Distracted Pedestrian Behaviour Classification

Sarwan Shah, Muhammad Abdullah Siddiqui, Waleed Bin Khalid
EE Class of 2021, Habib University

***Abstract*— This paper explores existing techniques for pedestrian detection and improves upon them in order to identify 'distracted' pedestrians that may be in danger, and suggests possible solutions to alert them. The main algorithms that are explored are Faster R-CNN, HOG based neural networks, General CNNs and Transfer learning algorithms.**

I. INTRODUCTION

The motivation behind this project is the rising concern of life safety that is associated with the crossing of road by pedestrians with certain distractions. According to a study by the UK government, 78.7 percent of pedestrian casualties in 2018 were on pedestrian crossings. According to the Swiss Accident Prevention Agency, one third of all traffic accidents in Switzerland involved pedestrians on a zebra crossing.

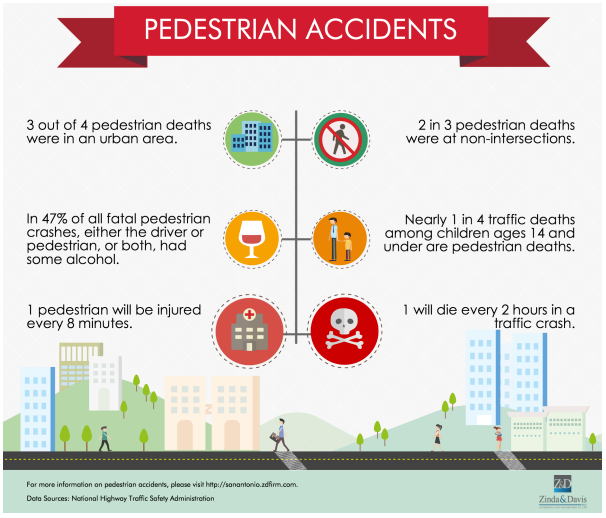

Fig. 1. Some statistics about pedestrians [10]

Up till now, a lot of work has been done to make sure that distracted drivers do not end up in an accident situation which can be lethal for a pedestrian. This work has spanned both technological and legislative solutions and some progress has been made certainly. But the fact of the matter is that something needs to be done for the distracted pedestrian as well who is totally unaware of his surroundings when he is crossing a road and hence can end up with serious injuries. Statistics show that the majority of accidents involving pedestrians take place on zebra crossings due to complacency of the individuals.

Our main goal in this project is to carry out detection of such distracted pedestrians, and recommend some alert mechanisms in order to prevent such accidents from happening.

II. LITERATURE REVIEW

The literature review aspect of the project was very enlightening. It helped us ground our ideas, which were a lot more ambitious when we started out the proposal. Some of the main conclusions we drew from the literature review portion are as follows:

- The HOG descriptor is rigid and not optimal for dealing with objects with articulated parts. For example, if pedestrians are partially occluded or are visible only from waist up due to proximity to camera, the detection accuracy is affected. This stems from the fact that bounding boxes annotated by humans aren't perfectly consistent and hence is one of the reasons why we need a solution akin to deformable parts model [1].
- One major way a paper inspired our project was how they recorded videos, divided it into frames and used it to train their detector. This was highly relevant to us because in practical applications for our algorithm we will of course be getting real live camera footage that will need to be evaluated by our program in real time to generate alerts [3].
- There are several different performance evaluation metrics that can be employed to evaluate our algorithms e.g. the per-window system, the per-image system, plot true positive rate versus false positive rate, plot miss rate versus false positive per window, plot precision versus recall curves etc [7].
- A joint deep learning model can be explored to combine feature extractors, deformation handling and then use classification to perform pedestrian detection [9].
- HOG can be really productive with stereo infrared images particularly for window building around pedestrians [1].
- The dataset construction will play a very key role, since image sizes, aspect ratios, scales and rotational and translational features of a dataset will be very crucial for a learning and testing problem which involves pedestrian detection and classification of the posture and position.

III. APPROACH

The first step was to acquire a variety of datasets concerning pedestrians from online sources. Then we shortlisted all datasets and images related to individual pedestrians only.

These datasets were then sorted and subsequently augmented to different degrees. The datasets were inputted to a variety of algorithms as outlined in the 'Experiments, Analysis and Results' section. The architrecure, image sizes and other hyperparameters were varied until an optimal result was obtained. The results were obtained via running the algorithm on the test data.

Finally, we combined the Fast R-CNN output with the other algorithms such that Fast R-CNN outputted individual pedestrian images which were filtered to remove false positives, and inputted to the algorithms concerning individual pedestrians to label them. In essence, this simulates a camera taking snapshots of an environment, identifying pedestrians, and categorizing the status of the pedestrian.

This is an alternative to the object detection algorithm directly determining what kind of pedestrian is showcased in the image. We were unable to do this because training such a model was simply taking so much time, and we were short on annotated data as well.

IV. DATA SET

A. Data Collection

We collected data from a variety of different sources. Most of the sources were pedestrian datasets online, which were sorted as per our convenience. The datasets that we looked into and made our own set were, Caltech data set, Inria Dataset, PETA Dataset, PennPad Dataset and a few unnamed datasets that we found via research papers on kaggle and google data search.

Some of the test data images were obtained from google images in the interest of them being completely novel.

B. Classes

Choosing classes was a very tough task for us. There is lots of individual pedestrian image data available online but since no significant work has been carried out in the domain of distracted pedestrians, we were unable to find any annotated or labelled data that we could use. About 20k+ images were sorted to get the dataset for our usage.

Therefore, we relied on our understanding of theoretical computer vision concepts and our conclusions from the various papers we read during research to determine a few classes.

The initial classes we created were titled 'Alert', 'Distracted', 'UpMobile', 'DownMobile', 'Carrier'. Here 'Alert' and 'Distracted' are self explanatory and were classified as an image appears to a naked eye. 'UpMobile' corresponds to someone using their phone to take a call i.e. the phone is upto their ear. 'DownMobile' corresponds to someone browsing their phone while walking i.e. the mobile is held out at roughly waist level. 'Carrier' corresponds to a person carrying a handbag or some other equipment.

Trying to implement our algorithm with this immediately ran into problems. The preliminary accuracy we obtained was about 7% which seemed unacceptable. Upon looking at the results more closely, we concluded that there was limited to no detection going on and even the 7% correctly labelled was by chance.

After some more research, we decided to be choosier with how we labelled our images into a category. First of all we removed the category of 'Carrier' because there was too much variety in the type of equipment being carried, and the pose in which the equipment was held, for it to be applicable at this early stage of algorithm development.

We further restricted our definition of each class, since our initial sorting was very arbitrary and had little common in terms of features. We set out to resolve this such that each category had clearly overlapping features that would make it easier for the computer to detect it.

'Alert' was now labelled to an individual pedestrian staring straight ahead. 'Distracted' was the label given to a person looking sideways or downwards. 'UpMobile' and 'DownMobile' were maintained as before. One paradox this approach presented us with was that if a pedestrian is looking sideways, it is hard to determine if he is 'Alert' or 'Distracted' in practical terms. This has not been resolved in this paper due to practical limitations. In actual applications we suggest that the determination be made by the camera observing what the pedestrian is looking at. If the pedestrian is looking sideways at an incoming car, he/she should be classified as 'Alert' and otherwise as 'Distracted'.

C. Data Manipulation

The manual sorting of images was done on individual pedestrian data from a variety of different datasets. This means that our data was bolstered by the fact that it had several different backgrounds, and different varieties of poses and pedestrian types.

After multiple iterations of going through loads of data to sort through them, our final count for images was about 1300 images. These were first of all divided into training and test data. We maintained 122 images for our test data which had 32 images in 'Alert' category, 27 images in 'Distracted' category, 32 images in 'DownMobile' category and 31 images in 'UpMobile' category. The total amount of training data was 1156 images which had 432 images in 'Alert' category, 274 images in 'Distracted' category, 265 images in 'DownMobile' category and 185 images in 'UpMobile' category.

For more deployment and practical purposes, we also used data from the output of the second approach mentioned in the Faster RCNN algorithm section of the paper. These were a total of 67 extracted pedestrian images which were sorted into the classes and another set of testing result was carried on this. From a practical point of view, this gives a good illustration of the framework we are trying to build. Because classification algorithms classify a pedestrian image, but for images with multiple pedestrians and objects we first detect the pedestrian object, classify it and then label the objects in the image using the classification results. A detailed analysis of this is mentioned later in the paper.

The data earmarked for training was further split into 85% being the training data and the other 15% being the validation data.

We carried out augmentation on the training data or x_train so as to bolster our training data. There were multiple iterations of augmentations, and the ones being used in the final algorithms are 'AugmentedData3', 'AugmentedData4' and 'AugmentedData5'. The augmentations involved flipping the image, taking grayscale versions of the image, adjusting the brightness, constrast or intensity. Different classes were augmented to increase their quantity by different magnitudes. The guiding principle being the fact that our final data needs to be balanced in between classes. The number of images in each augmented set were about 3000, 5500 and 12500 respectively, evenly distributed in the 4 classes.

V. EXPERIMENTS, ANALYSIS AND RESULTS

A. Pedestrian Extraction with Faster R-CNN

In this part, what we did was to use the Faster-RCNN algorithm, to detect the objects. There were two approaches for us in this arena.

The first one being, classifying the detected objects by directly using Faster-RCNN. This is a more standard approach, but our results were not very satisfactory. To our surprise not even a single object was detected and hence there was no classification. Now, our analysis for this situation is that, to detect and classify objects using faster RCNN requires very precise data, data that is not only variant in terms of featurues but also is very well annotated. Due to our lack of annotated data for the classes we decided earlier, we decided to take this task and annotated around 300 samples of the pedestrian images of the 4 classes, but setting appropriate bounding box around them. And then used this data for training and then used images of bigger scaled containing multiple objects (instances of our classes). The results were a total failure since image sizing was an issue. The data was not enough and the annotations were not perfect.

This lack of results eventually made up our second idea which turned out to be very successful in comparison. The second idea was just using the Faster RCNN algorithm to detect the objects which are "pedestrians". So the algorithm successfully made bounding boxes for every pedestrian that it saw in the test image. The training data for this was available with annotations. Some manipulations of the annotations was still done to get it into the required form for the algorithm to run. Once the bounding boxes were made, further programming to save the coordinates was done. Now using library and self made function, a careful code was written which would extract the pedestrian image out of a full image, then the 4 algorithms that were developed in this project were used to classify the pedestrian behaviour, and then by keeping track of the coordinates and the image name, the bounding box were remade but now with a class label. Figure 2 shows a sample output of the Faster RCNN algorithm.

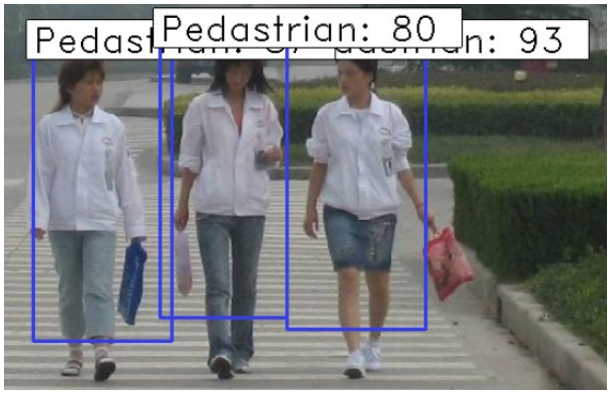

Fig. 2. Faster RCNN results of bounding boxes on a test image

B. Behaviour Classification using Neural Network

This was the starting point of our methodoloy, where we just used the simple Neural Network technique for classifying the images. As the demand of the network is all the images were flattened and then a neural network architecture was built. The architecture details can be found in the appendix section.

Before reaching this final architecture multiple optimizers, number of layers, neurons, dropout values, regularization options were explored and then a final architecture was chosen. The experimentation with neural networks, did not improve the results a lot and the difference between the choice of different combinations of hyper parameters and optimizers was not that significant to be put into a detailed comparison as done for CNN, and transfer learning. The final architecture that was chosen can be seen in the appendix files.

But this architecture was ran on 3 different training and validation set pairs, as in all of the cases for all algorithms mentioned in this paper.

The image size that was chosen for this model by 32x96 since a lot of promising results came out from this size. Not only this, roughly all the images in the dataset, were close to this size and resizing to this size did not have a lot of effect on the image quality.

The results for the three data sets are shown in the figure below:

Neural Network Results				
Train Set	Accuracy Train	Accuracy Validation	Accuracy test (original pedestrian images)	Accuracy test (extracted pedestrian images)
Augment3	95	85	23	22
Augment4	98	98	22	12
Augment5	92	92	24	12

Fig. 3. Neural Network Results

C. Behaviour Classification using Neural Network on extracted HOG feature vectors

A very common approach is to use feature extractors before passing an image to a neural network. So we decided to use Hog feature vectors, instead of flattened image vectors

to perform the same neural network classification and the parameters and the model of the neural network built earlier, was used here as well.

The results for the three datasets are as shown in the figure below:

Neural Network results on HOG feature vectors				
Train Set	Accuracy Train	Accuracy Validation	Accuracy test (original pedestrian images)	Accuracy test (extracted pedestrian images)
Augment3	95	90	28	44
Augment4	95	95	29	28
Augment5	93	93	22	7

Fig. 4. Neural Network Results on HOG feature Vectors

D. Behaviour Classification using CNN

Feature Extraction based Neural Networks have been around with us for over three decades now. However, recent advancements in computational power over the past decade have paved the way for a more powerful tools to emerge in the form for Convolutional Neural Networks (CNN). These immediately gained popularity, especially with the advent of AlexNet architecture in 2012, which demonstrated its power on the ImageNet Classification Challenge, with a huge improvement in classification accuracy compared to Feature Extraction based Neural Networks.

Naturally, the next step in our attempt to classify pedestrian behavior was to construct a Convolutional Neural Network optimized to the task. We utilized the Google Colab's GPU enviroment along with Keras library for this task.

After several iterations of design changes, hyper-parameter variations and testing the following CNN Architecture was finalized as to be best optimized for our task:

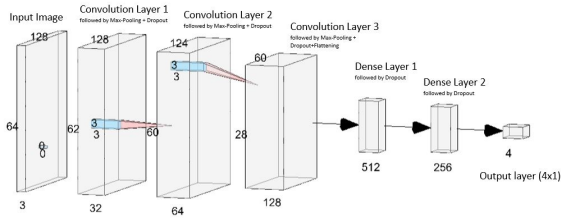

Fig. 5. Proposed CNN Architecture

The final architecture consisted of a total of 27,751,236 trainable parameters. All the convolution layers consisted of 3x3 filters and 'relu' activation. Each of the convolution layers were followed by a Max Pooling layer of stride 2, as well as a Dropout layer for regularization. The final output layer used 'softmax' activation.

The improvement in results from the initial to the final architecture are on the AugmentedData4 are tabulated below:

Accuracy	Initial	Final
Validation	70%	92%
Test	33%	60%

The results with AugmentedData3 and AugmentedData5 were generally poorer in comparison. AugmentedData3 gave a lower validation accuracy as well as a lower test accuracy. This was partly because CNN struggled to train its weights sufficiently due to the limited amount of overall data on AugmentedData3. On the other hand, AugmentedData5's heavy augmentation resulted in the model often getting over-fitted and loss of generalization, which was reflected in very high validation accuracy, but a low test accuracy, similar to AugmentedData3.

Furthermore, using categorical cross entropy as the loss function yield the best results. The following optimizers were tested with: SGD, Adagrad, and Adam. 'SGD' performed the best, while 'Adam' followed with only a slightly lower test accuracy in advantage of faster training times, whereas 'Adagrad' in general struggled. The results on AugmentedData4 for these 3 on the test data are tabulated below:

Accuracy	Test Accuracy
SGD	60%
Adam	58%
Adagrad	54%

A major challenge throughout the process was stopping the architecture from over fitting on the train set. To cater to this end over the optimization process, the L2 activity regularization was carried on some of the layers to penalize the training weight for too large changes, in addition to the dropout layers. This seems to have largely improved the training stability and accuracy.

Another consideration made was the batch size, as it also seemed to effect the quality of training and best results were achieved for a batch size of 16. In comparison batch sizes 8 and 32 or more, yield poorer results. Similarly, an image size 64x128 gave the best results with the proposed CNN. The tuning of these various parameters and hyper-parameters sufficiently resulted in the final proposed architecture at 40 epochs, SGD optimizer, with dropouts varying from 0.4 to 0.6. The training results for this architecture are given below:

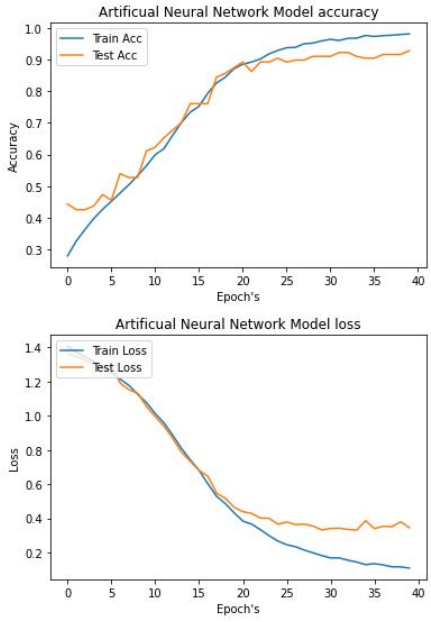

Fig. 7. Test Set Results

One the challenges faced was decreasing the training vs validation loss, and in the given time period we could not come up a fine tuned architecture that could reduce it any further. While all in all, we were satisfied with the resulted yielded by this CNN. We still strongly believe that the limited size of our training dataset was the key point that saturated our test accuracy to roughly around 60%, and with more training data, significantly better results could be achieved with this same architecture.

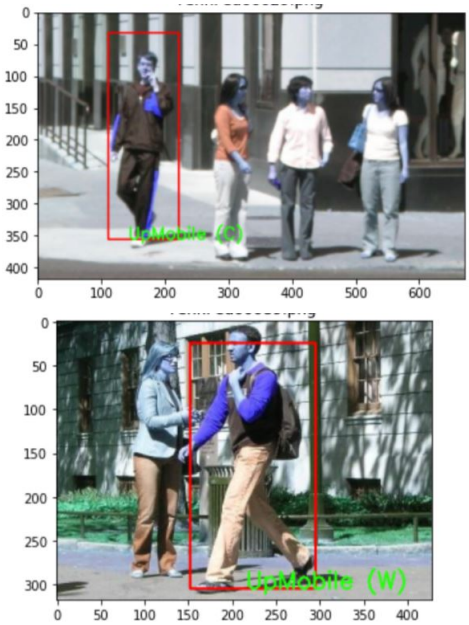

Fig. 8. Test Set Results with Image Annotations

E. Behaviour Classification using Transfer Learning

Transfer learning is a well known technique in which we exploit the availability of pre-trained algorithms with set weights to use as feature extractors for our convolutional neural networks and improve our classification. They are especially useful in cases when we have sparse data to help the model learn features.

We intended to explore a variety of algorithms but due to limitations of time, we were only able to explore a VGG-16 implementation. The architecture employed was simple, obtained after multiple trial and error and is believed to give optimal performance on the majority of hyper-parameter combinations and data-sets .

Flatten $\rightarrow$ Batch Normalization $\rightarrow$ Dense(2048) $\rightarrow$ Dense(1024) $\rightarrow$ Dense(512) $\rightarrow$ Dense(256) $\rightarrow$ Dense(4)

The number in the brackets signify the number of neurons. Dropout layers were placed after each dense layer. A variety of activations were used but we stuck with 'relu' for all dense layers accept last one which had 'softmax'.

All images were resized to a common set of dimensions before applying training/testing. The main orientation used were 32x32, 32x 96, 64x64, 64x128 and 48x144. Based on our literature review, we expected 64x128 to give the highest accuracy but for the configurations outlined, 32x96 sized image performed the best.

Image Size	AugData3	AugData4	AugData5
32 x 96	58%	63%	57%
64 x 128	52%	54%	51%
48 x 144	57%	58%	57%

Fig. 9. Accuracy varying with image size

We explored different batch sizes i.e. 8, 16, 32 and 64. The highest accuracy was obtained almost each time with a batch size of 16.

We used both binary_crossentropy loss and categorical_crossentropy loss functions in our attempts. Very poor and 'linearized' results were obtained with binary_crossentropy loss. Below is a graph obtained for binary cross entropy run to 20 epochs. The graphs for categorical crossentropy can be found later in this section or in the appendix.

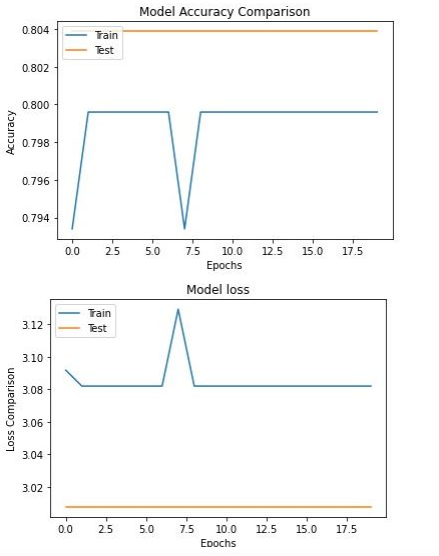

Fig. 10. Graphs for binary_crossentropy

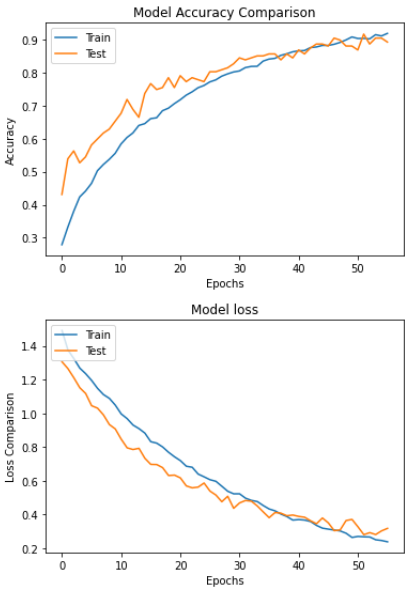

Fig. 12. Accuracy and Loss Graph for AugmentedData4

As for optimizers, we focused on changing between 'Adagrad', 'Adam' and 'SGD'. In general if we vary the learning rate, each optimizer performed best are 'lr = 0.01'.

In general, 'SGD' gave us the best results though 'Adam' was very competitive with AugmentedData3. This makes sense since 'Adam' is supposed to be especially good for sparse data.

Optimizer	Accuracy
SGD	63%
Adagrad	58%
Adam	57%

Fig. 11. Accuracy varying with optimizer for AugData4.

One interesting trend we noticed was that as we move from AugmentedData3 to AugmentedData5 (i.e. the number of images increasing), the training/validation graph trend improves but the test accuracy decreases slightly. Between the three different augmented datasets, AugmentedData4 in general gives the best compromise in terms of training/validation graph trend as well as on teh test data, in the least amount of time i.e. number of epochs.

The best accuracy obtained on the test data was 62.5% using AugmentedData4 at 56 epochs, with SGD optimizer, with dropout layers set at 0.4. Accuracy values close to 60% were obtained with other datasets and/or architecture at double or more number of epochs, so this configuration stands out. The graph corresponding to this performance is shown below:

Category	Accuracy	Ratio	FP's:	Alert	Distracted	UpMobile	DownMobile
Overall Accuracy:	61 %	69/112					
Alert Accuracy:	75 %	24/32			6.25 %	9.38 %	9.38 %
Distracted Accuracy:	75 %	15/20		20.0 %		0.0 %	5.0 %
UpMobile Accuracy:	48 %	15/31		22.58 %	12.9 %		16.13 %
DownMobile Accuracy:	51 %	15/29		13.79 %	13.79 %	20.69 %	

Fig. 13. Tabulated Accuracy Results

This configuration gave the best results with the test data on the objects outputted by R-CNN as well i.e. about 52%. We can then proceed to output the label on the original image with the bounding box to show something like follows:

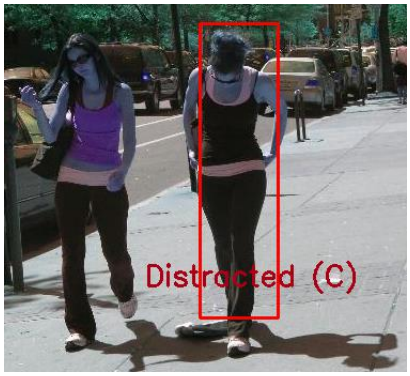

Fig. 14. Correctly Labelled Bbox

Fig. 15. Wrongly Labelled Bbox

Note thate here 'C' corresponds to correctly labelled and 'W' corresponds to wrongly labelled data.

F. Comparison Between All 4 Algorithms

The table below summarizes a comparison of the different algorithms performing on the two test data. The standalone data is identical to training images and was split from that randomly. The extracted data consists of the boundary boxes of individual pedestrians obtained from the Faster-RCNN algorithm.

The notation here is that TL refers to Transfer Learning. HOG + NN refers to Neural Network on extracted HOG Features. While Neural Net and CNN are the algorithms in their general forms.

Algorithm	Standalone Data	Extracted Data
Neural Net	23%	22%
HOG + NN	28%	44%
ConvNeuralNet	60%	33%
TL (VGG16)	62%	52%

We can observe that on the whole, the transfer learning approach yields the best results outstripping the rest. Simple Neural Networks yields the worst results. HOG feature extractor works better than CNN on R-CNN extracted data but worse than CNN on the standalone data. This leads us to believe that we will need more varied test data to test our algorithms on, to properly classify which of HOG+NN and CNN is the better performing option.

VI. RECOMMENDED SOLUTIONS

This paper explores a computer vision solution to distracted pedestrian detection that can, in a practical scenario, allow a system to make intelligent decisions in order to generate alerts to such a person and prevent an accident.

Although not strictly within the scope of this paper, we propose that visual, auditory and sensationary alerts may be generated in order to ensure that a pedestrian is shaken our of lethargy while attempting to cross a road. Feel free to refer to us for further elaboration upon this proposed solution.

VII. FUTURE CONSIDERATIONS & DIRECTION

We believe that we have merely scratched the surface of the potential that lies in this particular domain of pedestrian detection. There is lots of room for improvement and addition before this can be launched for a practical application.

For one thing, we of course need a much much larger collection of dataset. Alongside that, we need much more categories. Currently we obtained about 1300 images from 20k+ images. Part of that is because most of the images simply didn't fit into the 4 categories that we had devised. Some major categories that can be accounted for and would undoubtedly expand the dataset is people carrying equipment/handbags, people running, people wearing headphones/earphones etc.

One more possible improvement for our algorithms and dataset, which couldn't be implemented due to time limitations, is to add a category of empty background. An empty image with a corresponding label will serve to ensure that our algorithm is properly learning to recognize pedestrians.

Another additional aspect to consider is that currently our algorithm is only working for individual pedestrian detection. Ideally it should consider the environment as well to determine whether a pedestrian is alert or distracted. For example to record whether a pedestrian is looking sideways at an oncoming car or a billboard.

VIII. CONCLUSION

In conclusion, we have seen the behaviour of a variety of different algorithms with differently augmented datasets and several different architectres. In general, transfer learning with VGG16 can be taken to be the best performing algorithm. But we hesitate to state this claim outright until we have done more testing and on a larger dataset.

We believe we have also added a sorely missing dimension of novelty to the pedestrian detection area. A lot of work has been conducted on how to address distracted driving and reduce accidents, but identifying distracted pedestrians is an equally significant area for preserving human life. With the advent and ever-increasing usage of the mobile phone in particular, we believe this application will be an even bigger need in the future, and we for one are very excited at investigating this domain.

IX. Appendix

A. Data Collection and Augmentation

The various data folders that were used can be found on the following link: https://drive.google.com/drive/folders/1voQtKTz6erhE5FhXxGKFEqQkUR7GB4Jc?usp=sharing

The code for data augmentation can be found on the following link: https://colab.research.google.com/drive/1UrjRdBj6PLV7B5FsWZTVFpya1sIsVeBO?usp=sharing

B. Pedestrian Extraction with Faster R-CNN

The results of the faster RCNN algorithm are shown below. Alongside a few sample images. A total of 100 Training Images and 60 Test Images were used. The code and method can be found on this : https://colab.research.google.com/drive/1gj9Ini33HBrtaZVBwXZC6I2IsUZmD2zH?usp=sharing

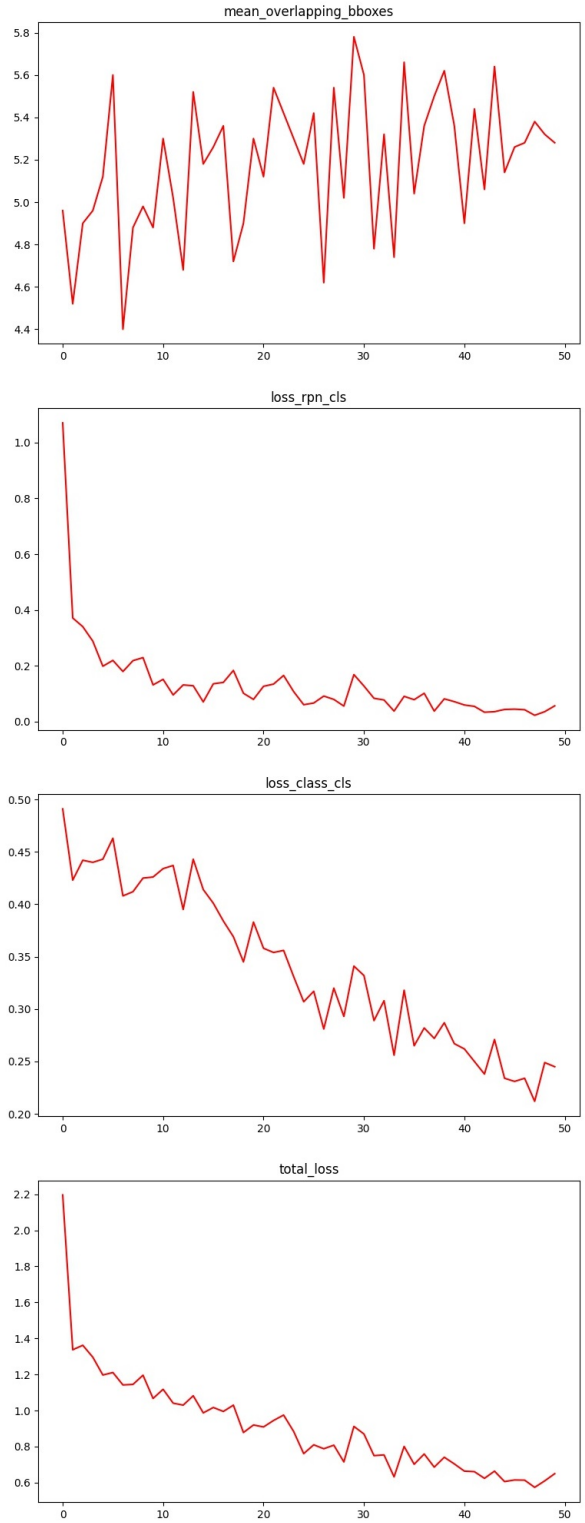

Fig. 16. Faster RCNN Results

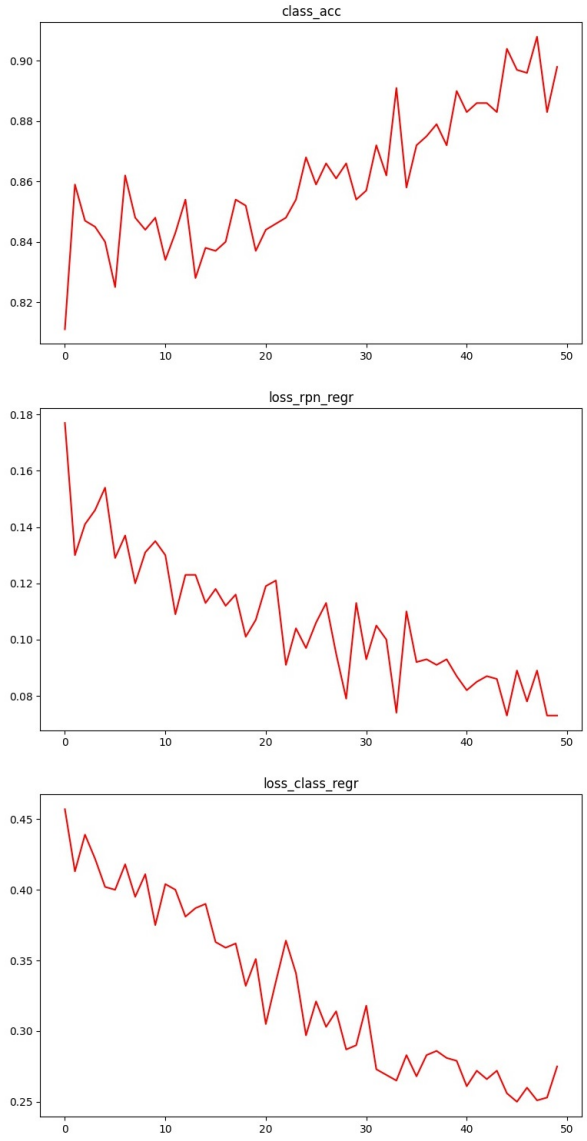

Fig. 17. Faster RCNN Results

C. Extracting Pedestrian Images from full images

The results for this code are as shown below. The details of the method can be fond in the colab link: https://colab.research.google.com/drive/1XDO2-a5ClMoxe91_d50OLK1iXUmeW4T2?usp=sharing

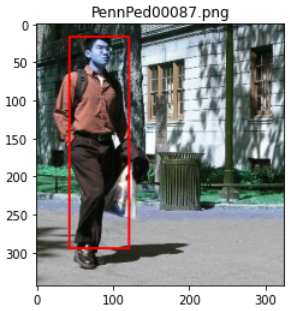

Fig. 18. Full Image

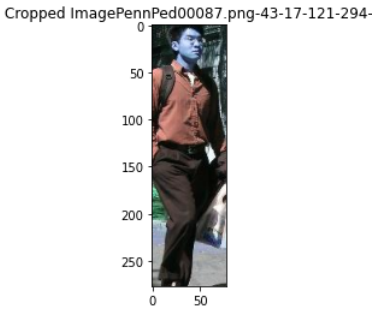

Fig. 19. Extracting Pedestrian Image from Full Image

D. Behaviour Classification using Neural Network

The code and method can be found using the link, https://colab.research.google.com/drive/1GAaIOgAgke_NFo4dVVTNVTGri3CJRbx3?usp=sharing

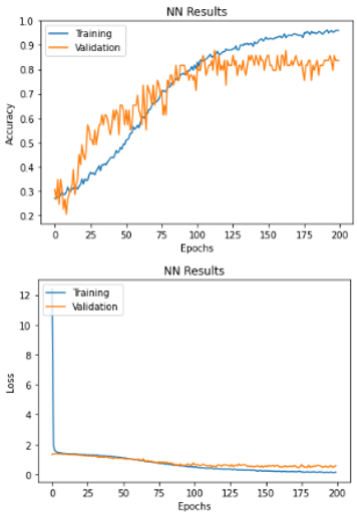

Fig. 20. Training Results with Augmented Data3

Category	Accuracy	Ratio	FP's:	Alert	Distracted	UpMobile	DownMobile
Overall Accuracy:	23 %	26/112					
Alert Accuracy:	21 %	7/32			53.12 %	15.62 %	9.38 %
Distracted Accuracy:	30 %	6/20		30.0 %		10.0 %	30.0 %
UpMobile Accuracy:	19 %	6/31		22.58 %	45.16 %		12.9 %
DownMobile Accuracy:	24 %	7/29		20.69 %	37.93 %	17.24 %	

Fig. 21. Test Results with Augmented Data 3

NN TESTING ON Object Test data 1 (testing dataof extracted objects)

Category	Accuracy	Ratio	FP's:	Alert	Distracted	UpMobile	DownMobile
Overall Accuracy:	22 %	14/63					
Alert Accuracy:	13 %	5/37			27.03 %	59.46 %	0.0 %
Distracted Accuracy:	31 %	5/16		25.0 %		43.75 %	0.0 %
UpMobile Accuracy:	66 %	4/6		33.33 %	0.0 %		0.0 %
DownMobile Accuracy:	0 %	0/4		75.0 %	25.0 %	0.0 %	

Fig. 22. Test Results with Augmented Data 3 on Extracted Objects

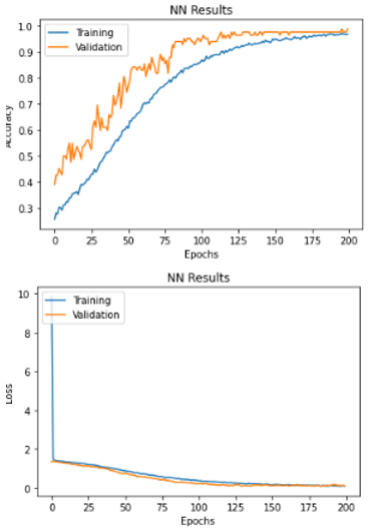

Fig. 23. Training Results with Augmented Data4

Category	Accuracy	Ratio	FP's:	Alert	Distracted	UpMobile	DownMobile
Overall Accuracy:	22 %	25/112					
Alert Accuracy:	3 %	1/32			6.25 %	75.0 %	15.62 %
Distracted Accuracy:	5 %	1/20		10.0 %		65.0 %	20.0 %
UpMobile Accuracy:	58 %	18/31		12.9 %	3.23 %		25.81 %
DownMobile Accuracy:	17 %	5/29		6.9 %	10.34 %	65.52 %	

Fig. 24. Test Results with Augmented Data 4

Category	Accuracy	Ratio	FP's:	Alert	Distracted	UpMobile	DownMobile
Overall Accuracy:	12 %	8/63					
Alert Accuracy:	2 %	1/37			2.7 %	94.59 %	0.0 %
Distracted Accuracy:	0 %	0/16		0.0 %		93.75 %	6.25 %
UpMobile Accuracy:	100 %	6/6		0.0 %	0.0 %		0.0 %
DownMobile Accuracy:	25 %	1/4		0.0 %	0.0 %	75.0 %	

Fig. 25. Test Results with Augmented Data 4 on Extracted Objects

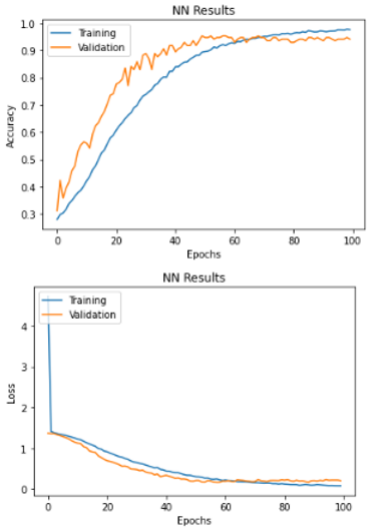

Fig. 26. Training Results with Augmented Data5

Category	Accuracy	Ratio	FP's:	Alert	Distracted	UpMobile	DownMobile
Overall Accuracy:	24 %	27/112					
Alert Accuracy:	0 %	0/32			15.62 %	15.62 %	68.75 %
Distracted Accuracy:	5 %	1/20		0.0 %		45.0 %	50.0 %
UpMobile Accuracy:	32 %	10/31		3.23 %	12.9 %		51.61 %
DownMobile Accuracy:	55 %	16/29		10.34 %	27.59 %	6.9 %	

Fig. 27. Test Results with Augmented Data 5

Category	Accuracy	Ratio	FP's:	Alert	Distracted	UpMobile	DownMobile
Overall Accuracy:	12 %	8/63					
Alert Accuracy:	16 %	6/37			21.62 %	35.14 %	27.03 %
Distracted Accuracy:	0 %	0/16		12.5 %		68.75 %	18.75 %
UpMobile Accuracy:	16 %	1/6		0.0 %	33.33 %		50.0 %
DownMobile Accuracy:	25 %	1/4		25.0 %	25.0 %	25.0 %	

Fig. 28. Test Results with Augmented Data 5 on Extracted Objects

E. Behaviour Classification using Neural Network on extracted HOG feature vectors

The code and method can be found using the link, `https://colab.research.google.com/drive/1GAaIOgAgke_NFo4dVVTNVTGri3CJRbx3?usp=sharing`

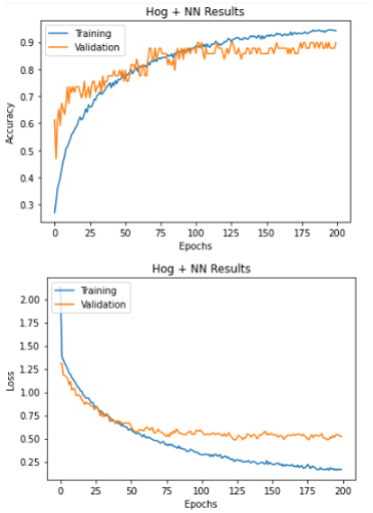

Fig. 29. Training Results with Augmented Data3

Category	Accuracy	Ratio	FP's:	Alert	Distracted	UpMobile	DownMobile
Overall Accuracy:	28 %	32/112					
Alert Accuracy:	3 %	1/32			0.0 %	0.0 %	96.88 %
Distracted Accuracy:	0 %	0/20		0.0 %		5.0 %	95.0 %
UpMobile Accuracy:	6 %	2/31		0.0 %	0.0 %		93.55 %
DownMobile Accuracy:	100 %	29/29		0.0 %	0.0 %	0.0 %	

Fig. 30. Test Results with Augmented Data 3

Category	Accuracy	Ratio	FP's:	Alert	Distracted	UpMobile	DownMobile
Overall Accuracy:	44 %	28/63					
Alert Accuracy:	45 %	17/37			27.03 %	16.22 %	10.81 %
Distracted Accuracy:	56 %	9/16		12.5 %		18.75 %	12.5 %
UpMobile Accuracy:	33 %	2/6		33.33 %	0.0 %		33.33 %
DownMobile Accuracy:	0 %	0/4		50.0 %	25.0 %	25.0 %	

Fig. 31. Test Results with Augmented Data 3 on Extracted Objects

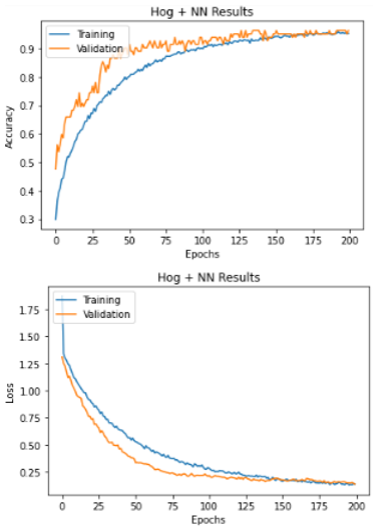

Fig. 32. Training Results with Augmented Data4

Category	Accuracy	Ratio	FP's:	Alert	Distracted	UpMobile	DownMobile
Overall Accuracy:	29 %	33/112					
Alert Accuracy:	87 %	28/32			6.25 %	0.0 %	6.25 %
Distracted Accuracy:	10 %	2/20		80.0 %		0.0 %	10.0 %
UpMobile Accuracy:	0 %	0/31		96.77 %	0.0 %		3.23 %
DownMobile Accuracy:	10 %	3/29		86.21 %	3.45 %	0.0 %	

Fig. 33. Test Results with Augmented Data 4

Category	Accuracy	Ratio	FP's:	Alert	Distracted	UpMobile	DownMobile
Overall Accuracy:	28 %	18/63					
Alert Accuracy:	37 %	14/37			2.7 %	10.81 %	48.65 %
Distracted Accuracy:	6 %	1/16		12.5 %		25.0 %	56.25 %
UpMobile Accuracy:	16 %	1/6		33.33 %	0.0 %		50.0 %
DownMobile Accuracy:	50 %	2/4		25.0 %	0.0 %	25.0 %	

Fig. 34. Test Results with Augmented Data 4 on Extracted Objects

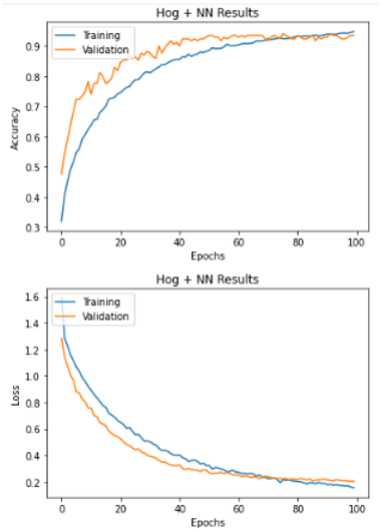

Fig. 35. Training Results with Augmented Data5

Category	Accuracy	Ratio	FP's:	Alert	Distracted	UpMobile	DownMobile
Overall Accuracy:	22 %	25/112					
Alert Accuracy:	9 %	3/32			50.0 %	15.62 %	25.0 %
Distracted Accuracy:	45 %	9/20		5.0 %		10.0 %	40.0 %
UpMobile Accuracy:	16 %	5/31		16.13 %	41.94 %		25.81 %
DownMobile Accuracy:	27 %	8/29		13.79 %	48.28 %	10.34 %	

Fig. 36. Test Results with Augmented Data 5

Category	Accuracy	Ratio	FP's:	Alert	Distracted	UpMobile	DownMobile
Overall Accuracy:	7 %	5/63					
Alert Accuracy:	0 %	0/37			24.32 %	0.0 %	75.68 %
Distracted Accuracy:	12 %	2/16		0.0 %		0.0 %	87.5 %
UpMobile Accuracy:	0 %	0/6		0.0 %	50.0 %		50.0 %
DownMobile Accuracy:	75 %	3/4		0.0 %	25.0 %	0.0 %	

Fig. 37. Test Results with Augmented Data 5 on Extracted Objects

F. Behaviour Classification using CNN

The relevant code and methods can be found in this Colab Notebook: `https://colab.research.google.com/drive/1Rw6hiljP-XKWW012bCOnpHy44odhXyns?usp=sharing`

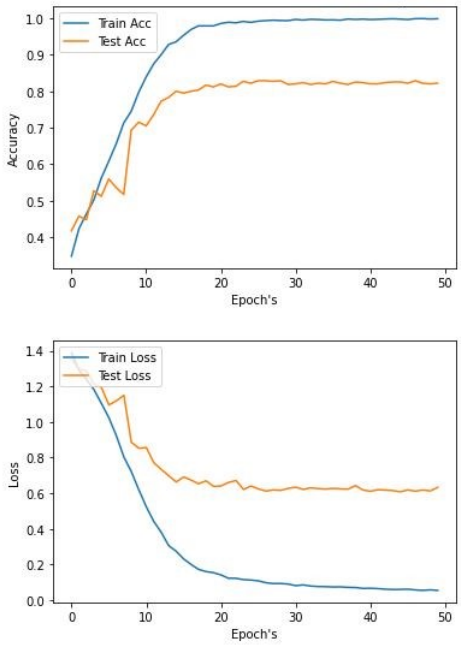

Fig. 38. Training Results with AugmentedData3

Overall Accuracy:	52 %	59/112
Alert Accuracy:	75 %	24/32
Distracted Accuracy:	60 %	12/20
UpMobile Accuracy:	35 %	11/31
DownMobile Accuracy:	41 %	12/29

Fig. 39. Test Results with Augmented Data 3

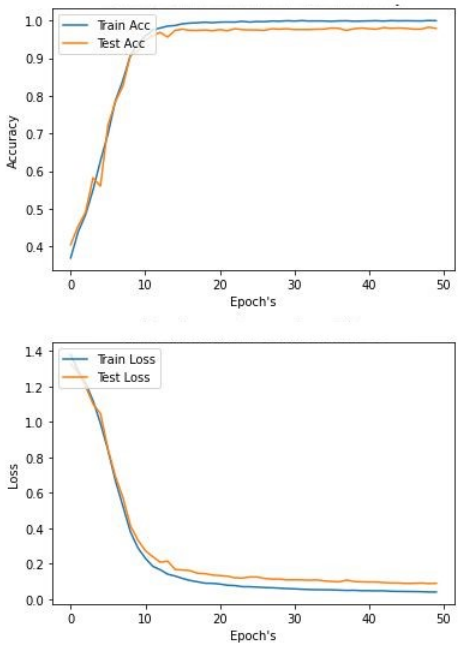

Fig. 40. Training Results with AugmentedData5

Overall Accuracy:	52 %	59/112
Alert Accuracy:	78 %	24/32
Distracted Accuracy:	65 %	12/20
UpMobile Accuracy:	38 %	11/31
DownMobile Accuracy:	55 %	12/29

Fig. 41. Test Results with AugmentedData5

G. Behaviour Classification using Transfer Learning

The code for transfer learning on colab can be found on the link: `https://colab.research.google.com/drive/1yz0CHqBnEzMUUWjZVmac3x4VAdQZ-kQa?usp=sharing`

Some of the results obtained are outlined below as an example.

The algorithm was applied on the standalone test data, and not the data outputted from object detection algorithm.

The configuration for the three sets of results below was with optimizer SGD, batchsize = 16, categorical crossentropy loss function and dropout layers set at 0.4.

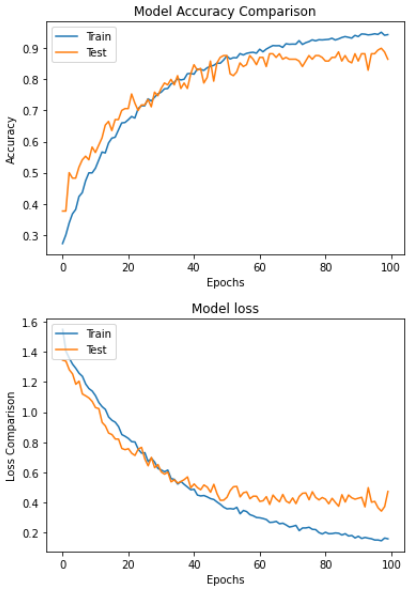

Fig. 42. AugmentedData3 Graphical Results

Category	Accuracy	Ratio	FP's:	Alert	Distracted	UpMobile	DownMobile
Overall Accuracy:	62 %	70/112					
Alert Accuracy:	75 %	24/32			6.25 %	9.38 %	9.38 %
Distracted Accuracy:	70 %	14/20		20.0 %		0.0 %	10.0 %
UpMobile Accuracy:	48 %	15/31		22.58 %	12.9 %		16.13 %
DownMobile Accuracy:	58 %	17/29		6.9 %	13.79 %	20.69 %	

Fig. 43. AugmentedData3 Tabular Results

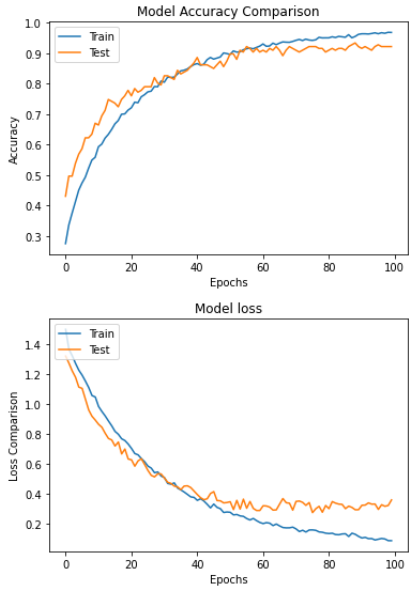

Fig. 44. AugmentedData4 Graphical Results

Category	Accuracy	Ratio	FP's:	Alert	Distracted	UpMobile	DownMobile
Overall Accuracy:	61 %	69/112					
Alert Accuracy:	75 %	24/32			3.12 %	9.38 %	12.5 %
Distracted Accuracy:	70 %	14/20		25.0 %		0.0 %	5.0 %
UpMobile Accuracy:	54 %	17/31		25.81 %	12.9 %		6.45 %
DownMobile Accuracy:	48 %	14/29		17.24 %	17.24 %	17.24 %	

Fig. 45. AugmentedData4 Tabular Results

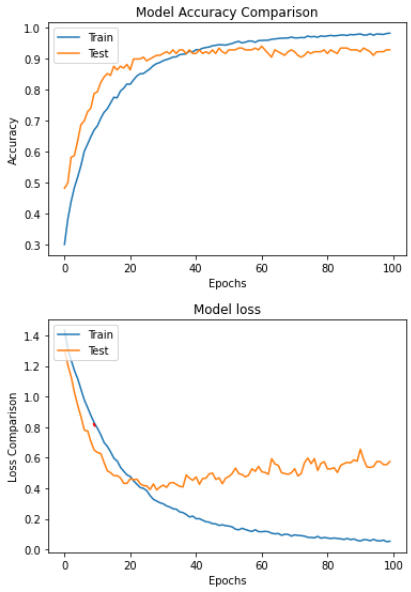

Fig. 46. AugmentedData5 Graphical Results

Category	Accuracy	Ratio	FP's:	Alert	Distracted	UpMobile	DownMobile
Overall Accuracy:	52 %	59/112					
Alert Accuracy:	75 %	24/32			6.25 %	6.25 %	12.5 %
Distracted Accuracy:	65 %	13/20		30.0 %		5.0 %	0.0 %
UpMobile Accuracy:	32 %	10/31		38.71 %	19.35 %		9.68 %
DownMobile Accuracy:	41 %	12/29		37.93 %	10.34 %	10.34 %	

Fig. 47. AugmentedData5 Tabular Results

The configuration for the three sets of results below was with optimizer SGD, batchsize = 16, categorical crossentropy loss function and dropout layers set at 0.5.

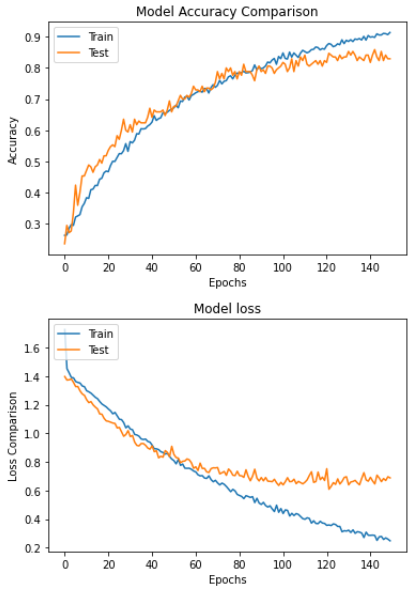

Fig. 48. AugmentedData3 Graphical Results

Category	Accuracy	Ratio	FP's:	Alert	Distracted	UpMobile	DownMobile
Overall Accuracy:	58 %	65/112					
Alert Accuracy:	68 %	22/32			12.5 %	9.38 %	9.38 %
Distracted Accuracy:	75 %	15/20		20.0 %		0.0 %	5.0 %
UpMobile Accuracy:	48 %	15/31		22.58 %	25.81 %		3.23 %
DownMobile Accuracy:	44 %	13/29		10.34 %	27.59 %	17.24 %	

Fig. 49. AugmentedData3 Tabular Results

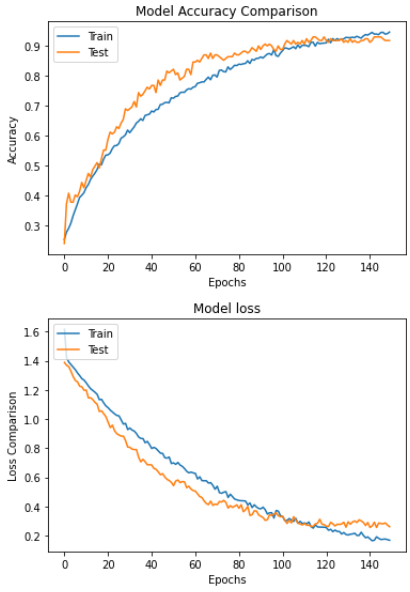

Fig. 50. AugmentedData4 Graphical Results

Category	Accuracy	Ratio	FP's:	Alert	Distracted	UpMobile	DownMobile
Overall Accuracy:	59 %	67/112					
Alert Accuracy:	78 %	25/32			15.62 %	3.12 %	3.12 %
Distracted Accuracy:	75 %	15/20		20.0 %		0.0 %	5.0 %
UpMobile Accuracy:	41 %	13/31		29.03 %	19.35 %		9.68 %
DownMobile Accuracy:	48 %	14/29		20.69 %	13.79 %	17.24 %	

Fig. 51. AugmentedData4 Tabular Results

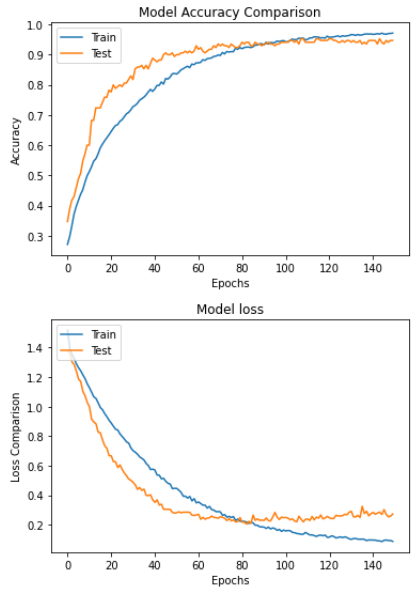

Fig. 52. AugmentedData5 Graphical Results

Category	Accuracy	Ratio	FP's:	Alert	Distracted	UpMobile	DownMobile
Overall Accuracy:	55 %	62/112					
Alert Accuracy:	84 %	27/32			9.38 %	3.12 %	3.12 %
Distracted Accuracy:	55 %	11/20		35.0 %		10.0 %	0.0 %
UpMobile Accuracy:	35 %	11/31		38.71 %	16.13 %		9.68 %
DownMobile Accuracy:	44 %	13/29		31.03 %	13.79 %	10.34 %	

Fig. 53. AugmentedData5 Tabular Results

REFERENCES

[1] F. Suard, A. Rakotomamonjy, A. Bensrhair, and A. Broggi, "Pedestrian detection using infrared imagesand histograms of oriented gradients," in2006 IEEE Intelligent Vehicles Symposium, 2006, pp. 206–212.

[2] Z. Li, K. Wang, L. Li, and F. Wang, "A review on vision-based pedestrian detection for intelligentvehicles," in2006 IEEE International Conference on Vehicular Electronics and Safety, 2006, pp. 57–62.

[3] G. Overett, L. Petersson, N. Brewer, L. Andersson, and N. Pettersson, "A new pedestrian dataset forsupervised learning," in2008 IEEE Intelligent Vehicles Symposium, 2008, pp. 373–378.

[4] P. Dollar, C. Wojek, B. Schiele, and P. Perona, "Pedestrian detection: A benchmark," in2009 IEEEConference on Computer Vision and Pattern Recognition, 2009, pp. 304–311.

[5] W. Ouyang and X. Wang, "Joint deep learning for pedestrian detection," in2013 IEEE InternationalConference on Computer Vision, 2013, pp. 2056–2063.

[6] C. Premebida, J. Carreira, J. Batista, and U. Nunes, "Pedestrian detection combining rgb and denselidar data," in2014 IEEE/RSJ International Conference on Intelligent Robots and Systems, 2014, pp.4112–4117.

[7] A. Dominguez-Sanchez, M. Cazorla, and S. Orts-Escolano, "Pedestrian movement direction recognitionusing convolutional neural networks,"IEEE Transactions on Intelligent Transportation Systems, vol. 18,no. 12, pp. 3540–3548, 2017.

[8] R. B. B. S. Jan Hosang, Mohamed Omran, "Taking a deeper look at pedestrians," 2015.

[9] D. O. Pop, "Detection of pedestrian actions based on deep learning approach,"Informatica, Babes,-BolyaiUniversity, Hal Achrives, 2019

[10] Pedestrian Accident Statistics via Condor Web