Maastricht University

# Computer Vision

## Assignment 2 : Deep Learning

*Authors*
Eliott Simon,
Foti Kerkeshi

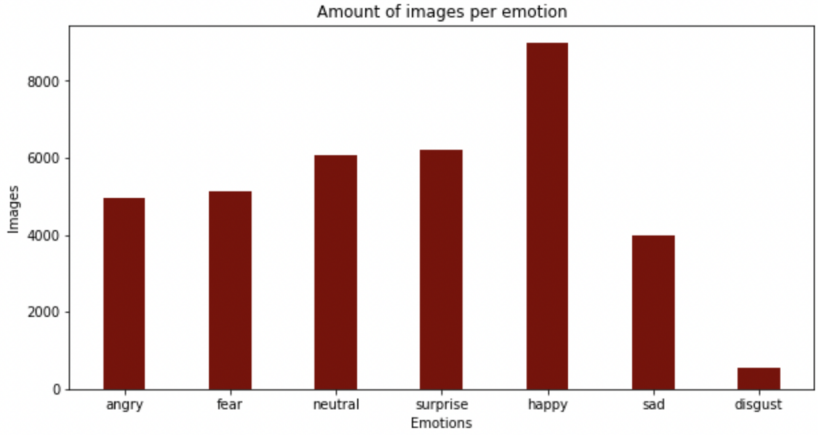
*Supervisors*
Mirela Popa
Bulat Khaertdinov


May, 2022

# 1 Data

The data used for this project is the FER2013 data set [1]. The grayscale images are represented as a 1-dimensional vector containing values in range [0,255]. The first step is to convert the vectors into equal-size (48x48) 2D images. Each image is labelled among different categories, which correspond to emotions (There are 7 in total). A sample of the dataset can be seen below:

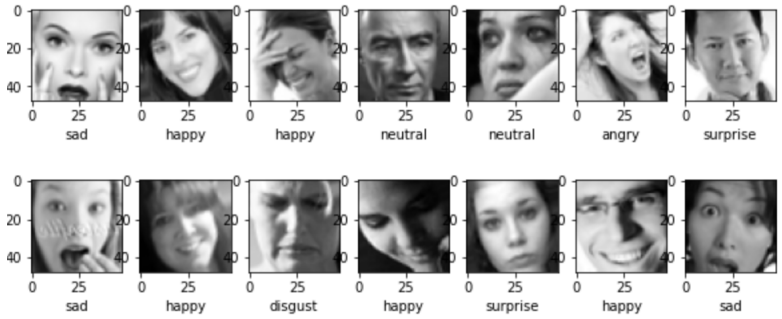

The amount of images for each class is the following:



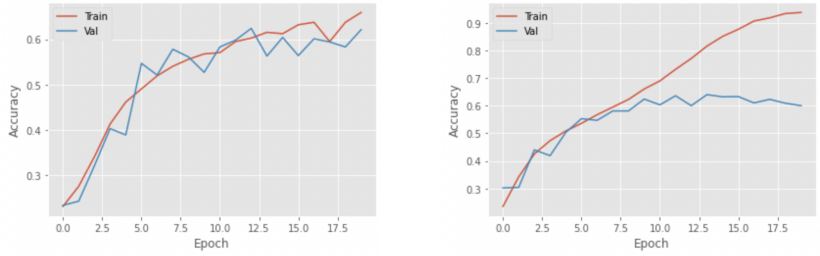

The dataset is quite unbalanced, as the 'disgust' class contains a lot less data than all the other categories. The data is normalized and the class is converted into a one-hot encoding.

## 2 Data Augmentation

Data Augmentation consists of creating synthetic data from the original dataset. The idea is to modify some properties of the original images, and create copies of it (which are slightly different). There are many operations we can perform on the original image, in our case we only perform a rotation by 20°, and horizontal flip. A sample of augmented images can be seen below:




The accuracy of the training/validation set throughout the epochs was retrieved both using/not using data augmentation:



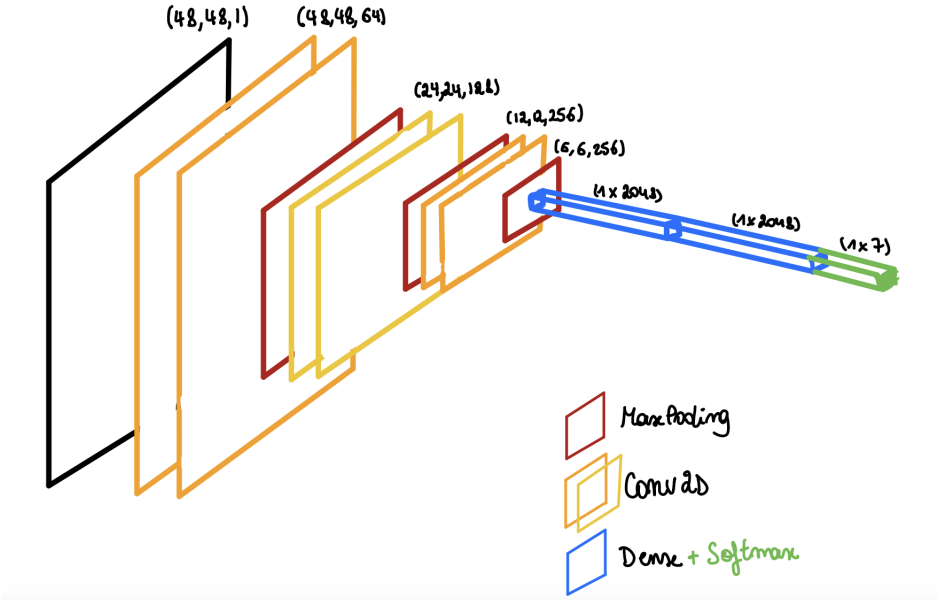

(a) With Data Augmentation (b) Without Data Augmentation

Figure 1: Comparison with/without data augmentation

It seems like data augmentation helps generalizing the data, since the accuracy of the training set is almost the same as the one of the validation set. When no data augmentation is used, it seems like the model is overfitting as the accuracy of the training set achieves really high results. This increased diversity of the training data from data augmentation reduces the variance of the model by making it better at generalizing. Henceforth, it is beneficial to use data augmentation in this project.

# 3 Model

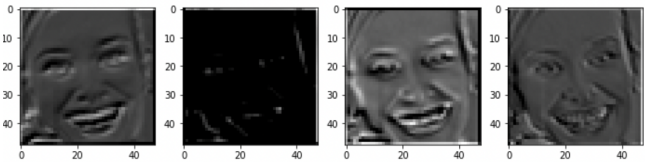
The model implemented for this project is a convolutional neural network. Several architectures (with different amount of Convolutional Layers) were implemented and tested (see the results in Section 5.), and the best model that we have is the following:



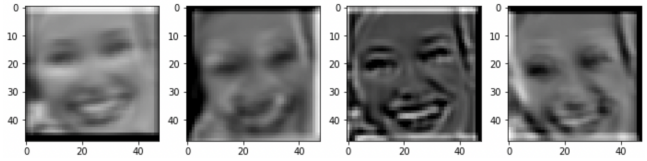

Each image is fed into two sequential convolutional layers which generate 64 feature maps each time. It allows learning some features from the image, but it is essential to downsample in order to learn more local features and reduce the dimension size, which is why we perform add a max pooling layer. Max pooling with size (2x2) will reduce the size of the input by half. We start with a 48x48 image, then after the first Conv2D + MaxPooling our images are size (24x24), then we perform another 2 Conv2D (with ouput channel 128). There is another MaxPooling which downsamples the feature maps to be of size (12x12). Another two ConvLayers, followed by a max pooling are applied. The result is 256 feature maps of size 6x6. This is flattened (made one dimensional), and then fed into a fully connected layer which helps predicting the class with the highest probability. The final layer contains a softmax activation function which is very useful in multi-label classification. In addition, some dropout is added between the Dense layers in order to avoid overfitting the data (see experiments with drop out in Section 3.2).

### 3.1 Kernel Size

The kernel size was set to 3x3 such that we spot smaller features such as teeth. Here we can see the effect of having different kernel size (on the first convolutional filter)



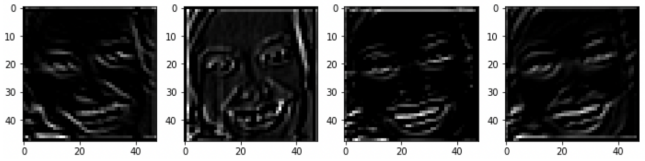
(a) Kernel Size (3x3), First Conv



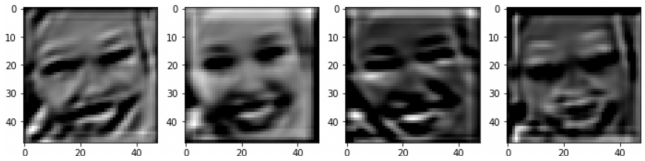
(b) Kernel Size (7x7), First Conv

Figure 2: First convolutional layer map

Once the images are passed into the second convolutional filter, this is how they look like:



(a) Kernel Size (3x3), Second Conv



(b) Kernel Size (7x7), Second Conv

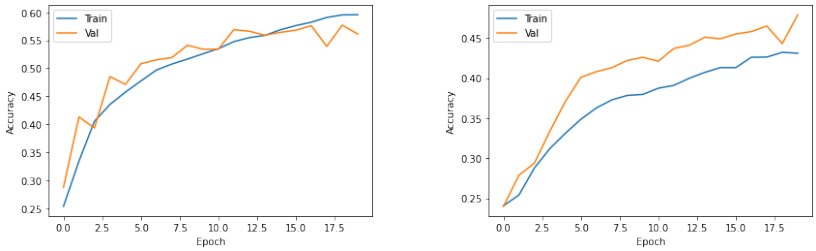
Figure 3: Second convolutional layer map

When the kernel size is set to (3x3) it detects small local features such as edges (we can clearly see the teeth), it is usually better for generalization.

## 3.2 Dropout Rate

The drop out allows to avoid overfitting, however the drop out rate has to be tweaked because it is important to not underfit either. Initially the dropout was set at 0.1, which represents the frequency rate of the inputs set to 0 during training and the results were as shown in the graph a) below, where the model was starting to overfit after epoch 18.

After that we run a trial with dropout value 0.5, meaning that during training 50% the features were set to zero. As shown in the below diagram, the accuracy on both training and validation set were lower comparing to the previous trial with dropout probability of 0.1. Also, an important thing to note here is the accuracy score of the training set being lower than the validation set, meaning that overfitting was reduced by omitting half of the feature detectors. This can also be interpreted as a prevention on the co-adaptation of neurons indicating a lower correlated behavior.[2]




(a) Dropout Layer 0.1

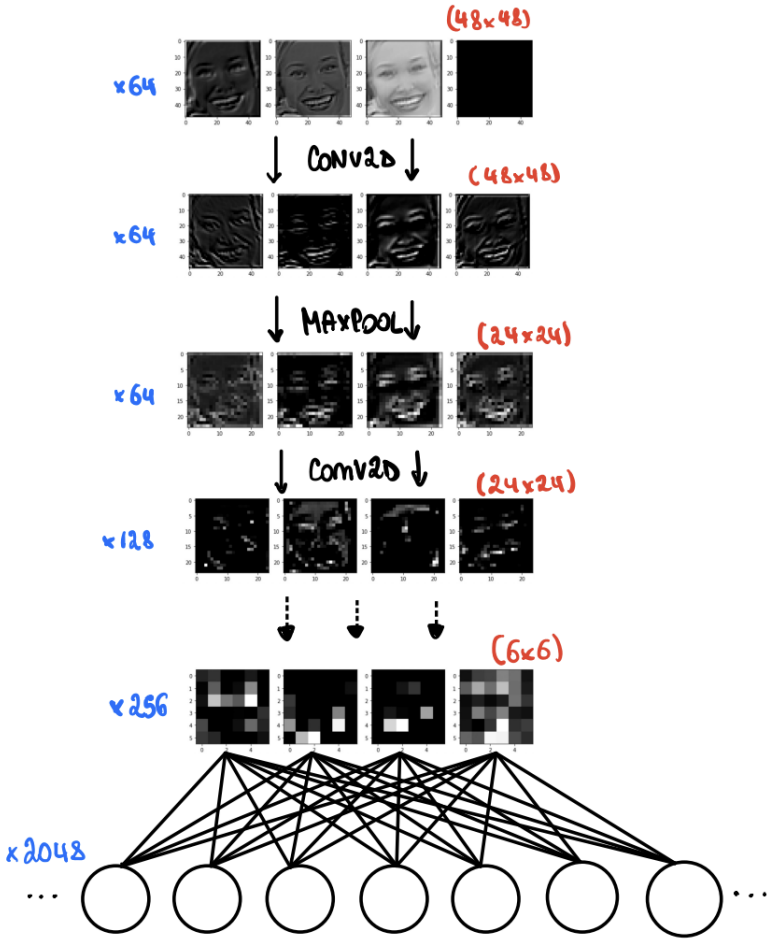
(b) Dropout Layer 0.5

Figure 4: Comparison of dropout layer probability

## 3.3 Convolution Layers

Another thing we experimented with was the number of convolution layers in the model architecture. We trained models with 1, 3, 6 and 8 convolutional layers. Increasing the number of convolutional layers provided better results during the training process. These results showed that depth was really important to achieve high accuracy scores. However, training a model with 8 convolutional layers led to overfitting. This behavior is also pointed out in the original ResNet paper, where the aggressive increase of layers causes degradation of the model performance.[3]. Considering that our dataset is not very big, keeping a model architecture with 6 layers is ideal. This was demonstrated by our results, which are displayed in 1.

# 4 Feature Maps




The first layer of the plot shows the feature maps after the first convolutional layer. After the first MaxPooling layer, the images are downsampled and the resolution is 24x24. Another convolutional layer is applied and generates 128 filters (in contrast to 64). The same process gets repeated until the last feature map (which corresponds to a maxpooling operation that reduces the images to being 6x6). After that, each 256 samples are fed into the Fully Connected Layer.

# 5 Results

Various models were implemented and tested for performance: For simplicity, we will define each model as the number of convolutional layer in their architecture: It seems that it is better to have more convolutional layers for this classification

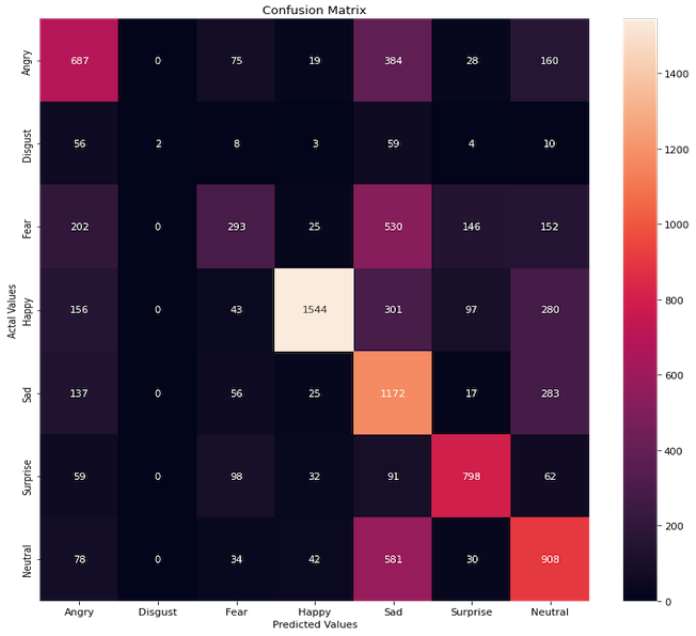
| **Num_Conv** | **1** | **3** | **6** | **8** |
|---|---|---|---|---|
| Accuracy | 45.9% | 55.6% | **63.0**% | 62.2% |
| Sec/Epoch(s) | 9 | 12 | 15 | 24 |

Table 1: Accuracy on test set on different model architectures

problem. The model with 8 convolutional layers seem to be overfitting (it has a high training accuracy but the testing accuracy is low compared to that), hence we keep the model with 6 convolutional layers.

Our evaluations on different number of epochs concluded that training the data for 5-10 epochs did not achieve the best results, however a high number of epochs within a range of 25-30 eventually led to model overfitting.

Below is showed the confusion matrix which visualizes the model's results by comparing actual and predicted results on the test data. Overall the number of True Positives is very high in all classes and the best results are achieved in the "Disgust" and "Happy" categories. However, the model struggles in its predictive power in the "Angry", "Sad" and "Neutral" classes where the number of False Positives (columns) is high.




# 6 Our testings

We also did testings on our benchmark video to evaluate the model's performance. The video contained a sequence of people making different emotion

faces in front of the camera. Using OpenCV CascadeClassifier to detect faces on each frame, the face regions were cropped and reshaped to a 48 x 48 pixel image and then fed in the trained model, from which we retrieved the output with the highest probability. As seen below from the two test cases, the results were very good and model showed great confidence values on its predictions. Besides, we also used the webcam to capture our real time emotions. The basic idea remains the same (cropping and resizing), then we feed it to the model

(a) Image 2
Angry: 0.04 — Disgust: 0.00 — Fear: 0.00 — Happy: 99.96 — Sad: 0.00 — Surprise: 0.00 — Neutral: 0.00

(b) Image 1
Angry: 0.00 — Disgust: 0.00 — Fear: 0.00 — Happy: 99.75 — Sad: 0.00 — Surprise: 0.25 — Neutral: 0.00

Figure 5: Our trials on two face tests

# References


[1] Manas Sambare. Fer-2013, Jul 2020.

[2] Geoffrey E. Hinton, Nitish Srivastava, Alex Krizhevsky, Ilya Sutskever, and Ruslan R. Salakhutdinov. Improving neural networks by preventing co-adaptation of feature detectors, 2012.

[3] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition, 2015.