Predicting Distracted Drivers

Milestone Report 2

Tracy Cardwell

https://github.com/tcardwell/Predicting-Distracted-Drivers/blob/master/3_Model.ipynb

Introduction

The National Highway Traffic Safety Administration estimated that in 2018, driver distraction was a factor in about 8 percent of all fatal crashes and 15 percent of all injury crashes. Every day about 8 people were killed and more than 1000 injured in crashes that involved a distracted driver. These estimates may be low as it is challenging to determine distraction in driver fatality crashes, and self-reported distractions are likely underreported.

Distractions include any activity that takes your attention away from driving and fall into three main categories:
• visual: looking at something not related to driving;
• manual: taking your hands off the steering wheel; and
• cognitive: taking your mind off driving.

Systems that detect distractions and report them to drivers in real-time could reduce distracted driving accidents and deaths.

This project aims to identify driver distractions from dashboard camera images.

Deep Learning

Initial Processing Plan

Convolutional Neural Networks (CNNs) were selected to model this data as they currently yield the best accuracy for image classification. The initial plan was designed to begin with the simplest model possible and increase complexity as needed to improve performance. The plan was defined as below:

1. Simple CNN with no data augmentation
2. More complex CNN with no data augmentation
3. More complex CNN with data augmentation
4. Transfer learning

Several pre-trained architectures were chosen to compare their results. MobileNet V2 was included as it was specifically designed to be light weight for mobile devices.

Data Preprocessing

A total of 22,424 labeled images were cropped to remove extraneous data and split into training, validation and test sets in preparation for modeling. The data included ten classes as follows:

- c0: safe driving,

- c1: texting – right hand,
- c2: talking on the phone – right hand,
- c3: texting – left hand,
- c4: talking on the phone – left hand,
- c5: operating the radio,
- c6: drinking,
- c7: reaching behind,
- c8: hair and makeup, and
- c9: talking to passenger.

Data augmentation options were tested and maximum parameter values identified.

Images were downsized as the original size of 640 x 480 caused resource issues in even the smallest CNN. Various image sizes and color modes were tested to determine their effect on model performance. In the end, 224 x 224 RGB images were used for all built from scratch CNN models. This allowed for better comparison with transfer learning as it is the default for most of the transfer learning models used. All pre-trained models were trained with their default input shapes.

ImageDataGenerator() was used to feed images to the models in batches as the images did not fit into memory. Images were rescaled to a range of [0,1] for all built from scratch CNN models. Architecture specific preprocessors were used instead of rescaling for all pre-trained models.

Training parameters

Batch size was set to 32 images for all training. Larger batch sizes were tested on some models and yielded larger losses.

Models were trained for 10-30 epochs, depending on performance and speed of convergence. Early stopping was used to minimize validation loss. Some models may have benefitted from longer training; however, epoch count was capped at 30 due to resource and time constraints.

Several optimizers were tested including Adam, RMSprop, and stochastic gradient descent. In almost every case, stochastic gradient descent with learning rate decay resulted in the lowest loss and most stable training. Some attempts with Adam, especially, resulted in highly vacillating losses and accuracy. In general, lower learning rates worked better with this data. Learning rates used with transfer learning models were lower than those used with scratch-built models, even when training only new output layers.

Models

Simple CNN with no augmentation

The first model tested was a very simple model with just a few convolution layers, ReLU activation, max pooling after each convolution and activation, flatten, and a few dense layers with one dropout layer. The model performed very poorly. Its accuracy remained below 50 percent, it was extremely overfit and had a persistent high cross entropy loss.

Accuracy	Validation Loss	Range of Class F1-scores	Epochs	Learning Rate (Adam)
.43	3.5	.18 - .57	10	.0005

More complex CNN with no augmentation

The second model was a deeper model to allow better learning of the sometimes subtle differences between classes. It included BatchNormalization to reduce error and improve performance. It also included a more complex fully connected block to better learn the classes, and dropout layers to help prevent overfitting.

This model performed better than the simple model but was still extremely overfit and poor at classifying the images.

Accuracy	Validation Loss	Range of Class F1-scores	Epochs	Learning Rate (SGD)
.48	2.3	.27 - .63	18	.1 - .004

More complex CNN with augmentation

The same model was trained with augmented data to add regularization and decrease overfitting.

The accuracy and class f1-scores increased a bit but the loss also increased significantly after the first couple of epochs and never came back down. Several adjustments were made to learning rate to address the increased loss but no real improvement was made.

Accuracy	Validation Loss	Range of Class F1-scores	Epochs	Learning Rate (SGD)
.53	3.4	.33 - .87	12	.05 - .01

Transfer Learning

Several transfer learning architectures were explored. The methodology used was as follows:

1. Create a model with a pre-trained base and the complex output block used in the scratch-built models. Freeze the base and train only the new layers.
2. Unfreeze part or all of the base and fine-tune.
3. Create another model with a pre-trained base and a simple output block for comparison with the complex version. Freeze the base and train only the new layers.
4. Train a model with the complex output block by fine-tuning the entire model from the outset, with no frozen layers.
5. Train a model with the simple output by fine-tuning the entire model from the beginning, with no frozen layers.

The models were first trained with a complex output block in hopes of improving performance in detecting subtle differences between classes. The thought was perhaps a complex block of fully connected layers would be better able to find these differences than a simple softmax output layer. This held true for the simpler models, so it was tested in the transfer learning models.

Some of these steps were omitted for some architectures, when it became clear they were not leading to improvements in the models.

ResNet50

1. Train new output separately, then fine-tune

This transfer learning model had significantly better performance than the scratch-built CNN. Unfreezing and fine-tuning additional blocks of the ResNet50 model did not improve performance. Tests were made unfreezing just one block, then two blocks while reducing the learning rate from that used at the end of the initial training. It would be interesting to see if a lower learning rate than that used would perform better.

Changing the output to a simple softmax layer did not change the performance of the model, so the complexity of the output block did not add anything and was unnecessary in this case.

	Accuracy	Validation Loss	Range of Class F1-scores	Epochs	Learning Rate (SGD)
Complex	.72	.98	.41 - .85	24	.001 - .0002
Simple	.72	1.05	.41 - .88	30	.001

2. Fine tune entire model from the start

Fine-tuning the entire ResNet50 model from the beginning yielded the best results.

The complex output resulted a very small increase in test accuracy over the simple output but also a bit more validation loss. The simple model could have benefitted from more training as the loss was still dropping slowly after 30 epochs.

	Accuracy	Validation Loss	Range of Class F1-scores	Epochs	Learning Rate (SGD)
Complex	.87	.57	.57 - .96	21	.001 - .00025
Simple	.84	.44	.58 - .99	30	.001 - .00025

One test was run with a flatten layer instead of global average pooling as had been used on all other models. This led to an increase in loss and drop in accuracy so was not repeated with any other models.

MobileNet V2

1. Train new output separately, then fine-tune

Only the complex model was tuned with the MobileNet V2 base frozen. Unfreezing the base and fine-tuning did not improve performance, just as with ResNet50. Fine-tuning was tested with the same learning rate in use at the end of the initial training. Perhaps a lower learning rate than that tested would have improved the model.

Accuracy	Validation Loss	Range of Class F1-scores	Epochs	Learning Rate (SGD)
.64	1.4	.26 - .80	20	.001 - .0005

2. Fine tune entire model from the start

Both the complex and simple output models were tested by fine-tuning the entire model. The models are very similar in performance with a small difference in test accuracy.

	Accuracy	Validation Loss	Range of Class F1-scores	Epochs	Learning Rate (SGD)
Complex	.82	.59	.43 - .95	27	.001 - .00025
Simple	.78	.58	.43 - .96	28	.001 - .00025

InceptionV3

1. Train new output separately, then fine-tune

Only the complex output model was trained with the Inception base frozen. No additional fine-tuning was performed.

Accuracy	Validation Loss	Range of Class F1-scores	Epochs	Learning Rate (SGD)
.62	1.2	.37 - .80	29	.001 - .00025

2. Fine tune entire model from the start

The performance of the complex and simple output models was very similar, with the simple model having a lower loss but a little more trouble with class c9. Class c9 had the lowest f1-score for both models. In each case, this was due to low recall. This is the first instance in which the complex output made a significant difference in class performance.

	Accuracy	Validation Loss	Range of Class F1-scores	Epochs	Learning Rate (SGD)
Complex	.82	.72	.50 - .95	21	.001 - .00025
Simple	.83	.57	.36 - .94	29	.001 - .0005

VGG19

1. Train new output separately, then fine-tune

Only the complex output model was trained with the base frozen. Additional fine-tuning was not performed.

Accuracy	Validation Loss	Range of Class F1-scores	Epochs	Learning Rate (SGD)
.62	1.7	.27 - .84	14	.001 - .0005

2. Fine tune entire model from the start

Accuracy and cross entropy loss were more variable when fine-tuning the entire VGG19 model than with other transfer learning architectures.

Note: The reported validation loss for the simple model is not the minimum as the model did not converge after 20 epochs of training. The minimum was on epoch 8 with a validation loss of .58 and a validation accuracy of .87. Had that model been saved and tested, VGG19 would have followed the pattern of the other transfer learning architectures in terms of similar or lower loss with the simple output design.

	Accuracy	Validation Loss	Range of Class F1-scores	Epochs	Learning Rate (SGD)
Complex	.80	.58	.52 - .96	20	.001 - .0005
Simple	.83	.83	.48 - .96	20	.001 - .0005

Findings

1. High variability in performance across classes

Every model had a wide range of precision and recall scores over the classes. All models had a difficult time with the images in three classes in particular: c8 – Hair and Makeup, c9 – Talking to Passenger, and c0 – Safe Driving.

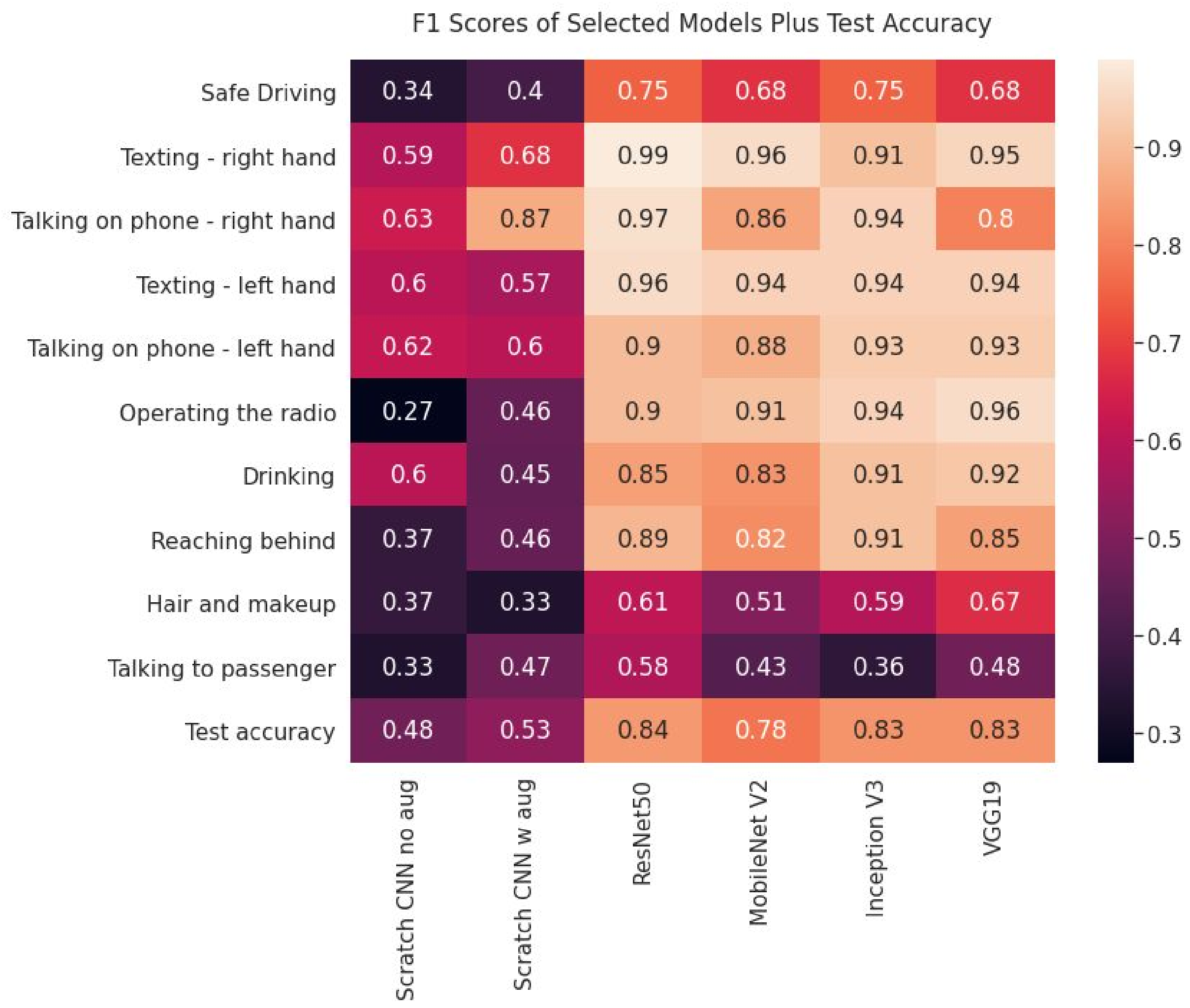
Reviewing the images, it is easy to see that Talking to Passenger often looks like Safe Driving. Most images in this class show the driver's head turned to look at the Passenger but some show the driver looking straight ahead. Most images show both hands on the steering wheel as in Safe Driving. Some drivers have their mouths open as if they are talking. Some have their mouths closed and do not appear to be talking.

Hair and Makeup class images have a wide variety of features. Some show hands touching the face, some touching the hair, some touching glasses. Some are using the right hand, some using the left. Some are not touching their face or hair at all but just looking in the mirror.

Most of the Safe Driving images show both hands on the steering wheel and looking straight ahead. Some images, however, show the driver looking to the right. Some show a driver smiling, which could be mistaken for an open mouth speaking.

It would be difficult for a human to correctly classify all the images as labeled.

The heatmap below compares the F1 scores for each class, for each model tested. The scores shown for the transfer learning architectures are for the simple output model versions. Test accuracy is also included.



	Scratch CNN no aug	Scratch CNN w aug	ResNet50	MobileNet V2	Inception V3	VGG19
Safe Driving	0.34	0.4	0.75	0.68	0.75	0.68
Texting - right hand	0.59	0.68	0.99	0.96	0.91	0.95
Talking on phone - right hand	0.63	0.87	0.97	0.86	0.94	0.8
Texting - left hand	0.6	0.57	0.96	0.94	0.94	0.94
Talking on phone - left hand	0.62	0.6	0.9	0.88	0.93	0.93
Operating the radio	0.27	0.46	0.9	0.91	0.94	0.96
Drinking	0.6	0.45	0.85	0.83	0.91	0.92
Reaching behind	0.37	0.46	0.89	0.82	0.91	0.85
Hair and makeup	0.37	0.33	0.61	0.51	0.59	0.67
Talking to passenger	0.33	0.47	0.58	0.43	0.36	0.48
Test accuracy	0.48	0.53	0.84	0.78	0.83	0.83

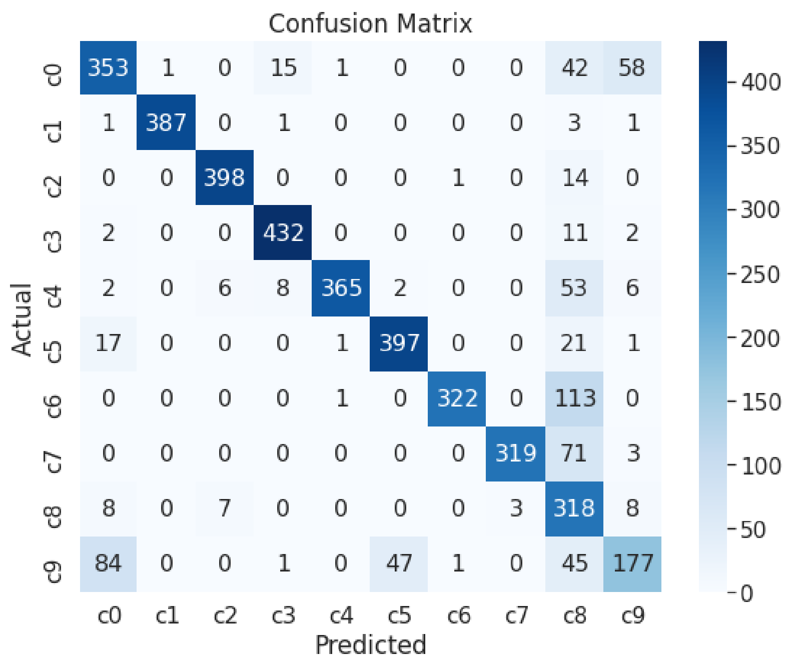
This clearly illustrates that class c9, talking to passenger, gives models the most trouble, followed by class c8, hair and makeup, and class c0, safe driving.

2. Transfer Learning was key to good performance

The above heatmap also shows a huge difference in performance between the scratch-built models and the pre-trained models. This classification problem benefited greatly from transfer learning.

3. Best model: Resnet50

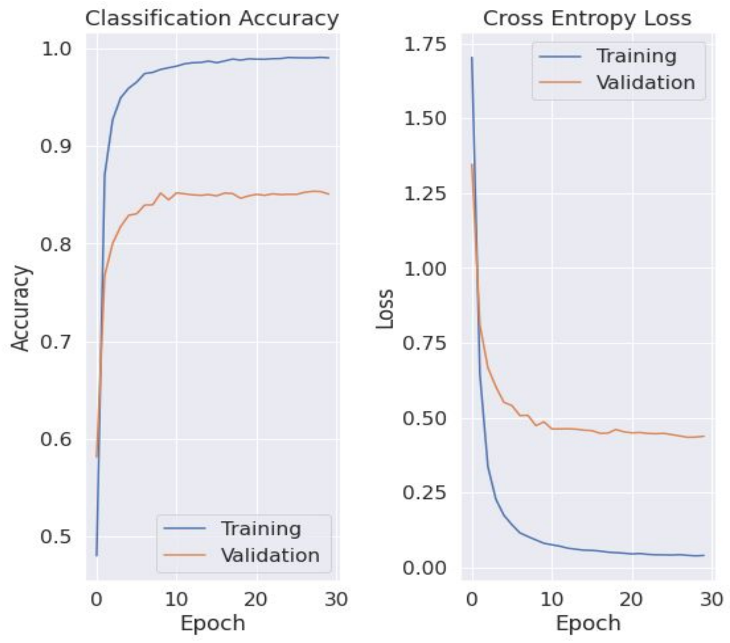
The ResNet50 model was the best model tested, based on test accuracy and class f1-scores. The confusion matrix for this model illustrates the difficulties the model had with classes c8 and c9. Class c8 had significant trouble with precision, with a large number of false positive predictions. Class c9 on the other hand, had more trouble with recall, with a large number of false negative predictions. Class c0 suffers from a number of false positives and false negatives, although not to the extent of classes c8 and c9.


Confusion Matrix

Actual \ Predicted	c0	c1	c2	c3	c4	c5	c6	c7	c8	c9
c0	353	1	0	15	1	0	0	0	42	58
c1	1	387	0	1	0	0	0	0	3	1
c2	0	0	398	0	0	0	1	0	14	0
c3	2	0	0	432	0	0	0	0	11	2
c4	2	0	6	8	365	2	0	0	53	6
c5	17	0	0	0	1	397	0	0	21	1
c6	0	0	0	0	1	0	322	0	113	0
c7	0	0	0	0	0	0	0	319	71	3
c8	8	0	7	0	0	0	0	3	318	8
c9	84	0	0	1	0	47	1	0	45	177

400
350
300
250
200
150
100
50
0

These training plots show the model loss and accuracy are nearly flat after 30 epochs, although the loss may have decreased additionally with longer training.



Further Work

Clearly, there are areas for improvement. The most obvious is the poor performance in classes c8, c9, and c0. As mentioned earlier, some of these images would cause challenges for human image classifiers.

It would be interesting to test whether preprocessing the data to segment heads and hands could improve performance. This could perhaps help models focus on the parts of the images that contain the most important data.

As the models performed differently on each class, it is possible an ensemble of models would perform better than a single model.

More experimentation in learning rate decay schedules could perhaps improve performance and reduce loss further.

It is possible that training for longer than 30 epochs could decrease loss in the ResNet50 model.

More work could be done on adjusting image augmentation parameters. Performance did not improve much with augmentation. Perhaps parameters were too high for some of the training images. I suspect the rotation parameter could be reduced.

Standardization of image data may help the scratch models as their input was just normalized by scaling. This would likely make the transfer learning models worse. All transfer learning input was processed with the custom pre-processing methods provided in Keras, which zero-center input with respect to the ImageNet dataset used in the initial training.