Traffic Sign Detection under Challenging Conditions Using Faster RCNN

Mohammad Ariful Haque, Sayeed Shafayet Chowdhury, Ahmed Maksud, Jubaer Hossain, Kinjol Barua

Department of Electrical and Electronic Engineering
Bangladesh University of Engineering and Technology
Dhaka,Bangladesh

arifulhoque@eee.buet.ac.bd

Roknuzzaman Rokon, Muhammad Suhail Najeeb, Nahian Ibn Hasan, Shahruk Hossain, Shakib Zaman and SM Raiyan Chowdhury

Department of Electrical and Electronic Engineering
Bangladesh University of Engineering and Technology
Dhaka,Bangladesh

sayeedchowdhury@eee.buet.ac.bd

***Abstract* – Traffic sign detection has different traditional methods which are not robust or effective. In this paper the state of the art faster RCNN method has been used to detect traffic signs from different challenged conditions. With the help of Kalman filter and Lukas-Kanade tracker the detection process is improved. Finally, a Convolutional Neural Network (CNN) is used to classify the signs of the frames.**

I. Introduction

Traffic sign recognition is a multi-class classification problem where the class frequencies are practically random. It deals with a real-time computer vision problem of high practical interest. Now-a-days, it has become an essential component of the Driver Assistance System (DAS) and Unmanned Ground Vehicle (UGV). Although, there are lots of algorithms regarding this topic, as German Traffic Sign Recognition Benchmark (GTSRB)[10] have shown that, there are still so many problems to be dealt with such as lens blur, colour distortion, Gaussian blur, over-exposure, occlusion, contrast degradation, distortion caused by environment (i.e. distortion due to haze, rain etc.) and even illumination and reflection problems which makes it both tough and challenging to obtain a considerable accuracy in sign detection.

II. Related Work

Traditional algorithms of traffic sign detection are based on hand crafted features (e.g. HOG [1]) and regular classifiers (e.g. SVM [2]), but they are not well discriminative when the database is very large and contains generic deep features. In that case, a Convolutional Neural Network (CNN) in conjunction with a Deep Neural Network (DNN) can learn and extract the features with high level of discrimination and robustness [1, 2 and 3]. A variety of CNN models have been proposed so far. In most of the cases, some preprocessing has to be done to increase the feature selectivity of the frames. Pierre *et al.* [3] fed the CNN-extracted high and low level features to the fully connected layers and the accuracy is up to 99.17%. Dan Ciresan, *et al.*[4] pre-processed the input images using contrast limited adaptive histogram equalization (CLAHE) and used multilayer perceptron (MLPs) to improve the performance of CNN trained with HOG feature descriptors and thus the accuracy increased up to 99.15% and further increased to 99.46% when the data is pre-processed using various contrast normalization methods.

It has been proved that, the generalization ability of the fully connected layers in CNN is limited and application of CNN learnt feature to a robust classifier is better than the previous one. Sharif *et al.* [5] applied the CNN learnt features to linear support vector machine classifier (SVM) and after testing on various standard dataset (e.g. PASCAL VOC 2007[6]) the performance of the detection system is found at its best. But due to unacceptable time cost for large dataset and inevitability of cross validation for training SVM has made its performance non-optimal. Gradually, various algorithms like random vector version of the functional link (RVFL) of which single hidden layer feed forward network (SLFN) is a special case have been evolved. Huang *et al.* [7] proposed a powerful SLFN based algorithm called extreme learning machine (ELM) [7, 8, and 9]. It chooses the hidden node parameters in a random fashion and determines the output weight by solving the smallest norm least square solution of SLFN. It is very much training-friendly and also time saving in the case of an extremely large dataset thus has outperformed the other conventional algorithms. Then comes the deep belief network (DBN) and combined with ELM, it has been proved the best in the case of pattern recognition.

In this paper, a traffic sign detection architecture is proposed where the detection is to be done from the video dataset provided by GTSRB. The details of the dataset are available at [10]. Firstly the given videos on 12 challenges are separated in sequential frames and then these images are fed to a CNN-ELM based 'Effect Classifier' to detect the class of challenges which the image distortion belong to. Then the processed images are further classified by 'Fast Regional Convolutional Neural Network' (FRCNN [10]) to find out the Region(s) of Interest (ROIs, here ROI is the region where the probability of the presence of a traffic sign is the maximum) from the images and eventually its output is fed to a 'Kalman Tracker' (KLT) which is based on Kalman filtering algorithm mainly for the real time detection. The signs of the tracked output of KLT are then detected using another CNN-ELM based ‘Sign Classifier’. There is also a feedback loop between KLT and Sign Classifier to sustain real time sign recognition. Experimental ensures that the accuracy of this architecture on the given dataset can reach up to (? %). In this paper, the fundamental algorithm is described in the schematic diagram of (1). Section (III) and (IV) introduce the mechanism of Effect classifier and preprocessing. Section V describes FRCNN respectively. The detailed discussion on Kalman Filter and KLT are also given in section (VI) and (VII) respectively. Section (VIII) discusses about sign classifier. Experimental results and comparison etc. are provided at section (IX). Finally, the conclusion and future work are given in section (X).

III. ALGORITHM OVERVIEW

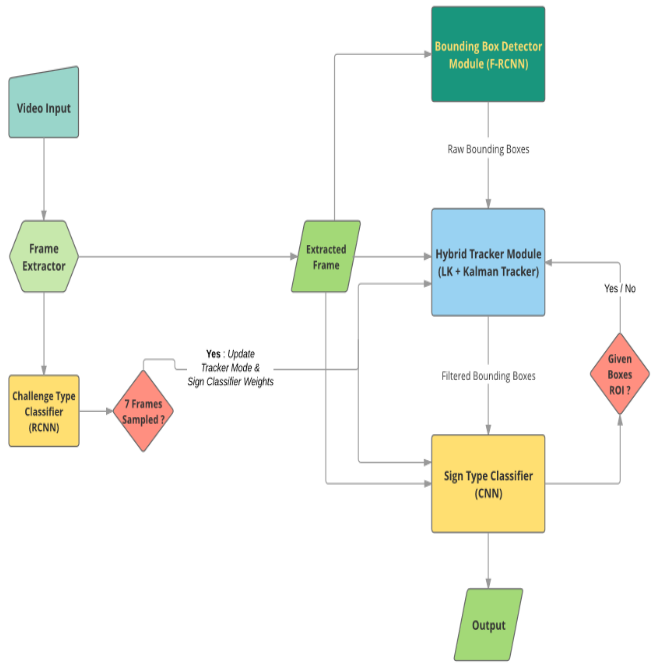

Video Input
Bounding Box Detector Module (F-RCNN)
Raw Bounding Boxes
Frame Extractor
Extracted Frame
Hybrid Tracker Module (LK + Kalman Tracker)
Yes / No
Given Boxes ROI ?
Filtered Bounding Boxes
Yes : Update Tracker Mode & Sign Classifier Weights
Challenge Type Classifier (RCNN)
7 Frames Sampled ?
Sign Type Classifier (CNN)
Output

Figure 1: Algorithm overview of traffic sign detection

The FRCNN takes a frame and proposes ROI boxes for it, which are then passed to the tracker module. The tracker module uses its own prediction based on either optical flow (Lucas Kanade) or Kalman Filtering, and compares it with the FRCNN output. It determines which co-ordinates are the best to take based on the fact that a box will move by only so much between frames. Besides keeping track of all the boxes, it uses feedback from the sign classifier to determine whether the boxes are valid. If the ROI box does not contain a traffic sign listed in the given labels, those boxes are discarded, and boxes in that same region are not accepted from FRCNN for 2 consecutive frames. The tracker system switches between Lucas-Kanade and Kalman dynamically based on the information from the challenge type classifier. For challenges where there are static objects on the frames, such as Dirty Lens (challenge 5) or Shadows (challenge 10), optical flow is not a suitable method for tracking and so we switch to Kalman Filter. All the blocks are described below according to the sequence they are operated for a complete detection and classification.

IV. CHALLENGE TYPE CLASSIFIER

Recurrent Convolutional neural networks [10] have a long history of being a successful algorithm for different tasks related to artificial neural network such as Handwriting-recognition [11], Speech recognition [12], Image recognition [13], Text classification [14] etc. In our specific set of problem the Recurrent CNN algorithm is used for classification of an image into 11 classes which correspond to 12 different challenge types.

An input frame from a video sequence is first feed forward through a recurrent convolutional neural network for successfully classify the frame into a type from among the 11 classes. Say an input ith frame $I_{abcde,i}$ of video sequence-type ‘a’, sequence number ’b’, challenge-source-type ‘c’, challenge-type ‘d’ and challenge-level ‘e’ is feed forward through the RCNN model. The frame is expected to be classified as challenge –type‘d’.

Recurrent convolutional layer (RCL) is considered to be the fundamental module of RCNN. The dynamic character of RCL can be visualized from equation [10] 1 and 2.

$$z_{ijk}(t) = (\mathbf{w}_k^f)^T \mathbf{u}^{(i,j)}(t) + (\mathbf{w}_k^r)^T \mathbf{x}^{(i,j)}(t-1) + b_k. \quad \ldots(1)$$

$$x_{ijk}(t) = g(f(z_{ijk}(t))), \quad \ldots\ldots..(2)$$

In the equation 1, u (t) and x (t-1) represents feed-forward and recurrent-input respectively. These represents the vectorized patches cantered at (i, j) of feature maps within the previous and current layer, w_k^f and w_k^r are the vectorized feed forward weights and recurrent weights, respectively and b_k is bias. In equation 2 ‘f’ is the rectified linear activation function. Figure 1 provides the network architecture [1] that has been used for our algorithm.

In our specific application 6 Recurrent Convolutional Layers are used. In each block there are 4 convolutional layers. After each convolutional layer there are Batch Normalization layers and Relu layers. Maxpooling layers have been included after every 2nd recurrent layer. Dropout layers have been included after every recurrent layer. Maxpooling layers have kernel size of 2 and stride of 1 and the dropout layers have dropout percentage of 10%. The output of the final RCL outputs the maximum among all the feature map, providing a feature vector which represents the image. Finally, a softmax classifier has been used to classify the output vector to 9 categories. The output is provided by equation 3[10].The overall training is done using the ‘adam’ optimizer and using cross-entropy as loss. The momentum for the dataset is 0.9. All weights have the same decay term. The input images are directly inserted from the video sequence.

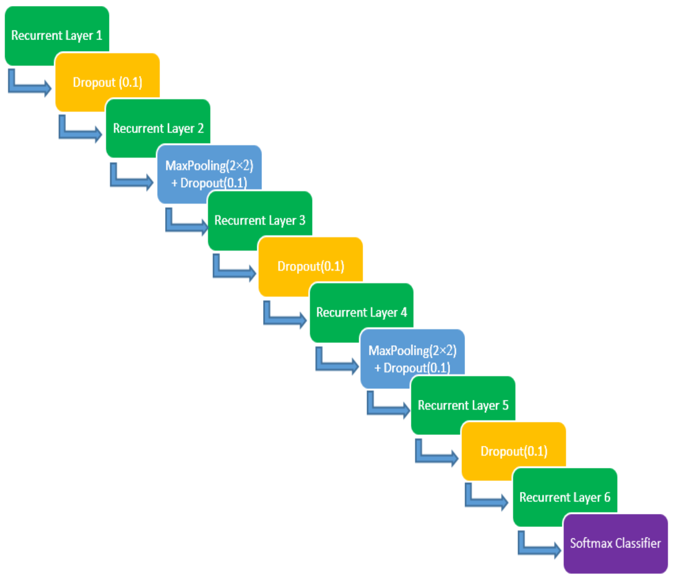


Figure 2: Overall architecture of RCNN

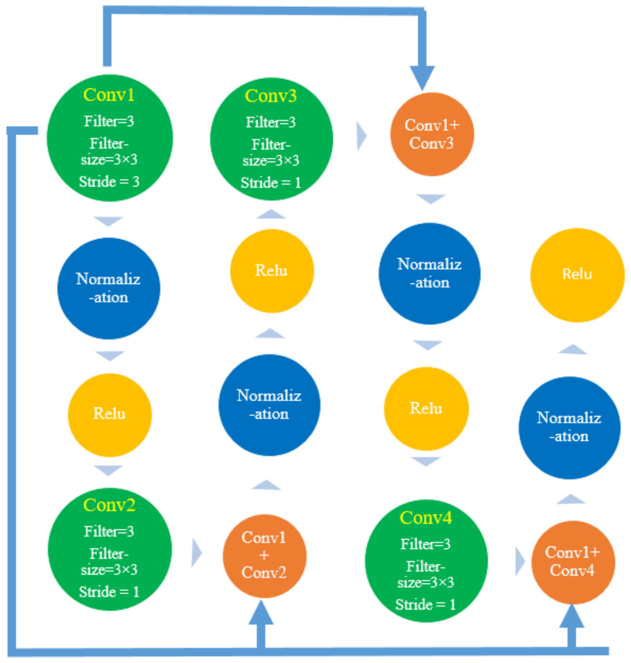

Conv1 Filter=3 Filter-size=3×3 Stride = 3
Conv3 Filter=3 Filter-size=3×3 Stride = 1
Conv1+ Conv3
Normaliz -ation
Relu
Normaliz -ation
Relu
Relu
Normaliz -ation
Relu
Normaliz -ation
Conv2 Filter=3 Filter-size=3×3 Stride = 1
Conv1 + Conv2
Conv4 Filter=3 Filter-size=3×3 Stride = 1
Conv1+ Conv4

Figure 3: Inherent Network within one Recurrent Layer

The dataset consists of 1,234,800 colour images each of size 125×125 pixels after resizing from a 400×400 pixels segment from the centre or the original image. There are 9 classes in the data set containing 12 different challenges. The 7 most significant challenge conditions which affect the FRCNN training have been modelled by 6 different Recurrent CNN models. Also there is a background model which is modelled for other challenge conditions and a no-challenge class. Each of the models is run for 50 epochs with a batch size of 50. The model is tested over a dataset of 529,200 images and an accuracy of 96.7% is achieved to classify challenged conditions. There are 128 filters with the size of 3×3.

V. ROI Detection with Faster RCNN

Faster Recurrent Convolutional neural networks [15] has been used to detect object because it consumes less time than RCNN [16]. There are two modules, one is region proposal network (RPN) [16] and fast- RCNN [17] detector. Region proposal network gives some rectangular object proposal and their objectness scores. It tells the detector module where to look at. RESNET 50 [18] network is used both in RPN and RCNN. Max overlap threshold used for RPN is 0.7 and RPN stride is 16.

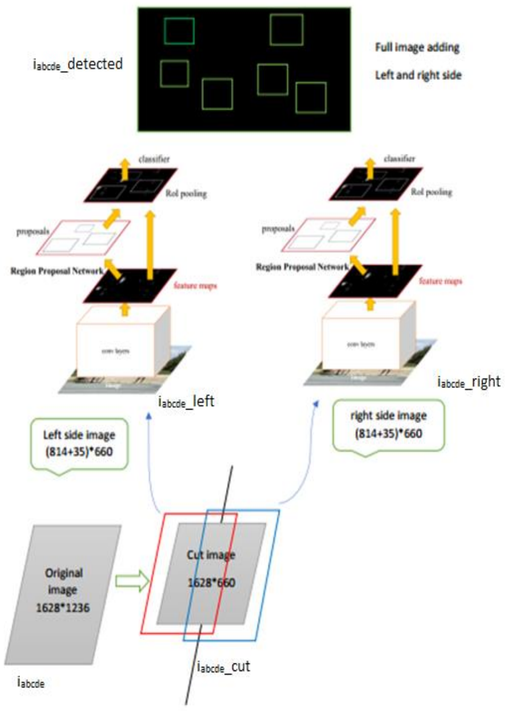


Figure 4: Schematic representation of ROI detection system [11]

In this competition, objects which should be detected are very small i.e. they occupy very few pixels in whole image. This is quite different from normal image datasets such as PASCAL VOC, COCO etc. State-of-the-art object detection algorithm fails to give satisfactory result to detect small objects in image [19]. Also, some hardware limitation is there that's why some technique is used here which give satisfactory result in short time. Dataset has some unique features and these are used for favours. Since traffic sign usually lies above the horizon as this is viewed from a vehicle, some lower portion of image is cut down which does not affect detection greatly but reduces computational time.

As shown in figure no 2 original image 'i_{abcde}' is 1628*1236, after cutting it become 'i_{abcde}_cut' (1628*660). Then image is cut into two part as left side 'i_{abcde}_left' (849*660) and right side 'i_{abcde}_right' (849*660) keeping overlap of 70 pixels. Two models are used to detect objects in two images 'i_{abcde}_left' and 'i_{abcde}_right' After classifying objects in both 'i_{abcde}_left' and 'i_{abcde}_right', two are combined and final image i_{abcde}_detected with bounding boxes are found.

Bigger ground truth box is taken to allow some background of image. Model trained with sign plus some background i.e. context helps in detection [19]. There are 14 different traffic signs but we grouped them in 9 groups, similar signs in same group, for CNN classifier which helps in detection. Non-maxima suppression has been applied to get biggest bounding box [14]. Anchor box scale is [50, 150] and ratio is [1, 1] which is smaller than suggested in paper [21]. Learning rate is 1x10^-4.

VI. Kalman Filter basics

Kalman filter is used for tracking traffic signs in the program. Kalman filter works by predicting and correcting the states of wide range of linear processes. At first it is assumed that the dynamic behavior of the sign to be tracked as x_k, where the subscript k indicates the time instance of current state. Here, we represent the centre of the mass of the bounding box of the traffic sign as the state x. x_k is to be estimated from the measurement of z_k. Here z_k corresponds to the output of the FRCNN used to detect the signs. A brief description of the mathematics used in this process is given below:

$$x_k = Ax_{k-1} + w_{k-1} \;\dots\dots\dots\dots\dots\dots (5)$$

Where A represents the transition matrix to relate the x_{k-1} at time instance $k-1$ to x_k at time instance k. And w_{k-1} is a vector which represents the process noise. It is assumed that the process noise is Gaussian $N(.)$ having the normal probability distribution $p(w) \sim N(0,Q)$. The measurement equation is

$$z_k = Hx_k + v_k \dots\dots\dots\dots\dots\dots. (6)$$

Where H is the measurement matrix, z_k is the measurement made at time instance k and v_k is the measurement noise. Like process noise, we also assume that measurement noise is Gaussian with normal probability distribution $p(v)$.

$$p(v) \sim N(0,R)$$

Equations (5) and (6) describe a linear model at time k. In equation (5), x_k is estimated from previous state x_{k-1}. x_k^- is assumed as a priori estimate of the state x_k. Let P_k be the covariance matrix of the process noise at time instance k. Then,

$$x_k^- = Ax_{k-1} + w_k \dots\dots\dots\dots\dots\dots (7)$$

$$P_k^- = AP_{k-1}A^T + Q \dots\dots\dots\dots\dots (8)$$

Now the posterior estimate of x_k is made by combining our prior estimate x_k^- and measurement z_k. The equations are as follows:

$$K_k = P_k^- H^T (HP_k^- H^T + R)^{-1} \dots\dots\dots (9)$$

$$x_k = x_k^- + K_k(z_k - Hx_k^-) \dots\dots\dots (10)$$

$$P_k = (1 - K_k H)P_k^- \dots\dots\dots\dots. (11)$$

K_k is the Kalman gain for the current state. The aposteriori state estimate x_k is used as the apriori estimate for the next time instance k+1 and the process goes on recursively.

In the dataset sometimes there are multiple signs in one frame. As FRCNN detects multiple signs at one instance, there are multiple predictions and multiple measurements. Here the challenge is to assign the measurement of the current state to the prediction of the current state which is based upon the estimate of the previous state. The nearest neighbour concept is adopted to solve the problem. As the positions of the traffic signs do not change abruptly from frame to frame, the Euclidean distance is calculated between the points of prediction and measurement of the current frame. It has been assumed that if any distance between a prediction and a measurement is less than 50 pixels, then that measurement belongs to that predictions. And that measurement-prediction pair is used for updating a priori estimate x_k^- to get aposteriori estimate x_k in equation (10).

VII. Lukas-Kanade method for traffic sign tracking

The Lucas–Kanade method [20] assumes that the displacement of the image contents between two nearby instants (frames) is small and approximately constant within a neighbourhood of the point p under consideration. Thus the optical flow equations can be assumed to hold for all pixels within a window cantered at p. The local image flow (velocity) vector (V_x, V_y) must satisfy,

$$I_x(q_1)V_x + I_y(q_1)V_y = -I_t(q_1)\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots.(12)$$

$$I_x(q_2)V_x + I_y(q_2)V_y = -I_t(q_2)\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots(13)$$

$$I_x(q_n)V_x + I_y(q_n)V_y = -I_t(q_n)\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots..(14)$$

where $q_1, q_2, \dots, q_n$ are the pixels inside the window and $I_x(q_i)$, $I_y(q_i)$, $I_t(q_i)$ are the partial derivatives of the image I with respect to position x, y, and time t evaluated at the point q_i and at the current time.

This system has more equations than unknowns. The Lucas–Kanade method obtains a compromise solution by the least square principle.

Harris corner detection method has been used to detect corners in images and Lucas-Kanade method is used to track those points and the optical flow vectors of those points are received. Then from FRCNN, the positions of the signs i.e. region of interests (ROIs) in the image are obtained. Then the nearest points for each ROI from Harris corner detection are detected and used their optical flow vectors to estimate the new positions of the ROIs in the next frame.

VIII. Traffic sign classifier using CNN

Current popular algorithms mainly use convolutional neural networks (CNN) to execute both feature extraction and classification in traffic sign detection. CNNs, like neural networks, are made up of neurons with learnable weights and biases. Each neuron receives several inputs, takes a weighted sum over them, pass it through an activation function and responds with an output. A CNN usually takes an order 3 tensor as its input, e.g., an image with H rows, W columns, and 3 channels (R, G, B colour channels).The input then sequentially goes through a series of processing. One processing step is usually called a layer, which could be a convolution layer, a pooling layer, a normalization layer, a fully connected layer, a loss layer, etc.

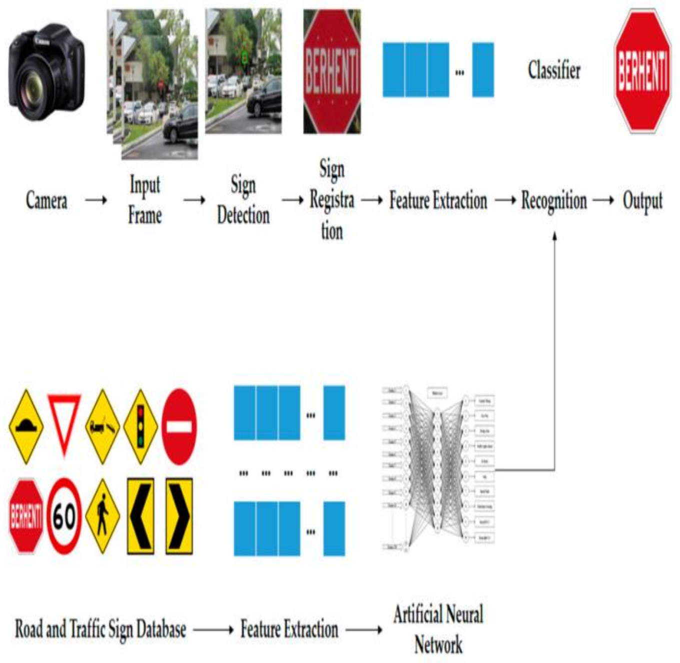
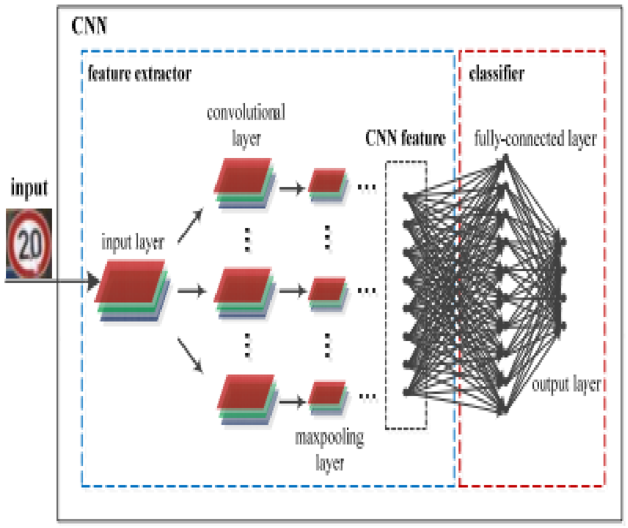


Figure 5: Schematic diagram of Traffic Sign Detection [21]

A convolutional Neural Network (CNN) works on volume i.e. works on an image of 32x32x3 where height, width and depth are 32, 32 and 3 respectively. The depth vector is represented as RGB channel. The performance could be further improved if other favourable classifiers are used and extreme learning machine (ELM) classifier is just the candidate.

The first contains alternating convolutional and maxpooling layers. The input of each layer is just the output of its previous layer (See Figure 2). As a result, this forms a hierarchical feature extractor that maps the original input images into feature vectors. Then the extracted features vectors are classified by the second part, that is, the fully-connected layers, which is a typical feed forward neural network.

Here a CNN network of two convolution layers with 32 filters is used and image size is 32x32x3 with RGB multi-colour channel and convolution kernel dimension is 3x3. The number of output nodes are 24 which represents the both 14 signs classes and other stuffs like tree, window, car, building etc. The complete network consists of 2 convolution layers, Max Pooling layers, dropout layers and then again two convolution layers. At last the final architecture terminates with a fully connected layer with 24 output classes or nodes that consists of 14 sign classes and other staffs extracted from Faster RCNN.



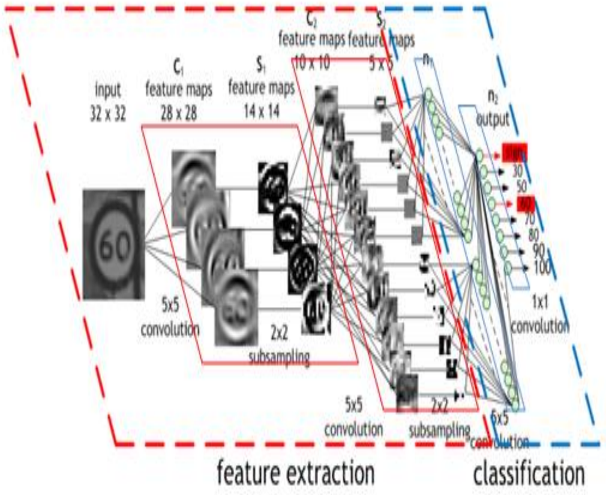


Figure 6: CNN Architecture for feature Extraction [22]

IX. Results

The results obtained after testing the dataset is given below in the table as the number of true positive, false positive, true negative, false negative frames. All the test dataset contains 1755 videos each with 300 frames.

True Positive	182767
False Positive	199573
True Negative	138049
False Negative	350407
Total Precision	47.80%
Recall	34.28%
Total Accuracy	36.84%

The primary algorithm initially consisted of pre-processing unit which actually provided processed frames from the challenge type classifier. The pre-processing unit consisted of methods like median filter for denoising frames, contrast enhancement for exposure control, gamma level control for darkening effect control, Richardson-Lucy algorithm for deblurring[24], guided image filter[25], wavelet transformation[26] for rain control etc. But there was no sign of improvement in the accuracy or precision in the output. Although the number of true positive frames increased.

X. Conclusion

The object detection algorithm Faster RCNN is considered to be the state of the art technology. That's why it's tested here as traffic sign detection. But still the accuracy is not over 60% even in the PASCAL VOC dataset. In the algorithm presented in this paper has used the Kalman filter and Lukas-Kanade tracker for boosting the accuracy which provided quite a bit satisfactory result.

XI. References

[1] J. Stallkamp, M. Schlipsing, J. Salmen, C. Igel.: Man vs. computer: benchmarking machine learning algorithms for traffic sign recognition, Neural Networks 32 (August) (2012) 323–332.

[2] A. Ruta, Y.M. Li, and X.H. Liu.: Robust class similarity measure for traffic sign recognition, IEEE Trans. Intell. Transp. Syst.11 (December(4))(2010) 847-855

[3] Sermanet, Pierre, and Yann LeCun.: Traffic sign recognition with multi-scale convolutional networks. Neural Networks (IJCNN), the 2011 International Joint Conference on. IEEE, 2011.

[4] Cireşan Dan, et al.: A committee of neural networks for traffic sign classification. Neural Networks (IJCNN), the 2011 International Joint Conference on. IEEE, 2011.

[5] Razavian Ali Sharif, et al.: CNN Features off-the-shelf: an Astounding Baseline for Recognition. ArXiv preprint arXiv: 1403.6382 (2014).

[6] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman.: The PASCAL Visual Object Classes Challenge 2012(VOC2012) Results. http:/pascalnetwork.org/challenges/VOC/voc2012/index.html

[7] Huang, Guang-Bin, Qin-Yu Zhu, and Chee-Kheong Siew.: Extreme learning machine: a new learning scheme of feedforward neural networks. Neural Networks, 2004. Proceedings. 2004 IEEE International Joint Conference on. Vol. 2. IEEE, 2004.

[8] Guang-Bin Huang, Qin-Yu Zhu, and Chee-Kheong Siew.: Extreme learning machine: theory and applications. Neurocomputing 70.1 (2006): 489-501.

[9] Guang-Bin Huang, et al.: Extreme learning machine for regression and multiclass classification. Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on 42.2 (2012): 513-529.

[10] Liang, M., & Hu, X. (2015). Recurrent convolutional neural network for object recognition. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (pp. 3367-3375).

[11] 13. Graves, M. Liwicki, S. Fernandez, R. Bertolami, ´ H. Bunke, and J. Schmidhuber. A novel connectionist system for unconstrained handwriting recognition. IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI), 31(5):855–868, 2009.

[12] 14. A. Graves, A.-r. Mohamed, and G. Hinton. Speech recognition with deep recurrent neural networks. In IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), pages 6645–6649, 2013.

[13] 15. Ebrahimi Kahou, S., Michalski, V., Konda, K., Memisevic, R., & Pal, C. (2015, November). Recurrent neural networks for emotion recognition in video. In Proceedings of the 2015 ACM on International Conference on Multimodal Interaction (pp. 467-474). ACM.

[14] Lai, S., Xu, L., Liu, K., & Zhao, J. (2015, January). Recurrent Convolutional Neural Networks for Text Classification. In AAAI (Vol. 333, pp. 2267-2273).

[15] S. Ren, K. He, R. Girshick, and J. Sun, "Faster R-CNN: Towards real-time object detection with region proposal networks," in Neural Information Processing Systems (NIPS), 2015.

[16] Jawadul H. Bappy and Amit K. Roy-Chowdhury, "CNN based region proposals for efficient object detection"

[17] R. Girshick, "Fast R-CNN," in IEEE International Conference on Computer Vision (ICCV), 2015.

[18] Kaiming He Xiangyu Zhang Shaoqing Ren Jian Sun, "Deep Residual Learning for Image Recognition", CVPR 2016 paper

[19] Chen, C.; Liu, M.-Y.; Tuzel, C.O.; Xiao, J., "R-cnn for small object detection."

[20] B. D. Lucas and T. Kanade (1981), an iterative image registration technique with an application to stereo vision. Proceedings of Imaging Understanding Workshop, pages 121--130

[21] http://www.mdpi.com/1424-8220/17/4/853/htm

[22] http://www.ntu.edu.sg/home/egbhuang/pdf/Traffic-Sign-Recognition-Using-ELM-CNN.pdf

[23] http://academic.mu.edu/phys/matthysd/web226/Lab01.htm

[24] Tai, Y. W., Tan, P., & Brown, M. S. (2011). Richardson-Lucy deblurring for scenes under a projective motion path. IEEE Transactions on Pattern Analysis and Machine Intelligence, 33(8), 1603-1618.

[25] He, K., Sun, J., & Tang, X. (2013). Guided image filtering. IEEE transactions on pattern analysis and machine intelligence, 35(6), 1397-1409.

[26] Portilla, J., Strela, V., Wainwright, M. J., & Simoncelli, E. P. (2003). Image denoising using scale mixtures of Gaussians in the wavelet domain. IEEE Transactions on Image processing, 12(11), 1338-1351.