

## **Final Project Report**

### **• Your group-ID and group members (name and uID)**

Chao Lu – u6686416  
Haoran Song – u6688461  
Rui Wang – u6688523  
Yousong Zeng – u6660070

### **• Title**

Traffic Sign Recognition

### **• Introduction**

Problem introduction:

The problem is about detecting and recognizing Croatian traffic signs. After obtaining the traffic sign images, we need to predict their shape and sign labels.

Problem significance:

This problem is related to automatic driving. The automatic driving system needs to understand the meaning of the traffic signs which are detected by sensors, then takes appropriate actions.

### **• Related work**

Ciresan and his group tried to use DNN to deal with the task and achieved the accuracy of 99.46% (Cireşan, D., 2012). Salti and his group used solid image analysis and pattern recognition techniques to detect dynamic traffic images, one difference from the traditional method is that they detected the area of interest, not just sliding windows, which makes their model robust to appearance change (Salti S., 2015). Kaplan and his group proposed an innovative approach, they combined color-based and shape-based methods, used circular traffic sign detection and recognition on color images (Kaplan Berkaya, S., 2016).

### **• Your approach: Conceptual design, method, algorithms, equations and figures.**

Basically, we divide our work into two parts.

First, we need to localize and extract the traffic sign images from these original images. We use the coordinates given by the dataset to implement this step. The dataset also gives information about which images should be treated as training data, which should be used as testing data. According to this, we split the dataset into training dataset and testing dataset. Based on the dataset again, we should predict the shape and sign labels of the traffic sign images. Now we have already had all the data. Then we tried to perform some optimization methods on the datasets to achieve a higher accuracy. Finally, we obtained processed datasets.

Next, we need to decide which algorithm to use. Basically, we discussed and tried KNN, CNN, SVM, of the three algorithms, CNN performs best and so we choose CNN. We used 3 convolutional layers, 3 pooling layers and 2 fully connected layers as our neural network. In the convolution and pooling layers, we chose Relu as our activation function and after each layer we dropout some of the neurons to avoid ourfitting. In addition, in the fully connected layer, we choose softmax as our activation function. We calculate the accuracy and training loss by calculating the average of the cross entropy. In particular, due to the variety of traffic signs and the insufficient number of data sets, we have optimized the neural network for training traffic signs.

#### • Experiment results & Conclusion

##### SVM & KNN model

We first try to solve this problem with the K-Nearest Neighbors (KNN) and Support Vector Machine (SVM) algorithms. Linear SVM shows the best results with a shape accuracy of 0.865800865800865 and a sign accuracy of 0.684523809523809. Table 1 shows accuracy of shape and sign for each algorithm as well as different parameters. We randomly select 20 pictures from the testing dataset to match the predicted results of shape and sign respectively. The results are shown in Fig.1 and Fig.2.

Algorithm	C	Kernel	Gamma	Shape Accuracy (test)	Sign Accuracy (test)
SVM	0.1	rbf	0.1	0.343614719	0.070887446
SVM	0.8	rbf	10	0.343614719	0.070887446
SVM	1	rbf	10	0.343614719	0.070887446
SVM	0.1	linear		0.865800866	0.68452381
SVM	1	linear		0.865800866	0.68452381
SVM	0.1	poly	1	0.834415584	0.601731602
SVM	0.8	poly	10	0.834415584	0.601731602
SVM	1	sig-moid	1	0.343614719	0.070887446
SVM	10	sig-moid	1	0.343614719	0.070887446
KNN(k=5)				0.840909091	0.475649351

Table 1 - Croatian traffic signs test set (1,708 images) predicted accuracy results.

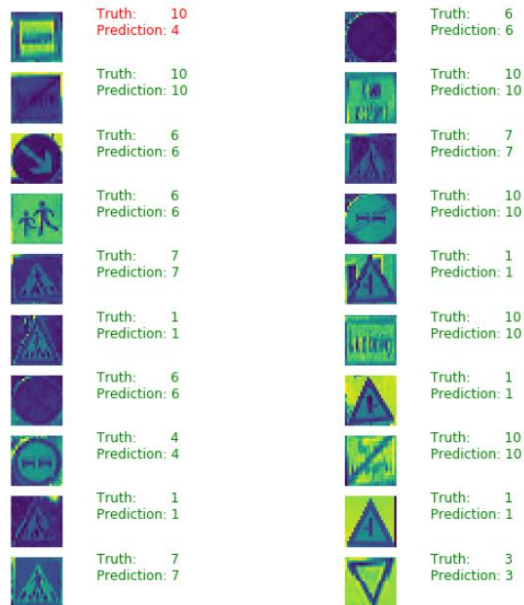


Fig.1 – Result of shape accuracy

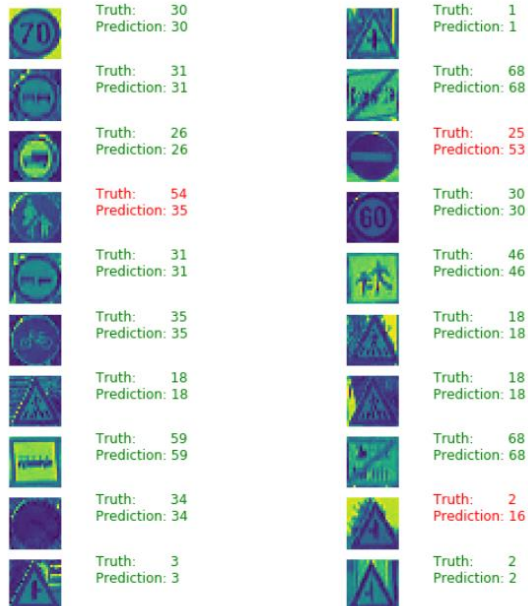


Fig.2 – Result of sign accuracy

Model (Convolutional Neural Network):

#### 1. Build the model

Convolutional Neural Network(CNN) technology is a major advance in image recognition technology. CNN made amazing progress in the ImageNet competition in 2012, increasing accuracy by 10%, this makes CNN technology pay more attention to it and has been applied in many fields. Meanwhile, CNN technology is appropriate for handling our current road sign recognition project, so we decided to use CNN to train our model. By reviewing the data, we found that CNN's main advantages are: Compared to traditional machine learning methods (svm, knn), CNN automatically extracts image features through convolution and pooling, and reduces the dimensions of training data, which is dealing with a large number of high latitudes. Data has a decisive advantage.

#### 2. For the shape of traffic sign

Since our project needs to separately identify the shape and sign of the road sign, we decided to build two convolutional neural networks to classify the shape and sign respectively. First we classify the shape. We reshape the image in the dataset into a 24\*24 format, and then one hot encoding the label in the dataset to make it binary, so we complete the preprocessing of the input data. Next is the build model. We choose tensorflow as our development environment to customize our neural network.

Through debugging, our model has two layers of convolution and pooling layers and one fully connected layer structure, with the maximum pooling and gradient down optimization, the learning rate is  $10^{-4}$ , and the training is performed step by step. After 2500 steps, the accuracy of the training set reaches 96%. The test set accuracy reached 92.3%. We randomly selected 20 images from the actual measurement to test and get the following results:



Fig.3 (shape)result

It can be seen that the model pair is exactly the correct prediction of the image. This is a good result. It can be seen that our model performs very well when identifying the shape of the road sign.

### 3. For the sign of traffic sign

Next we try to classify the sign. There are a total of 70 categories of symbols in the dataset, and some categories in the test set do not exist in the training set, which has some impact on the accuracy of our classification. We tried to train the sign with the model of the training shape, but the final accuracy was not Satisfactory, only 35%.

Therefore, we have to optimize the original model. We tried to increase the number of convolutional layers and tested them in batches with 4 convolutional layers. It was found that over-fitting occurred when the training batch was increased. So we finally set 3 convolution layers, 3 pooling layers and one fully connected layer. After determining the structure of the model, we began to optimize the model.

#### 4. Optimization

##### (1) Batch\_size

In the initial training, I chose batch\_size = 16 for training. When I found that 100 training steps, the loss of the training set has dropped to nearly 0, and the accuracy is only about 65%. So I decided to increase the batch\_size and tested it at batch\_size = 64,128,256. The test results are as follows:



Fig.4 batch\_size=64

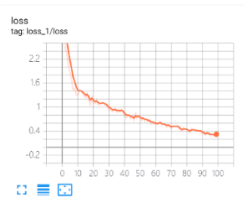


Fig.5 batch\_size=128

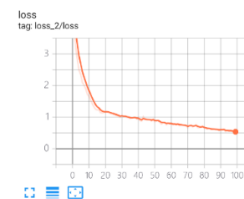


Fig.6 batch\_size = 256

I found that the larger the batch\_size, the faster the loss value drops, and the more severe the overfitting, so we adjusted the overfitting in next steps. And after processing, make a final selection of batch\_size.

##### (2) Model

We selected 700 images from the test data as the validation set, and tested the performance of the neural network in the validation dataset during the training process. We found that after training for a period of time, the accuracy of the training set continued to rise, while the accuracy of the validation data set has been around 50%, so we suspect that the neural network has been over-fitting, then the neural network has been adjusted accordingly for the phenomenon of overfitting.

- Dropout: I made a 50% dropout after each pooling layer, and 80% of the dropout before the activation function “softmax” which was passed through the fully

connected layer. The overfitting situation was alleviated. It was shown below:

Training set loss map when running 50 epochs:

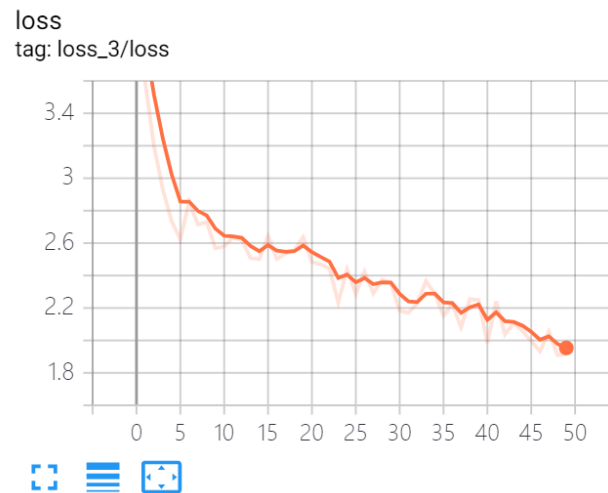


Fig.7 50 epochs training loss

We can see that the loss dropping is not as fast as before, indicating that there is still a lot of room for improvement in the subsequent training process, and when running 50 epochs, the accuracy of the training set and the validation set before and after the dropout is compared below: (Although the convergence speed is fast before dropout, the over-fitting phenomenon is also very serious.)

Epoch 47, Train accuracy 0.701155	Epoch 47, Train accuracy 0.553046
Epoch 47, Validation accuracy 0.56	Epoch 47, Validation accuracy 0.501429
Epoch 48, Train accuracy 0.70063	Epoch 48, Train accuracy 0.549895
Epoch 48, Validation accuracy 0.567143	Epoch 48, Validation accuracy 0.474286
Epoch 49, Train accuracy 0.705357	Epoch 49, Train accuracy 0.559349
Epoch 49, Validation accuracy 0.551429	Epoch 49, Validation accuracy 0.498571

Fig.8 acc before dropout

Fig.9 acc after dropout

We can find that the over-fitting phenomenon has been solved to some extent.

- Batch Normalization(BN): Firstly, Find the mean of each training batch data, and then find the variance, scale transformation and offset of each training batch data. For most activation functions, the gradient in this region is either the largest or gradient (such as the ReLU we are using), so BN can be seen as an effective means of combating gradient disappearance. In addition, BN is an effective way

to avoid over-fitting of models and ensure generalization ability through explicit control model complexity in machine learning. Codes related to BN are shown below:

```
fc_mean, fc_var = tf.nn.moments(h_conv2, axes = [0])
scale = tf.Variable(tf.ones([f1]))
shift = tf.Variable(tf.zeros([f1]))
h_bn_conv2 = tf.nn.batch_normalization(h_conv2, fc_mean, fc_var, shift, scale, 0.001)
```

Fig.10 codes for BN

Similarly, the overfitting phenomenon is alleviated.

- (3) Learning rate: In the initial training, we chose 1e-3 as the initial learning rate. As the training progresses, the training set loss drops directly to 0. We infer that there is a gradient explosion, so we adjusted the learning rate for the epoch that started the gradient explosion. I made adjustments to the learning rate after 200 epochs. The adjustment is as follows:

```
lr = tf.cond(tf.less(global_step, 200),
              lambda: tf.constant(0.001),
              lambda: tf.constant(0.0001))
```

Fig.11 change learning rate

The situation of the gradient explosion has been solved to some extent.

- (4) Deepen the network layer

We hope to increase the accuracy of the training set by increasing the number of neural network layers. As the number of layers of the neural network increases, the accuracy of training continues to rise. However, due to the limitations of the training data, the accuracy of the validation set does not rise. Even in the case of a decline, there is a phenomenon of overfitting. Therefore, after several tests, we selected a three-layer convolutional layer, a three-layer pooling layer, and a two-layer fully connected layer



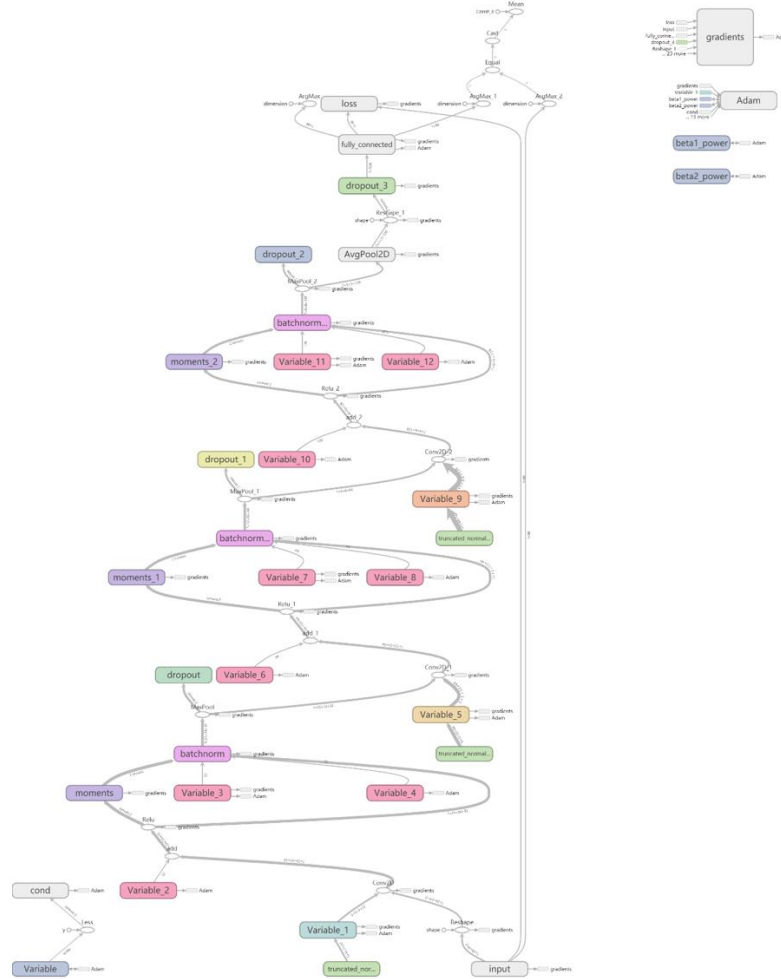
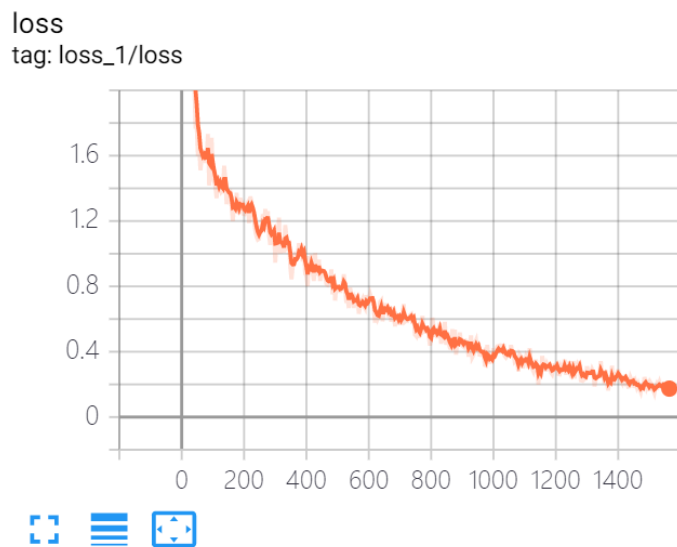


Fig.12 structure of CNN

## 5. Results:

The training loss after running 1600 epochs.



The accuracy of the training set and verification set is shown in the following figure:

Epoch 1599, Train accuracy 0.954874  
Epoch 1599, Validation accuracy 0.817028

Fig.14 last epoch training and validation accuracy

The test set accuracy rate is shown in the following figure:

train accuracy: 0.836491

Fig.15 Train accuracy

We randomly selected 20 images from the test set to match, and the results are shown below: (Green is correct, red is wrong)
















	Truth: 80 Prediction: 80		Truth: 35 Prediction: 35
	Truth: 69 Prediction: 67		Truth: 59 Prediction: 59
	Truth: 30 Prediction: 30		Truth: 80 Prediction: 80
	Truth: 36 Prediction: 36		Truth: 35 Prediction: 35
	Truth: 83 Prediction: 83		Truth: 50 Prediction: 50
	Truth: 80 Prediction: 80		Truth: 44 Prediction: 44
	Truth: 30 Prediction: 30		Truth: 80 Prediction: 80
	Truth: 48 Prediction: 48		Truth: 67 Prediction: 76
	Truth: 30 Prediction: 30		Truth: 8 Prediction: 8
	Truth: 6 Prediction: 6		Truth: 58 Prediction: 58

Fig.16 random test

In summary, the neural network we built can detect 95% of the shape of the traffic sign and 83% of the meaning of the traffic sign.

Potential future work: use RCNN to detect traffic signs and recognize it.

### • **Summary of Learning Outcome**

From this task, we learned quite a lot. Firstly, we learned how to do image task in python environment, which is difference from the previous tasks which are done in Matlab code. Then we learned how to build a convolutional neural network and how to adjust its various parameters to achieve an optimal accuracy for the specific task. In addition, we tried to run our model using both GPU and CPU, but due to the limitation of our devices, we used CPU finally to make sure that every member is able to run the program successfully.

### • **Peer-Assessment**

1. Rui Wang tried different algorithms including SVM, KNN, he helped us to find the best model.
2. Chao Lu did the pre-processing of dataset and we discuss the optimization of CNN
3. Yousong Zeng built the basic model of CNN and train the shape of train sign
4. I built the CNN for traffic sign detection and did the optimization of CNN especially for the sign information.

## Reference List

- Cireřan, D., Meier, U., Masci, J., & Schmidhuber, J. (2012). Multi-column deep neural network for traffic sign classification. *Neural Networks*, 32, 333-338. doi:10.1016/j.neunet.2012.02.023
- Salti, S., Petrelli, A., Tombari, F., Fioraio, N., & Di Stefano, L. (2015). Traffic sign detection via interest region extraction. *Pattern Recognition*, 48(4), 1039-1049. doi:10.1016/j.patcog.2014.05.017
- Kaplan Berkaya, S., Gunduz, H., Ozsen, O., Akinlar, C., & Gunal, S. (2016). On circular traffic sign detection and recognition. *Expert Systems with Applications*, 48, 67-75. doi:10.1016/j.eswa.2015.11.018