International Journal of Scientific Research in Computer Science, Engineering and Information Technology
ISSN : 2456-3307 (www.ijsrcseit.com)
doi : https://doi.org/10.32628/CSEIT217545

Traffic Sign Classification Using Convolutional Neural Network

Pranav Kale*, Mayuresh Panchpor, Saloni Dingore, Saloni Gaikwad, Prof. Dr. Laxmi Bewoor
Computer Engineering, VIIT, Pune, Maharashtra, India

Article Info
Volume 7, Issue 6
Page Number: 01-10

Publication Issue :
November-December-2021

Article History
Accepted : 01 Nov 2021
Published : 07 Nov 2021

ABSTRACT

In today's world, deep learning fields are getting boosted with increasing speed. Lot of innovations and different algorithms are being developed. In field of computer vision, related to autonomous driving sector, traffic signs play an important role to provide real time data of an environment. Different algorithms were developed to classify these Signs. But performance still needs to improve for real time environment. Even the computational power required to train such model is high. In this paper, Convolutional Neural Network model is used to Classify Traffic Sign. The experiments are conducted on a real-world data set with images and videos captured from ordinary car driving as well as on GTSRB dataset [15] available on Kaggle. This proposed model is able to outperform previous models and resulted with accuracy of 99.6% on validation set. This idea has been granted Innovation Patent by Australian IP to Authors of this Research Paper. [24]

Keywords: GTSRB dataset, Classify Traffic Sign, Convolutional Neural Network

I. INTRODUCTION

In recent months, automation in computer vision sector has increased on a large scale. Different types of technology are upcoming, modified in different categories in automation world. But to apply this Artificial Intelligence in real time environment and run it successfully is the crucial part. Traffic signs classification is one of the foremost important integral parts of autonomous vehicles and advanced driver assistance systems. Most of the time driver missed traffic signs due to different obstacles and lack of attentiveness. Automating the process of classification of the traffic signs would help reducing accidents. Overall this will reduce the possibility of accidents and also help driver to get information of speed limits and other signs while he concentrates on driving.

Traffic sign classification is the process of classifying traffic signs along the road, including speed limit signs, yield signs, merge signs. This is achieved by using a Neural network model in the field of Deep Learning based on Convolutional approach. Main Aim of this approach is decreasing the compilation time required and also to increase overall accuracy while requiring less computational power.

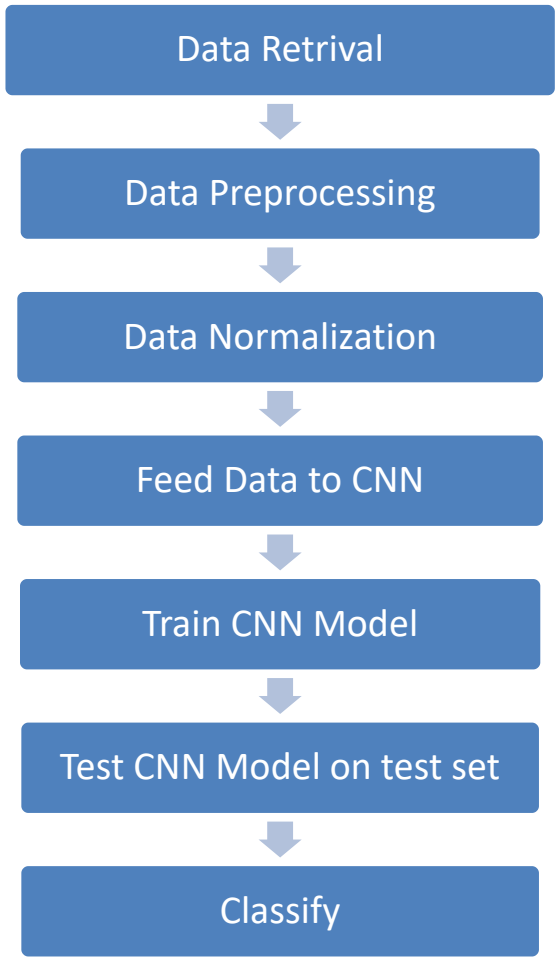
II. Data Set Used

Fig 2.1 represents all 43 different classes of GTSRB dataset. [17]

GTSRB dataset German Traffic Sign Recognition Benchmark is used for training, validation and testing purpose. This dataset is Single-Image, Multi-Class Classification dataset. It has not only more than 40 classes in total but also more than 50,000 images in total. These images are based on real life environment.

Fig 2.2: Sample overview of images in GTSRB dataset. [18]

III. Flowchart of Process of Classification



Data is transformed into a common shape of 30 x 30 x 3. This data is then ready for CNN model to train. After model is compiled, it is tested on testing dataset. Model is then ready to classify images.

IV. Working

GTSRB dataset is first transformed into size of 30 x 30 x 3. Normalization is a technique used to prepare data for use in machine learning. The goal of normalization is to change the values of numeric columns in the dataset to a common scale. It does normalization without distorting differences in the ranges of values. Each image is normalized so that value ranges from 0 to 1 for better performance and optimization of computational time. This normalized data is then randomly split into 75% training set and validation set as whole and 25% testing set. Training set has size of 39,209 images while validation set has 7842 images. Testing set consists of 12,631 images.

Training Data is then feeded to Convolutional neural network Model.

In CNN model [21], each input image is passed through a series of different convolutional layers. These layers have Filters, Kernels, Pooling layers and Fully-connected layers. As this is a multi-class problem, a SoftMax function is applied to classify these images. The output from this Softmax classifier will have values between 0 and 1.

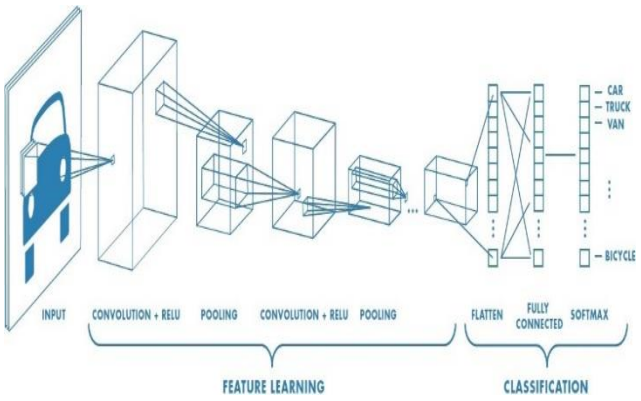


Fig 4.1 Complete Flow of CNN to process an input image and classify object based on classifier output values. [16]

As stated above, Convolutional Layer is first layer. It extracts features like vertical edges, horizontal edges from input image. It preserves relationship between pixels by learning features using small squares of input data. This is mathematical operation. It generally takes two inputs such as image matrix and filter or kernel.

Consider an Image matrix of Dimensions of (h x w x d), a filer of (fh x fw x d). Output of this layer is of size (h- fh + 1) x (w – fw + 1) x 1.

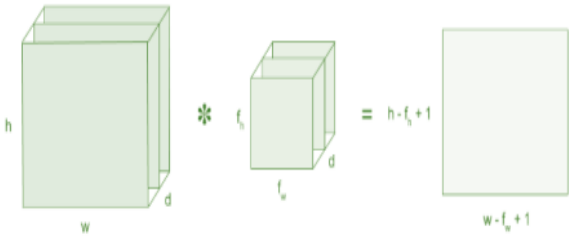


Fig 4.2: Image matrix multiplies kernel or filter matrix [16]

With different filters, various operations can be performed such as Identification, Edge detection, Blur and Sharpen.

These calculations may result in negative result. To avoid this a non-linearity is introduced.

ReLU – Rectified Linear Unit for a non-linear operation. Output is calculated by function f(x)= max (0, x).

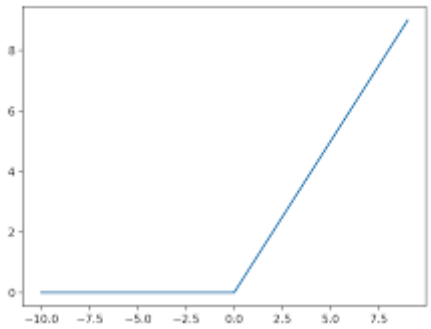


Fig 4.3: Line Plot of Rectified Linear Activation for Negative and Positive Inputs [19]

Other Non-linear functions are also available such as tanh or sigmoid. But in our model, ReLU gives highest accuracy and is perfectly fitted in tuning of CNN.

After this Pooling layer is used. Pooling layer reduces the number of parameters when images are too large. It is same as down sampling where images are reduced in dimensionality but retains all important features. Max pooling layer is used in this CNN mode.

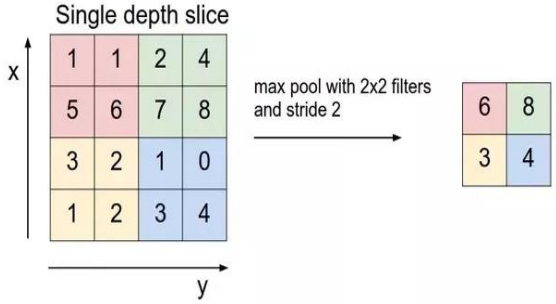


Fig 4.4: Max Pooling [16]

Max pooling layer takes the largest unit from the ReLU Feature map. As shown in above example.

All the above process is again repeated with different filter size or max pooling size with an addition Dropout layer.

Large neural nets trained on relatively small datasets can overfit the training data. To avoid the problem of Overfitting the data, a method of regularization named Dropout is used. [22]

The Dropout layer randomly sets input units to 0 with a frequency of rate at each step during training time, which helps prevent overfitting. Inputs not set to 0 are scaled up by 1/ (1 - rate) such that the sum over all inputs is unchanged.

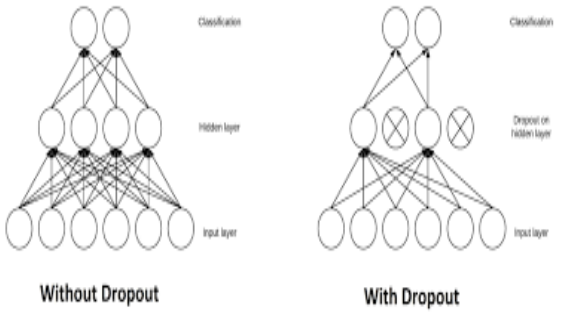


Fig 4.5: Dropout optimization comparison [20]

The above images show how the link to each neuron in neural network is skipped randomly.

After all Features are extracted completely, a Fully connected layer is used. For a fully connected layer to work, output from above layer needs to flattened into a vector. With use of Fully connected Layer, all features are combined together and model is created.

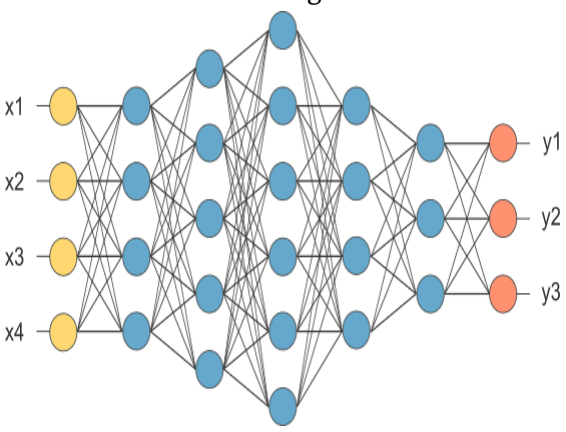


Fig 4.6 Fully connected Layer with 4 input and 3 output [16]

After that activation function of Softmax is used to classify the output as Stop sign, Speed limit Signs and so on.

V. Implementation of CNN

The Convolutional neural Model is created with help of Keras Framework 2.0 and TensorFlow v2.4.

The figure 6.1 is the basic outline of what model looks like.

First Layer is Convolutional layer. It has 32 Filters with Kernel Size of (5,5). Activation for this Layer is ReLU function. Input image is of shape (30,30,3). This layer gives output image of size (26,26,32). Total Parameters trained in this layer are 2432.

Second Layer is also a Convolutional Layer with same size of filter and kernel Size 32 and (5,5) respectively. It has activation function ReLU but this layer output gives image of size (22,22,32). Total parameters trained in this layer are 25,632.

Third Layer is a Max Pooling Layer. It has a pool size of (2,2). Input to this layer is of size (22,22,32) and its output image of size (11,11,32).

Fourth Layer is a Dropout Layer with a Dropout rate of 25%. It helps to overcome the problem of Overfitting.

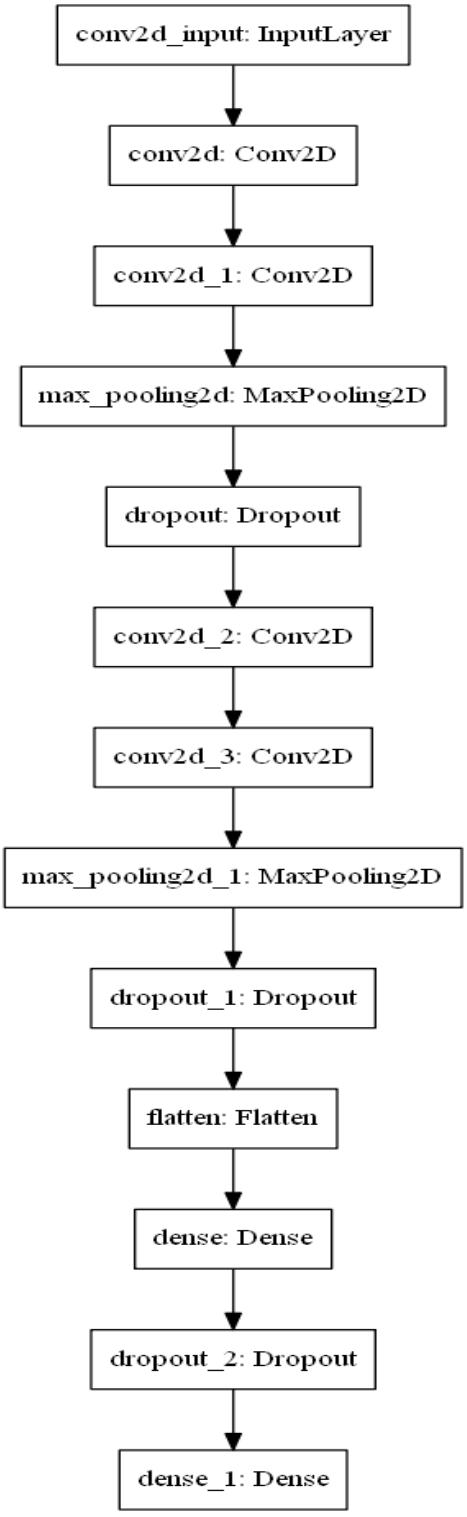


Fig 5.1: Structure Diagram of layers in CNN Model

Consider above 4 layers as one block where global features are extracted of input image. Another same block of 4 layer comprising of layer 5 ,6 ,7, 8 is used to extract local features of the input image.

So, 5th layer is Convolutional Layer with 64 Filters and kernel size of (3,3) and activation function ReLU. It gives output of shape (9,9,64) and total parameters trained are 18,496.

6th layer is also similar to 5th layer with same settings of filter, kernel size and activation function. It gives output of shape (7,7,64) and parameters trained are 36,928.

7th layer is a max pooling layer similar to 3rd layer with same pooling size of (2,2). Its output image is of shape (3,3,64).

8th layer is similar to 4th layer that is Dropout Layer with Dropout rate of 25%.

In 9th layer, input image is of size (3,3,64). So, it then flattened into a NumPy vector stacked in same column which has output size of (576). This is Flattening Layer.

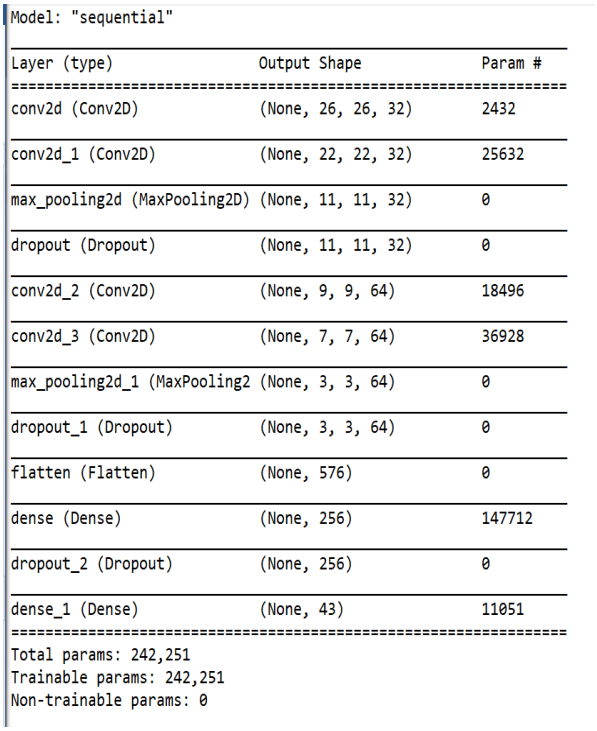
Fully Connected Layer is 10th layer where it dense to size of 256 that is, 576 inputs are combined to 256 outputs combining all the features extracted. Parameters trained in this layer are 1,47,712. Activation Function is again ReLU.

As Fully connected layers creates a model of its own interlinked dense network of numerous hidden internal layers. Dropout layer is used with a Dropout rate of 50%. This is 12th layer of CNN model.

8th layer is similar to 4th layer that is Dropout Layer with Dropout rate of 25%.

Final Layer of Model i.e. 13th layer of Model is also a Fully Connected Layer with input of shape (256) and output of shape (43). 43 is the total number of classes present in dataset. Here the activation Function used is SoftMax Classifier as it gives final classification result. Parameters trained here are 11,051.

Total parameters trained by this model are 2,42,251.



```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d (Conv2D)              (None, 26, 26, 32)        2432
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 22, 22, 32)        25632
_________________________________________________________________
max_pooling2d (MaxPooling2D) (None, 11, 11, 32)        0
_________________________________________________________________
dropout (Dropout)            (None, 11, 11, 32)        0
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 9, 9, 64)          18496
_________________________________________________________________
conv2d_3 (Conv2D)            (None, 7, 7, 64)          36928
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 3, 3, 64)          0
_________________________________________________________________
dropout_1 (Dropout)          (None, 3, 3, 64)          0
_________________________________________________________________
flatten (Flatten)            (None, 576)               0
_________________________________________________________________
dense (Dense)                (None, 256)               147712
_________________________________________________________________
dropout_2 (Dropout)          (None, 256)               0
_________________________________________________________________
dense_1 (Dense)              (None, 43)                11051
=================================================================
Total params: 242,251
Trainable params: 242,251
Non-trainable params: 0
```

Fig 5.2: Summary of CNN Model by Keras Framework

VI. Model Compilation

As Softmax Classifier outputs positive probability of multi-class, Loss function used to compile CNN model is Categorical Cross Entropy.

The categorical cross entropy loss function calculates the loss of an example by computing the following sum:

$$Loss = -\sum_{i=1}^{output\ size} y_i * \log \hat{y}_i$$

Equation 6.1: Categorical Cross Entropy Loss Function. [23]

Where ŷ i is the i-th scalar value in model output, yi is the target value.

This Loss is a good measure to distinguish two discrete probability distribution. Minus sign ensures that loss decreases when distributions get close to each other.

Adam optimizer is used as optimization algorithm to prevent gradient descent from exploitation.

Adam optimization is a stochastic gradient descent method that is based on adaptive estimation of first-order and second-order moments.

According to Kingma et al., 2014 [5], the method is "computationally efficient, has little memory requirement, invariant to diagonal rescaling of gradients, and is well suited for problems that are large in terms of data/parameters".

Model is then compiled by using above loss function and Adam optimizer for 20 epochs. Backword propagation is automated by Keras Framework.

Batch size used is 32 and validation set is used to validate the results.

Below table gives loss and accuracy on both sets for each epoch.

Epoch	Loss	Acc	Val_Loss	Val_Acc
1	1.2262	0.6531	0.1236	0.9652
2	0.2030	0.9390	0.0616	0.9837
3	0.1336	0.9604	0.0358	0.9909
4	0.0996	0.9706	0.0298	0.9917
5	0.0863	0.9745	0.0462	0.9867
6	0.0756	0.9778	0.0325	0.9913
7	0.0592	0.9832	0.0270	0.9943
8	0.0598	0.9823	0.0261	0.9954
9	0.0538	0.9838	0.0277	0.9939
10	0.0556	0.9840	0.0258	0.9945
11	0.0524	0.9847	0.0258	0.9938
12	0.0496	0.9854	0.0170	0.9955
13	0.0414	0.9878	0.0195	0.9952
14	0.0464	0.9869	0.0243	0.9948
15	0.0422	0.9883	0.0290	0.9943
16	0.0441	0.9877	0.0202	0.9955
17	0.0404	0.9887	0.0177	0.9963
18	0.0429	0.9881	0.0225	0.9959
19	0.0425	0.9872	0.0220	0.9963
20	0.0434	0.9881	0.0162	0.9966

This table indicates that with each epoch loss is decreasing and accuracy in increasing.

To further understand in more details let see the graph of epochs versus loss and epochs vs accuracy

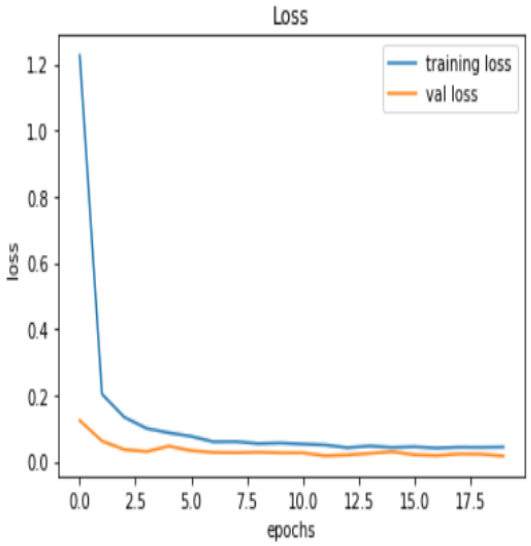


Fig 7.1: Graph of Loss function of training loss and validation loss

As it can be clearly seen that graph is nearly similar to ideal graph of loss function that should be increasing epochs, it is clear that there is gradient exploitation. Loss is decreasing with a stable curve.

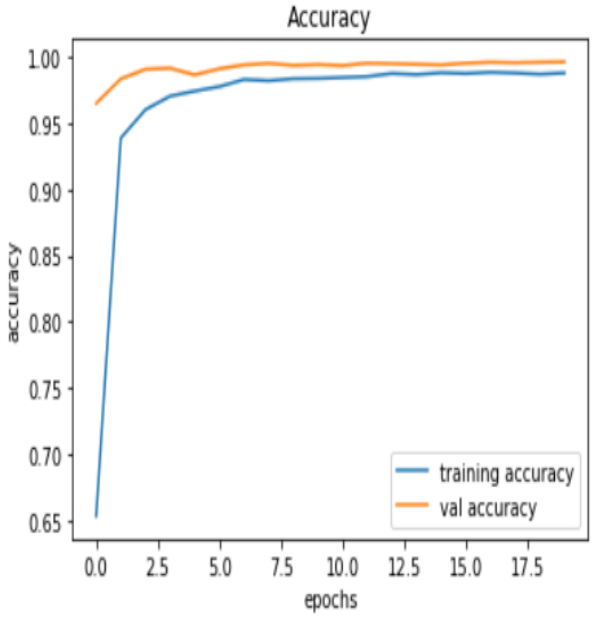


Fig 7.2: Graph of Accuracy v/s Epochs

It can be inferred from above graph that accuracy is increasing and has reach close to 1 that is perfect accuracy.

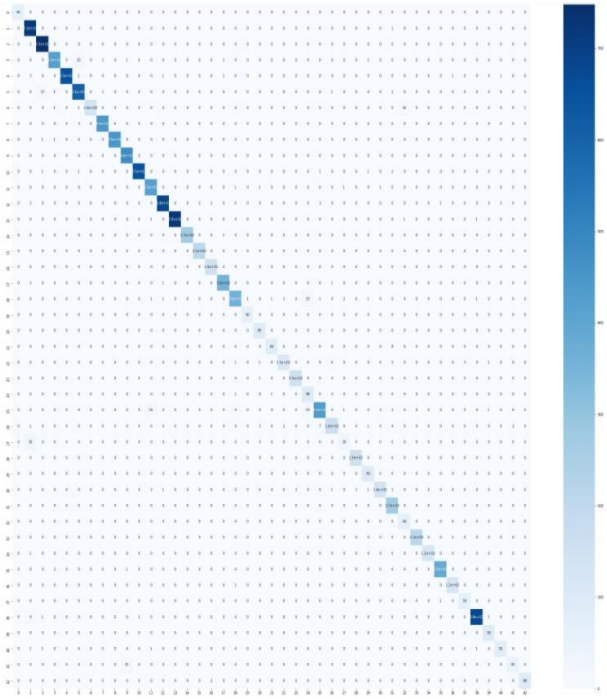


Fig 7.3 Confusion Matrix of Test Set of GTSRB

It can be clearly seen that confusion matrix is a perfect diagonal and prediction of result is accurate. This confusion matrix is of 43 classes i.e. of dimension (43 x 43). It gives summary of prediction result of classification of Traffic Signs.

Compilation Time required to train model:

- *For CPU compilation*:

Processor Used: AMD Ryzen 3600. **Time Required**: 44 sec per Epoch. Total Time: 880 sec.

- *For GPU Compilation*:

GPU Used: Nvidia GTX 1660 with computational power of 7.5

Cuda 10.1 and cudnn architecture required for GPU compilation.

Time required: 4 sec per Epoch.

Total Time required: 80 sec.

Some sample images are shown with their true label and predicted label. It can be inferred from the figure 7.4 that model is predicting correctly random images selected from test dataset.

Fig 7.4: Random images with their true and predicted classes

VII. Comparison of proposed Solution with other models

Method proposed above requires only 2,42,251 parameters. CNN model is finely tuned by using regularization techniques and optimization techniques. As data is normalised there in no problem of bias and variance. This CNN model is compiled in less time as compared to other models. This model also requires less Computation power for compilation purpose.

Recent high performed method Committee of CNNs [2] have used 25 networks and 3 convolutional layers with 2 fully connected layers along with manual data augmentation. Original dataset is modified by using translation, rotation etc., to get five modified version of that image. Committee of CNNs end up with total around 90 Million parameters whereas, our CNN method has around 2.4 lakh parameters.

Table below shows comparison of Different State-of-Art Highest Accuracy using different Algorithms:

Algorithm	Accuracy (%)
Proposed Method	***99.66***
Committee of CNNs [10]	99.46
Human Performance [1]	98.84
Multi-Scale CNN [2]	98.31
pLSA [3]	98.14
Random Forest [4]	96.14

VIII. Conclusion

The proposed solution of CNN model is highly fined tuned model with including different algorithms for optimization, regularization. This model requires less parameters to train as well as less computational cost. Time required for compilation of model is also less as compared to other models.

IX. REFERENCES

[1]. J. Stallkamp, M. Schlipsing, J. Salmen, and C. Igel. The German Traffic Sign Recognition Benchmark: A multi-class classification competition. In Proceedings of the IEEE International Joint Conference on Neural Networks, pages 14531460. 2011.

[2]. Sermanet, P., LeCun, Y. Traffic sign recognition with multi-scale convolutional networks. In Neural Networks (IJCNN), The 2011 International Joint Conference on (pp. 2809-2813). IEEE (2011)

[3]. Haloi, M. A novel pLSA based Traffic Signs Classification System. arXiv preprint arXiv:1503.06643 (2015).

[4]. Zaklouta, F., Stanciulescu, B., Hamdoun, O. Traffic sign classification using kd trees and random forests. In Neural Networks (IJCNN), The 2011 International Joint Conference on (pp. 2151-2155). IEEE (2011)

[5]. Diederik P. Kingma, Jimmy Ba, Adam: A Method for Stochastic Optimization, 3rd International Conference for Learning Representations, San Diego, 2015 , arXiv:1412.6980 cs.LG]

[6]. Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., ... Rabinovich, A. Going deeper with convolutions. arXiv preprint arXiv:1409.4842 (2014)

[7]. Ioffe, S., Szegedy, C. Batch normalization: Accelerating deep network training by reducing internal covariate shift. arXiv preprint arXiv:1502.03167 (2015)

[8]. He, K., Zhang, X., Ren, S., Sun, J. Delving deep into rectifiers : Surpassing human-level performance on imagenet classification. arXiv preprint arXiv:1502.01852 (2015)

[9]. Lai, Yan & Wang, Nanxin & Yusi, Yang & Lin, Lan. (2018). Traffic Signs Recognition and Classification based on Deep Feature Learning. 622-629. 10.5220/0006718806220629.

[10]. D. Cireşan, U. Meier, J. Masci and J. Schmidhuber, "A committee of neural networks for traffic sign classification," The 2011 International Joint Conference on Neural Networks, San Jose, CA, 2011, pp. 1918-1921, doi: 10.1109/IJCNN.2011.6033458.

[11]. Zhang, J., Wang, W., Lu, C. et al. Lightweight deep network for traffic sign classification. Ann. Telecommun. 75, 369–379 (2020). https://doi.org/10.1007/s12243-019-00731-9

[12]. F. Jurišić, I. Filković and Z. Kalafatić, "Multiple-dataset traffic sign classification with OneCNN," 2015 3rd IAPR Asian Conference on Pattern Recognition (ACPR), Kuala Lumpur, 2015, pp. 614-618, doi: 10.1109/ACPR.2015.7486576.

[13]. Dan Cireşan, Ueli Meier, Jonathan Masci, Jürgen Schmidhuber, Multi-column deep neural network for traffic sign classification, Neural Networks, Volume 32, 2012, Pages 333-338, ISSN 0893-6080, https://doi.org/10.1016/j.neunet.2012.02.023.

[14]. Zhe Zhu, Dun Liang, Songhai Zhang, Xiaolei Huang, Baoli Li, Shimin Hu; Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2016, pp. 2110-2118

[15]. http://benchmark.ini.rub.de/?section=gtsrb&subsection=dataset

[16]. https://medium.com/@RaghavPrabhu/understanding-of-convolutional-neural-network-cnn-deep-learning-99760835f148

[17]. Traffic sign recognition and classification with modified residual networks - Scientific Figure on ResearchGate. Available from: https://www.researchgate.net/figure/The-total-43-classes-in-GTSRB-From-top-to-bottom-there-are-four-categories_fig1_322945549 accessed 26 Nov, 2020]

[18]. An Efficient Traffic Sign Recognition Approach Using a Novel Deep Neural Network Selection Architecture: Proceedings of IEMIS 2018, Volume 3 - Scientific Figure on ResearchGate. Available from: https://www.researchgate.net/figure/Overview-of-the-GTSRB-Dataset_fig1_327389916 accessed 26 Nov, 2020]

[19]. https://machinelearningmastery.com/rectified-linear-activation-function-for-deep-learning-neural-networks/

[20]. https://www.baeldung.com/cs/ml-relu-dropout-layers

[21]. Valueva, M.V.; Nagornov, N.N.; Lyakhov, P.A.; Valuev, G.V.; Chervyakov, N.I. (2020). "Application of the residue number system to reduce hardware costs of the convolutional

neural network implementation". Mathematics and Computers in Simulation. Elsevier BV. 177: 232-243 doi: 10.1016/j.matcom.2020.04.031. ISSN 0378-4754.

[22].Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: a simple way to prevent neural networks from overfitting. J. Mach. Learn. Res. 15, 1 (January 2014), 1929–1958.

[23].Zhilu Zhang, Mert R. Sabuncu, Generalized Cross Entropy Loss for Training Deep Neural Networks with Noisy Labels 32nd Conference on Neural Information Processing Systems (NeurIPS 2018)., arXiv:1805.07836 cs.LG]

[24].Bewoor, Laxmi A.; Kale, Pranav and Panchpor, Mayuresh "A STREAMLINE TRAFFIC SIGN CLASSIFICATION SYSTEM UTILIZING CONVOLUTIONALNEURAL NETWORK MODEL" AUSTRALIAN Patent, 2021101273, April21, 2021.

Cite this article as :

Pranav Kale, Mayuresh Panchpor, Saloni Dingore, Saloni Gaikwad, Prof. Dr. Laxmi Bewoor, "Traffic Sign Classification Using Convolutional Neural Network", International Journal of Scientific Research in Computer Science, Engineering and Information Technology (IJSRCSEIT), ISSN : 2456-3307, Volume 7 Issue 6, pp. 01-10, November-December 2021. Available at
doi : https://doi.org/10.32628/CSEIT217545
Journal URL : https://ijsrcseit.com/CSEIT217545