Deep Convolutional Neural Networks for Emotion Detection under the Apsect of Image Scaling Attacks

Daniel Röder* (i6261485), Justus-Jonas Erker* (i6263960)

**MSc. Artifcial Intelligence Student, Department of Data Science and Knowledge Engineering, Maastricht University*

Abstract

This paper is part of the CNN Assignment for the Master in Artificial Intelligence Course: Computer Vision at the University of Maastricht. It demonstrates how emotions based on images can be classified using Deep Convolutional Neural Networks (DCNN). It is compared to a fine-tuned state-of-the-art model VGGnet which it is able to outperform. Apart from that the model is investigated under the aspect of scaling image attacks.

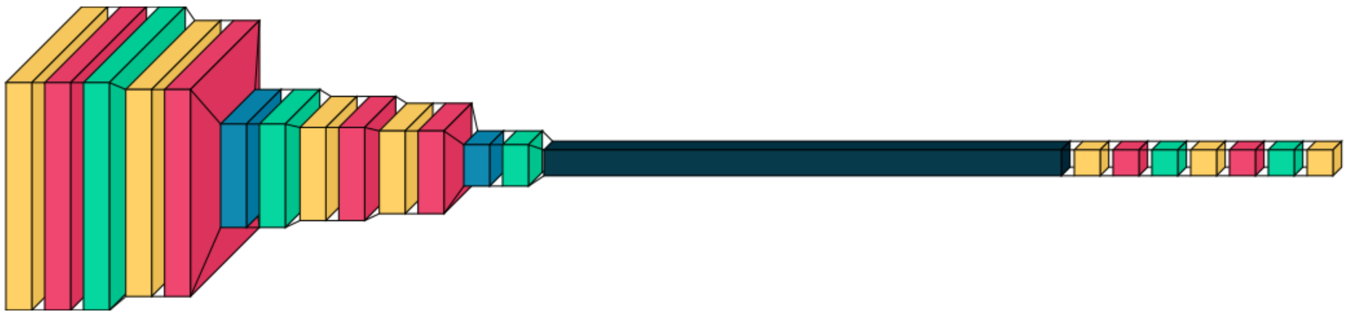
Figure 1: Deep Convolutional Neural Network with where Yellow = Conv2D layer, red = Batch normalization, green = Dropout, Blue = maxPooling2D, dark gray= Dense Layers and the last 7 boxes the output layer

1 Introduction

In this report we will focus on experimenting with different aspects of a Convolutional Neural Network (CNN) for multi-label classification in the domain of facial emotion detection. For the sake of examining the effect of different CNN architectures a group of models was created which will be discussed in the upcoming paragraphs. We will thereby focus on providing a rational for architecture used in our top performing model. Additionally we provide an another experiment from the domain of Image Scale Attacks in section 3.7.

2 Dataset

We use the facial emotion dataset from the Kaggle challenge provided in the assignment manual. The dataset contains 35,887 48x48 images each labeled with one of seven emotion classes. A first analysis of the dataset revealed three possible sources of errors which will be shortly discussed below.
A first analysis revealed the fact that the dataset is heavily unbalanced in respect to class 1 ("Disgust"). We will try to alleviate this problem by doing a thorough evaluation of the model results on one of the subsequent chapters.
Secondly we found that the dataset contains some broken images, which just seem to display part of a warning sign on an otherwise black background. We decided to remove those 25 images that we identified using thresholding on the amount of black pixels.
Another risk factor lays in some images still having a watermark from the stock imagery website on them.

2.1 Data preprocessing

Before feeding the dataset into our model it is prepared using a some basic preprocessing methods shortly presented below. Each image is converted into an Numpy array and then reshaped to (48,48,1) to fit the input of the upcoming models. Additionally each picture is normalized such that its pixel values are between 0 and 1 to allow for faster convergence.

3 Models

We chose Keras as our machine learning framework. Keras offers us all the modification options needed for this report while at the same time having an easy to use interface. Beginning with a baseline model we want to give an intuition for the different layers and methods being applied.

3.1 Baseline building blocks

Using the building blocks described in the following paragraph we created a multitude of models each slightly changing the amount of those blocks used or some internal parameters to examine there effect on the model performance. In the following paragraphs parts like Dropout are mentioned as layers due there usage in Keras.
The first building block is centered around a convolutional layer extracting features from the input by apply-

ing a set of filters. Next as a transition between the linear Conv2D layer and the non linear activation function we introduce a batch normalization layer.This allows some degree of independents form the scaling of the weights and should stabilize the learning process. Next we decided for each of the main layer (Conv2D,Dense) with exception of the output layer to use a ReLu activation function due to its low computational costs and its ability to introduce non-linearity. The results are then fed through a max pooling layer that reduces the numbers of features and makes the overall model more robust in regards to the position of features. Finally a dropout layer is added to the building block which randomly deactivated neurons during training therefore making the CNN less dependent on specific weights.
Block two and three are both centered around a Dense layer where block two introduces a batch normalization between dense layer and activation layer. Before the data from the convolution blocks can be fed into the dense layers it is flattened. Block three then uses a softmax activation function with seven output neurons corresponding to the seven emotions.

3.2 Model Training

The models are trained with several call backs such as EarlyStopping, ReduceLROnPlateau as well as ModelCheckpoints. EarlyStopping is a monitoring tool that stops the training process as soon as the validation loss does not improve for a certain amounts of epochs (here: 10) called patience to avoid overfitting. ReduceLROnPlateau on the other hand reduces the learning rate in case the validation loss does not improve after 5 epochs. Reducing the learning rate helps the gradient descent to not overshoot the goal as it will do smaller steps near the minima. The last callback ModelCheckpoint saves the best model in case the model gets worse over the training time. Various optimizers have been tried as part of the hyper parameter tuning process with different learning rates. Those include, Adam, Nadam as well as SGD (Stochastic Gradient Decent) with learning rates of 0.001, 0.008 and 0.01. Categorical cross entropy with a batch size of 32 and 64 is applied and all models are trained up to 250 epochs, keeping in mind the early stopping as well as the reduce learning rate on plateau. The best parameter of the model were observed with Nadam (learning rate 0.008) combined with a batch size of 32. All models are trained with a validation split of 0.1.

3.3 Experimenting with Building Blocks

In this section we build an assortment of different models based on the building blocks described in 3.1 to understand the effect different architectures have on the model performance. The result of a selection of models can be found below.

Table 1: Model testing

model	acc val	acc test	f1
123	0.51	0.51	0.51
1123	0.562	0.56	0.55
11123	0.573	0.58	0.57
111123	0.61	0.60	0.59
1*1*23	0.59	0.58	0.58
11223	0.57	0.58	0.58

* multiple Conv2D stacked in one block

The model name can be decoded as the building blocks described before (1-3). For this training 10% of the dataset where used for testing, 9% where used for validation and the remaining data was used for training. From this basic exploratory experiment we conclude that stacking multiple convolution layers as well as dense layers have a beneficial effect on the model performance. It should also be noted that this effect is not linear therefore adding additional layers yields diminishing rewards

3.4 Model Architecture

As it can be seen in the figure 1 the final model consists of 2 Convolutional Layers with 32 filters and a kernel size of 5x5 followed by a Max pooling layer. After that follow 2 Convolutional layer with 64 filters with a kernel size of 3x3 and a final max pooling layer. Both blocks use Batch Normalization as well as Dropout to fight against overfitting. The overall intend behind using in the first block with fewer filters, but a larger kernel is that it should focus on larger more general features of the image while the second block with more filters and a smaller kernel size is focusing more on fine details in those more general features. This can be observed by visualizing the convolution layers as follows in figure 2.

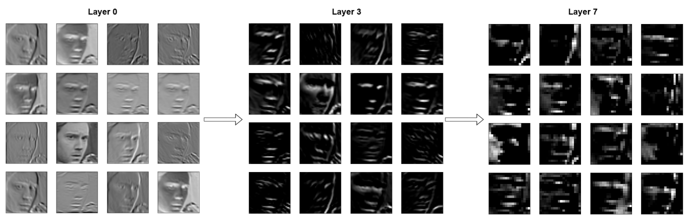

Figure 2: Convolutions of the CNN visualized

While it is hard to determine whether the model really learns this way, the trend can be observed. After those two blocks of convolutional layers a flatten layer squeezes the data into two following Dense layers with 64 neurons and then to 7 output neurons. Overall Dropout is used and increased in the later layer (fine-tuned by dropout "grid search") in between every layer to fight against massively overfitting that has been challenging throughout all experiments.

3.5 Model Evaluation

The best model was able to achieve a validation accuracy of about 63% with a loss of 1.02 as figure 4 and 3 shows. It was able to achieve a test accuracy of 61.8% and a weighted f_1 score of 0.61235.

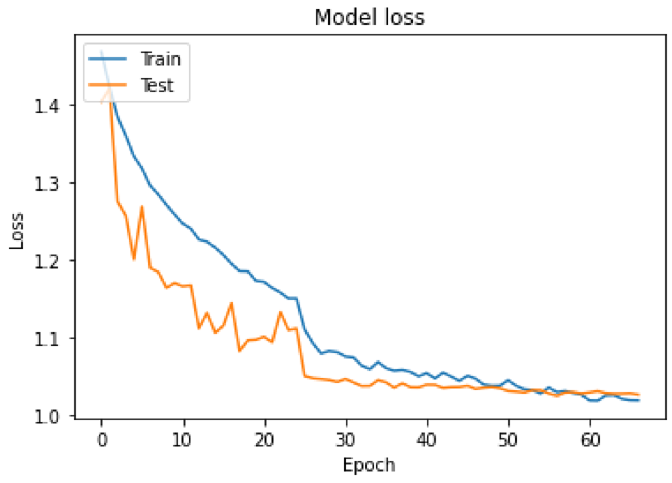

Figure 3: Loss of CNN over 69 epochs

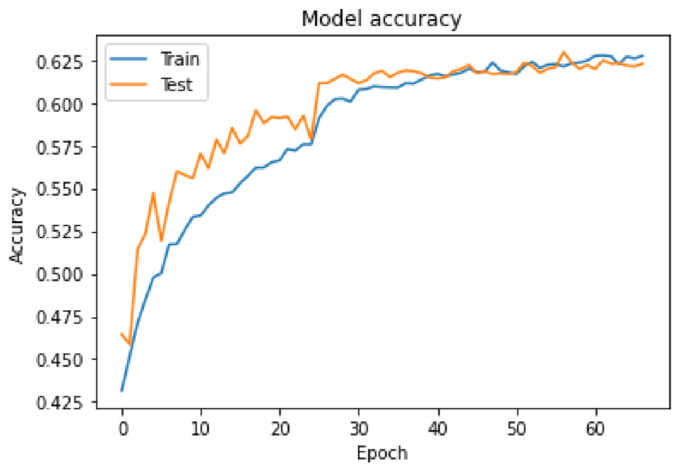

Figure 4: Loss of CNN over 69 epochs

Early stopping caused the model to stop after 69 epochs. The confusion matrix in figure 5 shows that the label happiness performance best while having the most frequent samples.

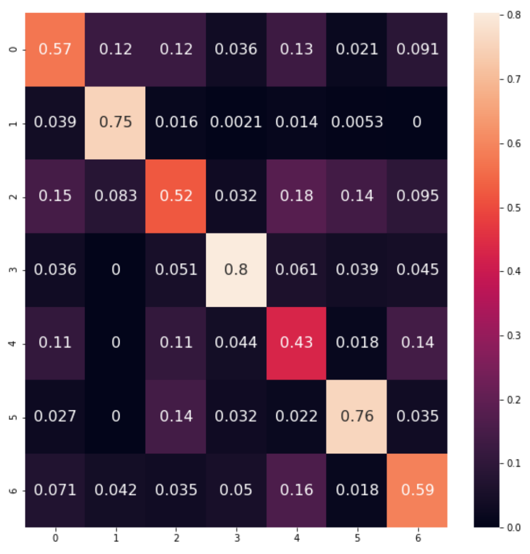

Figure 5: Confusion matrix of CNN on test data

Overall it seems like that emotions that are less expressive as sadness, neutral and fear are harder to classify than very expressive ones as surprised or disgusted.

3.6 Experiments with VGGnet

As an additional experiment the VGGnet 16 was trained with the same training data set using several approaches. The first approach which worked very poorly with a validation accuracy of less than 50% fine-tuned the model by adding two dense layers of 64 neurons after the Flatten layer with all remaining VGGnet layer frozen (similar to our CNN approach). In the second approach the same model structure was used but with all layers set to trainable. Despite using a small learning rate and a very high dropout the model was trending towards high overfitting with a validation accuracy of about 55%. However, the best approach was with same structure but making the last 3 convolutional layers trainable (while all others were frozen) with adding batch normalization between those as well as dropout followed by the same dense layers as for the other structures. This model was able to achieve an overall validation accuracy of 58% despite massive overfitting.

3.7 Image Scaling Attack

Image scaling attacks "automatically generate camouflage images whose visual semantics change dramatically after scaling." [Xiao et al.(2019)Xiao, Chen, Shen, Chen, and Li]. Applied is the algorithm introduced in the paper "Seeing is Not Believing: Camouflage Attacks on Image Scaling Algorithms" by [Xiao et al.(2019)Xiao, Chen, Shen, Chen, and Li] which takes a target image (the one which should be manipulated) as well as a source image (which class we want it to classify) which generates an output image that looks the same as the target image but has the class of the source image as the following figure 6 visualizes.

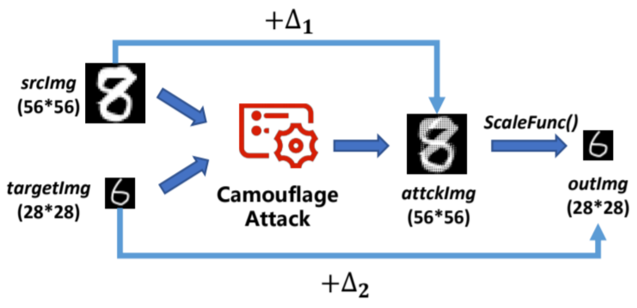

Figure 6: Image Scaling attack: Automatic attack image crafting [Xiao et al.(2019)Xiao, Chen, Shen, Chen, and Li]

After the model is trained, image scaling attacks on sample points of the test data set are executed. While not all attacks are successful, evidence was collected that the model is not prune to attacks as the following example shows if figure 7 shows.

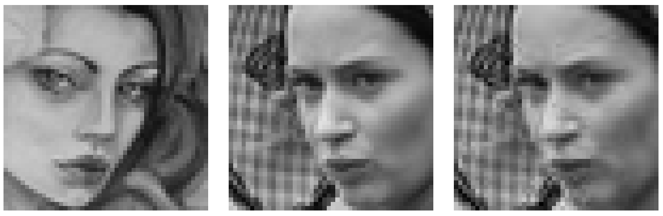

Figure 7: Left the source image with label: sad, middle the target image with label: angry, right the attack with label: sad

As it can be observed the images are very similar, but the attack image looks a little more blurry due to the scaling attack. While it might not be very problematic to fake an emotion in a model like this, but as image classification or recognition is more and more common in critical areas as autonomous cars, such attacks can have drastic consequences. Nevertheless, there are several approaches that can detect those attacks using color histogram or color scattering of the feature vectors.

2

References

[Xiao et al.(2019)Xiao, Chen, Shen, Chen, and Li] Qixue Xiao, Yufei Chen, Chao Shen, Yu Chen, and Kang Li. Seeing is not believing: Camouflage attacks on image scaling algorithms. In *28th USENIX Security Symposium (USENIX Security 19)*, pages 443–460, Santa Clara, CA, August 2019. USENIX Association. ISBN 978-1-939133-06-9. URL `https://www.usenix.org/conference/usenixsecurity19/presentation/xiao`.