Distracted Driver Detection – Milestone Report

Problem Statement:

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.

State Farm hopes to improve these alarming statistics, and better insure their customers, by testing whether dashboard cameras can automatically detect drivers engaging in distracted behaviors. Given a dataset of 2D dashboard camera images, State Farm is challenging Kagglers to classify each driver's behavior. Are they driving attentively, wearing their seatbelt, or taking a selfie with their friends in the backseat?

Dataset contains driver images, each taken in a car with a driver doing something in the car (texting, eating, talking on the phone, makeup, reaching behind, etc).

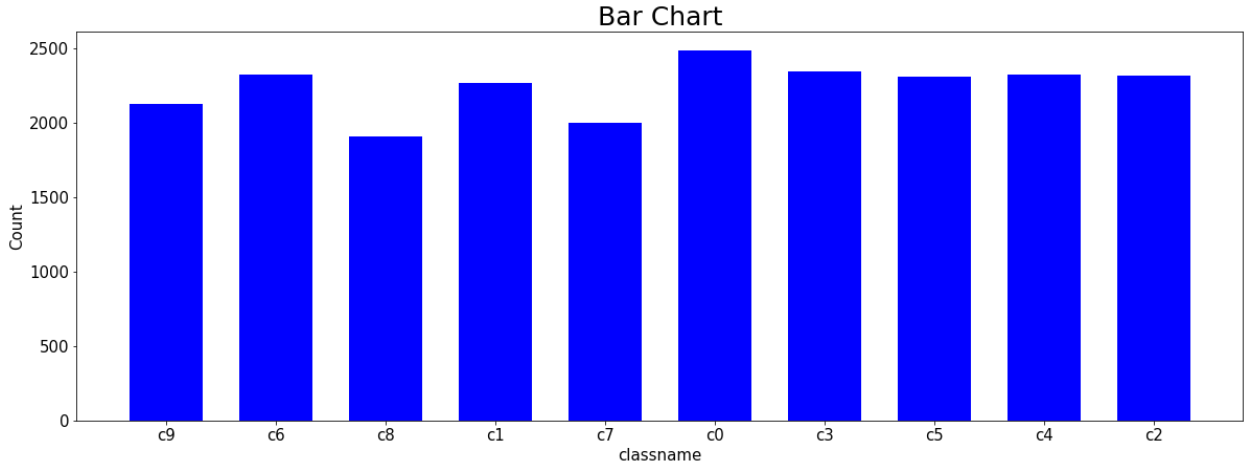
Goal is to predict the likelihood of what the driver is doing in each picture.

Distracted Driver Detection – Milestone Report

Dataset Description:

- **Image Size** - 480 X 640 pixels
- **Training Images count** - 22424 images
- **Image type** - RGB
- **Image field of view** - Dashboard images with view of Driver and passenger
- **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
- **Loss** - multi-class logarithmic loss

Training set Chart:



Balanced training set for 10 classes

Distracted Driver Detection – Milestone Report

Safe driving
(c0)

Texting with right hand
(c1)

Doing hair & makeup
(c8)

Few examples above

Deep Learning Background:

Convolutional Neural Networks

In order to tackle the distracted driver detection problem, the Convolutional Neural Network (CNN) model is utilized.

CNNs have proven to perform remarkably well on classifying images, and as such, are a great fit for this problem. It is worth noting that CNNs typically suffer from overfitting, which occurs when a model adapts too well on trained data, but performs poorly on new data, and thus is said to generalize poorly. This is an issue that we try to mitigate as much as possible throughout this work. CNNs rely on the idea that local understanding of an image is good enough, with the practical benefit of having fewer parameters, consequently reducing the computation time and data required to train the model.

Rather than have a fully connected layer for every pixel, CNNs only have enough weights to look at small parts of the image at a time. The process typically involves a convolution layer, followed by a pooling and an activation function, but not necessarily in that exact order. These three separate operations can be applied as separate layers to the original image, typically multiple times. Finally, a fully connected layer (or several) are added at the end in order to classify an image accordingly.

With the highly variable number of combinations and permutations, it can be difficult to find the exact one that gives the optimal performance.

CNN designs are driven by the community and research who thankfully have made some successful results, and made their CNNs publicly available for others to use and improve upon

VGG16

VGG is a CNN model proposed by Karen Karen Simonyan, and Andrew Zisserman, in their paper Very Deep Convolutional Networks for Large-Scale Image Recognition

VGG16, also commonly referred to as OxfordNet, is named after the Visual Geometry Group at the University of Oxford. More specifically, VGG16 is a deep 16-layer Convolutional Neural Network, with 13 convolutional layers, and 3 fully connected ones at the very top.

A visual representation of the VGG16 architecture is shown in Figure 1.

The model is able to achieve up to 92.7% top-5 accuracy on ImageNet4, which contains nearly 14 million images with over 1000 categories. The authors have not only made the model publicly available, but also its pre-trained ImageNet weights. This is something that is utilized in our work in order to not only decrease the computational cost, but also increase the accuracy of the model.

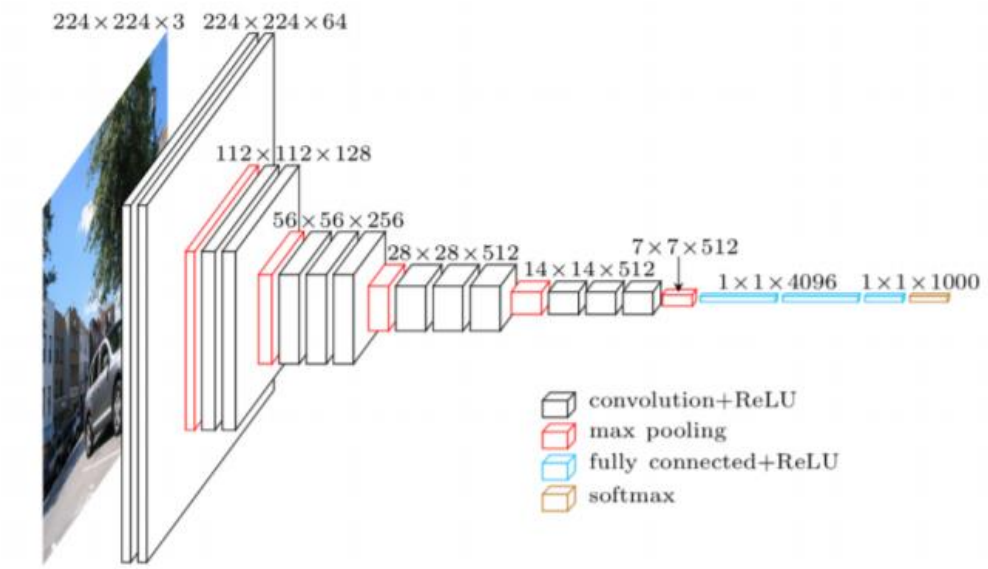


Figure 1: VGG16 Architecture Visualized[3]

Transfer Learning:

Transfer learning embodies the idea that knowledge gained whilst solving one problem, can be applied in solving a different, but related, problem.

Typically, Transfer Learning can be applied to machine learning, or more specifically CNNs. This is usually most appropriate when there is insufficient data or computational power, in which case a model with pre-trained weights on a much larger database to solve one problem can be applied. One strategy typically employed is to use the pre-trained convolutional network as a fixed feature extractor. This is done by completely removing the last fully connected layers, and then using the rest of the CNN as a feature extractor for a different problem. A linear classifier can then be trained using those features, as well as the new dataset to solve a different problem.

https://www.ontario.ca/page/distracted-driving
https://www.cdc.gov/motorvehiclesafety/distracted_driving/
https://crashstats.nhtsa.dot.gov/Api/Public/ViewPublication/812381

Distracted Driver Detection – Milestone Report

Data Split

The overall dataset provided by State Farm contains close to 100,000 images that fall under the 10 classes.

More specifically, the dataset was originally split up into training and testing data, with 22,424 and 79,726 images in each, respectively.

The main difference between the two is that the training data contained the ground truth labels of each image, while the testing images did not. This was done deliberately since it was a competition and like such, it allowed for some measure of success between different submissions.

In this work, however, in order to train and evaluate the performance of the models, the 22,424 training labeled images were split up further into 80% actual training data, 20% validation

This allows us to evaluate the model offline without having to submit to the Kaggle , but also test the model against unseen data and apply our own testing measures.

The initial concern is that it might not be enough data to train the model; however, this will be supplemented by data augmentation and transfer learning techniques.

Deep Learning Implementation:

Models to be used- Using Convolutional Neural Networks and Transfer Learning (VGG16)

Framework - Keras version

Loss: Categorical Cross Entropy

Metrics to be used: Accuracy, Precision, Recall, F1 score & Heatmap (Validation Train Predict Vs Validation Target)

Optimizer: rmsprop

Checkpoint: Model checkpoint by monitoring 'val_accuracy' and storing best weights

Output activation type: softmax

Image Pre-Processing:

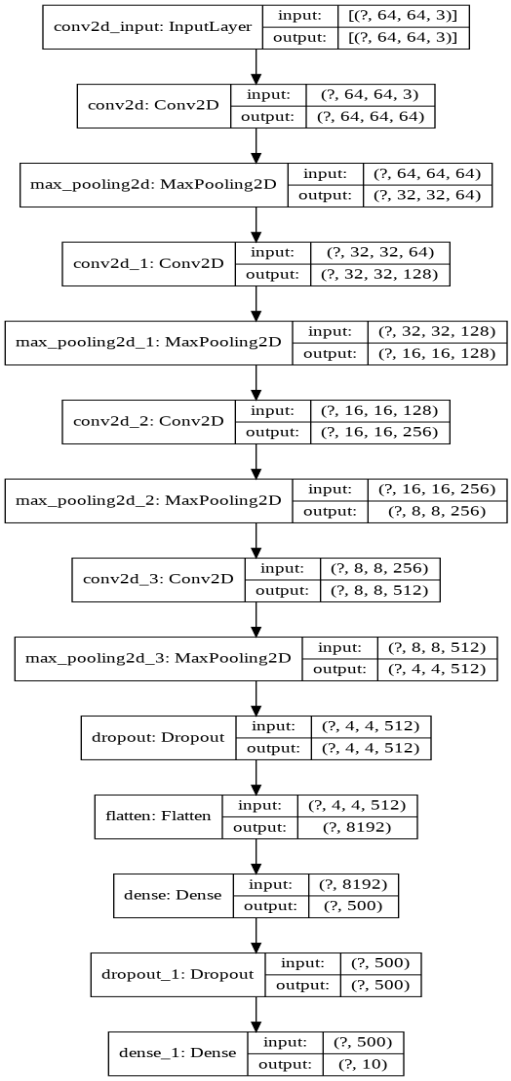
Much of the image manipulation had to be done manually and prior to the machine learning process, since it did not fit into memory using the limited hardware at our disposal.

As a result, the images had to be reduced significantly from 640 x 480 to 64 x 64 and grayscale as depicted in Figure.

Distracted Driver Detection – Milestone Report

Convolutional Neural Networks:

Layer layout:

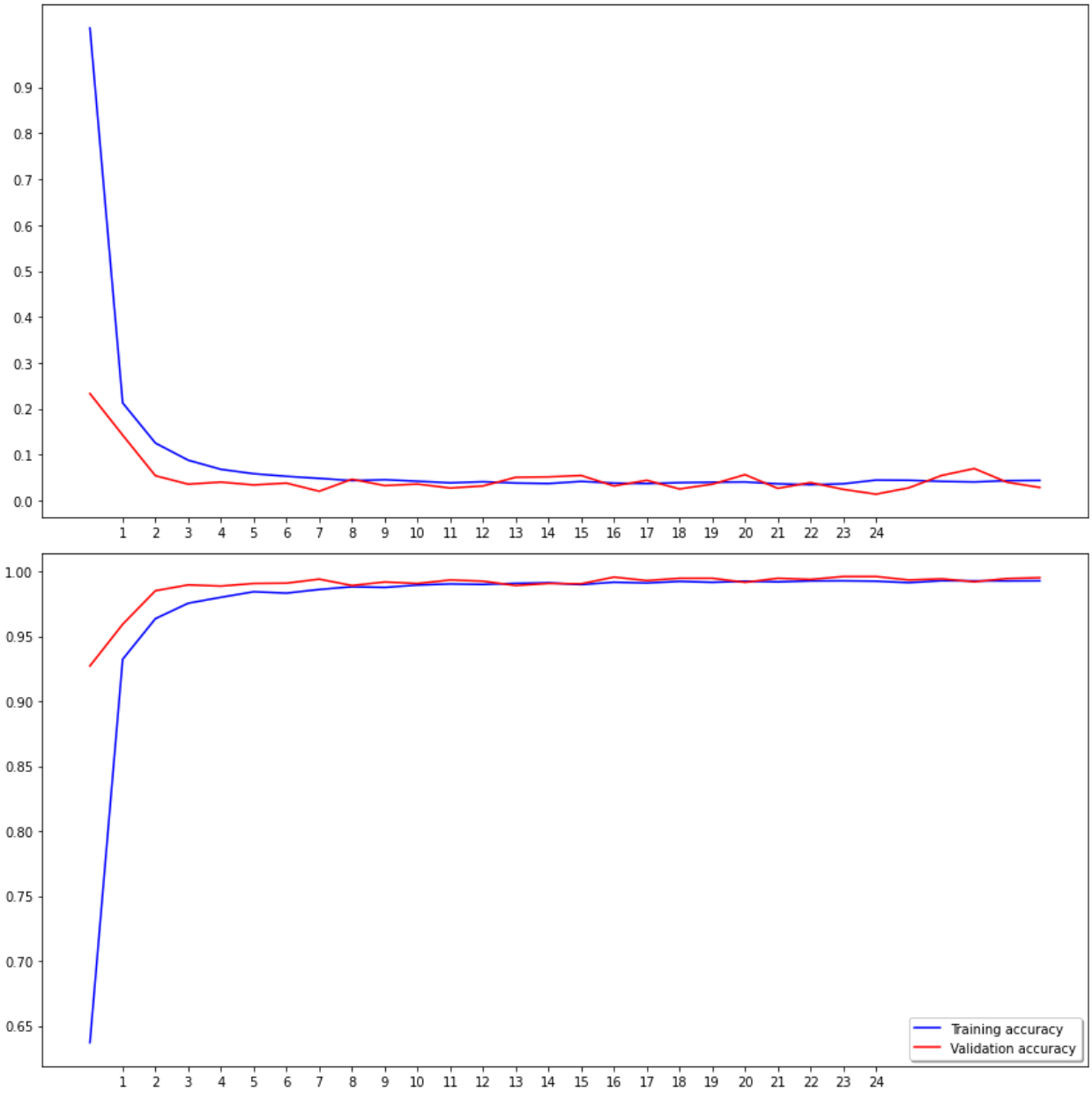


Distracted Driver Detection – Milestone Report

Epchs & Batch size used:

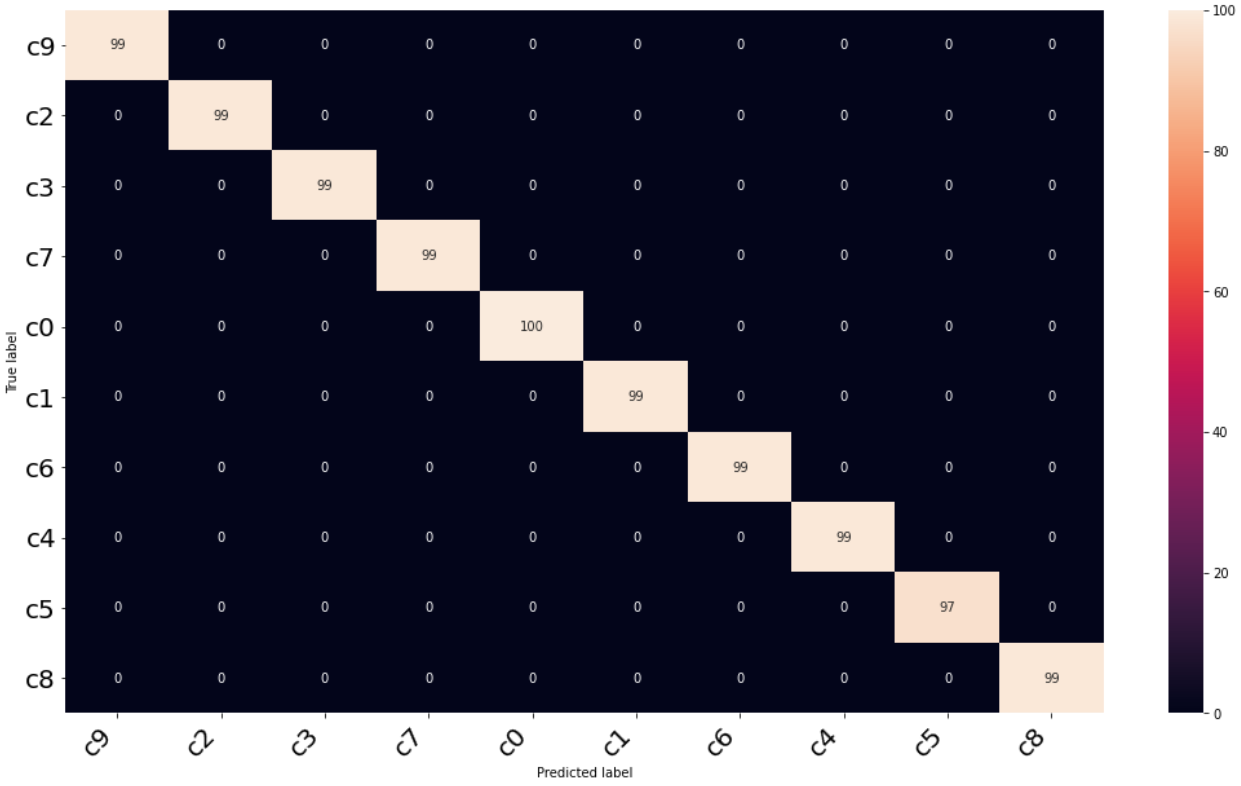
Epoch -30 **& Batch size –** 40

Loss Vs Accuracy for Training & Validation:



Heatmap :

Distracted Driver Detection – Milestone Report

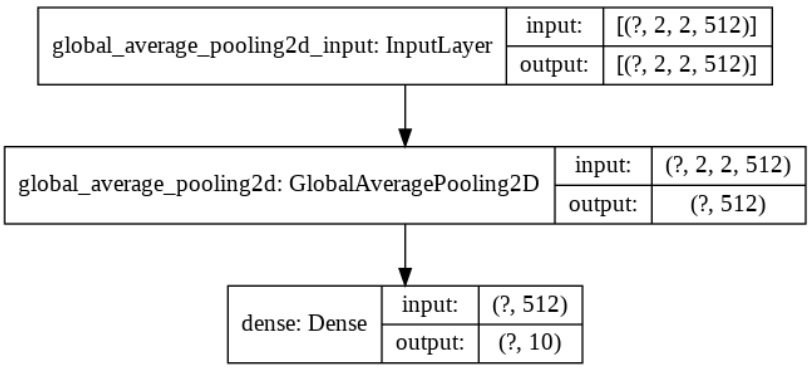


Metrics:

```
Accuracy: 0.995095
Precision: 0.995100
Recall: 0.995095
F1 score: 0.995088
```

Vgg16 Base:

Layer layout: (On top of VGG16)

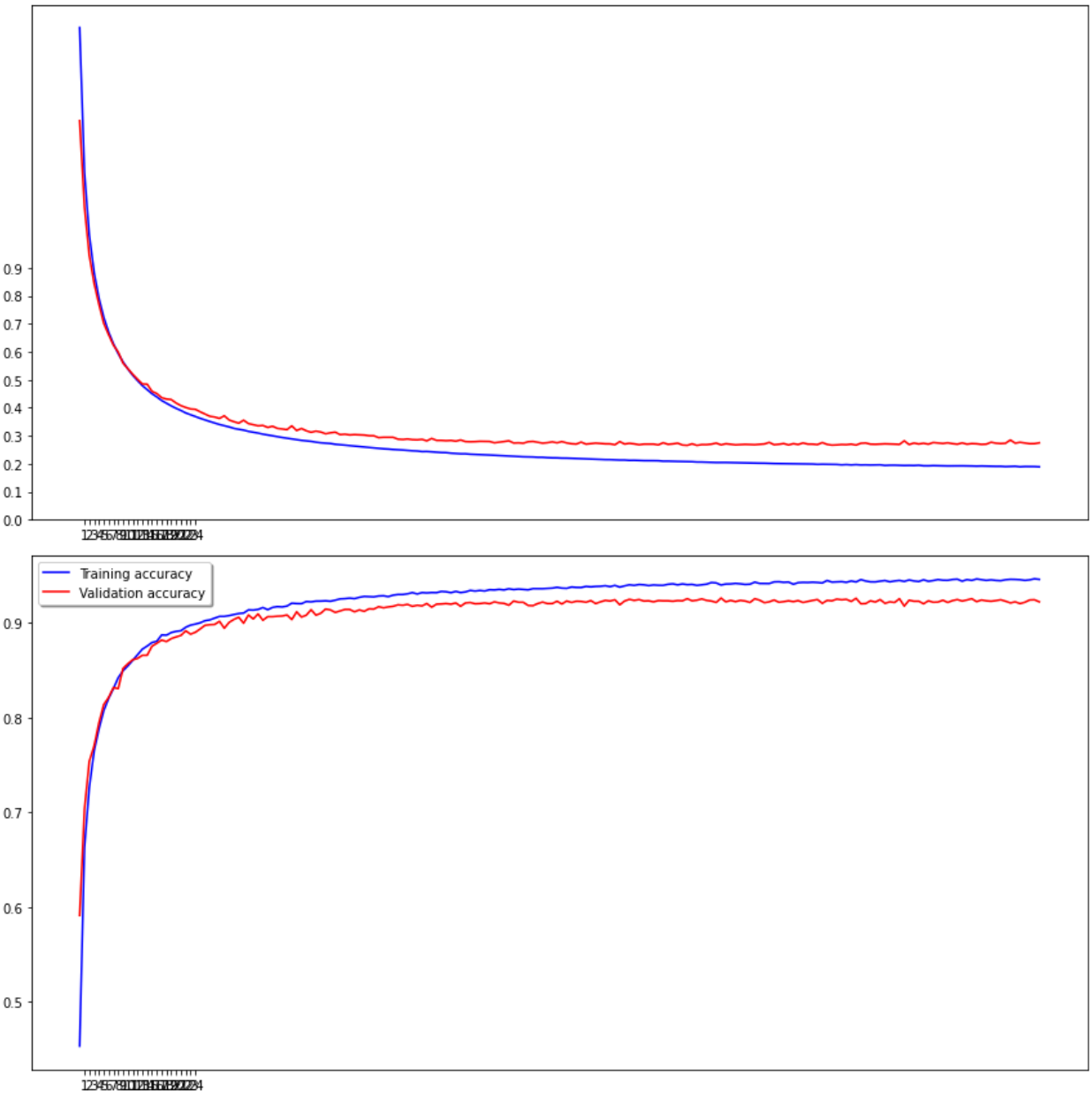


Distracted Driver Detection – Milestone Report

Epchs & Batch size used:

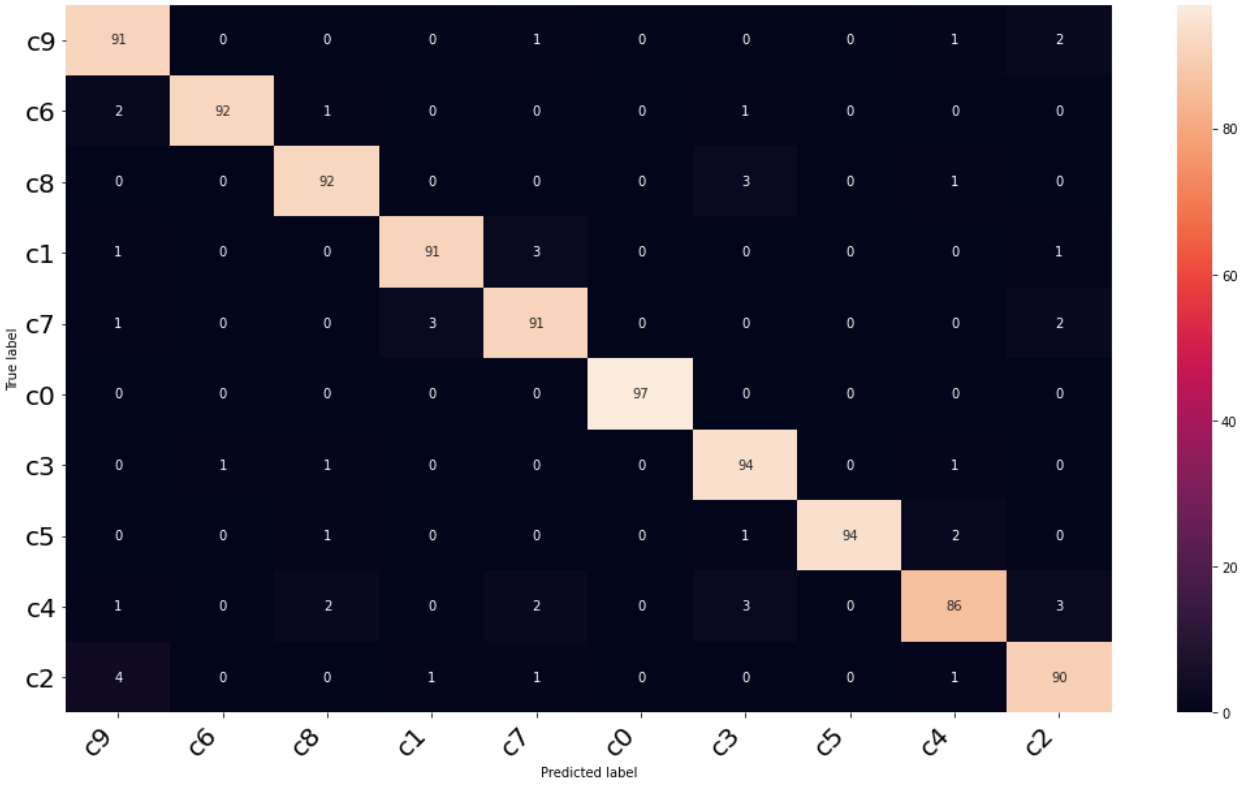
Epoch -200 **& Batch size –** 15

Loss Vs Accuracy for Training & Validation:



Heatmap :

Distracted Driver Detection – Milestone Report



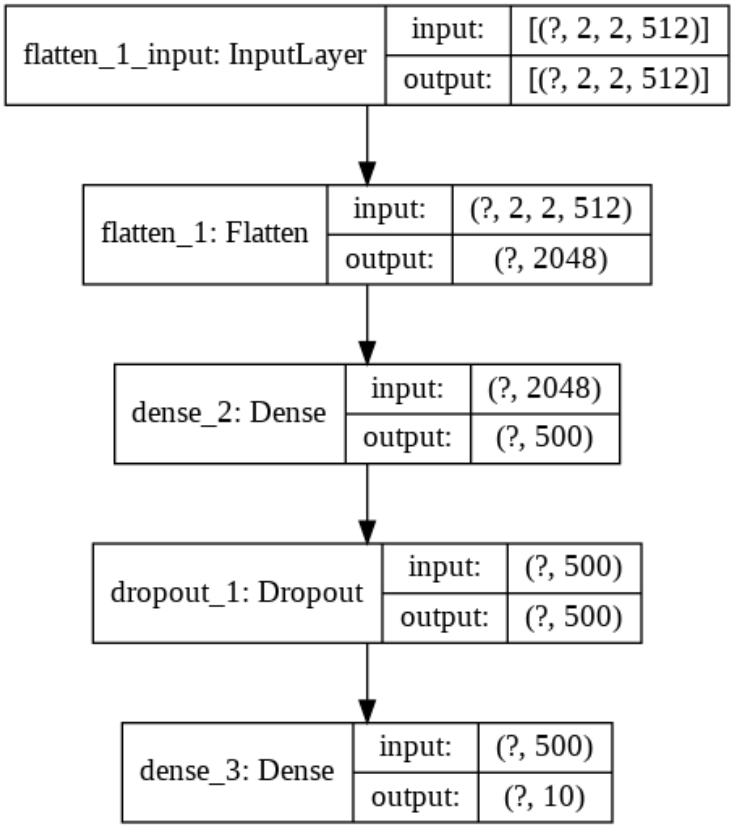
Metrics:

```
Accuracy: 0.922408
Precision: 0.923282
Recall: 0.922408
F1 score: 0.922602
```

Distracted Driver Detection – Milestone Report

Vgg16 with Feature Extraction:

Layer layout: (On top of VGG16)

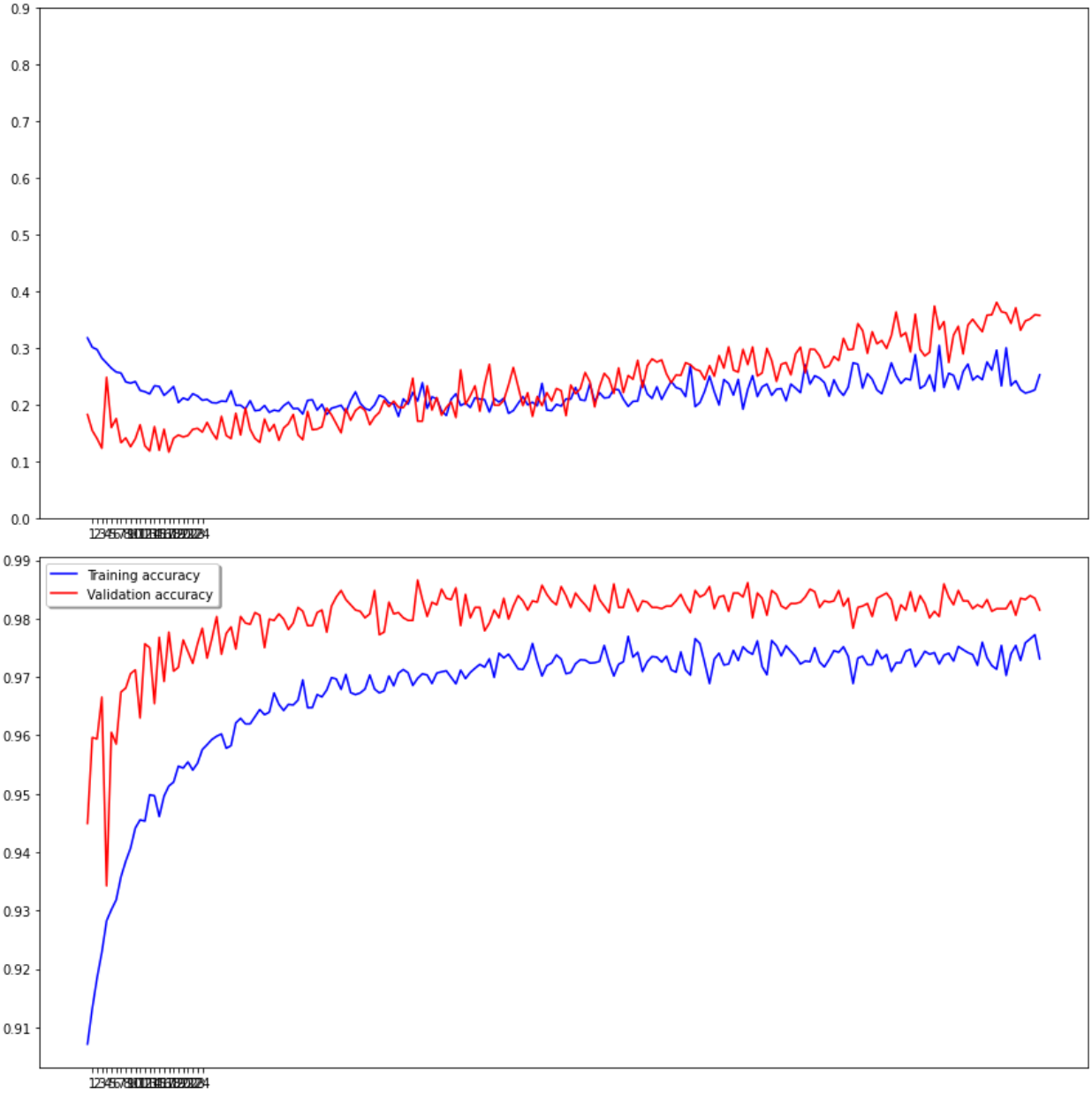


Epchs & Batch size used:

Epoch -200 **& Batch size –** 15

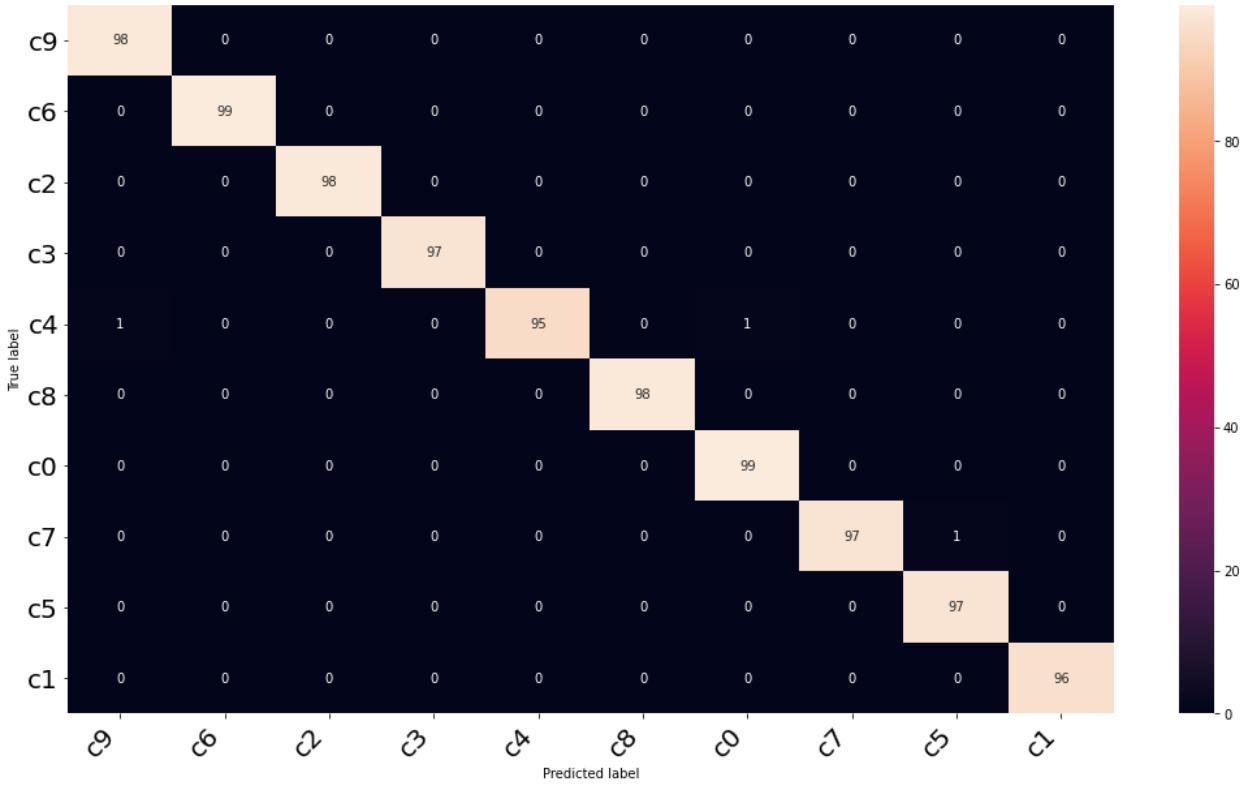
Distracted Driver Detection – Milestone Report

Loss Vs Accuracy for Training & Validation:



Distracted Driver Detection – Milestone Report

Heatmap :



Metrics:

```
Accuracy: 0.981494
Precision: 0.981659
Recall: 0.981494
F1 score: 0.981507
```

Conclusion*:*

This work has looked at solving the detection of distracted drivers through images obtained from the State Farm Distracted Driver Detection competition on Kaggle.

By using a pre-trained VGG16 network, extracting the bottleneck features, and retraining a new set of fully connected layers, the model was able to achieve 98.15% accuracy on test data.

Despite given the task of classifying very specific classes, the model is evidently able to accomplish that with great success.

Further evaluation revealed that the most miss-labeled class was reaching behind, often confused with the driver talking on their phone with the right hand. Overall, the model has proven to be very effective at predicting distracted drivers, and will hopefully, one day, aid in preventing further injuries and deaths resulting from distracted driving.

Distracted Driver Detection – Milestone Report

Future Scope of Work:

In this Capstone Project, we could use below methodological to further improve the models. Here are a few ideas of what I would try if time allowed:

- Using ImageDataGenerator instead of dimension converter
- Fine tune a few of the lower layers of the VGG16 network by freezing them and retraining the remaining ones.
- Using Restnet model for training & predicting