Task is Simple….

- Build a classification model to correctly identify what the driver is doing while driving the car.
- The driver could be:
- Insert the table related to class labels

But Challenging…

- Technical Challenges
  - A particular subject can only appear either in the training or the testing dataset
- Resource Challenges
  - CNN model with millions of trainable parameters on a local machine was resource intensive and was taking hours to finish the model

Our Solution….

- Technical Challenge
  - Use a pre-trained model – VGG16, VGG19, EfficientNetB0
  - Image Augmentation to add more variations to our dataset to make the model more generalizable
- Resource Challenge
  - Used Google CoLab to run our models

Modeling Workflow

- Establishing a baseline score with a 2-layer CNN model
- Run iterations using the three pre-trained models to improve the log-loss on the testing dataset
- Parameters varied and investigated:
  - Number of nodes in the FNN
  - Number of hidden layers in the FNN
  - Regularization- dropout/l2
  - Learning rate for the ADAM optimizer
- For all the models, the best model was saved using a model checkpoint option in Keras

Pre-Trained Models

- What is a Pre-trained Model?
  - A pre-trained model is a model created by some one else to solve a similar problem. We use this model as a starting point.
- Why use Pre-trained Models?
  - To “transfer” the learning from the pre-trained model to our model to avoid having to train our model from scratch
- How to use a Pre-trained Models?
  - Define the objective of our problem
  - Use a pre-trained model which was trained for a similar problem. For e.g. Image classification pre-trained model for an Image classification problem

Pre-Trained Models and Transfer Learning

- Using a pre-trained model to our specific problem is called Transfer Learning
- Use of pre-trained models takes 3 different forms:
  - Feature Extraction:- Directly use the weights and architecture and apply that learning to our problem. The only modification that would be required in this case is changing the output layer depending on our class labels
  - Transfer Learning:- Use the weights and architecture of the convolutional layers and train the user-defined FNN layers
  - Fine-tuning:- Tweak the already trained convolutional layers and train the model

Our Approach

- Transfer Learning using Pre-trained model
  - Since we did not a large dataset to train on, we decided not to train the convolutional layers and set this layers to not-trainable in our model
  - Set up our own FNN architecture connected to the pre-trained convolutional architecture and train the datasets on this model

Modeling Results

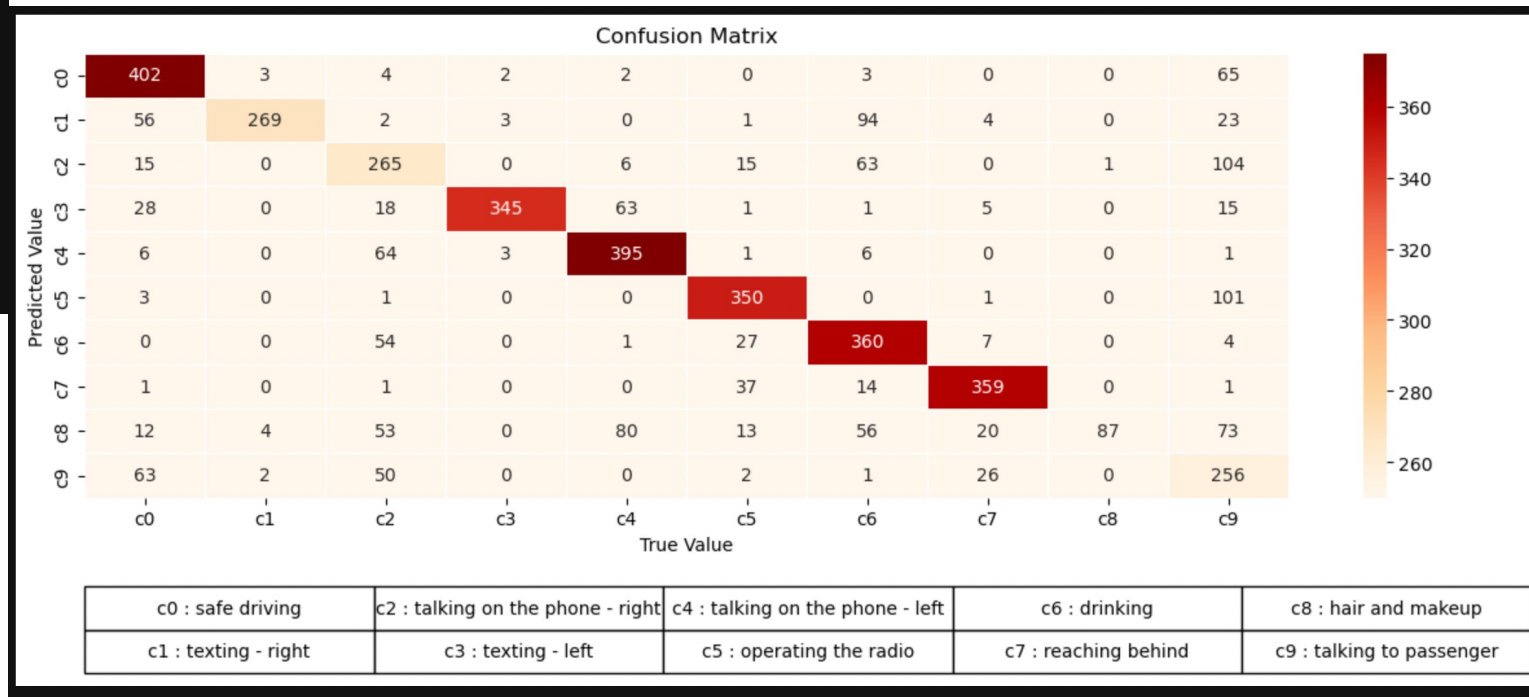
	Model	Training Loss	Training Accuracy	Validation Loss	Validation Accuracy	Kaggle Testing Loss	Model Type
0	base_model_kfold	0.000	1.000	0.005	0.999	26.643	Base-Kfold
1	base_model_gkfold	0.746	0.855	3.525	0.315	23.763	Base-Gkfold
2	best_model_vgg16_dropout_1024	0.193	0.937	0.929	0.690	0.731	VGG16
3	best_model_vgg16	0.181	0.944	0.891	0.718	0.752	VGG16
4	best_model_vgg19_l2_4096_1024_adam_lr	0.512	0.943	1.181	0.719	0.806	VGG19
5	best_model_vgg16_l2_1024	0.433	0.926	1.324	0.645	0.837	VGG16
6	best_model_vgg16_l2_4096_1024_adam_lr	0.574	0.944	1.280	0.720	0.850	VGG16
7	best_model_EfficientNetB0_v2_base	0.090	0.984	0.949	0.686	0.857	EfficientNetB0
8	best_model_EfficientNetB0_v2_augment	0.205	0.952	0.983	0.693	0.878	EfficientNetB0

- 1st model is the base model with usual train_test_split gave accuracy of 1 for both training and validation but a testing loss of 26.64
- 2nd model is the base model with Group Kfold train test split reduced the validation accuracy to 30% and improved the testing loss to 23.7
- Our best model performance with pre-trained model was for VGG16 model with a single hidden layer of 1024 nodes. The testing loss was 0.73

Classification Report and Confusion Matrix

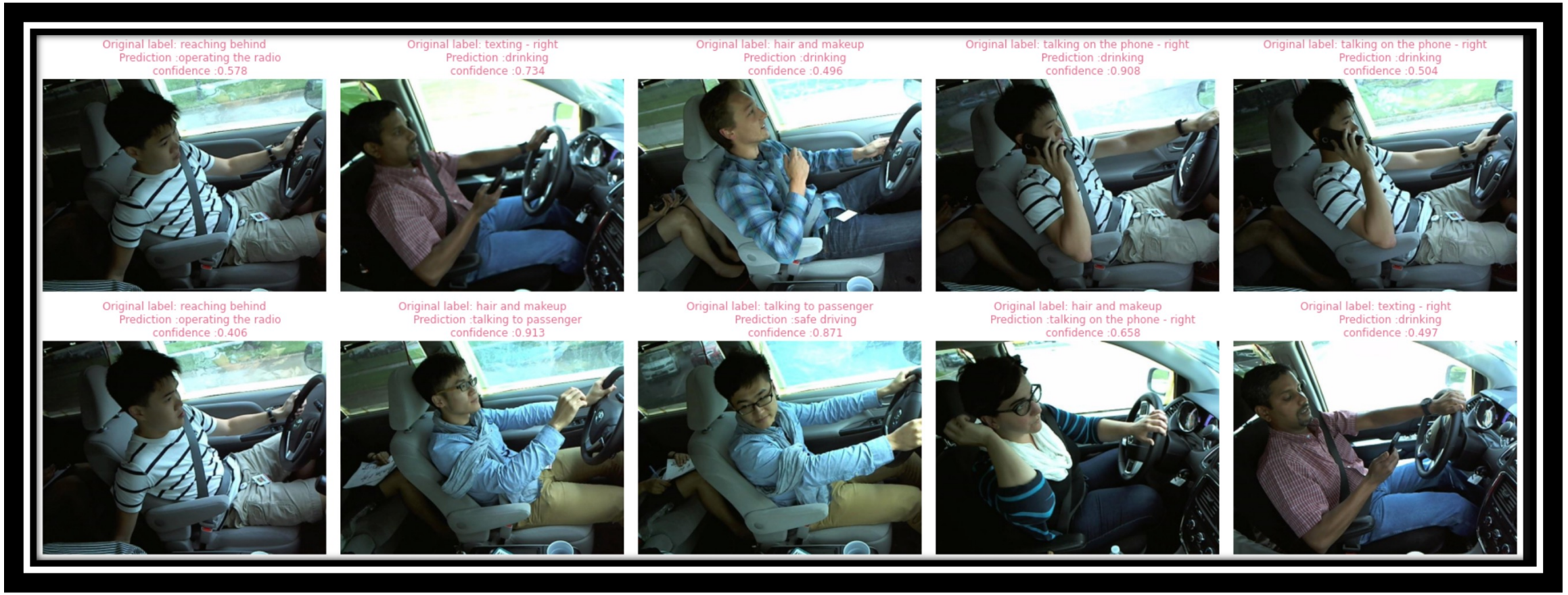
	precision	recall	f1-score	support
c0	0.69	0.84	0.75	481
c1	0.97	0.60	0.74	452
c2	0.52	0.57	0.54	469
c3	0.98	0.72	0.83	476
c4	0.72	0.83	0.77	476
c5	0.78	0.77	0.78	456
c6	0.60	0.79	0.69	453
c7	0.85	0.87	0.86	413
c8	0.99	0.22	0.36	398
c9	0.40	0.64	0.49	400
accuracy			0.69	4474
macro avg	0.75	0.68	0.68	4474
weighted avg	0.75	0.69	0.69	4474

Class labels c8 and c9 have the lowest f1-score and c3 and c7 have the highest



Mislabeled Prediction Analysis

Mislabeled Prediction Analysis



Conclusions

- Use of pre-trained models significantly improved the predictions of the CNN model
- A CNN model using a VGG16 pre-trained model gave testing loss on Kaggle of 0.73