Traffic Sign Recognition Using Convolution Neural Network

Venkata Rohil Wardhan Kancharla[1] Wen Dong[2]

1.Abstract

This paper proposes a technique for detecting and classifying traffic signs using a convolution neural network (CNN). It describes the characteristics, requirements, and difficulties faced in detecting and classifying traffic signs. Uncrewed cars are part of the Intelligent Transportation Systems (ITS) that have steadily advanced despite being on the road. A critical technological issue is how to carry out accurate and effective traffic sign detection and identification while removing interference caused by diverse ambient conditions. However, traditional visual object identification has limitations since it primarily depends on extracting visual features like color and edge. Deep learning formed the foundation for creating the convolutional neural network (CNN) for visual object recognition. Alerting the driver of a car on the road signs like speed is also an important aspect of Traffic Sign Recognition (TSR). A preprocessed dataset is taken where the grayscale data is already converted to RGB, and the image is converted into float numbers along with the mean normalization.

2.Introduction

There is a need to design a system for the recognition of traffic signs in the vicinity of the vehicle in Advanced Driver Assistance Systems (ADAS). It is investigated how different traffic indicators may be detected and categorized effectively. For the building of a holistic model with an all-encompassing answer, a two-stage process is suggested. As part of the first phase, the YOLO v3 algorithm is used to effectively localize road signs (You Only Look Once). The first step groups the traffic signs into four categories based on their forms. The second stage involves accurately classifying the discovered traffic signs into one of the 43 predefined categories. Another model with one convolutional neural layer serves as the foundation for the second stage. The German Traffic Sign Detection Benchmark (GTSDB) served as the model's training and validation datasets, comprising 630 and 111 RGB pictures, respectively. GTSRB was used to train the classification model, which achieved a testing accuracy of 0.868 using 66000 RGB pictures from the pure "NumPy" library.

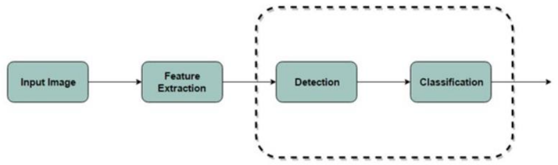


Fig 1: Overview of TSR

An effective overall system is created by adding a second convolutional layer to the second model before using it for final classification. Frames per second (FMS) between thirty-six and sixty-one were found during experiments on processing video files, making the system workable for real-time applications. The number of traffic signs that needed to be identified and categorized in each frame, which ranged from six to one, determined the frames per second.

3.Related Work

The early stages of traffic sign recognition use static color and shape characteristics. Transforming the image into HIS color space and using thresholding to extract the traffic signs was done as of the early stages in 2005. In 2008, using a Hough shape detector, the traffic signs have been detected by locating the edges of closed contours and with specific aspect ratios. The year 2010 is when traffic sign detection and classification has made real progress yielding good results by using HOG features with an SVM on a smaller self-generated dataset. In the following year, the same HOG features with a cascade filter have been used on small but real-time New Zealand dataset proving good results.

4. Model Setup

Deep Learning neural networks are classified into various based on their learning requirement.

Convolution Neural Network (CNN) has been used as the base in this model for detection and classification. Filters have been used to avoid overfitting of data. CNN is best for Image recognition and analysis, Natural Language Processing (NLP), recommendation systems and much more. The advancements in the field of computer vision and deep learning have been constructed with the use of CNN.

4.1 Convolution neural Network

Convolutional neural networks outperform other neural networks for inputs such as images, voice, or audio, for example. There are three basic categories of layers in them:

Convolution layer

Pooling layer

FC (Fully-Connected) layer

The convolutional layer is the top layer of a convolutional network. The Fully-Connected layer is the last layer, however convolutional layers can be followed by further convolutional layers or pooling layers. Greater areas of the image are identified by the CNN as it becomes more complicated with each layer. Early layers emphasize basic elements like colors and borders. The bigger features or forms of the item are first recognized when the visual data moves through the CNN layers, and eventually the desired object is recognized.

4.1.1 Convolution Layers

A convolution layer is where most of the computation occurs and is the core block of CNN. Convolution layer consists of a set of linear and non-linear operations. Unlike regular neural networks, convolution layer takes in mainly three parameters of the image like height, width and depth. Depth is the RGB of the image to determine how deep the image is and to classify the image.

4.1.2 Pooling Layer

It is the second layer of the convolution neural network. Its purpose is to gradually shrink the spatial size of the representation in order to lessen the number of parameters and computations required in the network and, as a result, to control overfitting. The Pooling Layer applies the MAX operation individually to each depth slice of the input and resizes it spatially. Filters of size 2x2 applied with a stride of 2, is the most common configuration which down samples each input depth slice by 2 along both width and height and discards 75% of the activations. Pooling layers reduce the dimensionality of the input to reduce input parameters known as down sampling. Limiting the input parameters helps CNN in reduced complexity and increased efficiency. Pooling layer helps CNN recognize features on the input even if the input is translated in the convolution layer. It mainly helps in extracting smooth and sharp features. There are mainly two types of pooling –

4.1.2.1 Max pooling which extracts patches from the input features and output the maximum value from each patch discarding others. Max pooling is the most common and popular type of pooling.

4.1.2.2 Average pooling reduces the feature map to a 1x1 array by averaging it with inputs of height and width but retaining the depth. Average pooling is only performed once before moving to the next layer.

4.1.3 Fully Connected Layers

FC layer is the last layer of the CNN architecture before the output where input of one layer is connected to another layer. The output of the high-level features in the data obtained from the previous layers is flattened and connected to the output layer. Each node in the output layer is directly connected to a node in the previous layer. There can be many convolution and pooling layers, but there will only one FC layer.

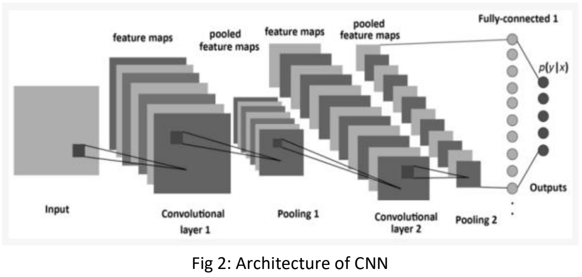


Fig 2: Architecture of CNN

4.2 Filters

Learned convolution weights are known as filters. A general indication of how well our network has learnt can be learnt by visually representing the learned weights. Various features of an image like vertical lines, horizontal lines, edges etc., can be identified using filters. A normal or gaussiandistribution is done on each filter after randomly initializing it. Number of color channels must remain same for the input image and filters. Filters learn more features in the deeper layers but perform extreme computations.

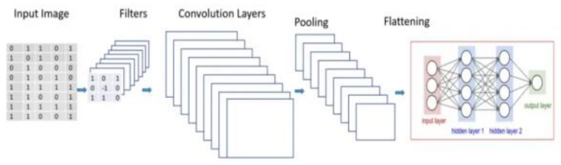


Fig 3: Architecture of CNN with Filters

4.3 Dropout

Dropout as the name suggests drops the training of some neurons (i.e., input data) to prevent the data from overfitting. Overfitting occurs when training data has more accuracy than testing data in terms of accuracy. When test data loss is much higher and very less for training data, overfitting occurs. It can be tackled by adding a dropout layer to the neural network model. A decimal value is assigned as the dropout value (i.e., 0.2) which indicates 20% of training nodes will not be trained. The data selected for dropout is chosen at random.

4.4 Activation Functions

Activation functions are used to evaluate the output of a neural network which is mapped to “0 or 1”, or “-1 or +1” based on the input data. It is also called transfer function. A default activation function is ‘ReLU’. Many activation functions are nonlinear some of which are sigmoid, Tanh, Softmax and ReLU. Activation functions are used in both the hidden layer and the output layers.

4.5 Padding

The number of pixel’s added to an image during CNN processing is defined in the padding. Padding is added to the outer layers of the image to allow more space for the filter to cover in the image.

4.6 Optimizer

Optimizers are programs or techniques that alter the neural network's weights and learning rate in order to minimize losses. Optimizer aims to minimize the function to solve optimization problems. Most common used optimizers are Gradient Descent, Adagrad, Adam and Stochastic Gradient Descent.

5. Dataset Used

The dataset consists of 43 different classes of images of traffic signs. Preprocessing of 86989 examples has been done after equalization of the dataset.

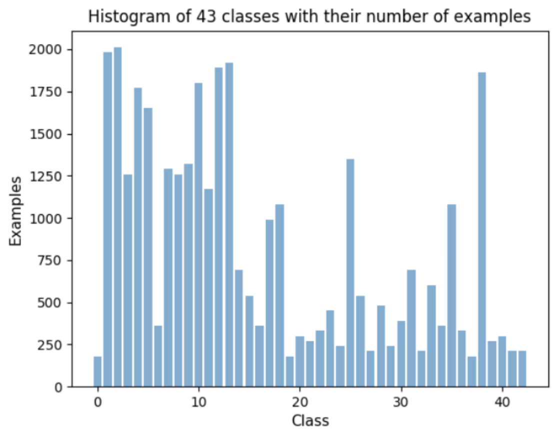


Fig 4: Original Dataset

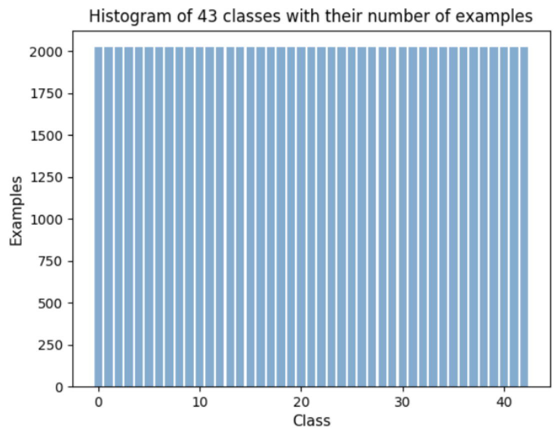
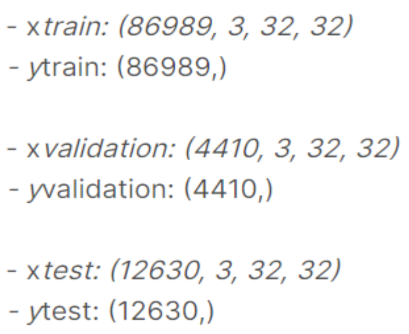


Fig 5: After Equalization

The source of the dataset is from Kaggle (https://www.kaggle.com/datasets/valentynsichkar/traffic-signs-preprocessed). The dataset comprises of RGB and Grayscale images with a mixture of inverted images as well. Different dataset for train, test ad validation has been taken. The data file used for this project is data2.pickle. The shapes of the dataset used for this project can be seen in the below figure 6.



```
- xtrain: (86989, 3, 32, 32)
- ytrain: (86989,)

- xvalidation: (4410, 3, 32, 32)
- yvalidation: (4410,)

- xtest: (12630, 3, 32, 32)
- ytest: (12630,)
```

Fig 6: Shapes of the whole dataset

6.Analysis of Dataset

The initial data is German Traffic Sign Recognition Benchmarks (GTSRB).

6.1 Preprocessing of Data

The data used for this project has already been preprocessed with a variety of parameters. The images have been preprocessed and reduced to 32x32 pixel images. The train. Valid and test data comprises of four keys defined by features, labels, sizes and cords. Features is a 4D array of raw pixel of traffic sign images. Label comprises of label id of the traffic sign image in 1D array. Size is a 2D array containing height and width of the image. Coord is a 2D array representing coordinates of bounding frame of the images. Text is defined to every class label along with shuffling and normalization and mean normalization of data.

6.2 Samples of Input Data

Samples of the input data are displayed as images and as text files from the data2.pickle file in the following figures.

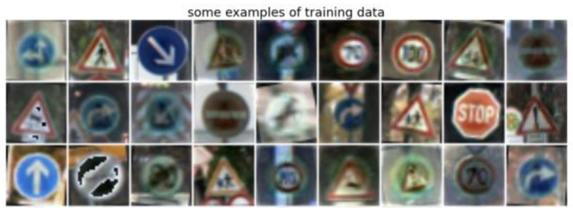
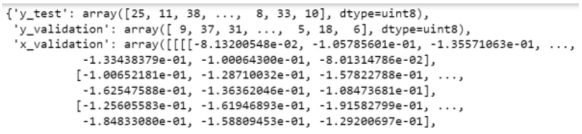
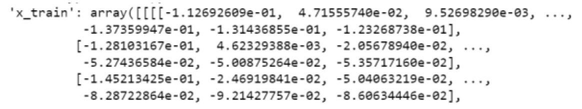


Fig 6: Exampled of Training Data

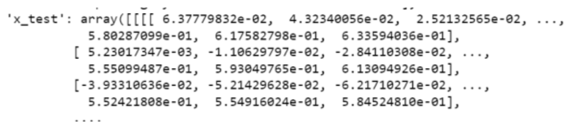


```
{'y_test': array([25, 11, 38, ...,  8, 33, 10], dtype=uint8),
 'y_validation': array([ 9, 37, 31, ...,  5, 18,  6], dtype=uint8),
 'x_validation': array([[[[-8.13200548e-02, -1.05785601e-01, -1.35571063e-01, ...,
          -1.33438379e-01, -1.00064300e-01, -8.01314786e-02],
         [-1.00652181e-01, -1.28710032e-01, -1.57822788e-01, ...,
          -1.62547588e-01, -1.36362046e-01, -1.08473681e-01],
         [-1.25605583e-01, -1.61946893e-01, -1.91582799e-01, ...,
          -1.84833080e-01, -1.58809453e-01, -1.29200697e-01],
```

Fig 7.1: Validation data in text



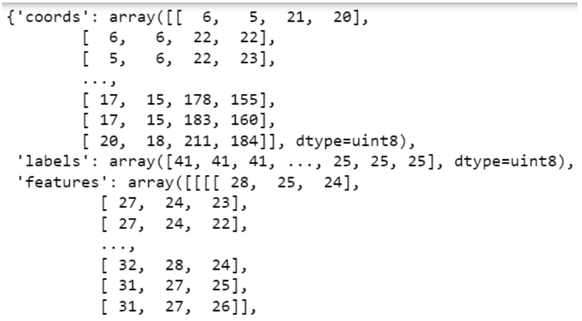
```
'x_train': array([[[[-1.12692609e-01,  4.71555740e-02,  9.52698290e-03, ...,
          -1.37359947e-01, -1.31436855e-01, -1.23268738e-01],
         [-1.28103167e-01,  4.62329388e-03, -2.05678940e-02, ...,
          -5.27436584e-02, -5.00875264e-02, -5.35717160e-02],
         [-1.45213425e-01, -2.46919841e-02, -5.04063219e-02, ...,
          -8.28722864e-02, -9.21427757e-02, -8.60634446e-02],
```



```
'x_test': array([[[[ 6.37779832e-02,  4.32340056e-02,  2.52132565e-02, ...,
          5.80287099e-01,  6.17582798e-01,  6.33594036e-01],
         [ 5.23017347e-03, -1.10629797e-02, -2.84110308e-02, ...,
          5.55099487e-01,  5.930049765e-01,  6.13094926e-01],
         [-3.93310636e-02, -5.21429628e-02, -6.21710271e-02, ...,
          5.52421808e-01,  5.54916024e-01,  5.84524810e-01],
         ....
```

Fig 7.2: Train and Test data in text

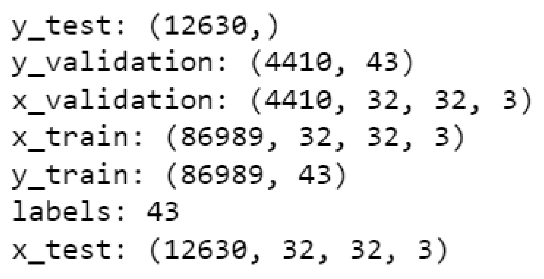
This is different to before the preprocessing of data shown in the below figure 8.



```
{'coords': array([[  6,   5,  21,  20],
       [  6,   6,  22,  22],
       [  5,   6,  22,  23],
       ...,
       [ 17,  15, 178, 155],
       [ 17,  15, 183, 160],
       [ 20,  18, 211, 184]], dtype=uint8),
 'labels': array([41, 41, 41, ..., 25, 25, 25], dtype=uint8),
 'features': array([[[[ 28,  25,  24],
         [ 27,  24,  23],
         [ 27,  24,  22],
         ...,
         [ 32,  28,  24],
         [ 31,  27,  25],
         [ 31,  27,  26]],
```

Fig 8: Train data before preprocessing

All the keys in the datasethave been converted to categorical valaue and their shapes have been displayed in figure 9 below.



```
y_test: (12630,)
y_validation: (4410, 43)
x_validation: (4410, 32, 32, 3)
x_train: (86989, 32, 32, 3)
y_train: (86989, 43)
labels: 43
x_test: (12630, 32, 32, 3)
```

Fig 9: Shapes of Input Data

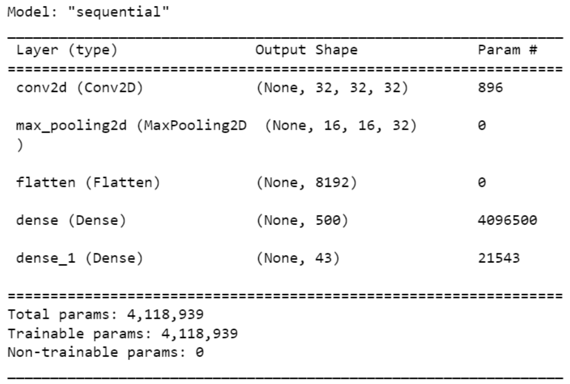
7.Implementation

A total of 3 different models have been trained. One without the addition of filters to display overfitting of data. The other 2 with the addition of filters.

7.1 Model-1

Model 1 uses 1 convolution layer with 32 nodes, input shape of (32, 32, 3) respectively. A kernel size of 3 and padding of same size with ‘ReLU’ activation function have been given as the hyperparameters in the convolution layer. 2-Dimensional max pooling of pool size 2 is used for the pooling layer. The data is then flattened using Flatten function to convert to 1 dimensional array in the Fully-Connected (FC) layer. The remaining neural network 500 filter nodes with activation function ‘ReLU’ on one layer of NN and 43 output nodes with activation function ‘Softmax’. The total trainable paarameters of this model are 4,118,939 input data. A learning rate of “1e^-3*0.95**(x+epochs)” with a batch size 5 nd verbose

1 is given. An “Adam” optimizer along with a loss “Categorical Crossentropy” is also used.



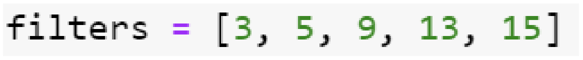
```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, 32, 32, 32)        896

 max_pooling2d (MaxPooling2D  (None, 16, 16, 32)       0
 )

 flatten (Flatten)           (None, 8192)              0

 dense (Dense)               (None, 500)               4096500

 dense_1 (Dense)             (None, 43)                21543

=================================================================
Total params: 4,118,939
Trainable params: 4,118,939
Non-trainable params: 0
_________________________________________________________________
```

Fig 10: Summary of Model 1

Figure 10 shows the summary of model 1.

7.2 Model-2

Model 2 also uses 1 convolution layer with 32 nodes, input shape of (32, 32, 3) respectively. A kernel size of 3 and padding of same size with ‘ReLU’ activation function have been given as the hyperparameters in the convolution layer. 2-Dimensional max pooling of pool size 2 is used for the pooling layer. The data is then flattened using Flatten function in the Fully-Connected (FC) layer. The remaining neural network 500 filter nodes with activation function ‘ReLU’ on one layer of NN and 43 output nodes with activation function ‘Softmax’. A learning rate of “1e^-3*0.95**(x+epochs)” with a batch size 5 is given. An “Adam” optimizer along with a loss “Categorical Crossentropy” is also used. Filters with different sizes of “3, 5, 9, 13, 15” respectively are used in model 2. The filters are taken of the input shape of “3x3, 5x5, 9x9, 13x13, 15x15”. The kernel size is adjusted to take as the size of the filter. The verbose is changed to 0.

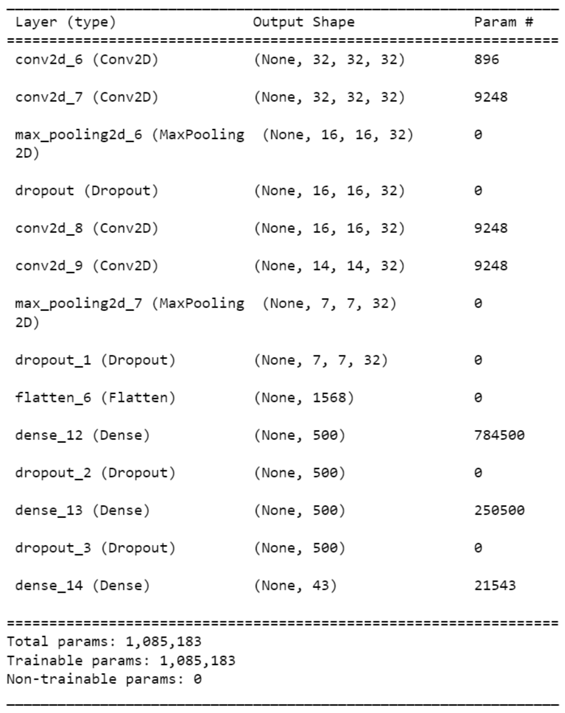


```
filters = [3, 5, 9, 13, 15]
```

Fig 11: Different Filter Sizes Trained

7.3 Model-3

For model 3, 4 convolution layers have been added with all input nodes of 32 with 2 max pooling layers and 4 dropout layers of 50% and 3 NN layers with (500, 500, 43) input nodes and with all activation function of ‘ReLU’ and last layer of ‘Softmax’ 3 padding same and 1 valid padding with 1 flatten layer. An optimizer ‘Adam’ and loss ‘categorical crossentropy’ with kernel size equal to the size of the filter. Only 3 filters of 3, 5, 9 have been used. Total trainable parameters are 1,085,183.



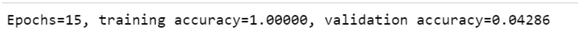
```
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d_6 (Conv2D)           (None, 32, 32, 32)        896

 conv2d_7 (Conv2D)           (None, 32, 32, 32)        9248

 max_pooling2d_6 (MaxPooling  (None, 16, 16, 32)       0
 2D)

 dropout (Dropout)           (None, 16, 16, 32)        0

 conv2d_8 (Conv2D)           (None, 16, 16, 32)        9248

 conv2d_9 (Conv2D)           (None, 14, 14, 32)        9248

 max_pooling2d_7 (MaxPooling  (None, 7, 7, 32)         0
 2D)

 dropout_1 (Dropout)         (None, 7, 7, 32)          0

 flatten_6 (Flatten)         (None, 1568)              0

 dense_12 (Dense)            (None, 500)               784500

 dropout_2 (Dropout)         (None, 500)               0

 dense_13 (Dense)            (None, 500)               250500

 dropout_3 (Dropout)         (None, 500)               0

 dense_14 (Dense)            (None, 43)                21543

=================================================================
Total params: 1,085,183
Trainable params: 1,085,183
Non-trainable params: 0
_________________________________________________________________
```

Fig 12: Summary of Model 3

8.Final Results

8.1 Model-1

For Model-1, the data is trained for 15 epochs or iterations.



```
Epochs=15, training accuracy=1.00000, validation accuracy=0.04286
```

Fig 13; Result of Model-1

The trained data is overfitted yielding very low validation accuracy of around 0.4%.

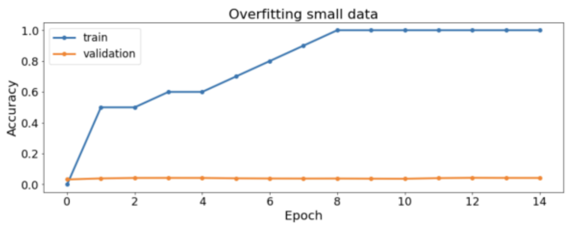
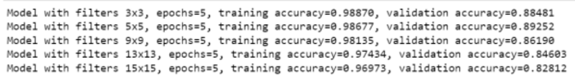


Fig 14: Graph of Model 1 Trained data

8.2 Model-2

In model-2, output validation accuracy for every filter with epochs 5 for every filter trained is displayed.



```
Model with filters 3x3, epochs=5, training accuracy=0.98870, validation accuracy=0.88481
Model with filters 5x5, epochs=5, training accuracy=0.98677, validation accuracy=0.89252
Model with filters 9x9, epochs=5, training accuracy=0.98135, validation accuracy=0.86190
Model with filters 13x13, epochs=5, training accuracy=0.97434, validation accuracy=0.84603
Model with filters 15x15, epochs=5, training accuracy=0.96973, validation accuracy=0.82812
```

Fig 15: Validation accuracy of every filter of model-2

The training and validation accuracy graphs have been displayed in the below figures.

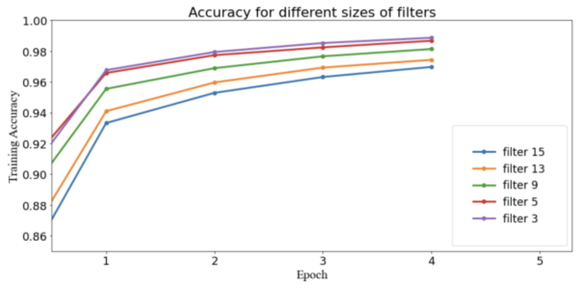


Fig 16: Training accuracy of Model-2 for different filters

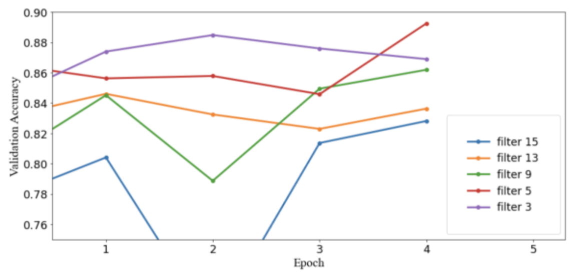


Fig 16: Validation accuracy of Model-2 for different filters

The testing accuracy of every filter for the model-2 has been shown in the below figure.

```
data2 filter 3 testing accuracy = 0.88124
data2 filter 5 testing accuracy = 0.88979
data2 filter 9 testing accuracy = 0.87736
data2 filter 13 testing accuracy = 0.86041
data2 filter 15 testing accuracy = 0.84838
```

Fig 17: Testing accuracy of every filter of Model-2

The classificationtime of every filter of model-2 has been shown the below figure.

```
data2 filter 3 classification time = 0.03897
data2 filter 5 classification time = 0.03657
data2 filter 9 classification time = 0.03689
data2 filter 13 classification time = 0.03597
data2 filter 15 classification time = 0.03395
```

Fig 18: Classification time of filters for Model-2

The output of the trained data with different filters are shown in the below figures. This shows how the classification and detection is done to the training models after data training.

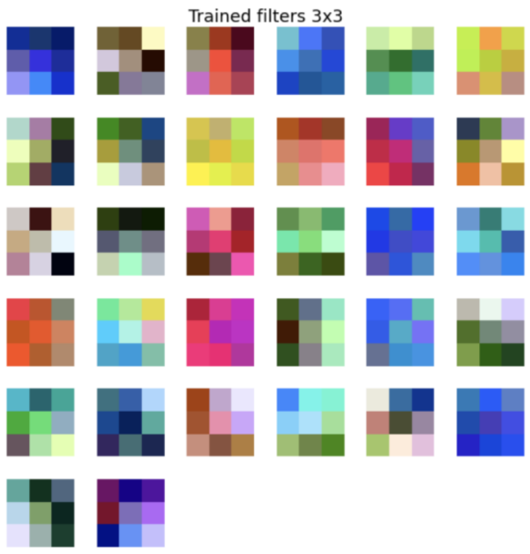


Fig 19: Trained output of Filter 3x3

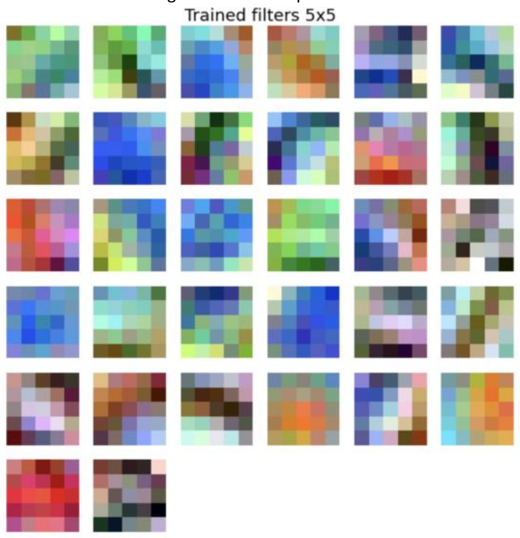


Fig 20: Trained output of Filter 5x5

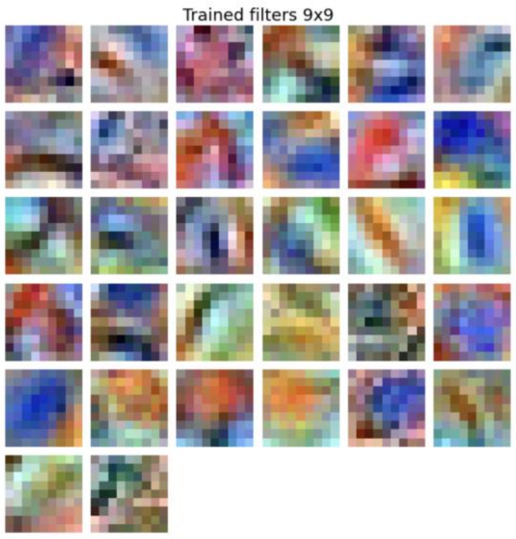


Fig 21: Trained output of Filter 9x9

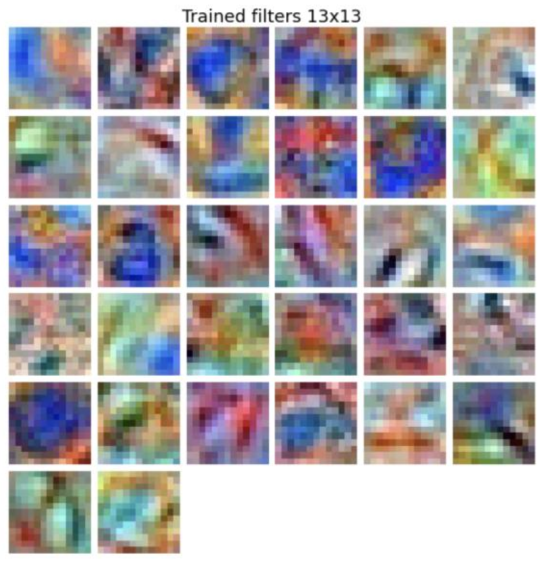


Fig 22: Trained output of Filter 13x13

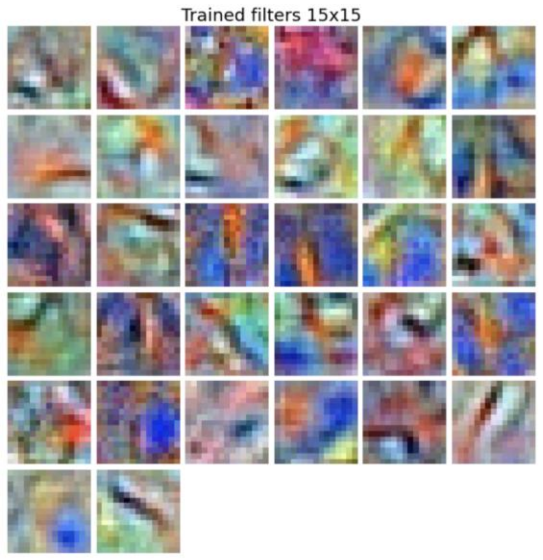


Fig 23: Trained output of Filter 15x15

8.3 Model-3

Training and validation accuracy of model-3 for different filter sizes can be viewed from the below figure.

```
Model with filters 3x3, epochs=5, training accuracy=0.86551, validation accuracy=0.93447
Model with filters 5x5, epochs=5, training accuracy=0.78795, validation accuracy=0.89773
Model with filters 9x9, epochs=5, training accuracy=0.02306, validation accuracy=0.03401
```

Fig 24: Training and Validation Accuracy of Model-3

The plots of train accuracy and validation accuracy for different filter sizes of model-3 can be seen from figure 2 and figure 26.

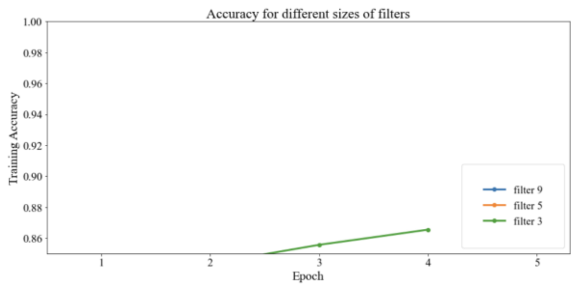


Fig 25: Plot of Training Accuracy of Model-3

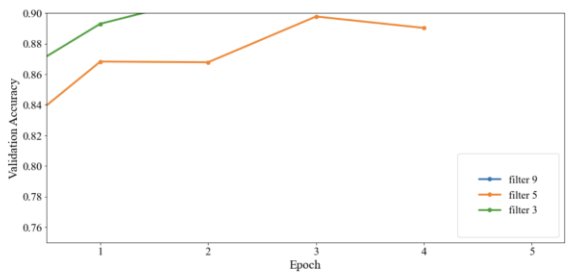


Fig 26: Plot of Validation Accuracy of Model-3

The accuracy of test data for model-3 can be viewed from the figure 27.

```
data2 filter 3 testing accuracy = 0.92201
data2 filter 5 testing accuracy = 0.87395
data2 filter 9 testing accuracy = 0.03088
```

Fig 27: Trresting Accuracy for various filters of Model-3

Classification time for different filters of Model-3 can be observed from figure 28.

```
data2 filter 3 classification time = 0.03604
data2 filter 5 classification time = 0.03740
data2 filter 9 classification time = 0.07459
```

Fig 28: Trained output of Filter 15x15

The display of how the traffic signs are cassified after passing through the model parameters is observed from the below figures 29, 30 and 31.

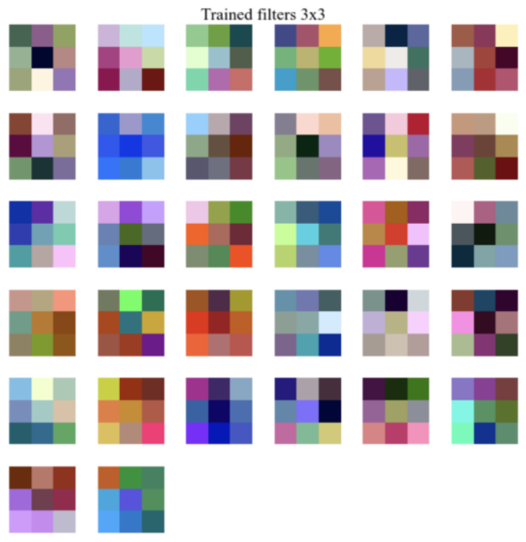


Fig 29: Trained output of Filter 3x3

Fig 30: Trained output of Filter 5x5

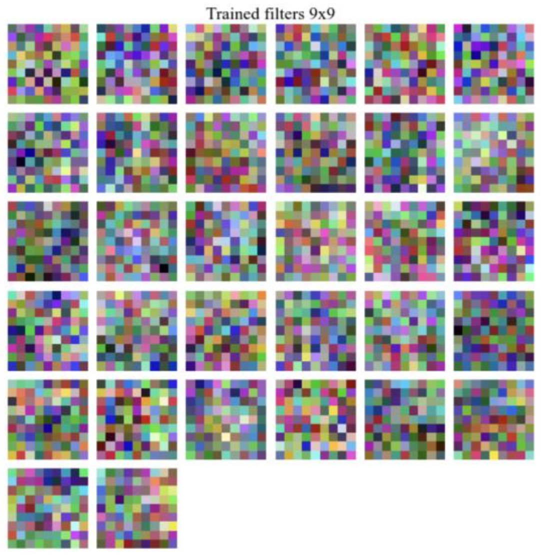


Fig 31: Trained output of Filter 9x9

9. Limitations

Although the traffic sign recognition is a core technology for the present and future of road safety and traffic community, there are some limitations. During severe weather conditions like heavy rain, snow the traffic signs cannot be recognized easily. Traffic signs may not be recognized when vehicle is moving at high speeds risking collisions and severe accidents.

10) Future Work

Road safety is the most important aspect of travel. Future work involves building a better deep learning model for effective detection and classification of traffic signs. This is one of the core technologies to pave the future of transportation through Self-Driving cars. Another future scope is the improvement of ADAS system and warning the driver to be vary of the sign crossed automatically through voice-assistant in the car.

9 References

[1] A. De La Escalera, L.E. Moreno, M.A. Salichs, J.M. Armingol **Road traffic sign detection and classification** IEEE Trans. Ind. Electron., 44 (6) (1997), pp. 848-859

[2] Á. Arcos-García, J.A. Álvarez-García, L.M. Soria-Morillo **Deep neural network for traffic sign recognition systems: an analysis of spatial transformers and stochastic optimisation methods** Neural Netw., 99 (2018), pp. 158-165

[3] Ren S., He K., R. Girshick, Sun J. **Faster r-cnn: towards real-time object detection with region proposal networks** Proceedings of Advances in Neural Information Processing Systems (2015), pp. 91-99

[4] Dai J., Li Y., He K., Sun J. **R-fcn: object detection via region-based fully convolutional networks** Proceedings of Advances in Neural Information Processing Systems (2016), pp. 379-387

[5] J. Deng, W. Dong, R. Socher, Li L.-J., Li K., Fei-Fei L. **Imagenet: a large-scale hierarchical image database** Proceedings of IEEE Conference on Computer Vision and Pattern Recognition, CVPR, IEEE (2009), pp. 248-255

[6] J. Huang, V. Rathod, C. Sun, M. Zhu, A. Korattikara, A. Fathi, I. Fischer, Z. Wojna, Y. Song, S. Guadarrama, K. Murphy **Speed/accuracy trade-offs for modern convolutional object detectors** 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2017), pp. 3296-3297

[7] S. Houben, J. Stallkamp, J. Salmen, M. Schlipsing, C. I gel **Detection of traffic signs in real-world images: the German traffic sign detection benchmark** Proceedings of the 2013 International Joint Conference on Neural Networks, IJCNN, IEEE (2013), pp. 1-8

[8] M. Mathias, R. Timofte, R. Benenson, L. Van Gool **Traffic sign recognitionâll how far are we from the solution?**

Proceedings of the 2013 International Joint
Conference on Neural Networks, IJCNN, IEEE (2013),
pp. 1-8
[9] Wang G., Ren G., Wu Z., Zhao Y., Jiang L.
**A robust, coarse-to-fine traffic sign detection
method** Proceedings of the 2013 International Joint
Conference on Neural Networks, IJCNN (2013),
pp. 1-5
[10]
Zang D., Zhang J., Zhang D., Bao M., Cheng J., Tang K.
**Traffic sign detection based on cascaded
convolutional neural networks** Proceedings of 2016
17th IEEE/ACIS International Conference on
Software Engineering, Artificial Intelligence,
Networking and Parallel/Distributed Computing,
SNPD (2016), pp. 201-206
[11] Zhu Y., Zhang C., Zhou D., Wang X., Bai X., Liu W.
**Traffic sign detection and recognition using fully
convolutional network guided proposals**
Neurocomputing, 214 (2016), pp. 758-766
[12] A. Mogelmose, M.M. Trivedi, T.B. Moeslund
**Vision-based traffic sign detection and analysis for
intelligent driver assistance systems: perspectives
and survey** IEEE Trans. Intell. Transp.
Syst., 13 (4) (2012), pp. 1484-1497
[13] F. Jurišić, I. Filković, Z. Kalafatić
**Multiple-dataset traffic sign classification with
onecnn** Proceedings of 2015 3rd IAPR Asian
Conference on Pattern Recognition,
ACPR, IEEE (2015), pp. 614-618
[14] R. Timofte, K. Zimmermann, and L. Van Gool.
**Multi-view traffic sign detection, recognition, and
3d localisation.** Machine Vision and Applications,
December 2011.