Driver Drowsiness Detection System using Machine Learning & Artificial Neural Networks

Sankalp Singh, Chilukuri K. Mohan

School of Information Studies, Syracuse University
ssingh56@syr.edu, ckmohan@syr.edu

Abstract

Drowsiness can be defined as a feeling of being lethargic and sleepy. A state of drowsiness when coupled with driving can be life threatening. Drowsiness can be caused due to several reasons like sleep deprivation, excess fatigue due to long road trips, excessive workload etc. Driving vehicles in such situations can lead to brief states of unconsciousness which can be very dangerous. Every year, a lot of people lose their lives due to road accidents. This includes both the drivers as well as the pedestrians. This also causes injuries and financial losses to a lot of individuals. In this paper, we try to leverage machine learning models and artificial neural networks to find out the critical factors that can help us in detecting drowsiness amongst drivers based on their facial features. We have developed an end-to-end drowsiness detection system which takes videos of a person as input and predicts and outputs a csv file that contains the time ranges in which that person was in a drowsy state. We believe, our proposed system will be helpful for the society and help us resolve one of the major problems of drowsy driving.

1 Introduction

Drowsy driving is a major societal problem which needs to be addressed. Feeling drowsy while driving is not just sleeping, but it could also be brief spans in which we do not have full concentration on driving and our eyes are closed or distracted. This small amount of distraction could prove fatal in many situations. To address this problem, we propose a drowsiness detection system capable of capturing the drowsy states of a driver even for small time frames and thus could help in preventing major road accidents, injuries and losses. Our system takes input in the form of videos of drivers, extracts image frames from these videos and also extracts the facial landmarks for each of these frames. One we have the facial landmarks extracted from the frames, we then leverage classification models and deep learning based artificial neural networks that predict the state of the driver as 'Alert' or 'Drowsy' for each of the frames extracted.

We have implemented several classification-based machine learning models like Logistic Regression, Naïve Bayes Classifier, Decision Trees, Random Forest, Gradient Boosting Machine, K-Nearest Neighbors, Linear Support Vector Machines. We have also implemented Simple Feedforward Neural Networks, Recurrent Neural Networks, Long Short-Term Memory, Gated Recurrent Unit and Convolutional Neural Networks to tackle the driver drowsiness problem. We have evaluated our models based on two evaluation metrics namely Recall and Accuracy.

Section 2 defines the problem statement and the 'Real-Life Drowsiness Dataset' provided by UT Arlington University that we have leveraged to train and test our machine learning models. Section 3 introduces the approach and algorithms that we have implemented for our system. Section 4 shows the results and the experiments we performed while predicting the drowsy state of drivers. Section 5 and 6 summarizes the discussions,

conclusions and future scope of the developed system.

2 Problem & Data Description

We have developed a drowsiness detection system that uses machine learning models and aims to detect drowsiness factors in drivers. For a given input video of a driver, our drowsiness detection problem aims to predict all those time frames in the video where the driver is drowsy, based on their facial landmarks extracted from the videos.

The dataset used in this work has been provided by UT Arlington University. 'Real Life Drowsiness Dataset' (RLDD) has been specifically curated by UT Arlington University to research in the domain of multistage driver drowsiness detection. This dataset consists of videos of 60 participants. Each video is 10 minute long and each participant has one video for each of the three states – 'Alert', 'Low vigilant', 'Drowsy'. The 'Alert' class consists of videos in which participants are fully conscious and are able to drive for long durations. 'Low Vigilant' class consists of videos in which participants showcase some subtle signs of sleepiness and driving in this state is not recommended. The third case - 'Drowsy' consists of videos in which participants are extremely sleepy and they have to put efforts in order to not fall asleep. Driving in this case is strictly not recommended.

In this research work, we have focused on the two states – 'Alert' and 'Drowsy'. We have treated this as a binary classification problem and have trained and tested our machine learning models on videos of 9 participants (18 videos) for each of the two cases – 'Alert' and 'Drowsy'. Using 18 videos for 9 participants, we were able to extract enough image frames and facial features for both the 'Alert' and 'Drowsy' states in order to train and test our models.

3 Approach & Algorithm

For our research work, we have followed the CRISP-DM approach which is the 'Cross Industry Standard Process for Data Mining'. It consists of following steps:

- Business & Data Understanding
- Data Preparation and Exploratory Data Analysis
- Modeling
- Evaluation
- Deployment

Below, we have showcased the end-to-end architecture of our drowsiness detection system. First step consists of uploading the input video of a driver to our system. This is followed by extraction of image frames from the videos at 1fps which is in turn followed by extracting all the facial landmarks for each of the frames. Next step is data transformation in which the features are transformed into a pandas dataframe. This step is followed by a feature engineering step to introduce lags in the features followed by normalizing and scaling of the final dataset. After this, we implement machine learning models and evaluate them based on specific evaluation metric. The final output is a csv file containing the predicted states – 0 for 'Alert' and 1 for 'Drowsy' – obtained using the best performing model.

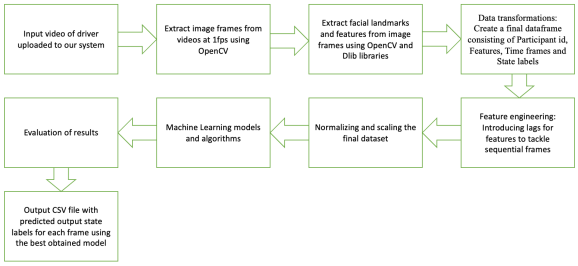

Figure1. Overall system architecture

3.1 Business & Data Understanding

For this step, we have tried to understand the overall business impact our system would have on the society. For data understanding, we initially went through the videos of each of the 9 participants for both the 'Alert' and

'Drowsy' cases to get a better understanding of what the videos actually represent.

3.2 Data Preparation and Exploratory Data Analysis

For this part, we have followed the following data pre-processing steps:

- Defining the core facial features helpful in predicting drowsiness state
- Extracting image frames and facial landmarks
- Data Transformations
- Feature Engineering
- Exploratory Data Analysis
- Normalizing and Scaling
-

3.2.1 Defining Core Facial Features

After researching on several facial features, we found the below four facial features that were the most useful in detecting drowsiness in a driver [1]:

1) Eye Aspect Ratio – This represents the ratio of length of eyes to the width of eyes. We found out that a person who is drowsy is likely to blink a lot and at the same time their eyes will become small. This will help our models to predict a person as drowsy when their eye aspect ratio becomes small over sequential frames.
2) Mouth Aspect Ratio – This represents the ratio of length of mouth to width of mouth. A person who is drowsy is more likely to yawn and this would result in a higher mouth aspect ratio helping our models in predicting the drowsy state.
3) Mouth Over Eye – This represents the ratio of Mouth Aspect Ratio to the Eye Aspect Ratio. When a person is drowsy, their mouth over eye ratio will increase as eye aspect ratio will decrease and simultaneously mouth aspect ratio will increase. This will be useful for our models while predicting the drowsy states.
4) Pupil Circularity – This represents the size of pupil inside the eye. A drowsy person will have a lower pupil circularity in comparison to an alert individual. Thus, this feature will also be useful in predicting the drowsy state.

3.2.2 Extracting Image Frames and Facial Landmarks

For extracting the image frames from videos, we have used OpenCV library provided by Python. Each video is 10 minute long and there is not much activity in the first 1 minute of each video, so we have extracted 1 frame per second from the 2nd minute till the end of video for all participants for both the cases - 'Alert' and 'Drowsy'. Each participant has two videos for both the states, so in total we have extracted 1080 frames for 1 participant for their 'Alert' and 'Drowsy' states. For all the 9 participants, we have extracted 9720 image frames for both the states. These image frames were enough to train and test our models for both the 'Alert' and 'Drowsy' states. We also created a few lists to store the participant id, time frame in seconds, frame labels (0- 'Alert' / 1- 'Drowsy') and features for each participant.

For each of the frames, we extracted 68 facial features using the OpenCV and Dlib libraries. Out of these 68 features, we only kept landmarks for eyes, mouth, lips and nose (Points 36-68) as they seemed to be the most useful data points to be fed as features to our models. We defined functions for each of the four core facial features using the landmark points extracted in the above step. Below is an image of the 68 facial landmarks provided by OpenCV library.

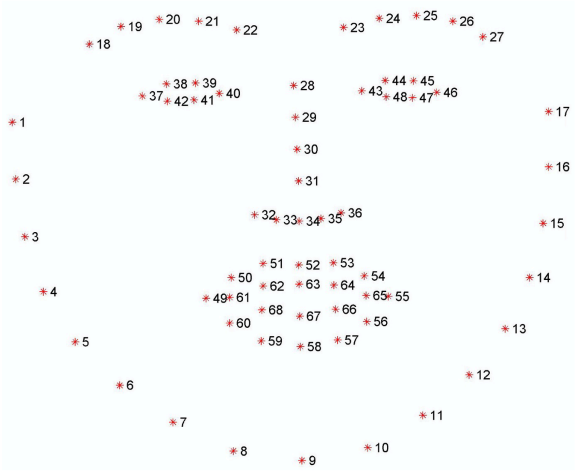

Figure2. Facial landmarks from OpenCV

3.2.3 Data Transformations

We collected the participant ids, time frames, features and frame labels for all 9 participants in the form of separate lists and transformed these lists to numpy arrays for further processing. After this, we converted these arrays into csv files for an easy access of features and labels at a later stage. Finally, we merged and converted all the csv's into a pandas dataframe consisting of all the data for 9 participants including 'Participant', 'TimeSec', 'EyeAspectRatio', 'MouthAspectRatio', 'PupilCircularity', 'MouthOverEye' and 'State' (0/1).

3.2.4 Feature Engineering

One of the major issues that we found while working on our drowsiness detection system was the problem of handling sequence of images. Using the above data, our models could predict the output state labels (Alert/Drowsy) for each frame, but we wanted to take care of sequential image frames as that is extremely important while predicting the state of an individual. For instance: suppose, there is an individual who has been drowsy since the last four frames but has become alert in the 5th frame. In this case, we cannot directly predict that individual as alert for the 5th frame. For tackling the sequence of images problem, we introduced the concept of lags in our four core features. We decided to feed the input features from previous four frames to our classification models so that they can make predictions based on the sequential data. We initially sorted the original dataframe with respect to the 'Participant', 'State' and 'TimeSec' and added new features with lags for each of the original features. So, in our final dataframe, we had 20 facial features including the 4 lags added for each of the previous 4 original facial features.

3.2.5 Exploratory Data Analysis

We have performed some visualizations on our final dataset in order to get more insights on the facial features. Below is a box plot of 'EyeAspectRatio' with the output state (0- 'Alert' / 1- 'Drowsy'). The box plot represents that an individual in Drowsy state will have a lower eye aspect ratio as compared to an alert individual as they will have smaller eyes.

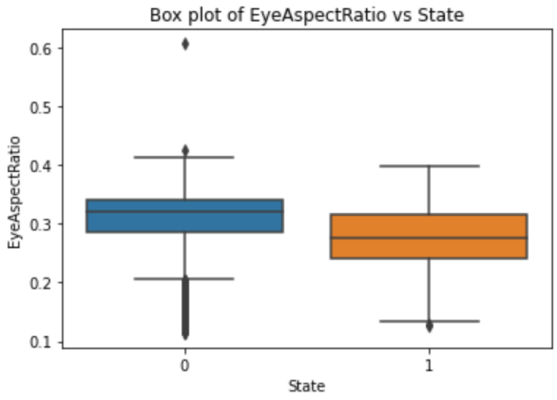

Figure3. Box Plot of EyeAspectRatio with State

Below is another plot that represents the distribution of output 'State' labels (0- 'Alert' / 1- 'Drowsy') in our final dataset. We can see from the distribution that our final dataset is perfectly balanced with respect to the output state labels.

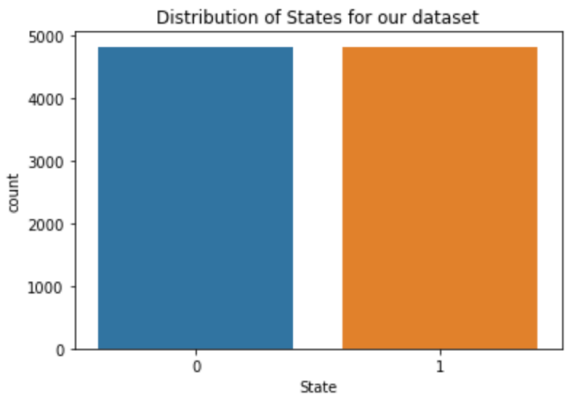

Figure4. Distribution of State labels for our final dataset

3.2.6 Normalizing and Scaling

We have scaled and normalized our final dataset using a standard scalar. Scaling and normalizing the data is really necessary for our problem as all the human beings have varying facial landmarks and features. Normalizing the data will help reduce bias in our models and help them to perform better on a new individual with unseen facial features. Normalizing will also allow all the features to contribute equally to the output 'State' label.

3.3 Modeling

For the modeling part, we have split the final dataframe containing features for all 9 participants into two parts – train_df and score_df. We have used the data from 7 participants to train our models. We have tested our models on the 8th participant, and we have scored the best obtained model using data from 9th participant.

For model building, we have followed the below two approches:

1) Approach1 – Participant wise split. Here, we have performed a participant wise train-test shuffle. We have trained our models on 7 participants and then tested our models on the 8th participant. This approach would help us evaluate our models on unseen facial features as the facial features of the test participant would not be present in the training dataset. This will help us estimate the out-of-sample accuracy of our models.
2) Approach2 – Random split. Here, we have performed a random train-test shuffle with 70% train data and 30% test data. This approach will help us evaluate our models on facial features of random participants.

Below, we have mentioned all the machine learning models we have implemented for both the above mentioned approches along with their evaluation metrics:

Type	Algorithm	Evaluation Metric
Classification	Logistic Regression	Recall
Classification	Naïve Bayes	Recall
Classification	Decision Trees	Recall
Classification	Random Forest	Recall
Classification	Gradient Boosting Machine	Recall
Classification	K-Nearest Neighbors	Recall
Classification	Linear SVM	Recall
Artificial Neural Network	Simple Feed Forward Neural Network	Accuracy
Recurrent Neural Network	Simple RNN	Accuracy
Recurrent Neural Network	LSTM	Accuracy

Recurrent Neural Network	GRU	Accuracy
Deep Learning Network	CNN	Accuracy

Table1. Machine learning algorithms used for the research

4 Results

We have evaluated our models for both the approaches using two evaluation metrics. We have used 'Recall' to evaluate all of our classification-based machine learning models. We have used recall as one of our evaluation metrics due to the fact that we want to reduce the total number of 'false-negatives' predicted by our models. We do not want our models to predict someone as 'Alert' when they are actually 'Drowsy' as that would defeat the overall purpose of our drowsiness detection system. As our final dataset is perfectly balanced with respect to the binary output 'State' labels, we have also used 'Accuracy' as the evaluation metric for our Artificial Neural Networks, Recurrent Neural Networks and Deep Learning Network.

Below is a table containing the performance scores of all models implemented using Approach1 (participant wise split):

Model	Recall	Accuracy
Simple Feed Forward Neural Network	**56**	44.30
Logistic Regression	50	49
Simple RNN	50	50
LSTM	50	**50**
GRU	50	50
CNN	50	50
Linear SVM	49	48
Random Forest	47	46
KNN	46	43
Naïve Bayes	45	44
Decision Trees	45	43
Gradient Boosting Machine	44	41

Table2. Results obtained using Participant wise train-test split

Below is a table containing the performance scores of all models implemented using Approach2 (random split):

Model	Recall	Accuracy
Gradient Boosting Machine	**92**	**91**
Random Forest	89	87
KNN	86	86
Simple Feed Forward Network	84	87
Linear SVM	74	75
Decision Trees	73	77
Logistic Regression	71	72
Naïve Bayes	70	70
Simple RNN	50	50.17
LSTM	50	50
GRU	50	50.07
CNN	50	50

Table3. Results obtained using random train-test split

We have also calculated the feature importance using Gradient Boosting Machine which is the best model obtained out of both the approaches. According to the feature importance plot, the top 5 attributes that are most important in predicting the drowsy state of an individual are – 'MouthOverEye', 'MouthOverEye_lag4', 'MouthOverEye_lag3', 'MouthAspectRatio_lag2', 'MouthOverEye_lag2'.

Below is the feature importance plot obtained by Gradient Boosting Machine:

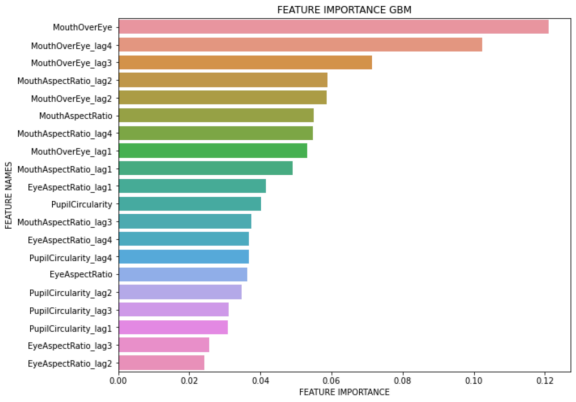

Figure5. Feature importance plot GBM

Area under the receiver operating characteristic curve for Gradient Boosting Machine shows that the GBM model is able to clearly distinguish between the alert and drowsy states 97.2% percent of the times. Below is the AUC-ROC curve for Gradient Boosting Machine:

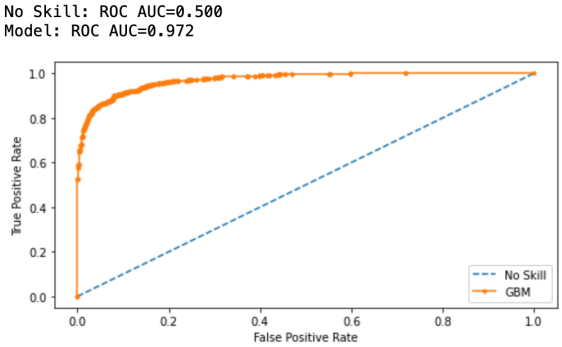

Figure6. AUC plot for GBM

5 Discussions

According to the results obtained above for both the approaches, we can conclude that the models implemented using the random train-test shuffle performed better in comparison to the models implemented using the participant wise train-test shuffle. This could be due to the fact that the participant wise shuffle allows the model to get trained on a set of participants and then the predictions are made on a participant whose features are not present in the training sample. The models trained on participant wise split are not able to make good predictions on out of sample data points or the unseen facial features. In contrast, the models trained using a random train-test shuffle are able to perform better as they are being trained and then tested on random participant's features leading to a higher evaluation score.

6 Conclusion & Future Scope

We have presented a drowsiness detection system that leverages machine learning algorithms to predict the time frames when an individual is in drowsy state while driving by using facial features 'Eye Aspect Ratio', 'Mouth Aspect Ratio', 'Mouth Over Eye' and 'Pupil Circularity'. We have implemented an end-to-end system which takes videos of drivers as inputs, extracts image frames from the videos, then extracts facial landmarks from the image frames and predicts the 'Alert' and 'Drowsy' state for each time frame. The system also generates a csv file as an output containing the participant id, time frame in seconds, all features of the participant and the predicted binary output 'State' – 'Alert' (0) / 'Drowsy' (1). We believe our system will be really beneficial for the society as this system will allow to reduce road accidents and injuries by predicting early drowsiness factors in drivers and alerting them before it's too late.

In the future, we plan to extend this machine learning system to a mobile application which can be used by individuals just as any other applications like Maps. This application will alert the individual via an alarm if it detects any early signs of drowsiness by extracting various facial landmarks and thus could prevent disastrous events.

7 Appendix

We have performed several experiments for the following models for both the approaches defined in Section 3.3 – Simple Feed Forward Neural Network, Simple RNN,

LSTM, GRU and CNN. For each of these models, we experimented with three different architectures. Below, we have showcased the three architectures for one of the models, namely LSTM:

1) Architecture1 - Single LSTM layer
2) Architecture2 - Single LSTM layer with 1 hidden layer
3) Architecture3 - Single LSTM layer with 2 hidden layers

We also experimented with other hyperparameters like optimizer, activation function and number of nodes in each hidden layer. For each of the three architectures, we conducted four trials with a fixed epoch=5 and a batch_size=20. For all the models, we calculated the mean validation loss, standard deviation of validation loss, mean time complexity and mean validation accuracy achieved after conducting the multiple trials. This approach helped us in selecting the network with the best size and best hyperparameters for all the neural network models based on the mean validation loss. We then used the best model from the trials and trained that model using epoch=20.

7.1 Experiment I

We have performed these experiments on all five neural network models using the modeling Approach1 - participant wise train-test split. Below is a summary of all the architectures tried for each of the models:

Model	**Mean Validation Loss after 4 trials**	**Validation Loss Standard deviation after 4 trials**	**Mean Computational efforts after 4 trials (sec)**	**Mean Validation accuracy after 4 trials**
Simple Feedforward model Architecture 1	**0.884108**	0.015248	45.405134	0.429804
Simple Feedforward model Architecture 2	1.414644	0.136610	58.286516	0.427006
Simple Feedforward model Architecture 3	1.259564	0.102251	78.785487	0.434468
RNN Architecture 1	0.693160	0.000006	312.215857	0.500233
RNN Architecture 2	24.505159	42.993534	468.784640	0.504664
RNN Architecture 3	**0.691977**	0.002347	457.718528	0.501866
LSTM Architecture1	**0.693134**	0.000039	408.593303	0.500933
LSTM Architecture2	0.693214	0.000049	430.072333	0.500000
LSTM Architecture3	0.693171	0.000008	430.032561	0.500466
GRU Architecture1	**0.693118**	0.000109	779.553214	0.500700
GRU Architecture2	0.693164	0.000100	765.057774	0.500466
GRU Architecture3	0.693154	0.000017	698.735447	0.500933
CNN Architecture1	**7.690729**	0.044072	127.678037	0.499987
CNN Architecture2	7.712496	0.000009	177.678906	0.500000

Table4. Experimental results for participant wise split approach

Below, we have summarized the best network size obtained for each of the models after performing multiple trials along with their hyperparameters:

Best Network Size	Validation Accuracy	Computational effort(sec)	Optimizer	Activation	Epochs & Batch Size
Simple Feedforward model Architecture 1	44.30	195.70	SGD	relu	20, 20
RNN Architecture 3	50	1598.41	Adam	relu	20, 20
LSTM Architecture 1	50	1340.77	SGD	tanh	20, 20
GRU Architecture 1	50	3420.66	SGD	tanh	20, 20
CNN Architecture 1	**50**	471.21	SGD	relu	20, 20

Table5. Best performing neural network models after Experiment I

7.2 Experiment II

We have performed these experiments using the modeling Approach2 by using a random train-test split of 70-30. Below is summary of all the architectures tried for each of the models:

Model	Mean Validation Loss after 4 trials	Validation Loss Standard deviation after 4 trials	Mean Computational efforts after 4 trials (sec)	Mean Validation accuracy after 4 trials
Simple Feedforward model Architecture 1	0.517974	0.008602	90.550432	0.744462
Simple Feedforward model Architecture 2	0.419295	0.005150	94.420630	0.797610
Simple Feedforward model Architecture 3	**0.346572**	0.009107	143.697042	0.834240
RNN Architecture 1	**0.693166**	0.000003	248.740467	0.498251
RNN Architecture 2	1.257553	1.128447	400.251374	0.498154
RNN Architecture 3	0.693830	0.000549	384.299400	0.497668
LSTM Architecture1	0.693162	0.000000	404.070966	0.498251
LSTM Architecture2	**0.693150**	0.000007	308.781515	0.500000
LSTM Architecture3	0.693158	0.000002	321.095826	0.497960
GRU Architecture1	**0.693148**	0.000026	694.872443	0.498154
GRU Architecture2	0.693176	0.000042	755.620360	0.499903

GRU Architecture3	0.693228	0.000082	698.515272	0.500777
CNN Architecture1	7.629479	0.073321	79.616614	0.500874
CNN Architecture2	7.595090	0.001308	119.225425	0.501749
CNN Architecture3	**7.594461**	0.000916	142.478073	0.501749

Table6. Experimental results for random split approach

Below, we have summarized the best network size obtained for each of the models after performing multiple trials along with their hyperparameters:

Best Network Size	**Validation Accuracy**	**Computational effort(sec)**	**Optimizer**	**Activation**	**Epochs & Batch Size**
Simple Feedforward model Architecture 3	**86.94**	408.45	Adam	Relu	20, 20
RNN Architecture 1	50.17	1047.94	SGD	Tanh	20, 20
LSTM Architecture 2	50	1340.77	SGD	Tanh	20, 20
GRU Architecture 1	50.17	2765.80	SGD	Tanh	20, 20
CNN Architecture 3	50.07	569.88	SGD	Relu	20, 20

Table7. Best performing neural network models after Experiment II

For the experiments performed for neural network models, we can infer that for approach 1, CNN model with just 1 CNN layer performed the best with the highest accuracy and lowest computational efforts in comparison to all other networks.

For the experiments performed using approach 2, Simple Feedforward network with 3 hidden layers performed the best with respect to the accuracy and computational efforts in comparison to f all other network.

8 References

[1]https://towardsdatascience.com/drowsiness-detection-with-machine-learning-765a16ca208a

[2]https://machinelearningmastery.com/how-to-perform-face-detection-with-classical-and-deep-learning-methods-in-python-with-keras/

[3]https://www.pyimagesearch.com/2018/06/18/face-recognition-with-opencv-python-and-deep-learning/

[4]https://www.pyimagesearch.com/2017/04/03/facial-landmarks-dlib-opencv-python/

[5]https://www.geeksforgeeks.org/opencv-facial-landmarks-and-face-detection-using-dlib-and-opencv/

[6]https://livecodestream.dev/post/detecting-face-features-with-python/

[7]https://medium.com/analytics-vidhya/facial-landmarks-and-face-detection-in-python-with-opencv-73979391f30e

[8] https://towardsdatascience.com/facial-mapping-landmarks-with-dlib-python-160abcf7d672

[9] https://towardsdatascience.com/robust-facial-landmarks-for-occluded-angled-faces-925e465cbf2e

[10] https://pypi.org/project/opencv-python/

[11] https://docs.opencv.org/master/

[12] https://docs.opencv.org/3.4/d8/dfe/classcv_1_1VideoCapture.html

[13] http://dlib.net

[14] Dataset - https://sites.google.com/view/utarldd/home http://vlm1.uta.edu/~athitsos/projects/drowsiness/

[15] Facial Features Monitoring for Real Time Drowsiness Detection
Manu B.N

[16] A Fatigue Driving Detection Algorithm Based on Facial Motion Information Entropy
Feng You,1,2 Yunbo Gong,1 Haiqing Tu,1 Jianzhong Liang,1 and Haiwei Wang

[17] A System on Intelligent Driver Drowsiness Detection Method
Ugra Mohan Kumar1, Devendra Singh1, Sudhir Jugran1, Pankaj Punia1, Vinay Negi2

[18] An Investigation of Early Detection of Driver Drowsiness Using Ensemble Machine Learning Based on Hybrid Sensing †
Jongseong Gwak 1,*, Akinari Hirao 2 and Motoki Shino

[19] Driver Drowsiness Detection Model Using Convolutional Neural Networks Techniques for Android Application
⇤† ⇤ ⇤
Rateb Jabbar , Mohammed Shinoy , Mohamed Kharbeche , Khalifa Al-Khalifa§, Moez Krichen‡, Kamel Barkaoui†

[20] Efficient Driver Fatigue Detection and Alerting System Miss. Kanchan Manohar Sontakke

[21] Facial Features Tracking applied to Drivers Drowsiness Detection
L. M. BERGASA, R. BAREA, E. LÓPEZ, M. ESCUDERO, J. I. PINEDO

[22] Feature selection for driving fatigue characterization and detection using visual- and signal-based sensors
1* 2 1 1 2 Khadidja Henni , Neila Mezghani , Charles Gouin-Vallerand , Perrine Ruer , Youssef Ouakrim

[23] Real-Time Driver Drowsiness Detection System Using Eye Aspect Ratio and Eye Closure Ratio
Sukrit Mehta, Sharad Dadhich, Sahil Gumber, Arpita Jadhav Bhatt