WWW.IJRPR.COM

International Journal of Research Publication and Reviews

Journal homepage: www.ijrpr.com ISSN 2582-7421

Usage of CNN for Classification of Traffic Signs

A.Satyanarayana[1], Mrs. Jaishri P.W.[2]

[1]Associate Professor,Siddhartha Institute of Technology and Sciences,India
2Asstt Professor,Siddhartha Institute of Technology and Sciences,India

ABSTRACT

Traffic sign classification is an important task in autonomous driving and assistant driving systems. In this thesis we do automatic learning of features and classification on traffic signs from images. First, we study several publicly available libraries for deep learning. Several CNN architectures are then tested under different parameter settings and scenarios, such as network depth, filter size, dropout rate and pre-processing by using original images and segmented images. The Traffic Sign Recognition Benchmark was used to train in a supervised way the CNN model. Pre-processing and segmentation are tested to make the training more robust and the network able to generate more independent features. The results obtained are good for all study cases and all 43 traffic sign classes. We reached test accuracies above 98% that are comparable to state-of-the-art performances.

Keywords: Convolutional Neural Networks (CNN), Traffic Sign Classification, Supervised Learning, German Traffic Sign Recognition Benchmark (GTSRB), Advanced Driver Assistance Systems (ADAS)

1 Introduction

Traffic signs have been designed to be easily readable for humans, who perform very well at this task. For computer systems, however, classifying traffic signs still seems a challenging task . Checking the road ahead, what is behind and the traffic, all while trying to maintain the speed may sometimes become quite difficult. Car firms are constantly looking to introduce new technologies to make this taskeasier and the development of Traffic Sign Recognition (TSR) systems can be considered a challenging real-world problem of high industrial relevance nowadays. Its purpose is to support the driver so that the chances of not noticing a change in speed limit or the warning of a potential hazard ahead shall be vastly reduced. TSR is also one of the foremost important integral parts of autonomous vehicles and Advanced Driver Assistance Systems (ADAS).

However, due to their underlying mechanism of binary classification, these methods have to face the unbalance between the number of positive and negative training samples. As a result, these methods are likely to achieve a local optimum or an over-fitting solution. In recent work, convolutional neural networks (CNNs) have been used to automatically learn feature representations of traffic signs. These DNN algorithms combine feature extraction and classification into a unified neural network. CNN methods are frequently considered using hand-crafted features such as a circle detector, and a histogram of oriented gradient or scale-invariant feature transform.

However, designing such features requires a good deal of time and it is hard to know what feature is robust to a specific task. Traffic signs may be divided into different categories according to function, and in each category, they may be further divided into subclasses with similar generic shape and appearance but different details. This suggests traffic-sign recognition should be carried out as a two-phase task: detection followed by classification. The detection step uses shared information to suggest bounding boxes that may contain traffic-signs in a specific category, while the classification step uses differences to determine which specific kind of sign is present. Current methods achieve near perfect results for both tasks, with 100% recall and precision for detection and 99.65% precision for classification. in particular train an ensemble of CNNs and do classification by averaging the decisions of the ensemble.

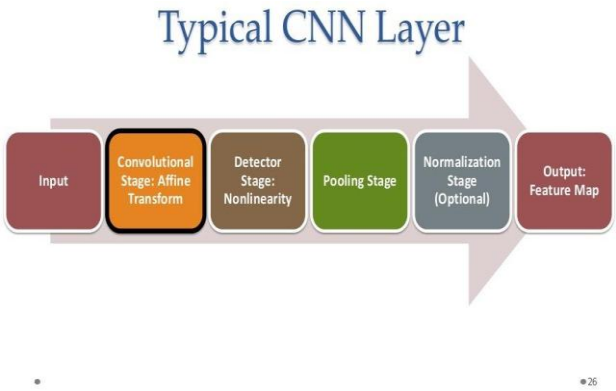


Fig.Typical cnn layer

2 Related work

In this paper we have done a definite Ns-2 based reenactment to contemplate and break down the exhibition differentials of DSDV and DSR in the half breed situation under shifting hub speed with various number of sources. Our work is the first trying to look at these conventions in mixture organizing climate. From the reenactment results we see that at lower hub speed, DSDV shows preferred parcel conveyance execution over DSR predominantly because of the moment accessibility of fresher and more current courses constantly. Then again, with higher hub speed, DSDV shows more disintegration in the parcel conveyance execution than DSR primarily because of its less flexibility to the exceptionally powerful organization geography. DSR's better presentation is ascribed to its capacity to keep up various courses per objective and its utilization of forceful storing system. Regarding the normal start to finish delay, DSDV beats DSR. The lackluster showing of DSR as far as normal start to finish delay is principally because of its source steering.

3 CNN

CNN is one of the neural network models for deep learning, which can be described by three specific characteristics, namely locally connected neurons, shared weights and spatial or temporal sub-sampling. The architecture of a CNN is designed to take advantage of the 2D structure of an input image (or other 2D input such as a speech signal). This is achieved with local connections and tied weights followed by some form of pooling which results in translation invariant features. A benefit of CNNs is that they are easier to train and have many fewer parameters than fullyconnected networks with the same number of hidden units. To provide an overview on the architecture characteristics, CNNs are composed of convolutional andsubsampling layers optionally followed by fully connected layers. The input to a convolutional layer is an m x m x r image where m is the height and width of the image and r is the number of channels, e.g., an RGB image has $r = 3$. The convolutional layer will have k filters of size n x n x q where n is smaller than the dimension of the image and q can either be the same as the number of channels r or smaller and may vary for each kernel. The size of the filters gives rise to the locally connected structure where each element is convolved with the image to produce k feature maps of size $mn + 1$. In convolutional layer, each neuron is connected locally to its inputs of the previous layer, which functions like a 2D convolution with a certain filter, then its activation could be computed as the result of a non linear transformation:

$$\alpha_{i,j} = \rho(f \quad x) \quad =$$

$$\rho(f_{i^J,j^J} x_{i+i^J,j+j^J} + b)$$

$$N_{out} = \frac{N_{in} - F}{Stride} + 1$$

$$i^J=1\, j^J=1$$

where f is an n x n weight matrix of the convolutional filter, x refers to the activations of the input neurons connected to the neurons (i,j) in the following convolutional layer.

$$P = \frac{F-1}{2}$$

ρ() is a nonlinear activation function (usually sigmoid or hyperbolic tangent), b is the bias, is the convolution operator. After the convolution each map may be sub sampled typically with mean or max pooling over pxp contiguous regions where p ranges between 2 for small images (e.g. MNIST) and is usually not more than 5 for larger inputs.

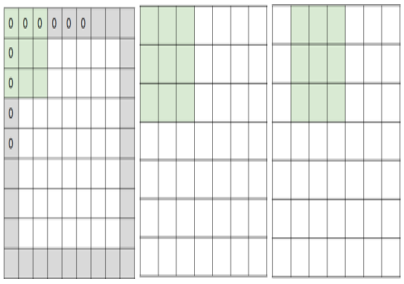


Figure : On the left: an example of zero-padding with 1-pixel border. Middle and right: areas superimposed on subsequent multiplications during stride-1 convolution.

One aspect of CNNs to pay attention to is to preserve the size of the data throughout the network which would otherwise decrease without zero padding. Without zero- padding the size of the output Nout is:

For this reason, having stride 1, to preserve the size of the input data past the convolutional layer, zero-pad P is set according to:

$$P = \frac{F-1}{2}$$

$$N_{out} = \frac{N_{in} - F}{Stride} + 1$$

Residual Neural Network (ResNet):

A variation of standard CNNs are Deep Residual Neural Networks. A description and a model for ResNets. ResNets take a standard feed-forward ConvNet and add skip connections that bypass (or shortcut) a few convolution layers at a time. Each bypass gives rise to a residual block in which the convolution layers predict a residual that isadded to the following block's input. These networks can gain accuracy from considerably increased depth. A residual network 8 times deeper than VGG net but still having lower complexity compared to it.

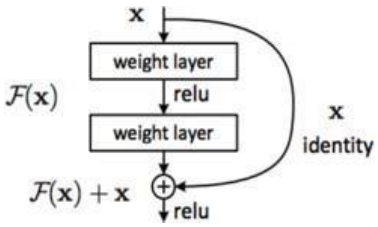


Figure: RNN

Residual layers address a problem of degradation that occurs as the depth increases andthis problem is not due to overfitting. The accuracy saturates first and then degrades rapidly. When extra-layers turn out to be unnecessary or evencause of degradation, an identity mapping between layers is the most effective solution. A residual connection allows to push the residual to zero and propagate along the stack an identity mapping, thus "hiding" the presence of extra (unnecessary) layers. To the extreme, if an identity mapping were optimal to a specific training propagation, it would be easier to push theresidual to zero than to fit an identity mapping by a stack of nonlinear layers. Along with the depth degradation, the internal covariate shift issue has been considered with ResNets. The internal covariate shift is caused by the fact that the distribution of each layer's inputs changes during training along with the change in the network parameter's values. As a consequence the training is a slower because it forces lower learning rates and makes it hard to train networks with saturating nonlinearities. Batch normalization fixes the distribution of the layer inputs as the training progresses.Since mini-batches are used in stochastic

gradient training, each mini batch produces estimates of the mean and variance of each activation. The statistics used in normalization then can be thought as fully participating to the gradient backpropagation. On top of that, a scale and shift transformation of type:

$$y_{k+1} = y_k * norm(x) + \beta_k$$

is being added to every residual learning block. The reason is that since full whitening of each input layer is costly, the simplification of normalizing each feature in- dependently is made. This will though alter what a layer represents. Scaling and shifting transformations allow the network to be able to represent the identity trans- formation.

Recurrent Neural Network (RNN):

The idea behind RNNs is to make use of sequential information. In a traditional neural network, we assume that all inputs (and outputs) are independent of each other. But for many tasks that may be a bad idea. To predict the next word in a sentence it is better to know which words came before it. RNNs are called recurrent because they perform the same task for every element of a sequence, with the output made dependent on the previous computations. Another way to think about RNNs is that they have a "memory" which captures information about what has been calculated so far. In theory RNNs can make use of information in arbitrarily long sequences, but in practice they are limited to looking back only a few steps. In Figure can be seen the basic structure of RNN:

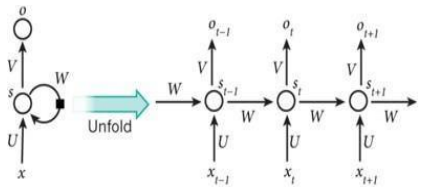


Fig:Basic structure of RNN

x_t is the input at time step t. For example, x_1 could be a vector corresponding to the second frame of a video. s_t is the hidden state at time step t. It is the "memory" of the network. s_t is calculated based on the previous hidden state and the input at the current step: $s_t = f(U_{xt} + W_{st-1})$. The function f usually is a nonlinearity such as tanh or a rectifier. s_{t-1}, s_{t-n}, which are required to calculate the first hidden state, are typically initialized to all zeroes. o_t is the output at step t. For example, to make prediction on the next frame in a video it would be a vector of probabilities across all possible pixels (the descriptors of the frame) in use in the given system. The literature around RNN is very wide. In Bi-directional RNN, training is accomplished by forwarding the data simultaneously in positive and negative time direction. Multi-dimensional RNN, introduced in, extend RNNs one-dimensional input data to multi-dimensional data, thereby extending the potential applicability of RNNs to vision, video processing, medical imaging and many other areas. Gated Feedback Recurrent Neural Networks (GFRNN) extends the existing approach of stacking multiple recurrent layers by allowing and controlling signals flowing from upper recurrent layers to lower layers.

Long Short-Term Memory (LSTM):

One of the appeals of RNNs is the idea that they might be able to connect previous information to the present task, such as using previous video frames might inform the understanding of the present frame. RNNs can basically learn to use the past information. But there are also cases where more context is needed.

These issues are the main motivation behind the LSTM model, introduced in 1997. This model makes use of a new structure called a memory cell. A memory cell is composed of four main elements: an input gate, an output gate, a neuron with a connection to itself and a forget gate. The self-recurrent connection has a weight of 1.0 and ensures that, barring any outside interference, the state of a memory cell can remain constant from one time step to another. The gates serve to modulate the interactions between the memory cell itself and the environment. The input gate can allow incoming signal to alter the state of the memory cell or block it. On the other hand, the output gate can allow the state of the memory cell to have an effect on other neurons or prevent it. The forget gate can modulate the memory cell's self-recurrent connection, allowing the cell to remember or forget its previous state, as needed.

4 Implementation

For this project, the running environment which we used is a cloud based running environment. So, we used Google Chrome Browser within which we used Kaggle Kernel, a cloud based running environment. We also imported our datasets from Kaggle.

For our Convolutional Neural Network, the datasets we used are German Traffic Signs Recognition Benchmark (GTSRB). We made sure all these datasets i.e. image files are of theformat .jpg, .jpeg, .png etc.

We imported Shuffle method from the sklearn's utility class i.e. (sklearn.util) to shuffle all the images in the dataset. Now the Python Imaging Library (PIL) is used to pixelize all theimages to a same size.

An image is taken from the dataset and each pixel of that image is given as an input to each and every node of the input layer of our CNN (model). From the input layer, the pixels are selected and sent to the next layers (Hidden Layers). Our model uses weights for connecting the nodes of two different layers for forward and backward propagation. For these mathematical functions are used i.e. to manipulate the weight(s) and activate the next node(s).Hence, these are called activation functions.

We used Numpy Library for these activation functions to work. Some of the activation functions which we used in our project are ReLu, Sigmoid, Softmax, Tanh etc.

Maxpooling2D and Averagepooling2D are also used in addition for activating the nodes indifferent layers of the model.

While training the model we used Conv2D to wrap the inputs of the hidden layers and produce a tensor of outputs. To accelerate the training process, we used Batch Normalizationtechnique and to prevent our model from getting overfitted we used Dropout technique.

During the training phase i.e. with all the front and back propagations going on in the model we find a regular deeply connected Neural Network layer which is the most common and frequentlyused layer. This is called as Dense Layer.

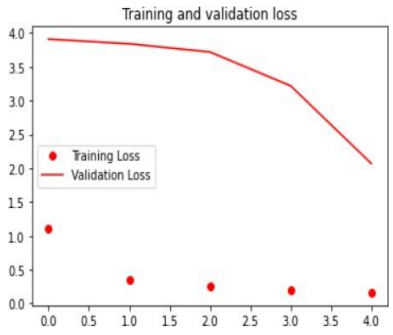
For converting the pooled feature maps into single columns so that they can be passes to a fully connected layer, we used the Flatten function. All these functions and techniques are from the Keras Library's Layer class i.e. (keras.layers) We used Sequential model imported from keras.model so as to have exactly one input tensorand one output tensor at a time.

To handle all the sparse gradients on noisy problems we used Adam (Keras Optimizer) and used Tensorboard, a visualization tool which calls back log events. It helps in visualizing thetraining graph.

(i) Accuracy vs Validation Accuracy

(ii) Loss vs Validation Loss of all the activation functions for study case.

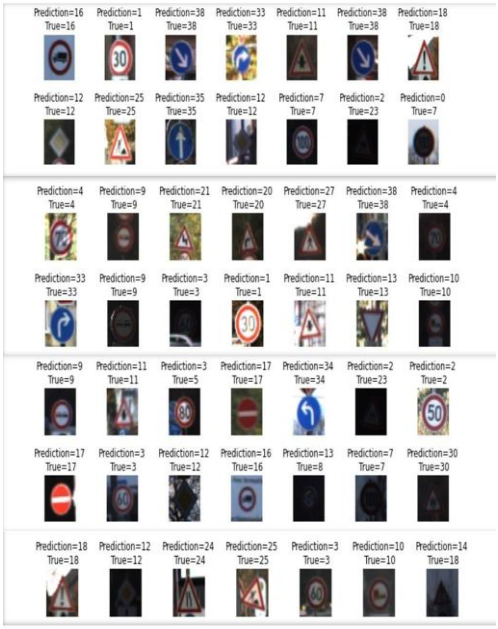
To get all the indices values to be plotted we used Argmax class from the Numpy library.For creating a Confusion Matrix we used

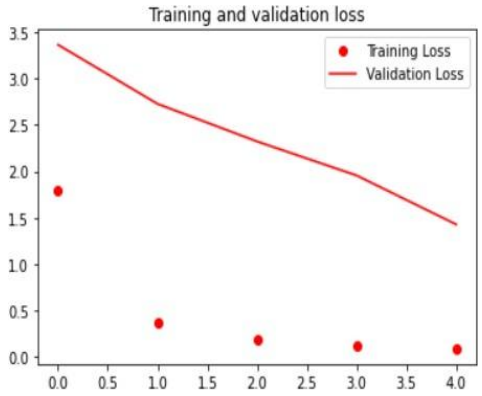


Metrics class from the Sklearn library.

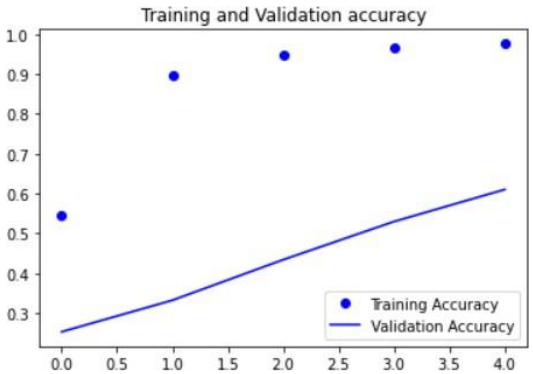
We used Seaborn, a data visualization library for creating a heat map and integrating it to theconfusion matrix.

Tanh Function

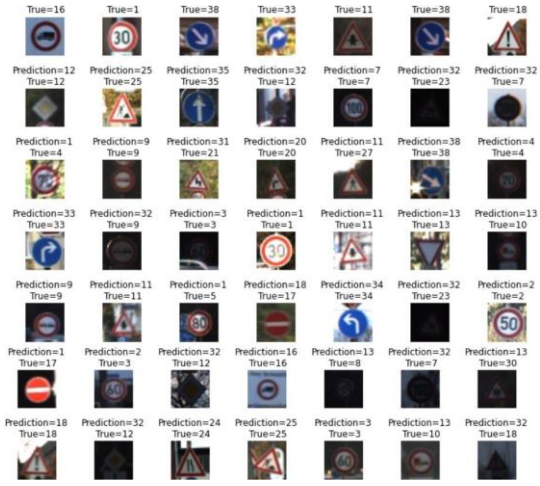


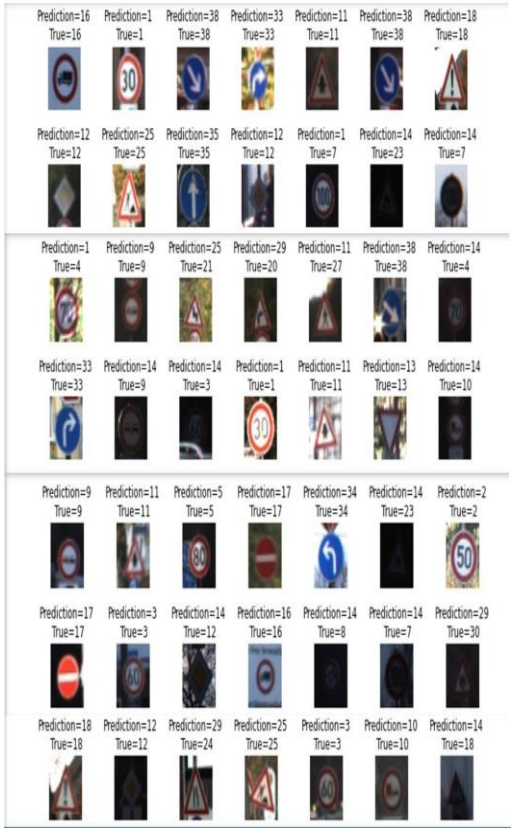


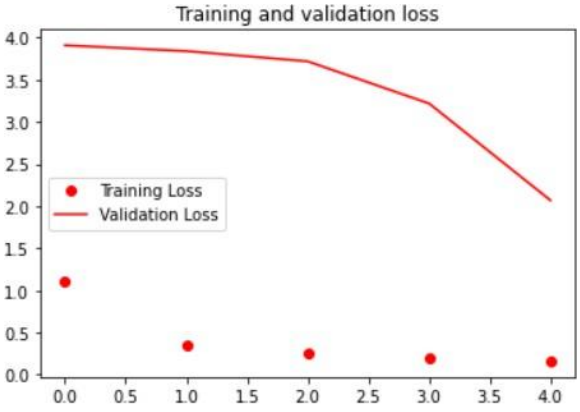
ReLu Function:

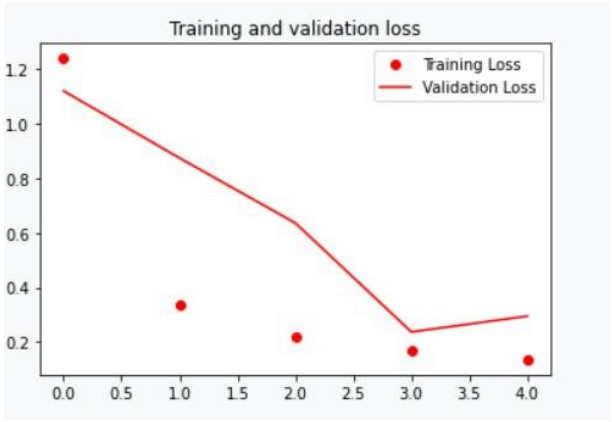


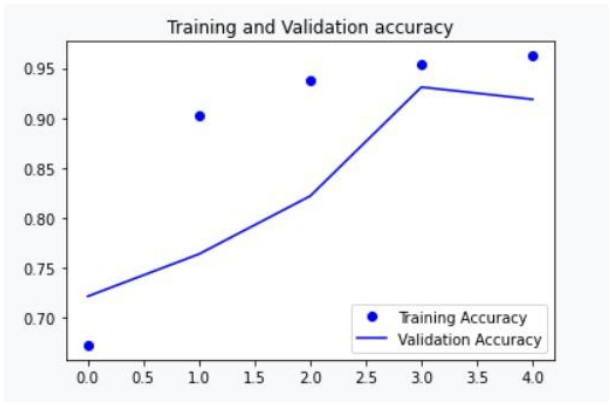
Sigmoid Function

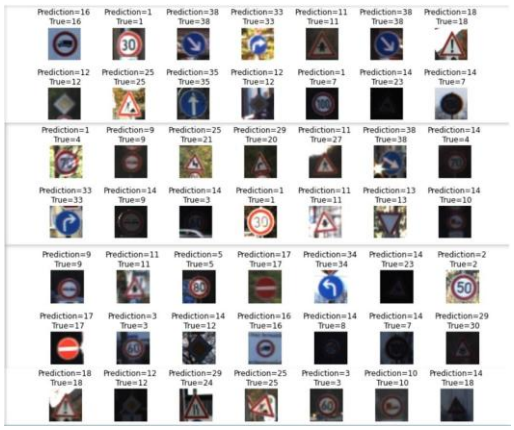




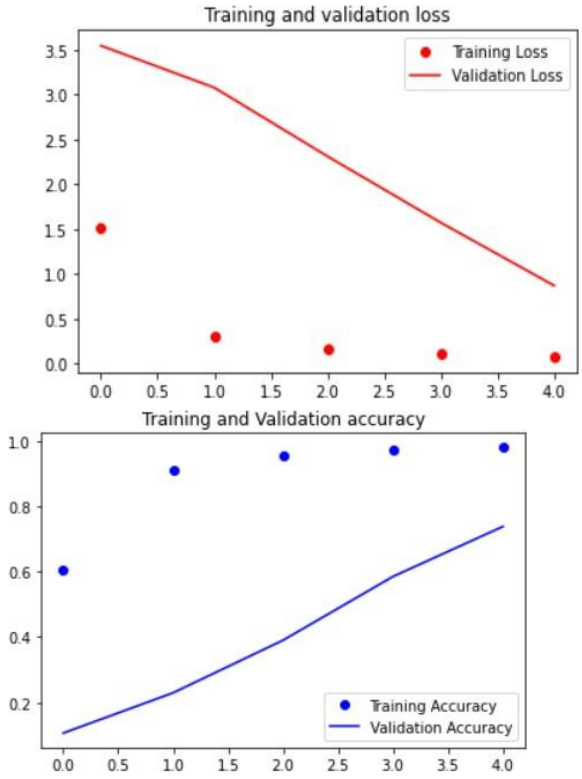

Training and validation loss
Training Loss
Validation Loss


Training and validation loss
Training Loss
Validation Loss


Training and Validation accuracy
Training Accuracy
Validation Accuracy



Swish Function:



5 CONCLUSION

This project proposes a deep convolutional network with a fewer number of parametersand memory requirements in comparisons to existing models. In this project several convolutional networks with different depth and type of layerswere trained and compared. One model was chosen and used to explore in more details the impact of different preprocessing normalizations.

In future , using a well-developed model could save lives in the future automation world. Conducting various researches on this, would be useful for the model to attain high accuracy due to which the road fatalities could be highly minimized.

REFERENCES

1]Li Deng, “Three Classes of Deep Learning Architectures and Their Applica-tions: A Tutorial Survey”, 2012.
2]Braunagel, C., Kasneci, E., Stolzmann, W., Rosenstiel, W. “Driver-activity recognition in the context of conditionally autonomous driving.” In 2015 IEEE18th International Conference on Intelligent Transportation Systems (pp. 1652- 1657) IEEE, 2015.
3]Smolensky Paul, 1986. "Chapter 6: Information Processing in Dynamical Sys- tems: Foundations of Harmony Theory".
4]J. Bergstra, O. Breuleux, F. Bastien, P. Lamblin, R. Pascanu, G. Desjardins, J. Turian,
5]D. Warde-Farley and Y. Bengio. June 30 - July 3, Austin, TX (BibTeX). “Theano: ACPU and GPU Math Expression Compiler”.
6]www.journal.frontiersin.org/article/10.3389/frobt.2015.00028/fullh4.
7]Y. LeCun and M.A. Ranzato, ICML 2013 tutorial.
8]www.cs.utexas.edu
9]G.E. Hinton and R.R. Salakhutdinov, "Reducing the Dimensionality of Data with Neural Networks", Science, 28 July 2006, Vol. 313. no. 5786, pp. 504 - 507.
10]www.wildml.com
11]http://deeplearning4j.org/restrictedboltzmannmachine.html
12]Ballas et al., “Delving Deeper into Convolutional Networks for Learning VideoRepresentations”, 2016.
13]http://deeplearning.net/tutorial/lstm.html.
14]http://benchmark.ini.rub.de/?section=homesubsection=news
15]https://searchcode.com/codesearch/view/15851013/