Driver Drowsiness

W207 - Group 2
Ray Hung, Mayank Rai, Kirti Prakash

https://github.com/kp-commit/mids-207-group2-final-project/tree/main

Table of Contents

Dataset

Driver Drowsiness Dataset

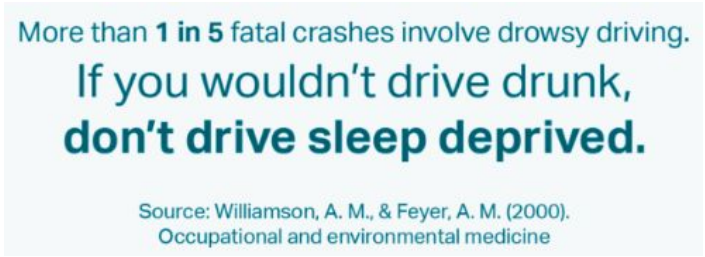
- Derived from the Real-Life Drowsiness Dataset, focusing on drivers' faces.
- Frames were extracted from videos using VLC software to create images.
- The *Viola-Jones algorithm* was applied to isolate regions of interest in the captured images.
- Initially used for training and testing Convolutional Neural Networks (CNNs) in a research paper titled "Detection and Prediction of Driver Drowsiness for the Prevention of Road Accidents Using Deep Neural Networks Techniques."

Source: Kaggle (based on paper: https://doi.org/10.1007/978-981-33-6893-4_6)

Motivation

Driver drowsiness is one of the leading causes of road accidents in the United States.

Our objective is to explore different CNN techniques with the aim of constructing a model capable of generalizing to unfamiliar images in efforts to improve public safety.

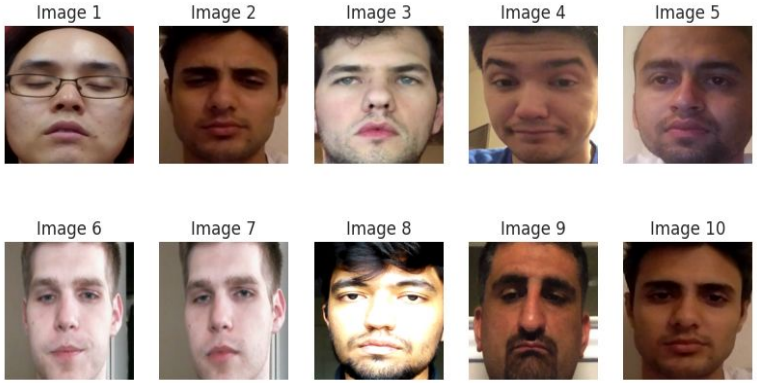


EDA

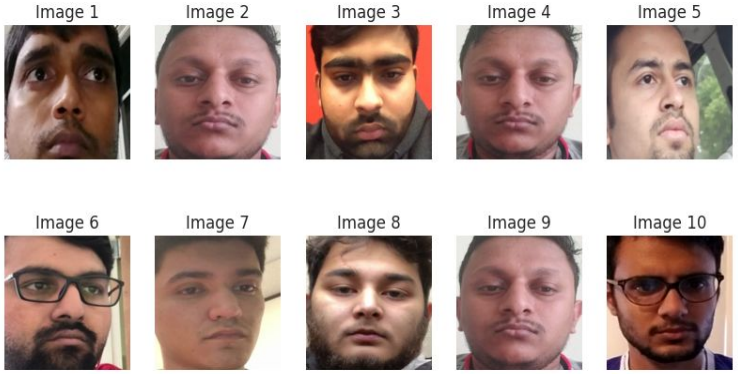
Images

total .png images: 41,793 Image Dimensions: **227 x 227** total size: **1.27GB**

Drowsy: **22,348**



Non Drowsy: **19,445**

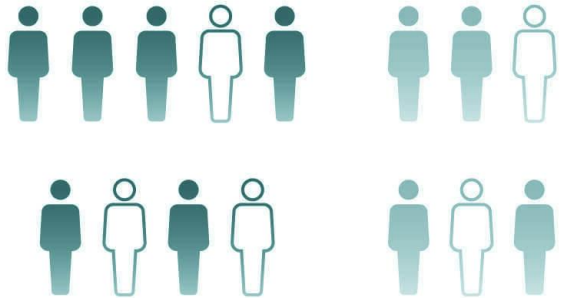


Issues with Image dataset

Karolinska Sleepiness Scale (KSS)	
Extremely alert	1
Very alert	2
Alert	3
Rather alert	4
Neither alert nor sleepy	5
Some signs of sleepiness	6
Sleepy, but no effort to keep awake	7
Sleepy, but some effort to keep awake	8
Very sleepy, great effort to keep awake, fighting sleep	9
Extremely sleepy, can't keep awake	10

- The original dataset included many **False Positive** and **False Negative** labels.
- **Stratified sampling** approach was adopted to address this problem by maintaining a balanced representation of these categories.

Stratified Sampling



Identified **26 persons represented in dataset.** Approx, **1,000 images per person.** This varies with some less resulting in a lot of noise.

Reduced data per person with sampling high quality images. Proposed **10 images per person = 260 * 2 classes** were annotated through an Inter Annotator Agreement.

150 random images each were selected from directories labeled **'Drowsy'** and **'Non-Drowsy'** to correct for **class imbalance** and then annotated with a labels for binary classification:

0 = Non Drowsy, **1** = Drowsy

Inter-annotator agreement (IAA)

Image	Kirti Review	Mayank Review	Ray Review	Fleiss' kappa
A0228.png	0.95	0.93	0.97	0.95
A0587.png	0.85	0.83	0.87	0.85
A0632.png	0.2	0.18	0.22	0.2
A0650.png	0.95	0.93	0.97	0.95

Interpreting IAA and IAR scores is not straightforward, as different annotation tasks may have varying levels of difficulty, ambiguity, subjectivity, or variability. Generally speaking, scores above 0.8 or 0.9 indicate high agreement or reliability.

Interpretation of Kappa

	Poor	Slight	Fair	Moderate	Substantial	Almost perfect
Kappa	0.0	.20	.40	.60	.80	1.0

Kappa	Agreement
< 0	Less than chance agreement
0.01–0.20	Slight agreement
0.21– 0.40	Fair agreement
0.41–0.60	Moderate agreement
0.61–0.80	Substantial agreement
0.81–0.99	Almost perfect agreement

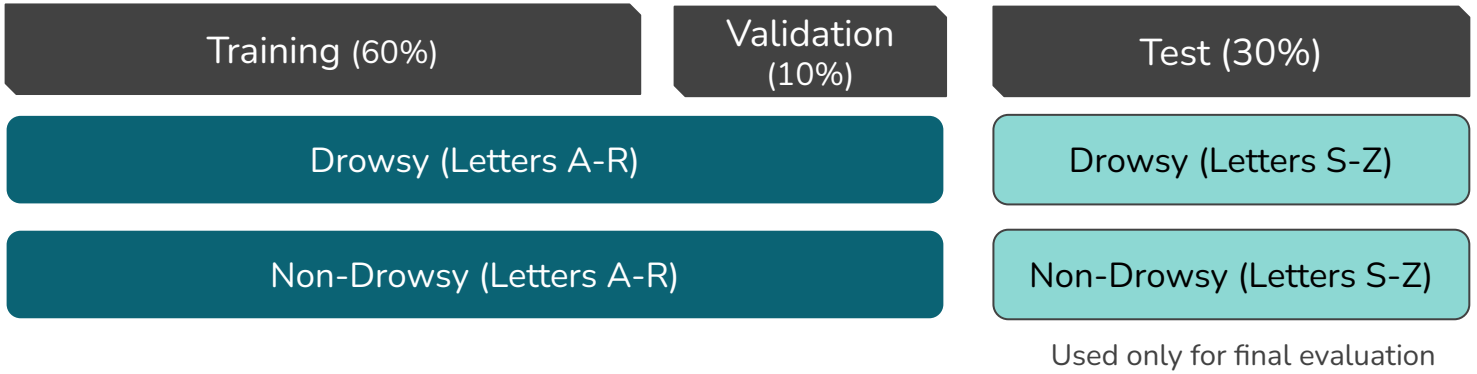
Kappa for **Drowsy**: **0.91**

Kappa for **Non Drowsy: 0.69**

Data Processing Methodology

Train/Test Split

- To account for imbalances, our team manually annotated **510 images** and then divided into separate subsets for training, validation, and testing.
- To evaluate the model's performance on previously unseen data, a distinct set of images featuring individuals with names starting from letters **S through Z** has been separated. This separation is intended to assess the model's ability to generalize to novel data.



Data processing

Train

Drowsy (206)
+
Non-Drowsy (164)

Step 1

Randomly select 150 training samples / class Drowsy(1)/NonDrowsy(0)

Balanced data selection.

Removes Biasness

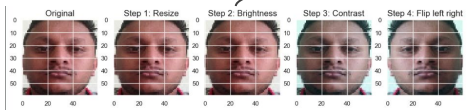
Step 2

Image resizing (80x80)
Normalization
Train/Val Split
Augmentation

Reduces model memory footprint.

Improves generalization

Train/Validation



Test

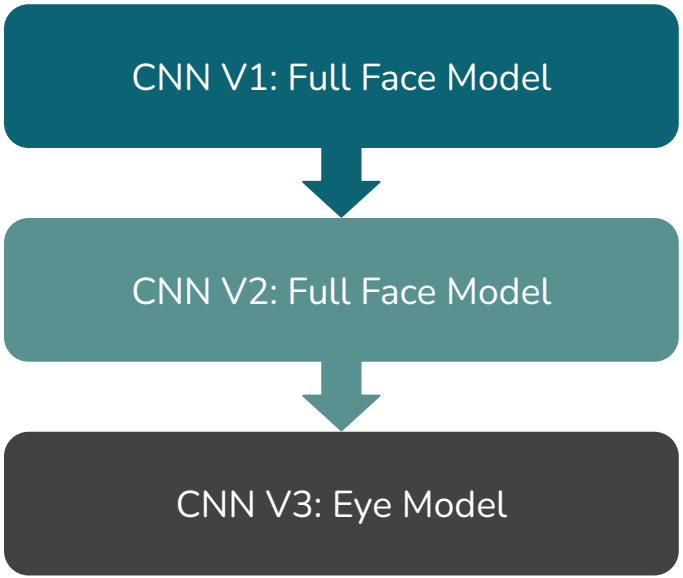
Drowsy (70)
+
Non-Drowsy (70)

Image resize(80x80)

Prediction/ Evaluation

Model Training Methodology

Approach to Model Training

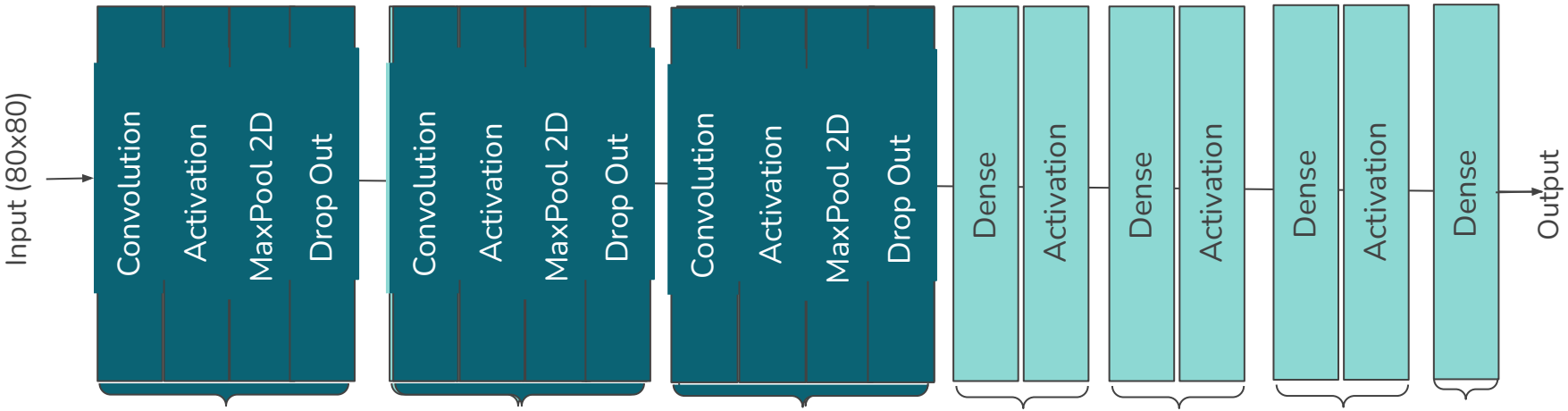


A baseline CNN model was trained using **3 convolutional filters** on **full face images**

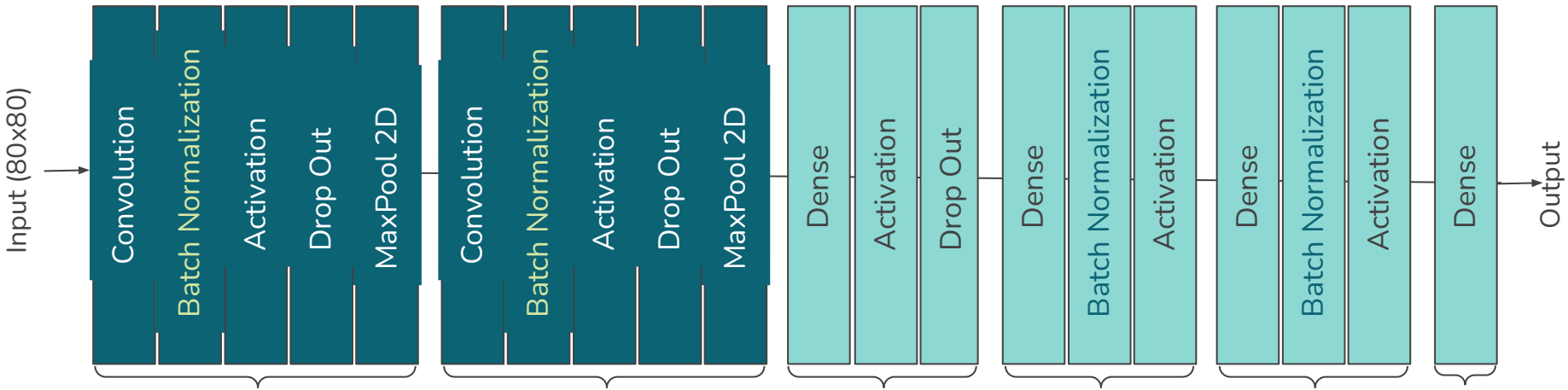
Trained with using **2 convolutional filters** on **full face images**

Trained using **2 convolutional filters** using **extracted eye images**

Baseline CNN v1: Full Face Model

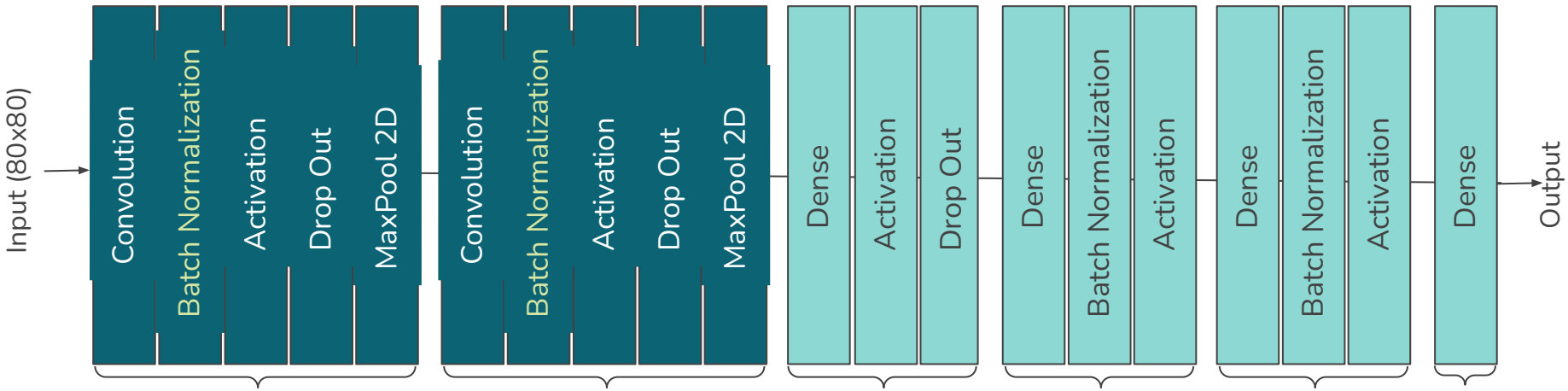


CNN v2: Full Face Model



Epochs: 30
Batch size: 32

CNN v3: Eye Model



Filters: 8
Kernel: (3,3)
Activation: relu
Strides: 1,1
Dropout: 0.3
Pool: 2,2

Filters: 16
Kernel: (3,3)
Activation: relu
Strides: 1,1
Dropout: 0.5
Pool: 2,2

Units: 128
Activation: relu
Dropout: 0.5

Units: 64
Activation: relu

Units: 32
Activation: relu

Units: 1

Epochs: 30
Batch size: 32

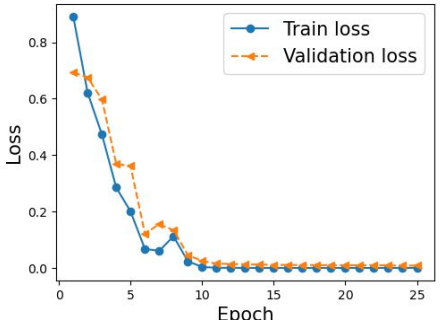
Tuning and Improvements

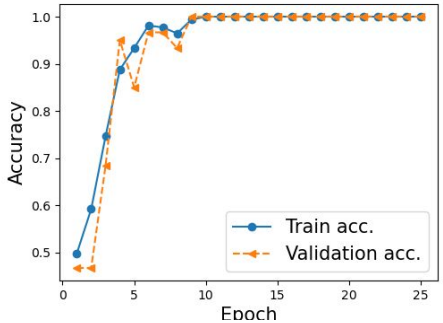
Hyperparameter Tuning

	CNN V1: Full Face	CNN V2: Full Face	CNN V2: Eye Extractor
Filters	[128, 320, 480], [128, 256, 360]	[8,8], [8, 16], [16, 8], [16, 16]	[8,8], [8, 16], [16, 8], [16, 16]
Optimizers	Adam, SGD, Adagrad	Adam	Adam
Loss	Binary Crossentropy	Binary Crossentropy	Binary Crossentropy
Kernel Sizes	(3,3), (5,5)	(3,3), (5,5)	(3,3), (5,5)
Pool Sizes	(2,2)	(2,2)	(2,2)
Learning Rate	0.001, 0.01	0.001	0.001
Batch Size	16, 32, 64	16, 32, 64	16, 32, 64
Epochs	15, 25	30, 40	30, 40

CNN v1: Baseline Model

	Learning Rate	Filters	Kernel Size	Strides	optimizer	Conv Dropout Rate	FC Dropout Rate	Units	Pool Size	Epochs	Batch Size	Train Accuracy	Validation Accuracy	Model Size
7	0.001	[128, 256, 360]	[(3, 3), (3, 3), (3, 3)]	[(1, 1), (1, 1), (1, 1)]	adam	[0.4, 0.2, 0.2]	[0, 0, 0]	[256, 128, 16]	[(2, 2), (2, 2), (2, 2)]	25	16	1.000000	1.000000	101.274052
1	0.001	[128, 320, 480]	[(3, 3), (3, 3), (3, 3)]	[(1, 1), (1, 1), (1, 1)]	adam	[0.4, 0.2, 0.2]	[0, 0, 0]	[384, 128, 32]	[(2, 2), (2, 2), (2, 2)]	25	16	1.000000	1.000000	189.126469
4	0.001	[128, 256, 360]	[(3, 3), (3, 3), (3, 3)]	[(1, 1), (1, 1), (1, 1)]	adam	[0.4, 0.2, 0.2]	[0, 0, 0]	[384, 128, 32]	[(2, 2), (2, 2), (2, 2)]	15	16	0.993750	1.000000	151.344975
2	0.001	[128, 320, 480]	[(3, 3), (3, 3), (3, 3)]	[(1, 1), (1, 1), (1, 1)]	adam	[0.4, 0.2, 0.2]	[0, 0, 0]	[256, 128, 16]	[(2, 2), (2, 2), (2, 2)]	15	16	1.000000	0.983333	126.555546
3	0.001	[128, 320, 480]	[(3, 3), (3, 3), (3, 3)]	[(1, 1), (1, 1), (1, 1)]	adam	[0.4, 0.2, 0.2]	[0, 0, 0]	[256, 128, 16]	[(2, 2), (2, 2), (2, 2)]	25	16	1.000000	0.983333	126.555546

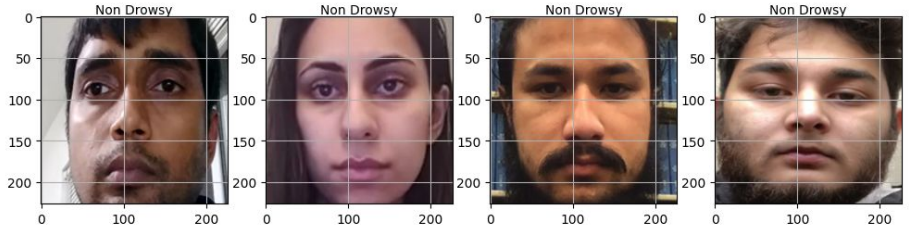




CNN v2:Full Face Image Inputs

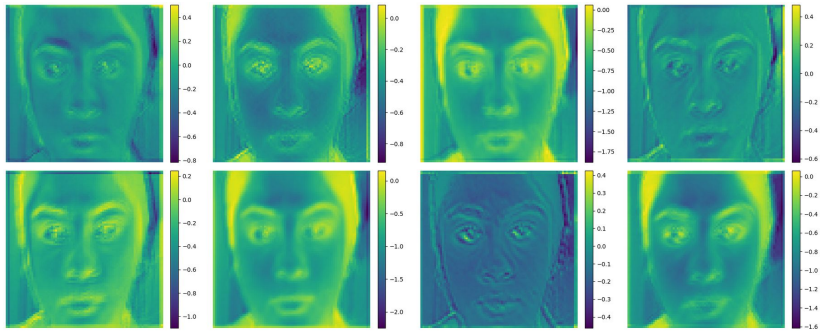
Drowsy

Non-Drowsy

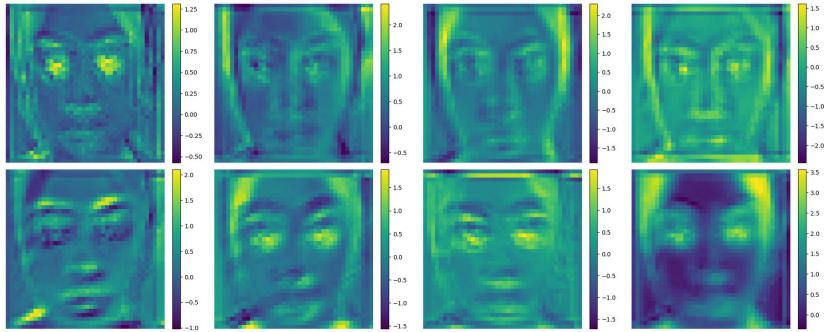


CNN v2:Convolutional Filters

Layer 1



Layer 2



Used in CNN to extract features and detect patterns with input data for full face images.

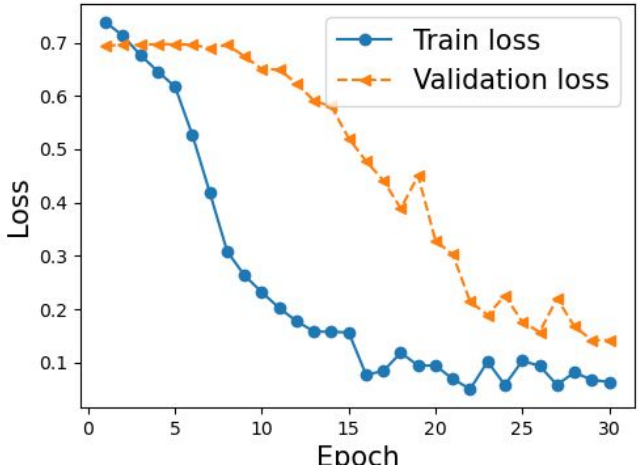
Adding additional layers increase feature diversity

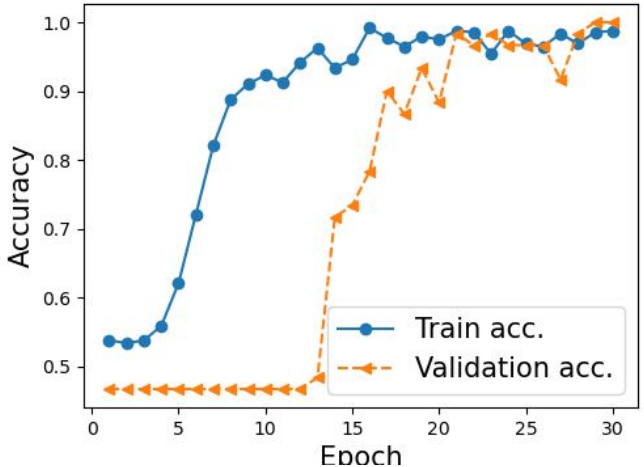
CNN v2:Hyperparameter Tuning

Learning Rate	Filters	Kernel Size	Strides	optimizer	Conv Dropout Rate	FC Dropout Rate	Units	Pool Size	Epochs	Batch Size	Train Accuracy	Validation Accuracy	Model Size
0.001	[16, 16]	[(5, 5), (5, 5)]	[(1, 1), (1, 1)]	adam	[0.3, 0.5]	[0.5, 0, 0]	[128, 64, 32]	[(2, 2), (2, 2)]	40	64	0.979166686534882	1	3.19610977172852
0.001	[16, 16]	[(3, 3), (3, 3)]	[(1, 1), (1, 1)]	adam	[0.3, 0.5]	[0.5, 0, 0]	[128, 64, 32]	[(2, 2), (2, 2)]	40	32	0.987500011920929	1	3.17755508422852
0.001	[16, 16]	[(3, 3), (3, 3)]	[(1, 1), (1, 1)]	adam	[0.3, 0.5]	[0.5, 0, 0]	[64, 64, 32]	[(2, 2), (2, 2)]	30	32	0.987500011920929	1	1.59918594360352
0.001	[16, 16]	[(5, 5), (5, 5)]	[(1, 1), (1, 1)]	adam	[0.3, 0.5]	[0.5, 0, 0]	[64, 64, 32]	[(2, 2), (2, 2)]	40	32	0.979166686534882	1	1.59918594360352
0.001	[8, 16]	[(5, 5), (5, 5)]	[(1, 1), (1, 1)]	adam	[0.3, 0.5]	[0.5, 0, 0]	[128, 64, 32]	[(2, 2), (2, 2)]	30	16	0.977083325386047	1	3.18146133422852
0.001	[8, 16]	[(5, 5), (5, 5)]	[(1, 1), (1, 1)]	adam	[0.3, 0.5]	[0.5, 0, 0]	[64, 64, 32]	[(2, 2), (2, 2)]	30	32	0.975000023841858	1	1.60309219360352
0.001	[8, 16]	[(5, 5), (5, 5)]	[(1, 1), (1, 1)]	adam	[0.3, 0.5]	[0.5, 0, 0]	[64, 64, 32]	[(2, 2), (2, 2)]	40	16	0.987500011920929	1	1.60309219360352
0.001	[8, 8]	[(5, 5), (5, 5)]	[(1, 1), (1, 1)]	adam	[0.3, 0.5]	[0.5, 0, 0]	[64, 64, 32]	[(2, 2), (2, 2)]	30	16	0.945833325386047	1	0.815586090087891
0.001	[8, 8]	[(3, 3), (3, 3)]	[(1, 1), (1, 1)]	adam	[0.3, 0.5]	[0.5, 0, 0]	[64, 64, 32]	[(2, 2), (2, 2)]	40	32	0.981249988079071	1	0.810214996337891
0.001	[16, 16]	[(5, 5), (5, 5)]	[(1, 1), (1, 1)]	adam	[0.3, 0.5]	[0.5, 0, 0]	[128, 64, 32]	[(2, 2), (2, 2)]	30	16	0.962499976158142	0.983333349227905	3.19610977172852
0.001	[16, 16]	[(3, 3), (3, 3)]	[(1, 1), (1, 1)]	adam	[0.3, 0.5]	[0.5, 0, 0]	[128, 64, 32]	[(2, 2), (2, 2)]	30	16	0.989583313465118	0.983333349227905	3.17755508422852
0.001	[16, 16]	[(3, 3), (3, 3)]	[(1, 1), (1, 1)]	adam	[0.3, 0.5]	[0.5, 0, 0]	[128, 64, 32]	[(2, 2), (2, 2)]	40	16	0.987500011920929	0.983333349227905	3.17755508422852
0.001	[16, 16]	[(3, 3), (3, 3)]	[(1, 1), (1, 1)]	adam	[0.3, 0.5]	[0.5, 0, 0]	[64, 64, 32]	[(2, 2), (2, 2)]	40	64	0.981249988079071	0.983333349227905	1.59918594360352
0.001	[16, 8]	[(5, 5), (5, 5)]	[(1, 1), (1, 1)]	adam	[0.3, 0.5]	[0.5, 0, 0]	[128, 64, 32]	[(2, 2), (2, 2)]	40	16	0.993749976158142	0.983333349227905	1.62125015258789
0.001	[16, 8]	[(5, 5), (5, 5)]	[(1, 1), (1, 1)]	adam	[0.3, 0.5]	[0.5, 0, 0]	[128, 64, 32]	[(2, 2), (2, 2)]	40	32	0.987500011920929	0.983333349227905	1.62125015258789
0.001	[16, 8]	[(5, 5), (5, 5)]	[(1, 1), (1, 1)]	adam	[0.3, 0.5]	[0.5, 0, 0]	[64, 64, 32]	[(2, 2), (2, 2)]	30	64	0.987500011920929	0.983333349227905	0.824131011962891
0.001	[16, 8]	[(3, 3), (3, 3)]	[(1, 1), (1, 1)]	adam	[0.3, 0.5]	[0.5, 0, 0]	[128, 64, 32]	[(2, 2), (2, 2)]	30	16	0.979166686534882	0.983333349227905	1.61050796508789
0.001	[16, 8]	[(3, 3), (3, 3)]	[(1, 1), (1, 1)]	adam	[0.3, 0.5]	[0.5, 0, 0]	[128, 64, 32]	[(2, 2), (2, 2)]	40	16	0.985416650772095	0.983333349227905	1.61050796508789
0.001	[16, 8]	[(3, 3), (3, 3)]	[(1, 1), (1, 1)]	adam	[0.3, 0.5]	[0.5, 0, 0]	[128, 64, 32]	[(2, 2), (2, 2)]	40	32	0.989583313465118	0.983333349227905	1.61050796508789
0.001	[16, 8]	[(3, 3), (3, 3)]	[(1, 1), (1, 1)]	adam	[0.3, 0.5]	[0.5, 0, 0]	[64, 64, 32]	[(2, 2), (2, 2)]	40	32	0.985416650772095	0.983333349227905	0.813388824462891
0.001	[8, 16]	[(5, 5), (5, 5)]	[(1, 1), (1, 1)]	adam	[0.3, 0.5]	[0.5, 0, 0]	[128, 64, 32]	[(2, 2), (2, 2)]	30	32	0.979166686534882	0.983333349227905	3.18146133422852

CNN v2: Best Model

Learning Rate	Filters	Kernel Size	Strides	optimizer	Conv Dropout Rate	FC Dropout Rate	Units	Pool Size	Epochs	Batch Size	Train Accuracy	Validation Accuracy	Model Size
0.001	[16, 16]	[(3, 3), (3, 3)]	[(1, 1), (1, 1)]	adam	[0.3, 0.5]	[0.5, 0, 0]	[64, 64, 32]	[(2, 2), (2, 2)]	30	32	0.987500011920929	1	1.59918594360352





CNN v2: Full Face Model Predictions

GT: Non Drowsy
Pr(Drowsy)=60%

GT: Drowsy
Pr(Drowsy)=86%

GT: Drowsy
Pr(Drowsy)=68%

GT: Non Drowsy
Pr(Drowsy)=3%

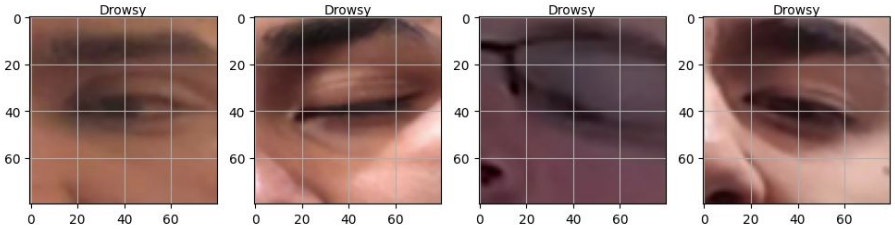
GT: Non Drowsy
Pr(Drowsy)=31%

GT: Drowsy
Pr(Drowsy)=76%

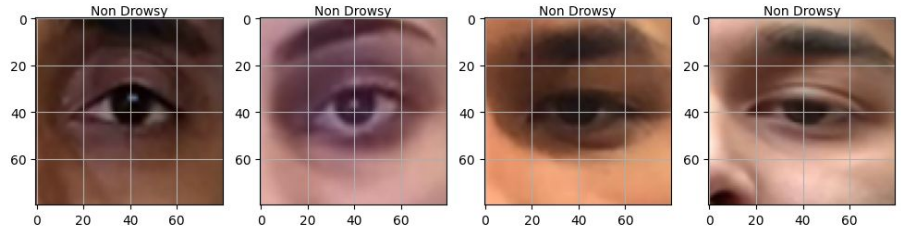
GT: Non Drowsy
Pr(Drowsy)=40%

GT: Non Drowsy
Pr(Drowsy)=20%

CNN v3: Eye Model Inputs



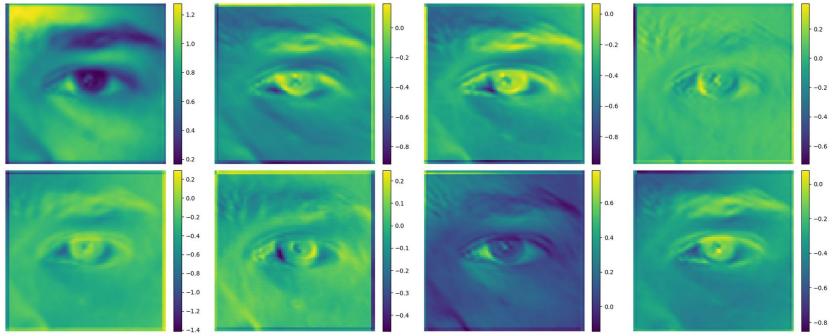
Drowsy



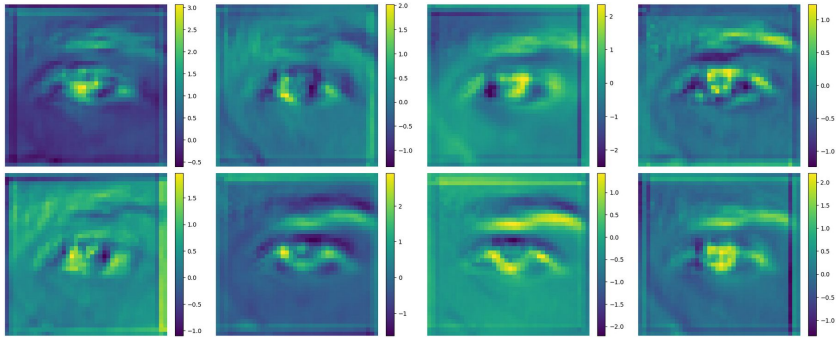
Non-Drowsy

CNN V3: Convolutional Filters

Layer 1



Layer 2

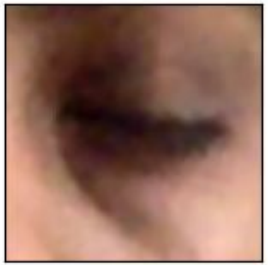


Applied the convolution filters to extract features and patterns from the eye extractor input data

CNN V3: Eye Model Predictions

GT: Non Drowsy
Pr(Drowsy)=83%

GT: Drowsy
Pr(Drowsy)=98%

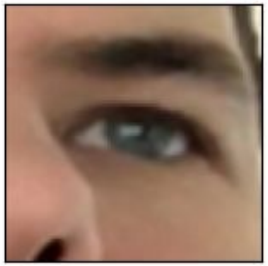


GT: Drowsy
Pr(Drowsy)=94%

GT: Non Drowsy
Pr(Drowsy)=12%

GT: Non Drowsy
Pr(Drowsy)=11%

GT: Drowsy
Pr(Drowsy)=99%



GT: Non Drowsy
Pr(Drowsy)=17%

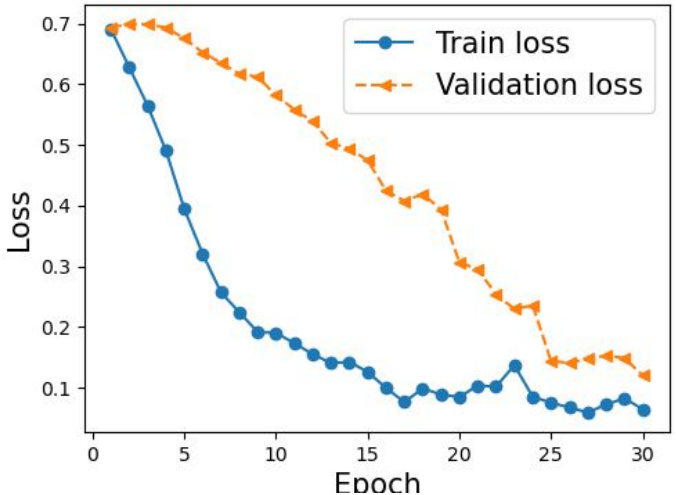
GT: Non Drowsy
Pr(Drowsy)=29%

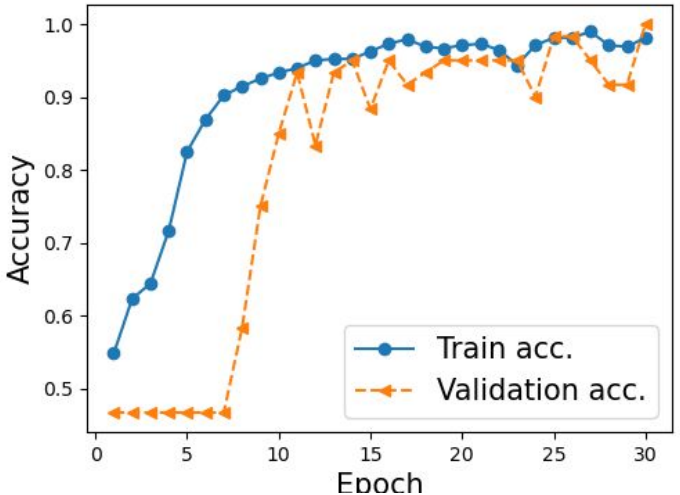
CNN V3: Hyperparameter tuning

Learning Rate	Filters	Kernel Size	Strides	optimizer	Conv Dropout Rate	FC Dropout Rate	Units	Pool Size	Epochs	Batch Size	Train Accuracy	Validation Accuracy	Model Size
0.001	[16, 8]	[(5, 5), (5, 5)]	[(1, 1), (1, 1)]	adam	[0.3, 0.5]	[0.5, 0, 0]	[128, 64, 32]	[(2, 2), (2, 2)]	40	16	0.977083325386047	1	1.62125015258789
0.001	[8, 16]	[(3, 3), (3, 3)]	[(1, 1), (1, 1)]	adam	[0.3, 0.5]	[0.5, 0, 0]	[128, 64, 32]	[(2, 2), (2, 2)]	30	32	0.981249988079071	1	3.17218399047852
0.001	[8, 16]	[(3, 3), (3, 3)]	[(1, 1), (1, 1)]	adam	[0.3, 0.5]	[0.5, 0, 0]	[128, 64, 32]	[(2, 2), (2, 2)]	40	16	0.975000023841858	1	3.17218399047852
0.001	[8, 8]	[(3, 3), (3, 3)]	[(1, 1), (1, 1)]	adam	[0.3, 0.5]	[0.5, 0, 0]	[128, 64, 32]	[(2, 2), (2, 2)]	40	16	0.954166650772095	1	1.60733413696289
0.001	[16, 8]	[(3, 3), (3, 3)]	[(1, 1), (1, 1)]	adam	[0.3, 0.5]	[0.5, 0, 0]	[128, 64, 32]	[(2, 2), (2, 2)]	40	16	0.983333349227905	0.983333349227905	1.61050796508789
0.001	[8, 16]	[(3, 3), (3, 3)]	[(1, 1), (1, 1)]	adam	[0.3, 0.5]	[0.5, 0, 0]	[128, 64, 32]	[(2, 2), (2, 2)]	40	32	0.983333349227905	0.983333349227905	3.17218399047852
0.001	[8, 8]	[(5, 5), (5, 5)]	[(1, 1), (1, 1)]	adam	[0.3, 0.5]	[0.5, 0, 0]	[128, 64, 32]	[(2, 2), (2, 2)]	40	32	0.989583313465118	0.983333349227905	1.61270523071289
0.001	[8, 8]	[(5, 5), (5, 5)]	[(1, 1), (1, 1)]	adam	[0.3, 0.5]	[0.5, 0, 0]	[64, 64, 32]	[(2, 2), (2, 2)]	40	32	0.985416650772095	0.983333349227905	0.815586090087891
0.001	[16, 16]	[(3, 3), (3, 3)]	[(1, 1), (1, 1)]	adam	[0.3, 0.5]	[0.5, 0, 0]	[128, 64, 32]	[(2, 2), (2, 2)]	40	32	0.981249988079071	0.966666638851166	3.17755508422852
0.001	[16, 8]	[(5, 5), (5, 5)]	[(1, 1), (1, 1)]	adam	[0.3, 0.5]	[0.5, 0, 0]	[128, 64, 32]	[(2, 2), (2, 2)]	30	16	0.966666638851166	0.966666638851166	1.62125015258789
0.001	[16, 8]	[(3, 3), (3, 3)]	[(1, 1), (1, 1)]	adam	[0.3, 0.5]	[0.5, 0, 0]	[128, 64, 32]	[(2, 2), (2, 2)]	30	32	0.964583337306976	0.966666638851166	1.61050796508789
0.001	[8, 16]	[(5, 5), (5, 5)]	[(1, 1), (1, 1)]	adam	[0.3, 0.5]	[0.5, 0, 0]	[128, 64, 32]	[(2, 2), (2, 2)]	30	16	0.979166686534882	0.966666638851166	3.18146133422852
0.001	[8, 16]	[(5, 5), (5, 5)]	[(1, 1), (1, 1)]	adam	[0.3, 0.5]	[0.5, 0, 0]	[128, 64, 32]	[(2, 2), (2, 2)]	40	32	0.985416650772095	0.966666638851166	3.18146133422852
0.001	[8, 16]	[(3, 3), (3, 3)]	[(1, 1), (1, 1)]	adam	[0.3, 0.5]	[0.5, 0, 0]	[64, 64, 32]	[(2, 2), (2, 2)]	40	32	0.985416650772095	0.966666638851166	1.59381484985352
0.001	[8, 8]	[(5, 5), (5, 5)]	[(1, 1), (1, 1)]	adam	[0.3, 0.5]	[0.5, 0, 0]	[128, 64, 32]	[(2, 2), (2, 2)]	30	16	0.966666638851166	0.966666638851166	1.61270523071289
0.001	[8, 8]	[(5, 5), (5, 5)]	[(1, 1), (1, 1)]	adam	[0.3, 0.5]	[0.5, 0, 0]	[128, 64, 32]	[(2, 2), (2, 2)]	30	64	0.981249988079071	0.966666638851166	1.61270523071289
0.001	[8, 8]	[(5, 5), (5, 5)]	[(1, 1), (1, 1)]	adam	[0.3, 0.5]	[0.5, 0, 0]	[128, 64, 32]	[(2, 2), (2, 2)]	40	16	0.970833361148834	0.966666638851166	1.61270523071289
0.001	[16, 16]	[(3, 3), (3, 3)]	[(1, 1), (1, 1)]	adam	[0.3, 0.5]	[0.5, 0, 0]	[128, 64, 32]	[(2, 2), (2, 2)]	30	32	0.981249988079071	0.949999988079071	3.17755508422852

CNN V3: Best Model

Learning Rate	Filters	Kernel Size	Strides	optimizer	Conv Dropout Rate	FC Dropout Rate	Units	Pool Size	Epochs	Batch Size	Train Accuracy	Validation Accuracy	Model Size
0.001	[8, 16]	[(3, 3), (3, 3)]	[(1, 1), (1, 1)]	adam	[0.3, 0.5]	[0.5, 0, 0]	[128, 64, 32]	[(2, 2), (2, 2)]	30	32	0.9812499880790710	1	3.17218399047852





Conclusion

Summary: Hyperparameter Tuning

	CNN V1: Full Face	CNN V2: Full Face	CNN V2: Eye Detector
Filters	[128, 320, 480], [128, 256, 360]	[8,8], [8, 16], [16, 8], [16, 16]	[8,8], [8, 16], [16, 8], [16, 16]
Optimizers	Adam, Adagrad, SGD	Adam	Adam
Loss	Binary Crossentropy	Binary Crossentropy	Binary Crossentropy
Kernel Sizes	(3,3), (5,5)	(3,3), (5,5)	(3,3), (5,5)
Pool Sizes	(2,2)	(2,2)	(2,2)
Learning Rate	0.001, 0.01	0.001	0.001
Batch Size	16, 32, 64	16, 32, 64	16, 32, 64
Epochs	30, 40	30, 40	30, 40

Best Hyperparameters

Model Summary

	CNN v1: Full Face	CNN v2: Full Face	CNN v3: Eye
Training Accuracy	100%	98.75%	98.12%
Training Validation Accuracy	100%	100%	100%
Test Accuracy	**66%**	**77%**	**83%**
Model Size	101.27 MB	1.59 MB	3.17 MB

Conclusion

- **What did we learn**
  - Adding more convolutional layers did not always yield higher validation accuracy
  - Using Adam optimizers yielded better results compared to SGD and Adagrad
  - Models with a high training and validation accuracy does not necessarily translate to high test accuracy

Conclusion (cont): Future work

- **How could be model be further improved**
  - Experiment using pre-trained models (i.e. VGG16, VGG19)
  - Train model based on limitations
    - Low light conditions, night, real driver background
    - Eye model expanded to two eyes instead of one + facial features
    - Subjects with sunglasses and better photo capture settings
    - Account for blinking, false photo captures
    - Additional data sources, subjects presenting greater population (handicap)

Appendix

Citations

1. Nasri, I., Karrouchi, M., Snoussi, H., Kassmi, K., Messaoudi, A. (2022). Detection and Prediction of Driver Drowsiness for the Prevention of Road Accidents Using Deep Neural Networks Techniques. In: Bennani, S., Lakhrissi, Y., Khaissidi, G., Mansouri, A., Khamlichi, Y. (eds) WITS 2020. Lecture Notes in Electrical Engineering, vol 745. Springer, Singapore. https://doi.org/10.1007/978-981-33-6893-4_6
2. Ghoddoosian, R. Galib. M, Athitsos, V. (2019). A Realistic Dataset and Baseline Temporal Model for Early Drowsiiness Detection. https://openaccess.thecvf.com/content_CVPRW_2019/papers/AMFG/Ghoddoosian_A_Realistic_Dataset_and_Baseline_Temporal_Model_for_Early_Drowsiness_CVPRW_2019_paper.pdf
3. Williamson AM, Feyer AM. Moderate sleep deprivation produces impairments in cognitive and motor performance equivalent to legally prescribed levels of alcohol intoxication. Occup Environ Med. 2000 Oct;57(10):649-55. doi: 10.1136/oem.57.10.649. PMID: 10984335; PMCID: PMC1739867. https://pubmed.ncbi.nlm.nih.gov/10984335/
4. Karolinska Sleepiness Scale (KSS), https://www.med.upenn.edu/cbti/assets/user-content/documents/Karolinska%20Sleepiness%20Scale%20(KSS)%20Chapter.pdf
5. Viola, P., Jones. M., (2001). Rapid Object Detection using a Boosted Cascade of Simple Features. https://www.cs.cmu.edu/~efros/courses/LBMV07/Papers/viola-cvpr-01.pdf
6. *wp2186170*. (n.d.). Wallpapercave. https://wallpapercave.com/wp/wp2186170.jpg
7. *How to use stratified random sampling in 2023*. (n.d.). Qualtrics. https://www.qualtrics.com/experience-management/research/stratified-random-sampling/
8. https://kajabi-storefronts-production.kajabi-cdn.com/kajabi-storefronts-production/file-uploads/blogs/22606/images/5481e13-3da0-b8e5-f87f-a5ff1b6da72c_eyeSight_-_Driver_Monitoring_Driver_Asleep_1.jpeg

NeurIPS Checklist

- Do the **main claims** made in the abstract and introduction accurately reflect the paper's contributions and scope main claims?
  - yes
- Reviewed **ethics review guidelines**?:
  - yes
- Did you discuss any potential **negative societal impacts** of your work?
  - yes
- Did you describe the **limitations** of your work?
  - Yes, images not available in low lighting conditions, various other image types, rotations, other population, check for demographics. male/female, race, check possible balance
- Did you state the full set of **assumptions** of all theoretical results?
  - Yes, we assume that model works in night conditions, could be done with image process, augmentation, image provided to use comes we assumption is driver equivalent. However, no steer wheel, in car/vehicle conditions not present.Other road background conditions not present. Simulate conditions presently with students.
- Did you include complete **proofs** of all theoretical results?
  - Yes
- Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)?
  - Yes
- Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)?
  - Yes, our note notebook printed out the data splits, hyperparameters and how we chose the best model
- Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)?
  - Yes
- Did you include the amount of **compute** and the type of **resources** used (e.g., type of GPUs, internal cluster, or cloud provider)?
  - Yes
- If your work uses existing assets, did you cite the creators?
  - Yes

Source: NeurIPS Checklist

NeurIPS Checklist (cont.)

- Did you mention the license of the assets?
  - Yes
- Did you include any new assets either in the supplemental material or as a URL?
  - Yes
- Did you discuss whether and how consent was obtained from people whose data you're using/curating?
  - Yes
- Did you discuss whether the data you are using/curating contains **personally identifiable information** or **offensive content**?
  - Yes, this is not a problem since we sourced our data from Kaggle which is public domain.
- Did you include the full text of instructions given to participants and screenshots, if applicable?
  - N/A
- Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable?
  - N/A
- Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation?
  - N/A

Source: NeurIPS Checklist

Contributions

	Kirti	Ray	Mayank
Modeling	Contributed to various aspects of the project, including EDA, data preprocessing, model training, hyperparameter tuning, and conducted a comprehensive review of the final code. Identified eye extractor as critical feature for model enhancement.	Contributed to various aspects of the project, including EDA, data preprocessing, model training, hyperparameter tuning, and conducted a comprehensive review of the final code. Explored using built-in pretrained models in Keras such as VGG16.	Identification of the project to be done on Kaggle, ideation and implementation of baseline V1, V2 and V3 architectures along with testing. Implementations in jupyter notebook. Discussions and presentation within team
Presentation	Contributed	Contributed	Contributed