Distracted Driver Behavior Classification

May 16, 2021

The AI Consulting Group

Elmer Atienza

elmeratienza2020@u.northwestern.edu

Yining Feng

e3k1k5@u.northwestern.edu

Narmada Gomatam

narmadagomatam2020@u.northwestern.edu

Abstract

Distracted Driver Behavior Classification (DDBC) is a computer vision model that distinguishes distracted driving behaviors from safe driving posture based on thousands of images of individuals driving collected from an in-car camera. The goal of the DDBC model is to present a solution as an alternative to sensor-based intelligence for distracted driving detection using different Convolutional Neural Network (CNN) algorithms: fully connected CNN, VGG-16, ResNet, Xception, MobileNet, and an ensemble model to output predicted class labels and the corresponding probabilities of driver postures to help facilitate faster claims processing, and support diminishing deductible incentives offered by auto insurance companies for promoting safe driving behaviors. The development process of DDBC involved factorial-type experiments on different configurations of native, pre-trained, and Automated Machine Learning (AutoML) models to identify the configuration with the most balanced distraction classification performance. The DDBC effort also identified different deployment options for making the DDBC machine learning model available for use both on cloud-based and edge-based applications.

1. Introduction

Distracted driving is any activity that diverts attention from driving, including talking or texting on your phone, eating and drinking, talking to people in the vehicle, fiddling with the stereo, entertainment or navigation system — anything that takes one's attention away from the task of safe driving. Texting is the most alarming distraction. Sending or reading a text takes a driver's eyes off the road for 5 seconds. At 55 mph, that is like driving the length of an entire football field with your eyes closed. Using a cell phone while driving creates enormous potential for deaths and injuries on U.S. roads. In 2019, 3,142 people were killed in motor vehicle crashes involving distracted drivers. (NHTSA,2019)

The DDBC model intends to provide inferences for distracted driving behaviors using thousands of images of individuals driving inside a vehicle. The model discussed in this paper could be beneficial for the following business applications in the auto industry for promoting safe driving behaviors:

a. Driver Behavior Detection for facilitating faster claims processing

Claims processing is one of the automation use cases in the insurance industry that is already seeing the great benefit by applying computer vision. Driver behavior images can help assess liability of a claim and can help determine whether distracted driving behaviors such as driving while texting, drinking, talking to passengers, are major causes of car accidents. Fast and efficient claims processing is paramount to success for insurance companies. The latest advances in computer vision algorithms using deep learning are achieving interesting results in the classification of images, object detection, and image segmentation. Although such applications are still emerging, an increasing number of auto insurance companies are starting to look at this technology as a way to make insurance claims processes easier and more efficient as one of the insurtech industry's biggest challenges.

b. Computer vision solution that can be used as an alternative to sensor-based solutions

Detectors for distracting activities in the market predominantly rely on data from multiple sensors in the vehicle to generate inferences for unusual driver activity. Data from sensors that

are installed on steering wheel, brake and gas pedals, driver's seat, speedometer, etc. are collectively used to detect unusual movements on these controls to provide inferences for unsafe driving patterns (e.g., unusual or unstable steering wheel movement, sudden change in speed, etc.). As the sensor-based method requires dedicated sensors for each index that needs monitoring (i.e., hand grip, foot contact, heartbeat, etc.), such a method could be very complex to coordinate and not too portable to implement.

The Distracted Driver Behavior Classification model presents a computer vision solution that can be an alternative to sensor-based intelligence for detecting unsafe driving behaviors. Using ensembles of deep neural networks and traditional machine learning models to learn from images of different drivers' positions and activities, a computer vision classifier model can provide similar types of inferences as the ones provided by sensor-based models for unsafe driving patterns.

c. Incentive to reduce insurance rate based on existing driving records

At the urging of safety advocates aiming to address distracted driving, various forms of incentives and campaigns have been initiated to spread awareness about distracted driving and promote safe driving behaviors. For instance, many auto insurance companies are offering an incentive called diminishing deductible, which is an additional coverage that rewards people for being a safe driver. As one continues to avoid car accidents and maintain a clean driving record over time, the amount of one's deductible will continue to reduce each year at policy renewal.

Meanwhile, computer vision technologies have been explored and developed to address distracted driving postures in addition to incentives and campaigns for preventing distracted driving. TrueMotion presented a smartphone app that monitors driving behavior using sensors in the smartphone. The app then gives an overall safety score based on the habits behind the wheel. Later, TrueMotion developed an app called "Mojo" that provides feedback and incentives to help users reduce distracted driving (Wise, 2018). The app provides an overall score that characterizes how distracted a user is while driving. Similarly, Floow Score is an app developed by The Floow Limited, which uses the phone sensors and telematic data, to score speed, smoothness of driving, distraction and time for the driver; this app is mainly used to aid insurance companies with pricing their policies and predicting risk for each driver more accurately (Brown, 2020).

Likewise, the Distracted Driver Behavior Classification model in this study can be used to support the diminishing deductible initiative by detecting various distracted behaviors. The model can be incorporated into a smartphone app to help raise drivers' awareness of their driving habits and associated risks, thus helping to reduce careless driving and promoting safe driving practices to reduce the accident rate.

2. Literature Review

Computer Vision is used as an underlying intelligence behind some of the state-of-the-art features in the automotive industry today. Tesla for instance uses its Autopilot cameras to feed its computer vision neural network to direct operation of its automatic wiper system (Lambert 2019). Advantech developed a modular in-vehicle platform that processes video data at edge in real time that uses computer vision as a second pair of eyes in detecting risks for accidents such as collisions, blind spot detection, and driver behavior recognition (Moss, 2020). Other manufacturers such as GoFleet (Abou-diab, 2019), SmartWitness (Singh, 2018), and Zenduit (Zenduit, 2021) use computer vision in portable devices for detecting driver distraction behavior.

Image classification for detection of distracted driver behavior can be implemented using traditional machine learning models or deep learning models. As for the classifying model, one main approach is to use convolutional neural work (CNN)-based models. Liu et al. proposed a convolutional two-stream network with multi-facial feature fusion to detect distraction activities by combining static and dynamic features (Liu et al., 2019). Eraqi et al. proposed a model based on an ensemble of convolutional neural networks and shows that a weighted ensemble of classifiers using a genetic algorithm yields a better accuracy (Eraqi et al., 2019). It is revealed that CNN-based models generally report high accuracy. Another major approach is to use Support Vector Machine (SVM). Osman et al. shows that both linear SVM classifier and nonlinear SVM classifier can reach reasonably high accuracy (Osman et al., 2019). Compared to CNN-based models, SVM models will not get the best accuracy, but the learning process is faster, and computational cost is lower than CNN. The third main approach is to use semi-supervised learning. Since the datasets obtained in this problem are likely to include many unlabelled images, using semi-supervised learning augments available data for training and can potentially improve prediction accuracy. Liu et al. proposed a semi-supervised model and revealed that with the additional unlabeled data, the semi-supervised learning methods improved the detection performance compared to the traditional supervised methods (Liu et al., 2015).

Besides these 3 main approaches, deep learning has gained more attention recently in distraction detection. Colbran et al. adopted a VGG-16 model that achieves an accuracy of 80%. The AlexNet models were also used in the study (Colbran et al., 2016). Based on the State Farm dataset, Lörincz et al. utilised region-based CNNs to detect the driver's head pose, hands and classify the object the driver is holding and thus detected distraction behaviours (Lörincz et al., 2016). In addition, deep learning techniques can also estimate the driver's head pose which is valuable information for distraction detection. Venturelli et al. proposed an approach which uses a simple CNN to estimate head pose such as yaw, pitch, and roll angle through a depth camera (Venturelli et al., 2016).

AutoML models have been considered and proposed as the image classifier engine for driver distraction behavior. Chen Chai et al. performed a study that proposed a domain-specific automated machine learning (AutoML) based on XGBoost called AutoGBM to self-learn the optimal models to detect distraction based on lane-keeping performance data. The model is tested based on driving simulator experiments of three driving distractions caused by phone usage: browsing short messages, browsing long messages, and answering a phone call (Chai et al. 2021). AutoML model for detecting distracted drivers was created using Microsoft Azure Custom Vision platform and trained with the same State Farm distracted driver dataset that DDBC was trained on (Martinez 2019).

Distracted driver classifier models are deployed on cloud and/or edge-based architecture to achieve full inferencing integration with camera-enabled IoTs. Such deployment was presented in the project of Abdu Gumaei et al., where a CNN-based driver distraction model was deployed on separate modules, including the distraction detection module deployed on edge devices in the vehicle environment, the training module deployed in the cloud environment (Gumaei et al. 2020).

3. Data

The whole dataset contains 22,424 training images categorized in 10 classes as well as 79,727 testing images from StateFarm's distracted driver detection dataset published on Kaggle

(9 classes for different distraction activities and 1 class for safe driving), and Figure 1 shows illustrating images for each category. The size of each image is 640 × 480 pixels. Each image in StateFarm dataset is associated with one of the following driver's postures: safe driving (C0), texting using the right hand (C1), talking on the phone using the right hand (C2), texting using the left hand (C3), talking on the phone using the left hand (C4), operating the radio (C5), drinking (C6), reaching behind (C7), hair and makeup (C8), and talking to passenger (C9).

Figure 1: *10 classes of Driver Postures from the StateFarm's Distracted Driver Dataset. C0 - C9 represent the label of each image in the dataset*.

However, the StateFarm's dataset contains images that are almost identical to each other as they are extracted from a video captured by a camera mounted in the car. In addition, it is noteworthy that the training data has multiple images of the same person within a class with slight changes of angle and shifts in height or width. Accordingly, the challenging part of this project is that it is very easy to overfit training models due to the limited number of drivers in the StateFarm's dataset. Because of high similarities between two images of the same driver, it is more likely for the training model to over-emphasize on the drivers' features instead of their driving behaviors. One solution for preventing overfitting is to synthetically get more images from the training set through Image Augmentation, which is a technique that creates more images from the original data through alteration such as shifting width and/or height, rotation, and zoom. There are 21 drivers from 10 classes included in the training dataset and 5 other drivers from 10 classes included in the validation set. Since there are approximately 2,200 images on average in each class as illustrated in Figure 2, there is minimal likelihood of the imbalance problem with the training dataset. In this study, the dataset was divided into 70% training and 15% held out test data and 15% for the validation purpose.

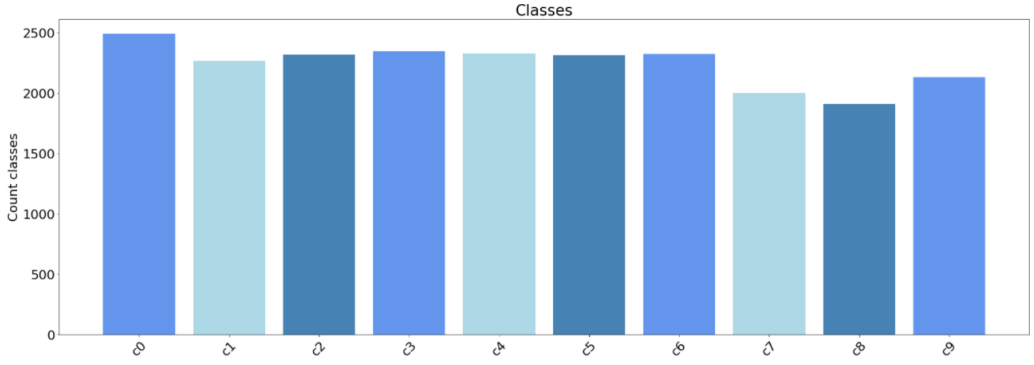


Figure 2: *Distribution of 10 classes of driver's postures among StateFarm's labeled training images*.

Each image needs to be preprocessed before going to the classifier, and the float chart of image preprocessing is shown in Figure 3. Each image is first converted into a high-dimensional 640 × 480 × 3 matrix based on its RGB values on each pixel, then resized to a 64 × 64 × 3 matrix using CV2 in order to improve the computing efficiency of the classifier, and finally flattened into a vector. For each flattened vector, a binary numerical label is assigned using one-hot encoding based on the category it belongs to. For example, an image labeled as C6, which refers to the drinking category was encoded with a feature vector like [0,0,0,0,0,0,1,0,0,0].

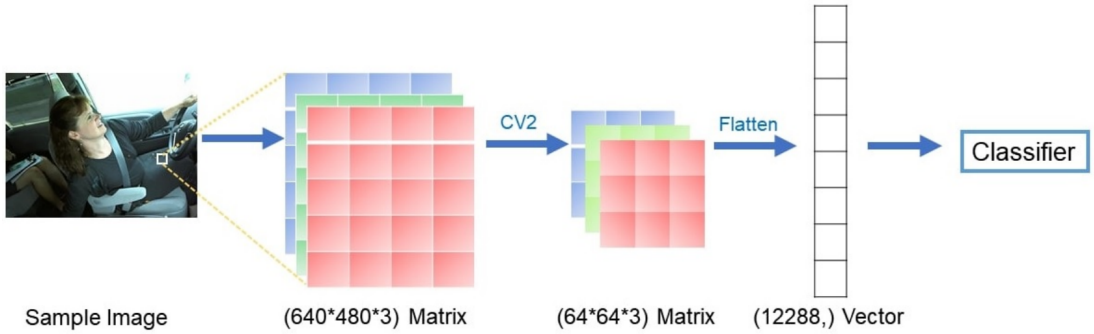


Figure 3: *Float chart of image preprocessing.*

4. Methodology

Different configurations of pre-trained Convolutional Neural Networks (CNN) were considered and evaluated for the proposed DDBC classification model. Transfer learning was used on top convolution layers of several transfer learning models to re-train specifically for classifying distracted behaviors using driver images from the State Farm distracted driving dataset. CNN models that are generated from different AutoML platforms were also considered, evaluated, and compared against the other pre-trained models that were considered for DDBC.

A factorial approach was used in the evaluation for the appropriate ML model to use for DDBC. Each CNN-based model was trained on different factors to evaluate training times and predictive performance of each model. Number of epochs and dense layers, hyperparameters for performance improvement (i.e., learning rate, batch normalization, dropout, etc.), optimizing and classifying algorithms were used as factors to generate performance-related data that were used for the evaluation. Predictive performance was evaluated from different points of views using several performance metrics that are appropriate for single-label multiclass classification: classification accuracy, log loss, precision, recall, F1-score, and confusion matrix. Unsupervised learning through visualization of intermediate activation of convolutional layers was also performed to get a better understanding on what the CNN models have learned in classifying for different distracted driving postures.

Working prototypes of applications that use DDBC models were also created for different operating platforms (e.g., IOS, Android, Windows, etc.), to present the viability of DDBC for practical use in action.

Convolutional Neural Network

A Convolutional Neural Network (ConvNet/CNN) is a Deep Learning algorithm which can take in an input image, assign importance (learnable weights and biases) to various aspects/objects in the image and be able to differentiate one from the other. The pre-processing required in a ConvNet is much lower as compared to other classification algorithms. While in

primitive methods filters are hand-engineered, with enough training, ConvNets have the ability to learn these filters/characteristics.

The architecture(Figure 4) of a ConvNet is analogous to that of the connectivity pattern of Neurons in the Human Brain and was inspired by the organization of the Visual Cortex. Individual neurons respond to stimuli only in a restricted region of the visual field known as the Receptive Field. A collection of such fields overlap to cover the entire visual area. The CNN model is devised as a 2-layer neural network in this study. There are 100 neurons in the hidden layer, and ReLU is used as the activation function for this layer. For the output layer, a softmax activation function is used for final classification. In this study, activations of all convolutional and fully connected layers are normalized, since batch normalization helps to improve the performance and stability of neural networks by explicitly forcing the activations through a layer of network to follow a unit Gaussian distribution. It reduces strong dependence on weight initialization, improves gradient flow through the network as well as allows higher learning rates.

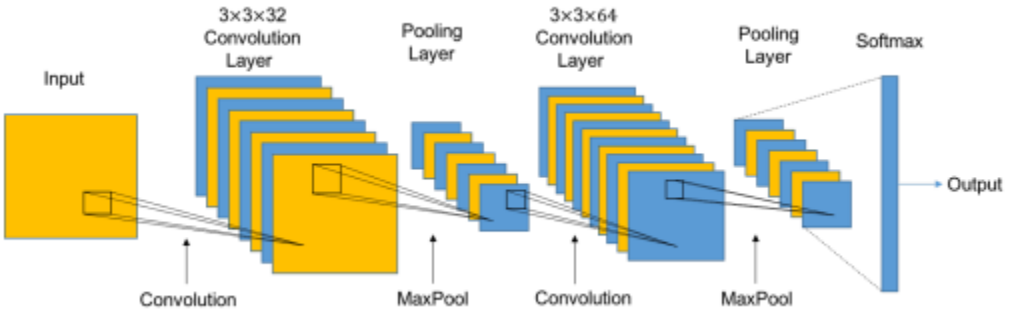


Figure 4: *CNN Architecture*

Transfer Learning

Training the CNN models for distracted driving detection is not a trivial task. Transfer learning is a commonly used technique in training deep neural networks. The networks which were pre-trained with millions of images can be reused for training purposes. In this study, for instance, the last FC layers were replaced with a multi-layer perceptron (MLP) classifier. It is a type of ordinary neural network, which includes several FC layers where each neuron is connected to all the outputs from the previous layer. The last FC layer computes the probabilities for each class. For multi-class classification, softmax regression is a popular choice. The MLP classifier consists of activation layers with 1,024 and 2,048 neurons as well as a ten-neuron FC layer which corresponds to the ten driving behaviours. Transfer learning helps save a significant amount of time because it saves the effort of training the whole model from scratch. However, one of the problems that transfer learning usually encounters is that it makes the model prone to overfitting when the nature of the learned data is rather different than the training data. In this case, it is necessary to fine-tune the model by unfreezing some pre-trained layers and then train them along with the FC layers. Fine-tuning may increase the training time but it could solve the over-fitting problem. The following hyper parameters were adjusted for each model: learning rate (α), weight decay (γ) and momentum (μ). In addition, applying dropout layers is an efficient way of reducing overfitting by randomly dropping out i.e ignoring some neurons in the training phase. It helps to reduce interdependent learning amongst the neurons. Therefore, linearly increasing dropout layers were added to a few convolutional as well as fully connected layers.

4.1 Models

VGG-16 Model

VGG-16 is a 16-layer CNN developed by Simonyan et al. for image recognition in the 2014 ImageNet large scale visual recognition challenge (ILSVRC). 3 × 3 filters are used for all convolutional layers (Simonyan & Zisserman, 2014). The network accepts the input image with a dimension of 224 × 224. The image is passed through a sequence of 16 convolutional layers. A multilayer perceptron (MLP) classifier including three fully-connected (FC) layers is used in addition to the convolutional layers to perform the classification. Rectified linear unit (ReLU) layers and max-pooling layers are used in the whole network to prevent the overfitting problem.

ResNet Model

He et al. from Microsoft Research proposed a new training framework called ResNet with 34 layers for very deep CNN training. The network is adopted from the VGG network (He et al., 2016). The input image size for these networks is 224 × 224. The convolutional layers have mostly 3 × 3 filters and the design follows two rules: (i) for the same output feature map size, the layers have the same number of filters. (ii) If the feature map size is halved, the number of filters is doubled in order to preserve the time complexity per layer. The down-sampling operation is performed by the convolutional layers that have a stride of two, hence no pooling layers. The network ends with a global average pooling layer and a 1000-class FC layer with the softmax function. In this network, a shortcut connection is added to each building block (two or three consecutive convolutional layers). Owing to this new training framework, they are able to train very deep networks, with 18-, 34-, 50-, 101-, and 152-weight layers, without encountering the degradation problem.

Xception Model

The Xception architecture has 36 convolutional layers forming the feature extraction base of the network. The 36 convolutional layers are structured into 14 modules, all of which have linear residual connections around them, except for the first and last modules. Specifically, the Xception architecture in this study contains 4 residual depth-wise separable convolutions where each convolution is followed by a batch normalization operation and a ReLUs activation function. The last layer applies a global average pooling and a soft-max activation function to produce a prediction. This architecture has approximately 60,000 parameters. In essence, the Xception architecture is a linear stack of depth-wise separable convolution layers with residual connections. This makes the architecture very easy to define and modify; it takes only 30 to 40 lines of code using a high level library such as Keras or TensorFlow-Slim, similar to an architecture such as VGG-16.

MobileNet Model

The MobileNet structure is built on depth wise separable convolutions except for the first layer which is a full convolution. All layers are followed by a batchnorm and ReLU nonlinearity with the exception of the final fully connected layer which has no nonlinearity and feeds into a softmax layer for classification. Figure 5 contrasts a layer with regular convolutions, batchnorm and ReLU nonlinearity to the factorized layer with depthwise convolution, 1 × 1 pointwise convolution as well as batchnorm and ReLU after each convolutional layer. Down sampling is

handled with strided convolution in the depthwise convolutions as well as in the first layer. A final average pooling reduces the spatial resolution to 1 before the fully connected layer. Counting depthwise and pointwise convolutions as separate layers, MobileNet has 28 layers.

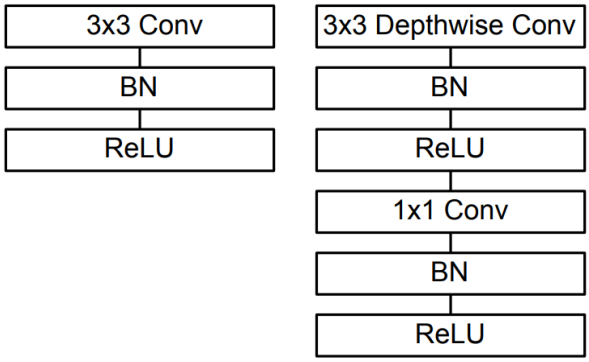


Figure 5: *Left: Standard convolutional layer with batchnorm and ReLU. Right: Depthwise Separable convolutions with Depthwise and Pointwise layers followed by batchnorm and ReLU.*

K-Nearest Neighbors

Based on the test set, it is evident that images are highly correlated because they are taken from the same in-car camera. Even though images from the same driver differ subtly, our model might still predict them in different categories. Therefore, the K-Nearest Neighbors Algorithm (K-NN) was introduced to get the most k relevant neighbors in terms of Euclidean pixel difference. In this way, the average probability can be calculated among the neighbors and overwrite their probabilities. This approach serves as a noise reduction and greatly reduces the effect of the outliers.

Ensemble Model

An ensemble model is the idea of combining (averaging) the results of several classifiers. In this study, an ensemble model was created by combining the results of CNN, VGG-16, ResNet, Xception, and MobileNet models. Given that the test set is about four times as large as the training set, overfitting can become a potential problem. By averaging different models, the variance of the trained model can be reduced and a lower loss can be achieved to prevent overfitting. In addition, K-NN was included into the ensemble model to help smoothen predicted probabilities for each class. To find the 10 nearest neighbors, outputs from the second last layer of the VGG-16 transfer learning model were used as features on the validation set.

AutoML

Deep Neural Network classifier models were created using three different AutoML platforms: from fully configurable Amazon Web Services (AWS) SageMaker to less configurable Apple CreateML for Image Classification and fully automated Google Cloud Platform (GCP) AutoML Vision development environments. Using AWS SageMaker, transfer learning was performed on different configurations of pre-trained CNN-based models. As the models provided by both CreateML and GCP AutoML cannot be configured by a user, the optimized model and hyperparameters chosen by these AutoML platforms for image classification tasks were used for the DDBC model. In CreateML, the DDBC model was trained using different data augmentation techniques (e.g., different levels of image cropping, exposures,

and blurs) that are available in the platform and on a different number of epochs. In GCP AutoML and except for confidence threshold, the model was trained using the hyperparameters that are appropriate for 10 node hours (or in processing metrics used in GCP), which is a little more than the minimum training node hours recommended by GCP based on the size of the training data for the classification task.

AutoML goes beyond creating machine learning architecture models. It can automate many aspects of machine learning workflow, which include data preprocessing, feature engineering, model selection, architecture search and model deployment. AutoML deployments can also be categorized by the format of training data used. Some AutoML solutions can handle multiple data types and algorithms.

Leveraging AutoML solutions offers multiple benefits that go beyond traditional machine learning or automation. The first is speed. AutoML allows data scientists to build a machine learning model with a high degree of automation more quickly and conduct hyperparameter search over different types of algorithms, which can otherwise be time-consuming and repetitive. By automating key processes — from raw data set capture to eventual analysis and learning — it can reduce the amount of time required to create functional models.Another benefit is scalability. While machine learning models can't compete with the in-depth nature of human cognition, evolving technology makes it possible to create effective analogs of specific human learning processes. Introducing automation, meanwhile, helps apply this process at scale — in turn, enabling data scientists, engineers and DevOps teams to focus on business problems instead of iterative tasks. A third major benefit is simplicity. AutoML is a tool that assists in automating the process of applying machine learning to real-world problems. By reducing the complexity that comes with building, testing and deploying entirely new ML frameworks, AutoML streamlines the processes required to solve line-of-business challenges.

4.2 Performance Evaluation

Predictive performance of CNN models that were considered for DDBC were evaluated from different point of views using the following performance metrics:

a. Accuracy
   - Driver images correctly classified against all training images

b. Log Loss
   - Determines how close the model's predictive probability to perfect predictive probability

c. Precision
   - True positive classifications against the total of true positives and false positives

d. Recall
   - True positive classifications against the total of true positives and false negatives

e. F1-score
   - Harmonic average of precision and recall

f. Confusion Matrix
   - Determines correctly and incorrectly classified images

Unsupervised learning was also performed using activation maps on intermediate convolutional layers to expose the features that the CNN models learned to use for classifying for different distracting behaviors.

4.3 DDBC Deployment Prototypes

Trained DDBC models from each AutoML platform were exported as deployable models for Edge deployment: ".DLRModel" for AWS SageMaker, ".mlmodel" for Apple CreateML, and ".tflite" for GCP AutoML. Mini projects were created to consume these ML models to simulate deployment on different edge devices. The ".DLRModel" was deployed to a deep learning-enabled AWS DeepLens camera so that this IoT device can use the DDBCS model to generate inferences for driver distraction behavior from video frames that it captures. An XCode project written in Swift was developed to consume the ".mlmodel", and deployed to an XCode Simulator to create a prototype that classifies still images in an iPhone for driver distracting behavior. A python project that uses the Tensorflow Lite framework was developed to consume the ".tflite" model and delegate the inferencing task to a Google edge device Coral to simulate a deployment of an application for detecting driver distraction behavior from images captured in a Linux-based and Android-based IoT device.

5. Results

5.1 Overall Result

As Table 1 shows, the K-NN Ensemble model outperforms all the other transfer learning models with the highest classification accuracy (92% validation accuracy) and the lowest output loss overall, but it also takes the longest time (797 seconds per epoch) to run the model. On the other hand, Mobilenet (without extra Dense Layers) is the best stand-alone transfer learning model that can be used to get pretty accurate (86% validation accuracy) and faster predictions with the least amount of resources (167 seconds per epoch). Although the ResNet model with extra layers has also achieved the same level of classification accuracy, it requires more time and resources to run (214 seconds per epoch) in comparison to the Mobilenet model. Therefore, to balance accuracy and speed, the MobileNet model is regarded as the best stand-alone transfer learning model for distraction driver behavior classification.

Model Type	Extra Dense Layers	Filters/ Units	Batch Normalization	Dropout (0.5 rate)	Classification Accuracy			Output Loss			Training Time (second /epoch)
					Train	Valid	Test	Train	Valid	Test	
Baseline CNN	No	32	Yes	No	0.91	0.63	0.62	0.34	1.32	1.35	224
VGG-16	No	64	Yes	Yes	0.78	0.85	0.82	0.62	0.50	0.56	181
VGG-16	Yes	64	Yes	Yes	0.86	0.82	0.79	0.42	0.57	0.69	178

ResNet	No	64	Yes	Yes	0.88	0.85	0.84	0.34	0.55	0.67	201
ResNet	Yes	64	Yes	Yes	0.89	0.86	0.85	0.32	0.47	0.56	214
Xception	No	64	Yes	Yes	0.80	0.83	0.82	0.61	0.55	0.58	241
Xception	Yes	64	Yes	Yes	0.86	0.84	0.82	0.39	0.52	0.57	236
MobileNet	No	64	Yes	Yes	0.91	0.86	0.85	0.28	0.39	0.40	167
MobileNet	Yes	64	Yes	Yes	0.89	0.83	0.80	0.32	0.63	0.65	171
K-NN Ensemble	No	64	Yes	Yes	0.94	0.92	0.89	0.18	0.24	0.30	797

Table 1: *Comparison of transfer learning models' performance and running speed*.

While each of the transfer learning models above has yielded decent results, there is a significant variance in the performance of each model for individual classes. From the Table 2 below, it is noteworthy that different models have the best accuracy for each class.

	VGG-16	MobileNet	Xception	ResNet	K-NN Ensemble
Safe Driving	54.4%	92.5%	78.9%	78.2%	95.3%
Texting-Right	96.2%	92.3%	93.2%	89.1%	96.6%
Talking-Right	95.1%	94.3%	99.4%	97.4%	96.5%
Texting-Left	98.5%	98.2%	98.2%	98.5%	96.2%
Talking-Left	87.9%	75.8%	76.5%	77.3%	94.8%
Operating the Radio	96.8%	99.0%	94.8%	97.5%	95.1%
Drinking	90.3%	86.2%	80.4%	79.3%	98.0%
Reaching Behind	81.1%	99.4%	99.1%	99.7%	95.4%
Hair and Makeup	63.6%	53.3%	40.1%	59.6%	92.8%

Talking to passenger	83.1%	68.5%	58.8%	79.5%	97.7%

Table 2: *Comparison of transfer learning models' prediction accuracy per class*.

Basic Convolutional Neural Network Model

Initially, a CNN model was built for solving this problem. The validation accuracy that was achieved for the basic CNN model was about 63.2% and a log loss of 1.62. It has been proved that CNNs are definitely better in performance than fully connected neural networks, as evidenced by Diagrams 1 & 2 in Appendix A.

It was worth noticing that among the 20 training epochs, the training accuracy increased to 90% but the validation accuracy was around 65%. It is a clear case of overfitting and it was noticed that the basic CNN was unable to predict the images of the 'talking to passengers' posture. Other classes were misclassified as 'talking to passengers'.

VGG-16 Model

For the VGG-16 model, there are some misclassifications between the class 'talking on the phone – left' and 'hair and makeup'. This is because these two activities have many common postures especially when the phone is occluded by the driver's face completely. On the other hand, 'safe driving' has often been misclassified with 'texting left' in some scenarios when the steering wheel blocked the left hand. However, the trained model has achieved the highest validation accuracy of 98.5% (for the 'texting left' class), as evidenced by Diagrams 1-3 in Appendix B.

ResNet Model

The maximum validation accuracy is 99.7% (for the 'reaching behind' class), as evidenced by Diagrams 1-3 in Appendix C. The ResNet model has the most misclassifications of 'texting right' with 'hair and makeup'. This is because the driver in some images put the phone too close to his/her face. This reason leads to the wrong prediction. In addition, 'talking to the phone – left' has also been misclassified as 'hair and makeup' because both behaviors have some similar postures.

Xception Model

The maximum validation accuracy of the Xception model is 99.1% (for the 'reaching behind' class), as evidenced by Diagrams 1-3 in Appendix D. The model is well-fitted. Similar to the VGG-16 and ResNet models, the Xception model has the most misclassifications between the classes of 'texting left' and 'safe driving' because of their similarity in postures especially when the phone is covered by the steering wheel. Other than that, the behavior of 'talking to the phone – left' has also been misclassified as other postures such as 'texting left' or 'hair and makeup'.

MobileNet Model

MobileNet's requirement for the least amount of training time overall suggests that the model converged faster than the other three transfer learning models. This means that the deeper the model is, the faster it fits the dataset. The maximum validation accuracy of the MobileNet model is 99.4% (for the 'reaching behind' class), as evidenced by Diagrams 1-3 in Appendix E. Similar to the VGG-16 model, the trained MobileNet model also has the most misclassifications of two postures such as between 'talking with the phone – left' and 'hair and makeup', and between 'texting left' and 'safe driving'.

K-NN Ensemble Model

The K-NN ensemble model was able to achieve a classification accuracy over 90% for each of the 10 classes, as evidenced by Table 2. Meanwhile, the mislabeling of either 'talking on the phone – left' and 'hair and makeup' or 'texting left' and 'safe driving' has been mitigated. Thus, the K-NN ensemble model outforms all the other stand-alone models in solving the misclassification of 'talking with the phone – left' and 'hair and makeup' as well as 'texting left' and 'safe driving' postures.

AutoML

Setup efforts and time are almost the same between GCP and CreateML, as both platforms were able to upload approximately 22,000 jpeg images in about 2 hours and about half hour to set up a fixed format dataset control file. However, AWS required an approval process for the computing environment required for training image classification models with the volume of data as large as the State Farm Insurance dataset. It took 3 full days for AWS to review and approve the request for processing environment requested for DDBC. Model training, inferencing, and prototyping activities using AWS were not completed for this DDBC development effort because of the delay in the approval of the request for a computing environment for these activities.

AWS turned out to be the most expensive AutoML platform as data storage and computing environment came at a cost for the DDBC effort, and additional cost could have been incurred if the training and deployment for DDBC proceeded using AWS. The computing resources included in the free subscription model was sufficient enough to support the data storage, ML model development, and training for DBBC. No subscription was required to use the resources and development tools to create and deploy the DDBC model using CreateML.

Performance results are presented in Appendix D for the single-label classification models that were created and trained using AutoML platforms GCP AutoML Vision (GCP) and Apple CreateML (CreateML). The configuration that generated the best precision and recall for GCP and CreateML took approximately 8 hours to execute. Training times are almost identical for the number of iterations performed for GCP and CreateML, but GCP turned out to be training on longer durations for sessions that involve 300 or more iterations. Without any image data augmentation, both AutoML models generated the highest precision and recall when converged at 200 iterations. Data image augmentation using image exposure and blur variations on the training images significantly increased training times, but further improved precision and recall for GCP and CreateML models.

At the neutral 0.5 confidence threshold, the AutoML models achieved nearly perfect training precision and recall scores in all distracted driving behavior classes with CreateML having the slight edge with only 3 out of the ten classes with 0.99 scores that is still nearly perfect. Both GCP and CreateML models also achieved very high classification/predictive training and validation accuracy, with rates at over 99% and 95% respectively under the optimal number of epochs/iterations.

5.2 Analysis and Interpretation

The most challenging posture to be classified is the 'safe driving' posture. This is due to the lack of temporal context in static images. In a static image, a driver would appear in a "safe driving" posture. However, contextually, he/she was distracted by doing some other activity. The 'texting left' posture was mostly confused with the 'talking to the phone – left' posture and vice versa. Same applied to 'texting left' and 'talking to the phone – right' postures. The 'operating the radio' posture was mainly confused with the 'safe driving' posture. That is due to lack of the previously mentioned temporal context. Apart from safe driving, the 'hair and makeup' posture has been mistaken as the 'talking to passengers' posture. That is because, in most cases, when drivers did their hair/makeup on the left side of their face, they needed to tilt their face slightly right (while looking at the frontal mirror). Thus, the network thought the person was talking to the passenger. Similarly, the 'reaching behind' posture has been mistaken as either 'talking to passengers' or 'drinking' posture. That makes sense as people tend to naturally look towards the camera while reaching behind. As for the drinking confusion, it is due to right-arm movement from the steering wheel to the back seat. A still image in the middle of that move could be easily mistaken for a drinking posture. The 'drinking' and 'talking to passengers' postures were not easily confused with other postures as 98% and 97.67% of their images were correctly classified as shown in Figure 6.

Actual \ Predicted	C0	C1	C2	C3	C4	C5	C6	C7	C8	C9
C0	95.34	0	0.33	0.65	0.11	0.43	0.43	0.87	0.11	1.74
C1	0.31	96.63	1.23	0.31	0.92	0	0.31	0	0.31	0
C2	0.29	3.23	96.48	0	0	0	0	0	0	0
C3	2.02	0.61	0	96.15	0.81	0	0.20	0	0	0.20
C4	0	0.33	0	4.90	94.77	0	0	0	0	0
C5	4.26	0	0	0.33	0	95.08	0	0	0	0.33
C6	0.74	0	0	0.25	0	0.74	98.01	0.25	0	0
C7	3.65	0	0	0	0	0	0	95.35	0	1.00
C8	3.79	0	0	0	0	0	1.38	0.34	92.76	1.72
C9	1.40	0	0	0	0	0	0.47	0.31	0.16	97.67

Figure 6: *Confusion matrix of the K-NN ensemble model created by combining the results of CNN, VGG-16, ResNet, Xception, and MobileNet models.*

The CNN model's validation accuracy was the lowest. Transfer learning models are a better choice for classifying and identifying the subtle 2D features of the images. In terms of transfer learning model performance, the Xception model has obtained the lowest validation

accuracy (84%) and it requires the longest training time (236 seconds per epoch) among the four transfer learning models. This is because the Xception's highest complexity (with 36 convolutional layers) consumes more computation time. In contrast, the VGG-16 model has achieved a higher validation accuracy (85%) with less time required for training (181 seconds per epoch). This is because the VGG-16 has the simplest architecture which is a sequential model with only 16 convolutional layers, although other models with more convolutional layers can still achieve higher accuracies. It is noteworthy that the MobileNet model with 28 layers has achieved the highest validation accuracy (86%) with the least amount of training time (167 seconds per epoch) overall. The second best performing ResNet model (with 34 convolutional layers) has achieved approximately the same accuracy level as the best performing MobileNet model, but it requires additional three dense layers and longer time for training (214 seconds per epoch). This is because unlike other models with standard convolutions, the MobileNet model is based on depthwise separable convolutions, which is a form of factorized convolutions which factorize a standard convolution into a depthwise convolution, and a 1×1 convolution called a pointwise convolution. While a standard convolution both filters and combines inputs into a new set of outputs in one step, the depthwise separable convolution splits this into two layers, a separate layer for filtering and a separate layer for combining. This factorization has the effect of drastically reducing computation and model size. Accordingly, Mobilenet is specifically developed with computational restraints in mind which suit the application in the car the best and it yields the lowest log loss (0.39 validation output loss) among the four stand-alone transfer learning models.

Nevertheless, including K-NN into the ensemble model can further improve the classification accuracy for each of the 10 classes. The idea is that after each of the transfer learning models completed predictions for all testing images, for each test image, its K most similar images (including itself) would be based on pixel-wise L2 euclidean norm. Due to the large quantity of images to be processed, the original test images had to be shrinked to 40 × 30 to shorten the computation time. Figure 7, as an example, shows one original test image and its four most similar images. The predicted probability of the test image was then replaced with average probabilities of all of these K images. This increased the stability of the performance of the model and lowered the log loss. In practice, applying K-NN to the 10-class predictions turned out to be very effective in terms of reducing the log loss (LB). It can generally improve the LB score by 0.03 ~ 0.1. For average calculation, both normal average and weighted average (weight 1 for the image itself, 0.9 for its first neighbor, 0.8 for its second neighbor... etc) were used. The weighted average yielded slightly lower log loss when K value increased. Results obtained from experimenting with different K values implied that optimal results were generated when K = 10.

Figure 7: *One original testing image and its four most similar images.*

The AutoML models turned out to have focused on the objects (e.g., phone and cup) held by the driver in the images, as both models were able to correctly classify all classes that involved talking or texting on the phone and holding a cup. The models struggled to classify images where the driver is not holding onto an object that became evident in poor classification results for drivers on safe driving, reaching behind, and talking to passenger postures. Confusion Matrix and sample images for true positives and false negatives are presented in Appendix F (Table F). Inference results for unseen images for each class using GCP and CreateML models are presented in Appendix G (Table G and Figure G).

Despite the excellent training scores, significant overfitting was observed overall as lower precision and recall scores were generated during validation, and much lower scores using unseen holdout dataset during testing. Precision and recall as low as 0.18 and 0.25 respectively was generated for some classes. The CreateML model reported slightly better predictive accuracy rates than those of the GCP model but GCP turned out to have the more reliable and consistent predictions across all the different classes for driving distraction behaviors. Consistency of predictions is very important for DDBC as the model needs the ability to correctly predict as many distracted behavior classes as possible. Appendix H (Table H) shows sample inferences on new images that are not part of training, validation, and test images. It can be dangerous (or not serving DDBC purpose) for example, if DDBC is not recognizing distracted behavior consistently (i.e., texting, talking on the phone, etc.) even if the driver is doing such behavior that impacts the driver's safety. It can also be very annoying for any user of DDBC to not be able to recognize safe driving positions consistently as DDBC will always send alerts to drivers for being distracted even if the driver is not distracted.

Cross-platform deployments (e.g., GCP model used for IOS deployment, or CreateML model for Google-based deployments) are not ideal for ML solution deployments, as these setups were proven to not work during the development process of DDBC. GCP provides the capability to deploy the DDBC ".mlmodel" for IOS, but did not actually work due to incompatibilities in signatures (e.g., number and data types of input and out parameters of the model). The GCP and CreateML models for DDBC only worked in the prototype that was developed using their own

proprietary environment. Inferencing time for distracted driver behavior classification averaged 0.31 milliseconds for both the GCP and CreateML models when used in the prototypes that were developed in Python project and Xcode projects, respectively. The inference results at 0.31 milliseconds returned back (or showed up) instantaneously when performed for a single image in the prototypes, but can still be very slow when it is eventually applied to frame-by-frame setup for video inputs. The Python and Xcode projects for the prototypes for DDBC can be accessed in the Github repository. Sample outputs from these prototypes are displayed in Figures 8.1 and 8.2 below.

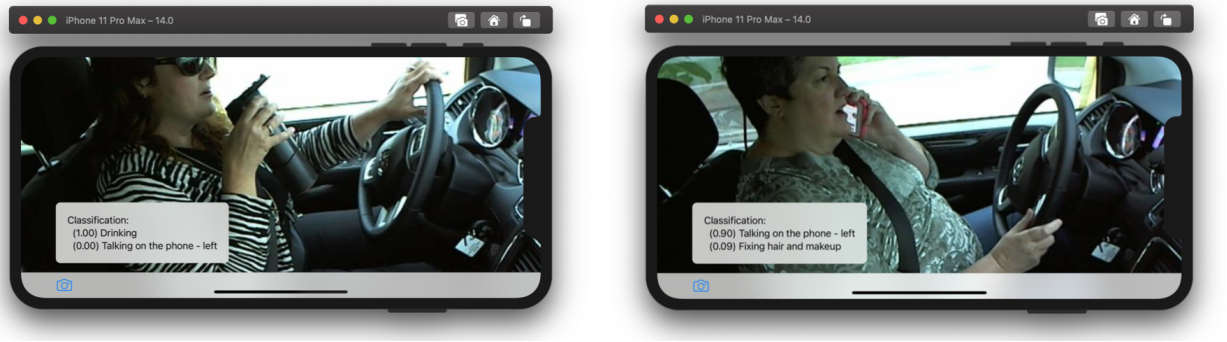


Figure 8.1: *Sample output of a prototype for an IOS app built on an Xcode project*

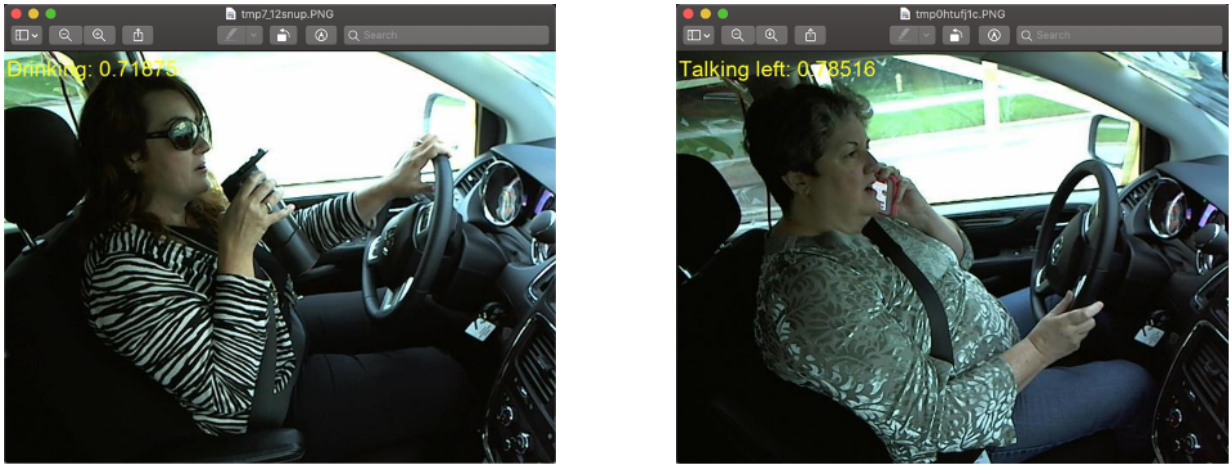


Figure 8.2: *Sample output of a prototype built on a Python project*

6. Conclusions

6.1 Conclusions

Distracted driving is a major problem leading to a striking number of accidents worldwide. Its detection is an important system component in semi-autonomous cars and driver safety. To solve this problem, this paper presented a real-time system for driver distraction identification that uses a learnable weighted ensemble of convolutional neural networks (CNNs), a new method for skin segmentation, a challenging distracted driver's dataset) on which the proposed solution was evaluated. The best model was the K-NN Ensemble model that achieved a classification accuracy over 90% for each of the 10 classes. This provides a baseline

performance for future research to benchmark against. One drawback to note was their performance overhead is much higher than their contribution.

The recommended AutoML model for DDBC for driver distraction behavior detection is the model created using GCP AutoML Vision that was trained for 200 iterations (with augmented images for exposure and blur added in the training data), as it was able to provide more consistent detection of distracted driver than the CreateML model.

DDBC presented a Computer Vision model that can classify different driver distraction behaviors. With relatively high classification accuracy, and prototypes that provide the foundation that can be built up for full scale deployments for cloud-based and edge-based solutions, DDBC proved to be a viable solution to replace or supplement sensor-based solutions for promoting driver safety. Similar to how the python project prototype used the Tensorflow Lite version of the DDBC model for inferencing, a similar architecture can be used to consume the model for inference tasks for Google-compatible deployments. Such deployments can include interfacing with other devices that can take actions based on DDBC inference (i.e., devices that can enable signals to alert the driver in scenarios where DDBC inferenced that the driver is texting while driving). Also similar to how the Xcode project-based prototype used the CreateML version of the DDBC model for inferencing, similar architecture can be used to support larger scale efforts/applications, such as applications using IOS devices like iPhone for distracted driver data collection that can help facilitate insurance claims processing, or incentive-based programs to lower insurance rates which are key objectives of the DDBC effort.

This model could facilitate faster claims processing in auto insurance. The insurance industry is adapting to Artificial Intelligence methodologies such as image processing to speed up an age old problem i.e. slow claims processing. DDBC model can help faster claim management process in insurance in which the insured's claim for compensation on an insured loss or damage is received, validated and verified, so the claim gets approved for compensation as quickly as possible. In an era where customers are the top priority, speeding up assessing the liability of a claim by using the DDBC model and making payments or taking appropriate next steps is definitely something that could earn customer satisfaction. Because high customer satisfaction levels can give a company a competitive edge, reducing the time it takes to settle insurance claims is one way to decrease the number of customer complaints and improve service.

The DDBC model can potentially be useful for auto insurance companies to charge drivers based on their actual driving rather than other statistics, such as their credit score, driving history, location, and age, etc. Incorporating the DDBC model into a mobile app would allow auto insurance companies to gain a true picture of the driving behavior of their policyholders. Inferences generated by the DDBC model can help evaluate how distracted a user is while driving. Meanwhile, images captured by the phone can also highlight the specific driving behaviors that have a strong correlation to risk. Insurers can use this information to price their policies more accurately, or to identify alternative options which will help customers avoid distracted driving behaviors. As the DDBC model's inferences and predictive accuracy would heavily influence an insurer's understanding of a specific driver's risk profile, it is important to ensure the robust quality of data going into the DDBC model in order to aid insurance companies with pricing their policies and predicting risk for each driver more accurately. Input data consistency can be achieved by normalizing the driving data collected by the phone, providing the added benefit of a device-agnostic program with uncompromised risk prediction. On the other hand, using the DDBC-based mobile app can help customers build up their safe driving profiles in a shorter period of time (e.g. two weeks) and receive more personalized insurance

premium offers accordingly. Benefits for consumers to adopt the DDBC-based mobile app include both the encouragement of safer driving — and the opportunity to earn discounts for safer driving behaviors — to make auto insurance fairer for drivers.

6.2 Directions for Future Work

Data augmentation using different exposure and blur levels on training data was performed during training for DDBC but as next steps for future work, data augmentation which involves color shifting and rotation of images could be applied to test out performance and accuracy. Data augmentation can also come to the rescue of the overfitting problem that was seen in some of the models above. A better methodology to detect a better face, hands, and skin would be a worthy exercise to pursue. The hand and face proposals could be manually labeled and used to train a Fast-RCNN (or any other object detector) to localize both faces and hands in one shot and evaluate it against existing models. A more powerful computing platform could be used to train these resource intensive models on a larger sample.

Model training and prototyping for DDBC using AWS SageMaker and Lambda functionality will continue. Generating models and prototypes under AWS opens up for another space of deployment opportunities for interfacing with Amazon assistant Alexa and all the Internet of Things (IoT) devices that Alexa interfaces with such as the Amazon Echo's suite of personal assistant products.

Bibliography

Abou-diab, Wisam. “Technology for Distracted Driving: Protect Your Fleet: GoFleet.” Go Fleet Tracking, February 27, 2019. https://www.gofleet.com/technology-for-distracted-driving.

Baheti, Bhakti, Suhas Gajre, and Sanjay Talbar. 2018. “Detection of Distracted Driver Using Convolutional Neural Network.” IEEE Xplore. https://ieeexplore.ieee.org/document/8575304.

Brown, Steven. 2020. “Working Collaboratively with Insurers and State Regulators to Bring Innovative Products to Market.” The Floow, December 17, 2020. https://www.thefloow.com/latest/working-collaboratively-with-insurers-and-state-regulators-to-bring-innovative-products-to-market/.

Chai, Chen, Juanwu Lu, Xuan Jiang, Xiupeng Shi, and Zeng Zeng. 2021. “An Automated Machine Learning (AutoML) Method for Driving Distraction Detection Based on Lane-Keeping Performance”. *arXiv preprint arXiv:2103.08311*.

Eraqi, Hesham M., Yehya Abouelnaga, Mohamed H. Saad, and Mohamed N. Moustafa. 2019. “Driver Distraction Identification with an Ensemble of Convolutional Neural Networks.” Hindawi. https://www.hindawi.com/journals/jat/2019/4125865/.

Feng, Demeng, and Yumeng Yue. 2019. “Machine Learning Techniques for Distracted Driver Detection.” Stanford. http://cs229.stanford.edu/proj2019spr/report/24.pdf.

Gumaei, Abdu, Mabrook Al-Rakhami, Mohammad Hassan, Atif Alamri, Musaed Alhussein, Abdur Razzaque, and Giancarlo Fortino. 2020. “A deep learning-based driver distraction identification framework over edge cloud.” ResearchGate. https://www.researchgate.net/publication/344431820_A_deep_learning-based_driver_distraction_identification_framework_over_edge_cloud.

He, Kaiming, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. “Deep residual learning for image recognition.” In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770-778. 2016.

Kumawat, Narendra. 2019. “Distracted Driver Detection using CNN.” Medium. https://medium.com/@nk-kumawat/distracted-driver-detection-using-cnn-ee5af6975bd7.

Lambert, Fred. 2019. “Tesla announces new ‘Deep Rain’ neural net for its automatic wipers.” electrek. https://electrek.co/2019/10/14/tesla-deep-rain-neural-net-automatic-wipers/.

Mahajan, Akhil. 2020. “Distracted Driver Detection Using Machine and Deep Learning Techniques.” Artificial Intelligence. https://ai.plainenglish.io/distracted-driver-detection-using-machine-and-deep-learning-techniques-1ba7e7ce0225.

Martinez, Viridiana. 2019. “AI can detect distracted drivers on the road.” Medium. https://medium.datadriveninvestor.com/ai-can-detect-distracted-drivers-on-the-road-6cebeecb0ead.

Moss, Robert. 2020. “AI and Computer Vision Drive Fleet Safety.” insight.tech. https://www.insight.tech/content/ai-and-computer-vision-drive-fleet-safety#main-content.

National Highway Traffic Safety Administration, December 31, 2019. https://www.nhtsa.gov/risky-driving/distracted-driving.

Omerustaoglu, Furkan, C. Okan, and SakarGorkem Kar. 2020. “Distracted driver detection by combining in-vehicle and image data using deep learning.” Applied Soft Computing. https://www.sciencedirect.com/science/article/abs/pii/S1568494620305950.

Otaibi, Abdullah. 2019. “Distracted Driver Detection.” Medium. https://medium.com/@abbz.hamil/distracted-driver-detection-d26f42905f87.

Pachigolla, Satya Naren. 2019. “Distracted Driver Detection using Deep Learning.” Towards Data Science. https://towardsdatascience.com/distracted-driver-detection-using-deep-learning-e893715e02a4.

Sheng, Weihua, Duy Tran, Ha Do, and Girish Chowdhary. 2018. “Real-time Detection of Distracted Driving based on Deep Learning.” The Institution of Engineering and Technology. https://www.researchgate.net/publication/326740203_Real-time_Detection_of_Distracted_Driving_based_on_Deep_Learning.

Simonyan, Karen, and Andrew Zisserman. “Very deep convolutional networks for large-scale image recognition.” *arXiv preprint arXiv:1409.1556* (2014).

Singh, Paul. 2018. “SmartWitness Launch Fatigue Detection Tool as 17% of Drivers Sleep at the Wheel.” Fleet News. Fleet News, November 5, 2018. https://www.fleetnews.co.uk/news/fleet-industry-news/2018/05/11/smartwitness-launch-fatigue-detection-tool-as-17-admit-to-sleeping-at-the-wheel.

Wise, Ed. 2018. “Mojo, an App That Can Reduce Distracted Driving.” Wise Insurance Group, March 28, 2018. https://wiseinsurancegroup.com/mojo-app-can-reduce-distracted-driving/.

Zenduit. 2021. “Live Safety Monitoring and Accident Prevention.” Zenduit. https://zenduit.com/products/driver-distraction-camera/.

Appendix A

Baseline CNN Model Figures

Diagram 1. Baseline CNN model Training/Validation Accuracy & Loss vs Epochs.

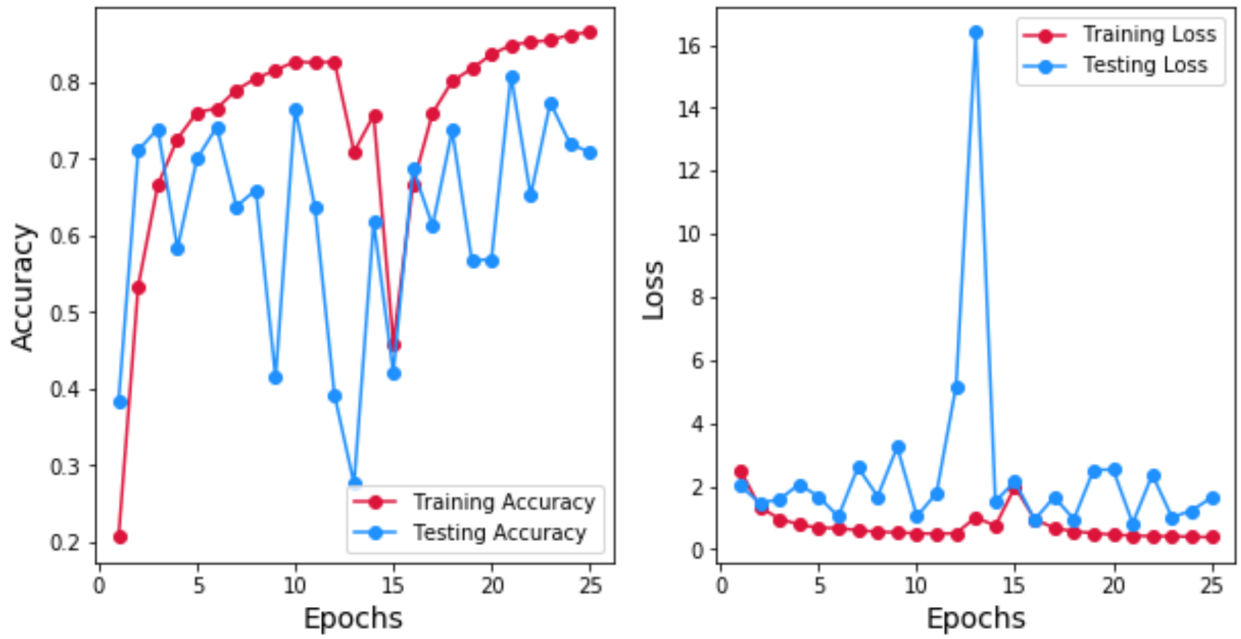
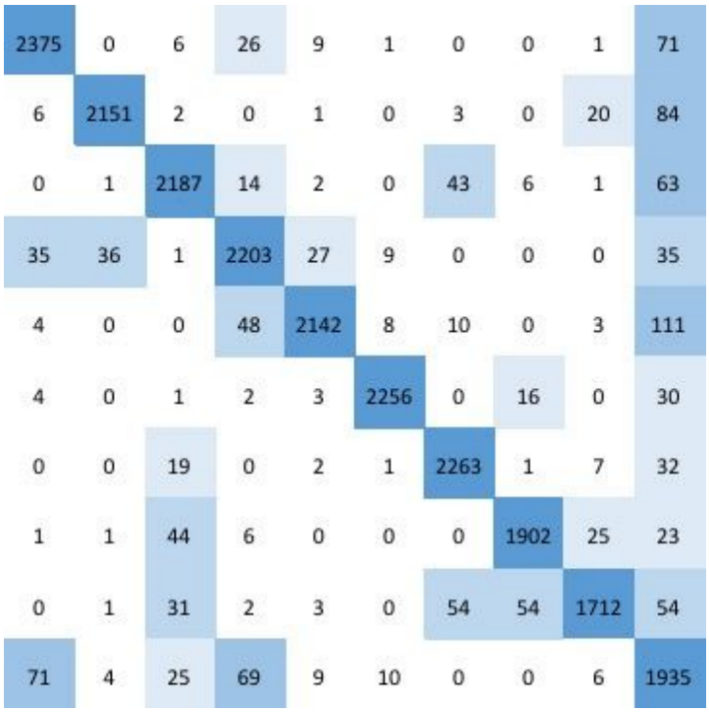


Diagram 2. Confusion matrix of the baseline CNN model.



Appendix B

VGG-16 Model Figures

Diagram 1. VGG-16 without Extra Layers Training/Validation Accuracy & Loss vs Epochs.

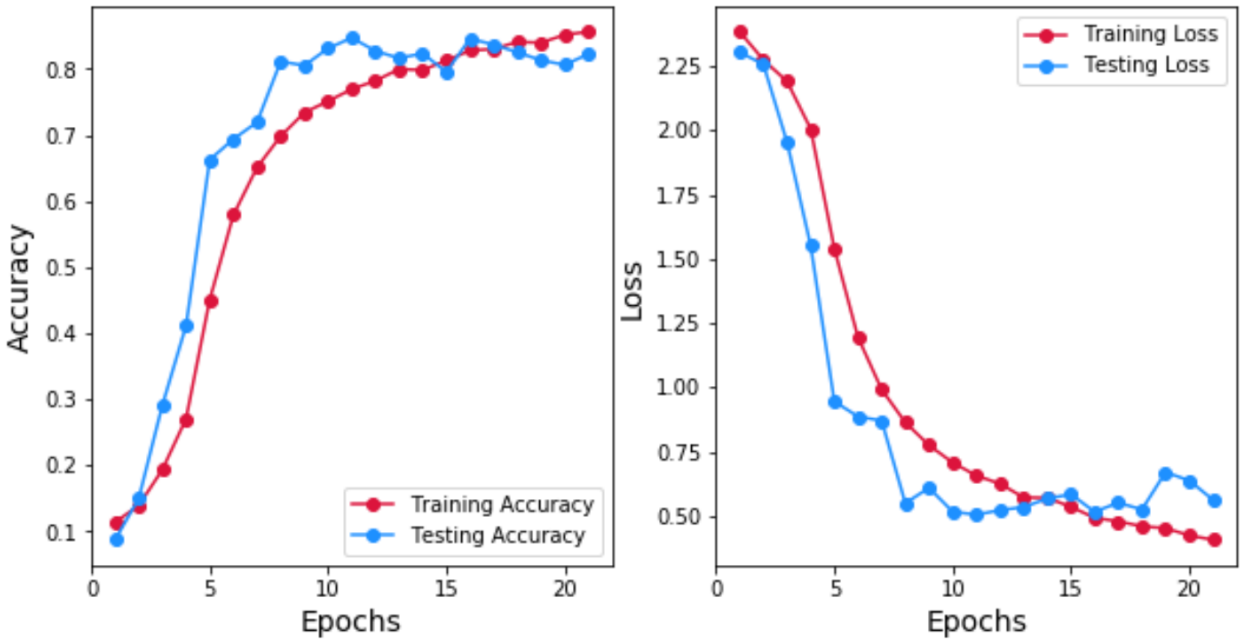


Diagram 2. VGG-16 with 3 Dense Extra Layers Training/Validation Accuracy & Loss vs Epochs.

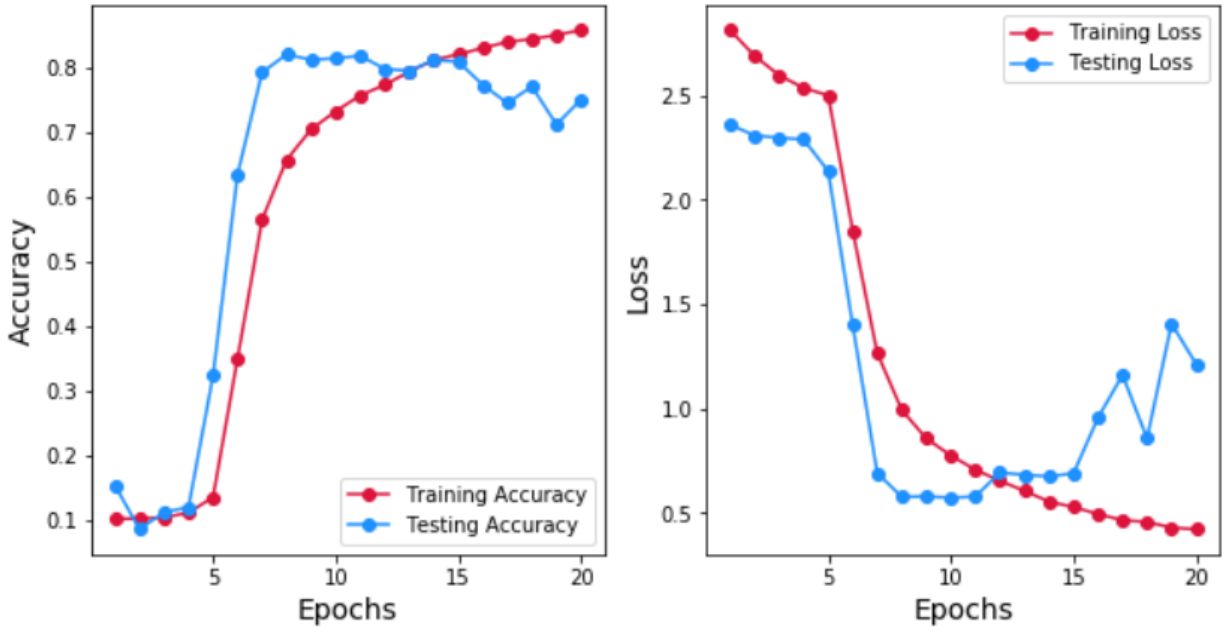
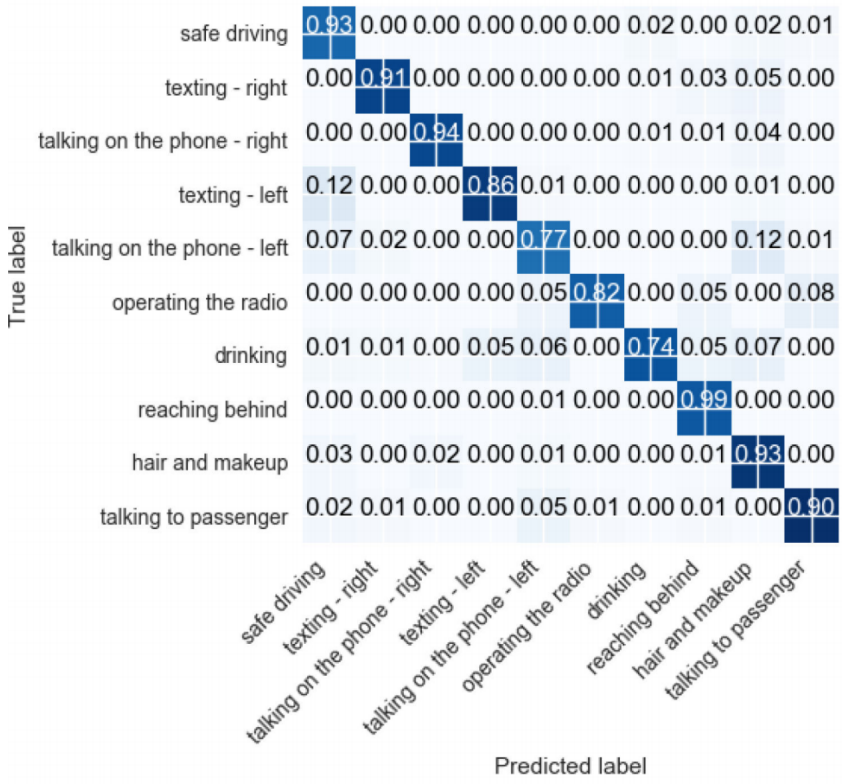


Diagram 3. Confusion matrix of evaluation results of the VGG-16 model.



Appendix C

ResNet Model Figures

Diagram 1. ResNet without Extra Layers Training/Validation Accuracy & Loss vs Epochs.

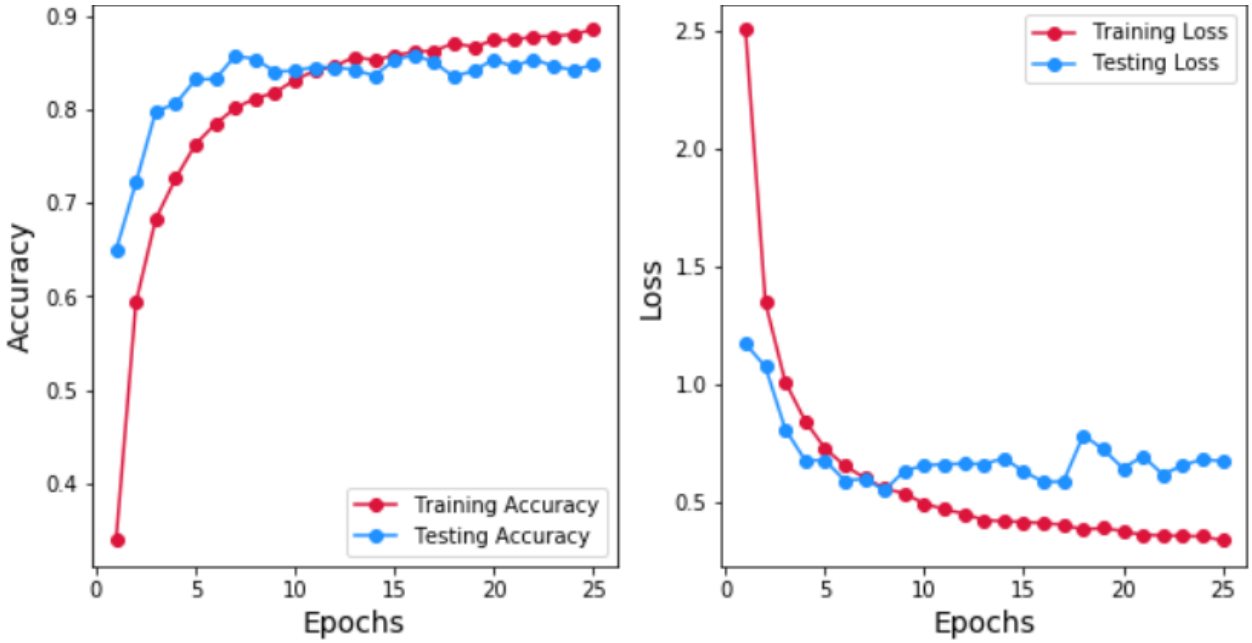


Diagram 2. ResNet with 3 Dense Extra Layers Training/Validation Accuracy & Loss vs Epochs.

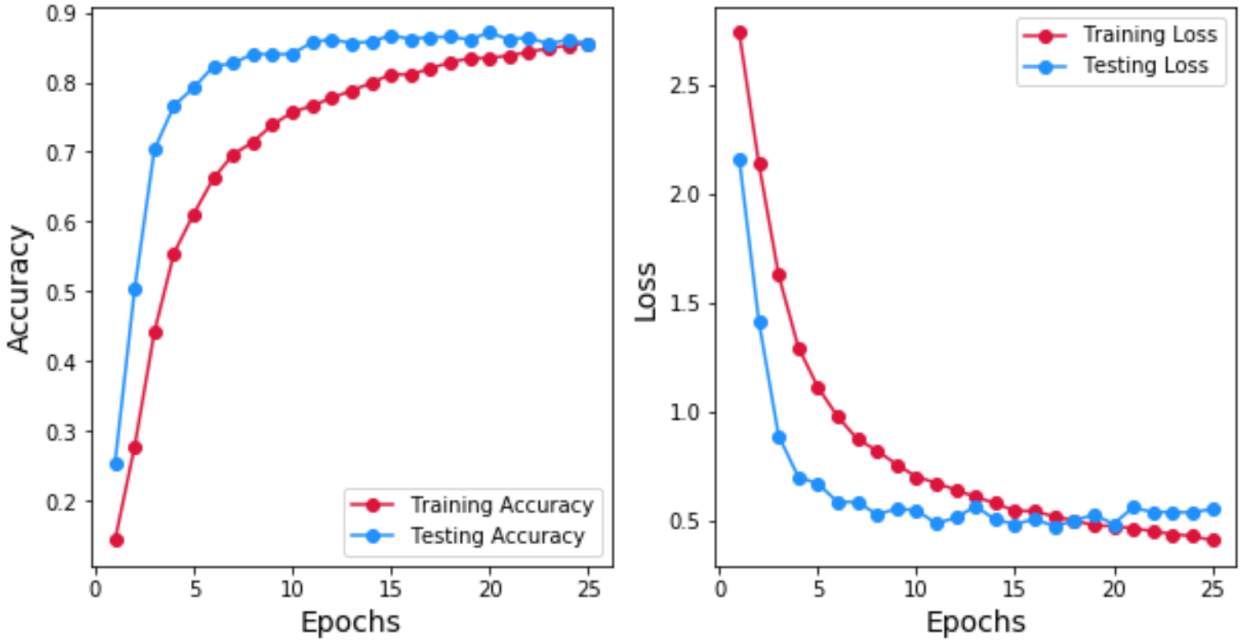
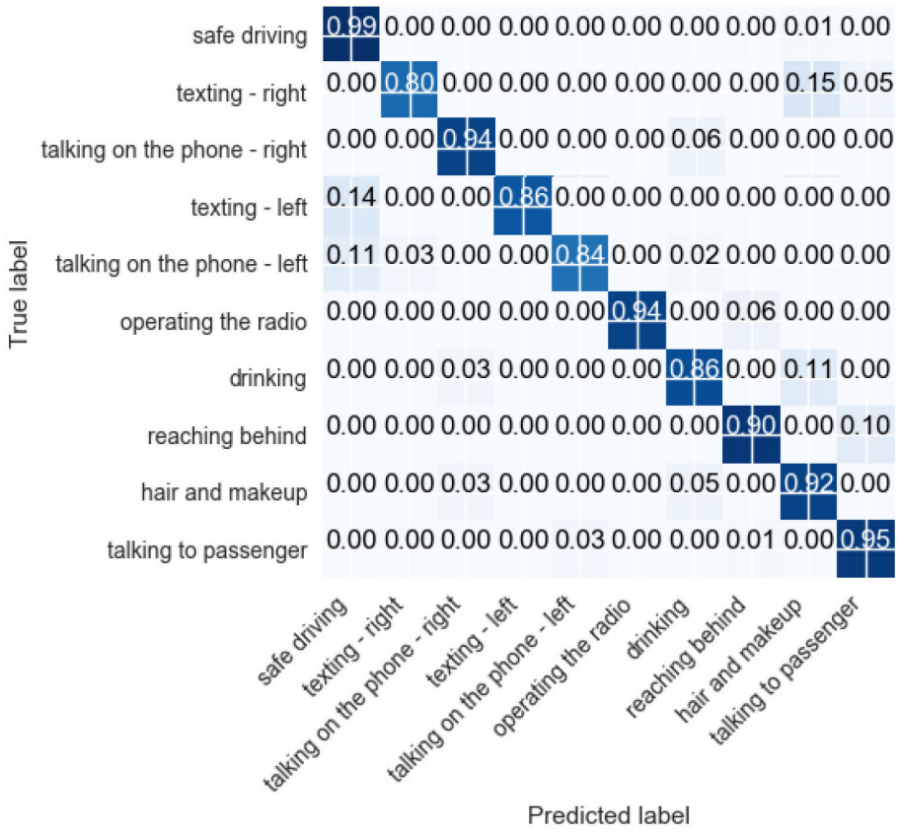


Diagram 3. Confusion matrix of evaluation results of the ResNet model.



Appendix D

Xception Model Figures

Diagram 1. Xception without Extra Layers Training/Validation Accuracy & Loss vs Epochs.

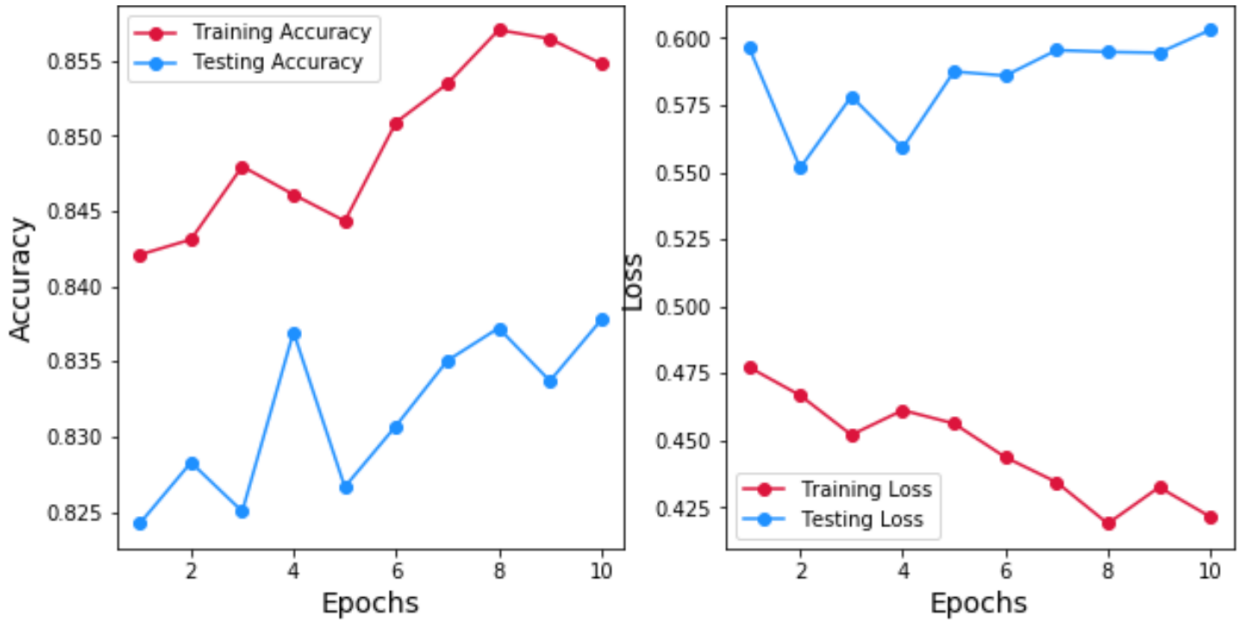


Diagram 2. Xception with 3 Dense Extra Layers Training/Validation Accuracy & Loss vs Epochs.

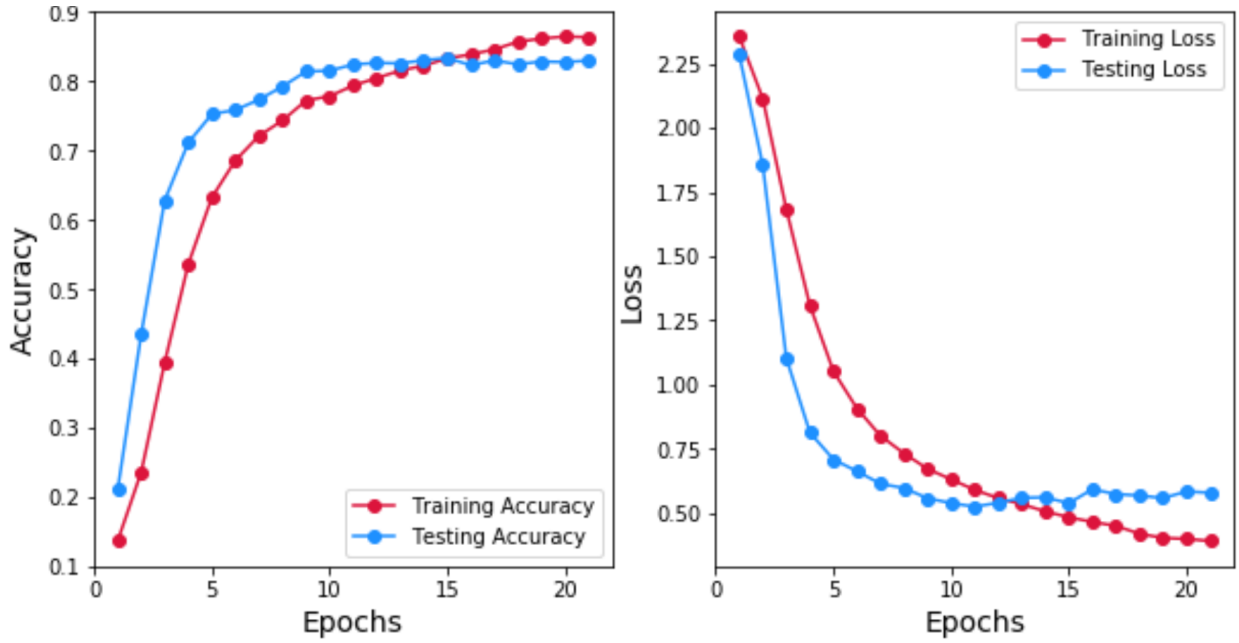
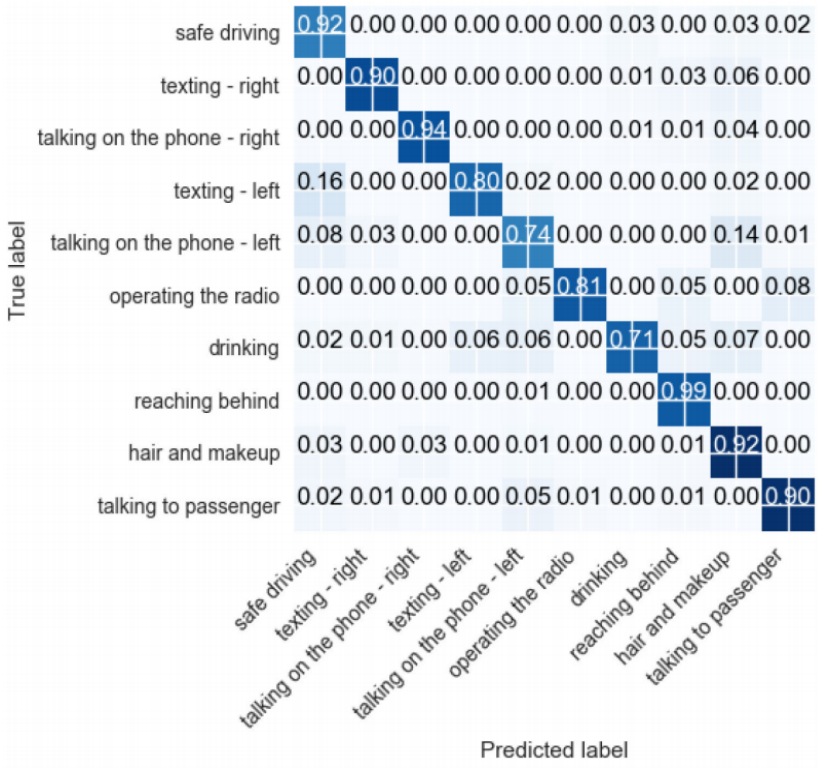


Diagram 3. Confusion matrix of evaluation results of the Xception model.



Appendix E

MobileNet Model Figures

Diagram 1. MobileNet without Extra Layers Training/Validation Accuracy & Loss vs Epochs.

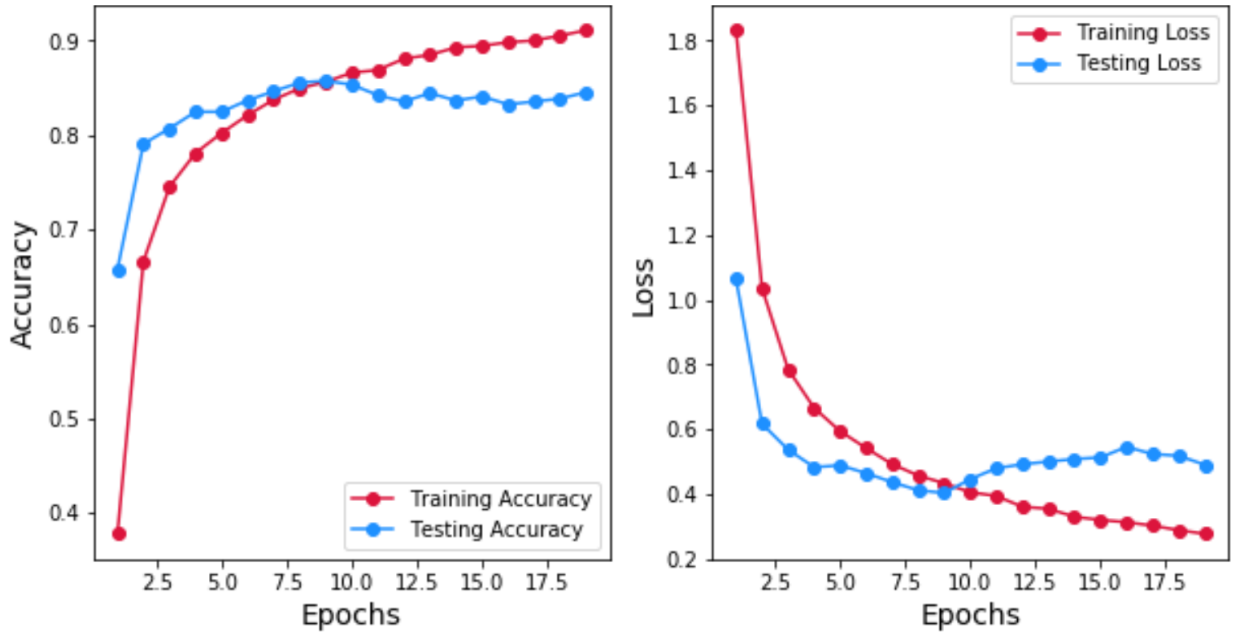


Diagram 2. MobileNet with 3 Dense Extra Layers Training/Validation Accuracy & Loss vs Epochs.

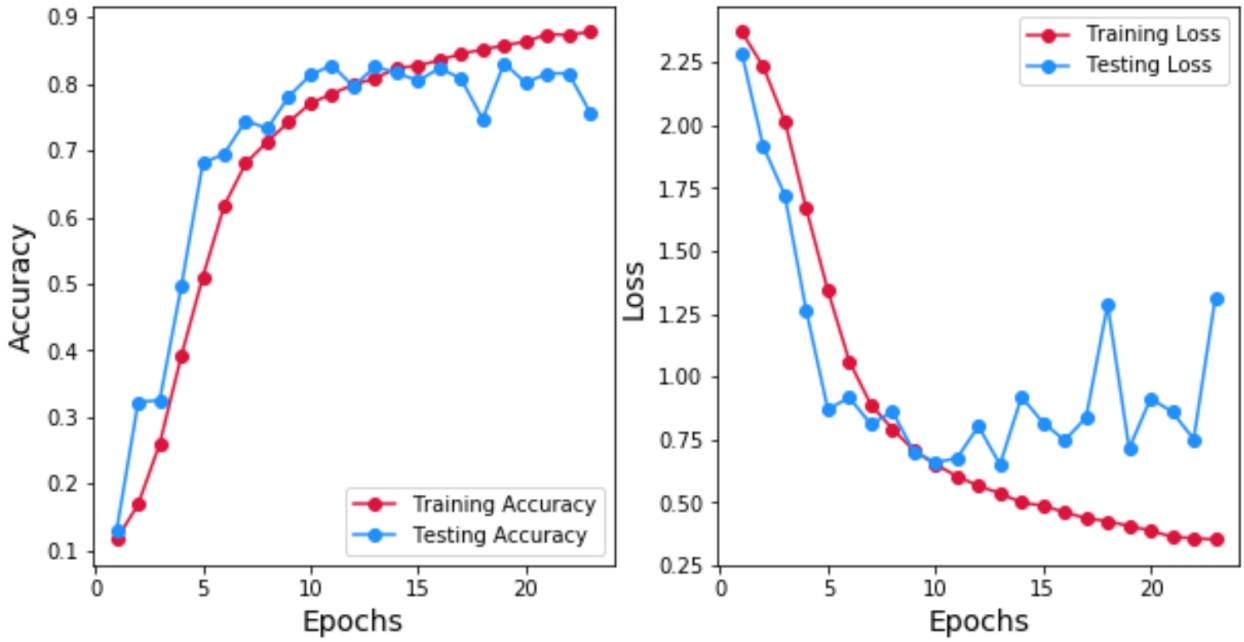
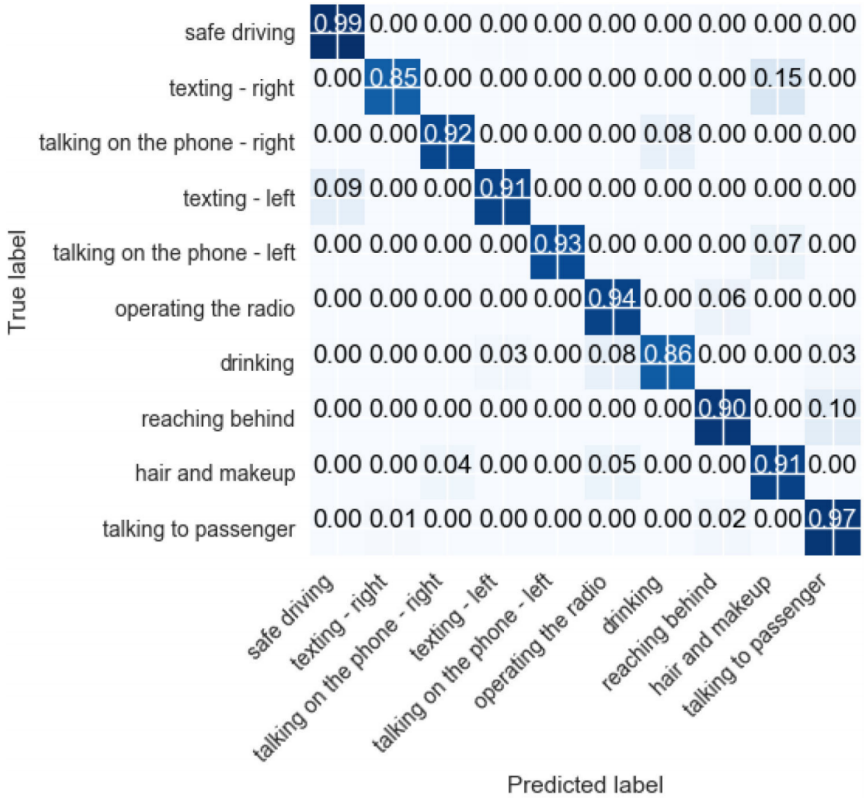


Diagram 3. Confusion matrix of evaluation results of the MobileNet model.



Appendix F: AutoML: Classification Performance Results

Table F: Precision and Recall for GCP and CreateML AutoML models.

DISTRACTION CLASS	Google Cloud Platform (GCP) AutoML Vision						Apple CreateML					
	TRAIN		VALIDATION		TEST		TRAIN		VALIDATION		TEST	
	PRECISION	RECALL	PRECISION	RECALL	PRECISION	RECALL	PRECISION	RECALL	PRECISION	RECALL	PRECISION	RECALL
	WITH IMAGE EXPOSURE AND BLUR DATA AUGMENTATION (200 ITERATIONS)											
	TRAINING ACCURACY: 99.4% VALIDATION ACCURACY: 95%						**TRAINING ACCURACY: 99.6% VALIDATION ACCURACY: 95.3%**					
	TRAINING TIME: 8 hours						**TRAINING TIME: 8 hours**					
Safe Driving	0.99	0.98	0.92	0.9	0.7	0.61	0.99	0.99	0.92	0.89	0.71	0.51
Texting - right	1	1	0.98	0.95	0.89	0.55	1	1	0.98	0.92	0.89	0.53
Talking on the phone - right	0.99	1	0.98	0.98	0.84	0.78	1	1	0.98	0.98	0.83	0.76
Texting - left	1	1	0.97	94	0.73	0.73	1	1	0.98	0.92	0.71	0.71
Talking on the phone - left	1	1	0.98	0.98	0.81	0.72	1	1	0.97	0.97	0.8	0.7
Operating the radio	0.99	0.98	0.97	0.96	0.77	0.7	1	1	0.95	0.98	0.75	0.69
Drinking	1	1	0.97	0.97	0.63	0.58	1	1	0.96	0.96	0.62	0.57
Reaching behind	1	1	0.98	0.97	0.88	0.84	1	1	0.96	0.98	0.87	0.85
Hair and makeup	0.99	0.99	0.95	0.94	0.48	0.81	0.99	0.99	0.92	0.96	0.44	0.79
Talking to passenger	0.99	0.99	0.9	0.98	0.46	0.68	0.99	0.99	0.9	0.97	0.45	0.67
	WITH 400 ITERATIONS											
	TRAINING ACCURACY: 91.3% VALIDATION ACCURACY: 72%						**TRAINING ACCURACY: 90.0% VALIDATION ACCURACY: 66.0%**					
	TRAINING TIME: 4 hours						**TRAINING TIME: 3 hours**					
Safe Driving	0.85	0.81	0.57	0.53	0.55	0.21	0.83	0.84	0.56	0.51	0.56	0.18
Texting - right	0.93	0.94	0.77	0.36	0.54	0.21	0.92	0.93	0.76	0.33	0.53	0.2
Talking on the phone - right	0.91	0.93	0.86	0.81	0.66	0.54	0.93	0.94	0.87	0.82	0.68	0.53
Texting - left	0.94	0.94	0.68	0.71	0.49	0.43	0.93	0.93	0.69	0.7	0.48	0.42

Talking on the phone - left	*0.93*	*0.93*	*0.81*	*0.61*	*0.69*	*0.34*	*0.94*	*0.94*	*0.79*	*0.59*	*0.68*	*0.33*
Operating the radio	*0.83*	*0.88*	*0.62*	*0.77*	*0.38*	*0.44*	*0.88*	*0.89*	*0.6*	*0.76*	*0.37*	*0.43*
Drinking	*0.92*	*0.91*	*0.53*	*0.87*	*0.26*	*0.71*	*0.93*	*0.93*	*0.54*	*0.89*	*0.25*	*0.69*
Reaching behind	*0.92*	*0.93*	*0.85*	*0.81*	*0.64*	*0.78*	*0.93*	*0.94*	*0.84*	*0.8*	*0.63*	*0.77*
Hair and makeup	*0.82*	*0.81*	*0.48*	*0.69*	*0.25*	*0.46*	*0.83*	*0.8*	*0.48*	*0.68*	*0.26*	*0.44*
Talking to passenger	*0.88*	*0.86*	*0.77*	*0.54*	*0.41*	*0.24*	*0.86*	*0.85*	*0.75*	*0.52*	*0.39*	*0.2*
	WITH 100 ITERATIONS											
	TRAINING ACCURACY: 97.8% VALIDATION ACCURACY: 94%						***TRAINING ACCURACY: 98.5% VALIDATION ACCURACY: 95.3%***					
	TRAINING TIME: 1 hour						***TRAINING TIME: 1 hour***					
Safe Driving	*0.97*	*0.98*	*0.87*	*0.9*	*0.51*	*0.57*	*0.96*	*0.96*	*0.88*	*0.91*	*0.49*	*0.55*
Texting - right	*0.98*	*0.98*	*0.94*	*0.96*	*0.79*	*0.5*	*0.99*	*0.99*	*0.96*	*0.97*	*0.83*	*0.47*
Talking on the phone - right	*0.98*	*0.98*	*0.96*	*0.95*	*0.88*	*0.58*	*1*	*0.99*	*0.97*	*0.96*	*0.89*	*0.59*
Texting - left	*0.99*	*0.99*	*0.97*	*0.95*	*0.42*	*0.85*	*0.99*	*0.99*	*0.97*	*0.95*	*0.43*	*0.86*
Talking on the phone - left	*0.98*	*0.98*	*0.96*	*0.96*	*0.68*	*0.63*	*0.99*	*0.99*	*0.97*	*0.97*	*0.66*	*0.65*
Operating the radio	*0.98*	*0.97*	*0.98*	*0.95*	*0.91*	*0.6*	*0.99*	*0.99*	*0.99*	*0.97*	*0.92*	*0.56*
Drinking	*0.99*	*0.99*	*0.96*	*0.95*	*0.58*	*0.62*	*0.99*	*0.99*	*0.97*	*0.96*	*0.57*	*0.6*
Reaching behind	*0.98*	*0.98*	*0.98*	*0.97*	*0.91*	*0.81*	*0.99*	*1*	*0.99*	*0.98*	*0.9*	*0.78*
Hair and makeup	*0.97*	*0.97*	*0.93*	*0.94*	*0.49*	*0.63*	*0.98*	*0.97*	*0.93*	*0.94*	*0.48*	*0.61*
Talking to passenger	*0.97*	*0.97*	*0.92*	*0.94*	*0.64*	*0.54*	*0.98*	*0.97*	*0.93*	*0.93*	*0.63*	*0.52*

Appendix G: AutoML: Confusion Matrix and Sample Test Inferences

Table G: Precision and Recall for GCP and CreateML AutoML models.

True Label \ Predicted Label	talking left	texting left	drinking	reaching behind	texting right	operate radio	talking right	talking passenger	hair makeup	safe driving
talking left	100%	-	-	-	-	-	-	-	-	-
texting left	-	100%	-	-	-	-	-	-	-	-
drinking	-	-	100%	-	-	-	-	-	-	-
reaching behind	-	-	-	100%	-	-	-	-	-	-
texting right	-	-	-	-	100%	-	-	-	-	-
operate radio	-	-	-	-	-	99%	-	-	0%	1%
talking right	-	-	0%	-	-	-	100%	-	-	-
talking passenger	-	-	-	-	-	-	-	99%	1%	-
hair makeup	-	-	0%	-	-	-	0%	-	99%	-
safe driving	-	-	-	-	-	-	-	-	0%	100%

Figure G: Sample test inferences from AutoML models.

True positives

Your model correctly predicted **texting left** on these images

Score: 0.67843133 | Score: 0.73333335 | Score: 0.7372549 | Score: 0.77254903 | Score: 0.7764706

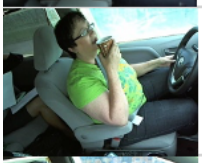
False negatives

Your model should have predicted **hair makeup** on these images

Score(s): 0.29803923 | Score(s): 0.33333334 | Score(s): 0.3529412 | Score(s): 0.44313726

Appendix H: AutoML: Inference Results on New/Unseen Images

Table H: Sample inference results (with confidence rate) for GCP and CreateML AutoML models on new/unseen images (green font are correct classifications, red are incorrect classifications)

DISTRACTION CLASS	Google Cloud Platform (GCP) AutoML Vision	Apple CreateML
	Texting - left (0.63)	Texting - left (0.88)
	Texting - right (0.72)	Texting - right (0.99)
	Talking on the phone - right (0.81)	Talking on the phone - right (0.99)
	Texting - left (0.85)	Safe driving (0.79)
	Talking on the phone - left (0.83)	Talking on the phone - left (0.99)
	Operating the radio (0.74)	Operating the radio (0.85)
	Drinking (0.30)	Drinking (0.92)
	Reaching behind (0.73)	Talking to passenger (0.70)
	Hair and makeup (0.77)	Hair and makeup (0.98)
	Operating the radio (0.31)	Operating the radio (0.51)