e-ISJN: A4372-3114
p-ISJN: A4372-3115

ISSN: 2321-7782 (Online)
ISSN: 2347-1778 (Print)
Impact Factor: 7.327

Volume 7, Issue 8, August 2019

International Journal of Advance Research in Computer Science and Management Studies

Research Article / Survey Paper / Case Study

Available online at: www.ijarcsms.com

Road Sign and Traffic Light Detection using Convolutional Neural Network and Keras Library

Rishabh Kumar Singh
Department of Information Technology
MITCOE
Pune – India.

Abstract: In countries where fatal accidents caused by human errors goes beyond 150k per year, system like Road Sign and Traffic Light detection and recognition plays a vital role in assisting drivers in automated driving. The system alerts driver about the road signs and traffic light by recognizing in advance. In this paper, an approach to tackle this issue based on Convolutional Neural Network has been established. This algorithm is best suited for image classification. This is due to the exceptional ability to extract points and distinctive features in images of each class. This makes use of convolution layers. This method is very efficient to use and produces high level of accuracy in terms of how it extracts features and make predictions with great precision. The results showed more than 99% accuracy on the testing dataset.

Keywords: Road Sign detection, Traffic Light Detection, Convolutional Neural Network, Machine Learning, Image Classification, Feature Extraction, Computer Vision, Advance Driver Assistance System(ADAS).

I. INTRODUCTION

In recent times all over the world, the most valuable information regarding road and its conditions faced by drivers are in terms of visual signals for example road signs boards posted and traffic light signals. These sign boards are a valuable part of the infrastructure which gives details about the current state, warnings, prohibitions, and speed limit restrictions for the road. This is an important information which human drivers tend to misread and often leads to road accidents and injuries to others. The drivers inattention and their ability to accidently or deliberately avoid road signs, drivers tiredness can easily be avo ided by using the road sign and traffic sign detection and recognition system. This system assists the driver to safely manoeuvre the vehicles on the road.

The Convolutional Neural Network is derived from perceptron model. It has changed the way we classify images. CNN or ConvNets is the deep learning architecture. They are most effective in facial recognition, object detection and various other applications. It has been empowering in self-driving cars as well. These networks are useful for analysing and classifying images because they are very effective at recognizing useful patterns within the images by understanding the spatial structure of the input relevant. Unlike traditional neural network which ignores the spatial structure such as pixel being closer together, pixels which are further apart, CNN are designed to use the spatial data for the benefit.

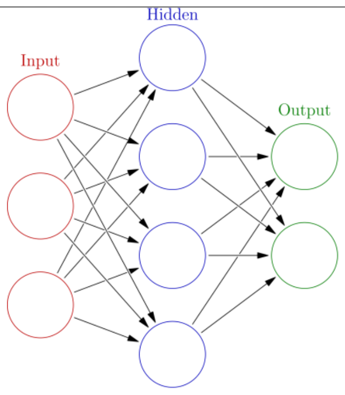


Figure 1 : Basic Perceptron Model

The CNN also required lower number of parameters when compared to artificial neural network. For example take a simple example of classifying a cat. The input image of cat goes through various convolutional layers and finally through the fully connected layer. Few similarities between the traditional and CNN is the input layer and fully connected layer which is essentially a multi-layer perceptron parameterised by weights and bias value that make use of the soft-max activation function in the output layer which after many convolutions outputs the abilities of the image belonging to some class. Appropriately, highest probability belongs to the image being a cat.

Convolutional Neural Network stems from the name Convolution which occurs in its network. It is known to process data that has a grid like structure. In case of gray-scale images, they consist of 2D array of pixels. Each pixel values ranges from 0 to 255 depending on the pixel intensity. Regular neural network often struggles with the complexity of the images required to compute image data.

Let’s imagine there is a RGB image data of 72 X 72 and since it is a 3 channel coloured image of depth 72 images, total pixel value corresponds to 15552. So, a neural network with for an input image would be much of a higher intensity image. This will lead to much higher complexity with the number of nodes and hidden layers increasing. There would be enough computational power to train the neural network. This leads to the use of convolutional neural network.

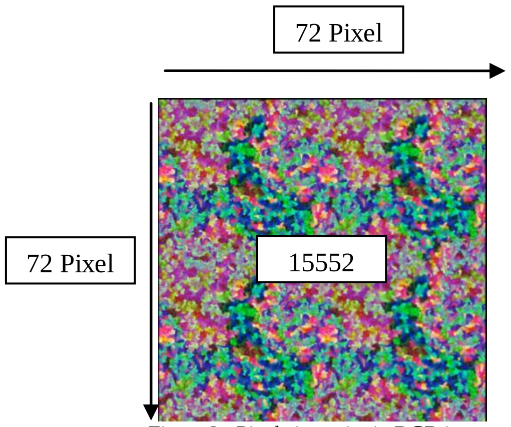


Figure 2 : Pixels intensity in RGB image

Another drawback to using regular neural network is the problem of over-fitting which occurs when model tailors itself very closely to the data that has been trained on reducing its ability to pin-point generalised features of test dataset that it has not seen.

Whereas in CNN, using pooling layers will continuously reduce the number of parameters in computations in the network.

Description

CNN is a specialised form of neural network. It consists of number of hidden layers which determines the depth and complexity of the neural network. Through some learning rate the network learns to adjust the weights of all the connections of the neuron and eventually come with a model with small training error. An input image feeds forward through the network and passes through all the layers and finally reaches the output layer where prediction is done based on the outputted probabilities.

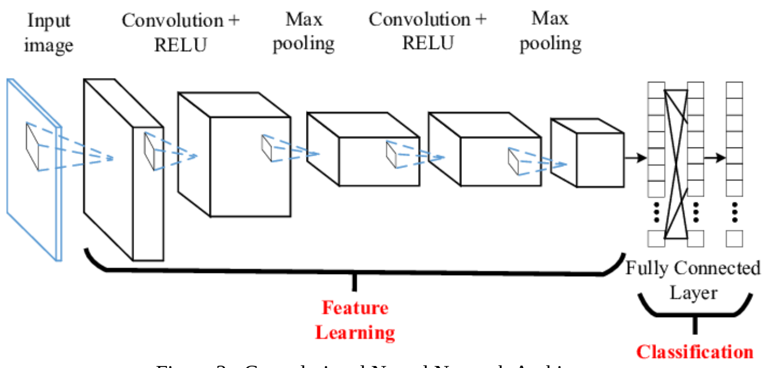
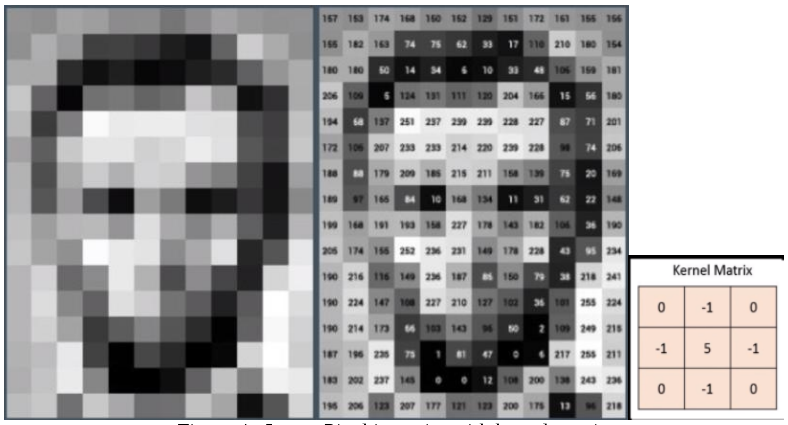


Figure 3 : Convolutional Neural Network Architecture

Convolutional Neural Network consists of three layers; convolutional, pooling and the fully connected layers. Made use of soft-max activation function to output predictions. Convolutional layers employs some form of operations called convolution, it is the main builing block of the network. Their primary goal is to extract and learn specific image features. Features that can be used to help classify the images. For example, I have an image in 2D scale with the pixel intensity values ranging from 0 to 255.



157	153	174	168	150	152	129	151	172	161	155	156
155	182	163	74	75	62	33	17	110	210	180	154
180	180	50	14	34	6	10	33	48	106	159	181
206	109	5	124	131	111	120	204	166	15	56	180
194	68	137	251	237	239	239	228	227	87	71	201
172	106	207	233	233	214	220	239	228	98	74	206
188	88	179	209	185	215	211	158	139	75	20	169
189	97	165	84	10	168	134	11	31	62	22	148
199	168	191	193	158	227	178	143	182	106	36	190
205	174	155	252	236	231	149	178	228	43	95	234
190	216	116	149	236	187	85	150	79	38	218	241
190	224	147	108	227	210	127	102	36	101	255	224
190	214	173	66	103	143	95	50	2	109	249	215
187	196	235	75	1	81	47	0	6	217	255	211
183	202	237	145	0	0	12	108	200	138	243	236
195	206	123	207	177	121	123	200	175	13	96	218

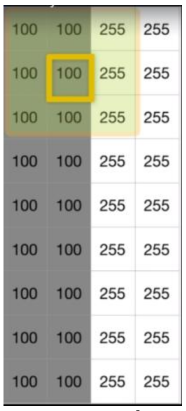
Kernel Matrix

0	-1	0
-1	5	-1
0	-1	0

Figure 4 : Image Pixel intensity with kernel matrix

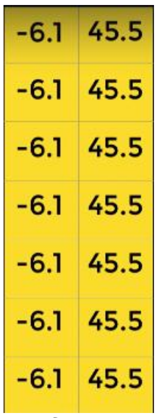
This entire image is an input and each pixel intensity is a node. All of these input images are processed by a weight filter node also known as kernel matrix. These filters are generally smaller in spatial dimensionality. This means applying kernel weight filter on images to create a feature map and this is termed as image feature extraction.

This is performed by sliding the kernel matrix onto the input image by a pre-defined step size. The amount by which the kernel is shifted in each operration is known as the stride. After this step, a feature map is created.



100	100	255	255
100	100	255	255
100	100	255	255
100	100	255	255
100	100	255	255
100	100	255	255
100	100	255	255
100	100	255	255
100	100	255	255

Figure 5 : Input image with Kernel Matrix



-6.1	45.5
-6.1	45.5
-6.1	45.5
-6.1	45.5
-6.1	45.5
-6.1	45.5
-6.1	45.5

Figure 6 : Feature Map

To compute feature map, take the pixel intensity and combine each pixel with corresponding kernel filter values. Take the sum of each and divide by total number of weight filter. For example :

100 * 0	100 * -1	255 * 0
100 * -1	100 * 5	255 * -1
100 0	100 * -1	255 * 0

Taking the sum of all the numbers and dividing :
0-100+0-100+500-255+0-100+0 = -55 / 9 = - 6.1

This way the entire feature of the image remains intact without losing any spatial feature and with reduction in dimensionality. The feature map contains specific feature of the image which was extracted from the original image. The kernel is often a detector which helps in creating a feature map. Accordingly, every kernel is designed to have distinct features which separates one kernel from another. Make use of gradient descent algorithm which minimizes the error. In this case, the error minimizes when the image is classified correctly.

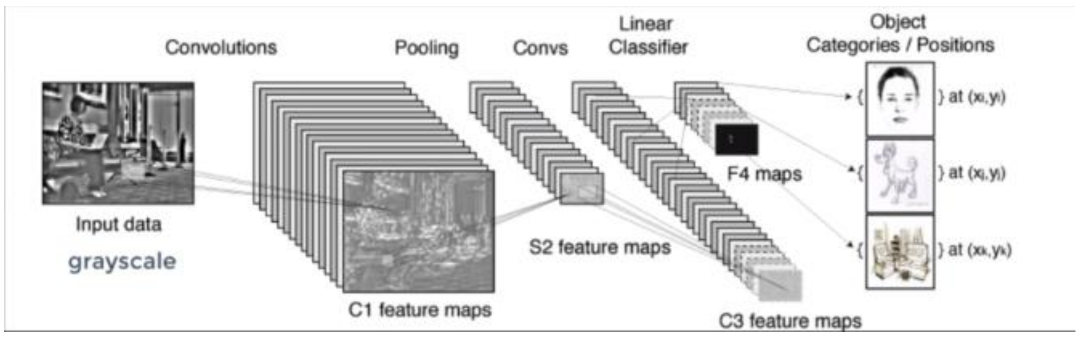


Figure 7 : Convolutional Neural Network steps

More the number of features, more is the depth of the convolution layer. Greater the number of features, better is the networks ability to classify the image. Different kernels can be used to predict different features in the images. In CNN, every node in the hidden layer is only connected to a very small region of the input volume. After generating the feature map, use Relu activation function. Any kind of neural network must have non-linearity since most of the real-world data on which predictions are required are all non-linear in nature. The operation of kernel and image sliding is linear, and is matrix multiplication operation is performed.

The pooling layer acts to shrink the image stack by reducing the dimensionality of the representation of each feature map. This reduces the computational complexity of the neural model. It retains the most important features of the image of interest. Pooling is done to avoid over-fitting. There are many different types of pooling layers like addition, average, max pooling layer. Max pooling layer is used to report the maximum output in the rectangular neighborhood i.e., specific a 2 X 2 kernel matrix and reduce the dimensionality such that it takes the maximum value in the neighborhood. This is done in the spatial dimension of the object. It ultimately scales down the feature map and retains only the maximum values. This also reduces the computational cost and complexity parameters in the images and helps to reduce over-fitting by providing the extracted form of the feature map. Even then it preserves the general features of the images. The maximum value of the filter in the max pooling is the maximum value in the image which describes the intensity of the image. Max pooling provides scale invariant of the image which is useful in detecting features in images. Pooling helps make the network remain unaffected by small translations in the input image or any distortion. This helps in generalizing features to the images. It provides a generalized extracted feature map.

A neural network can contain any number of features and can have any number of hidden layers. Depending upon the filters, it can have any number of feature map and further can have multiple convolution and pooling layer. Relu is applied to account for non-linearity and once again max pooling layer is applied to scale down the feature map. The deeper the network goes, more number of features are built on top of each other and thus more complex patterns can be encoded. In the being, account for low level features and later account for specific details.

CNN has two parts : extraction and classification. The classification determines based on the probability values of images in the training dataset. The output of the convolution and pooling layer is provided to the fully connected layer for processing. The output from the previous layer must be flattened into a 1D array of pixels to be feed into the fully connected layer of the network. The second part of the network is used for classification whereas the first is used to extract feature map. For the fully connected layer, the weight and bias values are adjusted to reduce the error based on gradient descent algorithm. Initially, random values are used for the kernel filtering and these weights are adjusted in order to minimize the error. The prediction values are compared with the labeled dataset to determine whether the prediction is correct or further adjustment are required. The overall error i.e., cross-entropy is calculated and if this value is large, weights and filters needs to be adjusted. This is done using back-propagation where gradient descent, which is the error.

Dataset Collection Process :

To collect real-world data, a high-definition (1080p) camera is mounted onto the vehicle. This camera had a field of view of 120 degrees with 20 frames per seconds. The vehicle is moved around the city during day and night time and high quality dataset is created for more than 25 different types of road signs and 3 types of traffic light. More than 50,000 images are collected over a period of time. The image dataset acted as training and testing set with more images being generated from these during the pre-processing of the images. Some of the road signs included left turn, right turn, stop sign ahead, speed limit sign boards, warning and prohibited sign boards. Each of the particular type consisted of more than 1800 images.

Pre-processing of Images :

The image dataset contains more than 25 classes of road signs and traffic light. The folder structure is such that there is a file name containing all the road signs and traffic sign names and folder containing all the images. These images are separated into train, validation and test set. Convert the RGB image into grey-scale image. For detecting the road signs, colour feature is not very significant. Most of the road signs are of the same colour and useful this is not very useful. The features that really matter are the edges, curves, shape and size which is the main focus. At the same time, the depth of the image is reduced from three to one channel. This makes the network more efficient and powerful in performing computations and helps to classify the data.

Furthermore, make use of histogram equalization method with an aim to standardize the lighting intensity in the images. Since, images has been collected at various time of the day and at night, the intensity varies greatly. To standardize the pixel intensity, made use of histogram equalization. After this method, all the images have the same lightning effect. The histogram intensity gives an idea about the pixel intensity. This also helps in higher contrast in the images which helps in image classification. This enhances the intensity in the image such that it is better distributed across image and at the same time enhancing image that occur at high intensity. It happens by flattening the resultant histogram grey values in the image. Next stage in the pre-processing is to normalise all the images to the same pixel level intensity.

Algorithm for Traffic Sign and Road Sign Detection using CNN and keras :

1. Import various python libraries.
2. Create a folder with training, testing dataset along with the labels.
3. Split out features and labels into training, validation and testing set.
4. Pre-process the images like converting to grayscale, augmenting images, normalising the image dataset.
5. Create image generator so that enough number of images is available to use.
6. Add convolution and max pooling layer and create multiple hidden layers. Flatten and dropout depending upon the number of features desired.

7. Use Adam optimiser and relu activation function to optimise the process.

8. To evaluate performance of the neural network, use evaluation metric accuracy score.

9. Accuracy score will define how good the CNN model is performing on the testing dataset.

Results :

The performance of a neural network can be obtained through the accuracy score and the optimal epoch rate of training and the validation dataset. The validation loss is higher as number of epoch increases.

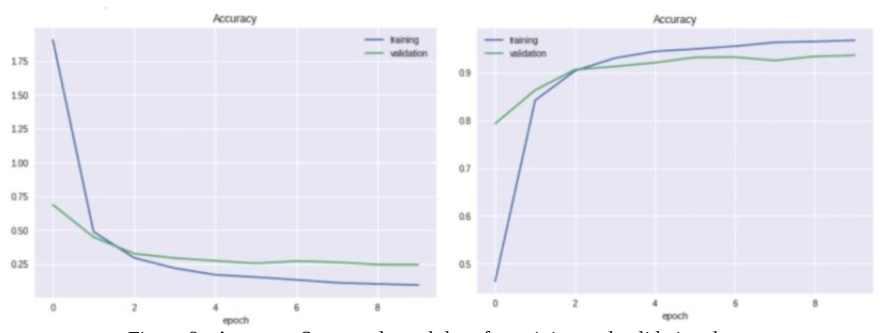
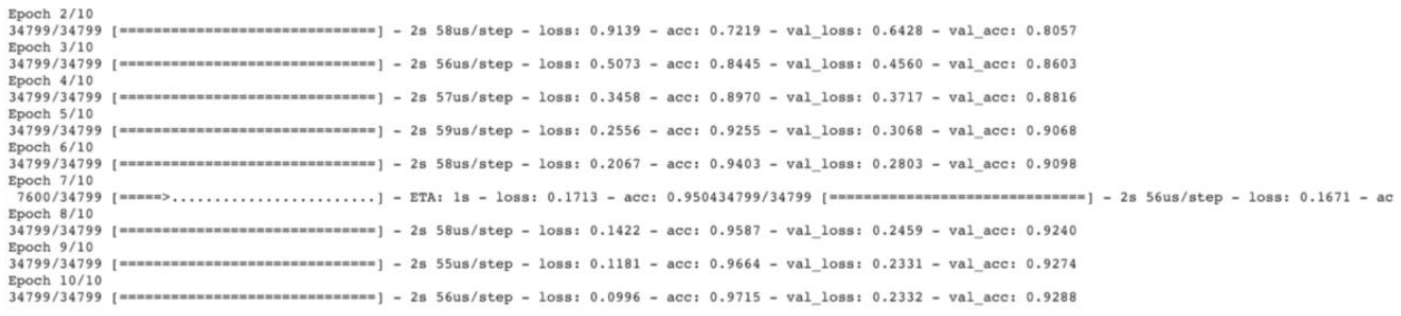


Figure 8 : Accuracy Score and epoch loss for training and validation dataset

After running the model on test data to check for accuracy, it shows that optimal results were not obtained and hence, fine tuning the model is required to improve upon the accuracy score.

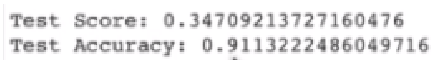
There were two main issues which needs to be dealt with; the accuracy and the over-fitting of the training data. Accuracy can be improved by variety of techniques and the modifications are case dependent. Fine tuning is reliant upon specific deep learning models. It is an important step and it can often improve the performance and accuracy significantly. To improve accuracy, altering the number of hidden layers and using dropouts, finding optimal learning rate contributes enormously.



```
Epoch 2/10
34799/34799 [==============================] - 2s 58us/step - loss: 0.9139 - acc: 0.7219 - val_loss: 0.6428 - val_acc: 0.8057
Epoch 3/10
34799/34799 [==============================] - 2s 56us/step - loss: 0.5073 - acc: 0.8445 - val_loss: 0.4560 - val_acc: 0.8603
Epoch 4/10
34799/34799 [==============================] - 2s 57us/step - loss: 0.3458 - acc: 0.8970 - val_loss: 0.3717 - val_acc: 0.8816
Epoch 5/10
34799/34799 [==============================] - 2s 59us/step - loss: 0.2556 - acc: 0.9255 - val_loss: 0.3068 - val_acc: 0.9068
Epoch 6/10
34799/34799 [==============================] - 2s 58us/step - loss: 0.2067 - acc: 0.9403 - val_loss: 0.2803 - val_acc: 0.9098
Epoch 7/10
 7600/34799 [=====>........................] - ETA: 1s - loss: 0.1713 - acc: 0.950434799/34799 [==============================] - 2s 56us/step - loss: 0.1671 - ac
Epoch 8/10
34799/34799 [==============================] - 2s 58us/step - loss: 0.1422 - acc: 0.9587 - val_loss: 0.2459 - val_acc: 0.9240
Epoch 9/10
34799/34799 [==============================] - 2s 55us/step - loss: 0.1181 - acc: 0.9664 - val_loss: 0.2331 - val_acc: 0.9274
Epoch 10/10
34799/34799 [==============================] - 2s 56us/step - loss: 0.0996 - acc: 0.9715 - val_loss: 0.2332 - val_acc: 0.9288
```

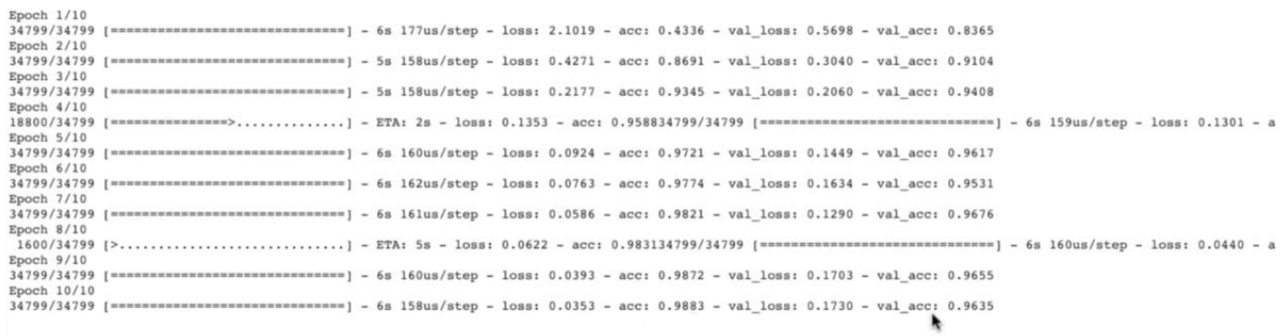
Making few alterations in the models as stated above lead to an accuracy score of 97 % and validation accuracy is 92 %. Therefore, this can concluded that these modifications improved our model accuracy. However, validation accuracy suffered quite a bit. However, these results still does not meet the standard level required to safely predict and classify the different types of sign boards.

The results obtained for test dataset was:



```
Test Score: 0.34709213727160476
Test Accuracy: 0.9113222486049716
```

This can further be improved by increasing the number of filters in the convolution layer. This can help the model extract more number of features and can improve the results.



```
Epoch 1/10
34799/34799 [==============================] - 6s 177us/step - loss: 2.1019 - acc: 0.4336 - val_loss: 0.5698 - val_acc: 0.8365
Epoch 2/10
34799/34799 [==============================] - 5s 158us/step - loss: 0.4271 - acc: 0.8691 - val_loss: 0.3040 - val_acc: 0.9104
Epoch 3/10
34799/34799 [==============================] - 5s 158us/step - loss: 0.2177 - acc: 0.9345 - val_loss: 0.2060 - val_acc: 0.9408
Epoch 4/10
18800/34799 [===============>..............] - ETA: 2s - loss: 0.1353 - acc: 0.958834799/34799 [==============================] - 6s 159us/step - loss: 0.1301 - a
Epoch 5/10
34799/34799 [==============================] - 6s 160us/step - loss: 0.0924 - acc: 0.9721 - val_loss: 0.1449 - val_acc: 0.9617
Epoch 6/10
34799/34799 [==============================] - 6s 162us/step - loss: 0.0763 - acc: 0.9774 - val_loss: 0.1634 - val_acc: 0.9531
Epoch 7/10
34799/34799 [==============================] - 6s 161us/step - loss: 0.0586 - acc: 0.9821 - val_loss: 0.1290 - val_acc: 0.9676
Epoch 8/10
 1600/34799 [>.............................] - ETA: 5s - loss: 0.0622 - acc: 0.983134799/34799 [==============================] - 6s 160us/step - loss: 0.0440 - a
Epoch 9/10
34799/34799 [==============================] - 6s 160us/step - loss: 0.0393 - acc: 0.9872 - val_loss: 0.1703 - val_acc: 0.9655
Epoch 10/10
34799/34799 [==============================] - 6s 158us/step - loss: 0.0353 - acc: 0.9883 - val_loss: 0.1730 - val_acc: 0.9635
```

The training and validation accuracy improved significantly as the score is well above 98% and 96% respectively.

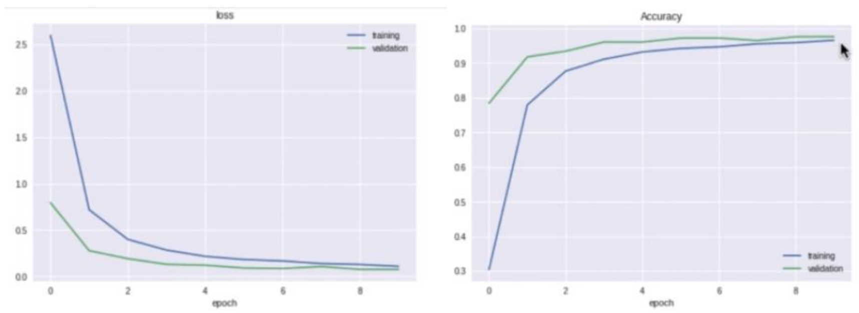
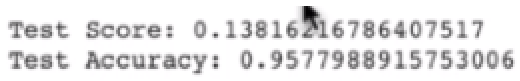


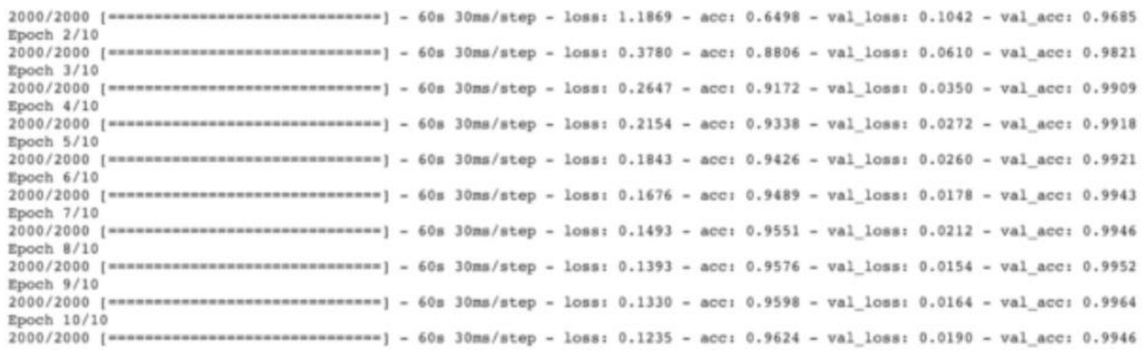
Figure 9 : Improved Accuracy Score for training and validation dataset

The test accuracy results showed an overall improvement.



```
Test Score: 0.13816216786407517
Test Accuracy: 0.9577988915753006
```

The test accuracy and test score can be further improved by using data augmentation technique. By using this method, additional dataset can be created for the training set. Altering the images in some way would give some additional images and this can improve the overall performance. Some of the ways could be to transform by rotating them along some axis (rotational), zooming into the images and a combination of both of these transformation. The new images are known as augmented images as these images allowed to augment the dataset. This allows the model to look at the dataset from a different perspective and extract relevant features from each image. Each image can be shifted horizontally, vertically, left, right and shear transformation. Shear is moving the images by some angle in the x axis and y-axis.



```
2000/2000 [==============================] - 60s 30ms/step - loss: 1.1869 - acc: 0.6498 - val_loss: 0.1042 - val_acc: 0.9685
Epoch 2/10
2000/2000 [==============================] - 60s 30ms/step - loss: 0.3780 - acc: 0.8806 - val_loss: 0.0610 - val_acc: 0.9821
Epoch 3/10
2000/2000 [==============================] - 60s 30ms/step - loss: 0.2647 - acc: 0.9172 - val_loss: 0.0350 - val_acc: 0.9909
Epoch 4/10
2000/2000 [==============================] - 60s 30ms/step - loss: 0.2154 - acc: 0.9338 - val_loss: 0.0272 - val_acc: 0.9918
Epoch 5/10
2000/2000 [==============================] - 60s 30ms/step - loss: 0.1843 - acc: 0.9426 - val_loss: 0.0260 - val_acc: 0.9921
Epoch 6/10
2000/2000 [==============================] - 60s 30ms/step - loss: 0.1676 - acc: 0.9489 - val_loss: 0.0178 - val_acc: 0.9943
Epoch 7/10
2000/2000 [==============================] - 60s 30ms/step - loss: 0.1493 - acc: 0.9551 - val_loss: 0.0212 - val_acc: 0.9946
Epoch 8/10
2000/2000 [==============================] - 60s 30ms/step - loss: 0.1393 - acc: 0.9576 - val_loss: 0.0154 - val_acc: 0.9952
Epoch 9/10
2000/2000 [==============================] - 60s 30ms/step - loss: 0.1330 - acc: 0.9598 - val_loss: 0.0164 - val_acc: 0.9964
Epoch 10/10
2000/2000 [==============================] - 60s 30ms/step - loss: 0.1235 - acc: 0.9624 - val_loss: 0.0190 - val_acc: 0.9946
```

The validation accuracy of the model improved significantly and reached up-to 99%.

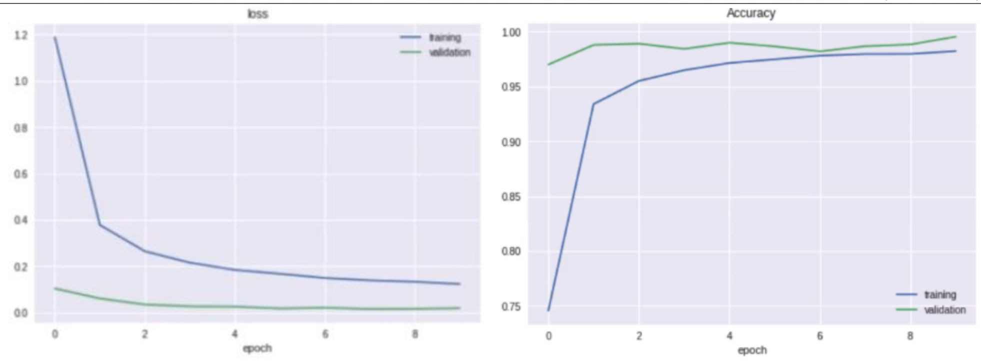


Figure 10 : Loss and Accuracy graph for training and validation dataset.

After these transformations, the final test result improved and showed an accuracy of more than 98%.

```
Test Score: 0.09644447928113152
Test Accuracy: 0.9790973872205896
```

This result gave an indication that model performed very well and can now be deploy on the hardware for making predictions in real-time.

II. Conclusion

Convolutional Neural Network for image feature extraction and classification proved to be significant game changer as compared to other regular neural network model. The test score and accuracy score achieved by using this model was in excess of 98% i.e., the model could predict the new images (test images) with an accuracy of more than 98%. This system can be used in autonomous vehicle which can reduce the number of accidents and assist drivers in maneuvering the vehicle safely. This can be introduced into the cars, trucks and other vehicles as a part of an Advance Driver Assistance System. By using a high definition camera and the software, this system will minimize the accidents rates significantly.

Acknowledgement

I would take this opportunity to thank professors, Head of the Department for their valuable guidance and for providing all the necessary facilities, which were indispensable in the completion of this research work. I am also thankful to all the staff members of the Department of Information Technology of MIT College of Engineering, Pune for their valuable time, support, comments, suggestions and persuasion. I would also like to thank the institute for providing the required facilities, Internet access and important books. I would also thank my parents and sister for always encouraging me to follow me heart and make a difference through the knowledge I have acquired over the years.

References

1. Practical Convolutional Neural Networks by Mohit Sewak, Md. Rezaul Karim, Pradeep Pujari Published February 2018.
2. Hands-on Machine Leaning with Scikit-Learn, Keras, and Tensorflow, 2nd Edition by Aurelien Geron Published September 2019.
3. Understanding Convolutional Neural Network (CNNs) by Neil Watson Published July 2017.
4. Python Machine Leaning : Perform Python Machine Learning and Deep Learning with Python, scikit-learn and tensorflow by Sebastian Raschka, Vahid Mirjalili, 2nd Edition Published September 2017.
5. Object Detection and Recognition using Deep Learning in OpenCV by Param Uttarwar Published April 2018.

Author(s) Profile

Rishabh Kumar Singh, received the B.E degree in Information Technology from Maharashtra Institute of Technology, College of Engineering in 2014. During 2014-2019, he stayed with MNC helping various teams in developing various machine learning algorithms and providing ADAS applications for autonomous vehicles. His specialization in computer vision has largely helped MNCs in providing solutions to their clients. He now works with Softchoice – an IT solution provider firm in the data science domain helping them optimizes their daily workflow.