Aviral Somani

CS3891 Final Project Report

Abstract

This is a paper detailing my attempts at creating a performant classifier for the DrivFace dataset using convolutional and deep feedforward networks and attain a high accuracy in each class, frontal, looking right, and looking left. To solve this problem, I used a neural network architecture using convolutional layers with a number of 3x3 kernels for edge detection paradigms, and dense feedforward layers to resolve the nonlinearities in the visible facial structure depending on the direction in which the subject is looking.

1. Introduction

This report evaluates the process and results of a project aiming to build and train a classifier for the DrivFace dataset from the UCI Machine Learning Repository. This is a database of image sequences of subjects taken while they are driving in real scenarios. The dataset consists of 4 drivers, two male and two females, photographed in numerous instances looking either right, left, or facing forward.

In order to solve this problem, I developed a number of architectures composed of both convolutional layers with 3x3 kernels for edge detection and dense feedforward layers with ReLU activation functions. My methodologies are described later, but I was able to start at a model with a large number of parameters and streamline the architecture to find a smaller network with similar performance.

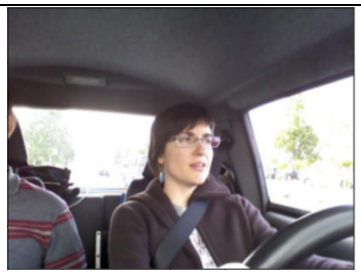
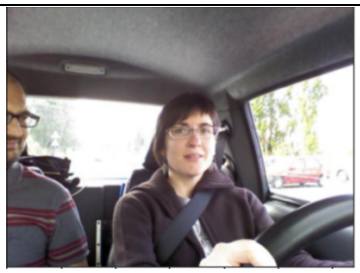
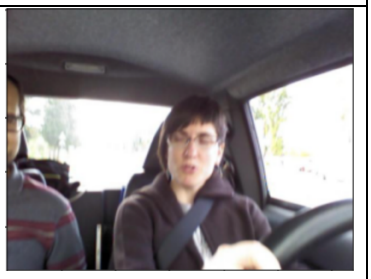
2. Problem

At its core, this is an image classification problem; however, if considered, most of these images are very similar to one another, which means that the number of instances should ideally be balanced similarly across all the classes, looking left (head angle between -45° and -30°), looking right (head angle between 30° and 45°) and frontal (head angle between -15° and 15°). This was, however, not the case because there were 546 frontals, 27 looking right, and 33 looking left, a resounding bias in favor of frontal facing examples.

Additionally, because data is collected from four subjects, we should ideally have the same number of samples from each subject. This was roughly the case for the first three, who had 179, 170, and 167 examples. The final one only had 90. The internal breakdowns mirrored the total breakdown, a large bias in favor of the frontal label, with much fewer examples of the looking

right and looking left examples. A notable lack is that there are no images where subject 3 is looking left.

Below are examples of the data.

	Looking Left	Frontal	Looking right
1			
2			
3	No image.		
4			

3. My Approach

As described earlier, my approach consisted of first creating a combination convolutional and deep feedforward network with approximately 17 million parameters. I was able to shave this down soon to 4.5 million parameters without severely sacrificing the final model accuracy. Initial architectures are shown below. The third is the same as the second with an Adagrad optimizer instead of Adam.

```
Model: "sequential_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_1 (Conv2D)            (None, 64, 64, 64)        1792
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 64, 64, 128)       73856
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 32, 32, 128)       0
_________________________________________________________________
conv2d_3 (Conv2D)            (None, 32, 32, 256)       295168
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 16, 16, 256)       0
_________________________________________________________________
flatten_1 (Flatten)          (None, 65536)             0
_________________________________________________________________
dense_1 (Dense)              (None, 256)               16777472
_________________________________________________________________
dropout_1 (Dropout)          (None, 256)               0
_________________________________________________________________
dense_2 (Dense)              (None, 128)               32896
_________________________________________________________________
dense_3 (Dense)              (None, 3)                 387
=================================================================
Total params: 17,181,571
Trainable params: 17,181,571
Non-trainable params: 0
_________________________________________________________________
```

```
Model: "sequential_2"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_4 (Conv2D)            (None, 64, 64, 256)       7168
_________________________________________________________________
max_pooling2d_3 (MaxPooling2 (None, 32, 32, 256)       0
_________________________________________________________________
conv2d_5 (Conv2D)            (None, 32, 32, 128)       295040
_________________________________________________________________
max_pooling2d_4 (MaxPooling2 (None, 16, 16, 128)       0
_________________________________________________________________
flatten_2 (Flatten)          (None, 32768)             0
_________________________________________________________________
dense_4 (Dense)              (None, 128)               4194432
_________________________________________________________________
dropout_2 (Dropout)          (None, 128)               0
_________________________________________________________________
dense_5 (Dense)              (None, 64)                8256
_________________________________________________________________
dropout_3 (Dropout)          (None, 64)                0
_________________________________________________________________
dense_6 (Dense)              (None, 32)                2080
_________________________________________________________________
dense_7 (Dense)              (None, 3)                 99
=================================================================
Total params: 4,507,075
Trainable params: 4,507,075
Non-trainable params: 0
_________________________________________________________________
```

Because I was dealing with 432x288 images, the easiest way to train models on this data was to first downsample to 64x64 as is visible in my first convolutional layer, and then create training and validation directories, each split up into 3 more directories, one for each label. Furthermore, in order to implement k-fold cross-validation. I used the offerings of the python os and shutil libraries to create temporary bin directories, each again divided internally by labels, and created training and validation directories for each of the k-fold cross validation iterations. I used the k-values of 3, 15, and 20.

For this, I had to find ways to randomly assign images to bins while ensuring that no bins were significantly different in size than other bins. In order to do this, bins had to be assigned on a label-by-label basis. Additionally, the assignments needed to be random but cyclical, meaning no bin could be assigned its $n+1$ th image before each other bin had n images in them already.

After finding the best model architecture without any regularization, I included kernel and bias level regularization parameters at each convolutional and dense layer to see if this could improve the final accuracy metric.

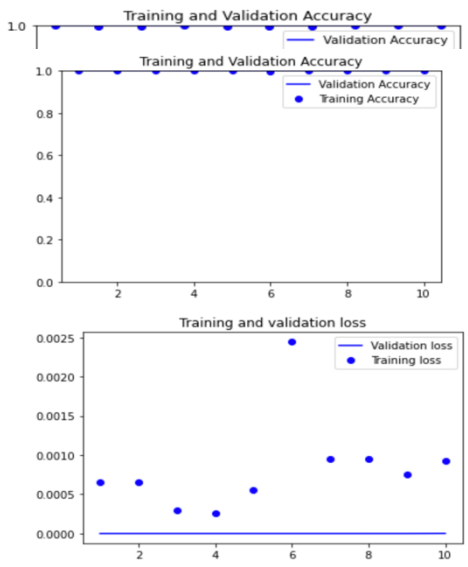
4. The Details

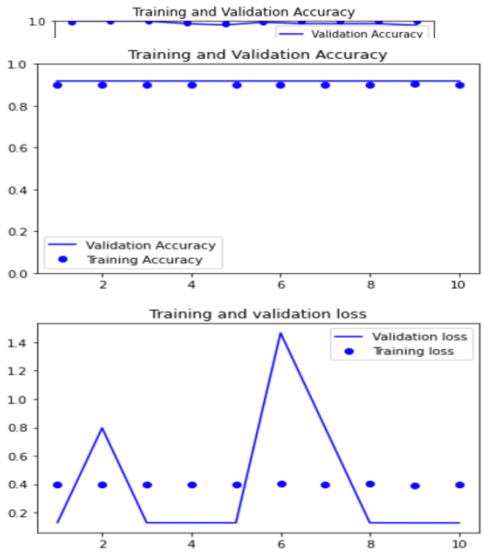
After training and evaluating each of the first three models, I found that the split of 20 on the second architecture tied with the first architecture with the best test accuracy on the held-out test set of 0.9756, in comparison to the second architecture with 0.9674. These differences are quite miniscule, and all of these models performed quite well.

On a randomly selected slice of 90% of the dataset, the second and third models attained accuracies of 0.9945 and 0.9963 compared to 0.9908 from the first model. Again, this difference is quite small; however, there is a positive correlation between the number of bins I used in my cross-validation and the final accuracy. This indicates that having more bins will mean that the model will be more generalizable and less overfit to patterns in each specific bins if the number of bins are smaller, because smaller bin sizes mean larger validation datasets.

Inclusion of the regularization parameter did not give me any sort of performance improvement. Instead, it decreased the testing accuracy on the entire dataset to 0.8995, which is a cut below anything the previous models were able to generate. It was also interesting that the validation loss for each training iteration with regularization ended up oscillating primarily between 0.8750 and 0.9167.

The final learning curves for each of the four models are below.





5. Conclusions

There are some difficulties in analyzing how performant this model that I have trained actually is, primarily because of the highly specific training and test data, that is also not in great quantity.

To this end, one of the techniques I employed to minimize overfitting and improve generalization while ensuring a sufficiently small model was k-fold cross validation. In doing so, I found that generally, the higher the number of bins used in the k-fold cross validation procedure, the more performant the model ended up being. This could be because the more bins there are, the more epochs for which the model trains and a greater variety of training and validation partitions that the model is exposed to.

All in all, I found that a relatively small model can be used on this dataset to classify the direction in which someone is facing to a very high degree of accuracy, (0.996). This was aided primarily by the use of CNN and deep feedforward combination networks and k-fold cross-validation. Additionally, for this use case the Adagrad optimizer outdid the Adam optimizer, however this could be confounded with the number of bins in cross-validation.

For future directions, I would find access to a stronger GPU and train more combinations of hyperparameters to see if there is an even smaller model with a stronger accuracy, however 0.996 would be quite difficult to top. Additionally, it might be imperative to build similar classifiers for a more diverse composition, because this dataset only contains Caucasian cisgendered lower to middle aged adults.

6. Sources

1. Chollet, François and Others (2015). Keras. Keras.io
2. Katerine Diaz-Chito, Aura Hernández-Sabató©, Antonio M. López, A reduced feature set for driver head pose estimation, Applied Soft Computing, Volume 45, August 2016, Pages 98-107, ISSN 1568-4946,
3. adrshm91, DrivFace_NoAugmentation – DeepGaze.ipynb, (2018), Github repository, https://github.com/adrshm91/DrivFace/blob/master/DrivFace_NoAugumentation%20-%20DeepGaze.ipynb