A *Real World Data Approach* to the Distracted Drivers Detection Problem
AML Final Project Report

Niranjan Rajesh, Sristi Bafna
CS-2490-1: Advanced Machine Learning
Dr. Lipika Dey, Dr. Tirthankar Dasgupta
12 May 2023

Motivation:

According to the CDC motor vehicle safety division, one in five car accidents is caused by a distracted driver. Sadly, this translates to 425,000 people injured and 3,000 people killed by distracted driving every year. Distracted driving is a widespread issue that diverts the driver's attention from the road and increases the risk of accidents. Research has identified three types of distractions: visual, manual, and cognitive. Laws have been enacted in many states to prohibit activities such as texting and using a cell phone while driving. However, these laws are not enough to prevent distracted driving incidents. To address this issue, we **propose leveraging computer vision technology to detect distracted activities of drivers and alert them.** We envision this technology being integrated into vehicles to prevent accidents caused by distracted driving, thereby improving road safety.

Another motivation is to explore the robustness of developed architectures in the face of realistic augmentations to the problem set. How well can we build models that are able to handle **real world variance in data** that was not present in its training set? We test this through augmentations.

Dataset:

State Farm Distracted Driver Detection | Kaggle

The dataset contains 3 files:

- **imgs.zip** - zipped folder of all (train/test) images
- **sample_submission.csv** - a sample submission file in the correct format
- **driver_imgs_list.csv** - a list of training images, their subject (driver) id, and class id

driver_imgs_list.csv contains 22,424 entries. The 10 classes to predict are:

- c0: safe driving
- c1: texting - right

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

Problem Statement/Research Questions:

1. Can we develop a vision architecture that is able to compete with state of the art architectures for a scenario that has real world applications like detection of distracted drivers?
2. Is the model able to generalize to variance in images that are representative of real word distortions modeled through augmentations? How does it compare with state of the art models?

Data Exploration

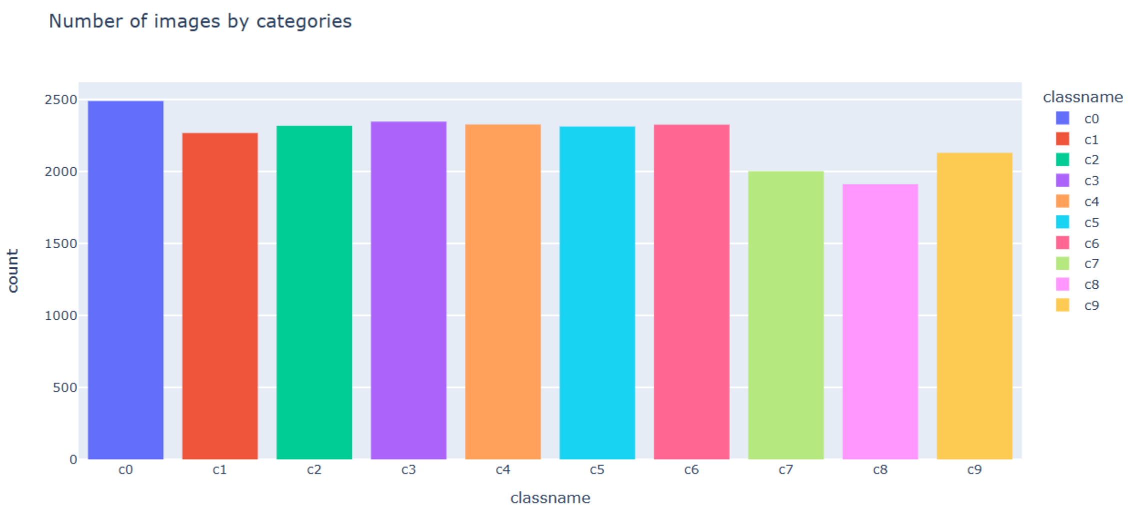
Class Distribution

The class distribution in the dataset is as follows:

- **c0**: safe driving - **Count**: 2489
- **c1**: texting (right) - **Count**: 2267
- **c2**: talking on the phone (right) - **Count**: 2317
- **c3**: texting (left) - **Count**: 2346
- **c4**: talking on the phone (left) - **Count**: 2326
- **c5**: operating the radio - **Count**: 2312
- **c6**: drinking - **Count**: 2325
- **c7**: reaching behind - **Count**: 2002
- **c8**: hair and makeup - **Count**: 1911
- **c9**: talking to passenger - **Count**: 2129

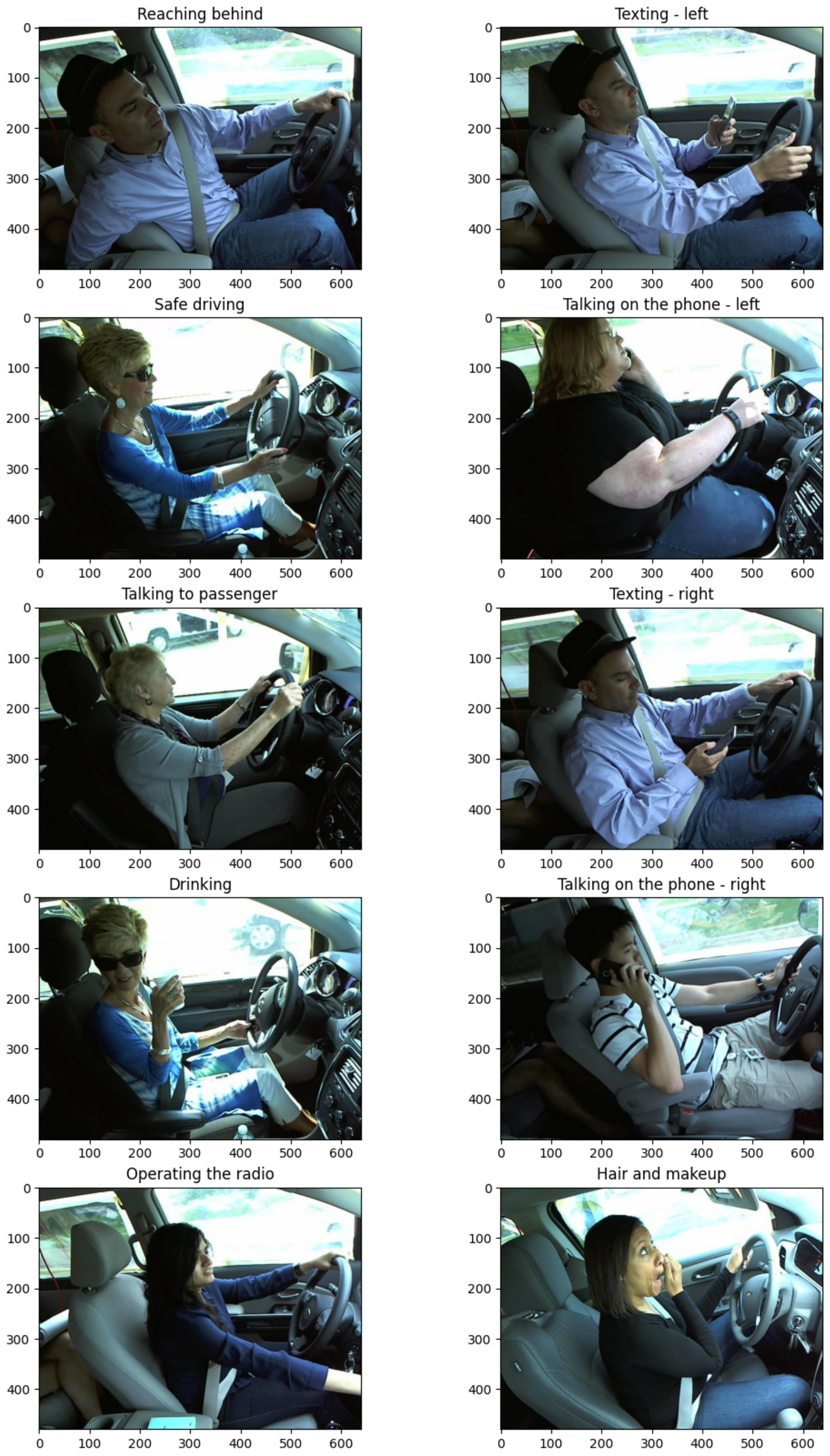
Total number of data points/images = **22,424**

As we can see, there is a balanced class distribution across the dataset.



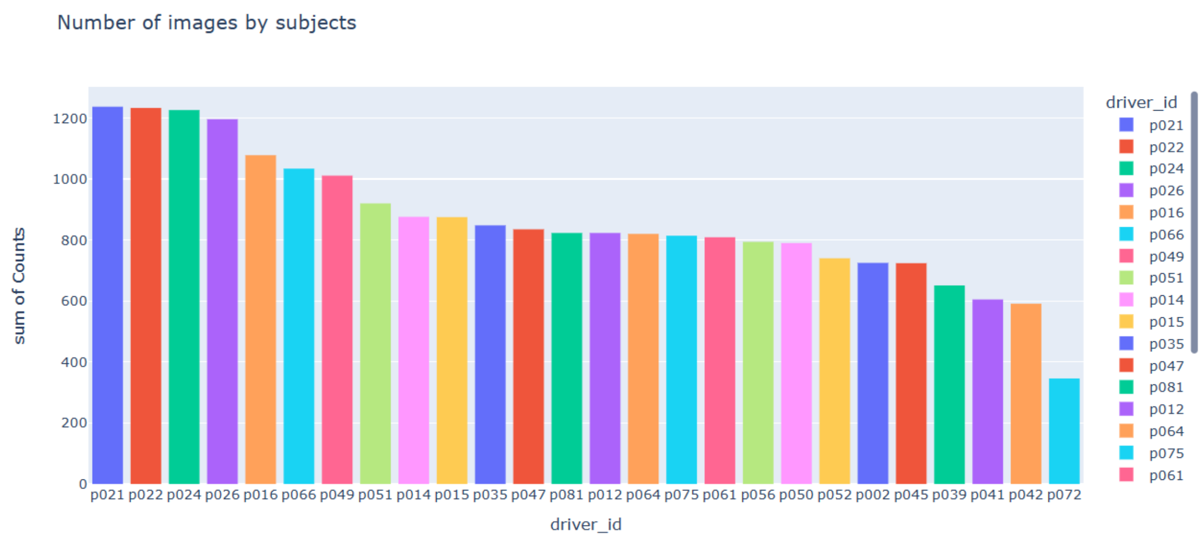
<u>Visualizing images from each class</u>

Sample images from each of the 10 classes are displayed below:

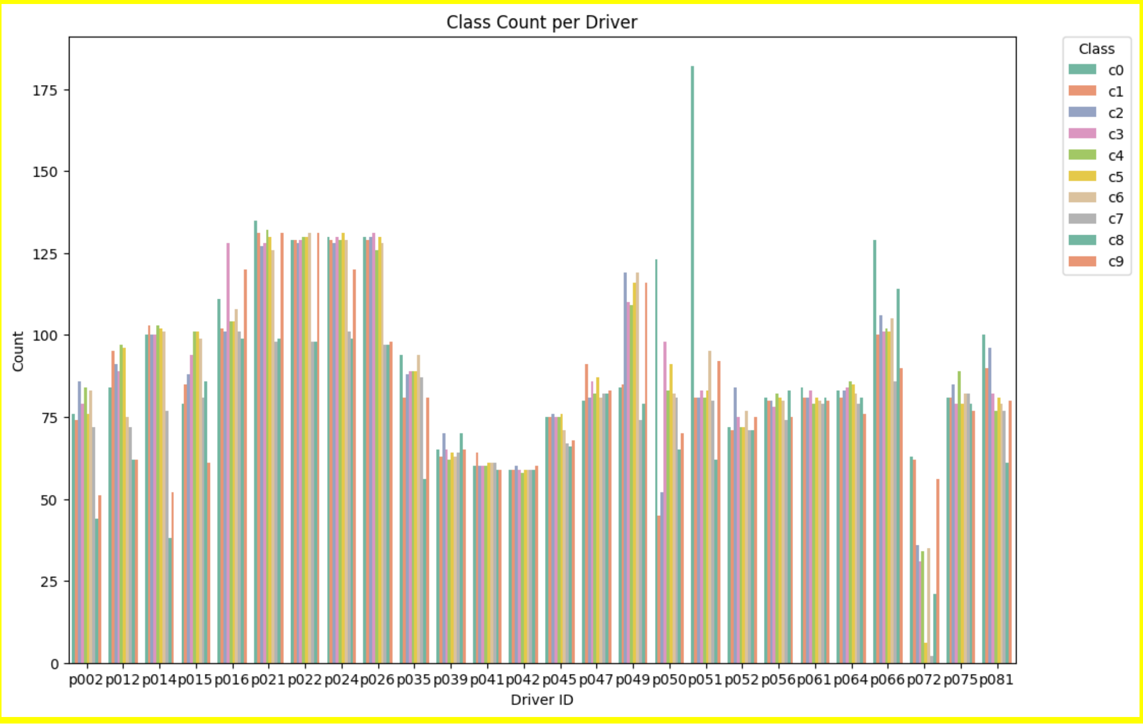

Reaching behind
Texting - left
Safe driving
Talking on the phone - left
Talking to passenger
Texting - right
Drinking
Talking on the phone - right
Operating the radio
Hair and makeup
0
100
200
300
400
500
600

Distribution of Data per subject

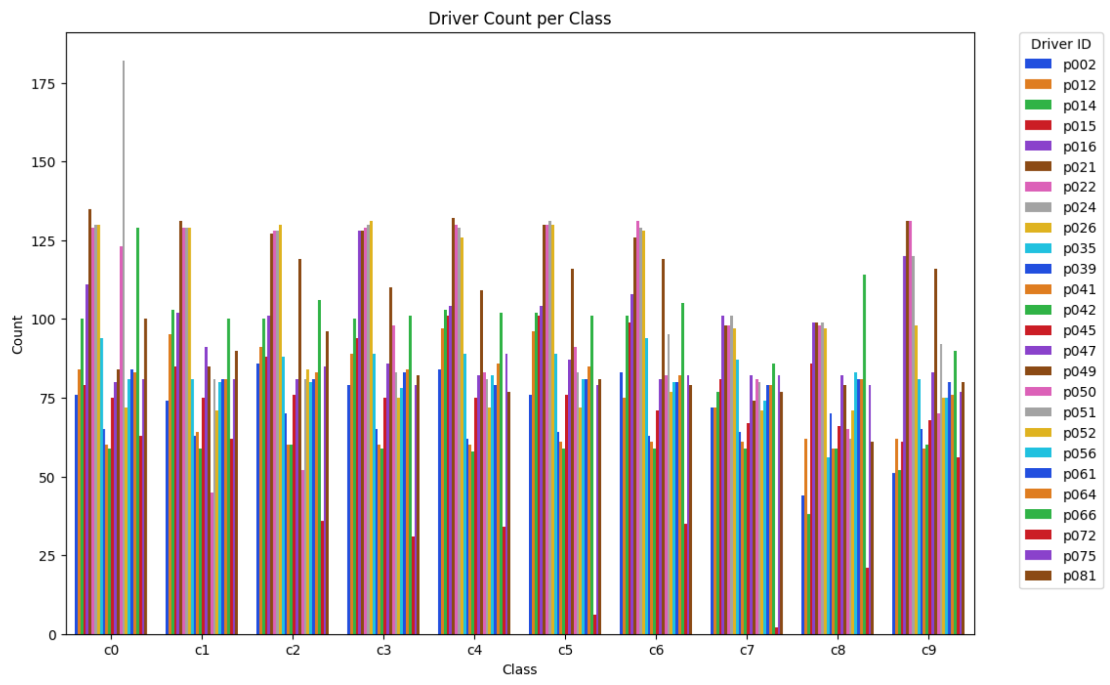
The images in the dataset were stills from video clips captured from the dashboard or rear view cameras of 26 people. A visualization of the number of images by each of the subjects is shown below:



Class Distribution per Subject:



Distribution of Data per class



We see that most classes are well distributed in terms of the drivers present in the class. This was a check to investigate whether some participants may have introduced bias by being overrepresented in some of the classes.

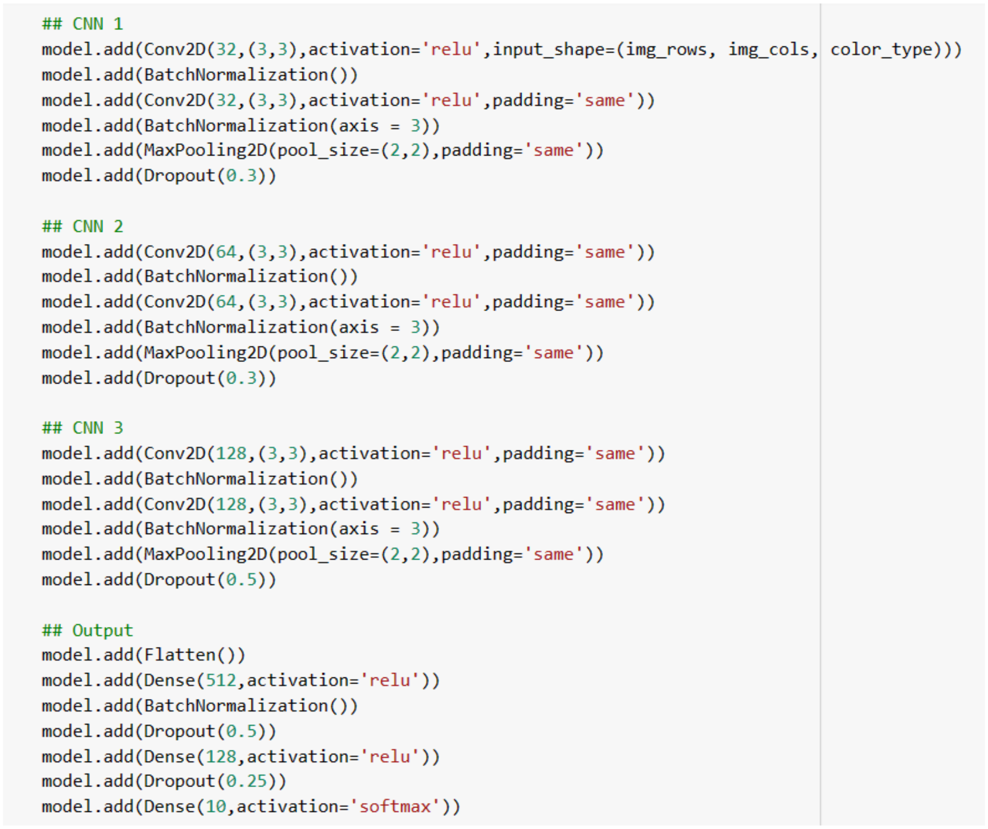
Models Chosen:

We chose 4 models for our task:

- Vanilla - Custom CNN
- MobileNet18

Vanilla - Custom CNN

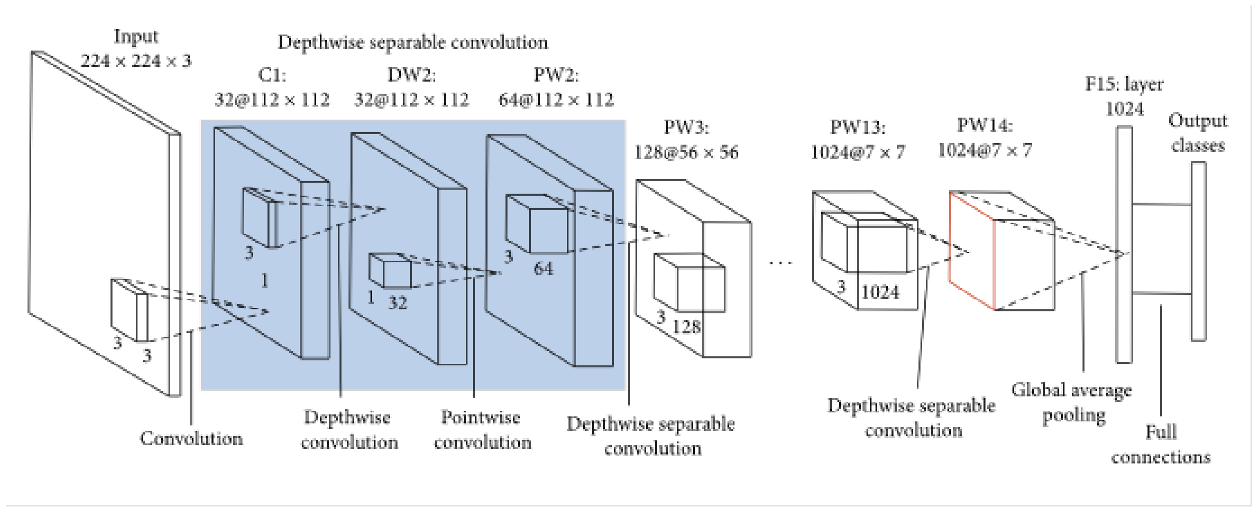
The CNN architecture was as follows:



```
## CNN 1
model.add(Conv2D(32,(3,3),activation='relu',input_shape=(img_rows, img_cols, color_type)))
model.add(BatchNormalization())
model.add(Conv2D(32,(3,3),activation='relu',padding='same'))
model.add(BatchNormalization(axis = 3))
model.add(MaxPooling2D(pool_size=(2,2),padding='same'))
model.add(Dropout(0.3))

## CNN 2
model.add(Conv2D(64,(3,3),activation='relu',padding='same'))
model.add(BatchNormalization())
model.add(Conv2D(64,(3,3),activation='relu',padding='same'))
model.add(BatchNormalization(axis = 3))
model.add(MaxPooling2D(pool_size=(2,2),padding='same'))
model.add(Dropout(0.3))

## CNN 3
model.add(Conv2D(128,(3,3),activation='relu',padding='same'))
model.add(BatchNormalization())
model.add(Conv2D(128,(3,3),activation='relu',padding='same'))
model.add(BatchNormalization(axis = 3))
model.add(MaxPooling2D(pool_size=(2,2),padding='same'))
model.add(Dropout(0.5))

## Output
model.add(Flatten())
model.add(Dense(512,activation='relu'))
model.add(BatchNormalization())
model.add(Dropout(0.5))
model.add(Dense(128,activation='relu'))
model.add(Dropout(0.25))
model.add(Dense(10,activation='softmax'))
```

- **Approach used**
  - We made a custom CNN with 3 blocks, each containing 2 convolutional layers, batch normalization, max pooling and dropout.
  - We trained this on our dataset for 20 epochs and batch size 40.
  - Baseline experiment - The custom CNN was tested on the validation dataset
  - Project experiment - The custom CNN was tested on the real-world augmented dataset.
- **Benefits**
  - The custom CNN is relatively simple and has high feature extraction and regularization properties. This enabled us to test its ability to generalize well on the real world datasets

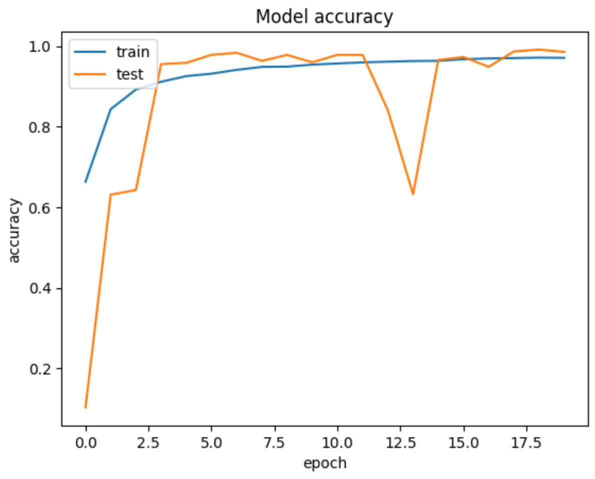
MobileNet

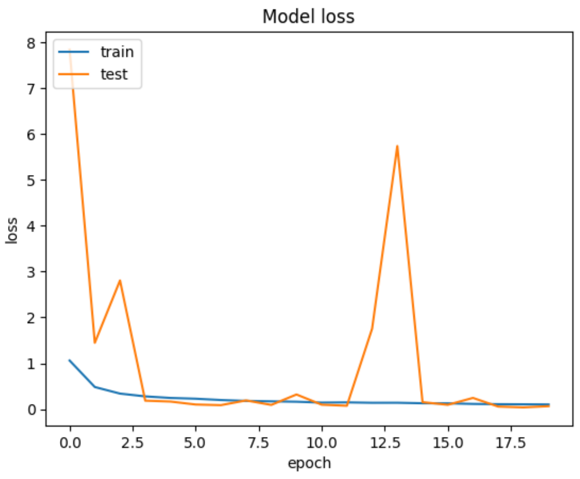


- **Approach used**
  - MobileNet V1 was trained from scratch on the same dataset without any pre training
- **Benefits**
  - MobileNet was chosen as a counterpart due to its highly simple and lightweight architecture that allows us to use it on large datasets like these at very little computational cost.

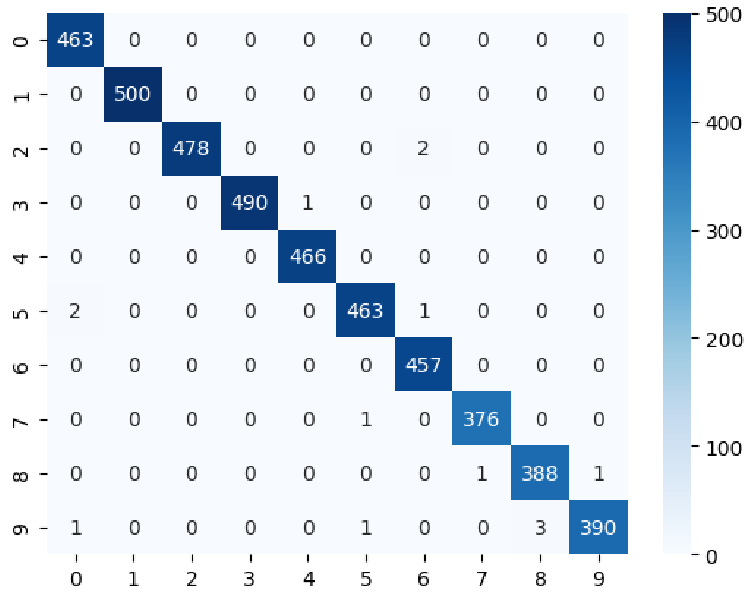
Results Evaluation

Vanilla - Custom CNN



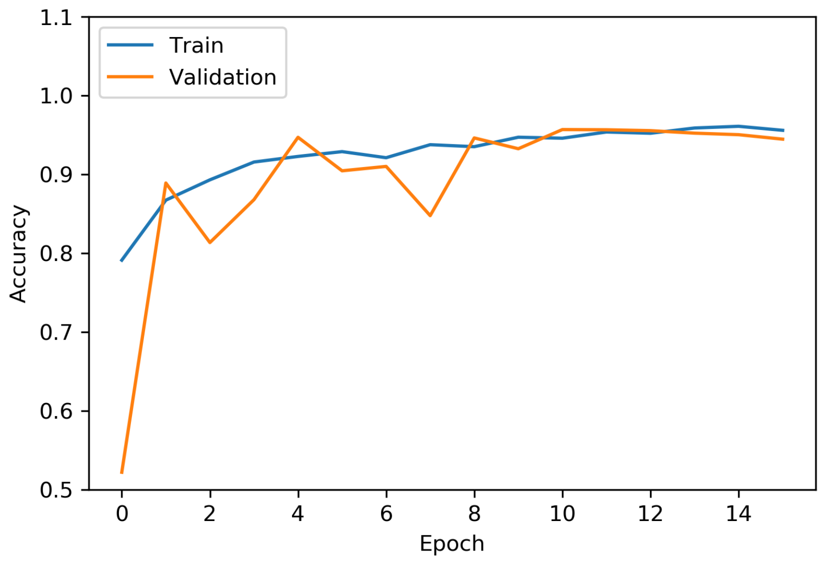


The custom CNN performs extremely well on the provided Distracted Drivers Dataset. As we can see, the model's accuracy and loss fluctuates heavily around the 13th epoch but the network converges again. The model got an accuracy of 99.69%. The confusion matrix for the same is shown below:



	0	1	2	3	4	5	6	7	8	9
0	463	0	0	0	0	0	0	0	0	0
1	0	500	0	0	0	0	0	0	0	0
2	0	0	478	0	0	0	2	0	0	0
3	0	0	0	490	1	0	0	0	0	0
4	0	0	0	0	466	0	0	0	0	0
5	2	0	0	0	0	463	1	0	0	0
6	0	0	0	0	0	0	457	0	0	0
7	0	0	0	0	0	1	0	376	0	0
8	0	0	0	0	0	0	0	1	388	1
9	1	0	0	0	0	1	0	0	3	390

<u>MobileNet</u>

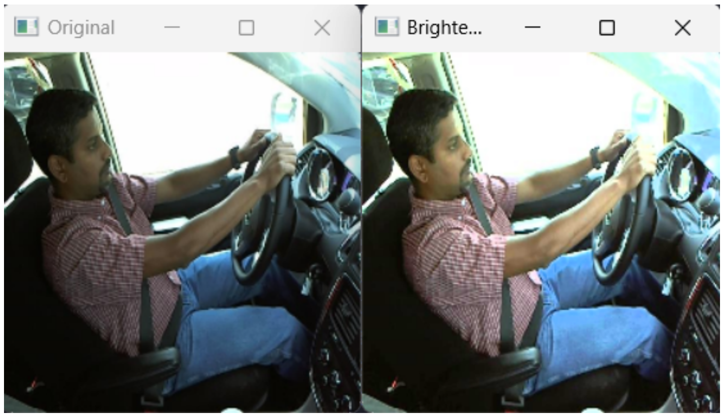


It is clearly visible that MobileNet achieved marginally lower validation accuracy although it seems to converge at a faster rate. This is quite predictable from the architecture as it was meant to be efficient on mobile devices with fewer parameters. This complexity difference between the vanilla model and the MobileNet model could be the reason for the slight difference in performance. Now we move on to analyzing their respective robustness to real-world variance through augmentations.

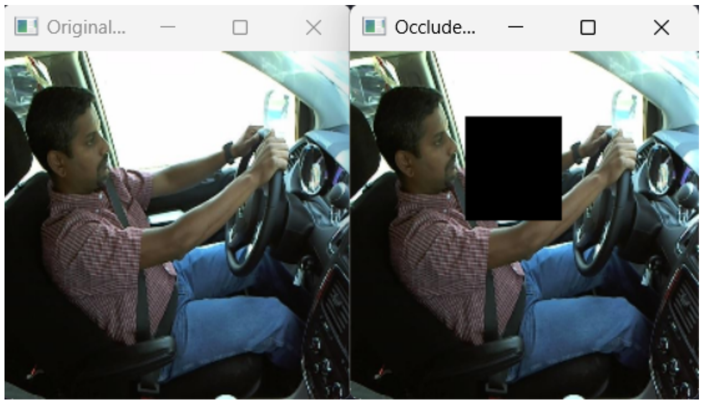
Augmentation Robustness

Account for real world variance in data found in images captured by camera dashboard

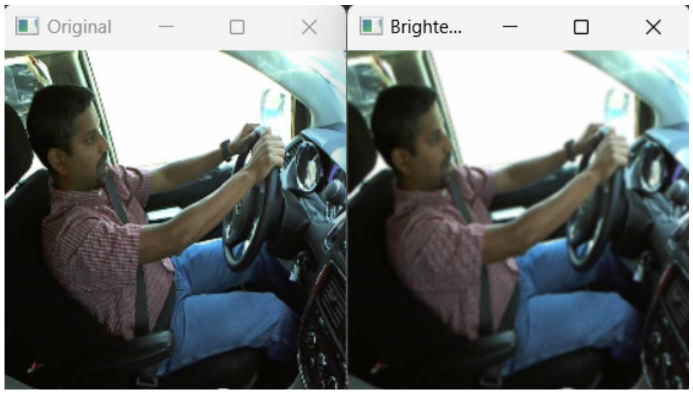
1. Lighting (Tunnels. Cloudy) - Random Brightness adjustment



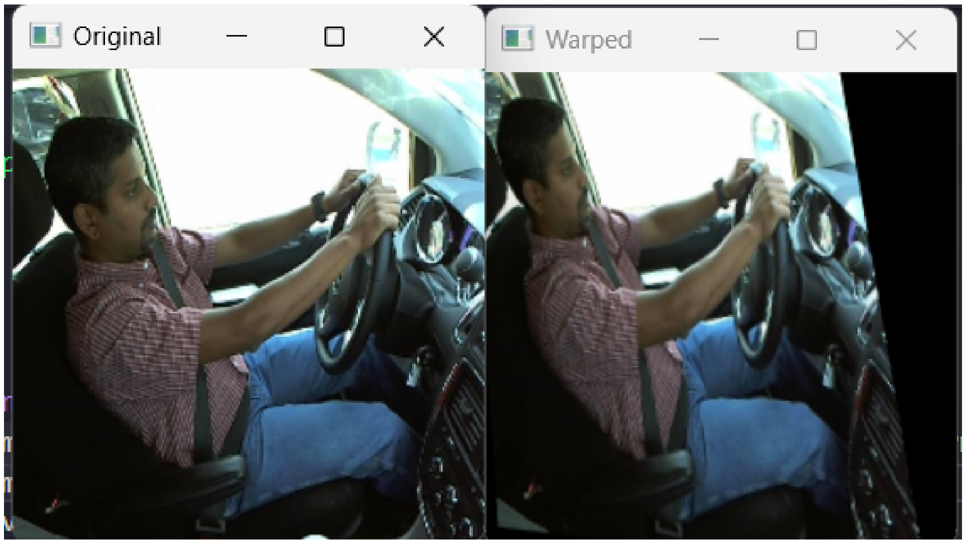
2. Obstructions - Random Occlusion



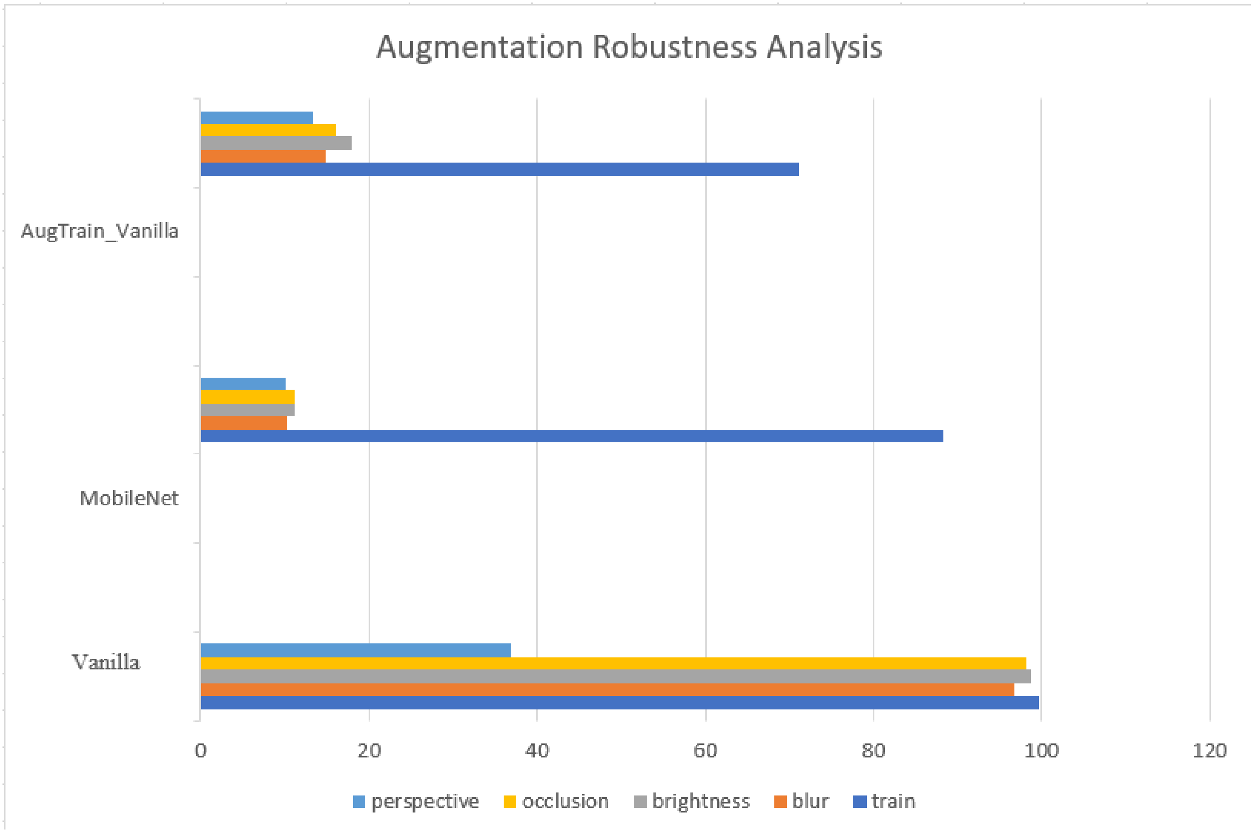
3. Blur (Camera focus) - Random Gaussian Blur



4. Camera Angle - Random Perspective Adjustment



How well do our models generalize to these augmented datasets?



For the robustness analysis, three models were chosen and their performances were contrasted. The Vanilla model was trained on the main dataset and achieved a near perfect validation accuracy during training as indicated by the blue bar. The model was then tested on datasets which introduced variance through these real-world augmentations. The model performed surprisingly well compared to its counterparts on each of the augmented datasets. Again, it is important to note that none of the models were exposed to any augmentations in the training data (except for the AugTrain_Vanilla model)

MobileNet which scored around 85% in validation accuracy during training with the original dataset failed to generalize its learnings to any augmented datasets and scored poorly in them

A third model was introduced to this experiment - AugTrain_Vanilla which was the same architecture as the Vanilla Custom CNN model but now was only trained on the real dataset that had undergone these augmentations in a systematic manner. Note that there were no unaugmented images in this dataset and all of it was made up of augmentations. This model, surprisingly, only performed marginally better when compared to MobileNet on the augmented datasets.

The results suggest that the vanilla model that was trained on all the images from the real dataset was able to learn rich representations that allowed it to generalize well to the augmented datasets and proved to be quite robust in the context of augmentations. This suggests that the greater complexity in the vanilla model allowed it to capture richer representations in the data that was able to be generalized to augmentations that simulate real world scenarios.

Code Repositories:

CNN -

1. https://colab.research.google.com/drive/1epNacJCNmmp6h3tAlAP90rizZ0MSUw7h?usp=sharing
2. https://colab.research.google.com/drive/1epNacJCNmmp6h3tAlAP90rizZ0MSUw7h?usp=sharing

MobileNet -
https://colab.research.google.com/drive/1j93QsDsIlAPzJrgHKAJgs6Ubf2jdBxIz?authuser=1#scrollTo=RPEoVFVKTqs4&uniqifier=1

Augmentations and Evaluations -
https://github.com/niranjanrajesh02/AML_Distracted_Drivers