PREDICTING DISTRACTED DRIVING

Tracy A. Cardwell

tracy.a.cardwell@gmail.com

https://www.linkedin.com/in/tacardwell/

EXECUTIVE SUMMARY

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. Cell phone laws have reduced injuries and deaths slightly in the last few years, but non-cell phone distractions remain high.

Real-time distraction detection systems could provide feedback to drivers and prevent accidents and deaths. The first step is creating models that are capable of making fast and accurate predictions of distraction. The goal of this project is to predict distracted driving with deep learning models.

A variety of scratch-built and transfer learning models were trained on distracted driver and safe driver images created by State Farm.

The best model was based on the ResNet50 architecture and made predictions with an 84% overall accuracy, with performance varying by distraction type. Two distractions in particular were challenging for the model.

Contents

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. Five percent of all drivers and eight percent of drivers 15 to 19 years old involved in fatal crashes were reported as distracted at the time of the crash.

These estimates may be low as it is challenging to determine distraction in driver fatality crashes, and self-reported distractions are likely underreported. Survey research indicates self-reporting of negative behavior is lower than actual occurrence of that behavior.[1]

Distractions include any activity that takes one's attention away from driving and fall in 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.

Texting combines all three types of distractions, making it especially dangerous. Sending or reading a text requires you to take your eyes off the road for about 5 seconds. If you are going 55 miles per hour, this means you are driving the length of a football field without looking.[2]

A study conducted by Virginia Tech's Transportation Institute concluded that taking your eyes off the road for more than just two seconds doubles your risk of a crash.

A AAA Foundation for Traffic Safety report reviewing dozens of studies in 2008 concluded that any cell phone use, even hands-free, quadruples crash risk.

In a AAA Foundation survey in 2013, many drivers admitted using phones while driving despite acknowledging the danger.[3]

Many partial solutions have been adopted, including bans on texting and hand-held phone use, as well as do-not-disturb apps for phones. While these methods have slightly reduced accidents related to cell phone distractions, they are not enough. These measures cannot be completely effective as driver distractions include much more than cell phones. The problem is just larger and more visible now that cell phones are ubiquitous.[4, 5, 6]

Systems that detect distractions and report them to drivers in real-time could further reduce distracted driving accidents and deaths. Insurance companies could offer such systems to drivers in exchange for a reduction in premium. They already have similar arrangements for systems that monitor driving habits.

The first step in developing a distraction detection system is developing models that are fast and accurate in predicting distractions. This project aims to identify driver distractions from dashboard camera images.

DATA

The data analyzed in this project was downloaded from https://www.kaggle.com/c/state-farm-distracted-driver-detection. The data contains 10 classes of images as listed below.

- 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
- c9: talking to passenger

Note: The images in this dataset were created by State Farm in a controlled environment. Driving was simulated by a truck pulling a car while the “driver” in the car acted out distractions. These "drivers" weren't really driving, so no one was in danger.

The dataset includes 22,424 labeled training images and 79,726 unlabeled test images. Only the labeled images were used for this project, so that models could be evaluated and ranked.

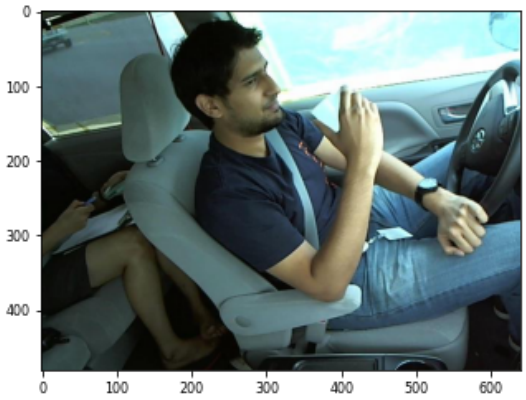
EXPLORATORY DATA ANALYSIS

The image attributes were first checked to determine size and mode distribution. All training images were 640 x 480 pixels, encoded in RGB mode.

Next, images from each class were sampled and pixel intensity histograms were plotted for each color channel. It was apparent from this sampling that many different drivers and cars were used to create the images. The results varied significantly, depending on the driver’s clothes, vehicle interior and seat colors, and the amount of light shining into the driver’s window.

The cameras in the vehicles appeared to be in slightly different locations. Some showed part of the dash and little to none of the back seat; whereas others showed none of the dash and a portion of the back seat.

There was a significant amount of extraneous data in many images. The images with the most data unrelated to drivers and distractions came from the cameras showing more of the backseat. Some images included space above the driver’s head, while others displayed the seat cushion or a console between the driver’s and passenger’s seat at the bottom of the frame.



The 22,424 training images were created with 26 drivers. The drivers were dressed the same in every image in which they appeared. Each driver was included in every class, although the drivers' images were not distributed evenly across the classes. In addition, the number of images per driver was not the same, ranging from 346 to 1237, or 1.5% to 5.5% of the total number of images.

The 10 classes are close to the same size in the training set, with each class containing between 8.5 and 11.1% of the total images.


```
Total number of images:  22424

Percentage of images per class:
classname
c0    11.099715
c1    10.109704
c2    10.332679
c3    10.462005
c4    10.372815
c5    10.310382
c6    10.368355
c7     8.927934
c8     8.522119
c9     9.494292
dtype: float64
```

DATA WRANGLING

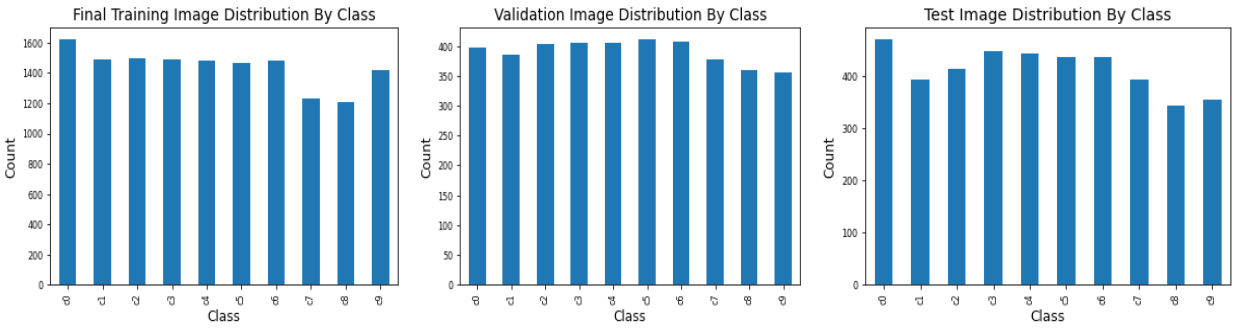
Train/Validate/Test Split

The images were split into three sets: training, validation, and testing. The sets were split by driver to avoid data leakage and ensure results would be generalizable to new drivers.

A pivot table was constructed with counts of images per class for each driver. This table was used with train_test_split() to split off first the test data, then the validation data. The goal was 60/20/20; however, the result was not exact due to splitting by driver and the uneven image counts per driver.

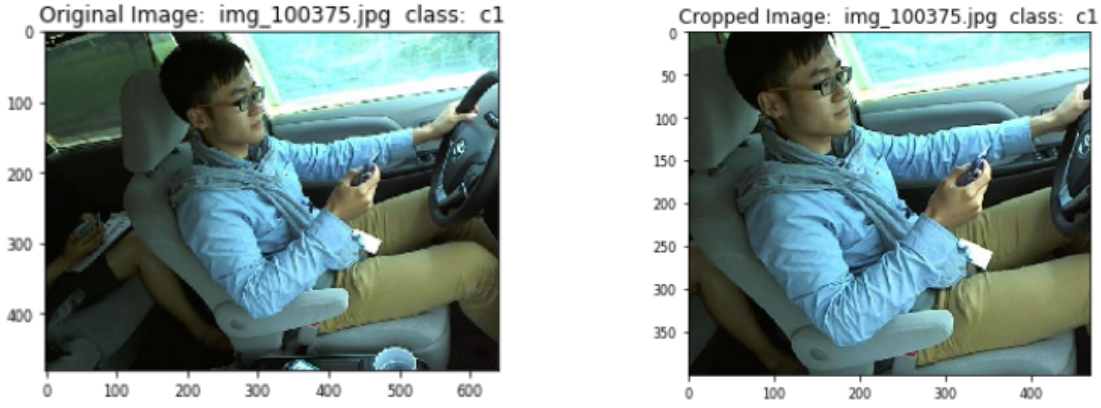
In the end, the split was 64.2/17.4/18.4, with the image count 14387/3907/4130. This is not many images for training, but a reasonable number were needed for hyperparameter tuning and for model evaluation.

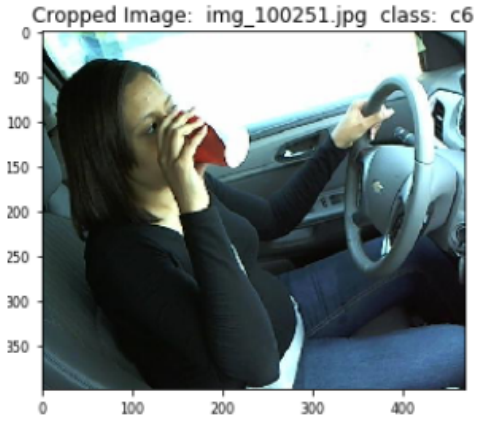
The distribution of each set of images across the classes was similar to the original.



Cropping

The images were cropped to eliminate as much extraneous data as possible while leaving the driver and distractions intact. Various cropping boxes were tested until one was identified that was a good compromise between cutting enough but not too much. The final crop took 40 pixels off the top, 40 pixels off the bottom, 120 pixels off the left, and 50 pixels off the right of each image. This reduced the images from 640 x 480 to 470 x 400 pixels.





Once the images were cropped and the train/validation/test sets were defined, the image directory structure for deep learning was built as illustrated below.

```
├── proc      <- The final, canonical data sets for modeling.
│   ├── imgs          <- image data
│   │   ├── train             <- Training images
│   │   │   ├── c0            <- c0 class
│   │   │   ├── c1            <- c1 class
│   │   │   ├── c2            <- c2 class
│   │   │   ├── c3            <- c3 class
│   │   │   ├── c4            <- c4 class
│   │   │   ├── c5            <- c5 class
│   │   │   ├── c6            <- c6 class
│   │   │   ├── c7            <- c7 class
│   │   │   ├── c8            <- c8 class
│   │   │   └── c9            <- c9 class
│   │   ├── valid             <- Validation images
│   │   │   ├── c0            <- c0 class
│   │   │   ├── c1            <- c1 class
│   │   │   ├── c2            <- c2 class
│   │   │   ├── c3            <- c3 class
│   │   │   ├── c4            <- c4 class
│   │   │   ├── c5            <- c5 class
│   │   │   ├── c6            <- c6 class
│   │   │   ├── c7            <- c7 class
│   │   │   ├── c8            <- c8 class
│   │   │   └── c9            <- c9 class
```

```
|   |   ├── test    <- Testing images
|   |   |   ├── c0          <- c0 class
|   |   |   ├── c1          <- c1 class
|   |   |   ├── c2          <- c2 class
|   |   |   ├── c3          <- c3 class
|   |   |   ├── c4          <- c4 class
|   |   |   ├── c5          <- c5 class
|   |   |   ├── c6          <- c6 class
|   |   |   ├── c7          <- c7 class
|   |   |   ├── c8          <- c8 class
|   |   |   ├── c9          <- c9 class
```

After the images were placed according to the above diagram, the data structure was processed into a zip file for transporting between platforms.

Augmentation

Image augmentation was used to improve results due to the small number of training images, and to assist with generalization of models.

Augmentation was done with Keras ImageDataGenerator(). Various options were visualized to determine the largest parameter values that could be used without drivers and distractions being cut out of the images.

The augmentation options used were:

- zoom = 0.2,
- shear_range = 20,
- width_shift = 0.2,
- height_shift = 0.1
- rotation_range = 30, and
- fill_mode = ‘nearest’.

Flips were not used due to right/left-handed distractions being different classes. Below are samples of the augmented 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

Images were downsized as the original size 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.

Flow_from_directory() was used to feed image files 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 could 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, 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 more 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 an even lower learning rate for fine-tuning would perform better.

Changing the output to a simple softmax layer did not decrease the performance of the model, so the complex 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 in 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 tests. This experiment led to an increase in loss and drop in accuracy so was not repeated.

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 were 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

Inception V3

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 more trouble with class c9. 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 most 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

CONCLUSION

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.

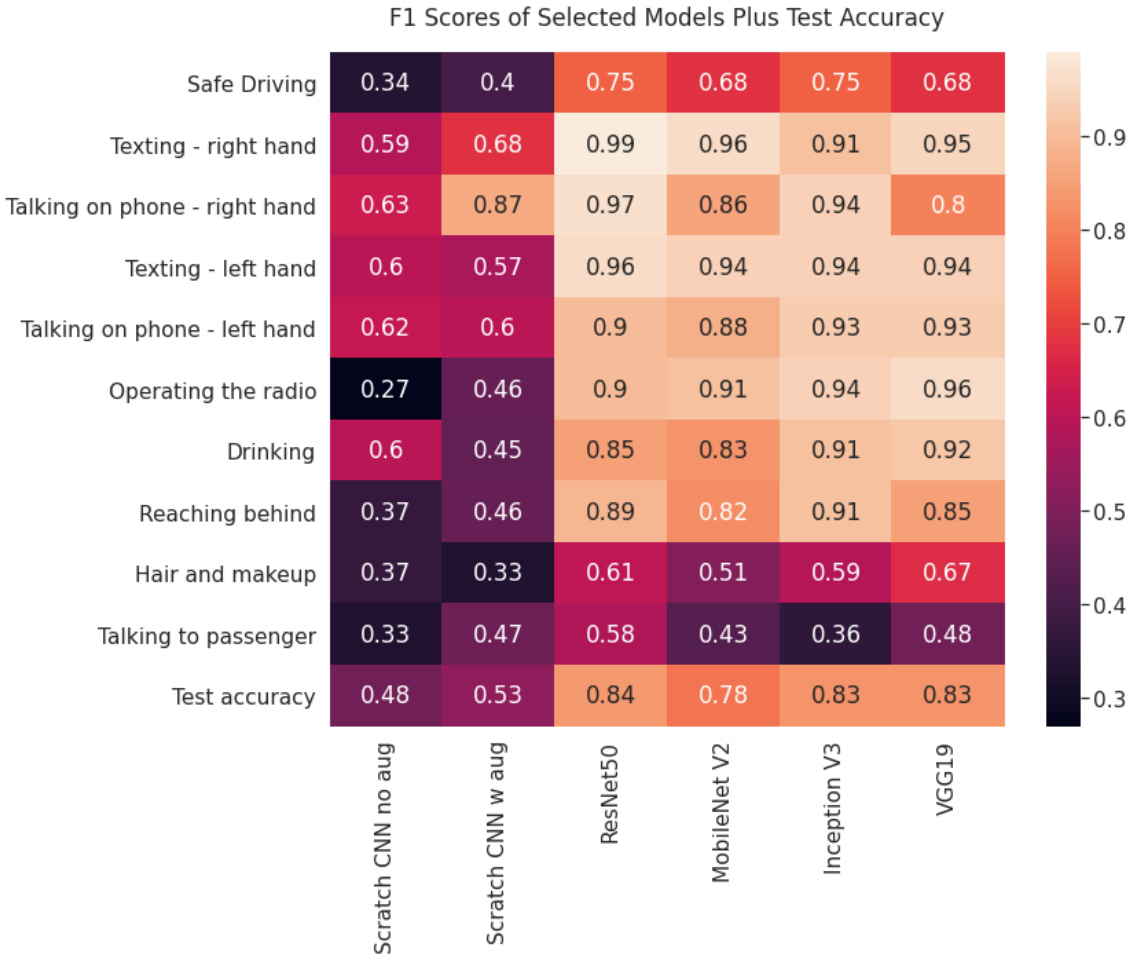
Reviewing the images, it is easy to see that Talking to Passenger often looks like Safe Driving. Most images in this class show the driver's head turned to look at the Passenger but some show the driver looking straight ahead. Most images show both hands on the steering wheel as in Safe Driving. Some 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 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.



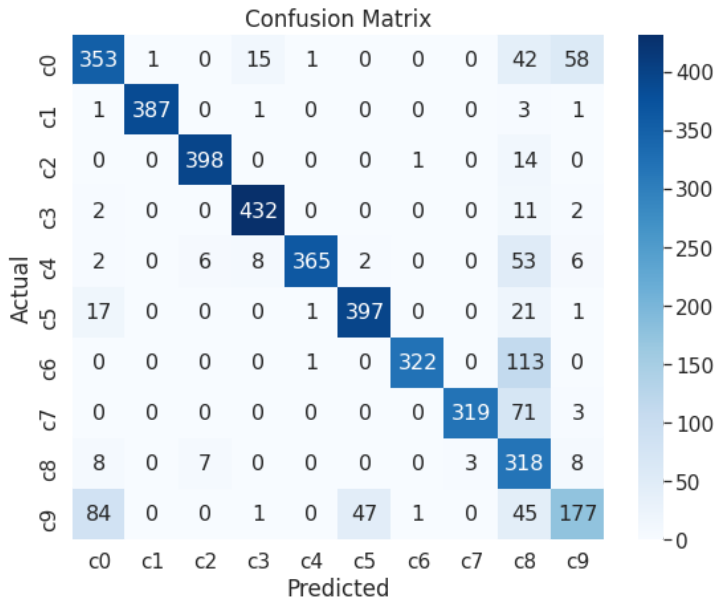
This clearly illustrates that class c9, talking to passenger gives models the most trouble, followed by class c8, hair and makeup, and class c0, safe driving.

2. Transfer Learning was key to good performance

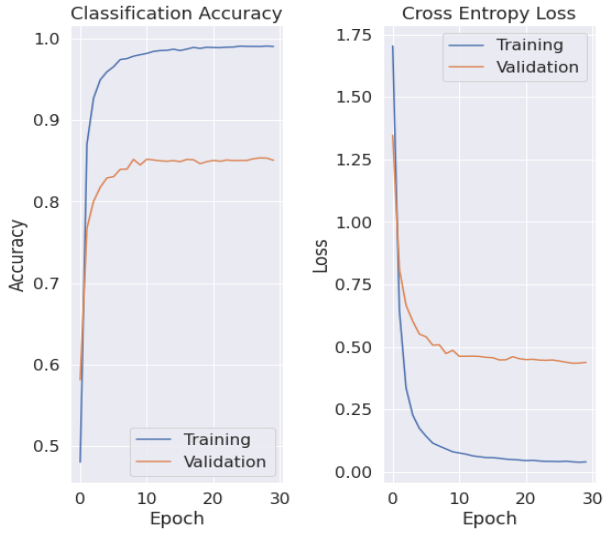
The above heatmap also shows a huge difference in performance between the scratch-built models and the pre-trained models. This classification problem benefited greatly from transfer learning.

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.



These training plots show the model loss and accuracy were nearly flat after 30 epochs, although the loss was slowly dropping and 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 the head and hands could improve performance. This could perhaps help models focus on areas 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.

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

Perhaps the cropping was too much for some of the images. It would be interesting to see if performance would improve with a smaller crop. It is certain that cropping is an improvement over no cropping, however, as performance was much worse before cropping the images.

More work could be done on adjusting image augmentation. As data augmentation is a type of regularization, it was expected augmentation would decrease loss. That did not happen. As a result, it is possible the cropping combined with augmentation pushed important data out of the images in too many cases. It would be interesting to see how the models would perform with smaller augmentation parameters.

Standardization of image data may help the scratch models as their input was just normalized by scaling. This would likely not improve transfer learning models, however. All transfer learning models used the custom pre-processing methods provided in Keras, which zero-center input with respect to the ImageNet dataset used in the initial training.

REFERENCES

1. National Center for Statistics and Analysis. (2020, April). Distracted driving 2018 (Research Note. Report No. DOT HS 812 926). National Highway Traffic Safety Administration. Retrieved from: https://crashstats.nhtsa.dot.gov/Api/Public/Publication/812926
2. Distracted Driving. (2019, Sep 16) Retrieved from: https://www.cdc.gov/motorvehiclesafety/distracted_driving/index.html
3. The Risks of Distracted Driving. (2017) Retrieved from: https://exchange.aaa.com/safety/distracted-driving/the-risks-of-distracted-driving/#.Xsm4jGhKiUk
4. Distracted Driving. (2020, Feb) Retrieved from: https://www.iihs.org/topics/distracted-driving
5. Distracted Driving. (n.d.) Retrieved from: https://www.ghsa.org/state-laws/issues/distracted%20driving
6. Distracted Driving. (2019) Retrieved from: https://injuryfacts.nsc.org/motor-vehicle/motor-vehicle-safety-issues/distracted-driving/