Traffic Sign Detection using CNN

Abhishek Ayachit	Pranshu Shrivastava	Sudip Bala	Yash Deshpande
862188073	862186760	862188812	862187779
aayac001@ucr.edu	pshri002@ucr.edu	sbala027@ucr.edu	ydesh001@ucr.edu

ABSTRACT

Autonomous vehicles are the latest fad in the Artificial Intelligence sphere. Companies from Tesla to General Motors are investing in models that allow vehicles to automatically navigate on roads without human aid. Computer Vision for Autonomous Vehicles is a broad research and development topic. We aim to focus on a smaller but an essential aspect of this topic: traffic sign detection. Our aim is to explore and implement Convolutional Neural Network architectures to correctly classify traffic signs. The major challenge is to be able to classify images that are not clearly understandable such as images taken under low light conditions or blurred images.

Furthermore, we aim to improve accuracy and reduce generalization error of our model by implementing data augmentation algorithms to allow our classifier to model out of distribution images.

INTRODUCTION

Traffic signs are essential aids which communicate the traffic rules and guidelines to be followed by vehicles on the road. What makes it essential is that violation of traffic signs can lead to major accidents. The important research aspect in Traffic Sign detection to accurately detect various types of signs from “go slower” to “turn right”. Misclassification of labels can lead to major accidents hence our aim is to develop classification models to detect and classify traffic signs with minimum misclassifications.

Convolutional neural networks are the state of the art deep neural networks for image classification. The major advantage of convolutional neural networks to other architectures is its ability to make use of the hierarchical patterns in images and spatial correlations amongst pixels in images to learn complex patterns. Compared to fully connected neural networks CNN architectures tend to use fewer parameters which makes them suitable for image classification tasks.

There are various techniques which can be employed in Convolutional neural networks to improve regularization i.e. to reduce variance. One such technique is pooling which is a non-linear downsampling technique which reduces the number of parameters without losing important information. There have been multiple Convolutional Neural Network architectures since 1998 such as LeNet, VGG, ResNet, Inception that have been successful for image classification tasks. For our project we limit the scope to Resnet architectures.

RESNET and its variants[1]

Before RESNET, most Convolutional Neural Networks had multiple layers stacked up, these deep neural networks architectures were prone to overfitting and gradient vanishing problems, especially during propagating errors during backpropagation. The deeper the network goes the more its prone to performance gets saturated and then degrades rapidly. ResNet solves this problem by identity shortcut connection which allows skipping multiple layers which greatly allows to reduce overfitting. Skipping over layers

helps to avoid the problem of vanishing gradients by reusing activation layers from one of the previous layers.

ResNet has multiple variants which correspond to the depth of the architecture from ResNet 20 to ResNet 1001. The deeper the architecture, the more parameters the architecture has.

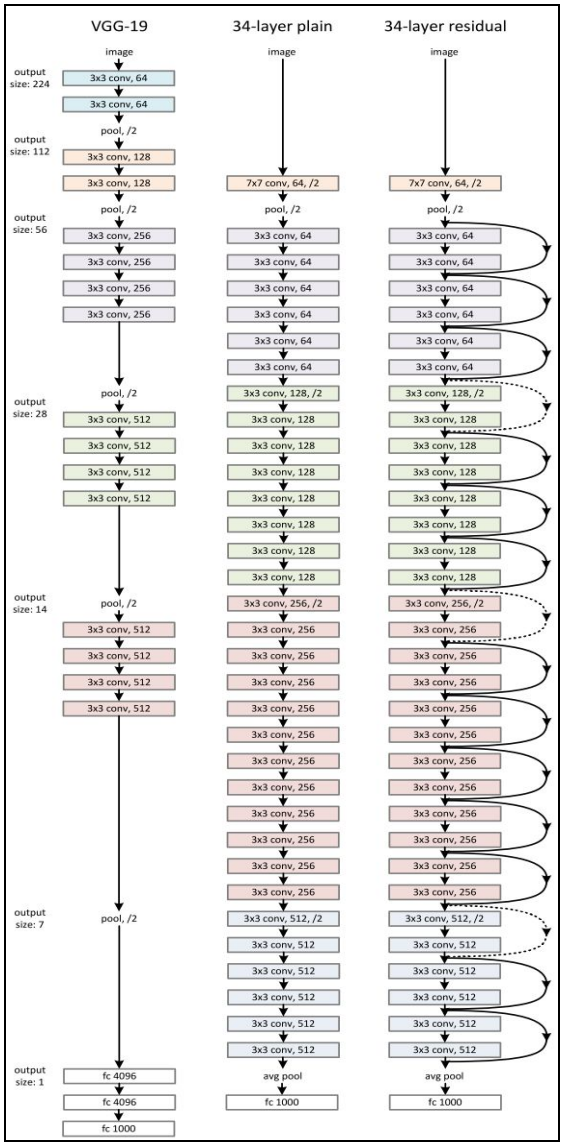

Fig 1: ResNet Architecture[1]

DATASET[2]

Link:https://www.kaggle.com/meowmeowmeowmeowmeow/gtsrb-german-traffic-sign

The dataset contains more than 50000 images with 43 classes making this a multiclass classification problem.

Content:

- Meta csv file: Path, ClassId, ShapeId, ColorId, SignId of the given image
- Training and Testing csv file: Width, Height, upper left and lower right x-y coordinates, ClassId, Path of the image

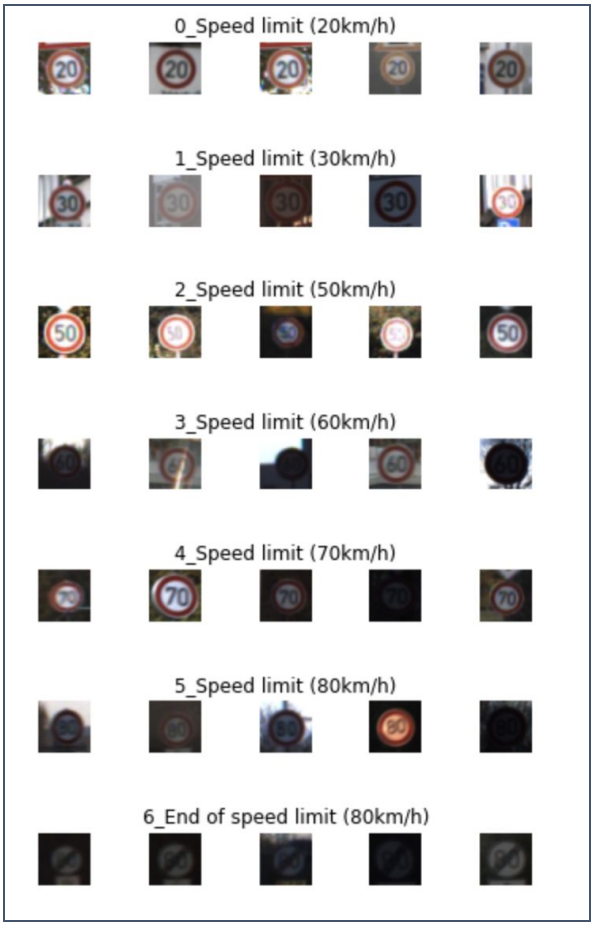

Fig 2: Dataset Example[2]

IMPLEMENTATION

We use GTRSB german benchmark dataset to build a classification model to classify traffic signs. Our input data is of 50000 RGB images of shape (50000, 32, 32, 3).

Data Preprocessing

We first split the dataset into data into training, testing and validation. We further normalize the data by using Z-Normalization which helps to better train our model.

For initialization of layers, we use HE-Normalization where weights are initialized keeping in mind the size of previous layers. It is particularly useful for ReLU activation function since it helps to attain global minimum of the cost function faster.
We also convert numerical label data to binary using one-hot encoding.

ResNet implementation

First we train our model using different ResNet architectures, to determine which architecture provides the best performance.

Fig 3 shows different training and validation accuracy and loss for different ResNet models. As per the results, ResNet 32 has better performance compared to other architectures. So we select ResNet 32 to implement our model.

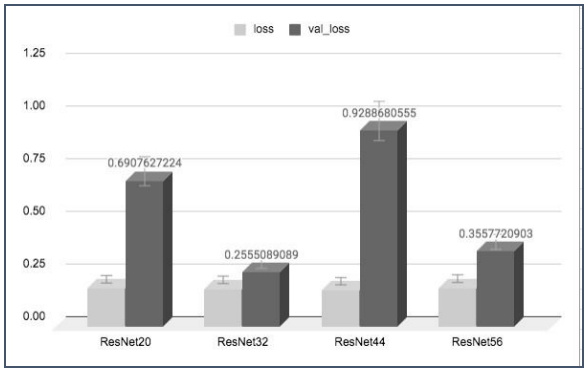

Fig 3: ResNet: training_loss vs validation_loss

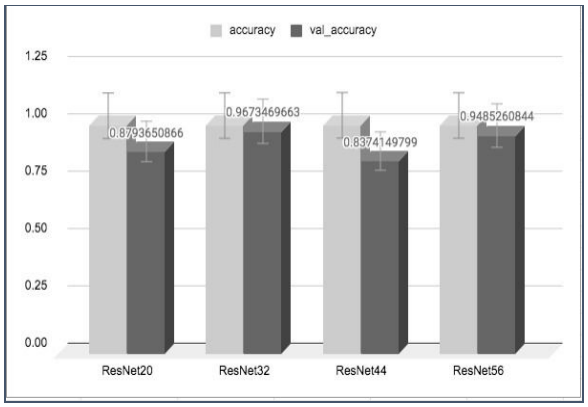

Fig 4: ResNet: training_acc vs validation_acc

Data Augmentation

Data Augmentation is the programmatic way of increasing the quantity of data to reduce overfitting of training models. Data augmentation increases bias and reduces variance. There are multiple ways to augment data, in this particular case we use two augmentation methods:

- Standard Augmentation which includes Shifting, Flipping, Rotation, Brightness, and Zooming

- Cutout Augmentation: Dropping out contiguous sections of inputs

Results for ResNet 32 after Data Augmentation

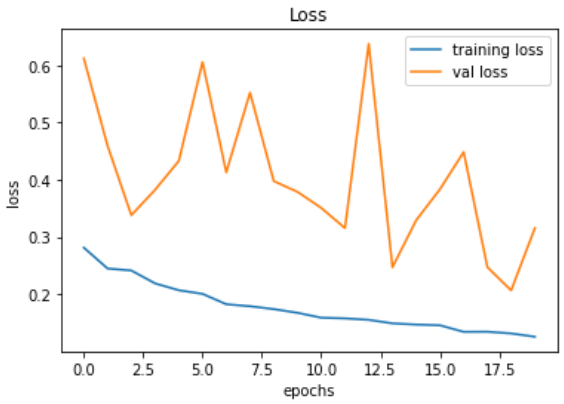

Fig 5: ResNet 32 with Data Augmentation: training_loss vs validation_loss

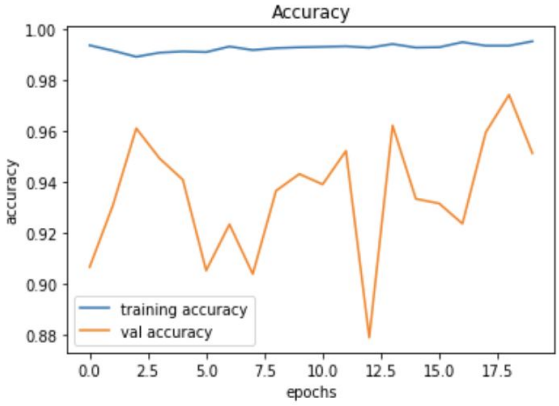

Fig 6: ResNet 32 with Data Augmentation: training_acc vs validation_acc

Although, training and validation is high but on a testing set which follows a very different distribution, we get accuracy 75.6. This proves that the ResNet model is not able to generalize well on testing data. Hence, there is a need to develop a new model that generalizes well on testing data.

Removing bias

Our training data is biased, some classes are under-represented. This under-representation of data will result in poor accuracy for some classes. We aim to alleviate this problem by removing bias by adding more samples using data augmentation.

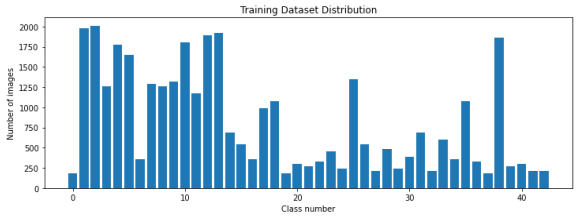

Fig 7: Training Dataset Distribution over 42 classes

Our Own Model

The failure of ResNet to perform well on testing data tells us that we need a simpler CNN architecture with high regularization.
So we design our own model as shown in fig 7. Adding multiple max-pooling layer and high dropout rate improves the testing accuracy since it learns a simpler function which is able to generalize well.
Fig 8 shows training and loss plots.

Layer (type)	Output Shape	Param #
conv2d_1 (Conv2D)	(None, 28, 28, 32)	2432
conv2d_2 (Conv2D)	(None, 26, 26, 64)	18496
max_pooling2d_1 (MaxPooling2	(None, 13, 13, 64)	0
dropout_1 (Dropout)	(None, 13, 13, 64)	0
conv2d_3 (Conv2D)	(None, 11, 11, 64)	36928
max_pooling2d_2 (MaxPooling2	(None, 5, 5, 64)	0
flatten_1 (Flatten)	(None, 1600)	0
dense_1 (Dense)	(None, 256)	409856
dropout_2 (Dropout)	(None, 256)	0
dense_2 (Dense)	(None, 43)	11051

Total params: 478,763
Trainable params: 478,763
Non-trainable params: 0

Fig 8: Specification of our CNN model

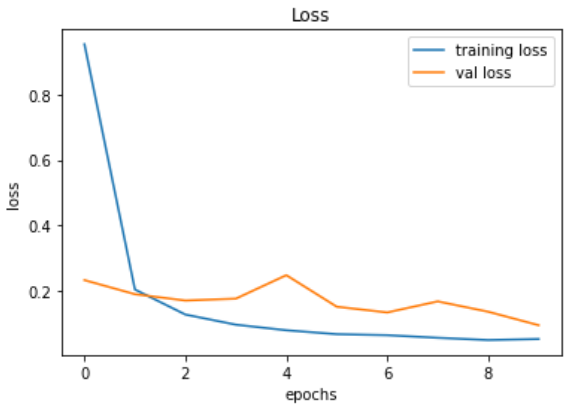

Fig 9: Our Model: training_loss vs validation_loss

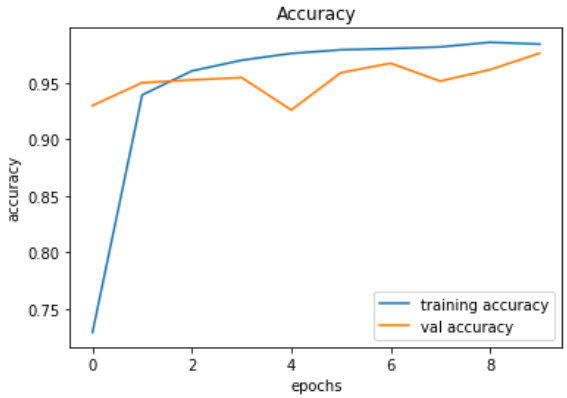

Fig 10: Our Model: training_acc vs validation_acc

	precision	recall	f1-score	support
Speed limit (20km/h)	0.97	1.00	0.98	58
Speed limit (30km/h)	0.99	0.96	0.97	745
Speed limit (50km/h)	0.99	0.98	0.99	758
Speed limit (60km/h)	0.94	0.95	0.95	441
Speed limit (70km/h)	0.95	0.99	0.97	635
Speed limit (80km/h)	0.98	0.80	0.88	771
End of speed limit (80km/h)	0.72	1.00	0.84	108
Speed limit (100km/h)	0.92	0.90	0.91	462
Speed limit (120km/h)	0.81	0.99	0.89	370
No passing	1.00	0.95	0.97	506
No passing for vechiles over 3.5 metric tons	0.98	0.99	0.98	656
Right-of-way at the next intersection	0.90	0.92	0.91	407
Priority road	0.95	1.00	0.97	655
Yield	1.00	0.98	0.99	736
Stop	0.99	1.00	0.99	267
No vechiles	1.00	0.89	0.94	237
Vechiles over 3.5 metric tons prohibited	0.95	1.00	0.97	142
No entry	1.00	1.00	1.00	360
General caution	0.81	0.97	0.88	325
Dangerous curve to the left	0.97	0.98	0.97	59
Dangerous curve to the right	0.83	0.95	0.89	79
Double curve	0.77	0.99	0.86	70
Bumpy road	0.95	0.98	0.97	116
Slippery road	1.00	0.68	0.81	220
Road narrows on the right	0.78	0.97	0.86	72
Road work	0.94	0.93	0.93	485
Traffic signals	0.84	0.87	0.86	175
Pedestrians	0.82	0.64	0.72	77
Children crossing	0.98	0.89	0.93	166
Bicycles crossing	0.99	0.91	0.95	98
Beware of ice/snow	0.65	0.70	0.67	141
Wild animals crossing	0.98	0.98	0.98	270
End of all speed and passing limits	1.00	0.98	0.99	61
Turn right ahead	0.98	1.00	0.99	205
Turn left ahead	0.99	0.97	0.98	123
Ahead only	0.99	0.99	0.99	390
Go straight or right	0.99	0.99	0.99	120
Go straight or left	0.97	1.00	0.98	58
Keep right	0.98	0.99	0.99	681
Keep left	0.96	0.99	0.97	87
Roundabout mandatory	0.91	0.94	0.93	87
End of no passing	0.82	0.92	0.87	53
End of no passing by vechiles over 3.5 metric tons	0.96	0.88	0.91	98
micro avg	0.95	0.95	0.95	12630
macro avg	0.93	0.94	0.93	12630
weighted avg	0.95	0.95	0.95	12630
samples avg	0.95	0.95	0.95	12630

Fig 11: Class-wise statistics

Results

The best way to summarize results is just not to display accuracy scores, but class-wise accuracy since it would better the understanding of classification. We also use precision, recall, f1 score and support rather than accuracy. Since these metrics provide a more nuanced understanding of the performance of the classifier.

Conclusion

We have trained various ResNet architectures and chosen the architecture that performs best. We implemented data augmentation methods to reduce the variance of the classifier. Moreover we eliminated the bias in classes of images.

In addition, we implemented our own architecture that generates the ROC score of 96 percent. Therefore, our final classifier is able to provide a reliable model for traffic sign classification.

References

[1]https://towardsdatascience.com/an-overview-of-resnet-and-its-variants-5281e2f56035
[2]https://www.kaggle.com/meowmeowmeowmeowmeow/gtsrb-german-traffic-sign
[3]https://keras.io/examples/cifar10_resnet/
[4]https://towardsdatascience.com/traffic-sign-detection-using-convolutional-neural-network-660fb32fe90e
[5]Zhe Zhu, Dun Liang, Songhai Zhang, Xiaolei Huang, Baoli Li, Shimin Hu; Traffic-Sign Detection and Classification in the Wild,; The IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2016, pp. 2110-2118
[6]R. Qian, Q. Liu, Y. Yue, F. Coenen and B. Zhang, "Road surface traffic sign detection with hybrid region proposal and fast R-CNN," 2016 12th International Conference on Natural Computation, Fuzzy Systems and Knowledge Discovery (ICNC-FSKD), Changsha, 2016, pp. 555-559, doi: 10.1109/FSKD.2016.7603233.