Emotion Classification Deep Learning Computer Vision Assignment

Wasti Murad - I6348497
& Jules Zeelen- I6350692

May 2024

1 Dataset Description

The data used is the Facial Expression Recognition (FER) dataset from Kaggle which has 35.000 images along with their emotion label. The emotion labels are: Angry, Disgust, Fear, Happy, Sad, Surprise and Neutral. The dataset is unbalanced, as can be seen from the frequency graph of the emotions in Figure 1, which might impact the accuracy of certain emotions.

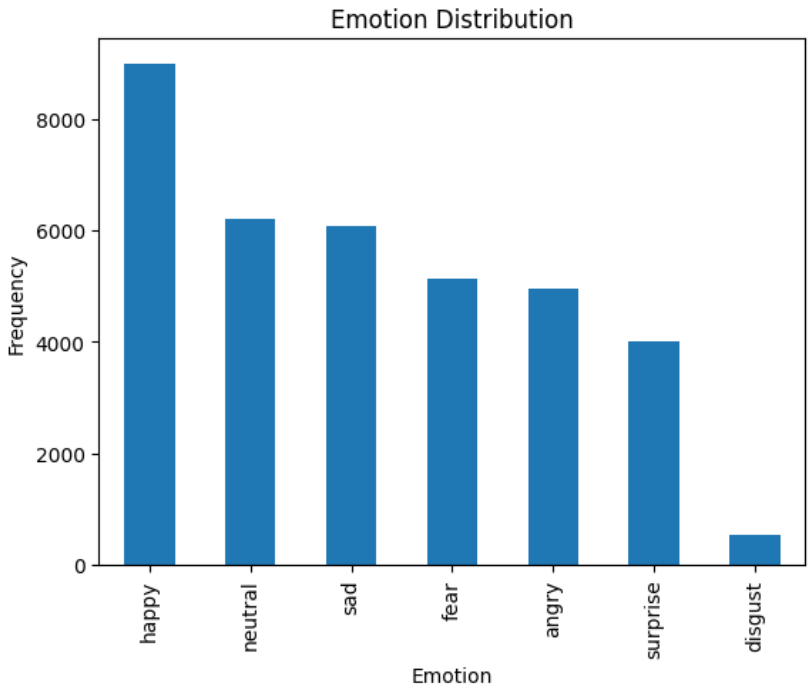

Figure 1: Histogram of the emotions

The data is then divided into training, validation and test. On the training dataset, some transformations (horizontal flip and random changes in brightness, contrast, saturation, and hue) are done to make the cnn model more robust. The three datasets are loaded via Dataloader such that they can be easily accessed when training the model.

2 Pretrained RESNET

We started with a pretrained ResNet-18 model, which is known for its efficient residual learning. This improved our training loss from 1.3555 to 0.3474, with a validation accuracy of 61.30% and a test accuracy of 62.16% over 15 epochs. However, training took over two hours on a CPU, highlighting the need for more efficient custom models or the use of GPU resources for faster and better performance.

3 Initial Model Architecture

After that we tried to create our own CNN model[1] on which we changed also the parameters to increase the model's accuracy.
The CNN model class defines a convolutional neural network with customizable parameters, including activation functions, kernel sizes, filters, pooling kernels, and dropout rates. It constructs convolutional and pooling layers based on these parameters, applies batch normalization if specified, and flattens the output before passing it through fully connected layers to produce the final output. The model can adapt to different input shapes and is designed to perform classification with a specified number of output classes.

3.1 Filters and Layers

We started with two tuples (64,4) and (16,2) where 64, 16 are the initial filter sizes and 4, 2 are the number of layers. Filter value is doubled every layer. We chose to proceed with the (64, 4) model, because it made the most accurate predictions on the validation data after 20 epochs which can be observed on figure 2. We choose feature extractor with 4 Convolutional Layers with 64, 128, 256 and 512 filters respectively.

[1] GitHub repository: `https://github.com/muradohi/Emotion-recognition-based-on-CNNs/tree/main`

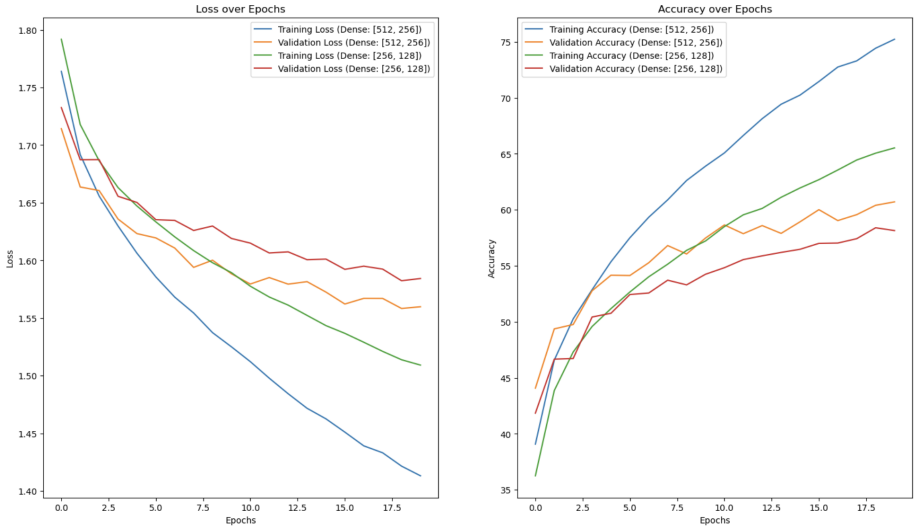

Figure 2: Loss and accuracy over epoch

3.2 Kernel Size

In the second set of experiments, we looked at different kernel sizes for all the convolutional layers. We didn't see any big changes in performance with different sizes. So, we chose to use the smallest kernel size, (3,3), because it is the fastest to compute.

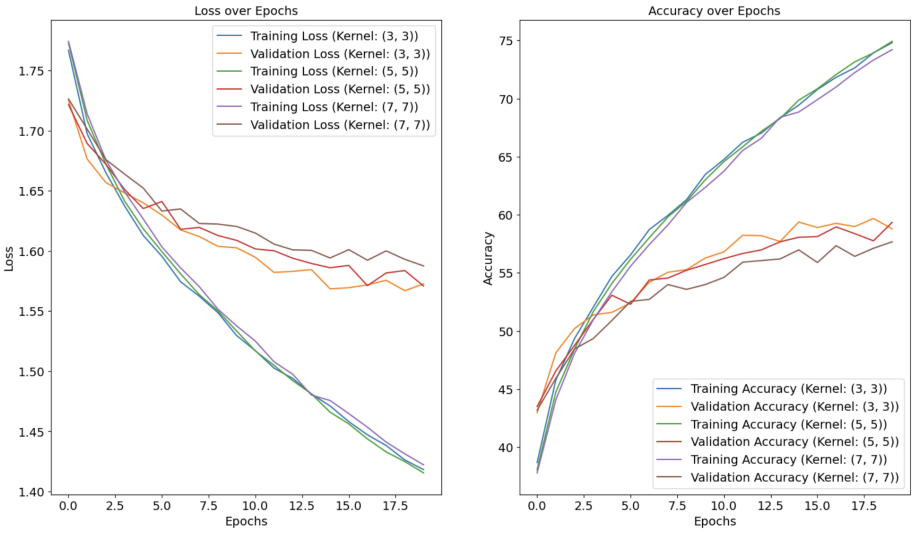

Figure 3: Loss and accuracy over epoch

3.3 Dropout

In order to determine a good strength of dropout regularization for our model, we tested different dropout-rates. The best trade-off between overfitting mitigation and generalizability was achieved with a dropout rate of 0.20. The choice is 20 percent Dropout for all layers (including dense layers).

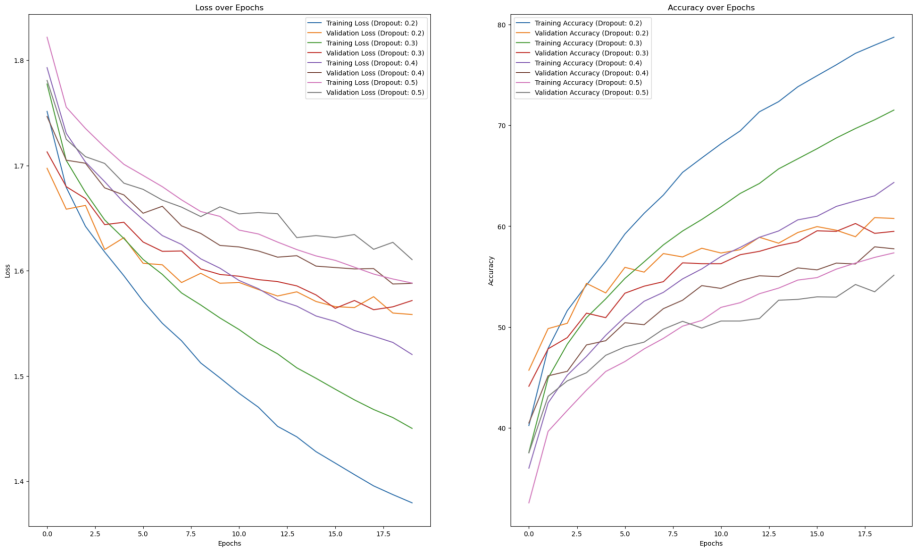

Figure 4: Loss and accuracy over epoch

3.4 Augmentation

We experimented with various augmentation techniques, including random horizontal flips and random rotations, but observed that these methods resulted in significantly lower validation accuracy and took long time to train the model. We decided to exclude augmentation from the final model architecture.

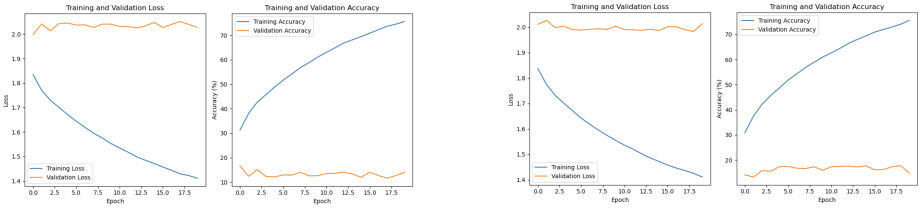

Figure 5: Random Horizontal Flip

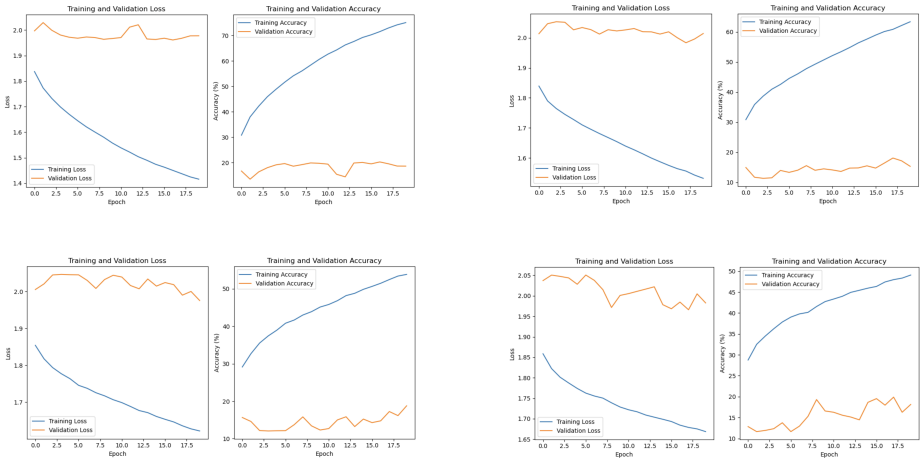

Figure 6: Random Rotations (0,10,20,30)

4 Final Model Architecture

The final model architecture is a convolutional neural network (CNN) consisting of four convolutional layers with increasing filter sizes (64, 128, 256, 512), each followed by batch normalization, and three max-pooling layers applied after the first three convolutional layers. The network also includes a flattening layer that transitions the output from the convolutional layers to two fully connected layers (with 512 and 256 neurons, respectively) before the final output layer that produces 7 class predictions. This architecture does not incorporate any data augmentation techniques.

Figure 7: Confusion Matrix

```
Test Accuracy: 0.6169
Test Precision: 0.6059
Test Recall: 0.6169
Test F1-Score: 0.6085
Classification Report:
              precision    recall  f1-score   support

           0       0.52      0.52      0.52       467
           1       0.00      0.00      0.00        56
           2       0.55      0.40      0.46       496
           3       0.77      0.83      0.80       895
           4       0.52      0.52      0.52       653
           5       0.77      0.76      0.77       415
           6       0.50      0.61      0.55       607

    accuracy                           0.62      3589
   macro avg       0.52      0.52      0.52      3589
weighted avg       0.61      0.62      0.61      3589
```

Figure 8: Classification Report

The final model demonstrates moderate performance on the test set with an overall accuracy of 61.69% and an F1-score of 60.85%. Notably, the model

performs well on *Happy* and *Surprise* emotions with F1-scores of 0.80 and 0.77 respectively, but struggles significantly with *Disgust*, which has an F1-score of 0.00. The confusion matrix highlights that *Angry* is often confused with *Sad* and *Fear*, while *Neutral* shows a significant number of misclassifications across various emotions.

Figure 9: True and predicted emotions

Figure9 highlights both the strengths and weaknesses of our model. We observe accurate predictions for emotions like *Happy* and *Angry*, indicating that the model has learned distinctive features for these emotions. However, we also see significant misclassifications, such as predicting *Happy* for *Disgust* and *Fear* for *Sad*, showing that the model struggles with distinguishing subtler or overlapping facial expressions.

5 K-fold Cross Validation

The k-fold cross-validation was employed to validate the CNN model's performance across different subsets of the training data. The data was divided into three folds, with the model trained on two folds and evaluated on the remaining fold iteratively. Despite the cross-validation process, the test accuracy and recall obtained were slightly lower than the final model's accuracy and recall, likely due to the test set being distinct from the training and validation data used during cross-validation.

```
Test Loss: 1.5898
Test Accuracy: 0.5698
Test Precision: 0.5650
Test Recall: 0.5698
Test F1 Score: 0.5645
Test Classification Report:
              precision    recall  f1-score   support

           0       0.44      0.49      0.46       467
           1       0.00      0.00      0.00        56
           2       0.44      0.36      0.40       496
           3       0.74      0.81      0.77       895
           4       0.45      0.54      0.49       653
           5       0.81      0.65      0.72       415
           6       0.52      0.49      0.50       607

    accuracy                           0.57      3589
   macro avg       0.49      0.48      0.48      3589
weighted avg       0.57      0.57      0.56      3589
```

Figure 10: K-fold classification report

6 Visualized filters and emotion activations

The visualization of filters and activations from our CNN is showcased in the figures below. Figures 11, 12, 13 and 14 depict the filters of the convolutional layers 1, 2, 3 and 4 respectively, along with activations for images of all 7 different emotions. The plotted filters show the learned weights of each convolutional layer, responsible for detecting various features in the input images such as edges, textures, and patterns. The activations highlight how these filters respond to specific features in the input images, indicating which features activate certain neurons, thus providing insight into the model's feature recognition capabilities.

The visualizations demonstrate that the convolutional layers are capable of developing increasingly complex and specialized feature detectors as the network progresses. Through these visualizations, the evolution of learning within the network is observed, showing how different layers capture and respond to different aspects of the input data. These insights are invaluable in understanding the model's behavior and efficacy in feature extraction, which is fundamental to its performance in tasks such as image classification.

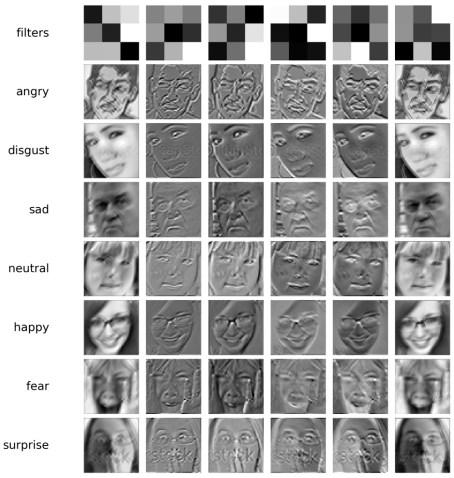

Figure 11: Filters and activations for Layer 1

Figure 12: Filters and activations for Layer 2

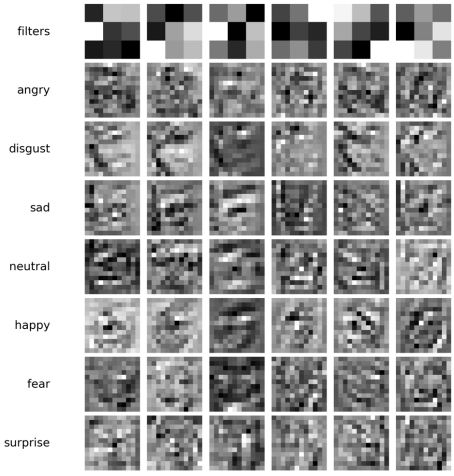

Figure 13: Filters and activations for Layer 3

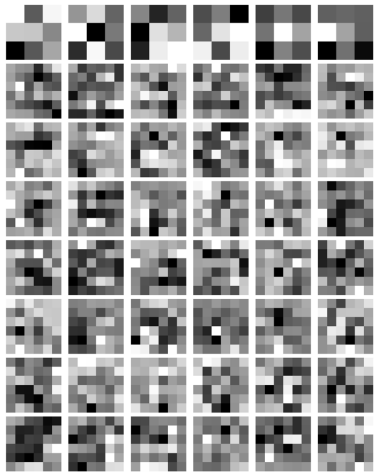

Figure 14: Filters and activations for Layer 4

7 Pretrained VIT model

We also tried a pretrained (training VIT on our own data took too long) Visual Transformer[2], which was trained on a bigger balanced dataset. The pictures on

[2]https://huggingface.co/dima806/facial_emotions_image_detection

the website show a much better confusion matrix and higher accuracy (Figures 15 and 16).

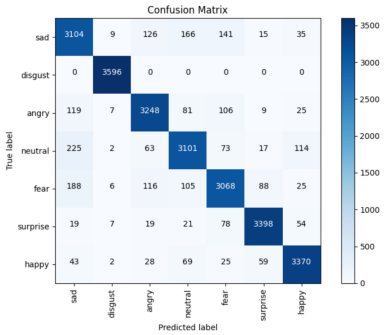

Figure 15: Confusion Matrix

Classification report:

	precision	recall	f1-score	support
sad	0.8394	0.8632	0.8511	3596
disgust	0.9909	1.0000	0.9954	3596
angry	0.9022	0.9035	0.9028	3595
neutral	0.8752	0.8626	0.8689	3595
fear	0.8788	0.8532	0.8658	3596
surprise	0.9476	0.9449	0.9463	3596
happy	0.9302	0.9372	0.9336	3596
accuracy			0.9092	25170
macro avg	0.9092	0.9092	0.9091	25170
weighted avg	0.9092	0.9092	0.9091	25170

Figure 16: Metrics

8 Webcam Test

Our best model and the Pretrained VIT model were tested on images of ourselves and the results can be seen in Figure 17 below. From these detections, it can again be seen that the VIT model is working perfectly while our best CNN model makes some mistakes.

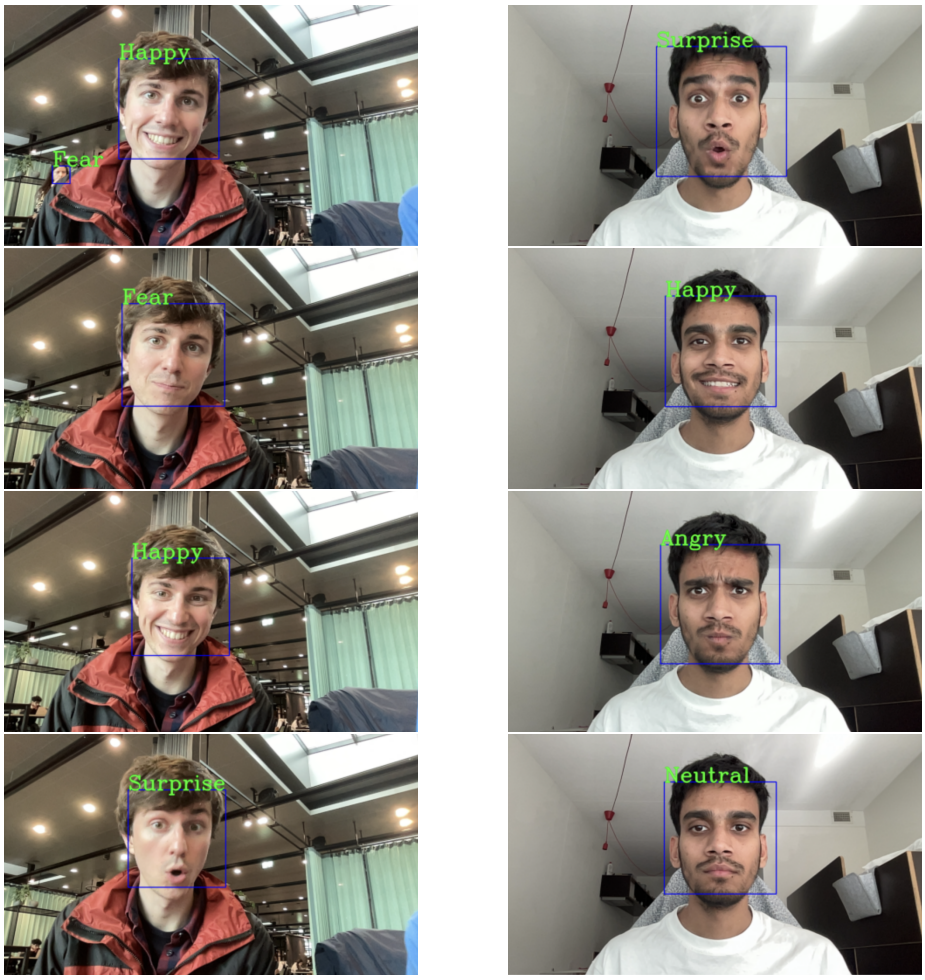

Figure 17: Camera detections; Left = Custom CNN model, right = Pretrained VIT

9 Conclusion

To conclude, We experimented with various models for emotion recognition and found that the transformer-based VIT model performed best. However, training this model requires significant computational resources, which we lacked, that is why we only could use the pretrained one. During hyperparameter optimization for our custom model, it's crucial to vary only one or two variables at a time for meaningful improvements in accuracy and generalizability. Despite these challenges, we improved our custom model's performance, achieving a validation accuracy of 60.7%.