Machine Learning Engineer Nanodegree

Capstone Project

Ilja Avadiev
October 28th, 2018

1 Definition

1.1 Project Overview

Human beings learn from the day they are born to recognize body language and facial expressions of other humans. “Reading” these expressions and reacting accordingly is an essential skill some people seem to be proficient in which in turn provides them with a competitive advantage in many areas.

Haggard and Isaacs discovered 6 so called micromomentary expressions (also called micro expressions), which appear and disappear in a fraction of a second and are only visible through slowly played video footage[1]. The expressions include anger, disgust, fear, happiness, sadness and surprise.

The dataset used in this project was originally used in a Kaggle competition. The task of the competition was to classify the correct micro expression. The winner of the competition, Yichuan Tang, used a Convolutional Neural Network[2]. Tang replaced a softmax activation function, used most commonly in the final layer of a classification CNN by a linear support vector machine (SVM). He shows, that by using a SVM the classification accuracy can be improved.

This project uses the same dataset to classify this microexpressions, using a CNN approach.

1.2 Problem Statement

Even for the most social and empathetic person it is not always easy to realize the emotional state of their conversational partner, especially considering that micro expressions are not directly recognizable. Using high speed cameras or photographs

1 See Ernest A. Haggard Kenneth S Isaacs, 1966, Micromomentary facial expressions as indicators of ego mechanisms in psychotherapy
2 See Yichuan Tang, 2015, Deep Learning using Linear Support Vector Machines

an automatic solution should be aimed at which classifies these expressions. The process could for example facilitate the acceptance of self-driving cars or robotic household help. The goal here is to create a good performing machine learning architecture which is able to correctly classify the underlying micro expression of static pictures in the dataset.

The following steps are necessary for that purpose:

1. Download and preprocess the data

2. Determine the correct measures of performance and write corresponding functions

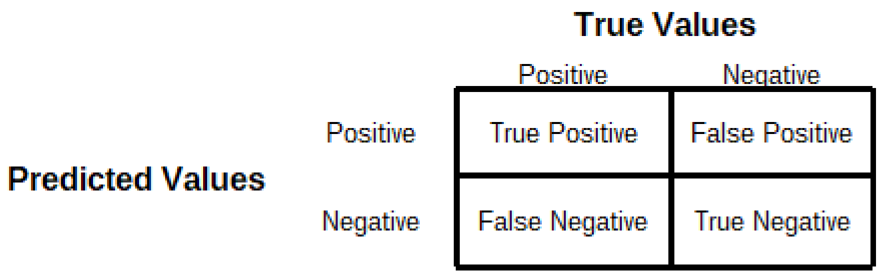
3. Experiment with different neural network architectures to determine which produces the best performance

1.3 Metrics

Accuracy is a common metric, when measuring the performance of a classification model. The accuracy paradox is often encountered when the data is strongly unbalanced and the model tends to classify each category into a the category with the most observations. The confusion matrix and ROC are good measures to avoid the accuracy paradox.

1.1.1 Confusion Matrix

A confusion matrix is a visual representation of a performance of a classification model.



		True Values	
		Positive	Negative
Predicted Values	Positive	True Positive	False Positive
	Negative	False Negative	True Negative

The picture above shows a confusion matrix for a binary classifier.
- True Positives contains a number (or percentage) of entries in the dataset, that were classified as being positive, which in turn are positive.
- True Negatives contains a number (or percentage) of entries in the dataset, that were classified as being negative, which in turn are negative.
- False Positives contains a number (or percentage) of entries in the dataset, that were classified as being positive, which in turn are negative. This is also called Type-1 Error.

- False Negative contains a number (or percentage) of entries in the dataset, that were classified as being negative, which in turn are positive. This is also called Type-2 Error.

The dataset used in the project contains seven different categories, therefore the confusion matrix will be a 7x7 matrix. Using the confusion matrix will be helpful to figure out if the model is particularly bad at classifying a certain group of emotions.

1.1.2 ROC and AUC

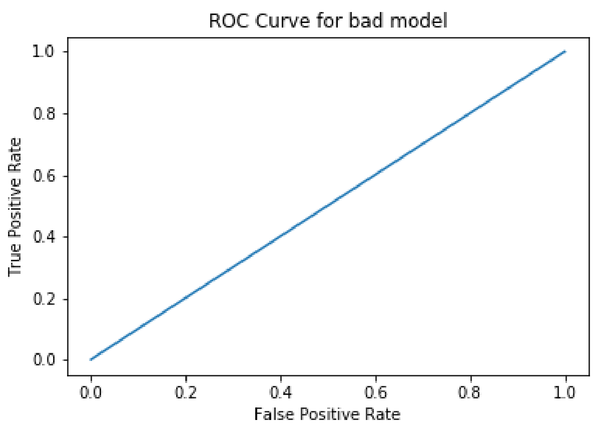
ROC (receiver operating characteristic curve) is a plot of True Positive Rate (TPR) on the Y – axis and False Positive Rate (FPR) on the X-axis for all classification threshholds.

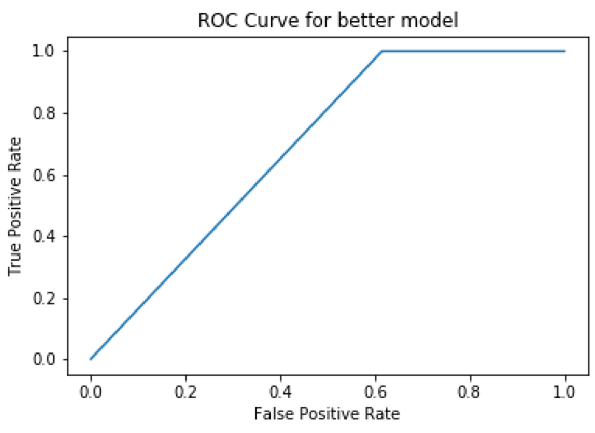
$$TPR = \frac{TP}{TP+FN}$$

True Positive Rate is the rate of True Positives and the sum of True Positives and False Negative.

$$FPR = \frac{FP}{FP+TN}$$

False Positive Rate is the rate of False Positives and the sum of False Positives and True Negatives.





The ROC Curve for a bad model draws a 45 degree line through the origin. A better model always draws points above this line. In order to compare different models the area under the ROC (AUC) is calculated for each model and compared. The values of the AUC can lie between 0.5 (45 degree line) and 1 (a perfect prediction model).

To calculate the AUC for a non binary model the so called One-Vs-All approach is taken. In this approach for each category the classification is seen as a binary problem where all other categories are seen as one “other” category. Therefore in this project 7 AUC values are going to be calculated for each model and compared individually.

2 Analysis

2.1 Data Exploration

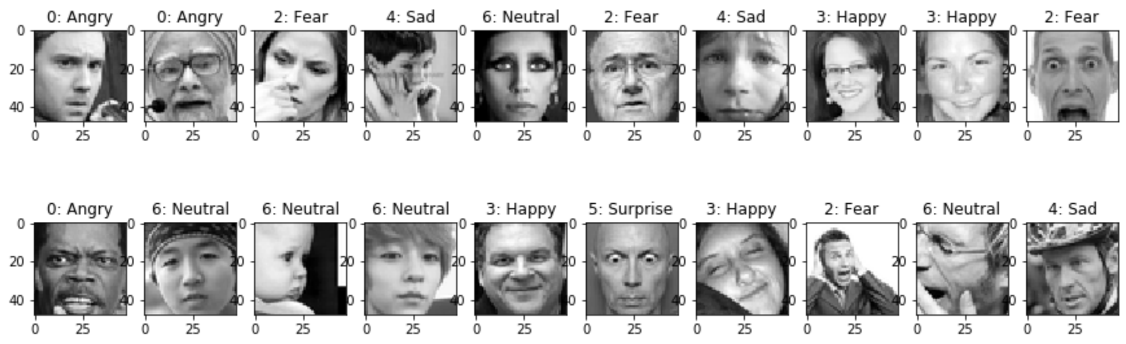
This project requires a large collection of images with corresponding labeled categories of facial expressions. The kaggle competition “Challenges in Representation Learning: Facial Expression Recognition Challenge” provides a dataset which consists of 48x48 pixel grayscale images. The dataset consists of 35887 examples, each containing emotion, pixels and usage variables.

- Emotion is a categorical variable represented by a numeric value.

0: Angry
1: Disgust
2: Fear
3: Happy
4: Sad
5: Surprise
6: Neutral

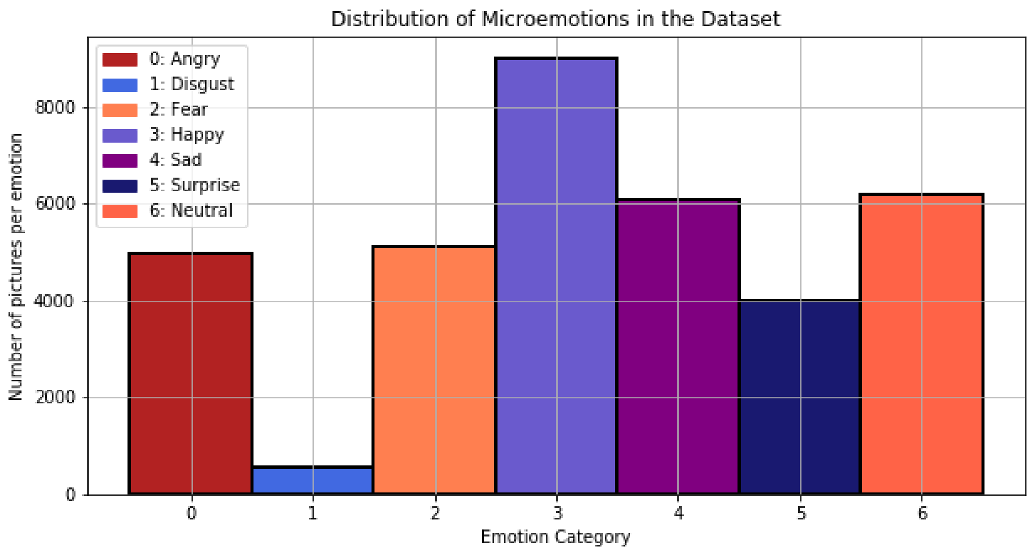
- Pixel is a string variable containing 48X48 white-space delimited values between 0 and 255. Each value in the string represents a greyscale value for a specific pixel in the picture. The location of the pixel corresponds to the location of the value in the string.

- Usage indicates whether the picture is intended for training or testing purposes.



The above example shows 20 pictures in the dataset. As the classification of the emotions of the dataset were done using the Google API and corrected by humans and humans estimation is expected to be 65+/-5% accurate, over 30% of the pictures might be inaccurately classified[3]. For example the first picture in the first and second row are both classified as angry. While the second row picture might clearly be seen as "angry" the first one is not different from a neutral expression.

2.2 Exploratory Visualization



The above plot shows the distribution of micro emotions in the dataset. The data is strongly unbalanced, especially when it comes to the disgust category. This distribution needs to be taken into account, when dividing the data into different

3 See Goodfellow et. al., 2013, Challenges in Representation Learning: A report on three machine learning contests

buckets. If a random split is taken, then the model might never encounter a large sample of the "Disgust" sample while training, therefore only optimizing for other the correct classification of other categories.

2.3 Algorithms and Techniques

Convolutional neural networks (CNN) are well known for their performance in visual classification. The ImageNet challenge, an image classification competition, brought forth AlexNet, VGGNet, ResNet and so on, deep CNN architectures that were able to perform extremely well in the competition.

Considering that CNNs have outperformed other architectures in visual classification, CNNs are a logical choice for the classification of facial expressions. Grayscale images are going to be used as input into the neural network, while a softmax activation function will be used for the calculation of the probability of a certain expression.

The deep learning library Keras will be used for the construction and training of the CNN. Tensorflow will be used as the underlying backend architecture. The following parameters can be tweaked in Keras in order to increase the performance of the CNN.

1. *Type and Parameters of Layers*

- *Fully connected: Classical Neural Network layer accepting a vector as input and creating an output, which is used in an activation function. A number of output neurons is used as input into the neural network. This layer is used at the end of the CNN.*
- *Activation Function: A function, which introduces non-linearity. For example Sigmoid or ReLU.*
- *Dropout: Determines the probability with which some of the nodes are ignored during an epoch in order to avoid overfitting.*
- *Convolutional: A locally connected neural network used for image classification. Unlike a fully connected layer, a tensor is used as input in order to detect spacial relationships. The two main parameters are stride and padding.*
- *Flatten: Creates a vector from a matrix in order for the CNN output to be used in a fully connected layer.*
- *Max Pooling Layer: Reduces the dimensionality of the CNN in order to avoid overfitting.*

2. *Optimizers*

- *Type of Optimizers: Which type of optimizer is used in gradient descent. For example RMSProp and Adam.*

- *Optimizer Parameters: Learning rate and decay rate of gradient descent,*

- *Loss Function: Categorical Cross-Entropy is going to be used in all models.*

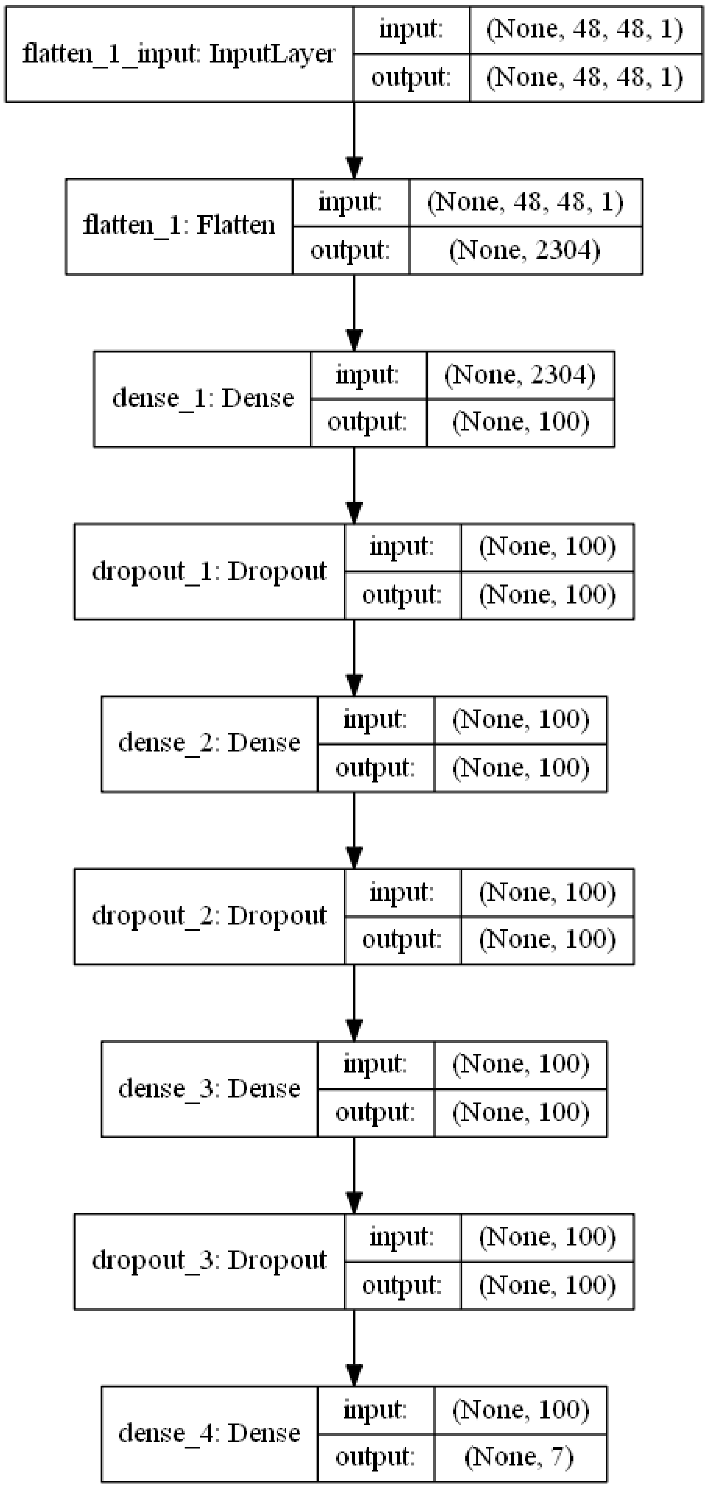
3. *Training parameters:*

- *Number of epochs*

- *Batch Size*

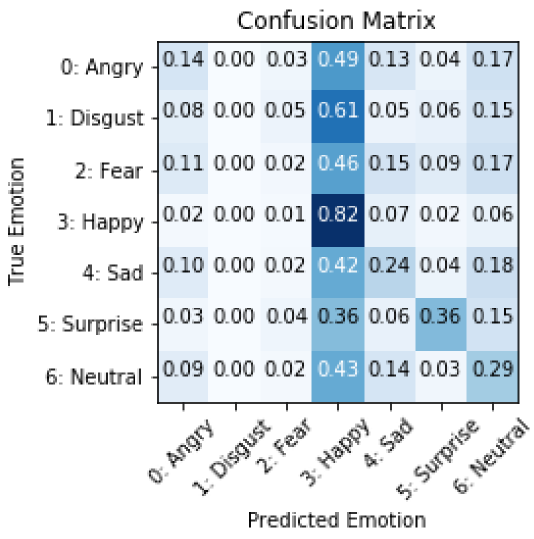
2.4 Benchmark

A vanilla multi layer perceptron (MLP) model is going to be used as the benchmark model in this project. The MLP is a relatively simple form of a neural network. It was not designed for the sole purpose of visual detection, but can still be used for classification purposes. As a CNN is primarily used for the classification of pictures the CNN is expected to outperform the MLP.



The 48x48 vector is going to be used as input in the MLP. Three hidden layers with 100 neurons and a ReLU activation function each used in the model. Between each hidden layer a Dropout layer is introduced with a probability of 20% of being deactivated.

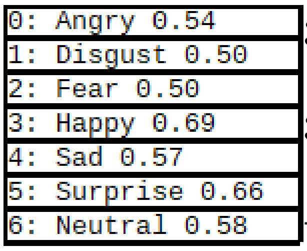
A softmax activation function in the output layer was employed. Categorical cross-entropy was utilized as a loss function. RMSProp with standard parameters was used for gradient descent.



	0: Angry	1: Disgust	2: Fear	3: Happy	4: Sad	5: Surprise	6: Neutral
0: Angry	0.14	0.00	0.03	0.49	0.13	0.04	0.17
1: Disgust	0.08	0.00	0.05	0.61	0.05	0.06	0.15
2: Fear	0.11	0.00	0.02	0.46	0.15	0.09	0.17
3: Happy	0.02	0.00	0.01	0.82	0.07	0.02	0.06
4: Sad	0.10	0.00	0.02	0.42	0.24	0.04	0.18
5: Surprise	0.03	0.00	0.04	0.36	0.06	0.36	0.15
6: Neutral	0.09	0.00	0.02	0.43	0.14	0.03	0.29

The above confusion matrix indicates that the "happy" micro expression dominates the model. For many categories, especially "disgust", the model falsely classifies many observations as "happy". The result is not surprising, as the "happy" is more often encountered in the dataset than any other category.

The AUC for each individual category confirms the results of the confusion matrix. The model has an AUC value of 0.5 for the disgust category, meaning that this MLP model is really bad at classifying the micro expression. The AUC for happy is on the other hand relatively high.



```
0: Angry 0.54
1: Disgust 0.50
2: Fear 0.50
3: Happy 0.69
4: Sad 0.57
5: Surprise 0.66
6: Neutral 0.58
```

3 Methodology

3.1 Data Preprocessing

The pixel data provided in the dataset is saved in a single column as a string value delimited by white-space. In order for the pixel values to be used in a machine learning architecture the string value needed to be transformed into a 4 dimensional numpy array in the first step.

The output of the data was a 4 dimensional array with the shape of (35887, 48, 48, 1), where each number has the following interpretation.

* Number of pictures (35887)

* Length of the picture (48 pixels)

* Width of the picture (48 pixels)

* Depth of the picture (1)

The depth dimension is set to 1 as the dataset only contains greyscale information and there is no need for a 3 dimensional color space. This additional dimension is still necessary, especially for transfer learning.

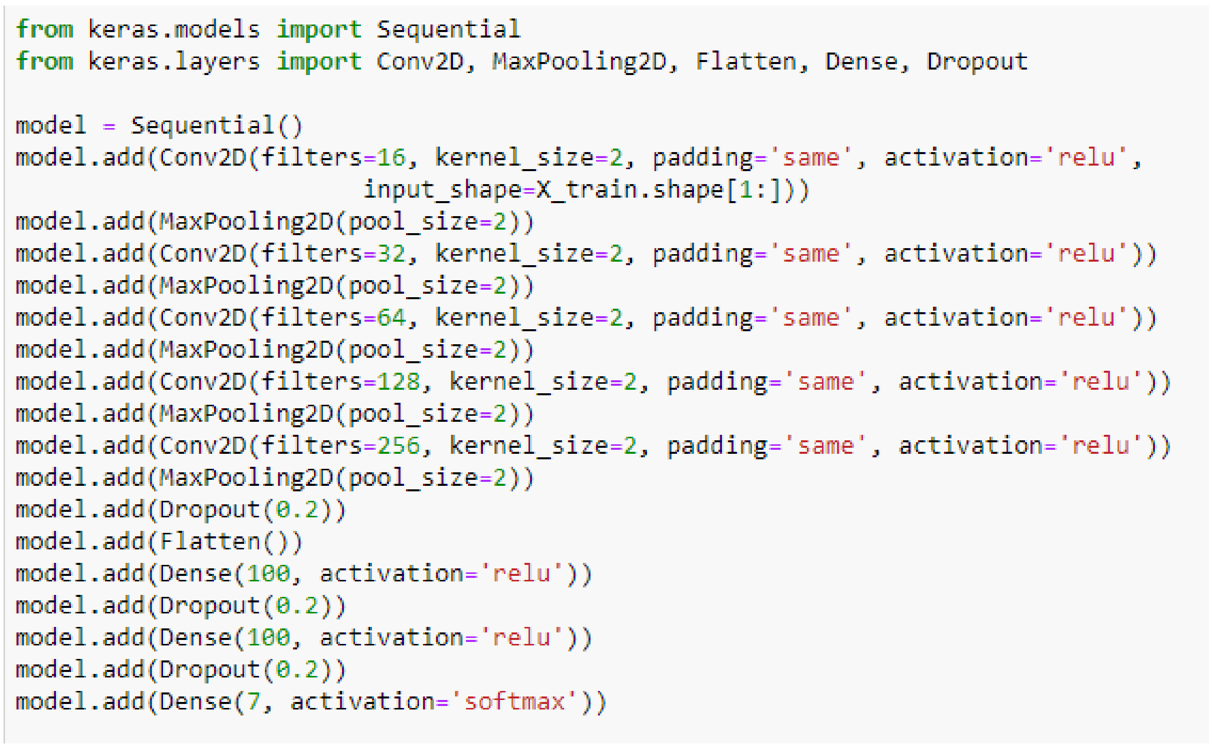
In order to reduce overfitting the data is split into a training, cross-validation and testing datasets. The "usage" flag is ignored, as there is no classification for cross-validation. Due to the imbalance in the distribution of the micro emotions the dataset is stratified while splitting the dataset in order to keep the distribution similar in the three datasets. In the first step the data is split into a training and testing buckets using an 80/20 ratio. The training set is additionally split into a training and cross-validation buckets using an 80/20 ratio once again.

The pixel values are normalized by dividing each value by 255.

Finally in order for the micro expression categories to be utilized in the neural network one hot encoding was performed.

3.2 Implementation

Keras was utilized throughout the project to generate a CNN architecture. The architecture always followed the same structure, where preprocessed numpy array was input into the convolutional layer in the first step. After each convolutional layer a Max Pooling layer was added. Altogether 5 CNN and 5 Pooling layers were used in the architecture. That way the length and the width of the dimension decreased while the depth increased gradually. Finally a succession of dropout and fully connected layers were implemented, where the last layer included a softmax activation function with 7 categories.



```
from keras.models import Sequential
from keras.layers import Conv2D, MaxPooling2D, Flatten, Dense, Dropout

model = Sequential()
model.add(Conv2D(filters=16, kernel_size=2, padding='same', activation='relu',
                        input_shape=X_train.shape[1:]))
model.add(MaxPooling2D(pool_size=2))
model.add(Conv2D(filters=32, kernel_size=2, padding='same', activation='relu'))
model.add(MaxPooling2D(pool_size=2))
model.add(Conv2D(filters=64, kernel_size=2, padding='same', activation='relu'))
model.add(MaxPooling2D(pool_size=2))
model.add(Conv2D(filters=128, kernel_size=2, padding='same', activation='relu'))
model.add(MaxPooling2D(pool_size=2))
model.add(Conv2D(filters=256, kernel_size=2, padding='same', activation='relu'))
model.add(MaxPooling2D(pool_size=2))
model.add(Dropout(0.2))
model.add(Flatten())
model.add(Dense(100, activation='relu'))
model.add(Dropout(0.2))
model.add(Dense(100, activation='relu'))
model.add(Dropout(0.2))
model.add(Dense(7, activation='softmax'))
```

The final implementation looked as follows.

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_1 (Conv2D)            (None, 48, 48, 16)        80
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 24, 24, 16)        0
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 24, 24, 32)        2080
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 12, 12, 32)        0
_________________________________________________________________
conv2d_3 (Conv2D)            (None, 12, 12, 64)        8256
_________________________________________________________________
max_pooling2d_3 (MaxPooling2 (None, 6, 6, 64)          0
_________________________________________________________________
conv2d_4 (Conv2D)            (None, 6, 6, 128)         32896
_________________________________________________________________
max_pooling2d_4 (MaxPooling2 (None, 3, 3, 128)         0
_________________________________________________________________
conv2d_5 (Conv2D)            (None, 3, 3, 256)         131328
_________________________________________________________________
max_pooling2d_5 (MaxPooling2 (None, 1, 1, 256)         0
_________________________________________________________________
dropout_4 (Dropout)          (None, 1, 1, 256)         0
_________________________________________________________________
flatten_2 (Flatten)          (None, 256)               0
_________________________________________________________________
dense_5 (Dense)              (None, 100)               25700
_________________________________________________________________
dropout_5 (Dropout)          (None, 100)               0
_________________________________________________________________
dense_6 (Dense)              (None, 100)               10100
_________________________________________________________________
dropout_6 (Dropout)          (None, 100)               0
_________________________________________________________________
dense_7 (Dense)              (None, 7)                 707
=================================================================
Total params: 211,147
Trainable params: 211,147
Non-trainable params: 0
_________________________________________________________________
```

```python
#Function to print out the Area under the ROC values using the one-vs-all approac
from sklearn.metrics import auc
from sklearn.metrics import roc_curve

def roc_auc(pred, true, num_categories):
    for i in range(num_categories):
        class_dictionary = {x:0 if x!=i else 1 for x in range(num_categories) }
        pred_class = np.array([class_dictionary[i] for i in pred])
        true_class = np.array([class_dictionary[i] for i in true])
        fpr, tpr, thresholds = roc_curve(true_class, pred_class)
        auc_category = auc(fpr, tpr)
        auc_str = f"{auc_category:5.2f} "
        print(emotions[i] + "" + auc_str)
```

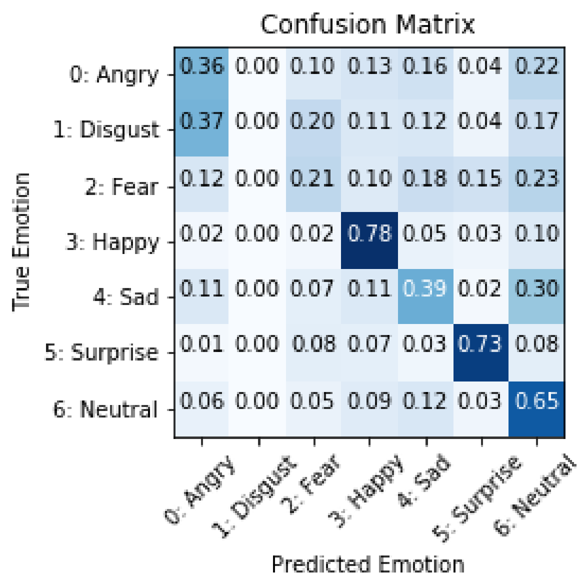
The roc_auc function was implemented in order to be able to generate the roc values using the one-vs-all approach.

```python
#function for the confusion matrix
#the implementation is based on
#http://scikit-learn.org/stable/auto_examples/model_selection/
#plot_confusion_matrix.html#sphx-glr-auto-examples-model-selection-plot-confusion-matrix-py
from sklearn.metrics import confusion_matrix
import itertools
def conf_matrix(true, prediction, path):

    cm = confusion_matrix(true, prediction)
    cm = cm.astype('float') / cm.sum(axis=1)[:, np.newaxis]

    plt.imshow(cm, plt.cm.Blues)
    plt.title('Confusion Matrix')
    tick_marks = np.arange(len(emotions))
    plt.xticks(tick_marks, emotions, rotation=45)
    plt.yticks(tick_marks, emotions)


    thresh = cm.max() / 2.
    for i, j in itertools.product(range(cm.shape[0]), range(cm.shape[1])):
        plt.text(j, i, format(cm[i, j], '.2f'),
                 horizontalalignment="center",
                 color="white" if cm[i, j] > thresh else "black")


    plt.ylabel('True Emotion')
    plt.xlabel('Predicted Emotion')
    plt.tight_layout()
    plt.savefig(path)
    plt.show()
```

The conf_matrix was a function based on the sklearn documentation. The function generated a confusion matrix using normalized (percentages) values.

3.3 Refinement

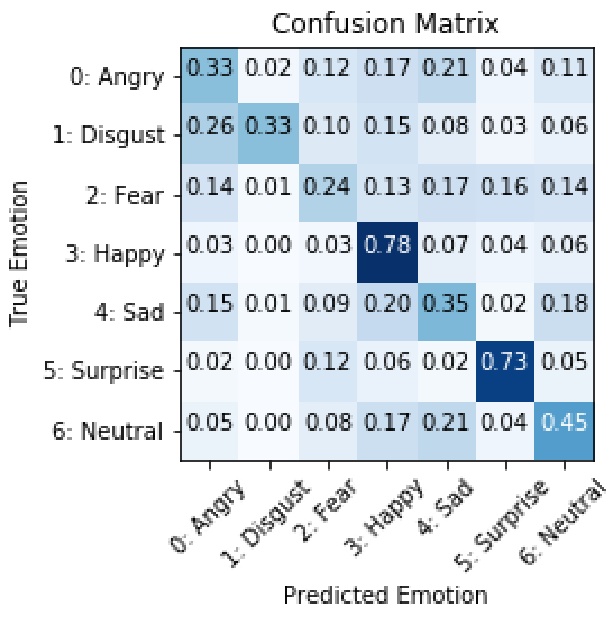
The first model implemented in the project was the model detailed in 3.2 without any augmentations or any type of regularizations or parameter adjustments.



True Emotion \ Predicted Emotion	0: Angry	1: Disgust	2: Fear	3: Happy	4: Sad	5: Surprise	6: Neutral
0: Angry	0.36	0.00	0.10	0.13	0.16	0.04	0.22
1: Disgust	0.37	0.00	0.20	0.11	0.12	0.04	0.17
2: Fear	0.12	0.00	0.21	0.10	0.18	0.15	0.23
3: Happy	0.02	0.00	0.02	0.78	0.05	0.03	0.10
4: Sad	0.11	0.00	0.07	0.11	0.39	0.02	0.30
5: Surprise	0.01	0.00	0.08	0.07	0.03	0.73	0.08
6: Neutral	0.06	0.00	0.05	0.09	0.12	0.03	0.65

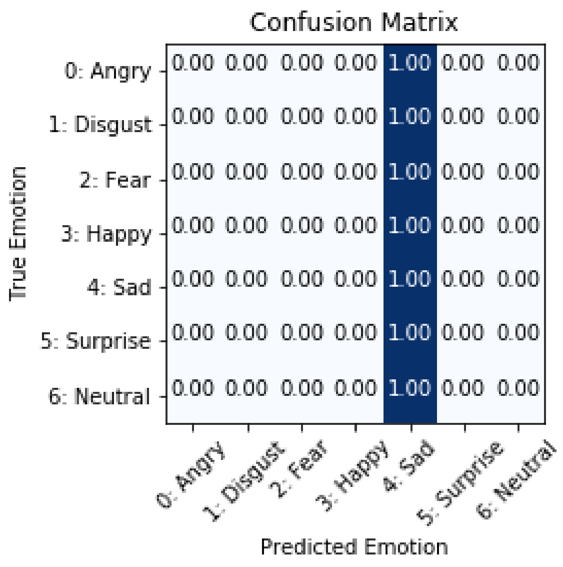
However when looking at the results of the confusion matrix it becomes quite clear, that not a single picture in the testing dataset was classified as “disgust”.

In order to overcome the problem different methodologies were tested.



True Emotion \ Predicted Emotion	0: Angry	1: Disgust	2: Fear	3: Happy	4: Sad	5: Surprise	6: Neutral
0: Angry	0.33	0.02	0.12	0.17	0.21	0.04	0.11
1: Disgust	0.26	0.33	0.10	0.15	0.08	0.03	0.06
2: Fear	0.14	0.01	0.24	0.13	0.17	0.16	0.14
3: Happy	0.03	0.00	0.03	0.78	0.07	0.04	0.06
4: Sad	0.15	0.01	0.09	0.20	0.35	0.02	0.18
5: Surprise	0.02	0.00	0.12	0.06	0.02	0.73	0.05
6: Neutral	0.05	0.00	0.08	0.17	0.21	0.04	0.45

As seen in the confusion matrix above when adding regularization and adding the parameter class_weight, that takes into account the distribution of micro expressions improved the quality of the classification of disgust, but reduced the classification quality of other expressions.



Using transfer learning on the other hand when using the weights of Imagenet did not produce the expected results. There were some hurdles that had to be overcome in order to make the model work. The biggest was the transfer of one dimensional greyscale image into 3 dimensional space. This was done by copying the greyscale channel 3 times. As imagenet was trained using colored images and a colored image was expected as input the result might be explained by an unfamiliar input.

4 Results

4.1 Model Evaluation and Validation

The final model that was used in this project was based on data augmentation. Data augmentation distorts original pictures in some ways with some probability in order for the model to learn the underlying patterns and thus reduce the danger of overfitting.

The following parameters were chosen due to the best performance out of those experimented with:

1. The activation function for all, except the last layer is ReLu

2. The activation function for the last layer is Softmax, in order to calculate the probability of a certain micro emotion

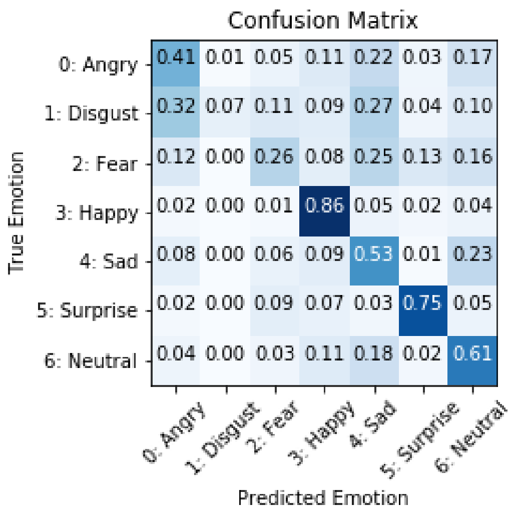
3. RMSProp is used as optimizer

4. Batch Size is 128

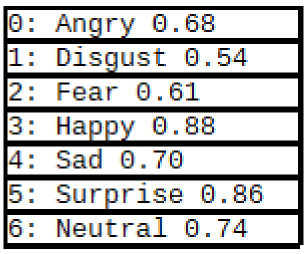
5. The number of epochs was 100, although early stopping with a patience factor of 10 was employed

As mentioned above data augmentation was used in order to make the model less sensitive to overfitting. For that purpose the images, which were perfectly aligned were shifted in the width and height. Additionally a horizontal_flip was set to true, so that the input was randomly flipped on the horizontal axis.

During preprocessing 20 percent of the data was separated and not used during training or validation step. That data was used for the final calculation of the confusion matrix or AUC values of the model and was not previously seen during training.



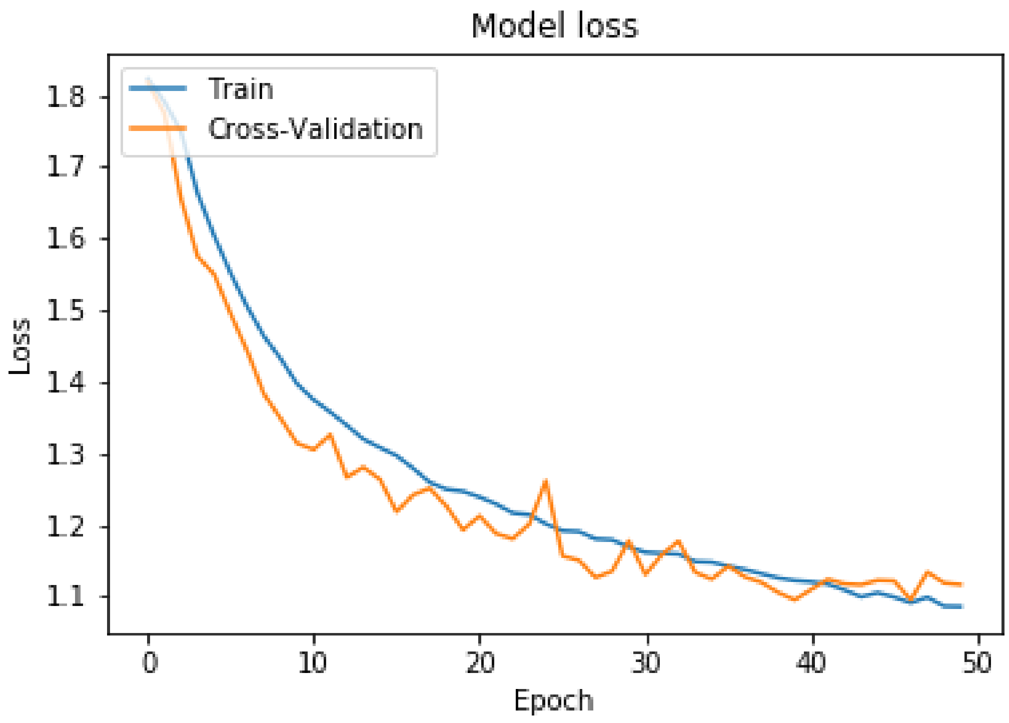
	0: Angry	1: Disgust	2: Fear	3: Happy	4: Sad	5: Surprise	6: Neutral
0: Angry	0.41	0.01	0.05	0.11	0.22	0.03	0.17
1: Disgust	0.32	0.07	0.11	0.09	0.27	0.04	0.10
2: Fear	0.12	0.00	0.26	0.08	0.25	0.13	0.16
3: Happy	0.02	0.00	0.01	0.86	0.05	0.02	0.04
4: Sad	0.08	0.00	0.06	0.09	0.53	0.01	0.23
5: Surprise	0.02	0.00	0.09	0.07	0.03	0.75	0.05
6: Neutral	0.04	0.00	0.03	0.11	0.18	0.02	0.61



```
0: Angry 0.68
1: Disgust 0.54
2: Fear 0.61
3: Happy 0.88
4: Sad 0.70
5: Surprise 0.86
6: Neutral 0.74
```

AUC for the augmented model

The model performs relatively well when faced with happy, surprise, sad or neutral emotions, but still has still trouble with anger, disgust and fear.



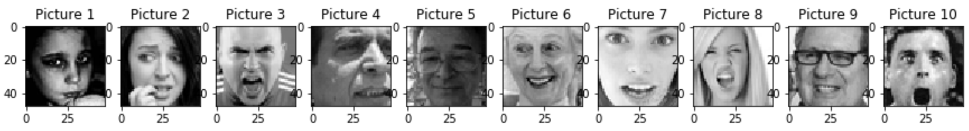
The graph above underlies the robustness of the model. When the model adjusts the weights by utilizing the data in the training sample in order to reduce categorical cross entropy the cross-validation loss tends to decrease as well, which might be an indication that there is no strong overfitting.

4.2 Justification

A direct comparison of the confusion matrix and AUC values between the original MLP model and the CNN model, which used augmented images, the CNN model beats the MLP model in each of the 7 categories. That said the performance for some categories (e.g. disgust) is not sufficient to be used in a live setting. Regularization for example can significantly improve the performance of some categories, but there is a tradeoff for other categories. The solution could not be used by any sort of machine, as a received micro emotion might trigger a different reaction, than the desired one.

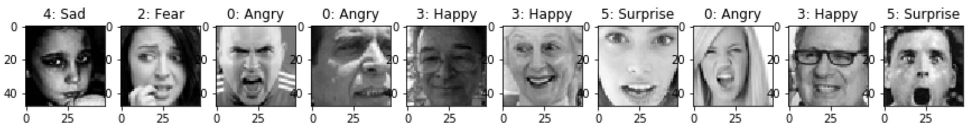
5 Conclusion

5.1 Free-Form Visualization



In order to test the quality of the possible performance of the model I tried to predict the labels of the dataset. For that purpose I drew 10 pictures of the dataset without looking at the labels. I assigned each picture a category and compared the result to the labels in the dataset.

Picture 1	Neutral
Picture 2	Fear
Picture 3	Angry
Picture 4	Happy
Picture 5	Happy
Picture 6	Happy
Picture 7	Surprise
Picture 8	Angry
Picture 9	Happy
Picture 10	Surprise



The realization is, that it is not easy to categorize the emotional state of a person. Two out of the 10 pictures were misclassified by me. At this point it is not quite clear if the quality of the dataset is not ideal or if it is generally not that easy even for some human beings to correctly classify the emotion of a random person without knowing the context in which the emotion is created.

5.2 Reflection

The following steps were undertaken to achieve the final results.

1. The dataset from Kaggle was downloaded
2. The data was analyzed to determine the necessary preprocessing steps
3. The necessary performance measure had to be implemented
4. The data was prepropesecced and divided into the training, cross-validation and testing datasets to avoid overfitting
5. A benchmark MLP model was created
6. Different models were trained and compared

A) A simple CNN Model

B) CNN model with regularization

C) CNN model with data augmentations

D) CNN based on transfer learning

The most complex and surprising result of the project was the difficulty to make transfer learning to work. Even at the end of the project I did not manage to create a functioning model which is based on a pretrained model.

5.3 Improvement

The performance of CNNs scales with the number of pictures in the dataset. Compared to Imagenet for example 30,000 pictures seem to be very few to achieve a good performance. Therefore a collection of additional labeled pictures might be a good approach.

Deeper neural networks with additional layers might also have an impact on performance.

Finally different activation functions might have produced different results.