CIS 520 - Project Final Report

Drowsiness Detection with Machine Learning

Team Name: SATA

Siming He, **A**hmad Amine, **T**anmay Verma, **A**dithyakrishna Venkatesh Hanasoge

***Abstract*—Drowsiness and Fatigue are amongst the significant causes of road crashes. Every year, there is an increase in the number of crashes due to the above two reasons. In this project, we propose an algorithm to detect this condition. We will have 3 labels : 0 for Alert, 5 for partially drowsy, and 10 for drowsy. Our dataset consists of 3 videos of 60 persons, where for each person there is an alert(0), partially drowsy(5) and drowsy(10) video. We will be sampling each of these videos at a rate of 1 frame per second to obtain images which form our dataset. We then detect faces from each of these images(frames) using SSD Resnet 10 and perform face landmark extraction using OpenCV. Subsequently, we will train different models and compare the performance of each model and propose the best model based on accuracy obtained on the test set.**

I. Motivation

Drowsy driving is a prevalent problem in today's world. The NHTSA estimates that there were 697 deaths from drowsy-driving related crashes in 2019 [**?**]. Additionally, [**?**] estimates that in 2017, drowsy driving was responsible for 50,000 injuries and nearly 800 deaths. Aside these casualties, the NHTSA estimates that these crashes "cost society $109 billion annually, not including propety damage"[**?**]. The main cause of drowsy driving is attributed to fatigue and lack of sleep, but "tackling these issues can be difficult when our lifestyle does not align with avoiding drowsy driving" [**?**]. Alternatively,[**?**] provides many tips to avoid drowsy driving, including stopping driving, scheduling a break every two hours or 100 miles, or drinking a caffeinated beverage with a 30 minute nap for the caffeine to enter the bloodstream. As such, a system that can detect drowsy drivers is a useful tool that can be used to direct the driver to some of these tips before it's too late. These crashes can then be avoided either by alerting the driver to keep them awake or potentially taking control from the driver to ensure safe stopping. Accordingly, we have set out to design an image-based system that can detect whether a given subject in an image looks drowsy in order to then take the suitable action (i.e alert or take control).

II. Related Work

Several people have tackled the problem of drowsiness detection in driving. A student group from UT Austin extracted the major facial landmarks like eye aspect ratio, mouth aspect ratio, pupil circularity, and mouth aspect ratio over eye aspect ratio from the image and used it to train various different models including naive bayes, logistic regression, KNN, MLP, Decision Trees, Random Forest, CNN, LSTM. LSTM and KNN gave the highest accuracy but KNN had high false negative rate than LSTM, so it was suggested to use LSTM as the final model. The overall accuracy achieved was 77%.

Another paper from UT Arlington suggests the use of blink sequence as the feature set for drowsiness detection using Hierarchical Multiscale Long Short-Term Memory (HMLSTM) network. They used human judgement as a baseline to compare the model, and the model was claimed to be performing better than the baseline.

Focusing on reduction of the heavy baseline models to a light weight model to allow the deployment of the model on the embedded systems, the research group from Samsung trained the baseline models using reduced feature set like the sub images of the eyes, mouth to recognize whether driver is drowsy or not. The proposed model achieved an accuracy of 89.5%.

Finally, another paper trying to detect drowsiness, modelled the use of CNN for feature extraction followed by feeding the output to the LSTM network giving overall a hybrid of CNN and LSTM.

III. Data Set

A. Available Data Sets:

There are two main datasets available online that we had planned to use for the purpose of drowsiness detection:

1) **UTA Real-Life Drowsiness Dataset**: Publically available dataset provided by the University of Texas at Arlington which "consists of 180 RGB videos, each around 10 minutes long, and is labeled as belonging to one of three classes: alert (labeled as 0), low vigilant (labeled as 5) and drowsy (labeled as 10)".

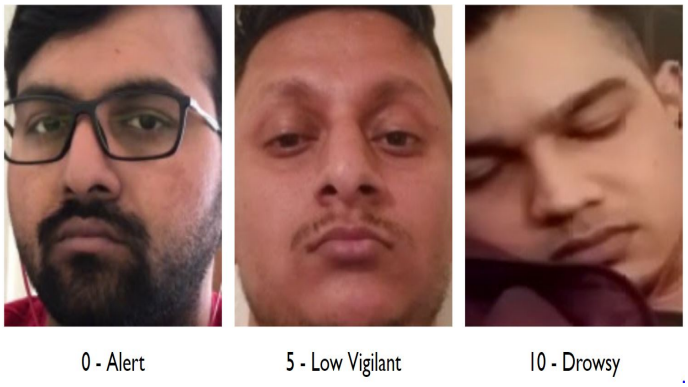

Fig. 1: Three classes

Dataset Parameters:

- P: For this dataset, the number of features p is simply the resolution of these videos but in this case, videos are of various resolutions and qualities. For our use-case, we will be standardizing all videos to 640x480 pixels for a total p size of $307,200$ if we consider all pixels.
- N: Since we are only interested in taking image frames, we will be downsampling the video frames down to 1 Hertz, or 1 frame a second. In this case, we will have a total number of samples $N = 1frame/second \times 60seconds/minute \times 10minutes/video \times 180videos = 108,000$

2) **Driver Drowsiness Detection Dataset**: Access restricted by License agreeement, provided by National Tsuing Hua University Computer Vision Lab. This dataset consists of 9 and a half hours of 640x480 videos in AVI format, captured at 15 frames a second and 30 frames a second.

Dataset Parameters:

- P: For this dataset, the number of featrues p is simply the resolution of these videos. Since videos have a resolution of 640x480 pixels, then $p = 307,200$ if we consider all pixels.
- N: Since we are only interested in taking image frames, we will be downsampling the video frames down to 1 Hertz, or 1 frame a second. In this case, we will have a total number of samples $N = 1frame/second \times 60seconds/minute \times 570minutesindataset = 34,200$
- Since our $P >> N$, it is necessary that we reduce our features to a manageable size.

Although we had gotten Dr. Ungar's signature for the agreement needed to get access to the second dataset well in advance, the owners did not give us access to the dataset till very recently. As such, we were not able to move forward with this dataset considering the time constraints. Instead, our work in this project was based on the UTA Real-Life Drowsiness Dataset.

B. Data Analysis:

As a first data analysis step, we will be looking at our data in its original form:

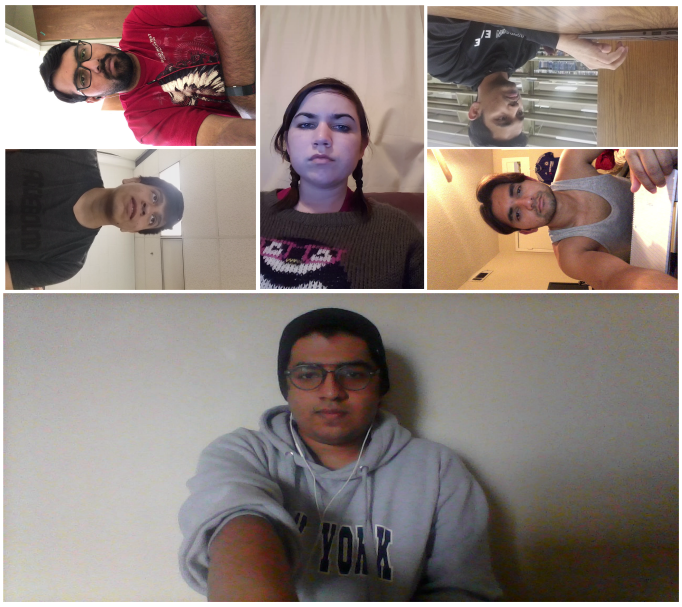

Fig. 2: Six Samples (first frame of video) from our dataset

There are a number of points that we can immediately note:

1) Our dataset is relatively diverse, with different skin tones, sexes, lighting conditions as well as samples where the subject is wearing glasses.
2) Our data is of different orientations:
   a) Straight up like the one at the bottom.
   b) Sideways like the images to the left.
   c) Upside-down like the image to the top right.

3) Our dataset is of varying image formats and sizes:
    a) **Portrait Orientation**: like the in the middle of dimensions 1080×1920
    b) **Landscape Orientation**: like the one at the top right of dimensions 1920×1080
    c) **Different Resolutions**: like the one at the bottom, of size 1280×720
    d) **Same Frame Rates:** All videos are captured at 30 FPS.

Additionally, we note the following:

1) The original data set is over 100 Gigabytes in size: Working with this dataset will make data cleaning and model training very slow.
2) We only have 3 classes of categorical data: each video has one label all through-out. As such, it would be very redundant to read all frames in a video due to the lack of per-frame annotations (i.e events like yawning).

As such, we need to resize and reorient our data so that all images are of the same size (height and width) and same orientation (straight-up). Additionally we need to handle the huge size (110GB) of this data-set. As a solution to both of these problems, we propose a number of feature extraction and data compression techniques in the next subsection.

C. Data Compression and Feature Extraction:

There are several approaches that we can take to down-sample and compress our data. We start off by down-sampling our videos.

1) Down-sampling: Down-sampling our videos can be simply done by extracting K frames each second from each video, where K is a hyper-parameter. At best, this will reduce the amount of data we have by a factor of $\frac{K}{30}$. However, due to video compression algorithms underlying .mp4 and .MOV files, the generated sampled images dataset is actually larger than the original dataset: the sampled frames came out to a total size of $\approx$ 100 Gigabytes which is 36.4% larger than the original dataset size (110 Gigabytes). However, we can now make use of these images to extract the most prominent features in the dataset and use those for training.

2) Feature Extraction: Now that we have our frames sampled, we have several approaches we can take to extract features from the dataset. These approaches are listed in order of compression size as follows:

- Image Resizing: By resizing all images to 480p (i.e 640x480 pixels), we would standardize our data as well as reducing the size of the dataset by 33.33% to $\approx$ $99.5GB$ calculated as follows: 60people $\times$3videos/person $\times(\approx$ 10)minutes/video $\times$60seconds/minute $\times$480height pixels $\times$640width pixels $\times$3channels/pixel $\approx$ $99.5GB$. However, this approach is not ideal as this incurs distortions (stretching) in the image due to resizing.
- **Face Landmarks:** Extracted using **extract_face_landmarks** function from the mlxtend library for a total of 68 feature points. This would equivalently reduce each frame to just 68.

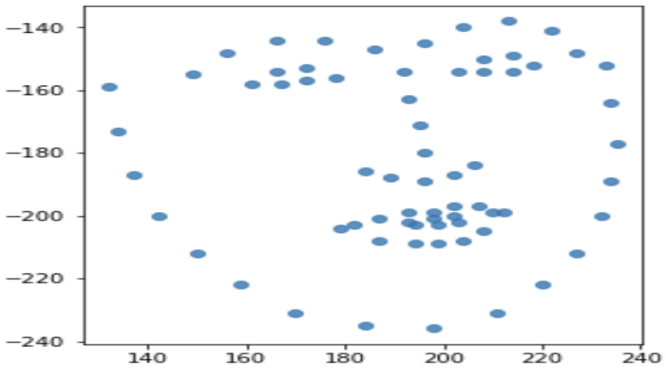

Fig. 3: Landmarks generated from images

- **Feature Construction:** Only Eye and mouth Face landmarks are needed from the above landmarks for a total of 31 points per frame.

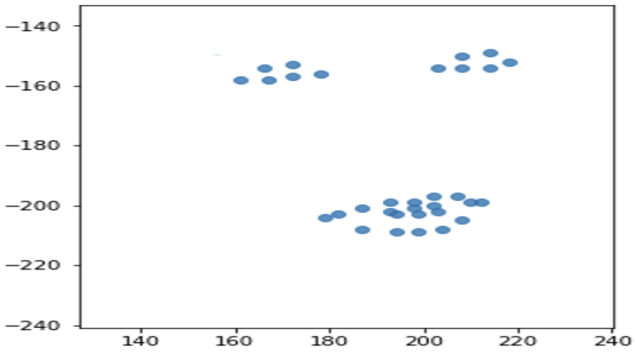

Fig. 4: Eye and mouth face landmarks

Features are then extracted including the following:

– Eye Aspect Ratio (EAR) - Ratio of length of the eyes to the width of the eyes

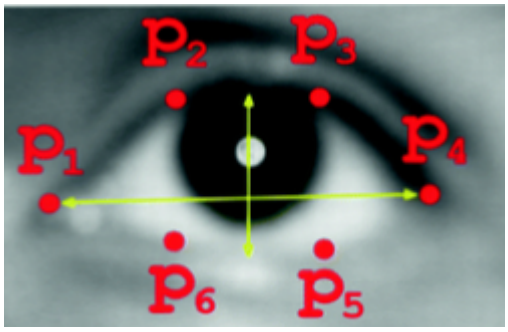

Fig. 5: EAR

– Mouth Aspect Ratio (MAR) - Ratio of length of the mouth to the width of the mouth

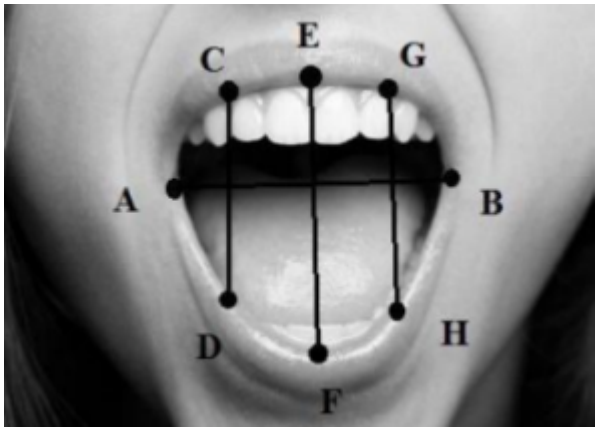

Fig. 6: MAR

– Pupil Circularity (PUC) - A measure complementary to EAR but places greater emphasis on circularity on the pupil instead of the eye
– Mouth aspect ratio over Eye aspect ratio (MOE) - It is simply the ratio of MAR over EAR

- CSV created for each person for alert, partially drowsy and drowsy conditions. CSV forms the data on which training was performed. Extracting these features took on average 3 hours to complete per person on the sampled frames using Google Colab. This would mean that feature extraction would take 7 and a half days of continuous runtime to extract all of the dataset. This however, is impossible to achieve due to Colab's 12 hour runtime limit. As such, feature extraction using just the sampled frames was unfeasible and we had to explore additional approaches.
- **Face Extraction:** Considering that all our information lies in our subject's face, our data can be significantly reduced in size by extracting only the faces in the sample frames. To do this, we use the single-shot detector (SSD) with Resnet10 pre-trained Caffe model. This proved to be very reliable and robust to the subject's orientation (i.e if they are slanted down or to the side). From this detected face, we select a 224×224 centered around the detection's center. We chose this size due to its ubiquitous use in commonly used pre-trained models like VGG-16 and ResNet. After running this algorithm on all of our data, we had to sift through the results and discard any false-positives that may have been detected. The resulting dataset size is only 6.63 GigaBytes, which is 6.03% of the original dataset and only 4.42% of the sampled frames' dataset.

3) Drowsiness MNIST Construction: Inspired by MNIST and the relatively poor performance obtained using the original dataset, we decided to try and construct our own dataset, in a bid to minimize the noise in the original images and to allow for faster training and testing (potentially) using simpler models. We name this dataset "Drowsiness MNIST" or "DNIST" based on the inspiration by MNIST. Unlike MNIST, this dataset was constructed into a 224×224 binary image to allow for easier use with more complex models like VGG16 and ResNet. This dataset was constructed by using the extracted face landmark features discussed previously as a binary mask and then drawing in the contours of this binary mask by linking the points together. Resulting samples from the dataset can be seen in figure 7. We had planned on using this dataset to train with our own custom CNNs and MLPs as well as transfer learning, but due to time constraints, we were unable to train models on this dataset.

Fig. 7: DNIST Samples for labels 0 through 10

4) Data Cleaning: Before running any of the feature extraction algorithms, we have to cleanup the dataset. Our dataset is mostly clean with correct labels and good videos. As such, we only have to reorient our subjects in the dataset to all be straight-up. To achieve this, we tried the following two approaches:

1) **Facial Landmark Test:** We check if facial landmarks exist in the frame; if not, we reorient the image by transposing and flipping. Even with this, if landmarks did not exist, we used mean imputation to account for missing data in hope for better accuracy.
2) **Manual Cleaning:** To provide the cleanest set of data, we sifted through each video and marked whether the resulting frames would need a transpose, a flip, or a transpose followed by a flip. We then go through the frames and apply the transformations based on the marking. This proved to be the safest and cleanest way to approach the problem as we wanted to ensure that we had the cleanest data possible considering how long the feature extraction would take and retakes would thus be too time consuming.

The final result of our data cleaning process, paired with the face extraction discussed previously can be seen below.

Fig. 8: Original image (left) and the resultant extracted face (right)

5) Limitation of Extraction of 4 Features: To learn more about the extracted features, we plot the distribution of data points for each of the lebels. The four features are represented as x-axis, y-axis, z-axis, and color of the data points as shown in Fig. 5. The data points for Alert, Low Vigilant, and Drowsy are not separated clear. It indicates that we may not get good classification result by solely using the four feature. Our experiment result in the next section aligned with this hypothesis.

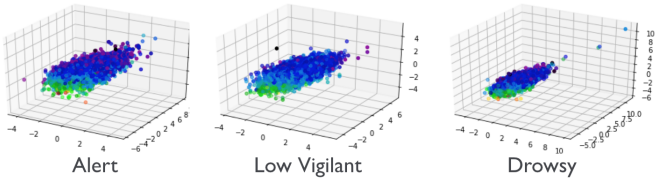

Fig. 9: Data Distribution for Each Label

6) Time Series Data: Video frames are sequential data. In addition to using each frame separately for classification. We believe that facial expressions in a time period would reveal more information about whether a person is drowsy or not. For example, drowsy people may close eyes more frequently or have smaller EAR overtime. Using time series data would capture the information. To create the time series data, we have a sliding window of size T (the number of frames for each time series data) with overlap of 20%. For example, when $T = 30$, if the first data point is from frame 1 to frame 30, the second data point is from frame 6 to 36. As a result, the dimension of the input time series data is $N \times T \times M$ where N is the number of data, T is the time step, and M is the number of features.

IV. Problem Formulation

In this project, we are interested in detecting drowsiness by using images/videos of human faces. Since we have labeled data, this is a supervised learning problem. Given face features X (key points on face or face image) and labels $Y \in \{-1, 1\}$ (drowsy and alert), we will mainly use discriminative models such as logistic regression, KNN, decision tree, and CNN for binary classification of drowsy and alert faces. Specifically, we want to estimate parameters for $P(Y|X)$ from the training data. Then, we can use the model to predict Y from X. . Since videos are sequential data, we would try LSTM Networks and predict Y by a sequence of data $P(Y|X_1, X_2, \cdots, X_t)$.

V. Methods

A. Models:

We tested using the following methods, in order of complexity:

1) Simplest Predictor: Random coin flips.
   We use the DummyClassifier provided by sklearn. When the classifier always predicts the most frequently label in the training set, we get the accuracy of 0.354. When the classifier makes prediction uniformly at random, we get the accuracy of 0.334.
2) Simple Logistic Regression on the four features extracted
   We evaluate logistic regression model with different strength of L2 penalty.

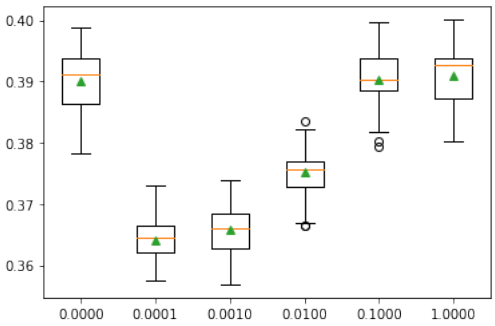

Fig. 10: Logistic Regression models with different L2 penalty

As show in the figure, model without penalty works best. We further evaluated the model with repeated stratified 10-fold cross-validation. After doing three repeats with ten folds, we get the following results:

	Mean
Accuracy	0.390
F1 ('macro')	0.377
Precision ('macro')	0.392
Recall ('macro')	0.386

The macro F1, macro Precision, and macro Recall are provided by sklearn; they calculate metrics for each label, and find their unweighted mean.

3) Simple Logistic Regression on all facial landmarks
Same as previous model, we evaluate logistic regression model with different strength of L2 penalty.

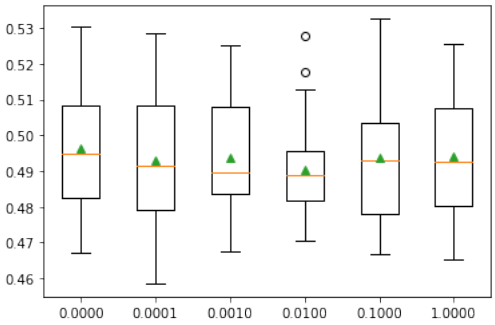

Fig. 11: Logistic Regression models with different L2 penalty

As show in the figure, model without penalty also works best. We further evaluated the model with repeated stratified 10-fold cross-validation. After doing three repeats with ten folds, we get the following results:

	Mean
Accuracy	0.496
F1 ('macro')	0.491
Precision ('macro')	0.499
Recall ('macro')	0.494

We observe that using full facial landmarks gives better result that using the four extracted features. It makes sense since four extracted features are not enough for this complex classification task.

4) K-Nearest Neighbor Classifier on the four extracted features
To make sure that using full facial landmarks outperforms using the four extracted features, we test KNN models on both data input. We test different K values from 1 to 1500 and find out that the accuracy is highest when K is 1150.

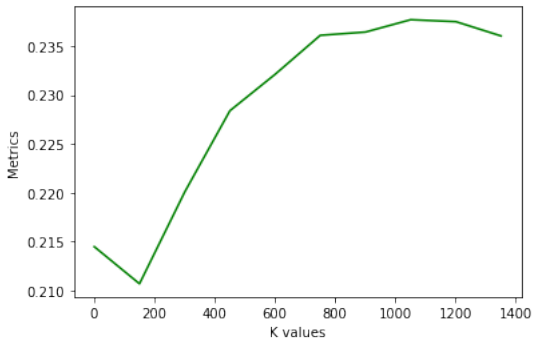

Fig. 12: KNN

As we can see, KNN performs terribly when we use the four extracted features as input. The highest accuracy is lower than the simple predictor.

5) K-Nearest Neighbor Classifier on the all face landmarks
We test different K values from 1 to 300 when we use all face landmarks as input and find out that the accuracy is highest when K is 90.

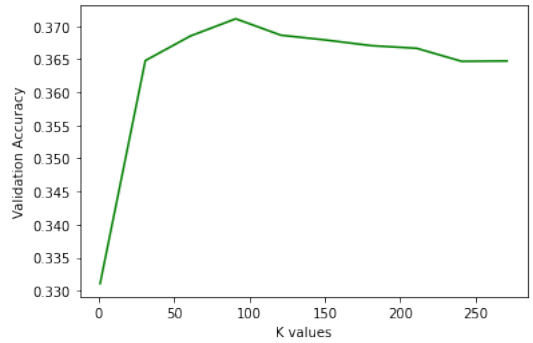

Fig. 13: KNN

When $K = 90$, we get the following result with repeated stratified 10-fold cross-validation:

	Mean
Accuracy	0.371
F1 ('macro')	0.370
Precision ('macro')	0.377
Recall ('macro')	0.373

The performance is much better than using the four extracted features as input.

6) Random Forests on all facial landmarks

For random forests models, we use grid search to fine tune the hyper-parameters. We tried tree depth of $16, 32, 64, 128$ and random state of $20, 40, 80, 160$. The best result is achieved when depth is 16 and random state is 40. The result is:

Accuracy	0.359
F1 ('macro')	0.336
Precision ('macro')	0.372
Recall ('macro')	0.362

7) Deep-Learning Approaches:

a) Transfer Learning:

We performed transfer learning using two pre-trained image-classification models: ResNet50 and VGG16. Each of these models were trained with varying learning rates, optimizers, datasets, and classification networks. The following parameters have been tested:

i) **Learning Rate:**

A) 0.0001
B) 0.001
C) 0.01
D) 0.03
E) 0.06

Eventually, we decided to stick to a learning rate of 0.01 which provided the best convergence after testing.

ii) **Optimizer:**

A) Adam Optimizer: With a learning rate of 0.001
B) Stochastic Gradient Descent (SGD): With variable learning rate and a momentum of 0.9.

While ADAM optimizer provided faster convergence, we found SGD to perform better with more epochs.

iii) **Head:** This refers to the replaced classification layer in the neural network. We used two different kinds of heads:

A) One Fully Connected Layer: This is a simple head which simply takes in all the features extracted by the model and passes through a dense layer and outputs 3 values corresponding to the 3 classes in our dataset. This is referred to as an FC Head in figure 16.
B) MaxPool2D + Dropout (20%) + a Dense Layer: This was a head recommended by one of the tutorials we found online for transfer learning.

iv) **Datasets:** We experimented with three different datasets for training:

A) Extracted Faces Dataset.
B) Downsampled Faces Dataset: This refers to the same dataset sampled uniformly as to decrease the repetition in frames in the data.
C) DNIST: Drowsiness MNIST Dataset constructed using the landmarks extracted from the faces dataset. Unfortunately, due to time constraints, we were not able to finish training using this dataset in time, but it is worth noting that we have at this point reached a testing accuracy of 42% while transfer learning using the second head and VGG16 is still running as we are writing this report.

The resulting training curves can be seen in the figures below.

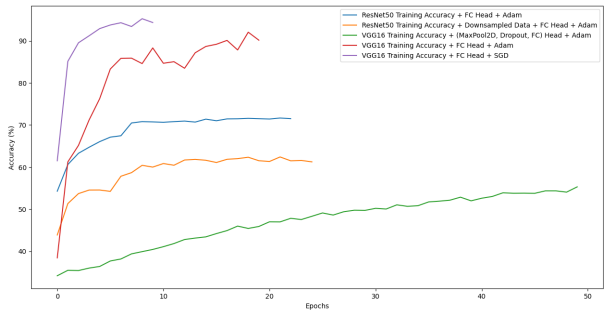

Fig. 14: Training Accuracy of the different models used

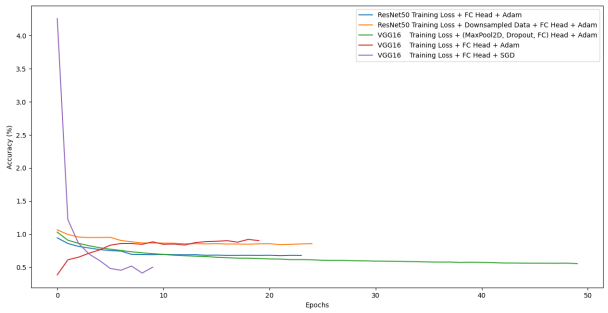

Fig. 15: Training Loss of the different models used

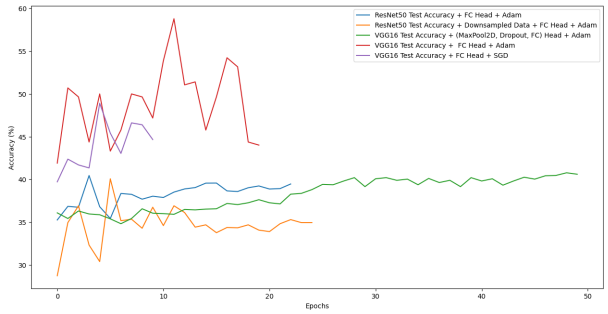

Fig. 16: Test Accuracy of the different models used

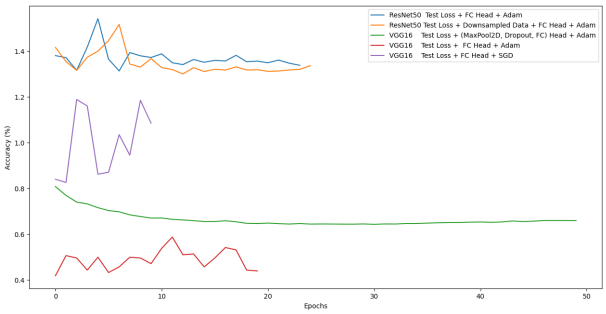

Fig. 17: Test Loss of the different models used

b) Long Short Term Memory Model

i) LSTM Model with Shuffled Input Data
Initially, we shuffle the input data before splitting the training and validation set. With 34 hidden layers, dropout rate of 0.8, initial learning rate of 0.005, learning rate decay of 0.98 per 100000 steps, and Adam optimizer, we achieved 93.8% accuracy. However, we realize that we shouldn't shuffle the input data before splitting training and validation set. Shuffling the input data before splitting will cause the repetition of a person in both training set and validation set. For example, the data point for frame 1 to frame 30 of video 1 could be in the training set while the data point for frame 6 to frame 36 of video 1 could be in the validation set.

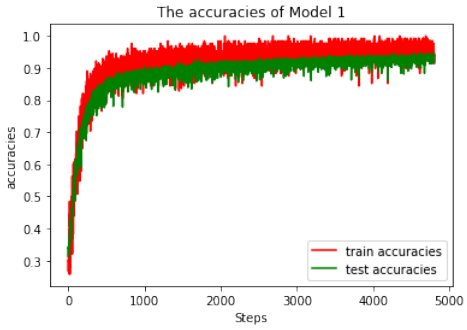

Fig. 18: LSTM Accuracy

ii) LSTM Model with Unshuffled Input Data
With 34 hidden layers, dropout rate of 0.8, initial learning rate of 0.005, learning rate

decay of 0.98 per 100000 steps, and Adam optimizer, we achieved the following result by using unshuffled input data.

Accuracy	0.512
F1	0.507
Precision	0.516
Recall	0.512

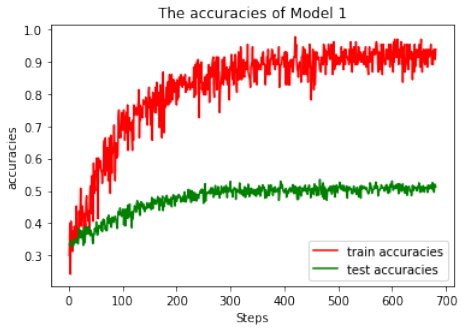

Fig. 19: LSTM Accuracy

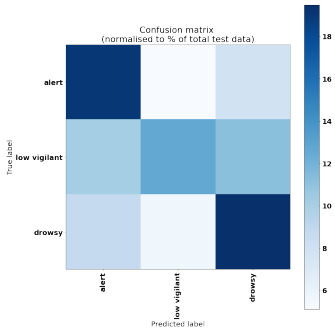

Fig. 20: LSTM Confussion Matrix

8) Auto ML: The Auto ML package used for training the model was Auto-Sklearn. Three feature sets were used separately, 1) The 4 feature set 2) All facial landmarks 3) Face and mouth landmarks. The dataset with all facial features landmarks gave the best accuracy on the validation set probably due to high number of features to train upon. The best classifiers obtained were Passive aggressive and Random forest while Adaboost and Extra features classifiers were over fitting on the training data hence excluded from the ensemble. After training the model on various time steps, per run time limit and CV folds, the best accuracy achieved was 0.46 at 1000 seconds with 35 second per run timestep. Further, we used major voting to optimize the output. Since in real life application we have all the past frames from the video, it's reasonable to make predictions on multiple frames and use major voting to get the result. We sample 60 frames (1 frame per second) and make predictions on each frame. Then, the most frequent label in the predictions is used as the result. By using major voting, we get better prediction:

Accuracy	0.523
F1	0.517
Precision	0.529
Recall	0.521

B. Evaluation

As the use case for such a system would be in automotive applications, where drowsy drivers are alerted, precision is of great importance for us. However, triggering the alert system too often can be annoying or even distracting for drivers as well. As such, it is important for us to also minimize false negatives in addition to achieving good precision. Accordingly, our model will be trained based on accuracy, but evaluated using the F1-score. The reason for this is that it is hard to train models using the F1-score, but we care about incorporating false negatives into the evaluation. Below is a formal definition of our evaluation and training metrics:

1) True Positives: Correctly classifying frames based on the state of the subject in those frames: i.e: drowsy subjects detected as drowsy (label 10), low vigilant subjects detected as low vigilant (label 5).
2) False Positives: Falsely classifying frames with alert (label 0) subjects as drowsy.
3) False Negatives: Falsely classifying frames with low vigilant (label 5) or drowsy (label 10) subjects as alert.
4) True Negatives: Correctly classifying frames with alert subjects as alert (label 0)
5) Precision: $\frac{TruePositives}{TruePositives+FalseNegatives}$
6) F1 Score: $\frac{2\times TruePositives}{2\times TruePositives+FalsePositives+FalseNegatives}$

VI. Experiments And Results:

1) Initially, we normalised our dataset and got poor accuracies and losses. We then tried using the data without normalising and achieved better accuracies and losses. This could be because the images were too noisy and therefore we noticed better accuracies and losses throughout the spectrum of models that we trained on.

2) Table of All Results
   In APPENDIX.

VII. Conclusion and Future Work

All in all, this is a pretty challenging classification problem. It's especially hard to distinguish between alert and low vigilant and between drowsy and low vigilant. In addition, the data set has a lot of noise. There are a lot of similar video frames under different labels which doesn't help the training. Even through the classification is challenging, we tried different models including classical machine learning models and modern deep learning models. Finally, we achieved 59.0% accuracy with VGG16 model.

In the future, it's helpful to find more data to solve the overfitting problem. Moreover, it's necessary to select informative frames from video instead of using all the frames, since many frames don't provide information regarding drowsiness.

VIII. References

1) Zhong, G. (2021, June 10). Drowsiness detection with machine learning. Medium. Retrieved December 13, 2021, from https://towardsdatascience.com/drowsiness-detection-with-machine-learning-765a16ca208a.
2) Reddy, B., Kim, Y.-H., Yun, S., Seo, C., amp; Jang, J. (2017). Real-time driver drowsiness detection for embedded system using model compression of deep neural networks. 2017 IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW). https://doi.org/10.1109/cvprw.2017.59
3) Ghoddoosian, R., Galib, M., amp; Athitsos, V. (2019). A realistic dataset and baseline temporal model for early drowsiness detection. 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW). https://doi.org/10.1109/cvprw.2019.00027
4) Faraji, Farnoosh Lotfi, F. Khoramdel, Javad Najafi, Ali Ghaffari, Ali. (2021). Drowsiness Detection Based On Driver Temporal Behavior Using a New Developed Dataset.

Appendix

Appendix on Next Page.

Table of All Results

	Data	Accuracy	F1	Precision	Recall
Logistic Regression	Four Extracted Features	0.390	0.377	0.392	0.386
Logistic Regression	Full Facial Landmarks	0.496	0.491	0.499	0.494
KNN	Full Facial Landmarks	0.371	0.370	0.377	0.373
Random Forest	Full Facial Landmarks	0.359	0.336	0.372	0.362
LSTM	Full Facial Landmarks	0.512	0.507	0.516	0.512
Auto-ML with Major Voting	Full Facial Landmarks	0.523	0.517	0.529	0.521
Resnet 50 with Adam	Sampled Image Dataset	0.4007			
Resnet 50 with Adam	Down-sampled images	0.4045			
VGG16 with Adam	Sampled Image Dataset	0.58			
VGG16 with SGD	Sampled Image Dataset	0.48			