Computer Vision:
Assignment 2 - Deep Learning for Emotion Recognition

Juliette Maes i6230492, Caroline Lange i6132733

May 30, 2023

1 Introduction

In this report, we present our findings and observations on developing a deep learning model based on Convolutional Neural Networks (CNNs) for the task of emotion recognition. Our objective is to train a model to classify facial expressions into basic emotions such as Anger, Disgust, Fear, Happiness, Sadness, Surprise, and Neutral. We explore various aspects of CNN architectures and parameters, including the number of layers, filters, pooling, normalization, activation functions, and more, to analyze and understand their role and effect on model performance and by that, their impact on the accuracy of emotion recognition. By examining the model's behavior on real-life sample videos, we aim to demonstrate the practicality and efficacy of our approach in recognizing emotions from facial expressions.

This report presents a comprehensive analysis of the CNN-based emotion recognition model, showcasing our experimentation with different architectures and parameters. The subsequent sections provide a detailed account of our methodology, experimental setup, results, and conclusions.

1.1 Convolutional Neural Networks

Convolutional Neural Networks are a class of deep learning models specifically designed for processing and analyzing visual data, making them widely used in computer vision tasks. CNNs excel in extracting meaningful features from images by leveraging the spatial relationships between pixels. The key characteristic of CNNs is their ability to automatically learn hierarchical representations of data. They consist of multiple layers, including convolutional layers, pooling layers, and fully connected layers. Convolutional layers apply filters to the input data, enabling the network to capture local patterns and features. Pooling layers reduce the spatial dimensions of the data, preserving important features while enhancing computational efficiency. Fully connected layers connect the extracted features to the final classification layer.

1.2 Data and Preprocessing

For training and testing our model we use the fer2013 [1] dataset. It is a collection of grayscale facial images used for emotion recognition. The data consists of 48x48 pixel images, which provide a diverse range of facial expressions that correspond to one of the seven emotion labels: Anger, Disgust, Fear, Happiness, Sadness, Surprise, and Neutral. Figure 1 shows a sample of the data and in Figure 2 we visualized the distribution of each emotion present in the dataset. It is clear that there is an imbalance in the data available of the emotions "Disgust" and "Happiness" specifically.

For our application we have normalized the data, however, there are more data augmentation methods that could be applied. Given the sample of the data in Figure 1, it could potentially be beneficial to consider flipping the images in order to introduce different orientations and viewpoints to the data used for training the model.

Figure 1: Sample of fer2013

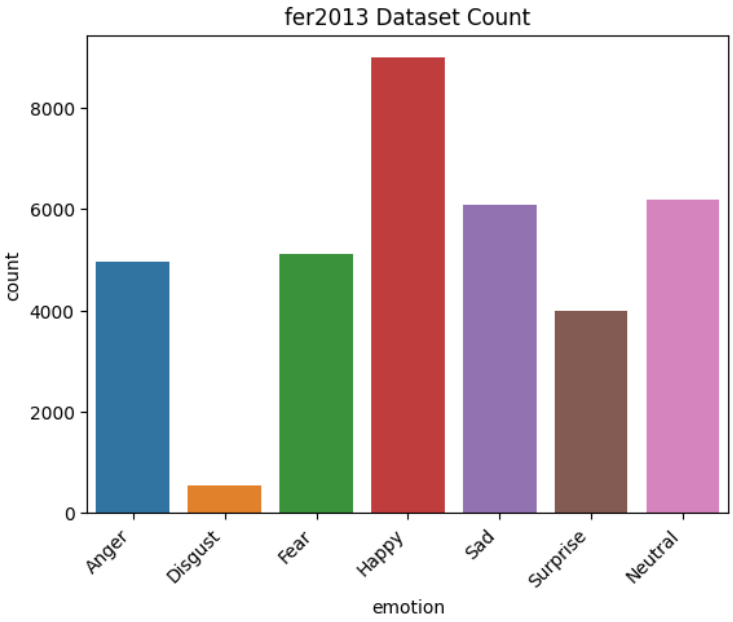


Figure 2: Count of Emotion Labels in fer2013

2 Experiments

To find our best model we run different experiments:

- Variation in training epochs: [50, 100]
- Variation in the number of convolution layers and fully connected layers: [[2,2], [3,2], [5,2], [5, 5]]
- Variation in optimizer learning rate: [0.1, 0.01, 0.001]
- Variation of optimizer: ADAM versus SGD

We chose to not exceed the layer sizes of a maximum of five convolutional layers and five fully connected layers due to the sizing of the images in our dataset. Given that the images are only 48x48 pixels big we argue that it may not provide enough spatial information for a deeper neural network to learn meaningful features. We know that as the network progresses through the convolutional layers, the spatial dimensions of the feature maps are downsampled by pooling operations. Even though this serves to extract the most important features of the input data, in the given case it may lead to a loss of detailed information if the network is too deep. This could ultimately make it difficult for the fully connected layers to effectively utilize the extracted features.

The results of our experiments can be found Tables 1 and 3, which show the outputs of the elapsed time for training and validating, as well as various epoch sizes, learning rates, and convolution and fully connected layer combinations when using the ADAM optimizer and the SGD optimizer, respectively. In the case of **ADAM**, the table shows the same results for many parameter combinations, but especially in all cases where the learning rate is 0.1. The best performances are achieved when the learning rate is 0.001.

Elapsed time	Epochs	LR	Conv2D	FC	Accuracy	Precision	Recall	F1
125.6151996	50	0.1	2	2	0.249373084	0.1	0.5	0.166666667
130.9508693	50	0.1	3	2	0.249373084	0.1	0.5	0.166666667
145.9086666	50	0.1	5	2	0.249373084	0.1	0.5	0.166666667
152.7911675	50	0.1	5	5	0.249373084	0.1	0.5	0.166666667
125.7872434	50	0.01	2	2	0.401783227	0.125	0.25	0.166666667
130.910651	50	0.01	3	2	0.414321538	0.4	0.25	0.28
146.5266106	50	0.01	5	2	0.249373084	0.1	0.5	0.166666667
153.6669083	50	0.01	5	5	0.249373084	0.1	0.5	0.166666667
126.7240162	50	0.001	2	2	0.507105043	0.5	0.5	0.444444444
132.5262671	50	0.001	3	2	0.540540541	0.5	0.375	0.416666667
147.5837038	50	0.001	5	2	0.534410699	0.25	0.25	0.25
156.3378325	50	0.001	5	5	0.533017554	0.4	0.25	0.28
258.0861878	100	0.1	2	2	0.249373084	0.1	0.5	0.166666667
271.2980082	100	0.1	3	2	0.249373084	0.1	0.5	0.166666667
297.206923	100	0.1	5	2	0.249373084	0.1	0.5	0.166666667
313.2612152	100	0.1	5	5	0.249373084	0.1	0.5	0.166666667
252.1364043	100	0.01	2	2	0.249373084	0.1	0.5	0.166666667
261.0029845	100	0.01	3	2	0.417107829	0.5	0.375	0.416666667
290.8231452	100	0.01	5	2	0.249373084	0.1	0.5	0.166666667
304.6246796	100	0.01	5	5	0.249373084	0.1	0.5	0.166666667
255.3409705	100	0.001	2	2	0.528838116	0.375	0.3125	0.266666667
262.5035422	100	0.001	3	2	0.543326832	1	1	1
296.363852	100	0.001	5	2	**0.560044581**	0.666666667	0.583333333	0.619047619
309.0849433	100	0.001	5	5	0.536639733	0.2	0.2	0.2

Table 1: Results training and validation **ADAM**

	precision	**recall**	**f1-score**	support
Anger	0.51	0.44	0.47	491
Disgust	0.54	0.58	0.56	55
Fear	0.44	0.41	0.43	528
Happy	0.72	0.79	0.75	879
Sad	0.43	0.43	0.43	594
Surprise	0.76	0.70	0.73	416
Neutral	0.52	0.55	0.54	626
accuracy			**0.57**	3589
macro avg	0.56	0.56	0.56	3589
weigthed avg	0.57	0.57	0.57	3589

Table 2: Results testing **ADAM**

Elapsed time	Epochs	LR	Conv2D	FC	Accuracy	Precision	Recall	F1
124.5966003	50	0.1	2	2	0.541933686	0.25	0.25	0.25
127.1510928	50	0.1	3	2	0.556422402	0.5	0.3125	0.35
144.2239749	50	0.1	5	2	0.565059905	0.5	0.375	0.416666667
150.0716381	50	0.1	5	5	0.505990527	0.5	0.3125	0.35
110.7491422	50	0.01	2	2	0.512398997	0.25	0.25	0.25
124.7167468	50	0.01	3	2	0.523544163	0.125	0.25	0.166666667
143.8650844	50	0.01	5	2	0.458623572	0.666666667	0.583333333	0.619047619
149.5787146	50	0.01	5	5	0.249373084	0.1	0.5	0.166666667
110.9036608	50	0.001	2	2	0.375034829	0.5	0.5	0.444444444
124.3726978	50	0.001	3	2	0.288659794	0.444444444	0.416666667	0.3
143.9583709	50	0.001	5	2	0.249373084	0.1	0.5	0.166666667
149.2210271	50	0.001	5	5	0.249373084	0.1	0.5	0.166666667
221.3580825	100	0.1	2	2	0.537754249	0.125	0.25	0.166666667
248.7247243	100	0.1	3	2	0.562830872	0.5	0.5	0.444444444
287.2640002	100	0.1	5	2	0.552800223	0.2	0.2	0.2
297.711082	100	0.1	5	5	**0.560601839**	0.666666667	0.583333333	0.619047619
220.9348757	100	0.01	2	2	0.516021176	0.5	0.3125	0.35
248.5310698	100	0.01	3	2	0.524101421	0.5	0.5	0.444444444
287.8113067	100	0.01	5	2	0.465032042	0.5	0.3125	0.35
297.721983	100	0.01	5	5	0.266926721	0.083333333	0.333333333	0.133333333
222.016973	100	0.001	2	2	0.414321538	0.125	0.25	0.166666667
248.3201282	100	0.001	3	2	0.384786849	0.5	0.3125	0.35
287.3551233	100	0.001	5	2	0.249373084	0.1	0.5	0.166666667
297.3626566	100	0.001	5	5	0.249373084	0.1	0.5	0.166666667

Table 3: Results training and validation **SGD**

	precision	**recall**	**f1-score**	support
Anger	0.50	0.45	0.47	491
Disgust	0.87	0.478	0.616	55
Fear	0.43	0.42	0.43	528
Happy	0.79	0.77	0.78	879
Sad	0.41	0.51	0.46	594
Surprise	0.78	0.67	0.72	416
Neutral	0.51	0.54	0.53	626
accuracy			**0.58**	3589
macro avg	0.61	0.55	0.57	3589
weigthed avg	0.59	0.58	0.58	3589

Table 4: Results testing **SGD**

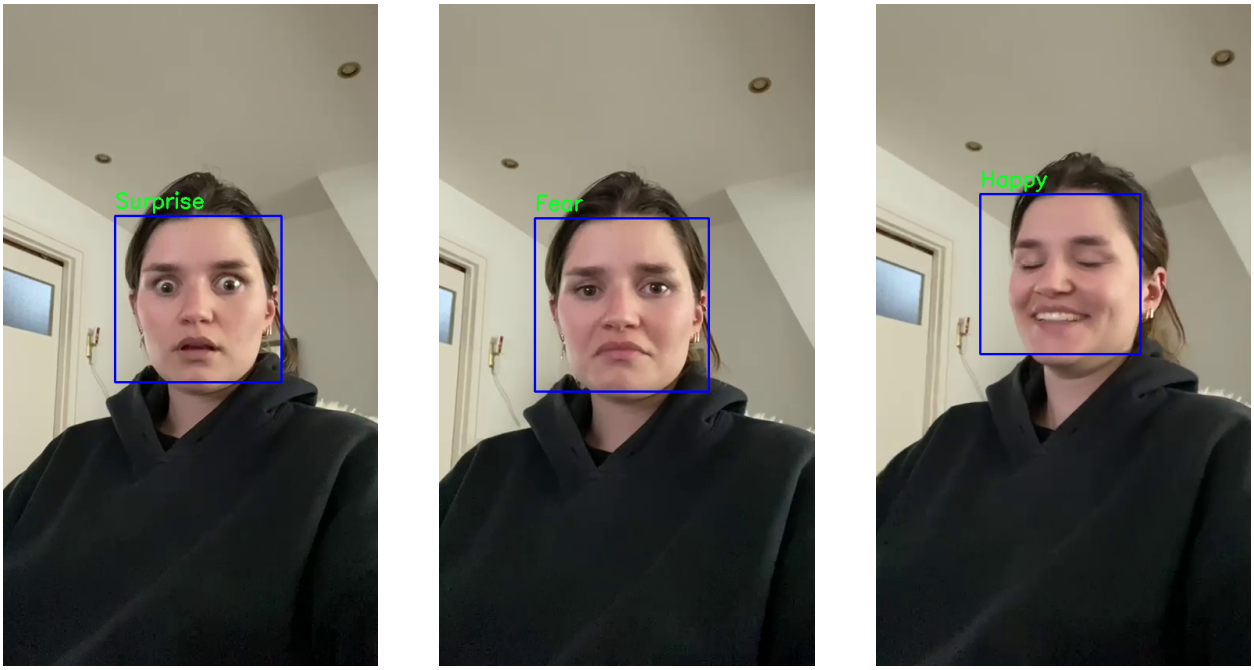
In the case of **SGD**, we also find same results for some parameter combinations, although they are not as many as for ADAM. Comparing the values in the two tables, it is interesting to see that while a learning rate of 0.1 resulted in the "worst" performances and a learning rate of 0.001 resulted in the "best" performances when using the Adam optimizer, the contrary seems to hold when using the SGD optimizer. Intuitively this makes sense: ADAM adjusts learning rates for each parameter based on their gradients. With a higher learning rate such as 0.1, it may update the parameters aggressively, which can cause overshooting and instability. But with a lower learning rate of 0.001, ADAM adjusts the parameters more cautiously, this can results in a more stable and accurate optimization process and thus lead to better performance. On the other hand, SGD uses a fixed learning rate for all parameters. When the learning rate is higher (0.1), SGD takes larger steps during the optimization, which can help escape shallow local optima and thus explore the parameter space more effectively. This can result in faster convergence and improved performance. However, when the learning rate is low (0.001), SGD takes smaller steps and may take longer to converge, ultimately leading to suboptimal performance.

For testing **ADAM**, we chose the model that was trained on: **100** epochs, with learning rate **0.001**, and with **5** convolutional and **2** fully connected layers; this one returned the best validation scores (see Accuracy, Precision, Recall, F1 in Table 1). And for testing **SGD**, we chose the model that was trained on: **100** epochs, with learning rate **0.1**, and with **5** convolutional and **5** fully connected layers; this one returned the best validation scores (see Accuracy, Precision, Recall, F1 in Table 3).

The results for testing the models on unseen data are depicted in Tables 2 and 3. It shows that we achieve an accuracy of 57% for the ADAM model and an accuracy of 58% for the SGD model. Looking at the testing results of both models, we see that they perform similarly on almost all emotions. The only bigger difference is apparent when it comes to "Disgust", where the SGD model seems to perform better at least in terms of precision and f1-score.

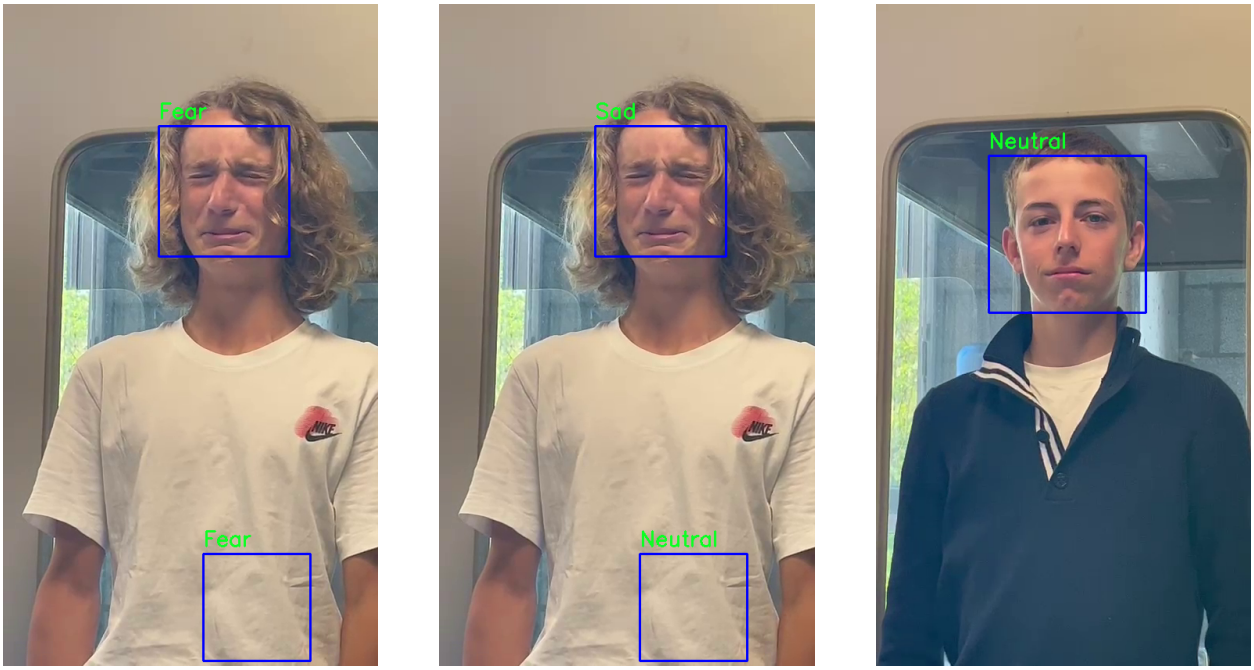
3 Performance on Real-Life Videos

We test our models on a number of real-life videos in order to evaluate its generalization ability in real-world scenarios. The approach was as such: A function was built in order to perform emotion recognition on a real-life video. It was tested on our two pre-trained models: 1) ADAM Optimizer, 100 epochs, 0.001 learning rate, and 5-2 layers and 2) SGD Optimizer, 100 epochs, 0.1 learning rate, and 5-5 layers. A pre-existing algorithm was used for face detection and was performed on each frame of the video to be processed. Detected faces were extracted and resized to 48x48 pixels before being converted to grayscale and stored in an array to match our CNN models. Bounding boxes were drawn around and the recognized emotion label was displayed on the frame. Some results of our implementation can be found in Figures 3, 4, and 5. One can observe that in some videos, the facial emotion recognition went well, whereas in others either the emotion was missed (caused by our models), or more than one instance in the frame was detected as a face (caused by the pre-existing algorithm).



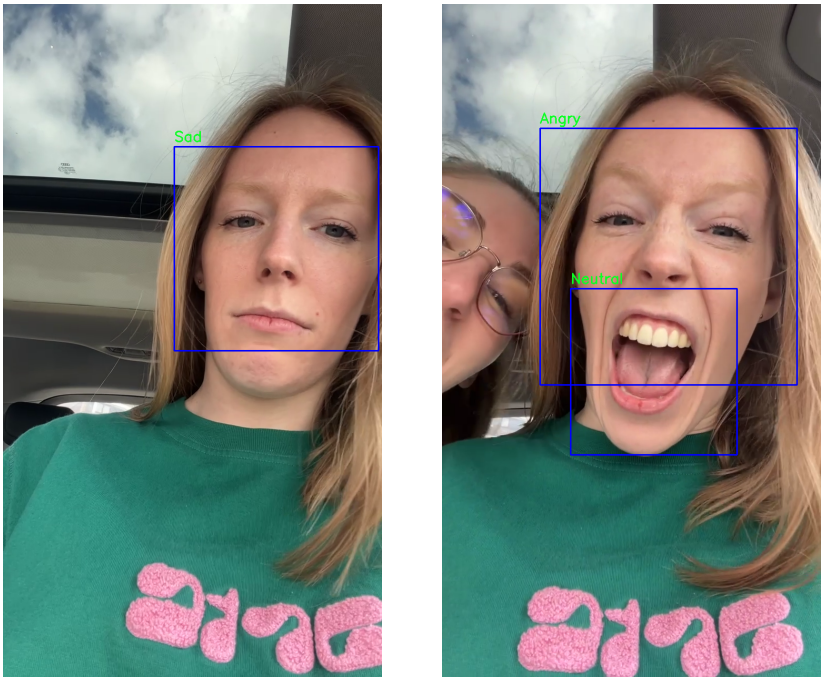
(a) Surprise (b) Fear (c) Happy

Figure 3: Emotion Detection results Video 1



(a) Fear missed face detection (b) Sad & Neutral (c) Neutral

Figure 4: Emotion Detection results Video 2 & 3



(a) Sad (b) Anger & Neutral

Figure 5: Emotion Detection results Video 4

4 Conclusion

In this assignment, we developed a Convolutional Neural Network (CNN) for facial emotion recognition. Our approach involved several steps of preprocessing our data, considering augmentation techniques, experimenting with different model architectures to train our models on, and ultimately evaluating their performances. Conducting various experiments has shown us the effect of playing with different parameter combinations. A special focus for us was on comparing different optimizers (ADAM & SGD) in the training process. We did find that models trained with these optimizers show substantial differences in performance when all other parameters involved are the same, however, we also found that given the appropriate parameters for each, the models show a similar performance of 57% (ADAM,100,0.001,5,2) and 58% (SGD,100,0.1,5,5) accuracy.

References

[1] NowYSM. Facial expression recognition(fer)challenge, Oct 2018.