An Appearance-based Approach to Detect the Wrong-way Movement of Vehicles Using Deep Convolutional Neural Network

Mutasim Billah Bin Ahmad
Faculty of Science & Technology
American International University-Bangladesh, Dhaka, Bangladesh
mutasim.bin.ahmad@gmail.com

Md. Rafsan Jany Chowdhury
Faculty of Science & Technology
American International University-Bangladesh, Dhaka, Bangladesh
rafsanjany26@gmail.com

Akif Ahmed
Faculty of Science & Technology
American International University-Bangladesh, Dhaka, Bangladesh
akif.ahmed5718@gmail.com

Kashfa Sehejat Sezuti
Faculty of Science & Technology
American International University-Bangladesh, Dhaka, Bangladesh
sezutikashfa@gmail.com

Tohedul Islam
Faculty of Science & Technology
American International University-Bangladesh, Dhaka, Bangladesh
islamtohedul@aiub.edu

ABSTRACT

To guarantee the enforcement of traffic rules, the identification of traffic rule violators is an exceptionally alluring yet difficult assignment to implement and the detection of the wrong-way movement of vehicles is one of them. In this paper, an appearance-based approach is proposed which detects the front and back side of the vehicles on a highway with the help of a deep convolutional neural network and decides whether a vehicle is moving along the wrong-way or not based on the user expectation to see the side of a vehicle on each side of the highway using a handcrafted region divider algorithm. The effectiveness of this strategy has been assessed on a primary data-set built on real-time traffic videos captured from several significantly busy highways of Dhaka Metropolitan City and proven quite productive with an accuracy of 96% on successful detection of wrong-way movement of vehicles.

CCS CONCEPTS

• **Computing methodologies → Neural networks**; **Object recognition**.

KEYWORDS

Traffic Rule Violation, Wrong Way Movement, Vehicle Detection, Deep Learning, CNN, YOLO

ACM Reference Format:
Mutasim Billah Bin Ahmad, Md. Rafsan Jany Chowdhury, Akif Ahmed, Kashfa Sehejat Sezuti, and Tohedul Islam. 2020. An Appearance-based Approach to Detect the Wrong-way Movement of Vehicles Using Deep Convolutional Neural Network. In *International Conference on Computing Advancements (ICCA 2020), January 10–12, 2020, Dhaka, Bangladesh.* ACM, New York, NY, USA, 7 pages. https://doi.org/10.1145/3377049.3377118

1 INTRODUCTION

In this modern era of highly populated societies, there is a need of automation in every part of our lives. We tend to automate all kind of processes in our life specially the ones that require human surveillance all the time. With the revolution of computer vision, researchers all over the world are trying to propose solutions for the tasks that can be automated through detection and decision making functions. With a very high population, metropolitan cities all over the world are facing problems in enforcing traffic rules and reducing traffic violations. The wrong-way movement of vehicles on highways is such a violation. The detection of such violations is such a tedious process where human surveillance can lead to major error. With having that in mind, this paper tends to find out through research whether the process of the detection of wrong-way movement of vehicles can be automated through an appearance-based approach with the assistance of a deep convolutional neural network.

2 RELATED WORK

A number of studies indicated that infringement of traffic standards plays an important role in the human factor. Traffic violations can be misdemeanors, depending on the type of violation that one conducts. Misdemeanor occurs when one becomes part of an accident. Wrong-way movement of vehicles continues to be an ongoing issue of traffic management because it increases the risk of fatal accidents and contributes significantly to congestion of the traffic. Since this study is trying to find a solution to automate the detection of wrong-way driving with the help of computer vision, object detection is a big part of this research. Object detection is an active topic in the research field of computer vision. Many methods have already been discovered to detect the presence of an object and still many research-works are going-on to discover new methods to increase the accuracy rate. A convolutional neural network (CNN) is such a technique that has a strong capacity to extract features from pictures in recent years. There are several algorithms for the detection. Among them two primary target detection algorithms that are seemed to be mostly used which are based on CNN: the regression algorithm and the regional proposal algorithm. The regional proposal technique extracts many suggested areas from a picture, which occurs by assessing the candidate regions that determine the category and position of the target. Algorithms such that Fast R-CNN [4], R-CNN [5], RFCN [3], SPP-Net [6], Faster R-CNN [13] etc. Compared to such regional proposal algorithms, regression algorithms such as SSD [9] and YOLO [12] do not need to acquire applicant frames to return a predicted target position in actual coordinates and identify its category. Till now, many researchers have worked on vehicle detection. Based on incremental learning and Fast R-CNN, K. Shi, N. Ma et al. proposed a model to detect forward vehicle [15]. In this paper they used KITTI data set and self-collected BUU-T2Y data set for the training purpose [15]. L. Wang et al. proposed a model with improved Faster R-CNN which detects vehicle based on drone images [17]. The Faster R-CNN is enhanced by using ResNet and building Feature Pyramid Networks (FPN) to obtain picture characteristics to address the issue of false detection and missed detection in vehicle tracking [17]. A. Samir et al. proposed a new architecture for assisting the driver in fog environment based on CNN [14] .In this document, they suggested a fresh architecture based on fast R-CNN for object detection in fogged pictures and a convolutional neural network (CNN) is intended to restore the sharp picture on the grounds of a reformulated atmospheric diffusion model for fog elimination [14]. G. Monteiro et al. worked on wrong-way drivers detection where they have used optical flow [10]. They took three staged in their account for the proposed solution. They are learning, detection and validation [10]. But the proposed method had no control over the learning process which creates a room for possibilities of making mistakes while learning. If a vehicle moves backward on a lane in that learning period, there would have been a false learning in that system. C. Vishnu et al. proposed a model that detects the motorcyclists without helmet using CNN [16]. They initially used the model of background subtraction to separate moving items [16]. After that they applied CNN to select motorcyclists and another CNN used for detect the motorcyclists without helmet [16]. The accuracy achieved by this model is very decent but running two CNNs sequentially generates issues in real-time detection.

3 PROPOSED MODEL

The proposed system in this paper has been designed such a way that it can work on real-time traffic surveillance videos. In this system, a system user is at first asked to set the regions on a highway and the side of the vehicles being expected to see by the system on each of the regions. Here, each of the frames of a video footage is considered as an input image to the system. The system itself is powered by a trained CNN model adopted from YOLO [12] for its remarkable performance (almost 45 frames in a second) on real-time detection. After the successful detection of the back and front side of the vehicles on an image, YOLO then returns the ROIs to

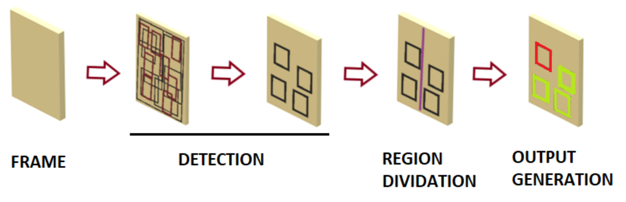

Figure 1: System Workflow of Wrong Way Movement Detection

the system with the following parameters: top-left point, height and width of the ROI. Vehicles showing the expected side on a region are marked green to depict its movement along the right way and for the opposite case, vehicles are marked red to indicate a wrong-way movement along the highway.

4 METHODOLOGY

4.1 Data Collection

The main focus of this study is to identify the vehicles on a highway in an incorrect manner which is why a data-set enriched with some excellent vehicle information is needed. At first, we had collected the Stanford Car data-set [7] only to train our system the front and back side of different cars usually seen on a highway. After that, we manually took some pictures of cars from different highways as primary data to make our system familiar with real-life scenarios. Also, we had collected some traffic surveillance footage from several busy highways of Dhaka Metropolitan City to be tested on the trained system.

4.1.1 Data Filtering. Though this data-set [7] is enriched with 16 thousand of car images, most of the images could not be taken to train the system since clearly, our goal is to detect only the front and back side of the vehicles. So, we only collected those images that show the front or back side of the cars perfectly. In that way, we could collect only 800 pictures, 400 pictures of each side, from that data-set [7]. Then we altered the data-set to extract a nice outcome. Some of them are done by cropping the images manually to eliminate the unnecessary information from those images to make the learning of our system more accurate.

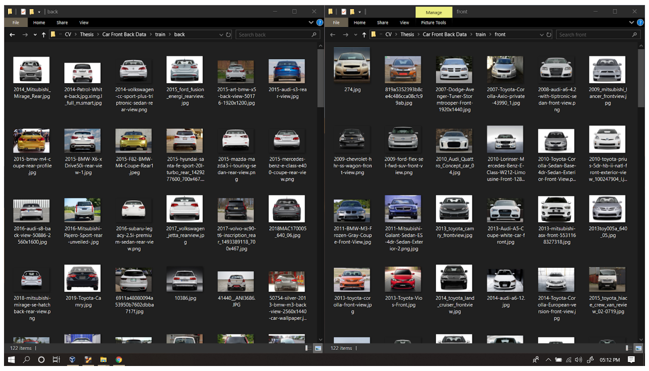

Figure 2: Filtered Stanford Car Dataset

4.1.2 *Data Augmentation.*

Data augmentation is one of the simplest ways through which we have optimized our training data-set. This process is very useful for small data-sets. It merely takes the input data-set and makes slight changes to it like rotation, scaling etc. This enabled us to construct more robust and non-over fitting models for the proposed system

4.1.3 *Data Labeling.*

Along with the 800 images collected from the Stanford Car data-set [7], we added 224 more images in our data-set that were taken from several busy highways of Dhaka Metropolitan city.

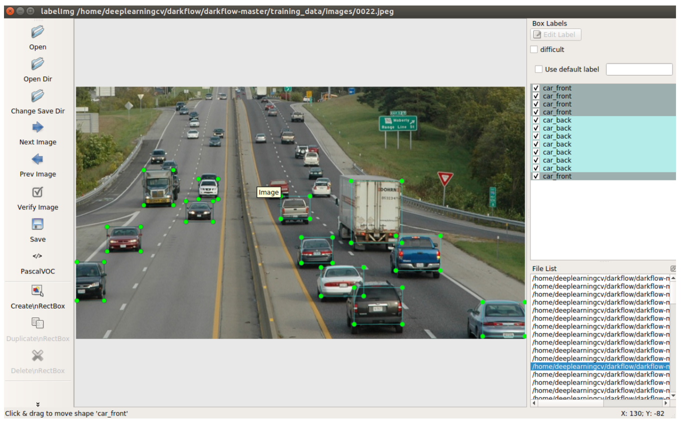

Figure 3: Data labeling

Then we marked down each of the cars from those highway images in a rectangular bounding box and labeled them accordingly (car_front or car_back) in PASCAL VOC format. These boxes are the regions of interest in an image for training. The exact positions of these ROIs in an image is saved in an XML file which is later on used to train our system.

4.2 Convolutional Neural Network

Convolutional neural networks enable computers to identify images of objects by evolving their actual images into some layers of incremental learning. A pile of feature maps is triggered each time it sees something and each layer will store a collection of characteristics or features from images such as lines, density, colors, edges etc. from that image or frame. It will update its learning time to time to generalize the same characteristics from the exposure of different images and their labels [8]. A basic CNN structure consists of the following parts: Input Layer, Convolution Layer, ReLU Layer, Pooling Layer, Flatten Layer, Activation Function, Fully Connected or Dense Layer, Loss Function and an Optimizer. These are described below:

4.2.1 *Input Layer.*

A picture or frame is considered as an input layer or volume consisting width, height and depth. It is a pixel value matrix. Channels are represented by the depth such as 3 channels for B, G, R and 1 channel for only gray-scale images. Number of layers here, must be divisible by 2 as many as required. 32, 64, 96, 224, 384, and 512 are some common numbers that are used for the number of input layers.

4.2.2 *The Convolution Layer.*

The Convolution Layer helps the classifier to learn the features and the characteristics from the images given as input. Being in the classifier, these layers extract various high- and low-level features and contribute to the overall learning. Image features are just the interesting portion of the targeted objects that make them unique from other objects.
Convolution is a mathematical word to describe the process of combining two features to create a new feature. This new feature or output is called a feature map. Feature maps are generated by sliding through the matrix from input layer or the previous convolutional layers since considering all the pixels from previous layers for calculation is too heavy for the system to execute. So, dot products are applied on receptive field and filter of all the dimensions. This generates a simple integer value for a cell in the feature map. Then on the next receptive field of that input image, the defined kernel moves through the stride and again calculates the dot product. This process is repeated again and again until the entire data-set has been passed through the neural network. The findings of this step will be passed to the next layer as input. In usual cases, a kernel or filter on the very first convolutional layer has a size of 5, 5 and 3 for color images. Features processed through a convolutional layer are the output volume produced by sliding the kernel through the images and then calculating the dot products from them. A local region with the same volume size as the kernels is known as receptive field. The total number of the kernels in a convolutional layer is called the depth for that particular layer. Depth columns or fibers are the collection of nodes or cells in feature maps that are pointing to the same reception field.
To produce a low quantity of production in space, the concept of stride has been introduced. This step has been configured such a way that the total amount of the result is an integer, not a floating-point value.
To maintain the size of the feature maps same as the size of input images, zero padding technique is applied by putting an extra layer of zeros by the edge of the input image. In that way, the border data do not get lost after the kernel slides though it and produces the feature maps. It helps to gain more accurate and relevant learning. Computation of an outcome of the configuration W2 x H2 x D2 has been derived below where the input configuration is W1x H1 x D1, receptive field is depicted as F, stride is S, the amount of zero padding used on the border is P and K is the depth:

$$W2 = \frac{W1 - F + 2P}{S + 1}$$

$$H2 = \frac{H1 - F + 2P}{S + 1}$$

$$D2 = K$$

The number of feature maps that we have extracted from our convolution layer, these outputs stacked together and treated as another 3d matrix which is the input to our next layer of our CNN.

4.2.3 *ReLU Layer.*

ReLU Layer applies an activation function that normalizes the values of the feature maps obtained from previous convolution layer. This layer does not change the volume size and hyper parameters do not exist. YOLO framework uses Leaky ReLU layer that broadens the range of learning. It is defined as:

$$f(x) = y; for\ x \geq 0\ and\ f(x) = -ay; for\ x < 0$$

4.2.4 Pooling Layer. This layer performs a task to reduce the size of the feature maps taken from the previous layer thus reducing the computational complexity of that proposed model. Such reduction helps the neural network to avoid an over-fitting state since it is designed to extract only the important features. It is applied on each of the feature maps contained by the layer. For such operation there exists some process namely max, sum and average pooling. Among them, max pooling is used mostly in convolutional neural networks since it extracts the most important features from the previous layer by considering the brightest pixels in that image. So, the proposed system adopted max pooling in each of its pooling layers having 2x2 size filters applied with a 2-stride step. Pooling sizes with bigger kernels are too damaging and generally result in poorer results since it omits some small details from the feature maps.

4.2.5 Flatten Layer. A CNN requires a flattening after the very last pooling layer of the network. Here the matrix is converted into a linear array so that the learning can be useful to the dense layer to apply SoftMax operation.

4.2.6 Fully Connected Layer. Insert Fully linked layers connect each neuron to each neuron in another layer in one layer. A SoftMax activation function is used by the last fully connected layer to classify the generated features of the input image into different classes based on the training data-set. SoftMax function turns numbers into probabilities that sum to one. SoftMax function outputs a vector that represents the probability distributions of a list of potential outcomes. This SoftMax function is used in our fully connected layer.

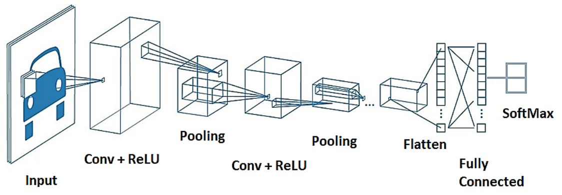

Figure 4: Different Major Parts of a CNN

4.2.7 Loss Function. Loss function is used to measure the inconsistency between the predicted value and the actual label. It is a non-negative value where the model's robustness improves as well as the loss function's value decline. We have used categorical cross-entropy in our model as our loss function. Cross-entropy shows the gap between what the models thinks the distribution of production should be and what the initial distribution really is. It has been described as [14],

$$H(y,p) = -\sum_i y_i \log(p_i)$$

Measurement of cross entropy is a commonly used squared error function. It is used when node activation represents the probability that each hypothesis could be true, i.e. when the output is a distribution of probability. In neural networks that have SoftMax activation in the output layer, it is used as a loss function [14]. Cross Entropy Loss with SoftMax function is used widely as the output layer. Now we are using the SoftMax derivative [17] we obtained previously to obtain the cross-entropy loss function derivative [2].

$$L = -\sum_i y_i \log(p_i)$$

$$\frac{\delta L}{\delta o_i} = \sum_k y_k \frac{\delta \log(p_k)}{\delta o_i}$$

$$= \sum_k y_k \frac{\delta \log(p_k)}{\delta o_i} \times \frac{\delta p_k}{\delta o_i}$$

$$= \sum_k y_k \frac{1}{p_k} \times \frac{\delta p_k}{\delta o_i}$$

From the equation of SoftMax,[1][2]

$$\frac{\delta L}{\delta o_i} = -y_i(1-p_i) - \sum_{k\neq 1} y_k \frac{1}{p_k}(-p_k.p_i)$$

$$= -y_i(1-p_i) + \sum_{k\neq 1} y_k.p_i$$

$$= -y_i + y_i p_i + \sum_{k\neq 1} y_k.p_i$$

$$= p_i\left(y_i + \sum_{k\neq 1} y - k\right) - y_i$$

yB•is a one hot encoded vector for the labels [10], so

$$\sum_k y_k = 1,\ and\ y_i + \sum_{k\neq 1} y_k = 1$$

So, we have,

$$\frac{\delta L}{\delta o_i} = p - i - y_i$$

4.2.8 Optimizer. Optimization is a parameter search method that minimizes our features or maximizes them. Usually we use indirect optimization when we train machine learning model. We choose a certain metric b•" such as accuracy, precision, or recall b•" that shows how well our model solves a problem. Equation of mini-batch gradient decent:

$$\frac{dE_{total}}{dw_s} = \frac{dE_{total}}{dOut_1} \times \frac{dOut-1}{dNetOut_1} \times \frac{dNetOut_1}{dw_s}$$

Equation of weight with learning rate (η):

$$w_s = w_s - \eta \times \frac{dE_{total}}{dw_s}$$

4.3 Model Training

In the training process, a total number of 1024 of images have been used. In each epoch the entire data-set is passed back and forth only once through the neural network. We pass the complete data-set to the same neural network several times. As the number of epochs rises, the neural network changes more numbers of times the weight and the curves move from under fitting to the ideal curve to over-fitting. The number of epochs is the number of times the model will cycle through the data. The more epochs are run, the more the model improves, up to a certain point. After that point, the model will stop improving during each epoch. For our model, we have set the number of epochs to 500. The instant transfer of the whole data-set through the network is an impossible task due to the limitation of memory. So, the data-set has been divided into a number of batches. Just as we split a large article into various sets / batches / parts such as introduction, related works etc. epochs, batch size and iterations, make the training process simple and comprehend. We need to split the data up into segments or Batches. Batch Size is the number of training samples we use in a single batch. In our model we have used 64 number of batch size. Iterations are the number of batches we need to complete one Epoch. In this case we used number of 16 (Number of images / Batch Size).

4.4 YOLO For Detection

YOLO or You Only Look Once adopts a totally extraordinary strategy. It uses a customary classifier that is re-purposed to be an item locator. YOLO splits the picture into a matrix of 13 by 13 cells at first. Each of these cells is in charge of foreseeing 5 bounding boxes. A bounding box portrays the square shape that encases an item. YOLO likewise yields a certainty score that reveals to us how certain it is that the anticipated bounding box really encases an expected object. This score does not utter a word about what kind of object is in the crate, just if the state of the case is any great to the system. For each bounding box, the cell likewise predicts a class. This works simply like a classifier and it gives a likelihood appropriation over all the potential classes. The certainty score for the bounding box and the class forecast are joined into one last score that discloses us the probability of this bounding box containing a particular kind of object.

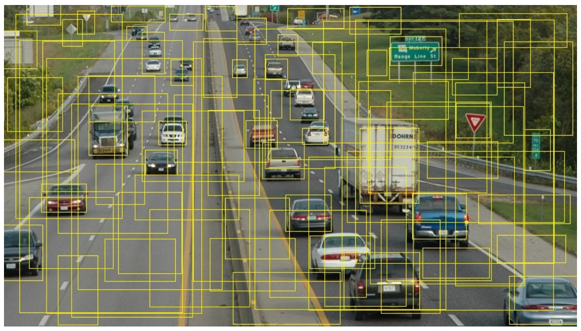

Figure 5: YOLO yielding confidence score for bounding boxes

Since there are 13G•13 = 169 framework cells and each of the cells predicts 5 boxes, we end up with 845 bounding boxes altogether. For an unknown reason, the greater part of these crates will have exceptionally low certainty scores, so we just keep the containers whose last score is greater than the threshold score provided. In this system, the threshold value is kept 0.2 or 20%.

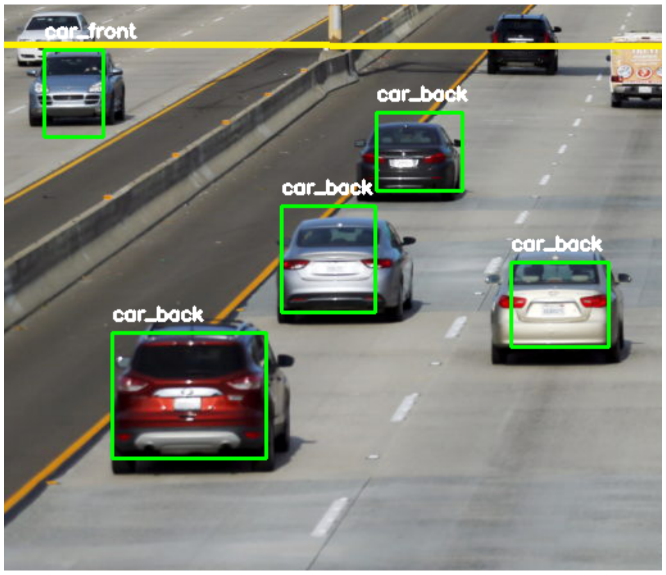

Figure 6: Prediction of YOLO with bounding box

Having such a low threshold point helps the system to detect a great number of ROIs under a detection border line. Above that, the cars or detected objects appear so tiny it becomes impossible to classify those models according to the learning of the model

4.5 Divide the Regions of Expectations

Before the detection process, the system user is asked to divide the highway image into regions and set the expected side of the vehicles the user intend to see on each of that particular regions. If the x coordinate value of the divider is mx, then

$$Cars(x, y, w, h)\, in\, left\, region, (mx - x) > 0$$

$$right\, region, (mx - x) < 0$$

Here we introduced a simple handcrafted calculation that checks if a particular car in a particular region showing the camera the expected side of the vehicles on that region set by the system user. It is marked green for an expectation match and red for other scenario, thus detects a wrong-way movement.

5 RESULTS AND FINDINGS

5.1 Test procedure

Here is an image of a highway road from Science Laboratory, Dhaka (Figure: 7). This highway is a two-way road but for testing the proposed model, we are considering it as a one-way road. That is why in this scenario, the system user is expecting to see only a particular side of the cars on both of the sides of the road, in this case, that will be the back side of the cars. Since the vehicles on the right region are showing their front sides to the camera, they are being considered on the opposite orientation of the user expectation thus marked as wrong-way movements by the system where the cars on the left are showing the same side as user expected. So, they are marked green by the system.

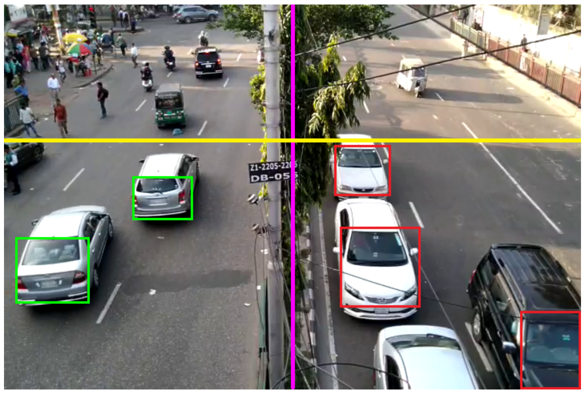

Figure 7: Cars on the right region is marked as wrong-way movements since the system user expected to see the back side of the vehicles on both of the regions

5.2 Findings

We have tested this proposed system on 100 test images of highways where 237 numbers of cars were showing the front side and 263 of cars were showing the backside of them. For this test, we have considered only the bottom one-third portion of each of the frames that we defined earlier as the region of detection of the proposed system

A confusion matrix below is showing our findings from the tests:

-	Front	Back
Front	229	8
Back	13	250

Table 1: Confusion Matrix

$$Accuracy := \frac{t_p + t_n}{t_p + t_n + f_n + f_p} = \frac{229 + 250}{229 + 250 + 8 + 13}$$
$$= 0.958 = 95.8\%$$

Taking Front as a positive result we have calculated Recall, Precision, F-score.

$$Recall(Front) := \frac{t_p}{t_p + f_n} = \frac{229}{229 + 13} = 0.946 = 94.6\%$$

$$Precision(Front) := \frac{t_p}{t_p + f_p} = \frac{229}{229 + 8} = 0.966 = 96.6\%$$

$$F - Score(Front) := \frac{2 \times Recall \times Precision}{Recall + Precision}$$
$$= \frac{2 \times 0.946 \times 0.966}{0.946 + 0.966} = 0.956 = 95.6\%$$

Taking Back as a positive result we have calculated Recall, Precision, F-score. Taking Front as a positive result we have calculated Recall, Precision, F-score.

$$Recall(Back) := \frac{t_p}{t_p + f_n} = \frac{250}{250 + 8} = 0.969 = 96.9\%$$

$$Precision(Back) := \frac{t_p}{t_p + f_p} = \frac{250}{250 + 13} = 0.951 = 95.1\%$$

$$F - Score(Back) := \frac{2 \times Recall \times Precision}{Recall + Precision}$$
$$= \frac{2 \times 0.969 \times 0.951}{0.969 + 0.951} = 0.959 = 95.9\%$$

After calculating the accuracy and f-score in both cases for front and back being positive we are getting a decent accuracy of 95.6% to 95.9%. This proves the acceptability of this approach for real time detection. This accuracy is expected to increase with a training having more data in our data-set that will help this model to more generalize its learning.

5.3 Comparison

Detection of the wrong-way movements of vehicles is not a common problem being solved in computer vision domain. Another computer vision-based solution to this problem has been discussed earlier in this paper proposed by G. Monteiro et al [10]. Their optical flow-based model had two different validation systems to prevent triggering a false alarm. A temporal validation to make sure the alarm is not being triggered by the noise and an appearance-based validation to trigger the alarm only for vehicles. However, the proposed model in this paper does not require such validations due to the nature of a deep convolutional neural network. This model first detects the objects of interest then decides whether to trigger an alarm according to the decision-making function it is given. So clearly, this model does require less steps which later resulted in a better performance of processing 320x240 pixel frames at 47 frames/s with a 3.2 GHz Intel Core i5 processor in a Linux machine over the model [10] which can process the frames of same dimension at 33 frames/s.

Again, according to the findings from an experiment described by Paula et al. [11], a convolutional neural network detects its objectives more accurately, about 6-7% more, than a Haar Cascade classifier which has been used at the appearance-based validation in the paper [10]. The model proposed in this paper has adopted a CNN at its core which resulted in a better accuracy of 96%.

6 FUTURE IMPROVEMENTS

This model is trained with a data-set of cars and car-like vehicles so it cannot identify all kinds of vehicles that move along the highway. So, the first and foremost target of the future work of this system will be to introduce more vehicles to the system that are commonly seen on highways. Along with that, we will try to train our model with an enhanced vehicle data-set to achieve a reasonable precision of the better consequence. We have a plan to carry out automatic region divider detection. It will fix several issues as the user of the system will no longer have to divide the region manually. For now, this system does not consider the fact whether the vehicle is actually moving or not. In future, the system will be upgraded which will allow it to consider only the vehicles that are actually moving.

7 CONCLUSION

At the core of the research behind this paper, the main objective was to find out whether such appearance-based approach is appropriate for the detection of the wrong-way movement of vehicles on highways. With that objective in mind, our system has been fed with car data-set collected from both primary and secondary [7] sources and got trained. We adopted a very popular framework YOLO [12] for detection and after applying our very own hand-crafted region divider algorithm the system was able to predict wrong-way movement of vehicles at a close range with almost 96% accuracy. Thus, the proposed approach is proven to be working perfectly. But this system is yet to be introduced with several other types of vehicles commonly seen on highways and also it does not run its detection only on the moving vehicles. Along with these limitations, the region divider will get improvement in our future work.

8 ACKNOWLEDGEMENTS

We would like to acknowledge all the members of our team, whose contributions have been crucial to the implementation of the proposed model. We would also like to thank Tohedul Islam for his guidance and assist in the writing of this paper.

REFERENCES

[1] Eli Bendersky. The softmax function and its derivative. https://eli.thegreenplace.net/2016/the-softmax-function-and-its-derivative/, 2016. Last accessed 14 January 2020.

[2] Paras Dahal. Classification and loss evaluation - softmax and cross entropy loss. https://deepnotes.io/softmax-crossentropy. Last accessed 14 January 2020.

[3] Jifeng Dai, Yi Li, Kaiming He, and Jian Sun. R-fcn: Object detection via region-based fully convolutional networks. In *Advances in neural information processing systems*, pages 379–387, 2016.

[4] R. Girshick. Fast r-cnn. In *2015 IEEE International Conference on Computer Vision (ICCV)*, pages 1440–1448, Dec 2015.

[5] Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 580–587, 2014.

[6] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Spatial pyramid pooling in deep convolutional networks for visual recognition. *IEEE transactions on pattern analysis and machine intelligence*, 37(9):1904–1916, 2015.

[7] Jonathan Krause, Michael Stark, Jia Deng, and Li Fei-Fei. 3d object representations for fine-grained categorization. In *Proceedings of the IEEE International Conference on Computer Vision Workshops*, pages 554–561, 2013.

[8] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

[9] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, and Alexander C Berg. Ssd: Single shot multibox detector. In *European conference on computer vision*, pages 21–37. Springer, 2016.

[10] Gonçalo Monteiro, Miguel Ribeiro, Joao Marcos, and Jorge Batista. Wrongway drivers detection based on optical flow. In *2007 IEEE International Conference on Image Processing*, volume 5, pages V–141. IEEE, 2007.

[11] Paula C Useche Murillo, Robinson Jiménez Moreno, and Javier O Pinzón Arenas. Comparison between cnn and haar classifiers for surgical instrumentation classification. *Contemporary Engineering Sciences*, 10(28):1351–1363, 2017.

[12] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 779–788, 2016.

[13] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In *Advances in neural information processing systems*, pages 91–99, 2015.

[14] Allach Samir, Ben Ahmed Mohamed, and Boudhir Anouar Abdelhakim. A new architecture based on convolutional neural networks (cnn) for assisting the driver in fog environment. In *Proceedings of the 3rd International Conference on Smart City Applications*, page 85. ACM, 2018.

[15] Kaijing Shi, Hong Bao, and Nan Ma. Forward vehicle detection based on incremental learning and fast r-cnn. In *2017 13th International Conference on Computational Intelligence and Security (CIS)*, pages 73–76. IEEE, 2017.

[16] C Vishnu, Dinesh Singh, C Krishna Mohan, and Sobhan Babu. Detection of motorcyclists without helmet in videos using convolutional neural network. In *2017 International Joint Conference on Neural Networks (IJCNN)*, pages 3036–3041. IEEE, 2017.

[17] Lixin Wang, Junguo Liao, and Chaoqian Xu. Vehicle detection based on drone images with the improved faster r-cnn. In *Proceedings of the 2019 11th International Conference on Machine Learning and Computing*, pages 466–471. ACM, 2019.