Machine Learning Engineer Nanodegree

Capstone Project: Detecting drivers' attention and mood using CNN

Ameen Al Baz
September 10th, 2018

I. Definition

Project Overview

The National Highway Traffic Safety Administration (NHTSA) 2016 data shows that an alarming 37,461 people were killed in 34,436 motor vehicle crashes, an average of 102 per day [1]. Most road accidents occurred due to human error. Thankfully advanced driver-assistance systems such as traction control, adaptive cruise control, lane departure warning, anti-lock brakes, and electronic stability control, have been adopted to mitigate human error and improve vehicle safety.

Despite these advanced driver-assistance systems, no current technology is implemented in vehicles that respond to the state of the driver while they are operating the vehicle. For example, collision assist will attempt to apply brakes if the driver is predicting a collision with an object or vehicle ahead, regardless if the driver is alert or not. This can frustrate some drivers who do not want to hear the warning sound of the system and will turn off or change the settings of the system to make it less strict, which can increase the risk of collision. Advanced driver-assistance systems mostly act on exterior conditions and have little to no access to information about the state of the driver.

The goal of this project is to implement distraction detection (primary objective) and mood detection (secondary objective) using machine learning. Distraction detection is the primary objective because it will be used as input to advanced driver-assistance systems to improve the safety of the driver. The mood detection is a secondary objective because it is not as critical and will be used for adapting the interior of the vehicle to the driver's mood and sentiment data collection.

Ideally, the dataset should include a collection of images that contain the driver operating a vehicle in real world driving conditions: with their head facing different

directions, including straight ahead, while expressing different moods, both while wearing sunglasses or without, and under different lighting conditions. Unfortunately, no such dataset exists yet and it will difficult to create the dataset from scratch. Thus, the closest dataset available will be used: CMU Face Images Data Set [2].

Problem Statement

The state of the driver must be determined through a non-obstructive method and with an accuracy at least higher than random chance. The direction the driver is facing, as well as their emotional state must be determined regardless of whether the user is wearing sunglasses or not while driving. With this information, advanced driver-assistance systems will also include the driver's condition along with the road conditions while it is in operation to provide a seamless and enhanced driving experience.

The first step in this project is to examine the data and determine the data size and balance, and to augment the images. Images can be augmented with slight rotations and shifts can be applied to increase the number of images available as well as balance the dataset. Duplicate images that have a reduced resolution from the original image, as well as corrupted images, will not be used in the project.

Because the problem involves classifying images, a convolutional neural network will be used. CNNs do not require as much pre-processing compared to other image classification algorithms because the network can learn the necessary filters that are otherwise hand-engineered. The proposed solution is to use a pretrained convolutional neural network as a feature extractor and training a machine learning classifier to classify the extracted features. The pretrained convolutional networks that will be explored are ones that were trained on images of humans. This increases the chances that the features extracted from these networks are relevant to this project, which involves classifying the state of humans.

Once the features are extracted, a smaller neural network will be trained to classify the extracted features. A separate CNN will also be trained from scratch, referred to in the project as a pure CNN. Ultimately, the goal is to demonstrate and compare the accuracy of the webcam and CNN system to an existing market solution in its ability to detect driver distraction and mood, with or without sunglasses.

Metrics

The evaluation metrics that will be used to evaluate the benchmark and proposed solutions are outlined in Table 1 below. All solutions will be evaluated by their accuracy to correctly classify the distraction state of the driver (primary objective in green) as well

as the mood of the driver, with and without sunglasses. The solution must work with sunglasses as to not restrict drivers from wearing eyewear that can improve their visibility on the road. Therefore, performance with or without glasses is captured in the metrics.

Table 1. Evaluation metrics for the benchmark and proposed solution.

	Distr. Acc. w/ glasses	Distr. Acc. w/ clear	Mood Acc. w/ glasses	Mood Acc. w/ clear
Benchmark				
Pretrained CNN + NN				
CNN				

II. Analysis

Data Exploration

The CMU Face Images data consists of 640 black and white face images of people taken and categorized with varying pose (straight, left, right, up), expression (neutral, happy, sad, angry), eyes (wearing sunglasses or not), and image size. Figure 1 below shows samples from the dataset.

Figure 1: Samples from the CMU Face Images data set labelled according to pose, expression, eyes/sunglasses, and image size.

The dataset was provided in PGM format and was converted with a 3rd party tool to JPG format in order for python to be able to interpret the images. Bad and corrupted images that were found in the dataset and were removed.

For the different image sizes in the set, the original 128x120 size image is reduced to half the resolution for a medium size image and reduced again for a small size image. Only the largest copy of the images in the dataset will be used for the capstone because smaller image resolutions are not relevant to the problem. The ID label will be dropped from the image as well as facial recognition is not in the scope of the problem. The labels remaining that will be used for the capstone are head orientation, mood, and sunglasses on/off. These labels align with the problem statement which requires the detection of head orientation and driver mood regardless of whether they are wearing sunglasses or not.

Exploratory Visualization

All individuals in the dataset pose straight, left right, up down, with a neutral, happy, sad or angry expression, with or without sunglasses. This makes the dataset very balanced for those labels. However, the ratio of males to females in the dataset is not specified. One image from every individual in the dataset is presented below to determine the male to female ratio. There are twenty individuals in the dataset.

Figure 2: Sample from every individual from the dataset to determine the gender ratio of the dataset (straight, neutral, open).

Evidently, even though the data seems balanced at first when it comes to labels, the data is not balanced for gender. This may cause any of the models trained in this project to be more accurate for males than females, because there are more male data to train on. Specifically, a pure CNN might learn feature extraction layers specific to males, that may not apply to females, while the neural network classifier built on top of the pretrained CNN for ImageNet might perform better since its feature layers were trained on a more balanced sample of humans.

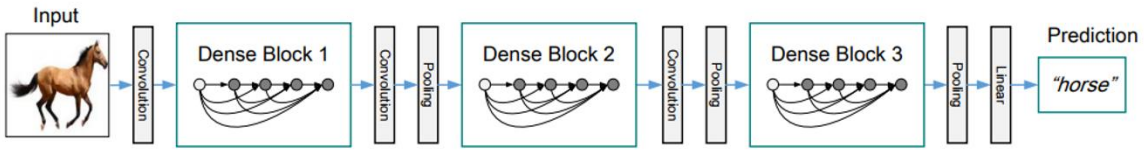
Algorithms and Techniques

As was demonstrated in the Udacity Machine Learning Engineer course, CNNs are the superior tools to use for image recognition. However, there are many different methods to implement CNNs. Therefore, two common CNN implementations will be used. The first CNN implementation involves using a pretrained CNN and removing the top layer responsible for classification. Without the classifier, the pretrained CNN becomes a feature extractor. Then, a simple neural network will be used to classify the extracted features. The CNN candidate chosen as a feature extractor is DenseNet121 for its relatively small size and high accuracy. MobileNet is a smaller in size but performs poorly compared to DenseNet121, while InceptionResNetV2 is the most accurate but large in size. Table 2 below shows the size and accuracy of the available models in Keras' Applications library.

Table 2. Size and performance of CNNs available in Keras [3].

Model	Size	Top-1 Accuracy	Top-5 Accuracy	Parameters	Depth
Xception	88 MB	0.790	0.945	22,910,480	126
VGG16	528 MB	0.715	0.901	138,357,544	23
VGG19	549 MB	0.727	0.910	143,667,240	26
ResNet50	99 MB	0.759	0.929	25,636,712	168
InceptionV3	92 MB	0.788	0.944	23,851,784	159
InceptionResNetV2	215 MB	0.804	0.953	55,873,736	572
MobileNet	17 MB	0.665	0.871	4,253,864	88
DenseNet121	33 MB	0.745	0.918	8,062,504	121
DenseNet169	57 MB	0.759	0.928	14,307,880	169
DenseNet201	80 MB	0.770	0.933	20,242,984	201

Figure 3: DenseNet-121 general structure. The last layer responsible for classification will be removed to make the CNN a feature extractor.



The model was imported into Jupyter Notebook with the command below, setting include_top=False to remove the classifier and loading the weights ='imagenet' to extract ImageNet features.

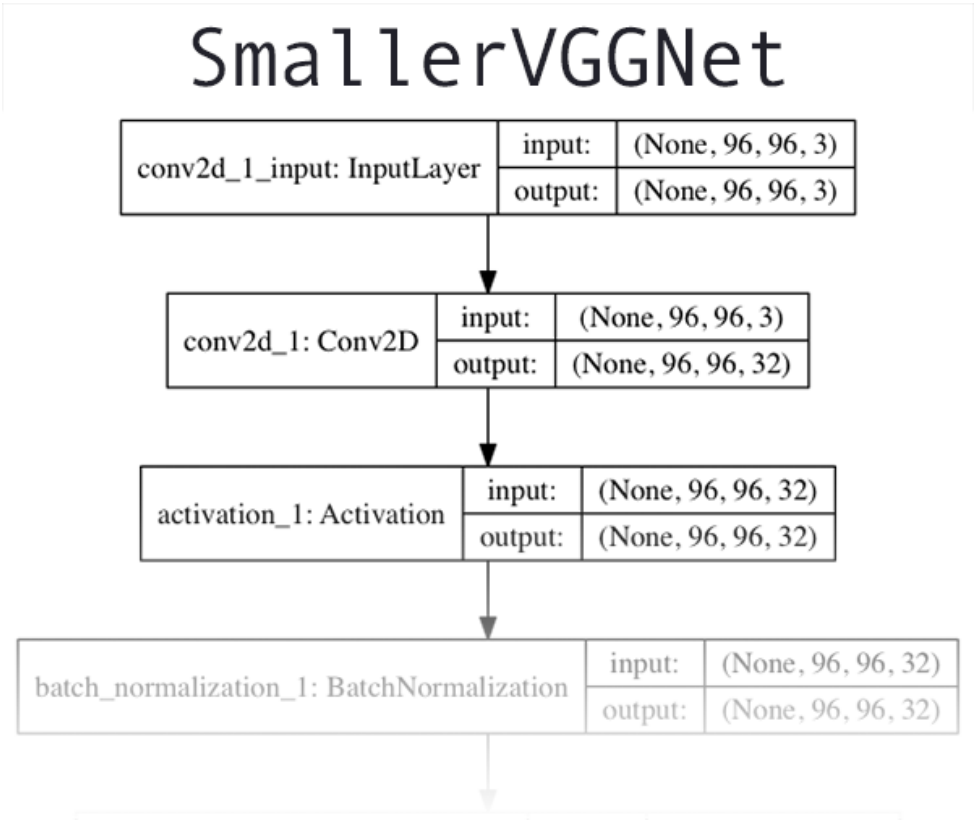
model = applications.DenseNet121(include_top=False, weights='imagenet')

ImageNet contains 1000 classes, including humans, and so it is assumed that the dataset explored in this capstone can be considered a subset of ImageNet. Therefore, the features extracted in DenseNet121 for ImageNet are relevant to this problem. Utilizing a pretrained network reduces the amount of time to train the model because the feature extractors are already pretrained. In addition, dataset can be processed through the feature extractors and the extracted features can be stored in a .npy file, so that feature extraction only need to occur once while the classifier NN is being tuned, saving even more time and computing resources.

The second CNN implementation involves training a pure CNN. The CNN architecture that will be used is SmallerVGGNet [4]. This smaller VGGNet variation was selected for this project to reduce training time while providing a similar performance to the VGGNet CNN. Figure 4 below explains the SmallerVGGNet architecture. A pure CNN allows the model to learn feature extraction specifically for the conditions presented in the dataset. This might achieve better performance than the ImageNet feature extractors since the feature extractors learned in the pure CNN will be more relevant to the problem, instead of achieving a more generalized feature extractor for ImageNet.

The 'rmsprop' optimizer with default parameters will be used to fit the two models to start and will be modified during refinement step. The input images will be resized to 224 x 224 as this is the input size for DenseNet121, and the SmallerVGGNet input layer will be modified to accept 224 x 224 images as well. Finally, data augmentation will be used in the refinement step to improve the models' performance (if necessary) as well as generate test images for sensitivity analysis.

Figure 4: SmallerVGGNet, a smaller variation of VGGNet, and will be trained on the dataset in this project.



Benchmark

Tobii Tech's eye tracking system is one of the newly available systems that can detect driver distraction and collaborate with the advanced driver assistance systems in the vehicles to respond accordingly [5]. However, the system does not have mood detection, and more importantly not reliable if the user is using eyewear [6]. No data was provided by Tobii Tech regarding accuracy. Because the product is advertised as fully functional, but does not work with sunglasses, it will have an assumed accuracy of 100% without sunglasses. In addition to the benchmark, the two proposed solutions will be compared to each other to determine which solution is more effective.

Table 3. Hypothetical score for Tobii Tech Eye Tracking from assumptions based on advertised information on their website.

	Distr. Acc. w/ glasses	Distr. Acc. w/ clear	Mood Acc. w/ glasses	Mood Acc. w/ clear
Tobii Tech Eye Tracking	0%	100%	0%	0%
Pretrained CNN + NN				
CNN				

III. Methodology

Data Preprocessing

Extracting the dataset shows that it is categorized into folders by the ID of the person in the photo. The categories that are of interest in the project are direction and the mood of the person. Opening one of the ID folders shows the images in .pgm format, which is not compatible with most python image manipulation libraries. The .bad format was also found, and signifies a corrupted image. Finally, the labels relevant to the project were included in the file names and separated by underscores. To summarize, preprocessing of the images includes:

1. The images must be extracted from the ID folders into one pool.
2. The corrupted images must be removed from the dataset.
3. The images must be converted to a compatible filetype, like jpg.
4. The images must be labelled.

To perform task one and two, Windows Search was used. Typing *.pgm reveals all the .pgm images in the dataset, regardless of what folder it is in, as well as excludes .bad filetype images. After the search is complete, the results were selected and copied to another folder. Then, Pixillion Image Converter was used to convert the .pgm images to .jpg images. To extract the necessary labels from the image, the filename labelling structure needs to be understood. For example: “an2i_left_angry_open.jpg” shows that the first label is the ID, the second label is the direction that person is facing, the third label is the mood, and the fourth label is whether he is wearing sunglasses or not. The

".split('_')" command was used on the extracted filenames in python to separate the labels into a list.

Once the data was preprocessed, it was resized to the same size as the input layer for DenseNet121 (224x224), the CNN that will be used as a feature extractor. The images were also scaled by 1/255 to keep the magnitudes in the image array between and including 0 and 1 to avoid issues during training and speed up the learning process faster. The images were passed through the DenseNet121 CNN with the top layer removed, and the resulting extracted features were saved to the computer hard drive using "np.save()" so that this step does not have to be repeated. To train on the extracted features, "np.load()" can be used to load the extracted features. To process the labels for training and validation, the LabelBinarizer algorithm was used from sklearn's preprocessing library to one-hot encode the labels to make them compatible with CNNs.

Implementation

To simplify the problem, the classification tasks were divided into two steps, and each step would implement the two solutions. This means that four models were trained in this project:

1. Classify the direction (primary objective)
    a. CNN+NN (1)
    b. pure CNN (2)
2. Classify the emotion (secondary objective)
    a. CNN+NN (3)
    b. pure CNN (4)

For the CNN+NN solution, the CNN used to extract features was 'DenseNet121' with the ImageNet weights. The classifier trained on these features is a basic NN, with an architecture summarized in Figure 4. This architecture was trained twice, once to classify the direction, and another for mood/emotion.

For the pure CNN solution, a 'smallerVGGNet' was implemented to learn both feature extraction and classification of the dataset. The architecture for this CNN is summarized in Figure 5. This architecture was also trained twice, once to classify the direction, and another for mood/emotion:

For both solutions, the activation functions used after the layers are ‘relu’, and the classification / output layer activation functions use ‘softmax’ to obtain results between and including 0 and 1. The ‘rmsprop’ optimizer with default parameters will be used to fit the two models to start and will be modified during refinement step. The loss function will be set to ‘binary_crossentropy’ and the metrics to ‘accuracy’. The models will be trained for 50 epochs with a batch size of 12, but only the weights that yield the highest validation accuracy will be stored. This is achieved with ‘ModelCheckpoint’ with the save_best_only parameter set to True.

Figure 4: Simple NN classifier that flattens the array of extracted features and outputs a label array of size 4.

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
flatten_52 (Flatten)         (None, 50176)             0
_________________________________________________________________
dense_103 (Dense)            (None, 256)               12845312
_________________________________________________________________
dropout_150 (Dropout)        (None, 256)               0
_________________________________________________________________
dense_104 (Dense)            (None, 4)                 1028
=================================================================
Total params: 12,846,340
Trainable params: 12,846,340
Non-trainable params: 0
_________________________________________________________________
```

Figure 5: ‘smallerVGGNet’ to be explored as a solution candidate against the first solution to classify the primary and secondary objective.

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_178 (Conv2D)          (None, 224, 224, 32)      896
_________________________________________________________________
activation_207 (Activation)  (None, 224, 224, 32)      0
_________________________________________________________________
batch_normalization_207 (Bat (None, 224, 224, 32)      128
_________________________________________________________________
max_pooling2d_106 (MaxPoolin (None, 74, 74, 32)        0
_________________________________________________________________
dropout_155 (Dropout)        (None, 74, 74, 32)        0
_________________________________________________________________
conv2d_179 (Conv2D)          (None, 74, 74, 64)        18496
_________________________________________________________________
activation_208 (Activation)  (None, 74, 74, 64)        0
_________________________________________________________________
batch_normalization_208 (Bat (None, 74, 74, 64)        256
_________________________________________________________________
conv2d_180 (Conv2D)          (None, 74, 74, 64)        36928
_________________________________________________________________
activation_209 (Activation)  (None, 74, 74, 64)        0
_________________________________________________________________
batch_normalization_209 (Bat (None, 74, 74, 64)        256
_________________________________________________________________
max_pooling2d_107 (MaxPoolin (None, 37, 37, 64)        0
_________________________________________________________________
dropout_156 (Dropout)        (None, 37, 37, 64)        0
_________________________________________________________________
conv2d_181 (Conv2D)          (None, 37, 37, 128)       73856
_________________________________________________________________
activation_210 (Activation)  (None, 37, 37, 128)       0
_________________________________________________________________
batch_normalization_210 (Bat (None, 37, 37, 128)       512
_________________________________________________________________
conv2d_182 (Conv2D)          (None, 37, 37, 128)       147584
_________________________________________________________________
activation_211 (Activation)  (None, 37, 37, 128)       0
_________________________________________________________________
batch_normalization_211 (Bat (None, 37, 37, 128)       512
_________________________________________________________________
max_pooling2d_108 (MaxPoolin (None, 18, 18, 128)       0
_________________________________________________________________
dropout_157 (Dropout)        (None, 18, 18, 128)       0
_________________________________________________________________
flatten_54 (Flatten)         (None, 41472)             0
_________________________________________________________________
dense_107 (Dense)            (None, 1024)              42468352
_________________________________________________________________
activation_212 (Activation)  (None, 1024)              0
_________________________________________________________________
batch_normalization_212 (Bat (None, 1024)              4096
_________________________________________________________________
dropout_158 (Dropout)        (None, 1024)              0
_________________________________________________________________
dense_108 (Dense)            (None, 4)                 4100
=================================================================
Total params: 42,755,972
Trainable params: 42,753,092
Non-trainable params: 2,880
```

Refinement

Training and validating the NN classifier on the extracted features yields the results for direction detection in Figure 6. The results show that the neural network was able to classify the images with 0.8259 training accuracy and 0.8365 validation accuracy. The graphs show that over 50 epochs, the accuracy and the model loss converged. However, there are oscillations in the graphs which could indicate a high learning rate.

To refine the solution, learning rates and optimizers were manually tuned. Through trial and error, the optimizer will be switched to 'sgd' with a learning rate of '0.0001'. Applying these parameters and retraining the model yields the results in Figure 7. The results show an improvement; the neural network was able to classify images with 0.9909 training accuracy and 0.9856 validation accuracy. In addition, the curves converge overtime without oscillation.

After the results of the CNN+NN solution were improved, the model for the pure CNN solution was trained with the 'sgd' optimizer with a learning rate of '0.0001'. The results are displayed in Figure 8. The results show that the CNN accuracy improved steadily over time and converged to their final values of 0.8574 training accuracy and 0.9038 validation accuracy. The results show good performance, but it does not outperform the pretrained CNN + NN classifier solution.

To classify emotions in the images, the models were retrained with the y-targets set to the emotion labels. The models did not train well with the 'sgd' optimizer with a learning rate of '0.0001', and so different optimizers and parameters were tested. Finally, 'adam' optimizer with default settings seemed to provide the best results for both the pretrained CNN + NN classifier and pure CNN when classifying emotions.

The two models were retrained again for emotion detection, and the 'adam' optimizer with default parameters show the best training and validation performance. The results are shown in Figure 9 and Figure 10. Both models seem to hit their accuracy limit at around 70% accuracy before the classifiers start overfitting.

Figure 6: Results from training and validating the NN classifier with ‘rmsprop’ and default parameters to classify direction detection.

```
5s 10ms/step - loss: 2.7346 - acc: 0.8259 - val_loss: 2.5753 - val_acc: 0.8365
```

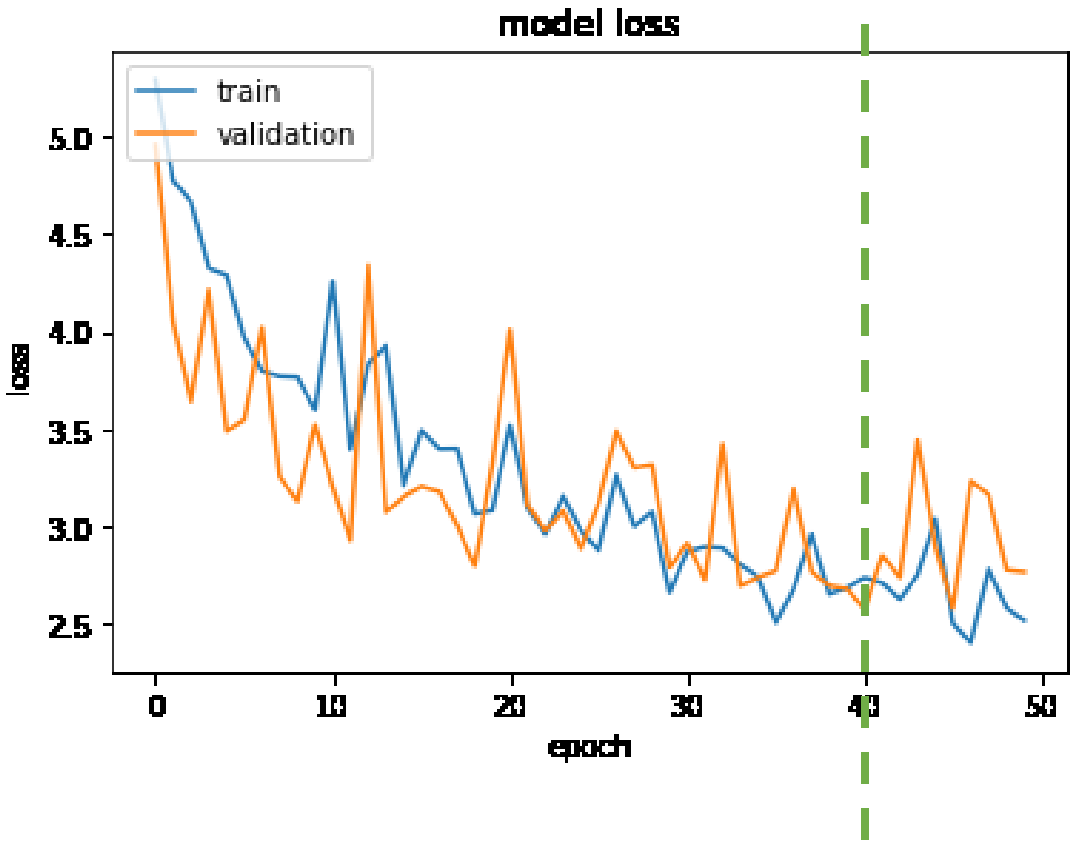


Figure 7: Results from training and validating the NN classifier with ‘SGD’ and ‘0.0001’ learning rate to classify direction detection.

```
3s 6ms/step - loss: 0.0379 - acc: 0.9909 - val_loss: 0.0641 - val_acc: 0.9856
```

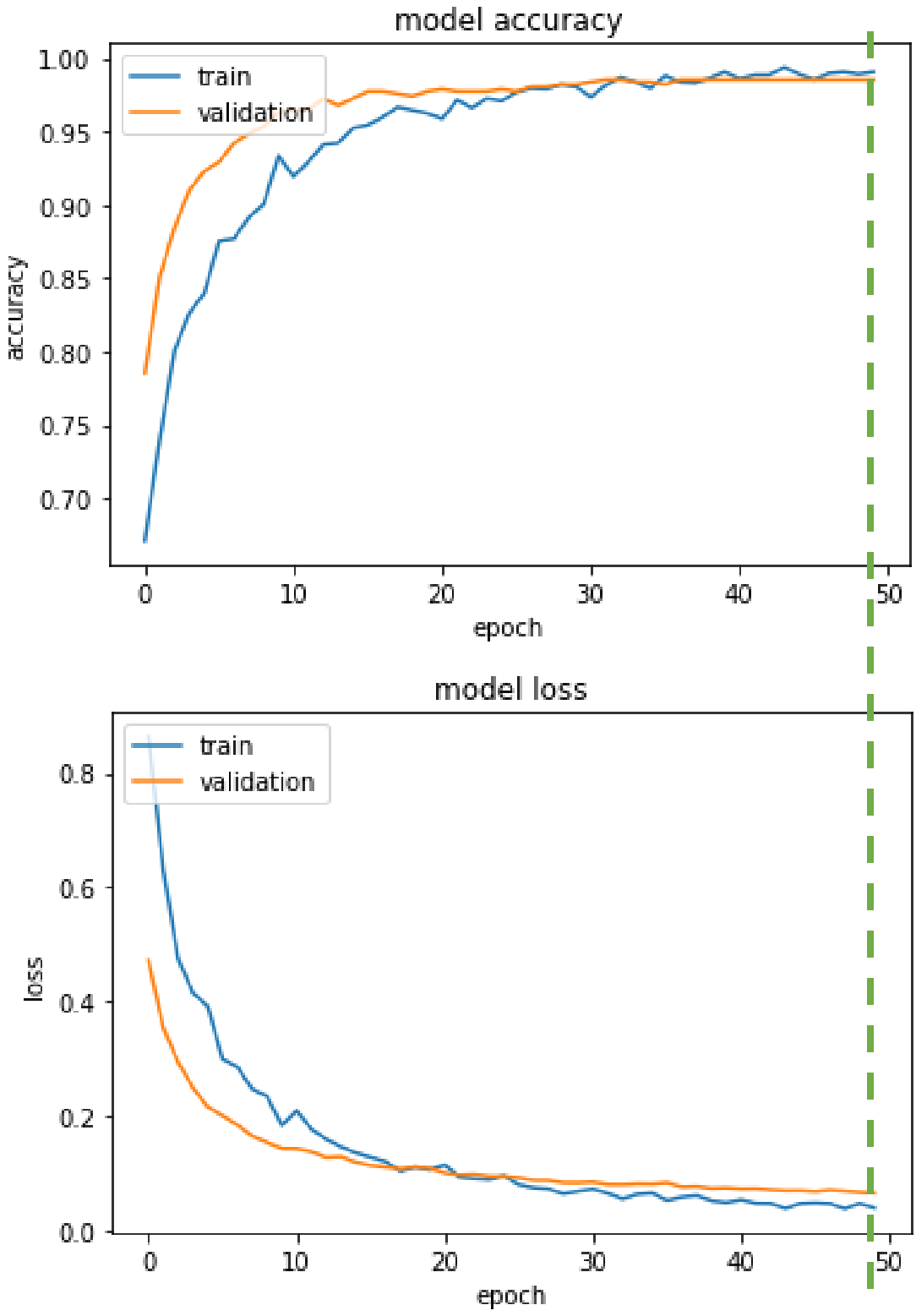


Figure 8: Results from training and validating the pure CNN classifier with ‘SGD’ and ‘0.0001’ learning rate to classify direction detection.

```
124s 265ms/step - loss: 0.3591 - acc: 0.8574 - val_loss: 0.2476 - val_acc: 0.9038
```

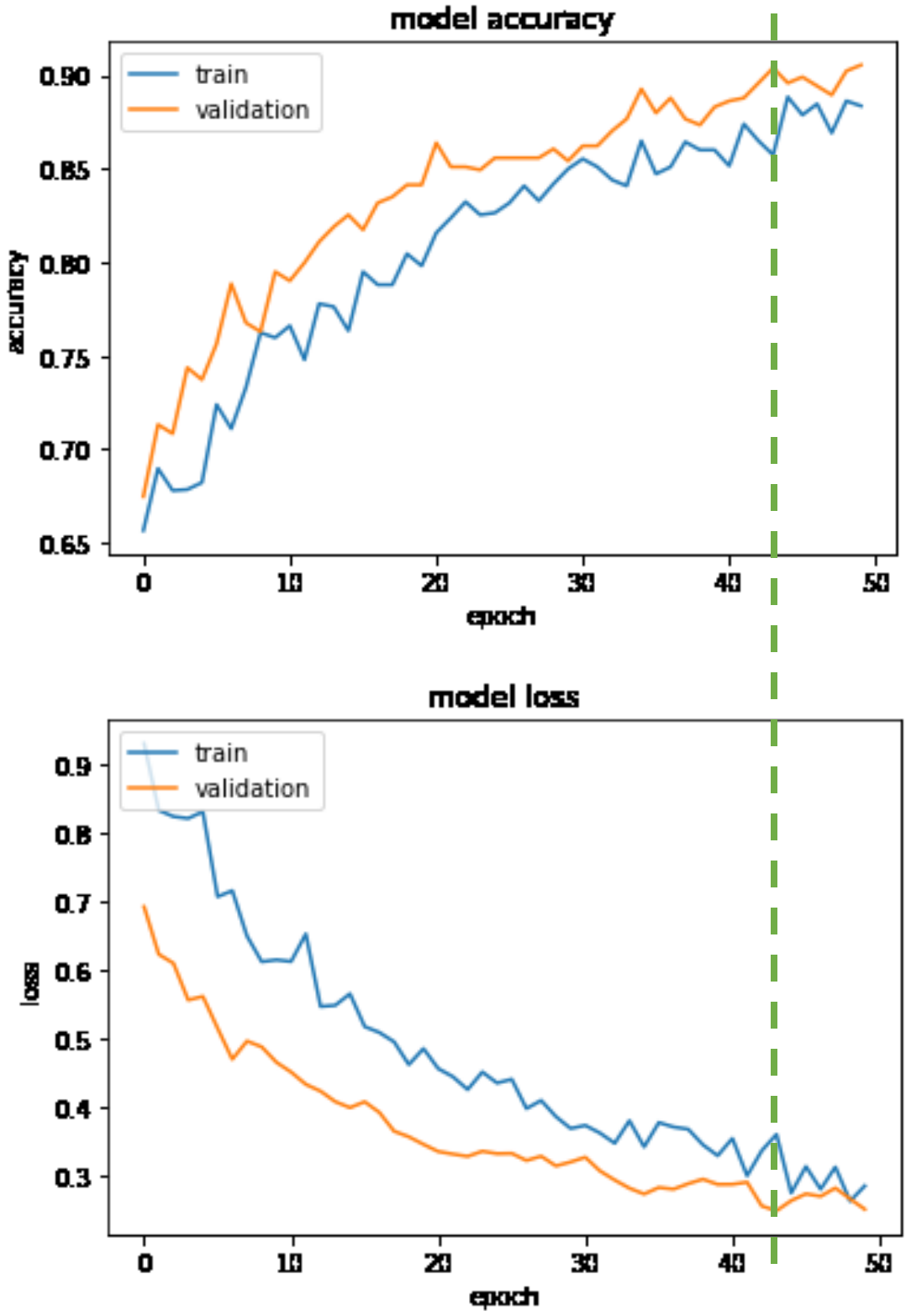


Figure 9: Results from training and validating the NN classifier with ‘adam’ optimizer and default settings.

```
3s 7ms/step - loss: 0.6550 - acc: 0.7046 - val_loss: 0.6712 - val_acc: 0.6955
```

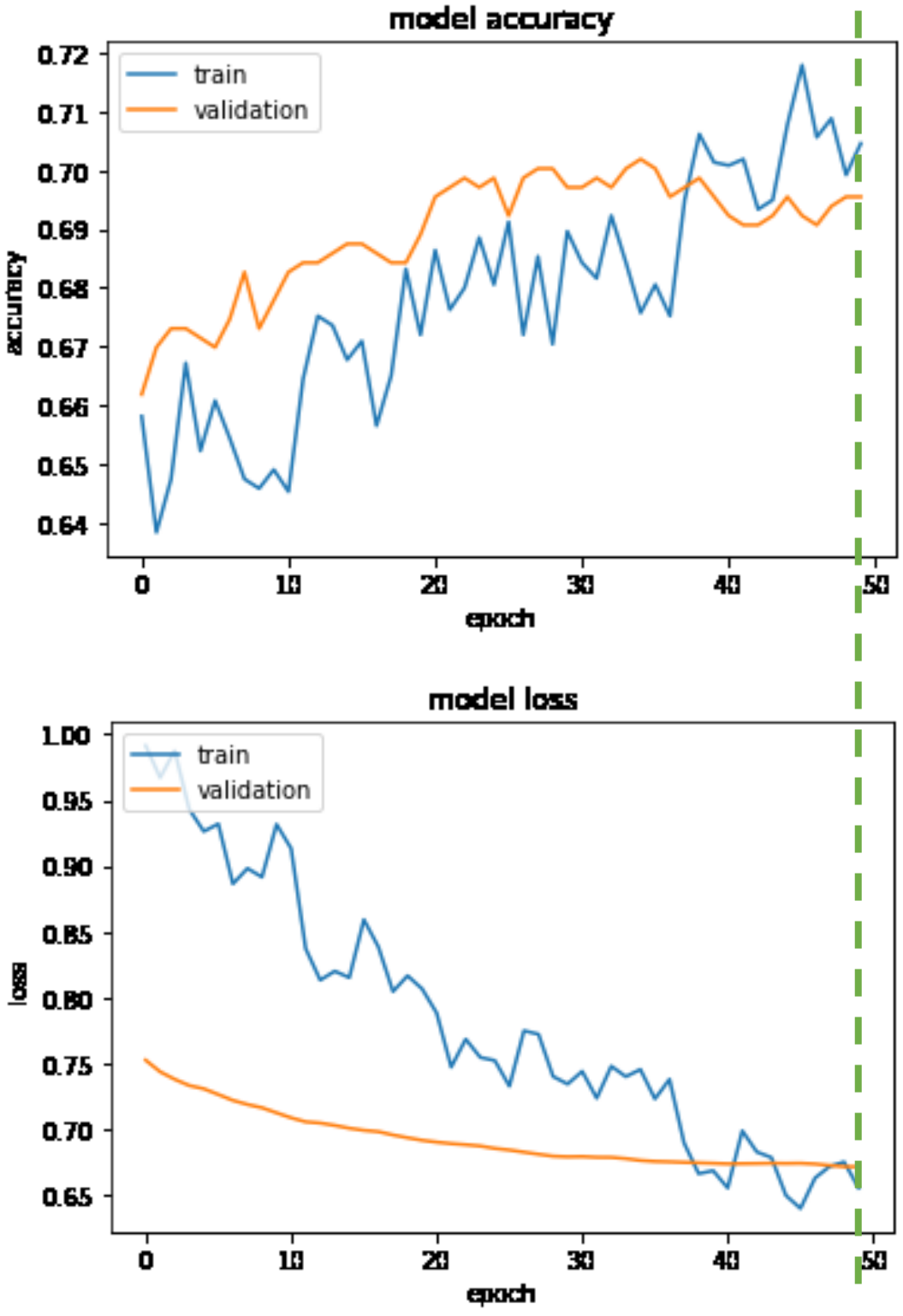
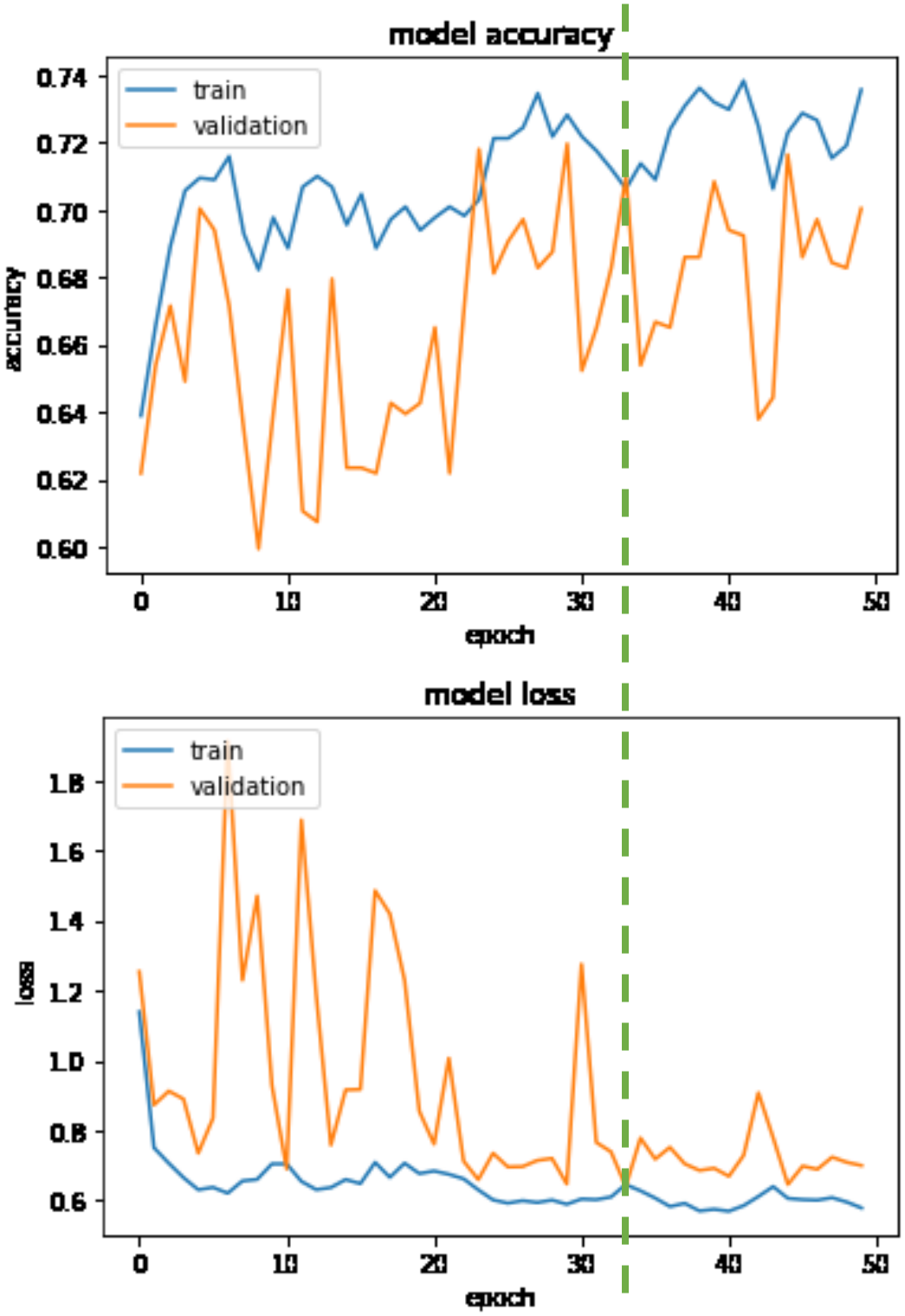


Figure 10: Results from training and validating the pure CNN classifier with ‘adam’ optimizer and default settings.

```
128s 273ms/step - loss: 0.6454 - acc: 0.7062 - val_loss: 0.6400 - val_acc: 0.7099
```



IV. Results

Model Evaluation and Validation

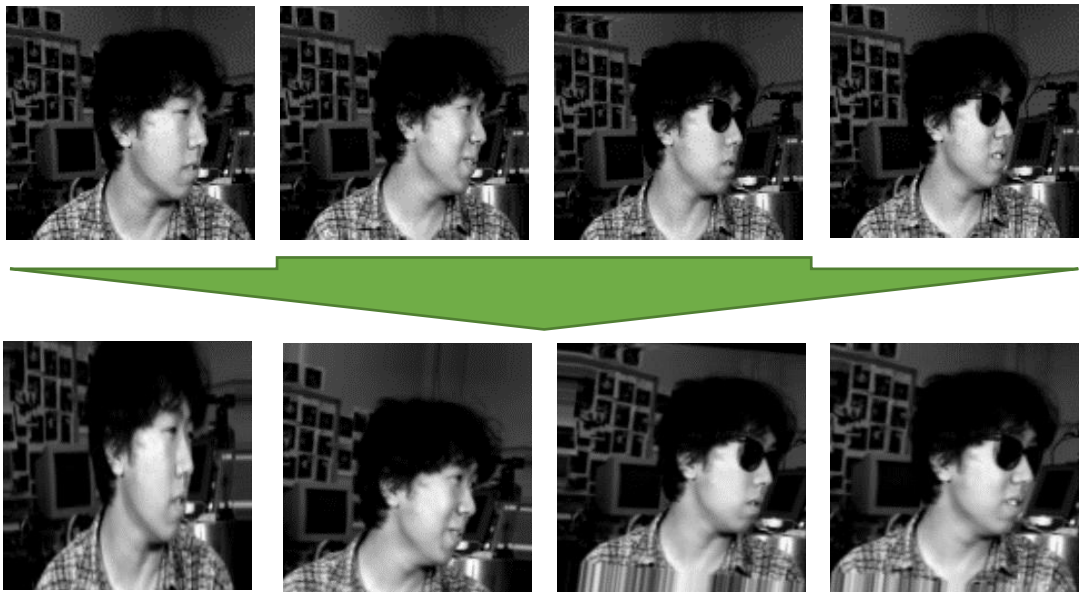
The final models were reasonable and align with the solution expectations. The models' parameters were chosen through trial and error based on two criteria:

1. Higher validation score
2. Smooth converging curves to the score

The CNN+NN model is robust to perturbations and generalizes well to unseen data. This was ensured because the model was trained with an Image Data Generator. This function augments the images and was set to generate and augment changes with rotations, width shifts, height shifts, shears, and zooms. The parameters selected are demonstrated in the function below:

datagen = ImageDataGenerator(rescale=1. / 255, rotation_range=10, width_shift_range=0.1, height_shift_range=0.1, shear_range=0.2, zoom_range=0.2, fill_mode="nearest")

Figure 11: Image augmentation sample. The first row is the original sample, and the second row is the augmented sample.



Examples of the image augmentations that were outputted from this function are presented in Figure 11. Image flipping was not used as part of image augmentation because that would interfere with location classification. For example, flipping an image labelled as 'left' would cause the individual in the image to turn right, which means that 'left' would be the incorrect label.

The results can be trusted if the background environment remains the same. However, checking the performance of the model on actual driver images does not yield consistent results. In Figure 12, the image of the straight-facing driver is classified as right because it scores a probability of 0.39 for right and 0.30 as straight. For the second picture, it was able to classify the distracted driver with a classification probability of 0.5. This is because the model was not trained on various driver backgrounds and vehicle configurations.

Figure 12: Testing the model on real pictures of drivers. It does not perform reliably due to new features extracted that were not encountered in training. More real-world driving data needs to be used for training for more reliable results.

Justification

The models' final solutions and the results were compared to the benchmark established earlier in the project in the table below:

Table 4. Score table that portrays the best robust scores for the pretrained CNN+ NN solution, pure CNN solution, and the current market solution: Tobii Tech Eye tracking.

	Distr. Acc. w/ glasses	Distr. Acc. w/ clear	Mood Acc. w/ glasses	Mood Acc. w/ clear
Tobii Tech Eye Tracking	0%	100%	0%	0%
Pretrained CNN + NN	**98.96%**	**98.96%**	69.55%	69.55%
Pure CNN	90.38%	90.38%	70.99%	70.99%

Evidently, the results show that the machine learning models are superior to the benchmark solution. They can perform regardless of whether the driver is wearing sunglasses or not, as well as provide some indication of the driver's emotions. Meanwhile, the benchmark can only provide distraction information when the user is not wearing sunglasses, and does not provide any information on the mood of the driver.

Specifically, the pretrained CNN + NN classifier performs best in classifying driver distraction with 98.96% accuracy, regardless of whether the driver is wearing sunglasses or not. The model seems to perform on par with the pure CNN solution when it comes to emotion classification. Although the model performs better for detecting distracted driver compared to detecting driver emotion, it is an acceptable solution because detecting distracted driver is the primary objective (safety) while detecting driver mood is the secondary objective (amenities and data collection).

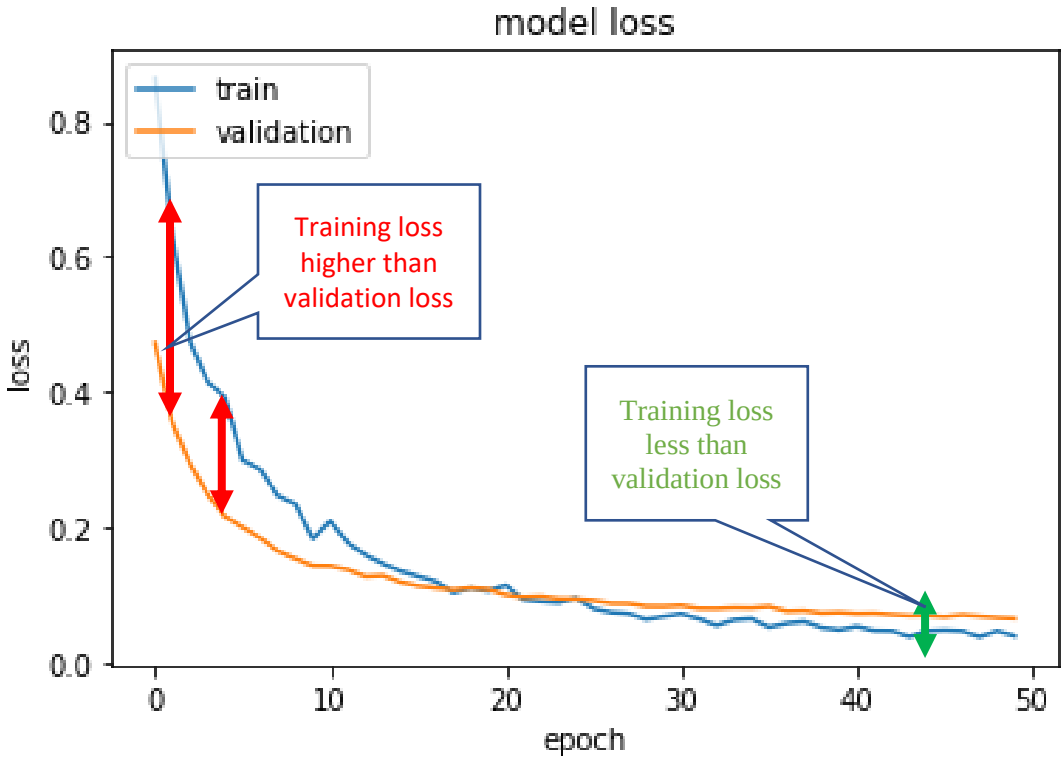
The pretrained CNN + NN will be chosen as the final solution for this project. The final solution may not necessarily be significant enough to have solved the project. The project demonstrates that machine learning will improve on the current market solution and allow even more seamless driver distraction detection, regardless if the driver is wearing sunglasses or not. However, it is recommended that a specialized CNN specific to this problem is trained on actual driver images to accurately provide driver distraction detection while being smaller and less resource intensive to be able to run in a vehicle.

V. Conclusion

Free-Form Visualization

In general, correctly trained models should show a higher validation loss than training loss. However, many of the results obtained in Keras shows the training loss to be higher than the validation loss. This caused a lot of confusion and was investigated. Figure 13 below demonstrates a case where this is apparent.

Figure 13: Results from training and validating the NN classifier with 'SGD' and '0.0001' learning rate to classify direction detection. The training loss is much higher than the validation loss.



Investigating the Keras website [7], the following information was found:

> **Why is the training loss much higher than the testing loss?**
>
> A Keras model has two modes: training and testing. Regularization mechanisms, such as Dropout and L1/L2 weight regularization, are turned off at testing time.
>
> Besides, the training loss is the average of the losses over each batch of training data. Because your model is changing over time, the loss over the first batches of an epoch is generally higher than over the last batches. On the other hand, the testing loss for an epoch is computed using the model as it is at the end of the epoch, resulting in a lower loss.

In addition to the second point mentioned on the Keras regarding the model changing over time, both models implemented in the project make use of dropout regularization and this contributes to the training loss being higher than the validation loss and this will be evident when graphing the Keras model history. This effect is seen throughout the data obtained during training in this project.

Reflection

The project followed the process below from start to finish:

1. Determine a problem that needs to be solved, or can have the current solution be improved. Where can machine learning be used to improve the solution or part of the solution?
2. Research the problem and background, and identify a benchmark.
3. Obtain data or similar data that can be used for demonstration to make a compelling case for obtaining the true data.
4. Clean and study the data.
5. Determine the appropriate algorithms for the data and the problem.
6. Test the models and tune the hyperparameters on a smaller subset of the data.
7. Test the model and hyperparameters on the full dataset
8. Compare models to benchmark.
9. Reflect on results.

The most interesting aspect of the project was realizing just how feasible it was to demonstrate how applying machine learning to replace or compliment an existing solution can dramatically increase performance and robustness of the solution. It is possible to demonstrate this solution improvement to most problems, including healthcare, maintenance, etc.

The most difficult aspect of the project was testing different hyper parameters and models for large data. To mitigate this, the hyperparameters were tested on a smaller subset on the data until an acceptable performance was achieved. Then those parameters were used in training the full set of data.

The final model and solution fit the expectation for the problems, and it should be used in a general setting to solve these types of problems. It is less resource intensive to use a pretrained network to extract features and use a simple neural net to classify the features. This should be the first implementation to determine potential and feasibility before a dedicated specialized machine learning solution can be developed.

Improvement

There is lots of room for improvement in this project. Using cloud machine learning solutions can also improve training time but was not used in this project. Data augmentation could have been implemented more extensively to generate more data and improve the results for mood detection. XGBoost was researched as a classifier to replace the NN classifier, but was not implemented because it was difficult to apply in multiclass solutions. A simplification of the classes (distracted vs not distracted) could have simplified applying XGBoost to classify the extracted features. Obtaining more data, especially real-world driver images data, will dramatically improve reliability of the model. The data should include more female images, and different types and shapes of eyewear.

Setting the final solution as the new benchmark, an even better solution can definitely exist. Although there is not much room for improvement regarding distraction classification, there is room for improvement regarding mood classification. However, since this is a secondary objective, resources were primarily allocated to solve the primary objective. If a business case was made to demonstrate the importance of emotion detection of drivers, more resources can be invested into emotion classification to increase the performance from its current ~70% accuracy.

References

[1] https://crashstats.nhtsa.dot.gov/Api/Public/ViewPublication/812451.

[2] https://archive.ics.uci.edu/ml/datasets/cmu+face+images

[3] https://keras.io/applications/

[4] https://www.pyimagesearch.com/2018/04/16/keras-and-convolutional-neural-networks-cnns/

[5] https://www.tobii.com/tech/products/automotive/

[6] https://help.tobii.com/hc/en-us/articles/210249865-Glasses-lenses-and-eye-surgery

[7] https://keras.io/getting-started/faq/#why-is-the-training-loss-much-higher-than-the-testing-loss