Project Report

1. Requirement

The target of this project is to train a Machine Learning model to recognize the traffic signs in Germany.

2. Data Loading

The training data, validation data, and test data have already been provided separately by project team, so the first thing to do is loaded the '.p' files (pickled data) into array of [, 32, 32, 3] – which means 32 x 32 image with 3 channels (RGB).

3. Dataset Summary & Exploration

In this section, we summarize the dataset by showing the number of examples and the image shape. We also display a list of image samples for comparison purpose (at least when we search online to find other samples we can have idea how the German traffic signs look like).

4. Design and Test a Model Architecture

In this section, we build a Deep Learning Neural Network using LeNet provided by the project team. To improve the accuracy, we fine-tuned the Batch Size and introduce the Dropout. The keep_prob is set to 0.5 in training, while set to 1 in validation and test.

The LeNet is a standard one, which has 2 convolution layers, 2 pooling layers, and 3 fully connection layers. See the table below:

Layer	**Type**	**Input**	**Output**
1	Convolutional	32x32x1	28x28x6
2	Pooling	28x28x6	14x14x6
3	Convolutional	14x14x6	10x10x16
4	Pooling	10x10x16	5x5x16
6	Fully Connected	400 (flattened)	120
7	Fully Connected	120	84
8	Fully Connected	84	43

Relu() is used as the activation functions for the Convolutional layers. And AdamOptimizer is used as the optimisation function.

As the initial LeNet can't guarantee 0.93 accuracy on Test Set, I tried two methods trying to improve the accuracy: Regularisation and Dropout. And found Dropout has a better result. After introducing Dropbox, the accuracy of Test Set can keep above 0.93 (sometimes reach 0.95) after 200 epoch training.

5. Test Model on New Images

I used Google to search 'Germany Traffic Sign' images. Picked a few and ran tests. It is found that there are a few criteria to follow when choosing pictures:

1) The traffic sign cannot be too small compared to the whole picture. If the traffic sign stays in a corner (in this case there must be huge background noise), the algorithm would have very good chance to mis-categorize.
2) The picture should not be too large. I guess it might cause ambiguity when the program to shrink the picture.
3) The background needs to be clear. It doesn't matter if the background is vague, or foggy, or even has colour. The key point is it shouldn't have a mixture of different colour. I think it may 'distract' the algorithm

I picked 6 pictures complying to the criteria. The result is 5/6. The algorithm is able to recognize 5 of the 6 pictures successfully.