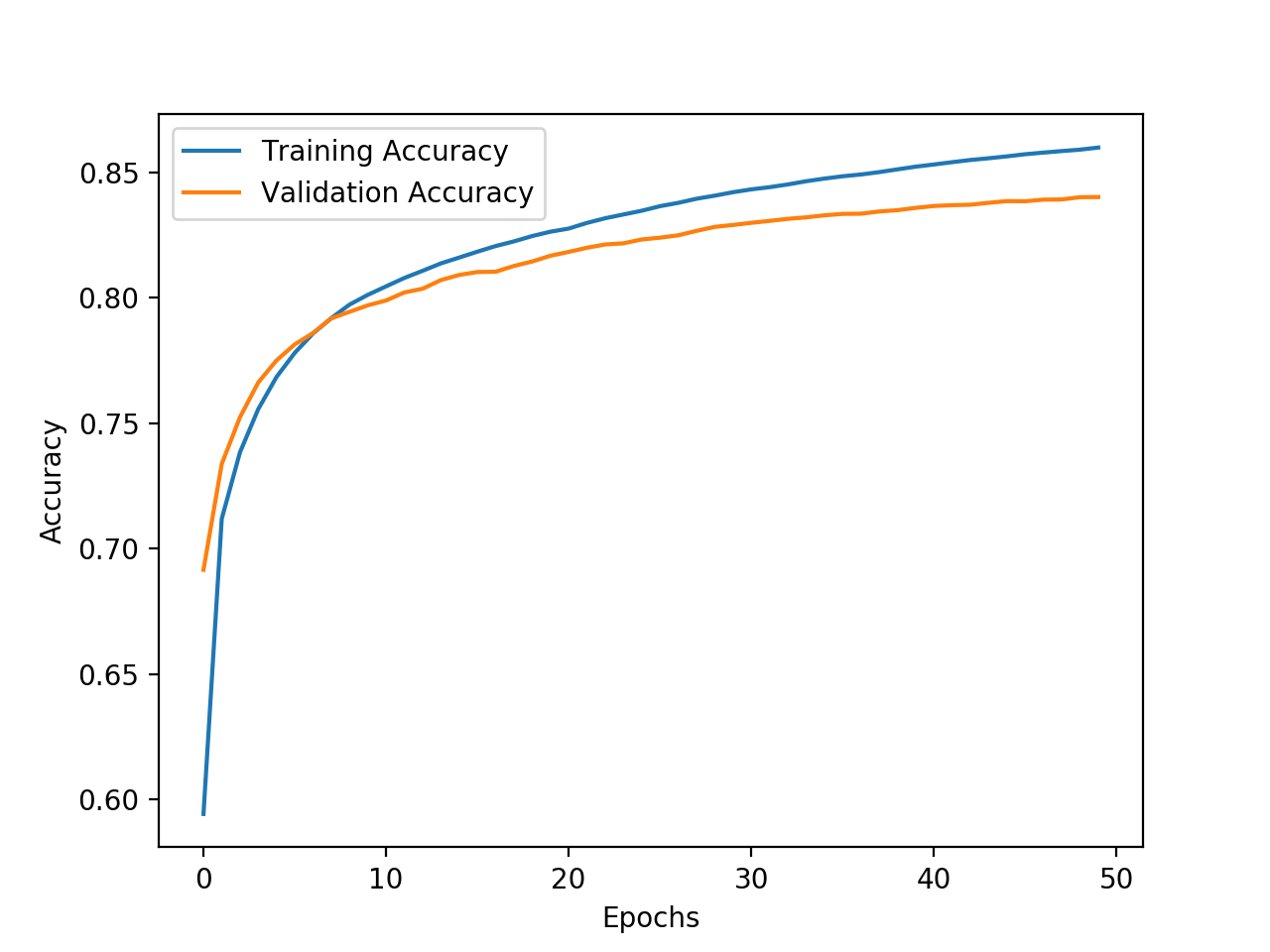
描述已自动生成

Figure 6: Performances of Modified Neural Networks, Left: Original Neural Network, Right: With 2 hidden Layers

Next, we tried out different layer numbers and their influences on the model’s performance. To validate the comparison between different models, we let the total number of weights and biases to be the same, so that the total model flexibility is comparable. On the left side of the charts, we have the original model whereas the right side is the model with the doubled hidden layer.

**4 Results**

This part included the individual contributions of the two team members Huimin Zeng and Zhenrui Yue in the programming assignment.

**4.1 Huimin Zeng**

I implemented the entire neural network framework with the layer and activation components as well as the forward and backward steps, I also implemented the training function with momentum and mini-batch stochastic gradient descent. Then I tested the neural network and eliminated some problems and bugs of this programming assignment, I also wrote some part of this report.

**4.2 Zhenrui Yue**

I individually implemented some parts of the neural network including the activation functions, softmax function and cross-entropy loss, moreover, I also wrote the forward and backward functions and was responsible for debugging and running tests to debug and optimize the neural network and its performances. I implemented some tests to generate tables and charts and wrote the most parts of this report.

**5 Individual Contributions**

This part included the individual contributions of the two team members Hao Xiang, Huimin Zeng, Xingyi Yang and Zhenrui Yue in the programming assignment.

**5.1 Hao Xiang**

XXXX

**5.2 Huimin Zeng**

XXXX

**5.3 Xingyi Yang**

XXXX

**5.4 Zhenrui Yue**

XXXX