Deep Learning Project - Mood in the room

Heidi Lunde – Group 28

11/12-2022

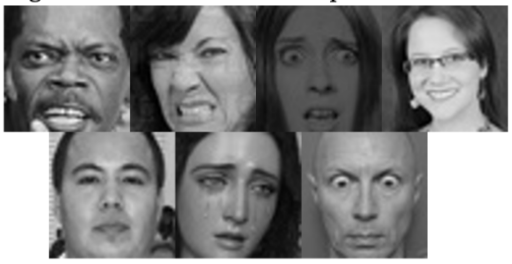
Introduction

The facial expressions of individuals are a major tool in the process of human interaction, as they allow us to recognize the feelings and intentions of those we are communicating with. People are able to figure out the sentiment of others, such as joy, sorrow, or rage, by taking note of their facial expressions and the tone of their voice. Facial expressions are a primary source of communication between individuals.
With that said, detecting overall mood in a group using a convolutional neural network and OpenCV is essential for understanding how individuals interact with each other. By quantifying the emotional responses of a group, we can gain insight into how they interact, how they form relationships, and how they respond to their environment. This data can be used to inform decision-making, improve team dynamics, and better understand the emotional well-being of those involved. Additionally, it can be used to identify areas of improvement, such as reducing stress levels or improving communication. With the help of convolutional neural networks and OpenCV, these insights can be gathered quickly and accurately, allowing for more effective interventions and a better understanding of group dynamics.

Data

We chose the dataset "Face expression recognition dataset" from Kaggle (URL: Face expression dataset) to train our convolutional neural network. The dataset contains 48x48 pixel grayscale images of faces. The dataset consists of two folders, a train folder, and a validation folder. Each folder consisted of 7 subfolders for each mood. The moods that were trained were: angry, disgust, fear, happy, neutral, sad and surprise. Image examples of the moods from the dataset are in figure 1. We gathered the train and validation folders in one folder and divided it into 80% training data, 10% validation data, and 10% test data. The training data consisted of 28710 images, the validation data consisted of 3589 images and the test data consisted of 3588 images.

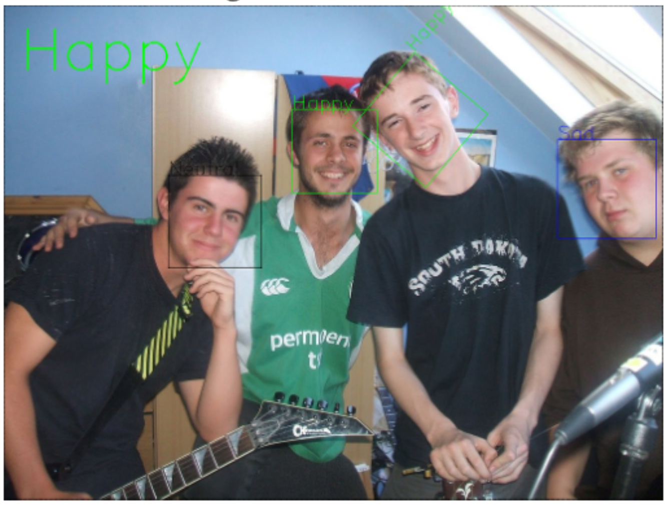
Figure 1: Emotions in face expression dataset



Emotions in reading-order:
Angry, disgust, fear, happy, neutral, sad, surprise

Then we needed a dataset of group images to calculate the overall mood in the image. The dataset is found in the paper "Understanding Groups of Images of People" by A. Gallagher and T. Chen (URL: Group image dataset, Group2a.zip). This dataset consisted of 437 group images. In figure 2 you see an example of the face detection and network used on one of the images in the group image dataset.

Figure 2: Example on facedetection in a group image from the dataset

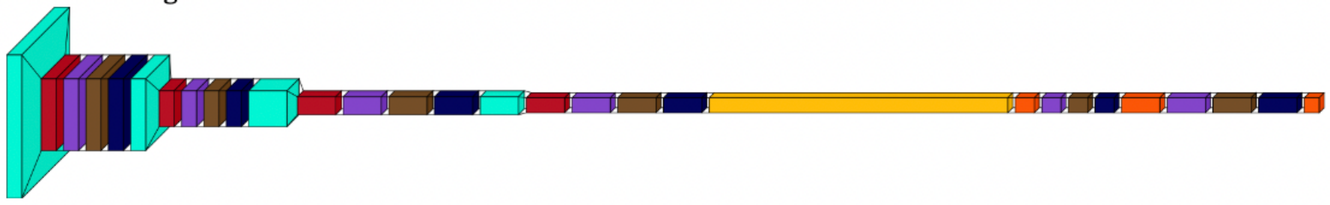


Rotationmatrix used to rotate tilted face

The network structure

Because our network input consists of the faces in a group image, the OpenCV function that is used to detect faces in a group image will be explained before diving into the neural network and lastly, the function that combines them will be explained.

Figure 3: Network structure



🟦 : Conv2d, 🟥 : MaxPooling, 🟪 : BatchNormalization, 🟫 : Relu, 🟦 : Dropout, 🟨 : Flatten, 🟧 : Fully connected

OpenCV facial classifier

OpenCV[1] is a computer vision library that provides a set of tools for image processing, analysis, and machine learning applications. In particular, OpenCV provides a face classifier that is used to detect faces in images or videos.

The OpenCV face classifier is based on a cascade of classifiers that are trained on a large set of positive (faces) and negative (non-faces) images. The classifiers are trained using a feature extraction algorithm, such as Haar features[2], that is able to detect edges, lines, and other features in a face image. The cascade of classifiers is used to filter out false positives and improve accuracy. The classifiers are trained using a supervised learning technique, such as the Adaboost algorithm, to identify patterns in the data and accurately classify faces from non-faces.

The classifier support face detection in various positions, angles, and sizes.

However, we noticed a number of false positive faces and discovered by changing the parameters minNeighbors and scaleFactor, we could make the function more "strict" when classifying faces. Then we discovered several faces that weren't classified as a face. We discovered the pattern of the unclassified faces was when the head was tilting towards another face. To make up for this error we multiplied the rotation matrix on the group image and the error was solved.

The output of this code is the faces in the group image. If the image contained 5 people, the output would be 5 images of their respective faces.

The convolutional neural network

Convolutional neural networks (CNNs) are particularly well suited for this task due to their ability to capture complex visual patterns.

The structure of our network is visualized in figure 3, after the last fully connected layer we add a log softmax.

The first layer of our CNN is a convolution layer, which is responsible for extracting features from the input data. Each neuron in the convolution layer is connected to a subset of the pixels in the input image. The weights associated with each neuron determine how the neurons interact with the pixels in the input data.

The next layer is a pooling layer, which is responsible for downsampling the features extracted by the previous layers. The pooling layer reduces the dimensionality of the data by combining the features extracted by the previous layers into larger, more abstract features.

Then a batch normalization layer is used. It is a layer added to the neural network to normalize the inputs of each layer. The layer normalizes the inputs of each layer by subtracting the mean and dividing by the standard deviation. This helps to reduce the internal covariance shift that can occur during training and helps improve the generalization of the model.

Next is the activation layer, we have chosen Rectified Linear Unit (ReLU) as our activation function. The function is non-linear, which means that it can learn complex patterns and non-linear features from the data. It works by taking the input and setting all negative values to zero, while allowing all positive values to remain the same.

Then, we introduce a dropout layer. A dropout layer is a regularization technique that helps reduce overfitting by randomly "dropping out" a certain number of neurons during each iteration of the training

[1] OpenCV, *OpenCV modules*, accessed 10 December 2022 https://docs.opencv.org/4.x/

[2]OpenCV, *Face Detection using Haar Cascades,* accessed 10 December 2022, https://docs.opencv.org/3.4/d2/d99/tutorial_js_face_detection.html

process. This prevents the network from relying too heavily on a single neuron or feature and encourages the network to learn more broadly. In addition, the dropout process's randomness helps break the weights' symmetry, which further reduces overfitting.

This is repeated in a total of 4 convolutional layers after which the flatten layer is used. A flatten layer in a convolutional neural network is used to convert the input data into a single-column vector. This makes it easier for the fully-connected layers to process the data, as they require a single input vector

With the data converted to a single column vector two fully connected layers with each a batch normalization, Relu, and a dropout layer is inserted into the network.

Lastly, a fully connected layer and a log softmax are inserted. The weights associated with the neurons in this layer determine how the network interprets the input data and makes predictions.

The binding function

When combining OpenCV and the convolutional neural network we define a function that takes a group image as input, identifying the faces with the modified OpenCV facial classifier and converting them to grayscale, because our network is trained on grayscaled images. The faces are given to the trained network that gives each picture an output weight tensor of length 7. The weights represent which mood they are in. Each face gets classified with the highest weight. Lastly, the overall mood is calculated by summarizing the weights of each mood and finding the mood with the highest value. The output of the function is the group image with mood classifications and overall mood.

Training and hyperparameters

When we started the training process of the network, we chose the Adam optimizer because of our previous experience in this course. As loss function, we used cross-entropy, because it is well-suited to the task of classification. It measures the difference between the predicted class probabilities and the true class probabilities for each sample. The loss value is then directly related to the differences between the predicted and true class probabilities, it is a good indicator of how well the model is performing.

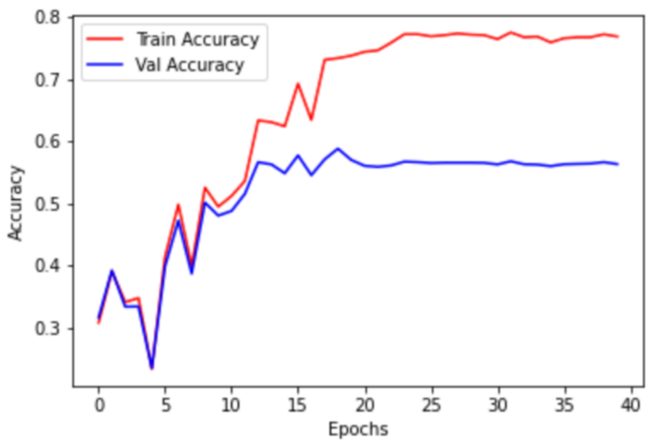
When we had chosen our optimizer and loss function, we trained the model without the dropout regularization layers. This gave a very high training accuracy but a low validation accuracy resulting in an overfitted model. Then we inserted the dropout layer with a probability of 0.25, this resulted in the validation loss converging faster. However, we got a higher validation accuracy when using a probability of 0.1 and also the training accuracy decreased while the validation data increased. This is a sign that our model fitted our data better with 0.1, but the model still stagnated and was too inaccurate. We wanted the accuracy to increase even more and decided to add a learning rate on plateau scheduler while training the model. This adjusted the learning rate when validation loss worsened twice an epoch in a row and allowed us to increase our initial learning rate. Then we added weight decay to the optimizer to help avoid overfitting.

When adjusting the batch size and the number of epochs we discovered our model performed better when we decreased the batch size. We tried to train the network with 100 epochs, but the accuracy seemed to converge between 15-25 epochs.
We ended with a learning rate of 0.001%, a batch size of 128, and the number of epochs was 40.

Network performance

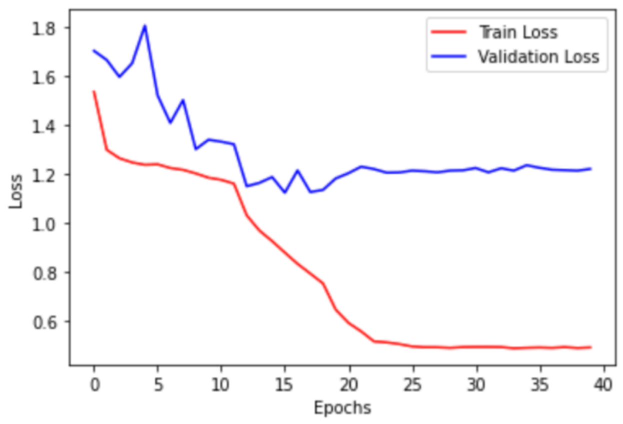
With the network explained in the previous sections, the test accuracy ended at 60.5% and the plot is shown in figure 4.

Figure 4: Accuracy plot



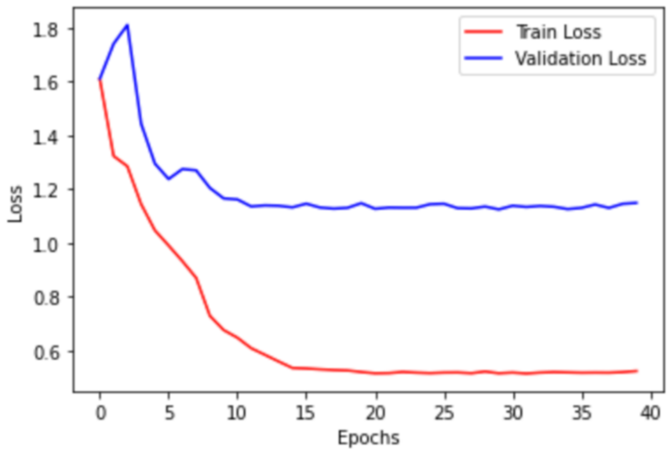
The validation loss function for this network converged around epoch 20 as seen in figure 5. It is a little late because as we know from this course when a loss curve drops quickly it often ends low.

Figure 5: Loss plot



However, if we adjust on some of the parameters in the learning rate on plateau scheduler, we get a validation loss curve that converges around epoch 11 as seen in figure 6. Unfortunately, this network only performs with a test accuracy of 58%, which is why we chose to proceed with the previous network.

Figure 6: Loss plot for network with changed parameters



When deciding if this is a good model to predict the mood in a group, we needed to check the level of correctness on some pictures from the group image dataset, and by qualitatively judging the performance, we think the model is surprisingly accurate even though the test accuracy could be better. When reading other papers within this area it looks like the validation accuracy can't perform better than around 65%[3] with this data.

Conclusion

Our main task was to train a network to help decide the mood in a group. When qualitatively evaluating the group images, we do accomplish the task to find the mood in the room as seen in figure 2. However, our validation loss is higher than we wanted it to be and that resulted in a test accuracy of 60.5%. When reading other papers within this area we found that we should not expect a validation accuracy of over 65% with this data.

In this project, we learned that a network would easily stagnate when using too many regularization parameters, but it will be overfitted when using too few. We were a bit surprised at how little it took our model to either stagnate or overfit.

We did also learn that the OpenCV facial classifier had some weaknesses as tilted heads and that it was important to choose the right values for the parameters to avoid false positive faces.

Some of the things that strengthened our model are the following. Our model is quite good at classifying people with accessories even though our training data didn't include accessories like sunglasses, glasses etc. We also fixed the undetected tilted faces problem, and we convert the input to grayscale to match the training data. That means accessories, colored pictures, and tilted faces will not be a problem for our model since our model is already aware of these obstacles.

Some weakness our model is facing is that the faces in the training data are very exaggerated and can possibly result in wrong classifications of the moods and it might the cause of the low validation accuracy. The training data was also only in a 48x48 pixel format, which makes it more difficult for our network to find patterns of lines in the face, as facial lines are a crucial factor to determine a mood. Lastly, it can seem misleading if an output picture has the overall mood as "neutral" and there are 2 out of 3 people that are "happy", this is because we calculate the overall mood with the weights and not the count of classified moods.

Code

The code is attached as a zip-file to the appendix with the report.

[3] Kaggle, *Dataset Notebooks*, accessed 10 December 2022, Dataset Notebooks