INTERNATIONAL JOURNAL OF ADVANCE RESEARCH, IDEAS AND INNOVATIONS IN TECHNOLOGY

ISSN: 2454-132X
Impact factor: 4.295
(Volume 4, Issue 2)
Available online at: www.ijariit.com

Traffic sign detection and recognition for autonomous vehicles

Mohit Singh
mohitsingh15021995@gmail.com
IMS Engineering College, Ghaziabad, Uttar Pradesh

Manish Kumar Pandey
manishpandey6999@gmail.com
IMS Engineering College, Ghaziabad, Uttar Pradesh

Lakshya Malik
lxy.malik@gmail.com
IMS Engineering College, Ghaziabad, Uttar Pradesh

ABSTRACT

Traffic sign recognition and detection is an important part of any autonomous vehicle. However, the real challenge lies in the detection and recognition of these traffic sign from the natural image in real time and with accuracy. This paper gives an overview of the traffic road sign detection and recognition system, we developed and implemented using an artificial neural network which is trained using real-life datasets. This paper presents the usage of convolution neural network along with GTSRB dataset as an implementation of our project to attain real-time result with accuracy. The system developed based on this methodology can be implemented in public transports, personal cars, and other vehicles in order to keep drivers alert and reduce human errors that lead to accidents. The project has a wide implementation of self-driving vehicles.

Keywords: *Convolution neural networks, GTSRB, Traffic signs, Recognition, Detection.*

1. INTRODUCTION

Nowadays, there is a lot of attention being given to the ability of the car to drive itself. One of the many important aspects for a self-driving car is the ability for it to detect traffic signs in order to provide safety and security for the people not only inside the car but also outside of it. The traffic environment consists of different aspects whose main purpose is to regulate the flow of traffic, make sure each driver is adhering to the rules so as to provide a safe and secure environment to all the parties concerned. We have focused our project on the German traffic signs and a few of the traffic signs which we have in our dataset is as shown in the figure below. We used the German traffic sign dataset. The dataset consisted of 43 different types of German traffic sign. The problem we are trying to solve has some advantages such as traffic signs being unique thereby resulting in object variations being small and traffic signs are clearly visible to the driver/system. The other side of the coin is that we have to contend with lighting and weather conditions. The main objective of our project is to design and construct a computer-based system which can automatically detect the road signs so as to provide assistance to the user or the machine so that they can take appropriate actions. The proposed approach consists of building a model using convolutional neural networks by extracting traffic signs from an image using color information. We have used convolutional neural networks (CNN) to classify the traffic signs and we used color based segmentation to extract/crop signs from images.

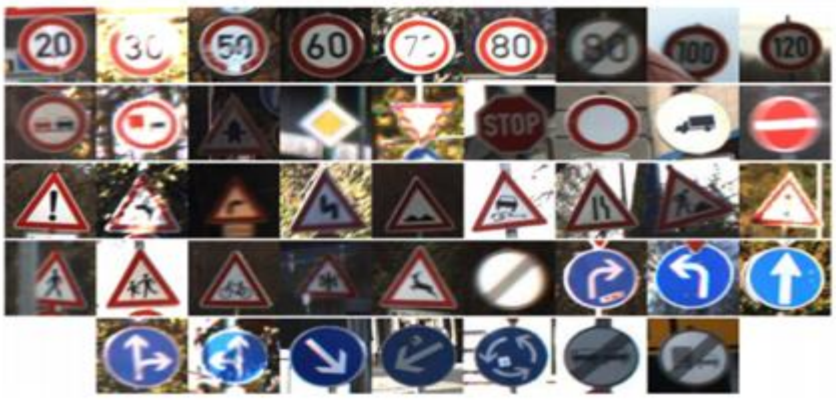

Fig 1: Different traffic signs

2. METHODOLOGY

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 (if any). (We note that the words 'detection' and 'classification' have different meanings in the general object recognition community where, as exemplified by the Image Net competition, classification means giving an image a label rather than an object, and detection means finding the bounding box of an object in a specific category.)

A. GTSRB Data Set

The German Traffic Sign Recognition Benchmark (GTSRB) is a publicly available data set containing 51839 images of German road signs, divided into 43 classes. The data set was published during a competition held at the 2011 International Joint Conference on Neural Networks (IJCNN). Images in the data set exhibit wide variations in terms of shape and colour among some classes, as well as strong similarities among others (e.g. different speed limit signs). The data pose several challenges to classification, including varying lighting and weather conditions, motion-blur, viewpoint variations, partial occlusions, physical damage and other real-world variabilities (some samples considered difficult to classify. Furthermore, resolution is not homogeneous and as low as 15×15 pixels for some images. For these reasons, human performance on this data set is not perfect, and estimated at around 98.84% on average. An additional challenge in training classification algorithms on the GTSRB data is that the 43 classes are not equally represented. The relative frequency of classes 0, 19, and 37, for example, is only 0.56%, significantly lower than the mean 1/43 = 2.33%. Moreover, since the data set was created by extracting images from video footage, it contains a track of 30 images for each unique physical real-world traffic sign instance. As a result, images from the same track are heavily correlated.

B. Convolutional Neural Network

The convolutional neural networks (CNN or ConvNet) are a class of deep and feed forward artificial neural network that has successfully been applied to analyzing visual imagery. CNN's use a variation of multilayer perceptrons designed to require minimal preprocessing. A CNN consists of an input and an output layer, as well as multiple hidden layers which consist of convolutional layers, pooling layers, fully connected layers and normalization layers. Here in our project, we define a convolutional neural network which is then later trained with the GTSRB training dataset. During the training, the CNN use to learn and classify data based on given classes. A CNN architecture is formed by a stack of distinct layers that transform the input volume into an output volume through a differential function. A few distinct types of layers commonly used are:

1) Convlution Layer: The convolution layer is the core building block of CNN. The layers' parameters consist of a set of learnable filters, which have a small receptive field. But extend through the full depth of input volume. During the forward pass, each filter is convolved across width and height of the input volume, computing the dot product between the entries of the filter and the input and producing a 2-dimensional activation map of that filter. As a result, the network learns filters that activate when it detects some specific type of feature at some spatial position in the input. Stacking the activation maps for all filters along the depth dimensions form the full stack output volume of the convolution layer.

2) Pooling Layer: Pooling in CNN is a form of down-sampling. There are several non- linear functions to implement pooling among which max pooling is most common. It partitions the input image into a set of non-overlapping rectangles and, for each sub-region, outputs the maximum. The intuition is that the exact location of a feature is less important than its rough location relative to other features. The pooling layer serves to progressively reduce the spatial size of the representation, to reduce the number of parameters and amount of computation in the network, and hence to also control overfitting. It is common to periodically insert a pooling layer between successive convolutional layers in a CNN architecture. The pooling layer operates independently on every depth slice of the input and resizes it spatially. The most common form is a pooling layer with filters of size 2x2 applied with a stride of 2 downsamples at every depth slice in the input by 2 along both width and height, discarding 75% of the activations. In this case, every max operation is over 4 numbers. The depth dimension remains unchanged.

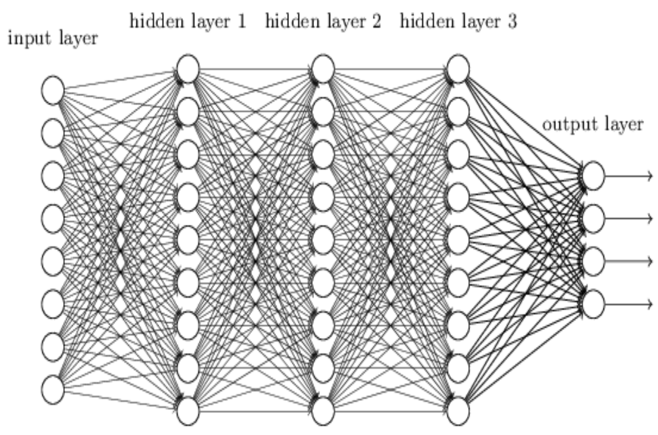

Fig 2: convolutional neural networks (CNN)

3) ReLU layer: ReLU is the abbrevation of Rectified Linear Units. This layer applies the non-saturating function f(x)=max(0,x). It increases the nonlinear properties of the decision function and of the overall network without affecting the receptive fields of the convolutional layer.

4) Fully connected layer: The high-level reasoning in a neural network is done via fully connected layers, after several convolutional and max pooling layers. Neurons in a fully connected layer have connections to all activations in the previous layer. Their activation can hence be computed with a matrix multiplication followed by a bias offset.

We here in our project of creating a system of traffic sign detection and recognition and alerting the driver regarding the same have used the above dataset to train the convolutional neural networks and test them for accuracy and durability

For each classifier, the training stage can be described in the following steps:

Step 1. Data collection. Two corresponding classes are acquired based on their labels.

Step 2. Data processing. In order to measure the similarities of MPPs (Max pooling positions) belonging to a same class, all of the MPPs sequences come from the same class are accumulated together and normalized by dividing the number of samples. So that we get a series of sequences that indicate the probability of appearance for each of the max value positions. Therefore, two probability sequences are achieved, namely, p1 and p2.

Step 3. Activation selection. Based on the values of the two probability sequences p1 and p2, the highest n (n = 5, 10..., 4000) channels will be activated. To be more specific, features of each class will be selected and reduced to n dimension according to the two probability sequences.

Step 4. Classifier initialization. The purpose of this step is to select a decision matrix for each classifier with a dimension of 2 × 4000. The corresponding decision matrix d(x, y) is initialized by the probability value of each selected channel in p1 and p2, the value of rest channels are set to be 0.

Step 5. Classifier fine-tuning. The obtained decision matrix d(x, y) needs further tuning. In each iteration, if the current classifier predicts incorrect label, the corresponding decision matrix is fine-tuned by adding the product of current activated MPPs sequence and a learning rate.

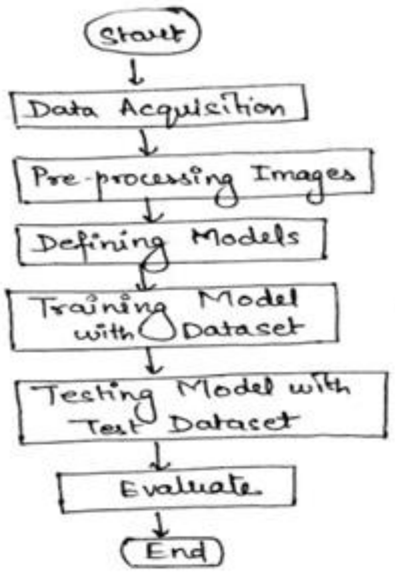

Fig 3: Step by Step Training Diagram

3. EXPERIMENT

In this section, we will first introduce the implementation details of our CNN model, including the architecture selection and training. We will implement our CNN in Keras. Keras is a deep learning library written in python and allow us to do quick experimentation. After training and testing the model we would finally do performance comparison based on accuracy with given number of parameters the model is evaluated.

A. Implementation details

To implement our system, a computer with 2 GHz Intel core i7 with 8GB 1600 MHz DDR3 memory is employed. We here are using anaconda python with theano and keras and other libraries installed for running our code and getting output.

1) Dataset Acquisition: We need to download 'Images and annotation' for training and test set from the GTSRB website and extract them into a folder. Also, we need to download 'Extended annotations including class ids' file for the test set. Then we must organize these files in our directory for further accessing in code.

2) of images: Since the representative image consist of noises and vary a lot in illumination and size we need to equalize and normalize these images. Hence we use functions of histogram equalization in HSV color space and resize the images to standard size.

Once the above process is done then we pre-process all the training images and store into numpy arrays. We also get labels of images from the paths as described in the directory structure. We also convert targets to one-hot form as is required by keras.

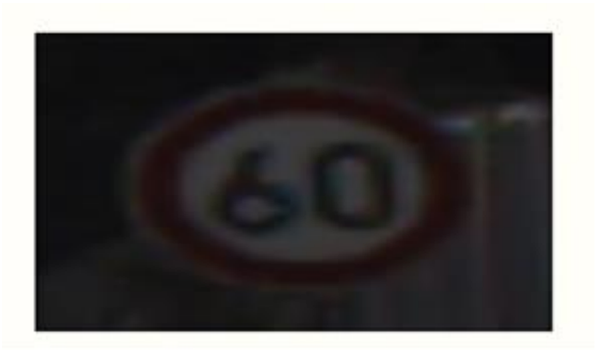

Fig 4: Input Image to pre-process the image

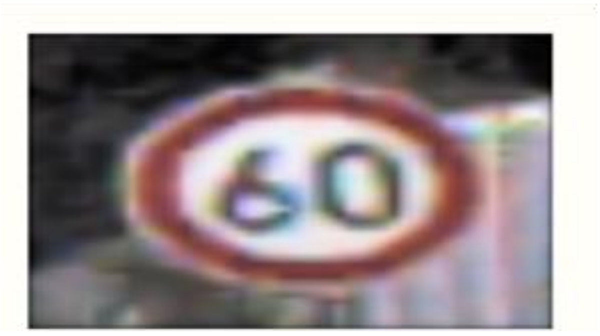

Fig 5: Processed Image

3) Defining Models: After the above steps being completed we define our models. We used feed-forward network with 6 convolutional layers followed by a fully connected hidden layer. We also used dropout layers in between because Dropout regularizes the network, i.e. it prevents the network from overfitting.

All our layers have ReLU activations except the Output layer. Output layer uses softmax activation as it has to output the probability for each of the classes.

Sequential is a Keras container for a linear stack of layers. Each of the layers in the model needs to know the input shape it should except, but it is enough to specify input_shape for the first layer of the Sequential model. Rest of the layers do automatic shape inference.

To attach a fully connected layer (aka dense layer) to a convolutional layer, we reshape/flatten the output of the conv layer. This we achieved by Flatten layer.Before training the model, we configured the model, the learning algorithm and compile it. We optimized the loss function and used optimizer and metric for optimizing and finding the accuracy of our model.

4) Training: After the above process we trained our designed model. During the training, our model will iterate over batches of the training set, each of size batch_size. For each batch, gradients will be computed and updates will be made to the weights of the network automatically. One iteration over all the training set is referred to as an epoch. Training is usually run until the loss converges to a constant.

We have also added a few features such as Learning Rate Scheduler which is decaying learning rate over epochs to help learn better and Model Checkpoint to save the model with best validation accuracy. We implemented these features via callback feature of keras.

5) Loading test data: After the model is completely trained we then load the test data and evaluate our model on it.

6) Testing with the real environment: After being satisfied with the result we now use the above system with live camera footage and use a beep sound to alert the driver in case a traffic sign is detected.

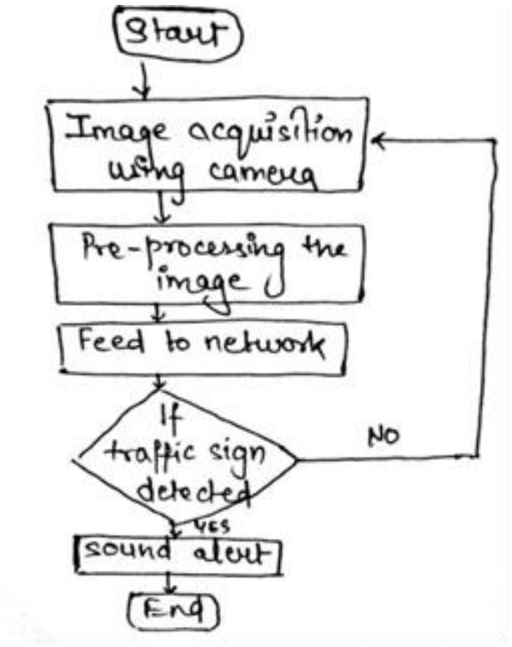

Fig6: Step by Step Processing Diagram

4. RESULT

The accuracy we obtained from the test data set is 97.92%. That was very close to that of human which is nearly 98.84%. However, with this amount of accuracy, we can certainly tune our model to juice out some more accuracy which can be sufficient for this system real-life application. With live footage, the system was able to produce beep sounds to alerts the driver when the sign was detected.

5. CONCLUSION

In this paper, a novel traffic sign detection and recognition system along with driver alert system are proposed. With the accuracy and real-time results, the proposed system has potential to be used in autonomous as well as non-autonomous driving vehicles where it can be used to alert the driver regarding the road side traffic signs that are coming up or passing by. This system if imposed with the help of mechatronics can be used to automatically take actions regarding the traffic signs that appear.With certain improvements with the model, this system yields promising results.

6. REFERENCES

[1] “Traffic Sign Detection in Static Images using Matlab”, Garcia, M.A.; Sotelo, M.A.; Gorostiza, E.M. Emerging Technologies, and Factory Automation, 2003. Proceedings. ETFA apos;03. IEEE Conference Volume 2, Issue, 16-19 Sept. 2003 Page(s): 212 - 215 vol.2.
[2] Rubén Laguna, Rubén Barrientos, L. Felipe Blázquez, Luis J. Miguel *Department of Electrical and Automatic Control and Systems Engineering, University of León, E.II.I.I., Campus de Vegazana s/n, 24071 León, Spain.
[3]. World Health Organization, "Global Status Report on Road Safety 2015," 2015.
[4] M.L. Eichner, T.P. Breckon, “Integrated Speed Limit Detection and Recognition from Real-Time Video”, Proc. IEEE Intelligent Vehicle Symposium, The Netherlands, 2008
[5] K.A. Ishak, M.M. Sani, N.M. Tahir, S.A. Samad and A. Hussain, “A Speed limit Sign Recognition System Using Artificial Neural Network”, 4 th Student Conference on Research and Development, pp. 127-131, 2006
[6] A. Bargeton, “Improving pan-European speed-limit signs recognition with a new “global number segmentation” before digit recognition”, In Proc. Conf. IEE Intelligent Vehicles Symposium, pp. 349-354, 2008
[7] M. Szarvas, U. Sakai and J. Ogata, “Real-time pedestrian detection using LIDAR and convolutional neural networks”, in Proc. IEEE Intell. Veh. Symp., pp. 224-229, 2005
[8] Y. Lecun, L. Bottou, Y. Bengio and P. Haffner, “Gradient-based learning applied to document recognition”, Proceedings of the IEEE, vol. 86, no. 11, pp. 2278-2324, 1998
[9] S. Haykin, “Neural Networks and Learning Machines”, 3rd ed., Prentice Hall, 2008
[10] D. Nistér, H. Stewénius, “Scalable recognition with a vocabulary tree”, CVPR, issue 2, pp. 2161-2168, 2006
[11]Y.-Y. Nguwi and S.-Y. Cho, “Emergent self-organizing feature map for recognizing road sign images”, Springer, Neural Computing & Applications, vol. 19, no. 4, pp. 601-615,2009