Capstone Project

Machine Learning Engineer Nanodegree

Dhilan Chandrasekara

21st December 2017

1 Definition

1.1 Project Overview

Over the past decade, the field of computer vision has seen a meteoric rise. For the first time ever, software is being developed to perform state of the art operations, such as driving autonomous vehicles [1] and diagnosing skin melanomas better than doctors [2].

Recognition of facial expressions is a task that comes naturally to humans, given it is a key component of non-verbal communication. However, human-level performance in this task has been difficult to arrive at computationally, given the visual pattern recognition nature of the problem.

In this project, I utilized convolutional neural networks (CNNs) to build a classifier to identify different facial expressions, using the FER2013 dataset available on Kaggle [3].

1.2 Problem Statement

1. Download the necessary dataset
2. Pre-process the data for input into a CNN
3. Explore the effectiveness of using a CNN built by scratch to build the classifier
4. Investigate different transfer learning models for classification
5. Analyse performance of each method

1.3 Metrics

Categorical cross entropy loss will be the method used to measure the performance of models. This is due to the desired outputs of the classifier being of a categorical nature, and the frequency of occurrence for each category being non-uniform, in the FER2013 dataset.

Categorical cross entropy is defined as follows:

$$H_{y'}(y) := -\sum_i y'_i \log(y_i)$$

For each prediction vector y_i for the appropriate target vector y.

In addition to the above metric, accuracy will be noted for each classifier. Whilst accuracy on it's own is not a sufficient metric to measure the performance of a given classifier, it gives insight into

the potential for the given classifier to be used in the context of an application that is designed to recognize facial expressions from a camera. In this use case, accuracy would one of the most important metrics. Categorical cross entropy, however, will be the most important metric considered when deciding whether a given classifier is superior to another in this project.

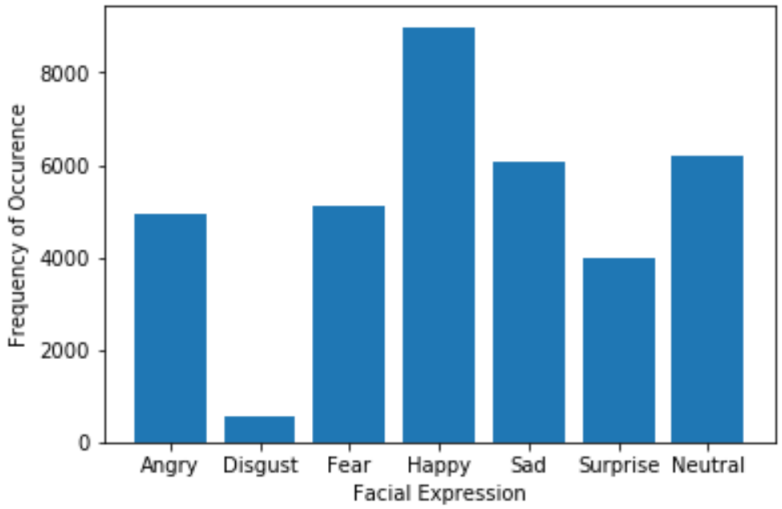
2 Analysis

2.1 Data Exploration

The FER2013 dataset consists of greyscale images that are cropped to have the face centred and taking up most of the image. There are seven different expressions (angry, disgust, fear, happy, sad, surprise, neutral). In total there are 38,887 images, and each image is sized 48 x 48 pixels.

In training the different models that will be tested throughout the project, 70% of the data will be used for training, 10% for validation, and the remaining 20% for testing.

Shown below is the relative frequencies of occurrence for each facial expression category:



The distribution is non-uniform with respect to the different categories a very low relative number of samples for the 'disgust' category.

2.2 Exploratory Visualization

As can be seen in the above samples, each face image is suitably cropped, so that the face is generally centred and takes up most of the space of the image. In the images, the faces tend to look either directly at the camera, or to the left or right. Since facial expression is independent of which direction the person is looking at, this is a strong sign that image augmentation, particularly involving horizontally flipping an image would be beneficial to use on this dataset for the given application. Translational image augmentation could also be used, since it the correct facial expression is independent of the relative position of the face in the image.

For this project, augmented images – both horizontally flipped and translated in both directions – will be used in an attempt to improve classifier performance.

2.3 Algorithms and Techniques

The classifier used for the project will be a Convolutional Neural Network. First, a benchmark model, detailed in the below section, will be trained on the dataset. Afterwards, more complexity will be added to the model, through changing parameters such as the number of convolutional layers, the number of filters, the amount of max pooling, dropout, and number of dense layers.

Model Parameters:

- Number of epochs
- Batch size
- Network architecture
- Types of layers
- Layer parameters

To speed up training time, a GPU will be used for training. For smaller scale networks, such as the benchmark model, a locally available GPU will be used (NVIDIA GTX 950), however for more memory intensive networks, such as the advanced from-scratch model and the transfer learning models, a GPU compute based cloud server will be used, with Amazon Web Services (AWS).

After investigating the performance abilities of convolutional neural networks built from scratch in Keras, transfer learning approaches will be investigated. In this project the VGG-19 pre-trained network was investigated. Restriction on image input size didn't allow for the use of other high performance image pre-trained networks, such as Resnet50 and Inception. These models would be suited to a dataset with higher resolution images.

2.4 Benchmark

A relatively simple convolutional neural network will be used as a benchmark model. Shown below is a summary of the model structure:

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_1 (Conv2D)            (None, 48, 48, 16)        80
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 24, 24, 16)        0
_________________________________________________________________
flatten_1 (Flatten)          (None, 9216)              0
_________________________________________________________________
dense_1 (Dense)              (None, 100)               921700
_________________________________________________________________
dense_2 (Dense)              (None, 7)                 707
=================================================================
Total params: 922,487.0
Trainable params: 922,487.0
Non-trainable params: 0.0
_________________________________________________________________
```

It is predicted that the single convolutional layers, in addition to a single fully connected layer, will allow the model to identify simple aspects of the image, such as lines and spots, and thus give reasonable classification performance on the dataset. No image augmentation will be used when testing the benchmark model.

3 Methodology

3.1 Data Pre-Processing

Given that the images available in the FER2013 dataset are already pre-processed appropriately, only a minimal amount of pre-processing will be required. The raw pixel values from the publicly available CSV files are processed to form an appropriate image.

Pre-Processing Steps:

- Convert from raw pixel values to tensor format
- Horizontally flip and translate images and add these to the training set

When using the transfer learning model VGG19, the model is pre-designed to accept colour images, which have three channels for the R, G and B values, with values ranging from 1 to 255. Greyscale images only have one of such values, and to use the transfer model, values must be filled in for each of the three channels. In this project, the same greyscale "darkness" values was copied to each of the three RGB channels. This allowed for the image to be input into the transfer learning model.

3.2 Implementation

After the suitable dataset was obtained, the following process was undertaken to build the facial expression classifier:

1. Split the total data available into training, testing and validation sets.
2. Generate useful visualizations relating to the data, such as the frequency of each expression, and some image visualizations of the raw pixle data.
3. Define the following helper functions:
    a. df_to_tensor(…) – Takes as input a pandas DataFrame, which obtains an organized collection of each pixel value for each image. This DataFrame is then converted to a tensor, a four-dimensional Numpy[citation to numpy source maybe] array that can be suitably input to a CNN using the Keras and TensorFlow frameworks.
    b. augment_data(…) – Creates two ImageDataGenerator [citation to Keras documentation], and specify the appropriate translational and rotational augmentations that will be applied to the image data.
4. Set up a GPU compute server on Amazon Web Services to conduct training
5. Test the baseline model on the dataset, without using image augmentation
6. Make incremental improvements to the baseline model, such as adding more convolutional layers, filters, dropout layers and max pooling layers
7. Import the transfer learning model, VGG19
8. Pre-process the data to have the same greyscale value for each RGB channel
9. Test the transfer learning model (pre-trained weights with two dense layers at the end) on the dataset
10. Make changes to the transfer learning model, such as adding more dense layers, adding dropout, and changing which layers are freezed (pre-trained weights from previous training are kept)
11. Compare the results of using each of the three approaches

To speed up the process of training, each of the networks were trained on an Amazon Web Services GPU compute instance.

In each case of training the CNN classifiers, RMSProp will be used as the optimization technique. This was chosen as a default given its proven performance in other classification tasks and was not experimented over in this project.

3.3 Refinement

The general approach of the project was to continue to add convolutional layers and additional filters to see an improved performance, and to take appropriate measures to reduce overfitting.

Most of the early convolutional neural network architectures that were tested, that comprised only of convolutional layers, max pooling layers and dense layers, were found to quickly overfit to the training data, often before the 10th epoch of training. Adding dropout was a suitable fix for this problem, however having too high a level of dropout caused decreases in overall model performance.

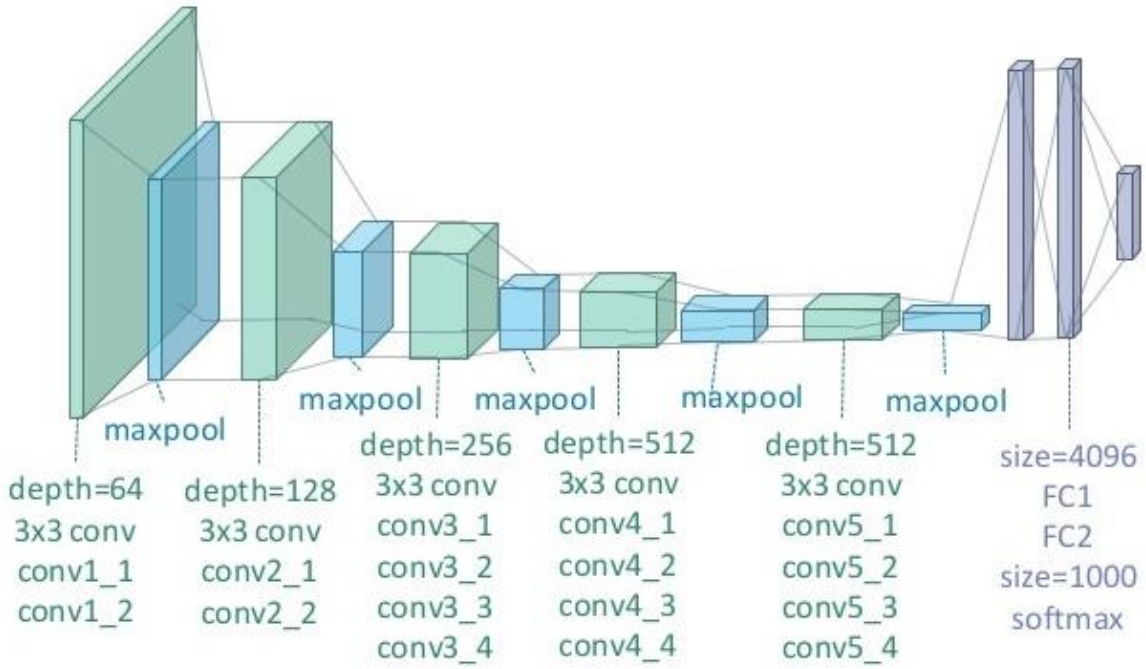
Below is the high performance advanced model that was built from scratch (without the use of transfer learning):

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_14 (Conv2D)           (None, 48, 48, 16)        80
_________________________________________________________________
max_pooling2d_14 (MaxPooling (None, 24, 24, 16)        0
_________________________________________________________________
conv2d_15 (Conv2D)           (None, 24, 24, 32)        2080
_________________________________________________________________
max_pooling2d_15 (MaxPooling (None, 12, 12, 32)        0
_________________________________________________________________
conv2d_16 (Conv2D)           (None, 12, 12, 64)        8256
_________________________________________________________________
max_pooling2d_16 (MaxPooling (None, 6, 6, 64)          0
_________________________________________________________________
flatten_5 (Flatten)          (None, 2304)              0
_________________________________________________________________
dense_7 (Dense)              (None, 100)               230500
_________________________________________________________________
dense_8 (Dense)              (None, 7)                 707
=================================================================
Total params: 241,623.0
Trainable params: 241,623.0
Non-trainable params: 0.0
_________________________________________________________________
```

For the transfer learning model VGG19 was explored. Four dense layers were added to the end of the model, and an appropriate softmax activation function for the dataset was added to the end of the network. Below is a summary of the model:

Layer (type)	Output Shape	Param #
input_1 (InputLayer)	(None, 48, 48, 3)	0
block1_conv1 (Conv2D)	(None, 48, 48, 64)	1792
block1_conv2 (Conv2D)	(None, 48, 48, 64)	36928
block1_pool (MaxPooling2D)	(None, 24, 24, 64)	0
block2_conv1 (Conv2D)	(None, 24, 24, 128)	73856
block2_conv2 (Conv2D)	(None, 24, 24, 128)	147584
block2_pool (MaxPooling2D)	(None, 12, 12, 128)	0
block3_conv1 (Conv2D)	(None, 12, 12, 256)	295168
block3_conv2 (Conv2D)	(None, 12, 12, 256)	590080
block3_conv3 (Conv2D)	(None, 12, 12, 256)	590080
block3_conv4 (Conv2D)	(None, 12, 12, 256)	590080
block3_pool (MaxPooling2D)	(None, 6, 6, 256)	0
block4_conv1 (Conv2D)	(None, 6, 6, 512)	1180160
block4_conv2 (Conv2D)	(None, 6, 6, 512)	2359808
block4_conv3 (Conv2D)	(None, 6, 6, 512)	2359808
block4_conv4 (Conv2D)	(None, 6, 6, 512)	2359808
block4_pool (MaxPooling2D)	(None, 3, 3, 512)	0
block5_conv1 (Conv2D)	(None, 3, 3, 512)	2359808
block5_conv2 (Conv2D)	(None, 3, 3, 512)	2359808
block5_conv3 (Conv2D)	(None, 3, 3, 512)	2359808
block5_conv4 (Conv2D)	(None, 3, 3, 512)	2359808
block5_pool (MaxPooling2D)	(None, 1, 1, 512)	0
flatten_6 (Flatten)	(None, 512)	0
dense_9 (Dense)	(None, 1024)	525312
dropout_1 (Dropout)	(None, 1024)	0
dense_10 (Dense)	(None, 1024)	1049600
dropout_2 (Dropout)	(None, 1024)	0
dense_11 (Dense)	(None, 1024)	1049600
dropout_3 (Dropout)	(None, 1024)	0
dense_12 (Dense)	(None, 1024)	1049600
dropout_4 (Dropout)	(None, 1024)	0
dense_13 (Dense)	(None, 7)	7175

As can be seen, this transfer learning model is vastly different in it's architecture as compared to the previous models. The convolutional layers section of the model can be broken up into five different blocks. Each of these blocks consists of many convolutional layers, which each will identify patterns in the images, with more complex patterns being recognized later in the network. In between adjacent blocks is a max pooling layer, decreasing the total dimensionality of image passed on to the next layers, in an effort to decrease overfitting. Seen in this way, the VGG-19 architecture is rather systematic, with each subsequent block identifying are number of patterns which are passed on to the next block to be further analysed. An illustration of this can be seen below [8]:



The high training time of the transfer learning model meant it was difficult to undergo a high amount of iteration with different supplemental architectures. Eventually a relatively simple model was found, by adding four fully connected layers at the end of the VGG-19 model, with sufficient dropout to reduce overfitting.

Initially, a large amount of the VGG-19 model's weights was frozen during training, particularly the weights for the convolutional layers in the first and second layers. This was due to the belief that the network was already trained to classify images, and the first few convolutional layers would be associated with identifying simple aspects of the image, such as lines and shapes. Identification of these patterns, in conjunction with more analysis later in the network, was believed to be helpful to the facial recognition classification task. However, after experimentation it was found that better performance was obtained by keeping the architecture of the VGG-19 model but retraining all of the weights. This is likely since only greyscale images are used in this application, whereas the VGG-19 model was likely trained mostly with coloured images.

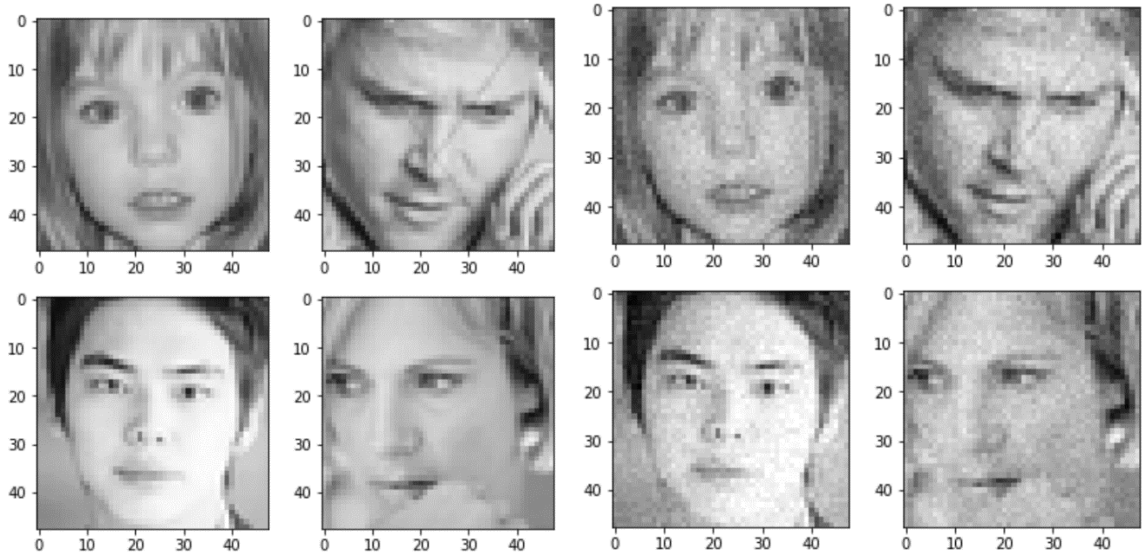
4 Results

4.1 Model Evaluation and Validation

	Baseline	**Advanced**	**VGG-19 Transfer**
Categorical Cross Entropy Loss	1.43213	1.17480	1.07692
Accuracy	45.43	54.92	61.03

As can be seen from above, increased performance in both accuracy and categorical cross entropy loss is seen with more complex architectures.

In order to determine how robust the final model is to noise, a small amount of Gaussian noise was added to the RGB values of each of the test images, and evaluated. Below is an illustration of the addition of the noise, of standard deviation 6.375 for each RGB channel:



Below is a summary of the VGG-19 transfer learning model's performance with varying Gaussian noise. The columns give the varying standard deviation of noise that was added to the samples.

	6.375	**8.415**	**12.75**	**15.3**
Categorical Cross Entropy Loss	1.12705	1.20526	1.34060	1.382
Accuracy (%)	60	59.67	55	53.33

The results show that the final model is relatively robust to noise, with only minor drops in categorical cross entropy loss for suitable amounts of noise. Higher amounts of noise heavily reduced the performance of the classifier, as seen by the last two columns above, however it must be noted that with such excessive amounts of noise it is difficult for even a human to decipher the correct emotions.

4.2 Justification

As can be seen there is a substantial improvement of both accuracy and categorical cross entropy loss moving from the baseline to the advanced model. An even further increase in performance is obtained by utilizing the existing VGG-19 architecture. The increases in model complexity allow for more complex patterns to be discerned from the images, such as the orientation of the eyebrows and the cheek muscles.

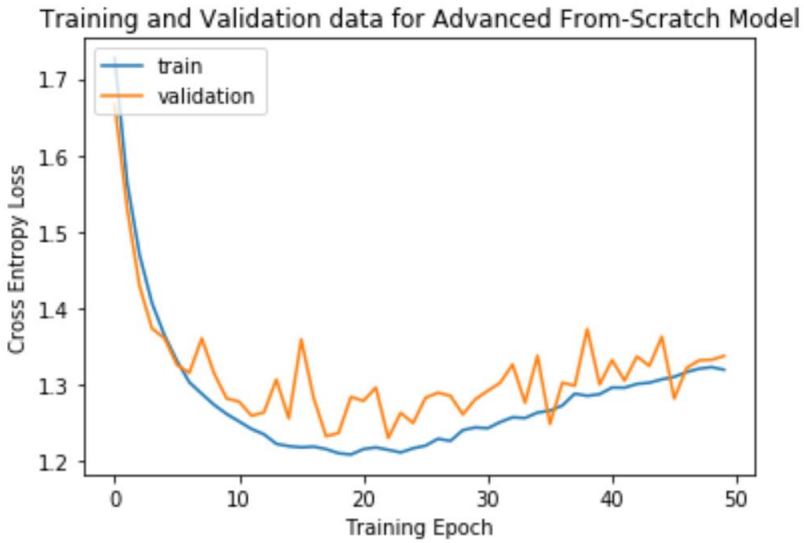
5 Conclusion

5.1 Free-Form Visualization

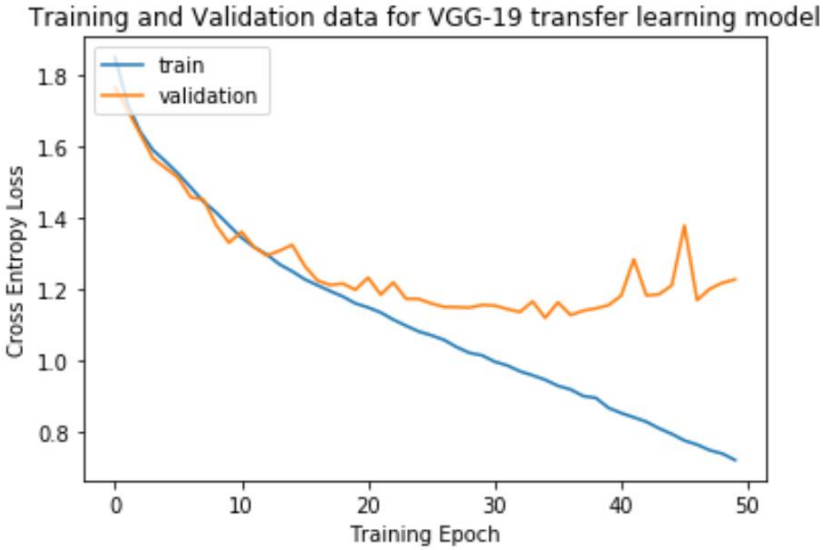
Below is a graph of the training and validation loss for the baseline model:

By the validation loss, it is shown that the baseline model can learn a small amount of the characteristics of the data in the first few epochs, however after this it quickly overfits. This can

most likely be attributed to high dimensionality of the network, having a high number of convolutional filters in the first layer and minimal max pooling layers or dropout. By the 20th epoch the model has essentially memorized the majority of the training data (shown by the low training loss) whilst being unable to generalize well to unseen data.



Above is the training characteristics of the advanced model. Not only was the convolutional neural network more complex in it's architecture in this example, image augmentation was utilized. Interestingly, when validation loss started to increase (usually a sign of overfitting), the training loss increased also. This is likely due to the fact that with image augmentation being used, the model overfitted to noise in previously seen training data examples, but wasn't able to generalize to the new augmented images introduced in training in later epochs.



The final architecture involving the VGG-19 model with four dense layers appended to the end of the network, resulted in the above training graph. A lower categorical cross entropy loss of 1.0769 was obtained, showing the effectiveness of using pre-existing, high performance CNN architectures. The model overfits the training data around the 40th epoch, although a checkpoint system was used to ensure the weights that were obtained before overfitting were maintained.

5.2 Reflection

Building the facial expression recognition classifier was a lengthy process that required many different steps.

Firstly, the dataset was obtained, which was in a less than desirable format, a .csv file comprising of a list of each pixel value for each image. Suitable processing was done to extract the image from this list of values. After this initial step however, minimal pre-processing was required, due to the images already being significantly and desirably pre-processed.

What was most difficult about the project for me was finding ways to improve classifier performance by changing the neural network architecture. On many occasions I made the model more complex in hopes to improve performance, but the model often just overfitted the data. Overcoming this overfitting was difficult too, as it was unclear in which ways max pooling and dropout layers should be added to prevent over-fitting but maintain high model performance. In fact, at times it was difficult to get better performance than the baseline model with more complex CNN architectures.

5.3 Improvement

In this project, only one pre-trained model was used for transfer learning, VGG19. In the future, other pre-trained models could be explored – models that were trained on greyscale images rather than RGB, since this would be expected to give better results. In addition, pre-trained models that are more specific to the problem domain, classification of faces, could have been explored to gain higher performance with the transfer learning models.

The low resolution of the dataset images prevented the use of the Inception and Resnet50 models as planned. In the future, a variety of different datasets, those of higher image resolution, could be explored to investigate the true potential of implementing transfer learning models.

In addition, only a small amount of the parameters available in training CNNs were iterated over. This includes the number of convolutional layers, number of dense layers, amount of dropout, max pooling layers, image augmentation and number of filters. Other parameters such as momentum, kernel size and stride could be investigated in the future in an effort to further improve performance of the classifier.

References

[1] M. Bojarski, D. Del Testa, D. Dworakowski, B. Firner, B. Flepp, P. Goyal, L.D. Jackel, M. Monfort, U. Muller, J. Zhang, X. Zhang, J. Zhao, K. Zieba, NVIDIA Corporation, Holmdel, NJ 07735, *End to End Learning for Self-Driving Cars* https://arxiv.org/pdf/1604.07316.pdf, 2016.

[2] A. Esteva, B. Kuprel, R.A. Novoa, J. Ko, S.M. Swetter, H.M. Blau, S. Thrun, *Dermatologist-level classification of skin cancer with deep neural networks,* 2017

[3] https://www.kaggle.com/c/challenges-in-representation-learning-facial-expression-recognition-challenge/data Dataset provided by: Pierre-Luc Carrier and Aaron Courville

[4] https://machinelearningmastery.com/display-deep-learning-model-training-history-in-keras/

[5] https://towardsdatascience.com/transfer-learning-using-keras-d804b2e04ef8

[6] V. Mavani, S. Raman, K. Miyapuram, *Facial Expression Recognition using Visual Saliency and Deep Learning*, 2017

[7] A. Savoiu, J. Wong, *Recognizing Facial Expressions Using Deep Learning,* 2017

[8] https://github.com/tejaslodaya/neural-style-transfer