CSE616 Neural Networks and Their Applications Project

Shahira Hany Hussein Mohamed Amin (2101341)

May 2022

In this project, I implemented classification of traffic signs using convolutional neural networks (CNNs). I used three different CNN architectures proposed by Shustanov and Yakimov in [1] for traffic sign classification. I used Jupyter Notebook and TensorFlow library to train and test the three networks using the German Traffic Sign Recognition Benchmark (GTSRB) dataset [2].

1 The GTSRB Dataset

I used the GTSRB dataset for training and testing the network. The dataset consists of 43 classes as shown in Fig. 1: 1) Speed limit (20km/h), 2) Speed limit (30km/h), 3) Speed limit (50km/h), 4) Speed limit (60km/h), 5) Speed limit (70km/h), 6) Speed limit (80km/h), 7) End of speed limit (80km/h), 8) Speed limit (100km/h), 9) Speed limit (120km/h), 10) No passing, 11) No passing veh over 3.5 tons, 12) Right-of-way at intersection, 13) Priority road, 14) Yield, 15) Stop, 16) No vehicles, 17) Veh $>$ 3.5 tons prohibited, 18) No entry, 19) General caution, 20) Dangerous curve left, 21) Dangerous curve right, 22) Double curve, 23) Bumpy road, 24) Slippery road, 25) Road narrows on the right, 26) Road work, 27) Traffic signals, 28) Pedestrians, 29) Children crossing, 30) Bicycles crossing, 31) Beware of ice/snow, 32) Wild animals crossing, 33) End speed + passing limits, 34) Turn right ahead, 35) Turn left ahead, 36) Ahead only, 37) Go straight or right, 38) Go straight or left, 39) Keep right, 40) Keep left, 41) Roundabout mandatory, 42) End of no passing, and 43) End no passing veh $>$ 3.5 tons

The dataset is divided into training set and test set. The training set consists of a total of 39202 images, where the number of images in each class is shown in the bar graph in Fig. 2. The test set consists of a total of 12630 images. Fig. 3 shows 25 random images from the test set, where for each image, the x label is the width of the image, and the y label is the height of the image.

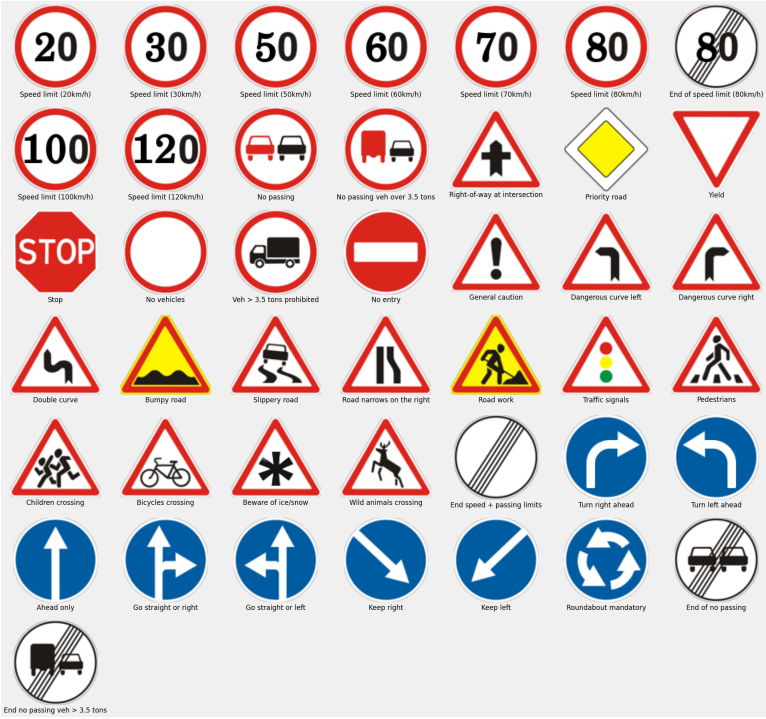

Figure 1: The 43 Classes in the GTSRB Dataset

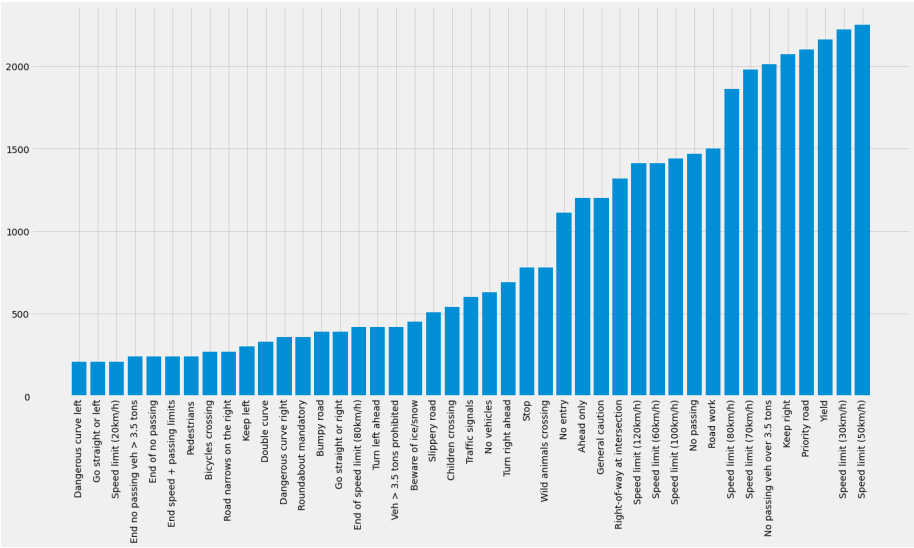

Figure 2: Training Data Statistics (Number of Images in Each Class)

Figure 3: 25 Random Images From the Test Set

2 Data Processing

I processed the data as follows:

- I downloaded the training data from: `https://www.kaggle.com/datasets/meowmeowmeowmeowmeow/gtsrb-german-traffic-sign`.
- I loaded the training data and resized all images to a fixed width and height. I used width= 30 and height= 30 for the second and third CNN architectures, and I used width= 90 and height= 90 for the first CNN architecture. Also, each image consists of three channels: R, G, and B.
- I (randomly) shuffled the training data.
- I split the training data into training set and validation set with a ratio of 70:30 (i.e. the training set consists of 27446 images, and the validation set consists of 11763 images).
- I performed one hot encoding of the training labels and validation labels by mapping each class to a binary vector that consists of all zero values except for the index of the class, which is set to 1.

3 The three CNN Architectures

3.1 First CNN Architecture

The first CNN consists of the following layers:

- Convolutional layer with a stride of 2 and 4 kernels of size 7×7
- Convolutional layer with a stride of 2 and 8 kernels of size 5×5
- Convolutional layer with a stride of 2 and 16 kernels of size 3×3
- Convolutional layer with a stride of 2 and 32 kernels of size 3×3
- Convolutional layer with a stride of 1 and 16 kernels of size 2×2
- Convolutional layer with a stride of 1 and 8 kernels of size 2×2
- Convolutional layer with a stride of 1 and 4 kernels of size 2×2
- Fully connected layer of size 64 with exponential linear unit activation
- Fully connected layer of size 16 with exponential linear unit activation
- Fully connected layer of size 43 with softmax activation

3.2 Second CNN Architecture

The second CNN consists of the following layers:

- Convolutional layer with a stride of 2 and 16 kernels of size 3×3
- Fully connected layer of size 512 with exponential linear unit activation
- Fully connected layer of size 43 with softmax activation

3.3 Third CNN Architecture

The third CNN consists of the following layers:

- Convolutional layer with a stride of 2 and 16 kernels of size 3×3
- Convolutional layer with a stride of 2 and 32 kernels of size 3×3
- Convolutional layer with a stride of 2 and 64 kernels of size 3×3
- Fully connected layer of size 512 with exponential linear unit activation
- Fully connected layer of size 43 with softmax activation

4 Training the Network: Data Augmentation, Loss Function, and Hyper-Parameters

To train the network, I used TensorFlow.

- I used ImageDataGenerator in TensorFlow to perform data augmentation with: rotation_range = 10, zoom_range = 0.15, width_shift_range = 0.1, height_shift_range = 0.1, shear_range = 0.15, horizontal_flip = False, vertical_flip = False, and fill_mode="nearest".
- I used categorical cross-entropy loss function.
- I used "accuracy" as the metric.
- I used the following hyper-parameters:
  - Learning rate = 0.001
  - Number of epochs = 30
  - Optimizer: adam with a learning rate decay of $\frac{\text{learning rate}}{\text{Number of epochs} \times 0.5}$
  - Batch size = 50

Figures 4, 5, and 6 show the training loss and validation loss (right) and training accuracy and validation accuracy (left) for the first, second, and third CNNs, respectively. We can see that the training loss and validation loss decrease with increasing the number of training epochs while training accuracy and validation accuracy increase as the number of epochs increases. It is also clear in Figures 4, 5, and 6 that the third CNN reached the lowest training loss and validation loss, followed by the second CNN, and then the first CNN, which has the highest training loss and validation loss. Moreover, the third CNN reached the highest training accuracy and validation accuracy, followed by the second CNN, and then the first CNN, which has the lowest training accuracy and validation accuracy.

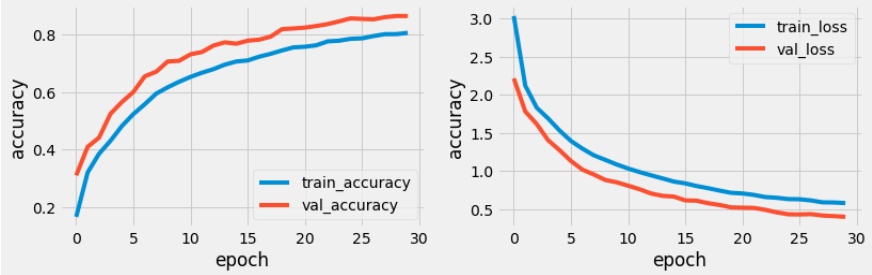

Figure 4: Training Loss and Validation Loss (Right) and Training Accuracy and Validation Accuracy (Left) for the First CNN

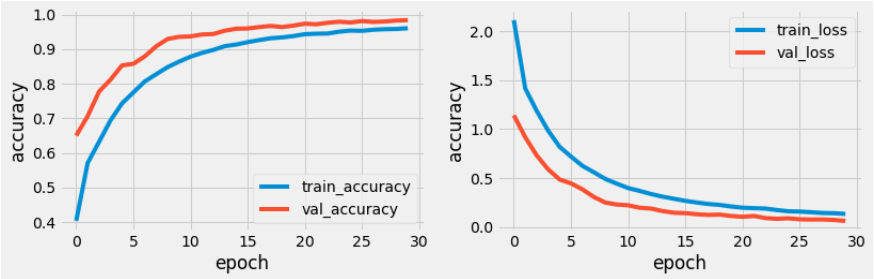

Figure 5: Training Loss and Validation Loss (Right) and Training Accuracy and Validation Accuracy (Left) for the Second CNN

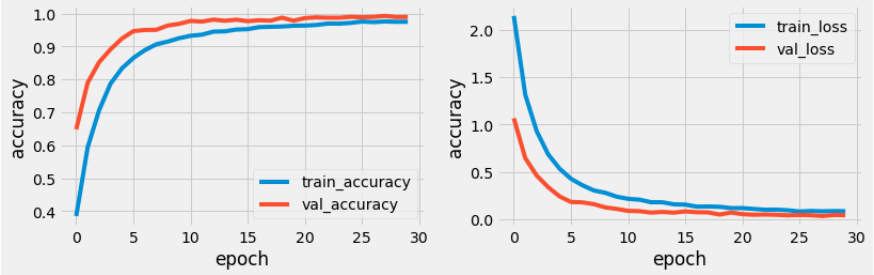

Figure 6: Training Loss and Validation Loss (Right) and Training Accuracy and Validation Accuracy (Left) for the Third CNN

Table 1 summarizes the training loss, validation loss, training accuracy, and validation accuracy that each of the three CNNs reached at the end of training. Table 1 also shows the total number of parameters and the training time per parameter for each of the CNNs. It is clear from Table 1 that the first CNN has the longest training time followed by the third CNN, and then the second CNN, which has the shortest training time.

	Total Parameters	Train Time/Parameter	Train Acc.	Val. Acc.	Train Loss	Val. Loss
First CNN	12,015	0.1326	0.8058	0.8644	0.5772	0.3965
Second CNN	1,628,651	0.000245	0.9605	0.9839	0.1351	0.0639
Third CNN	177,227	0.00198	0.9751	0.9897	0.0824	0.0402

Table 1: Training Results for the Three CNNs

5 Testing the Network

Figures 7, 8, and 9 show 25 prediction images for the first 25 images in the test set made by the first, second, and third CNN, respectively, where for each prediction, the x label indicates the actual class number and predicted class number. A correct prediction is labeled in green color whereas a wrong prediction is labeled in red color. It is clear from Figures 7, 8, and 9 that the first CNN has the greatest number of wrong predictions, followed by the second CNN, and then the third CNN which has the smallest number of wrong predictions.

Figure 7: Predictions on the First 25 Test Images Made by the First CNN

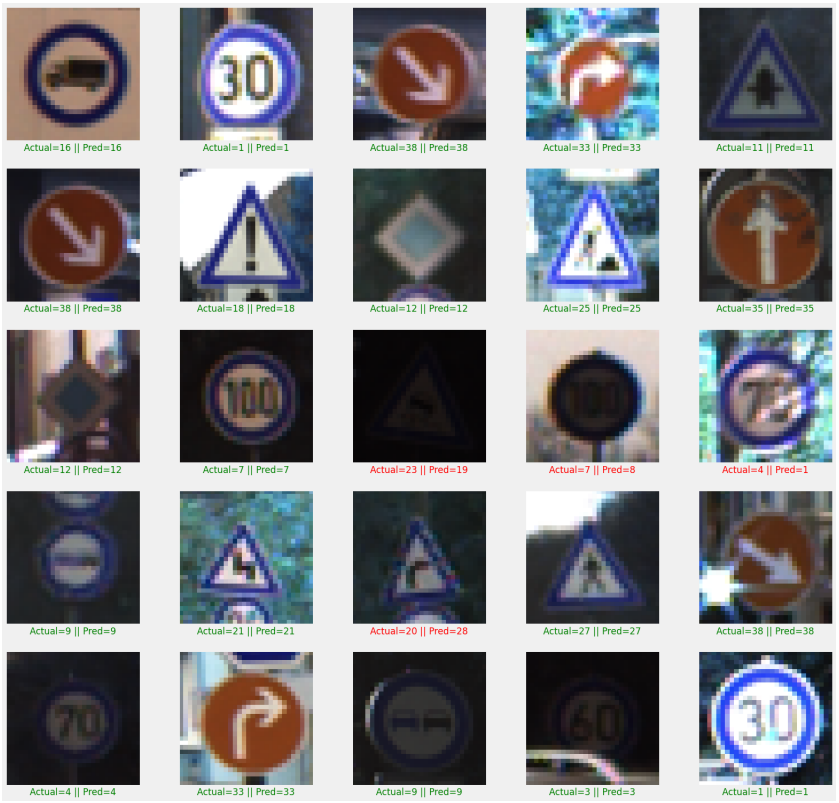

Figure 8: Predictions on the First 25 Test Images Made by the Second CNN

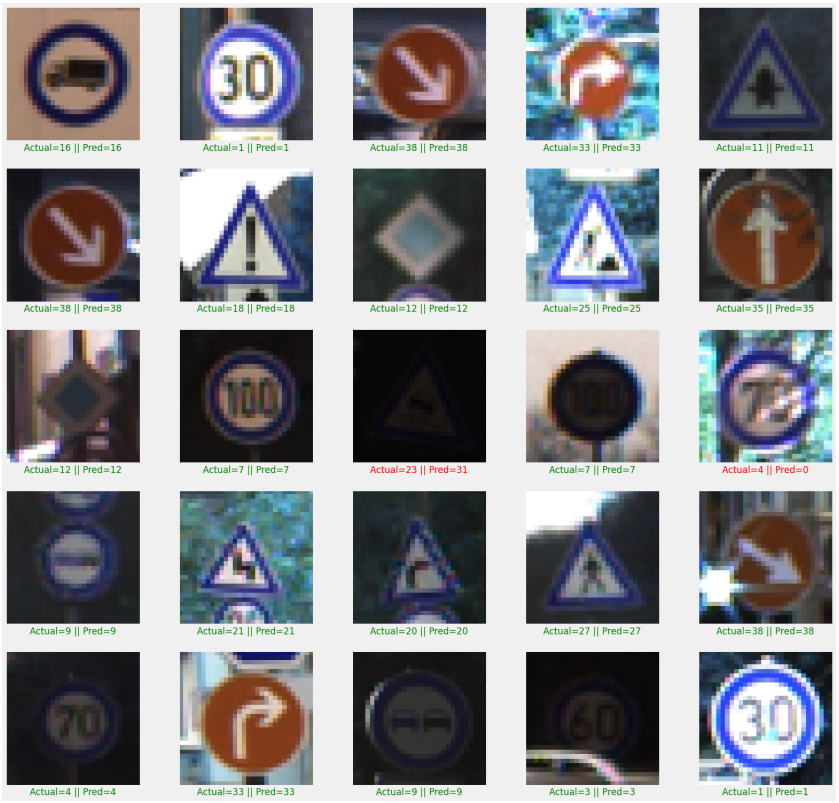

Figure 9: Predictions on the First 25 Test Images Made by the Third CNN

Table 2 lists the test accuracy achieved by the first, second , and third CNN, where we can see that third CNN has the highest test accuracy, followed by the second CNN, and then the first CNN, which has the worst test accuracy.

	Test Accuracy
First CNN	81.9160
Second CNN	82.2802
Third CNN	88.2343

Table 2: Test Accuracy for the Three CNNs

Figures 10, 11, and 12 show the confusion matrix for the first, second, and third CNNs, respectively. From Figures 10, 11, and 12, we can see that the third CNN has the greatest diagonal elements and smallest off-diagonal elements, which means that it makes the least number of wrong predictions.

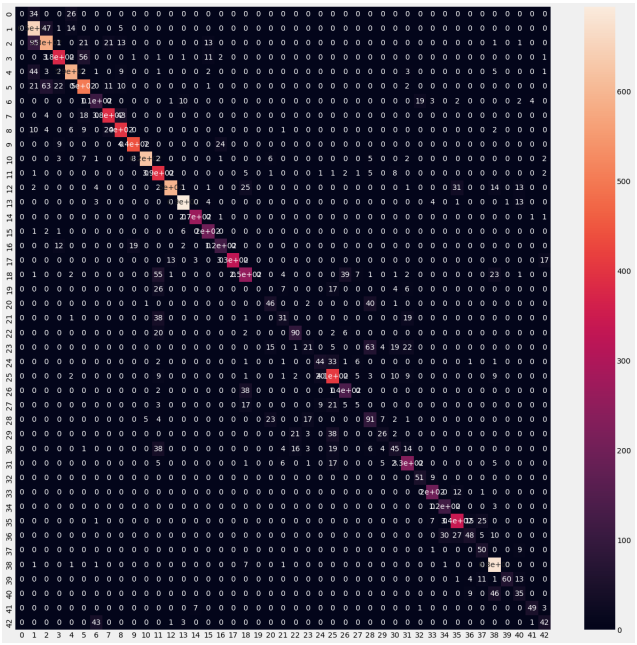

Figure 10: Confusion Matrix for the First CNN

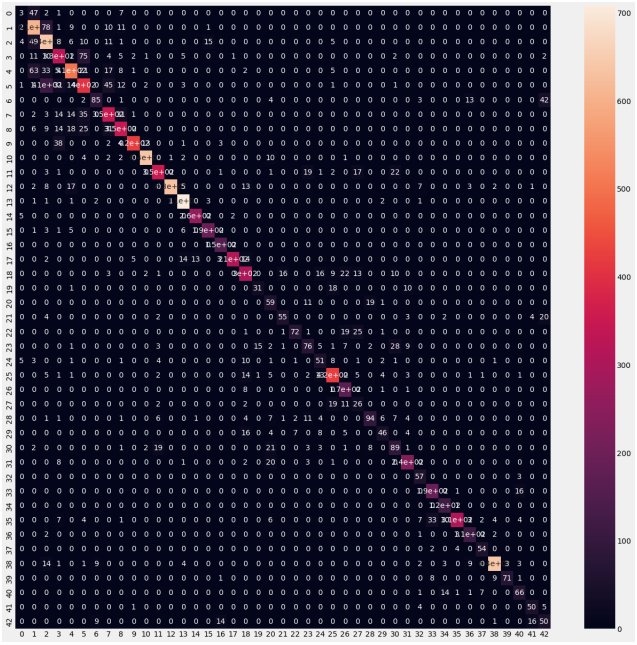

Figure 11: Confusion Matrix for the Second CNN

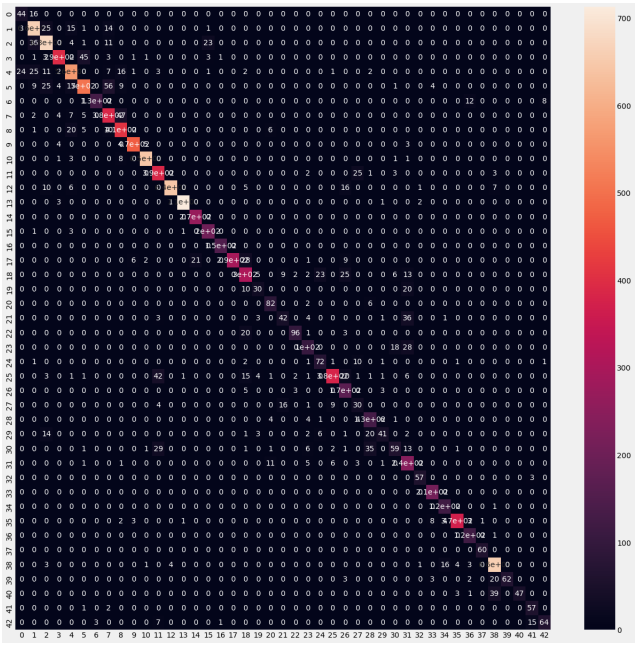

Figure 12: Confusion Matrix for the Third CNN

6 Conclusion

I used three different convolutional neural network models proposed in [1] for classification of traffic signs and compared their performance. I used TensorFlow to train and test the network using the German Traffic Sign Recognition Benchmark (GTSRB) dataset [2]. The third CNN model achieved the highest test accuracy, followed by the second CNN, and then the third CNN which achieved the lowest test accuracy. The first CNN had the shortest training time per parameter followed by the third CNN, and then the first CNN, which had the longest training time per parameter.All my code is available at:

`https://github.com/ShahiraAmin/neural_project.git`

References

[1] A. Shustanov and P. Yakimov, "CNN design for real-time traffic sign recognition," *Procedia engineering*, vol. 201, pp. 718–725, 2017.

[2] "GTSRB - german traffic sign recognition benchmark," https://www.kaggle.com/datasets/meowmeowmeowmeowmeow/gtsrb-german-traffic-sign.