# Facial Expression Recognition using Convolutional Neural Networks

**Author: Erik Greblo**

**Mentor: assoc. prof. Marina Bagić Babac**

**Abstract**

**Purpose –** This seminar shows the process of designing a convolutional neural network model capable of recognizing facial expressions. It compares models with different architectures and complexities and their accuracy on a large dataset, following a research paper.
**Design/Methodology/Approach –** A CNN-based approach is used to classify grayscale images of faces into seven expression categories. Multiple models with varying depths, kernel sizes, and regularization techniques such as dropout and batch normalization are trained and evaluated on the FER2019 dataset.
**Findings –** Deeper convolutional networks generally achieved higher accuracy, especially when combined with regularization. The best-performing model reached over 61% accuracy on the test set.
**Originality/Value –** This paper offers a comparative analysis of CNN architectures for facial expression recognition, providing insights into how different architectural choices affect performance.
**Keywords –** Machine Learning; Deep Learning; Convolutional Neural Networks; FER.
**Paper Type –** Seminar.

## 1 Introduction

When two people communicate with each other, their interaction can be interpreted both from the words they speak, and the expressions of their bodies. For them, it is intuitive to combine both ways of expression to construct the final interpretation of the encounter. But what if one of the dimensions of expression is lost? This is present in SMS messaging, where individuals communicate only trough words, sentences, and paragraphs, without any punctuation and expressions. Emoticons can add some depth into the conversation, but SMS are still inferior to a face-to-face conversation, where people can observe body and facial expressions, constructing a better understanding of the situation.

Humans have evolved to intuitively understand facial expressions and classify them into distinct categories, which cannot be said for neural network models. Simple fully connected neural networks are great for tasks such as classification of handwritten digits, but the task of recognising facial expressions is much more complex for such a simple model to score a good accuracy.

Model that can reliably classify a facial expression from a picture of a person's face can be extremely useful in science and medicine. This seminar displays how such a model could be constructed, how it compares against simpler models and how it could be upgraded in the future.

## 2 Dataset

Every model needs to have a dataset on which it is trained. This paper uses *Face expression recognition dataset* (FER) from Kaggle [1]. It consists of train and validation folders, each with separate folders with images of different facial expressions. Dataset contains images classified into 7 different expressions: angry, disgust, fear, happy, neutral, sad and surprise. There are of course a lot more expressions that a face can make, but this is a great start to evaluate the computers capabilities into facial expression recognition.

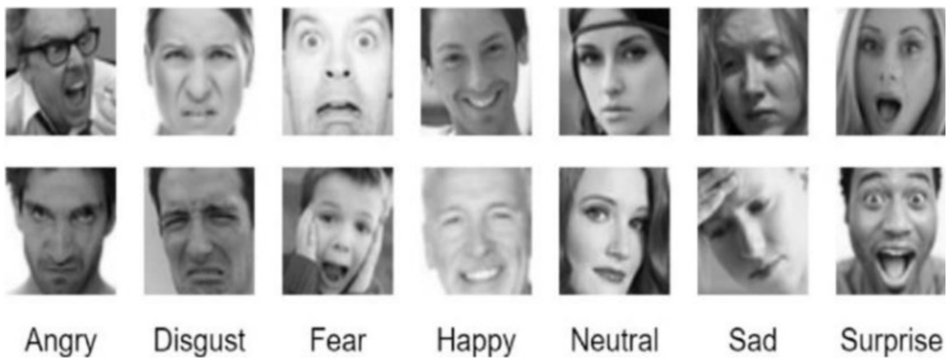


Figure 1 Images for each expression in the dataset

Models in this paper are trained on a dataset with about 29000 different grayscale images, with dimensions of 48x48. Every image consists of a person's face expressing one of the 7 expressions listed above. To test the model's accuracy on images that it didn't already see and learn from, there are two additional datasets, validation dataset and test dataset, each with 3500 images. It is necessary to point out that the dataset isn't balanced, which means that there is not the same number of images for every expression, as seen in the figure below. This is important to consider when evaluating the model. High accuracy doesn't necessarily mean that the model correctly identifies every expression. If there is a small number of pictures labelled as a certain facial expression, even if the model never correctly labels them, the accuracy wouldn't change as much, just because of the small quantity of wrongly predicted images. This could be resolved with balancing the dataset, either gathering more images from the categories that have the fewest amount or, if that is not possible, balance the dataset by limiting the number of images for each expression to the number of images for the expression that has it the lowest. In this case this wouldn't be optimal, because the expression with the fewest images has only 547 images.

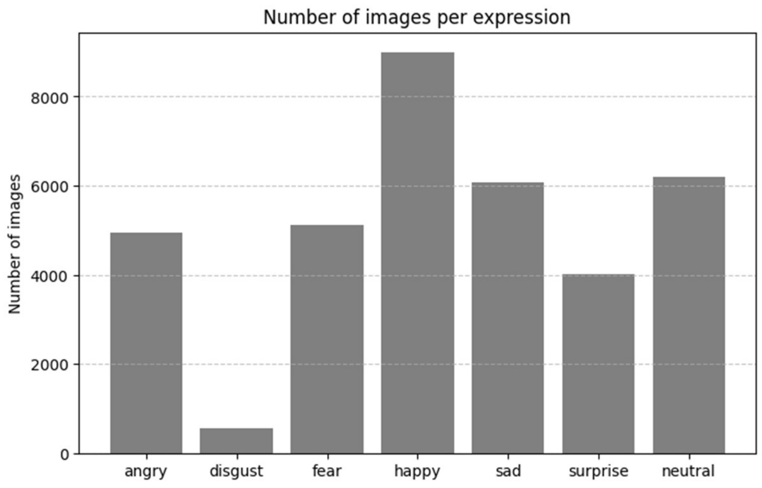


Figure 2 Quantity of each expression in the dataset

## 3 Model architecture

The simplest model that can be trained on this dataset is a simple fully connected neural network model. A model with only one layer gets every single pixel to one of its inputs, calculates the outputs by multiplying inputs and weights, and the biggest value in the output gets chosen as the model's prediction to the inputted image. This method is viable for simpler tasks, but facial expression recognition is not one of them.

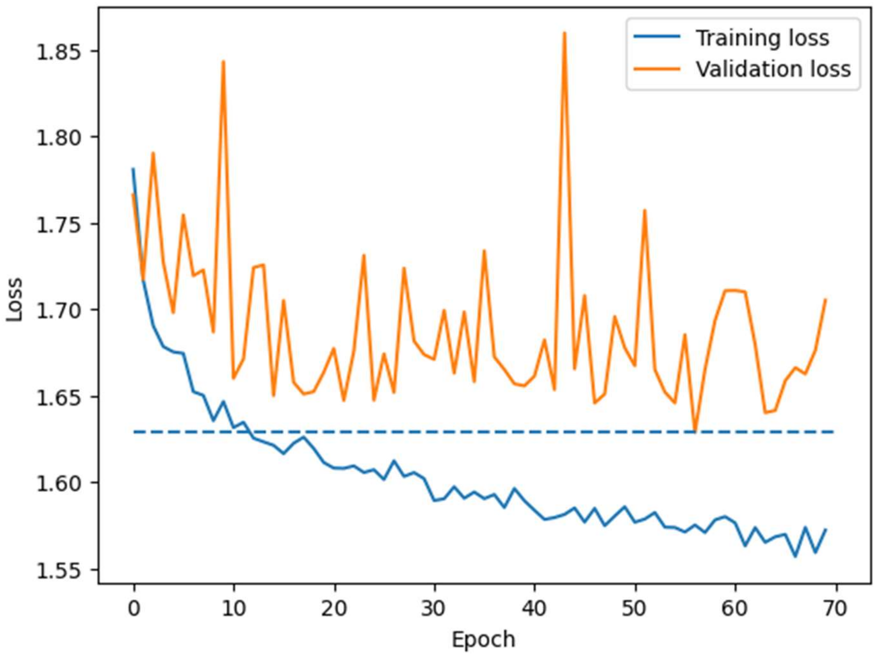


Figure 3 Training and validation loss for the fully connected model

In 30 epochs, the fully connected model with one fully connected layer achieved the accuracy of 35.35% on the test dataset. From the figure above it is obvious that the simplicity of the model is not up to the task for this task. To properly recognize facial expressions, model needs to learn shapes, patterns, and features in the images for every expression in the dataset. Fully connected models don’t learn anything about the relationship of a pixel with its neighbours; hence they cannot detect patterns. Convolutional neural networks, however, are perfect for the task.

## 3.1 Convolutional model

A convolutional model is a model that contains at least one convolutional layer. Convolutional layer consists of filter, whose weights are the parameters of the model which get calibrated during training to output optimal predictions. This filter acts as a mask over the image, sliding across every pixel and multiplying its weights with a small area of the input, over which the filter is currently placed. With this method, a single output not only depends on a single input pixel and models weights, but also on the neighbouring pixels and their weights. This allows the model to detect edges, textures and much more complex patterns which make it a great tool for this task.

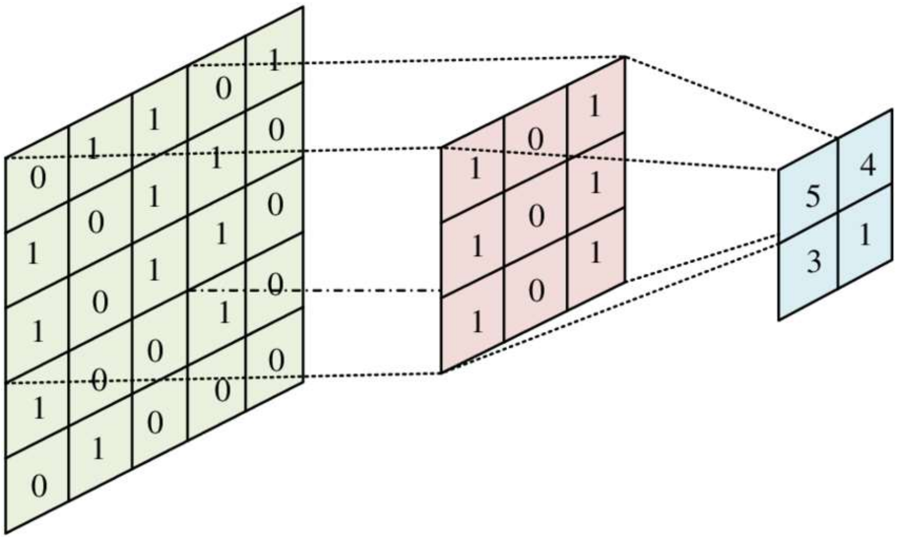
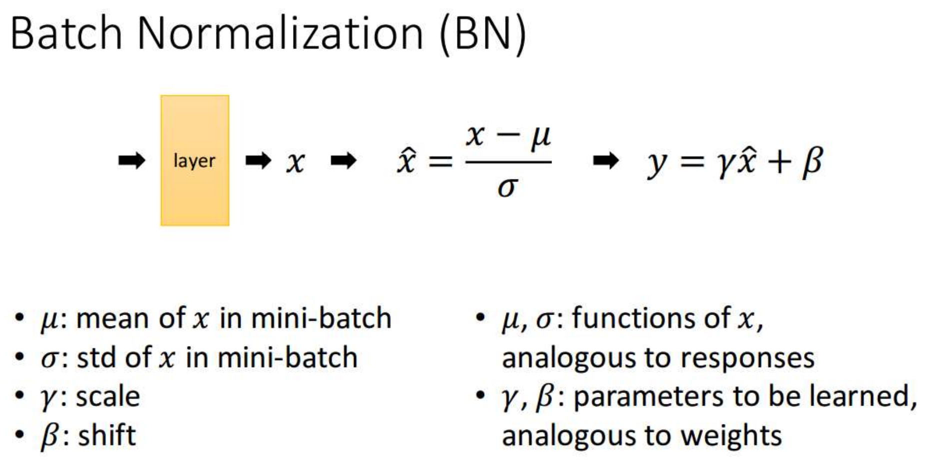


Figure 4 Visualization of the convolutional layer [2]

## 3.2 Batch normalization

Each image consists of 48x48 pixels, each with a grayscale value ranging from 0 to 255. It is common practice to normalize the input to the model in a range from 0 to 1. If this step was not preformed, the higher values of the pixels could have more influence on the resulting prediction than the lower values. When normalizing the entire picture, the maximum value is 1, which represents a fully white pixel with the original value at 255.

Following this methodology, the same method can be applied to hidden layers of the model. Output values from the layers don't have the restriction of being in between 0 and 1. This is where the normalization can be applied to batches. Batch normalization is a layer that is put between other hidden layers. It has 2 parameters, beta and gamma, which are used for scaling and shifting of the values. This process stabilizes learning, allows higher learning rates and acts as a regularizer.



Batch Normalization (BN)

$$\rightarrow \boxed{\text{layer}} \rightarrow x \rightarrow \hat{x} = \frac{x-\mu}{\sigma} \rightarrow y = \gamma\hat{x} + \beta$$

- $\mu$: mean of $x$ in mini-batch
- $\sigma$: std of $x$ in mini-batch
- $\gamma$: scale
- $\beta$: shift
- $\mu, \sigma$: functions of $x$, analogous to responses
- $\gamma, \beta$: parameters to be learned, analogous to weights

Figure 5 Batchnorm process [3]

### 3.3 Dropout

To further improve model’s ability to achieve better accuracy and reduce the chance of overfitting, a dropout layer is placed after the batch normalization layer. Dropout layer randomly turns-off a percentage of its neurons, which means that every forward pass through the model results in a slightly different output even when the input is the same, as shown in Figure 6. This makes the model learn more robust features of the images and further reduces overfitting.

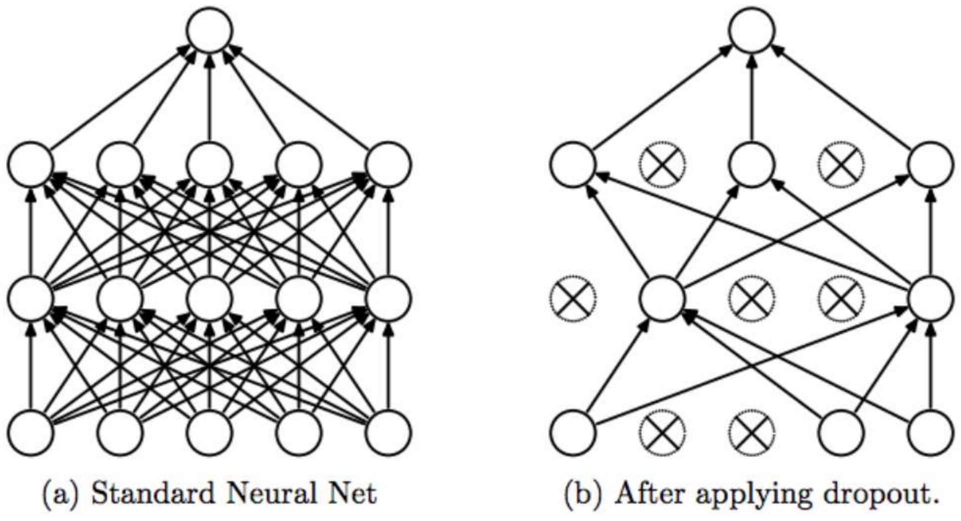


Figure 6 Dropout visualization [4]

### 3.4 Max-pooling

Max pooling is a layer that performs down sampling, an operation that reduces the dimensionality of the output. This is achieved with a filter, like the convolution layer, but max pool filter doesn't have any parameters that need to be trained. It has a hyperparameter which determines the size of the filter. Moving the filter across the input, the output at one location is the maximum value in the image input that is under the filter.

This method increases performance, reduces noise and condenses the useful information. Furthermore, max pooling increases the invariance to translation, larger the filter, larger the invariance.

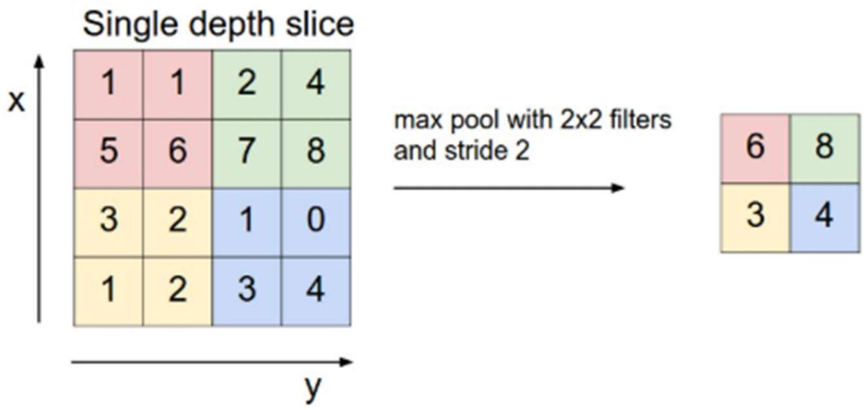


Figure 7 Max pooling visualization [5]

## 4 Experiments

Using the beforementioned layers together in a deep model can produce a far better accuracy than any simple fully connected model could. To test this hypothesis, researchers [6] have recommended the following architecture:

[Conv-(SBN)-ReLU-(Dropout)-(Max-pool)]M -

[Affine-(BN)-ReLU-(Dropout)] N - Affine – Softmax

First model that was learned on the given dataset will be called the *shallow model*. It consists of 2 convolutional layers and a single fully connected layer. First convolutional layer consisting of 32 3x3 filters, and the second of 64 5x5 filters. Dimensions of the single fully connected layer is 512 neurons. With cross-validation, it is concluded that the best hyperparameters for learning this model (learning rate and weight decay) are the following:

| Parameter | Value |
|---|---|
| Learning rate | 1e-5 |
| Weight decay | 1e-7 |

Table 1 Hyperparameters for the shallow model achieved by cross validation

Training the model over 120 epochs using beforementioned hyperparameters, the shallow model achieved the accuracy of 53.21% on the testing dataset. This is a significant improvement over the fully connected model, but it still leaves the question of how many more convolutional and fully connected layers should a model have to achieve a better performance. To answer such questions, another model is created and trained, called the *deep model*.

Deep model consists of 4 convolutional layers. First layer has 64 3x3 filters, second has 128 5x5 filters, and last two layers have 512 3x3 filters each. Model also consists of 2 fully connected layers, first with 256 neurons and the second with 512. Cross validation stated that the best hyperparameters for the deep model are as follows:

| **Parameter** | **Value** |
|---|---|
| Learning rate | 1e-4 |
| Weight decay | 1e-7 |

Table 2 Hyperparameters for the deep model achieved by cross validation

After 100 epochs of training with the optimal hyperparameters, the deep model achieved the accuracy of 59.07%, almost a +6% improvement from the shallow model. To investigate if this is the limit of this architecture, another model can be constructed with even more layers and even more parameters to train and to possibly lower the loss and increase the overall accuracy over the testing dataset. For this reason, the *deeper model* is constructed, which has an additional convolutional layer than the deeper model, with 256 3x3 filters. Optimal hyperparameters are the same as in the table above.

After training the deeper model through 100 epochs, it achieved the accuracy of 61.76% on the test dataset, an improvement of +2.7%. Further addition of convolutional and fully connected layers doesn’t improve the model's performance drastically.

## 5 Results

After training several models with different levels of complexities and types of architectures, it is determined that the task for facial expression recognition is perfectly suited for models that utilize convolutional networks. It is observed that the optimal number of convolutional layers is 5, after which the additional layers don’t considerably improve the accuracy. To better visualize the comparison of the loss function during training and validation, Figure 7 shows how each model compares to the others.

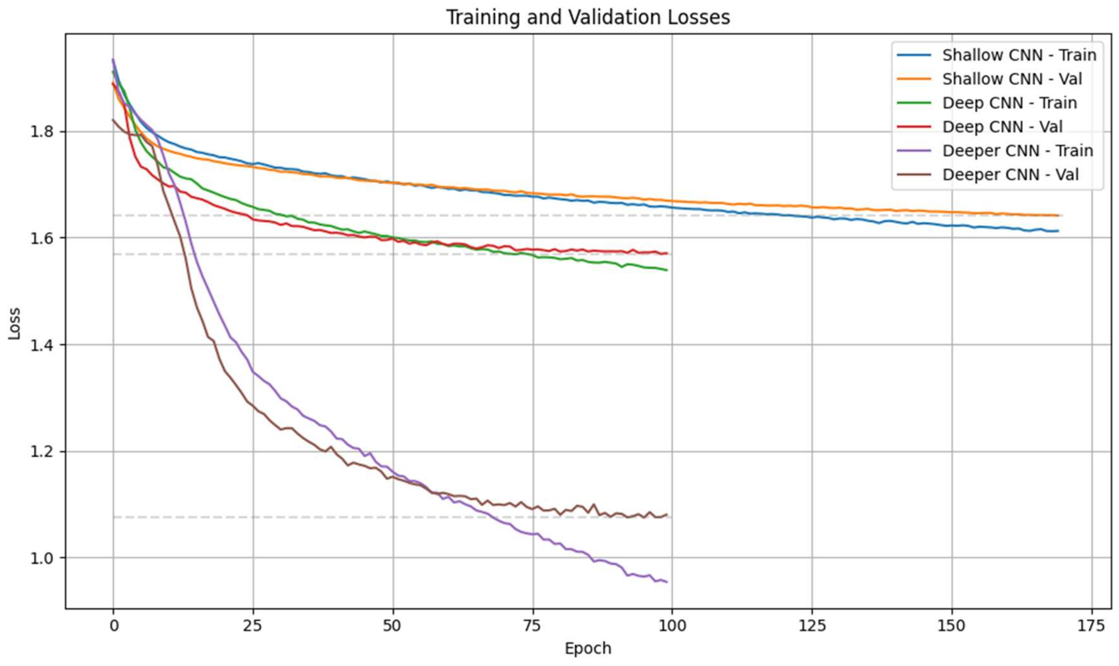


Figure 8 Training and validation losses for different models through epochs

Because of a small learning rate of the shallow model, it needed to be trained for longer for the loss curve to flatten, deeper models had a higher learning rate, so the loss reached the minimum faster. To understand the accuracy of each expression for the models, the table below shows how accurate the models are when observing each expression separately.

| Expression | Shallow CNN | Deep CNN | Deeper CNN |
|---|---|---|---|
| **Angry** | 32.0% | 45.9% | 49.7% |
| **Disgust** | 0.0% | 0.0% | 39.2% |
| **Fear** | 19.4% | 24.2% | 23.8% |
| **Happy** | 79.9% | 84.2% | 86.8% |
| **Sad** | 46.4% | 49.6% | 51.0% |
| **Surprise** | 64.4% | 73.3% | 81.4% |
| **Neutral** | 62.2% | 65.8% | 65.1% |

Table 3 The accuracy for each expression in the shallow, deep and deeper models

Original dataset is not balanced. There are far more ‘happy’ expressions that there are ‘disgust’ expressions. This is not ideal in training the model because it will achieve a high accuracy even if it never correctly identifies some expressions because the dataset doesn’t contain much of them. This can be easily visualized with a confusion matrix.

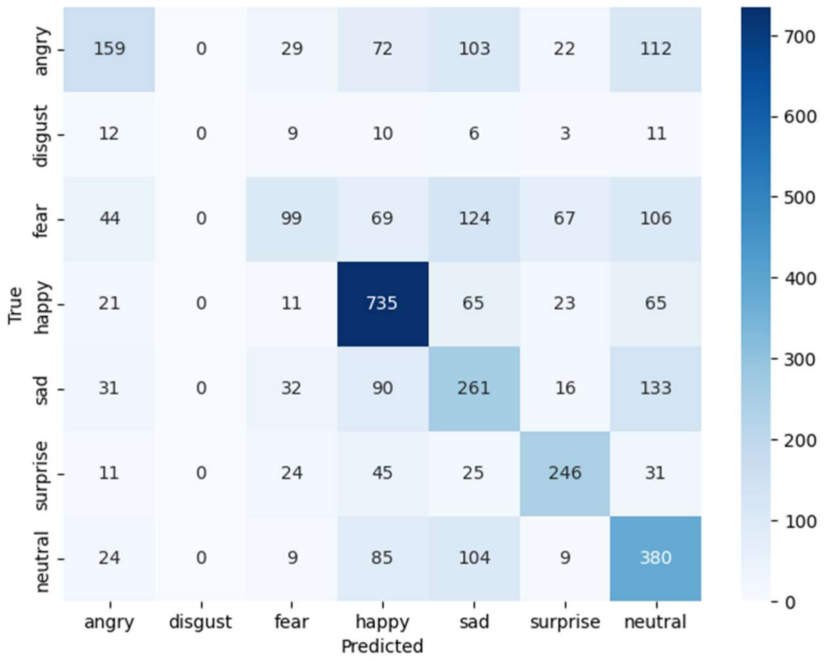


Figure 9 The confusion matrix for the deep model

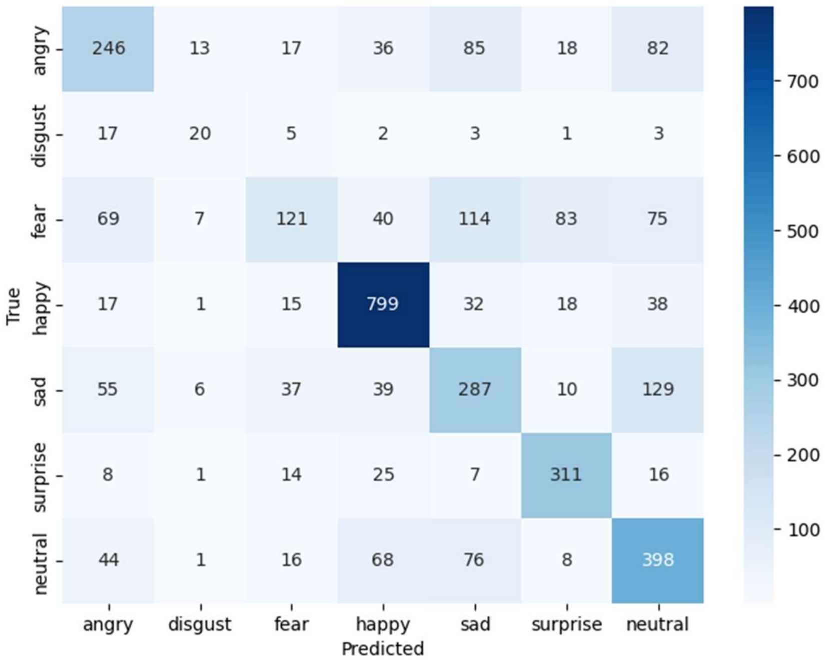


Figure 10 The confusion matrix for the deeper model

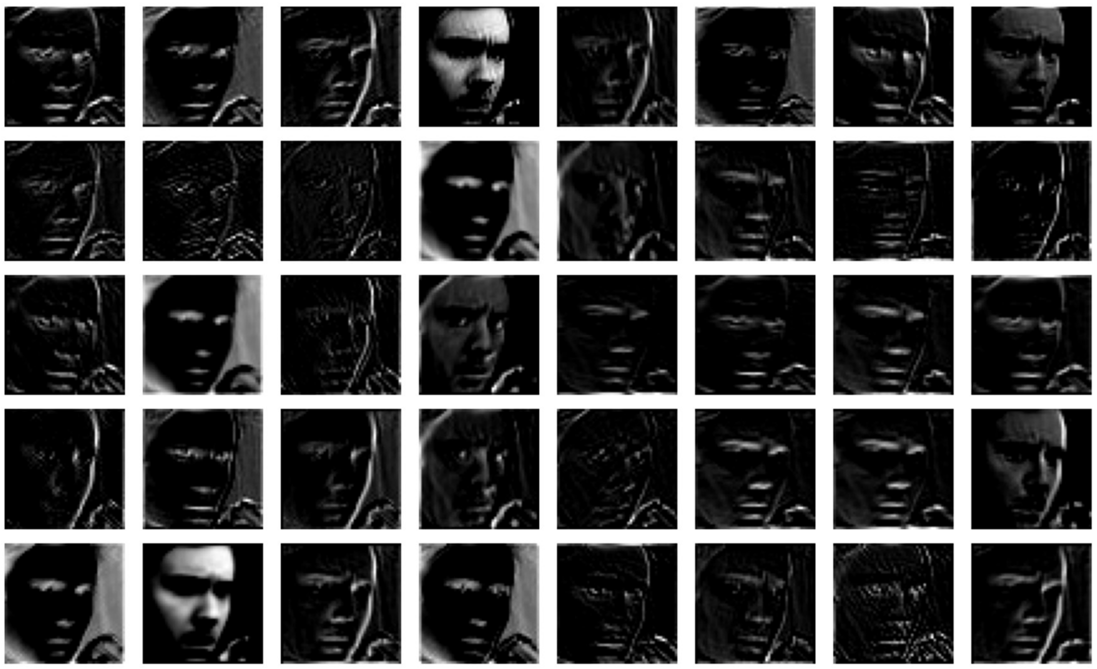


Figure 11 Visualization of activations of the first layer of the deeper model

## 6 Conclusion

Convolutional networks are a powerful tool in computer vision, solving problems that standard neural network models couldn't solve because of their simplistic design. Their ability to learn increasingly abstract representations through stacked layers makes them highly effective for complex visual understanding, which is needed in the detection of facial expressions. The best accuracy in this paper achieved a model with 5 convolutional and 2 fully connected layers, but this could be further improved with a larger dataset, more computational power and addition of new technology.

## References

[1] Face expression recognition dataset (2019). Available: https://www.kaggle.com/datasets/jonathanoheix/face-expression-recognition-dataset
[2] https://www.researchgate.net/figure/Convolution-operation-diagram_fig1_374552194
[3] https://medium.com/hitchhikers-guide-to-deep-learning/9-introduction-to-deep-learning-with-compute-vision-normalization-batch-normalization-ba5f60c77cf3
[4] https://medium.com/@utsavraj.ptn04/dropping-the-knowledge-bomb-understanding-dropout-layers-in-deep-learning-0612f517269d
[5] https://www.researchgate.net/publication/362742399_Convolutional_Neural_Network_CNN_A_comprehensive_overview
[6] Shima Alizadeh, Azar Fazel (April 2017). Convolutional Neural Networks for Facial Expression Recognition. Available: https://arxiv.org/abs/1704.06756
[7] https://cs231n.github.io/convolutional-networks/
[8] https://pytorch.org/