Drowsiness Detection using CNN-Based Feature Extraction with Recurrent Neural Networks

Ali Ekin Gurgen, Begum Ortaoglu, and Sonia Gu

1. Motivation

Long haul truck driving forms the backbone of domestic commerce, but it exposes its drivers to many of the dangers of road-based transportation. In particular, the long hours that these drivers must be on the road for causes many of them to become tired at the wheel. In a report by the Federal Motor Carrier Safety administration (FMCSA) [1], an estimated 13% of truck driver accidents were caused by driver fatigue, often perpetuated by the long hours that they must work for and inadequate sleep on the road. In 2013, it was estimated that 72,000 crashes in the U.S. were due to driver drowsiness, 800 of which ended fatally [2]. These crashes were either directly or indirectly caused by fatigue – when drivers are drowsy, they are more likely to let their guard down and may forget to check their blind spots or accidentally veer towards oncoming traffic. This is extremely dangerous because it not only threatens the driver's life, but it also threatens the drivers around the truck, since these trucks can weigh anywhere from 10-80 thousand pounds, and the debris can cause accidents among neighboring cars as well.

In 2015, the FMCSA published the Electronic Logging device (ELD) rule that mandated that drivers must log their hours electronically instead of on physical paper logs so that the government and trucking companies could monitor whether drivers were operating within the regulations that dictated how many hours a driver can drive in a day and the frequency to which they should take breaks [3]. However, many truckers expressed privacy concerns using these ELDs and there were protests in 2017 to abolish the ELD rule [4]. Many companies are now starting to create technologies that will allow trucking companies to determine driver drowsiness [5]. SmartCap Technologies has created a helmet that drivers can wear that will be able to use brainwave measurements to determine a driver's alertness. The measurements are connected to an app that will tell drivers right before they enter a microsleep [6]. Although this device has shown to be useful in preventing microsleeps, drivers may be hesitant to wearing helmets while driving and adoption may be poor.

Other companies have started to develop more non-intrusive methods of determining driver drowsiness. Seeing Machines has developed a camera that drivers mount onto

their dashboards. These cameras can use face and gaze detection to monitor driver safety and send alarms to the driver if they are below some threshold [7].

2. Goal

It is important for drivers to be safe and adhering to all precautions when they drive, and an important indicator of safety is driver drowsiness. Our research project aims to follow the steps of Seeing Machines and build the *minimal* model, using convolutional and recurrent neural networks, to gauge how drowsy a person is. The model may then be used in systems that can safely alert drivers to take a break and get off the road.

3. Previous Work

Previous work on this topic has used artificial neural networks to predict a driver's drowsiness based on various behavioral and psychological indicators such as heart rate, respiration rate, head movement, blink duration, frequency and PERCLOS (percentage of eyelid closure). Karolinska Sleepiness Scale (KSS), a 9-graded Lickert scale has been the common metric used to define the subjective self-assessment of drowsiness [8]. Even though the physiological features are a more accurate representation of the driver's sleepiness, a data scale that is labeled in the KSS scale is more accessible to the researchers. A lot of different algorithms were used to achieve this task: including k-nearest neighbours, decision trees, artificial neural networks, ensemble like random forests and deep learning [8].

A lot of the models in the literature use a predetermined set of features. This predetermined set of features mostly focuses on the features extracted from the eye and head movements. Some studies explore drowsiness according to the changes in the head position tilt since that's likely to be an indicator of drowsiness. More commonly, studies focus on facial features such as "yawning, eye closure and erratic head movements" [9]. In a study conducted by M. Omidyeganeh, A. Javadtalab, and S. Shirmohammadi, face region and the eye region were extracted from the face as well as the mouth [9]. A state, such as yawning or closed eyes, is assigned to these features and later combined by a decision making logic to output a drowsiness decision. In another study, a predetermined set of features from a detection model was fed into Artificial Neural Networks, followed by a second neural network if the drowsiness level was uncertain, using mean-squared error as the loss function of the model [8].

Another study used various CNN architectures as classifiers [10]. Alex-Net and VGG-net were used as feature extractors. This study also tried combining optical flow to detect the direction of the motion and feed that into a CNN. The disadvantage of feeding whole frames of data was that the network detected a lot of motion that is

unrelated to the levels of drowsiness. Especially if the drivers moved around a lot, the network couldn't make any intelligent guesses on drowsiness and learned unimportant actions as indicators of drowsiness. 3D CNNs were used to process temporal data to process multiple data frames. Even though 3D CNNs were able to produce much better results, a handicap of this approach is that it is too computationally expensive to be used in real life scenarios [11] . In other studies Long Short-Term Memory networks were used in combination with CNNs as an alternative to this [12].

One of the papers that we drew inspiration from leverages a combination of CNNs as well as multiple LSTMs [13]. In this paper, the eyes and the mouth were extracted from faces first and fed into CNNs to extract temporal features to output a label on each of these patches of the faces. Later these labels were combined in the temporal portion of the model where a feature extraction concatenation was performed combined with three Time Skip Combination Long Short Term Memory (TCS-LSTM) layers. These were summed element-wise and finally refined in a single LSTM to make a final classification [13].

The Long-term Multi-granularity Deep Framework is another sophisticated approach to this problem [14]. Here, 8 patches from the face, sections mostly including parts of the eyes, mouth, and nose, were extracted from the face and fed into networks that were fine-tuned for each of these segments. These 8 patches together with the image of the face were fed into a similar structure that uses LSTM networks to achieve a good accuracy in the drowsiness detection task [14].

4. Design and Implementation

Many of the available open-source implementations of drowsiness detection systems utilize a manual feature extraction step prior to the neural network training. They typically extract facial features of particular interest, such as eyes and mouth, and some of them even conduct pre-processing on sequences of images to extract features such as blink duration and yawning. In pursuit of providing a *minimal* solution to the drowsiness detection problem, we endeavored to accomplish an implicit feature extraction through several convolutional layers. The flattened output of the CNN layers were then fed into a Recurrent Neural Network (RNN) pipeline, which made a prediction of drowsiness at the end. In this section, we will discuss the details of the design and implementation in more detail.

4.1 Computing Platform

A Google Cloud Platform VM instance with 4 CPUs and 16 GB memory running on Ubuntu 20.04 was utilized as the computing platform. Even though our implementation did not require an extensive computation power such as several GPUs, we still needed

a virtual machine to download and process tens of gigabytes of videos. It was also very helpful for all members of the team to be able to easily access the data and the code at anytime.

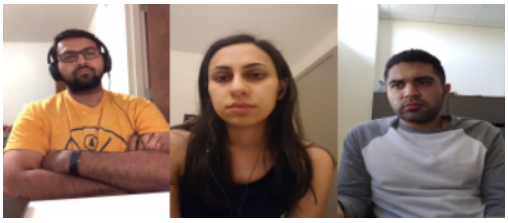
4.2 Dataset and Preprocessing

Finding an image dataset for drowsiness detection was not an easy task since almost all of the available data was in video format. In this project, we used the University of Texas at Arlington Real-Life Drowsiness Dataset (UTA-RLDD), [15] which was specifically targeted for multi-stage drowsiness detection. The UTA-RLDD dataset consists of 30 hours of RGB videos of 60 participants. Each participant had one video for each of the three classes: alert (labeled as 0), low vigilant (labeled as 1), and drowsy (labeled as 2). Figure 1a shows sample frames taken from the UTA-RLDD dataset [15].

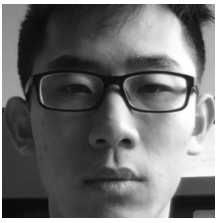
After downloading all 180 videos, we extracted images from the videos. The rate of selecting images from the videos was 1 frame per second and the images were saved using an indexing convention that preserved their temporal continuity. Furthermore, we performed face and eye detection steps during this image extraction where we used OpenCV's Cascade Classifiers along with Haar features. We put the detected faces in nominal sizes and positions as well as converting them into grayscale images. The eyes were extracted by cropping a pre-defined 256x100 pixel rectangle from the face images. During this step, we noticed that our feature detectors were not able to detect features in some of the videos due to lack of video quality and obscure lighting. We simply ignored the videos in which it was difficult to detect features in and ended up with 120 videos which resulted in approximately 72000 grayscale images. Figure 1b shows a sample processed face image.

4.3 Data Loading

For each batch, we randomly sampled n (window size = 50) consecutive frames from k (batch size = 8) participant videos. It was important that the n frames be consecutive because we wanted to preserve the time domain as we loaded them into the RNN.



(a) *Sample frames of three participants*



(b) *A sample frame after preprocessing*

FIGURE 1. *Sample frames from UTA-RLDD*

We used a Keras custom data generator to load our images into the neural network. We had to ignore the data from around one third of the participants because their video quality was too poor to detect any faces, so within our data generator, we made sure to ignore the frames from these participants as well.

4.4 CNN + LSTM Structure

In total, we had three training iterations until achieving successful results. For our first and second iterations, we used the images of the faces and eyes, respectively, we had extracted using Haar features. The following neural network model framework, shown in Figures 2 and 3, were used on both sets of data.

Our neural network used a CNN backbone to extract features from each frame, and a long short-term memory (LSTM) layer to learn from a series of frames. Our CNN layers for feature extraction, illustrated in Figure 2, were comprised of two time distributed convolution layers that used a dropout of 0.2 followed by two two-dimensional max pooling layers. As the number of convolutional layers increased, the channel size increased, but the kernel size decreased with a stride length of 2 and max-pooling after each convolutional layer. These layers are used to extract features from the images.

While the convolutional layers were used to achieve image feature extraction, we leveraged time distributed layers so that we could seamlessly transition from the convolution layers to the LSTM layer. The time distributed layers allowed us to run the CNN layers for feature extraction (See Figure 2) for each frame separately, while still being able to pass a series of frames into the LSTM. The LSTM was necessary since lots of information about drowsiness can be captured throughout the time axis. For example, the network may be able to now draw conclusions about yawning and it may help us handle a participant blinking. In this case, a single frame will not give us the best prediction of whether a person is drowsy – if they are blinking, the model would predict that they were drowsy when in reality, they could be blinking while they were alert or drowsy. Therefore, the time axis was important so that these special frames would be taken into the larger context of the scene. Time distributed layers also make sure that the entire list of convolution flows can find the same features given a sequence of images [16].

Dropout regularization is used to make sure that all features are taken into account instead of focusing on a single feature and to prevent over fitting. The CNN layers for feature extraction are then flattened to be fed into the Recurrent Neural Network that will make a prediction on the drowsiness level. Recurrent Neural Networks let information persist and allow longer term learning. LSTMs are a special kind of RNNs, capable of learning long term dependencies [17].

The output of the LSTM layer is fed into two dense layers, one with a ReLU activation and the other a softmax. A cross-entropy loss is calculated over the overall output and

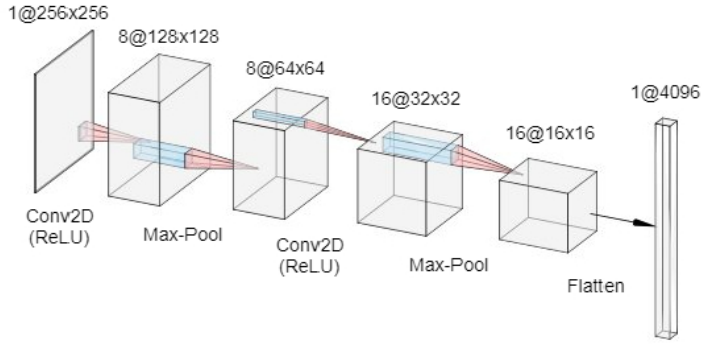


FIGURE 2. *CNN Layers for Feature Extraction*

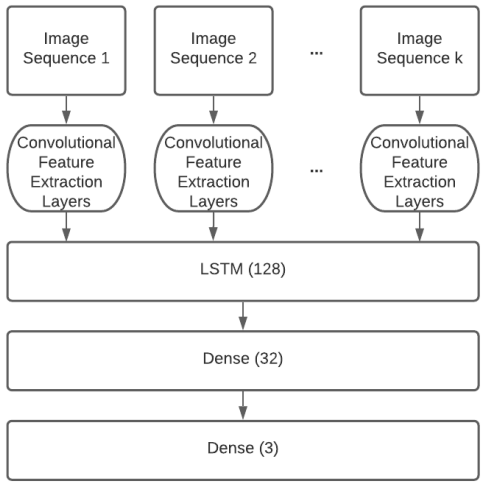


FIGURE 3. *Full Neural Network Diagram*

the network makes a single prediction for the entire sequence of images rather than giving a drowsiness prediction for each image. The full diagram of the neural network is illustrated in Figure 3.

4.5 Extracting Eyes

After training both models above using the faces, which were extracted with Haar features, and eyes, cropped from the faces in a rudimentary fashion, we decided to use an algorithm to detect the exact coordinates of a person's eyes.

For our third iteration, we used face landmark detection to crop each participant's right and left eye from the face images. Using the Dlib libraries, we ran all of the images of faces using the facial landmark detector which identifies the locations of various facial landmarks (e.g. mouth, eyebrows, eyes) using 68 coordinates [18]. Figure 4 shows all the facial landmarks that the detectors can identify.

By using coordinates 37 to 42 on the left eye and coordinates 43 to 47 on the right eye, we were able to crop both eyes from the face and resize the images to a 64 by 32 pixels. The resulting images match those shown in Figure 5, and zooming specifically into the eyes made it much more clear which were labelled alert and which were labelled drowsy.

4.6 Final Neural Network Model

The model of the neural network that we used for the third iteration is very similar to the model in the first two. We added batch normalization layers after the convolutional layers to increase the speed of convergence and decreased our learning rate to 0.0001. The sizes of the inputs also decreased because we were only training with the eye images. The resulting CNN model for feature extraction is illustrated in Figure 6.

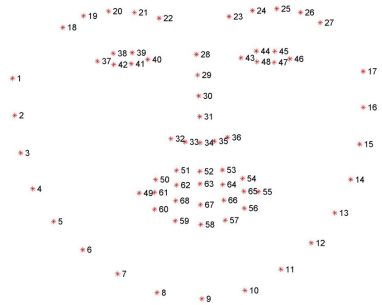
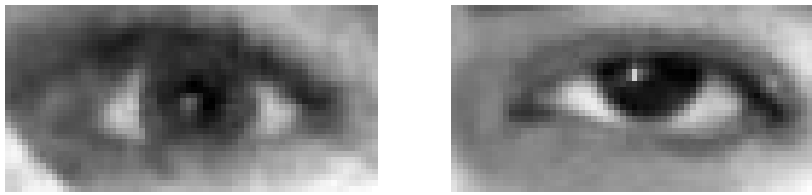


FIGURE 4. *Face landmarks using Dlib*



(a) *Alert labeled eye from participant 10* **(b)** *Drowsy labeled eye from participant 10*

FIGURE 5. *Sample images of extracted eyes*

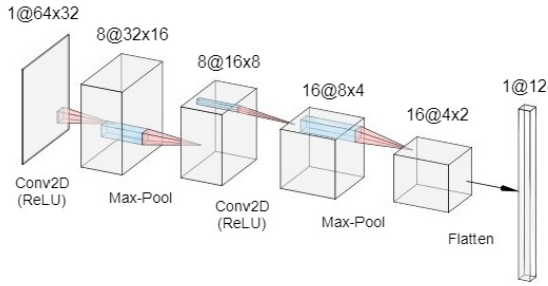


FIGURE 6. *Final CNN Model for Feature Extraction*

Moreover, realizing that the multi-stage drowsiness detection is a very challenging task, we decided to remove the middle label, which indicated a low vigilant state, making our task only a binary classification problem of drowsy or alert. As a result, the loss function that we used changed to binary cross entropy and the full neural network that reflects these changes is shown in Figure 7.

5. Results

This section will present both unsuccessful and successful results of the project. Section 5.1 discusses the failed attempt of training the neural network pipeline using the images of faces. Section 5.2 will discuss our attempt to train the same NN using images of eyes that were manually cropped out as explained in Section 4.5. After discussing the shortcomings of these attempts, we will finally present our successful training which reached a final test accuracy of 78% on the image dataset that we constructed from the UTA-RLDD.

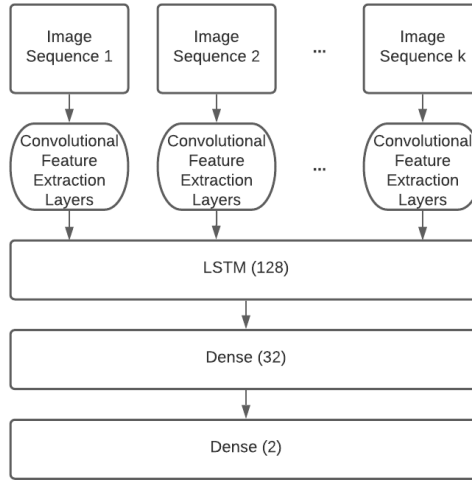
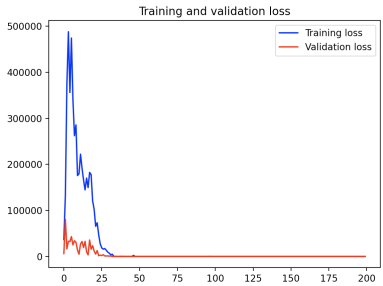


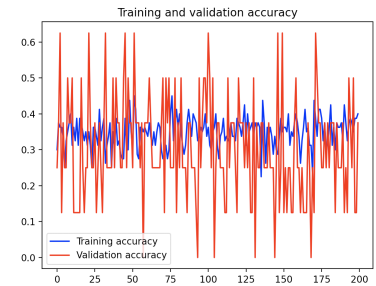
FIGURE 7. *Full Neural Network Diagram with Binary Classes*

5.1 Training with Faces

Figure 8 shows the results of training our neural network based drowsiness detection pipeline using full face images sampled from the videos. We trained for 200 epochs for approximately 9 hours. As you can see in Figure 8b, training and validation accuracy curves simply fluctuate around 33%, which is same as random guessing. The loss curves, on the other hand, demonstrate sharp decreases. The final test accuracy was 30%. Based on these results, we can clearly see that the network was not learning the intended task.

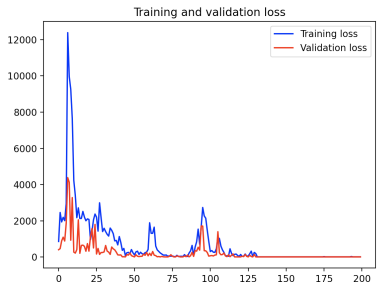


(a) *Training and Validation Loss Curves*

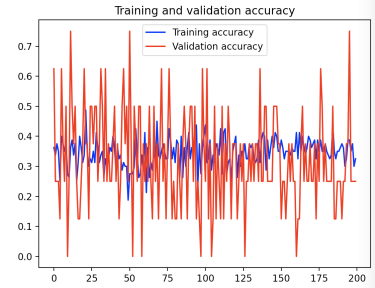


(b) *Training and Validation Accuracy Curves*

FIGURE 8. *Training Results (Training with Faces)*



(a) *Training and Validation Loss Curves*



(b) *Training and Validation Accuracy Curves*

FIGURE 9. *Training Results (Training with Manually Cropped Out Eyes)*

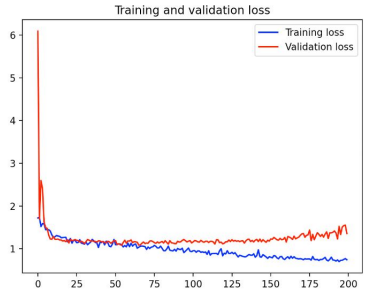
5.2 Training with Eyes Manually Cropped Out

Figure 9 shows the results of training our neural network based drowsiness detection pipeline using images of eyes that were manually cropped out from the face images using a pre-determined rectangular box. We trained for 200 epochs for approximately 8 hours. As you can see in Figure 9b, training and validation accuracy curves simply fluctuate around 33%, which is same as random guessing. Similarly to the results in Section 5.1, the loss curves demonstrate sharp decreases that do not match the behavior of the accuracy. The final accuracy was 34%. Based on these results, we can clearly see that the network was not learning the intended task and we speculate that there might have been some implementation errors in these attempts.

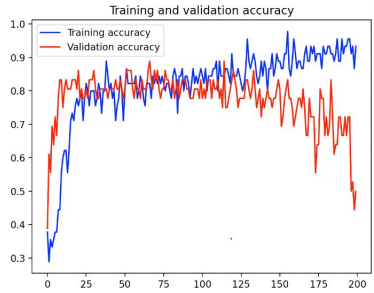
5.3 Training with Eyes Extracted with facial landmarks

After our unsuccessful attempts, we implemented the eye extraction procedure explained in Section 4.5 and made a few more improvements that are explained in more detail in the Discussion section. In order to eliminate any implementation errors that might have occurred in our earlier attempts, we also wrote our data loading and training codes from scratch. Below, in Figure 10, you can see the loss and accuracy curves where we have seen an over-fitting behavior. We reached a final test accuracy of 62% in this attempt. Dropout rate of 0.2 and no other regularizers were used in this attempt.

Finally, after increasing the dropout rate to 0.3, adding layer weight regularizers and early stopping at 100 epochs, we have acquired the the loss and accuracy plots given in Figure 11. This model has reached a final test accuracy of 78%, which we believe is a great performance on this dataset.

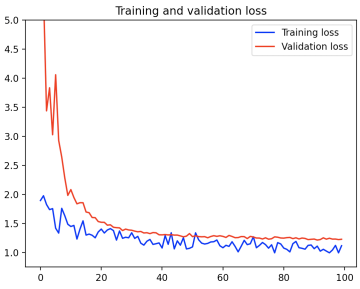


(a) *Training and Validation Loss Curves*

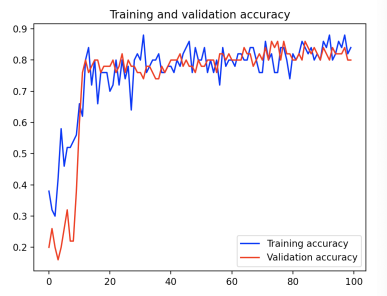


(b) *Training and Validation Accuracy Curves*

FIGURE 10. *Training Results with Over-fitting (Training with facial landmark-extracted eyes)*



(a) *Training and Validation Loss Curves*



(b) *Training and Validation Accuracy Curves*

FIGURE 11. *Training Results (Training with with facial landmark-extracted eyes)*

6. Discussion

Our models in training with faces and training with manually cropped eyes both did not learn how to predict drowsiness. While the loss decreased, as shown in Figure 8a and Figure 9a, the accuracy oscillated around .33, illustrated in Figure 8b and Figure 9b. We believe there are many possible reasons for these faulty results. Seeing the discrepancy between the loss and accuracy curves, we can speculate that there might have been an implementation error in these two attempts. In fact, we wrote the data processing, loading and training code from scratch while implementing our final attempt.

Both sets of data in the failed attempts also had shortcomings. It was too difficult to extract any meaningful data from the entire face – any indication of changes in the sizes of the eye due to drowsiness was too slight to notice to the naked eye, and we expected it to be difficult for the network to recognize too because there were so many other features to focus on as well (for example, the mouth and nose). In addition, there were some images where the Haar face detection produced faulty images as well. Some frames extracted noses or mouths instead of faces, and while we were able to remove many of the flawed faces, since we initially extracted 72 thousand frames, it was difficult to remove all of them. In addition, there were some faces where the participants wore glasses or had their hands in front of their faces. In this case, detecting any eye features would also be difficult because of the obstructing objects.

On the other hand, the original images of the eyes were extracted by simply cropping a fixed height interval of the face images. This simple technique often led to faulty images such as the one in Figure 12. Many of the same reasons why the face images were not robust also applied to these cropped eye images as well.

In these two versions, there were 3 labels: alert (label 0), low vigilant (label 1) and drowsy (label 2). To the naked eye, it was difficult to distinguish who was drowsy, who was low vigilant, and who was alert, even when watching the original videos from which we extracted the frames from. We realized that people's facial features did not change very distinctly from one label to another. In fact, the average accuracy of the human observers was under 60% on the UTA-RLDD dataset [15]. Therefore, for the third iteration, we decided to remove label 1 and just distinguish between alert

FIGURE 12. *Faulty eye cropping*

and drowsy. After removing the low vigilant labelled data from our data-set, it was easier to infer drowsiness from the images using our eyes.

Our first few attempts of training the final model with facial landmark-extracted eyes had some over-fitting problems as it can be seen in figure 10. We believe that these were simply due to model complexity and we have resolved them by adding layer weight regularizers to our CNN layers, increasing the dropout rate, and early stopping at epoch 100.

We saw that we had the best results with the eyes extracted using face landmark detection. Our final model achieves 78% accuracy, whereas the author of the original paper from which our dataset is retrieved from achieved a 65.2% accuracy. We believe that this 13% increase in accuracy is mainly due to the reduced label set – the original paper distinguished between the three labels alert, low vigilant, and drowsy [15]. Another benchmark that we can compare against is Alexnet, which achieved a 65.9% accuracy on a similar drowsiness dataset [10, 13].

This also suggests that most information regarding the drowsiness level is in the eyes. In our previous attempts, other facial features and movements might have been adding information into the data that was not necessarily an indicator of drowsiness and could even have detrimental effects on the network by making it learn how to recognize the subject rather than their drowsiness level.

6. Future Work

Given the constrained time of this project, we had to reduce our 3-stage drowsiness detection problem to a binary classification problem. In future work, it would be interesting to tackle the multi-stage drowsiness detection problem with our approach. Although it is a much more challenging task, distinguishing between different stages of drowsiness would allow such a detection system to warn the subject at an early time. This can have very beneficial consequences especially in safety-critical systems such as the previously mentioned long haul truck driving.

Another interesting direction one could take in this topic is to incorporate the data such as yawning, blink duration and blink frequency. Although the previous work on the drowsiness detection problem use these data by utilizing manual feature extraction methods, it would be interesting to see how our approach with CNN+LSTM can utilize this data to improve its performance.

References

[1] The Large Truck Crash Causation Study - Analysis Brief. July 2007. Available at <https://www.fmcsa.dot.gov/safety/research-and-analysis/large-truck-crash-causation-study-analysis-brief>.

[2] Marcus Viles. The Deadly Dangers of Truck Driver Fatigue. Sept. 2020. Available at <https://www.vilesandbeckman.com/truck-driver-fatigue/#:~:text=Truck%20driver%20fatigue%20is%20a,Large%20Truck%20Crash%20Causation%20Study>.

[3] The ELD Mandate: A Trucking Industry 101. Available at <https://www.eldfacts.com/eld-mandate/#:~:text=The%20Federal%20Motor%20Carrier%20Safety,comply%20passed%20in%20December%202017.>.

[4] William B Cassidy. Truckers step up ELD protests as mandate looms. Dec. 2017. Available at <https://www.joc.com/regulation-policy/transportation-regulations/us-transportation-regulations/truckers-step-eld-protests-mandate-looms_20171206.html>.

[5] Julie Weed. Wearable Tech That Tells Drowsy Truckers It's Time to Pull Over. Feb. 2020. Available at <https://www.nytimes.com/2020/02/06/business/drowsy-driving-truckers.html>.

[6] SmartCap Technologies: Our Product. Nov. 2020. Available at <http://www.smartcaptech.com/life-smart-cap/>.

[7] Seeing Machines: Technology. June 2020. Available at <https://www.seeingmachines.com/technology/>.

[8] Charlotte Jacobé de Naurois et al. Detection and prediction of driver drowsiness using artificial neural network models. Paper presented at the: *Accident Analysis & Prevention* 126 (2019), pp. 95–104.

[9] M. Omidyeganeh, A. Javadtalab and S. Shirmohammadi. Intelligent driver drowsiness detection through fusion of yawning and eye closure. Paper presented at the: 2011 IEEE International Conference on Virtual Environments, Human-Computer Interfaces and Measurement Systems Proceedings. 2011, pp. 1–6. DOI: 10.1109/VECIMS.2011.6053857.

[10] S Park et al. Driver drowsiness detection system based on feature representation learning using various deep networks. Paper presented at the: *Springer* (2016), pp. 154–164.

[11] J Yu et al. Representation learning, scene understanding, and feature fusion for drowsiness detection. Paper presented at the: *Springer* (2016), pp. 165–177.

[12] T-H Shih and C-T Hsu. MSTN: Multistage spatial-temporal network for driver drowsiness detection. Paper presented at the: *Springer* (2016), pp. 146–153.

[13] Jing-Ming Guo and Herleeyandi Markoni. Driver drowsiness detection using hybrid convolutional neural network and long short-term memory. Paper presented at the: *Multimedia Tools and Applications* 78.20 (2019), pp. 29059–29087. DOI: 10.1007/s11042-018-6378-6. Available at <https://doi.org/10.1007/s11042-018-6378-6>.

[14] Jie Lyu, Zejian Yuan and Dapeng Chen. Long-term Multi-granularity Deep Framework for Driver Drowsiness Detection. Paper presented at the: *CoRR* abs/1801.02325 (2018). arXiv: 1801.02325. Available at <http://arxiv.org/abs/1801.02325>.

[15] Reza Ghoddoosian, Marnim Galib and Vassilis Athitsos. A Realistic Dataset and Baseline Temporal Model for Early Drowsiness Detection. 2019. arXiv: 1904.07312 [cs.CV].

[16] Patrice Ferlet. How to work with Time Distributed data in a neural network. Nov. 2019. Available at <https://medium.com/smileinnovation/how-to-work-with-time-distributed-data-in-a-neural-network-b8b39aa4ce00>.

[17] Olah Christopher. Understanding LSTM Networks. Sept. 2015. Available at <https://colah.github.io/posts/2015-08-Understanding-LSTMs/>.

[18] Anil Kaynar. Extract Eyes in Face İmages With Dlib and Face Landmark Points. Oct. 2019. Available at <https://medium.com/@aanilkayy/extract-eyes-in-face-i%CC%87mages-with-dlib-and-face-landmark-points-c45ef480c1>.