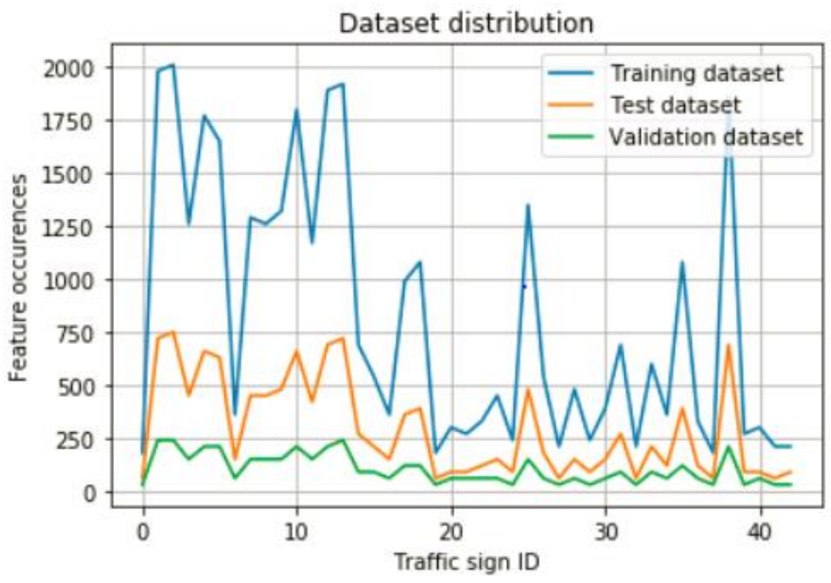
Traffic sign classifier

Dataset summary:

The number of data points available for each traffic sign are split into training, validation and test sets uniformly as shown below



However, the data needs to be shuffled to prevent over fitting the CNN model. The data also needs to be normalized to ensure uniform correction on all features during backpropagation. This will prevent the model from running too slow.

The images are converted into grayscale images to enable detection of edges better and reduce computational intensity.

Augmentation could further improve the accuracy but is very computationally intensive so I refrained from using it in this model since I got the required accuracies without it.

Model architecture:

I used the architecture from LeNet lab as the model was also in grayscale and got an accuracy of about 92% and by making minor modifications, was able to achieve an accuracy of 94%. The layers are setup as below:

1. 5x5 convolution (32x32x1 → 28x28x6)
2. ReLU

3. 2x2 average pooling (28x28x6 → 14x14x6)
4. 5x5 convolution (14x14x6 → 10x10x16)
5. ReLU
6. 2x2 average pooling (10x10x16 → 5x5x16)
7. Flatten layers (5x5x20 → 500)
8. Fully connected layer (500 → 120)
9. ReLU
10. Dropout layer
11. Fully connected layer (120 → 100)
12. ReLU
13. Dropout layer
14. Fully connected layer (100 → 43)

Model training:

Adam optimizer from the LeNet lab was used in this case with the below parameters

Batch size = 200

Epochs = 40

Mu = 0

Sigma = 0.1

Learning rate = 0.001

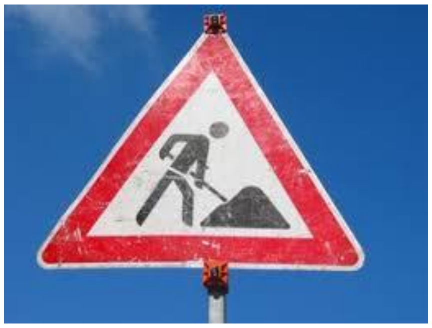
Dropout keep probability = 0.6

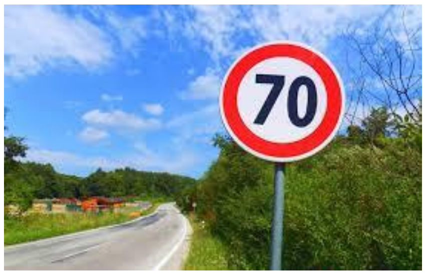
Solution approach:

I followed the LeNet example predominantly. I had the option to pre-process the data to achieve the required accuracy or modify the model. I choose the later due to reduced computational complexity and hence the time taken to run the model. The small modifications involved in adding extra layers both convolution and activation layers. I understood how the layers affect the validation accuracy by adding and removing different layers.

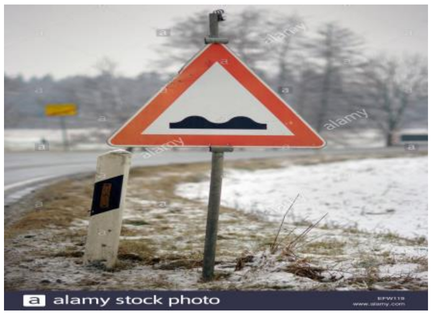
New images:

I picked the following German traffic signs from the internet. The difficulty with each image is described below:

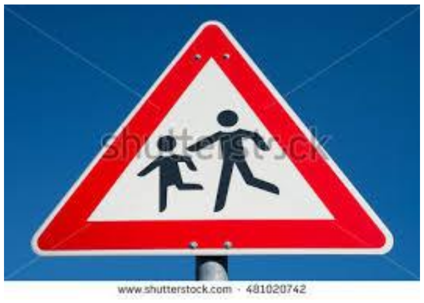

This image is fairly straight forward but I was curious if the model would confuse it with children crossing. Surprisingly was classified correctly.



This image was chosen to verify how the classifier would behave when the image has other components and the traffic sign is not centered in the frame.



The brightness of the traffic sign blends in with the background in this image and the extension of the pole would make the classification hard and as expected it was classified as a traffic signal.



This is chosen to compare with the road work image. Surprisingly the road work image was classified correctly and this image was misclassified because the extracted shapes look like 2 parallel lines.

This is a fairly simpler image to classify and was used to check the base accuracy of the CNN model (Also consolidation prize after misclassifying some images)

Performance on the new image set:

Though the accuracy on the test set was 91.8%, the performance on the new images was 50%. This could be eliminated by processing the data further using rotate, augment, shift, blur etc.

Model certainty:

From the below image, it can be said that there is lot of scope for improvement in the model certainty. Interesting observations were made and also confirmed that image processing is an important step for the classifier.

```
Image 1
Image Accuracy = 1.000

Image 2
Image Accuracy = 0.500

Image 3
Image Accuracy = 0.333

Image 4
Image Accuracy = 0.250

Image 5
Image Accuracy = 0.400
```