Maastricht University

Department of Data Science and Knowledge Engineering

Master's Program in Artificial Intelligence

Computer Vision

June 5, 2021

Assignment2 - Deep Learning for Emotion Classification

Caio Guirado
Dhruv Rathi

https://github.com/caioguirado/CV-Assignment-2

Contents

1 Introduction **2**

2 Implementation **2**

3 Experiments **2**

3.1 Number of Layers Effect . . . 2

3.2 Convolution Kernel Size Effect . . . 3

3.3 Pre-trained Architectures . . . 3

3.4 Learnt Features Visualization . . . 4

4 Feature selection - Experiments **4**

4.1 Motivation . . . 4

4.2 Xception Architecture . . . 5

4.3 Training Facial Landmarking model . . . 5

4.4 Dataset Generation . . . 5

4.4.1 Face Dataset . . . 5

4.4.2 Mouth Dataset . . . 6

4.5 Experiments . . . 6

4.5.1 Face Dataset Results . . . 6

4.5.2 Mouth Dataset Results . . . 6

5 Conclusion **7**

A Learned Filters **8**

Abstract

In this paper, Deep Learning for image classification is applied through Convolutional Neural Networks(CNN) to classify emotions from faces pictures. Experiments were conducted to investigate the differences in output by varying parameters related to the model's architecture and the use of fine-tuned pre-trained models. Further, we also contribute two new datasets, i.e. cropped facial region and cropped mouth region obtained using trained CNN models, and we discuss the results of these datasets for the task of emotion classification. Finally, the learnt convolution filters were plotted to reason about the the intuition of the features found.

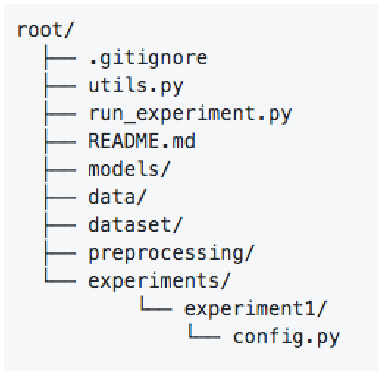
1 Introduction

Image classification is a popular task in computer vision. Several approaches have been developed over the years in order to accurately tell what is the correct label for a given image, but recently Deep Learning approaches have shown an enormous potential when trained with big amounts of data. One of the first successful applications of convolutional neural networks applied to a real dataset that achieved state of the art performance when it was released can be found in [2]. After that, different architectures were developed testing extension on CNNs parameters. Our goal is to implement a basic CNN model, train it based on the FER dataset [1], evaluate its performance and compare with other initial setups based on different parameters. The main parameters of interest are the number of layers, the kernel size of the convolutions, and pre-trained architectures. For the task of emotion recognition, it is evident that there are set of dominating features that stands out more than others, these features are mostly facial cues expressed by eyes and mouth, therefore, we implemented a facial landmark detection model, and produce two novel datasets, one of cropped faces and the other dataset is cropped portion of images which contains the lower part of the face, i.e., lips and mouth. We compare the results of these datasets on the task of emotion recognition against the provided benchmark dataset.

[1]https://www.kaggle.com/ashishpatel26/facial-expression-recognitionferchallenge

2 Implementation

In order to perform many different experiments, and don't lose track of documentation, a framework was built in Python with separation of concerns, so that basically a configuration file can be enough to automatically generate an experiment. The framework supports several data pre-processing and augmentation pipelines, models, and datasets, as well as custom parameters like loss function, learning rate, batch size. Figure 1 shows the framework overall structure. The results were measured over the public set of the original dataset.



```
root/
  ├── .gitignore
  ├── utils.py
  ├── run_experiment.py
  ├── README.md
  ├── models/
  ├── data/
  ├── dataset/
  ├── preprocessing/
  └── experiments/
          └── experiment1/
                └── config.py
```

Figure 1: Framework structure

3 Experiments

3.1 Number of Layers Effect

In order to understand how the addition of layers impacts the model performance, three CNNs were built, all of them with the same convolution parameters, but with incremental sizes of number of fully connected layers. Table 2 shows the information about the three models' architectures. Table 1 shows the results after training the models for 40 epochs. Figure 2 shows the training process of those models.

Exp	Accuracy	F1	Balanced Acc
CNN 1	0.80	0.80	0.78
CNN 2	0.83	0.83	0.81
CNN 3	0.77	0.77	0.76

Table 1: Models performance on public validation set

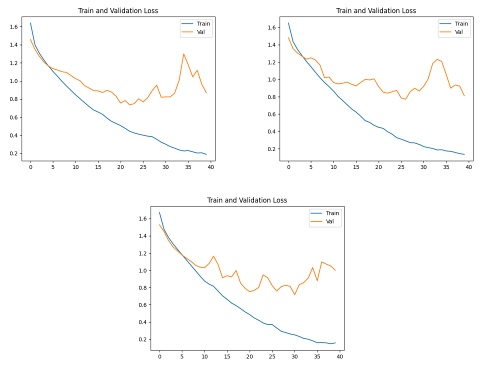


Figure 2: Top left: CNN 1; Top right: CNN 2; Bottom: CNN 3

In all of the architectures tested, the data shows that the performance on the validation set flats around the 20th epoch, being prone to overfit in the following iterations. The accuracy, f1-score and balanced accuracy performances doesn't show significant difference, Showing that the addition of many more fully connected layers in the end of the net, in this case didn't make much difference. Of course, many other things can be changed to try to make this layers addition to improve the model's performance, such as data augmentation, change the convolution parameters, add dropout, etc.

3.2 Convolution Kernel Size Effect

A new experiment was conducted to understand the effect of the convolution kernel size in the model's performance. In this, a comparison between CNN 1, presented in the previous section, and a cloned architecture, changed only to convolutions kernel size equals 3 was executed. We call this new version CNN 4. Figure 3 shows the training history of the new architecture presented. Table 3 presents the accuracy and the f1-score for both networks.

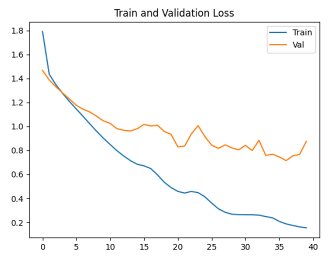


Figure 3: New CNN with kernel size = 3

From the training history, it's interesting to notice that the validation performance for CNN 4 doesn't flat over the 20th iteration, showing a continuous trend of decrease over epochs. The final metrics however, show a similar behaviour of both. On the intuition side, since the images are small in size, and the specific task of recognizing emotions can deal with very delicate and small regions of a face, it seems that smaller convolution kernel sizes might be able to process that local behaviour better.

3.3 Pre-trained Architectures

In this work a pre-trained architecture was also tested to compare with the initial configurations that were created without many parametric experimentation. GoogLeNet [4] pre-trained on ImageNet dataset was chosen to be compared with the other models already presented in the previous sections. In order to match the output size for our domain, the last fully connected layer was replaced by one with 7 output nodes. Table 4 presents the results of training the pre-trained GoogLeNet architecture.

The obtained results show that we obtained a better performance with the fine tuning of the pre-trained model. This follows the intuition that by having a much deeper structure not only on the fully connected part, but also on the convolution part, and by having already acquired knowledge, this architecture is able to outperform the previous presented ones.

Model	FC Layers
All (Convolution)	$Conv(out_channels = 10, kernel_size = 5)$ $MaxPool(kernel_size = 4)$ $Conv(out_channels = 10, kernel_size = 5)$ $MaxPool(kernel_size = 4)$
CNN 1	$Linear(kernel_size = 34^2 * 20, output_size = 120)$ $Linear(kernel_size = 120, output_size = 250)$ $Linear(kernel_size = 250, output_size = 7)$
CNN 2	$Linear(kernel_size = 34^2 * 20, output_size = 250)$ $Linear(kernel_size = 250, output_size = 120)$ $Linear(kernel_size = 120, output_size = 120)$ $Linear(kernel_size = 120, output_size = 120)$ $Linear(kernel_size = 120, output_size = 7)$
CNN 3	$Linear(kernel_size = 34^2 * 20, output_size = 250)$ $Linear(kernel_size = 250, output_size = 120)$ $Linear(kernel_size = 120, output_size = 120)$ $Linear(kernel_size = 120, output_size = 120)$ $Linear(kernel_size = 120, output_size = 120)$ $Linear(kernel_size = 120, output_size = 50)$ $Linear(kernel_size = 50, output_size = 50)$ $Linear(kernel_size = 50, output_size = 7)$

Table 2: Three custom CNNs created for experiments. First row shows the common convolution part of the network, and each following row describes the fully connected configuration of each.

Exp	Accuracy	F1
CNN 1	0.80	0.80
CNN 4	0.80	0.80

Table 3: Models performance on public validation set

3.4 Learnt Features Visualization

One thing that it's possible to do after the model is trained is to visualize the activations of an input signal and observe their output. This can help to understand better what was determinant to the model for concluding that certain image belonged to a specific class. The results are shared in Annex A. From them, it's possible to notice that the model learned to focus on important regions to recognize features, such as eyes and mouth.

4 Feature selection - Experiments

4.1 Motivation

For the task of emotion recognition, it is evident that a set of features dominate over others, these features are mostly facial cues that express emotions such as eyes - frowning, crowed, crying; mouth - smiling, etc. This motivates us to select these features explicitly and therefore help the model in learning them better, rather than over-fitting on noisy features that are recurring but irrelevant. We do this with the help of a facial landmark detection model that servers us with 68 facial points and we produce two sets of datasets from it, as discussed below. The landmarking model is CNN based architecture, derived from [1].

class	precision	recall	f1-score
0	0.87	0.83	0.85
1	0.90	0.77	0.83
2	0.84	0.82	0.83
3	0.91	0.96	0.93
4	0.81	0.84	0.83
5	0.92	0.91	0.91
6	0.88	0.85	0.86
accuracy	0.87	0.87	0.87
macro avg	0.88	0.85	0.86
weighted avg	0.87	0.87	0.87

Table 4: Pre-trained GoogLeNet Performance Results

4.2 Xception Architecture

The Xception architecture consists of 36 convolutional layers structured into 14 linear residual connected modules, with the exception of the input and the output module. The model modules can broadly be divided into three levels, *Entry flow*, *Middle flow*, *Exit flow*, these modules are interdependent and the accompanying code richly follows this structure of implementation. The architecture was implemented with PyTorch library.

4.3 Training Facial Landmarking model

The Xception net was trained on [3], the *ibug 300W Large Face Landmark Dataset* which contains 7000 images with labels of 68 facial landmarking coordinates per face. The dataset is preprocessed with random application of partial cropping of the face and corresponding labels, along with rotation, changes in image saturation and hues. The training was performed on Tesla P100 GPU and took 6 hours.

4.4 Dataset Generation

As the model is trained now, we further generate two datasets as discussed earlier.

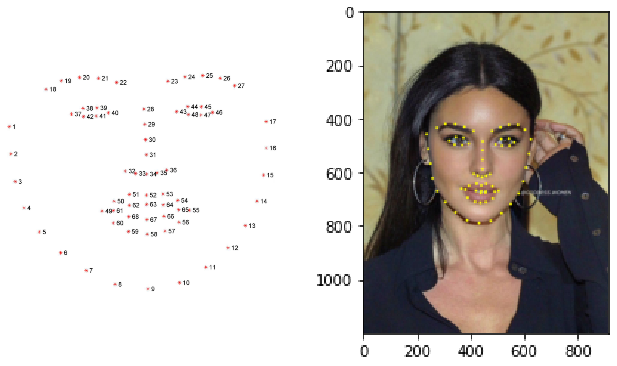


Figure 4: 68 Facial landmarks and Model sample output

4.4.1 Face Dataset

The first data generation is based on the premise that other than the face, all features present in an image, e.g. the background, hair style, hair color, etc do not play a generalised role in emotion detection which the model might learn and overfit. Therefore, we built a pipeline to run each image present in the Kaggle dataset and generate it's corresponding 68 facial landmarkings, the image is then cut from point 1 with width = [1-17] and point 20 to height = [9-20]. A sample of the output dataset is shown in figure 5.

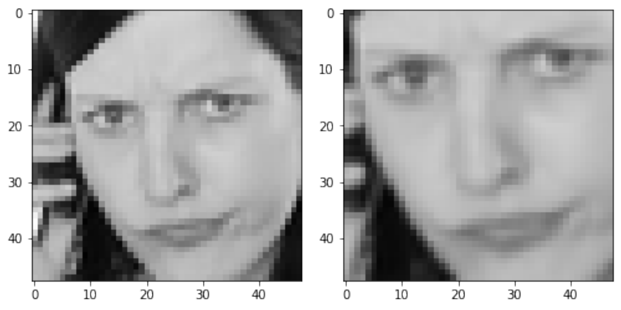


Figure 5: Face Dataset sample 1

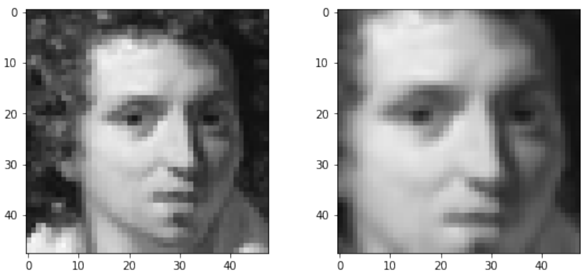


Figure 6: Face Dataset sample 2

4.4.2 Mouth Dataset

The premise for generating this dataset is that majority of the emotions involve the movement of mouth muscles.

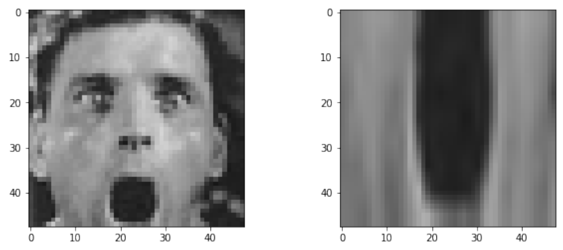


Figure 7: Mouth Dataset sample 1

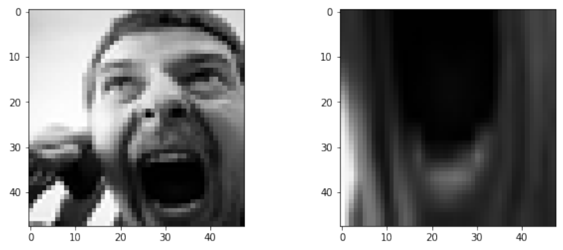


Figure 8: Mouth Dataset sample 2

4.5 Experiments

The new datasets are used for training the Google Net based CNN model described in 3.3.

4.5.1 Face Dataset Results

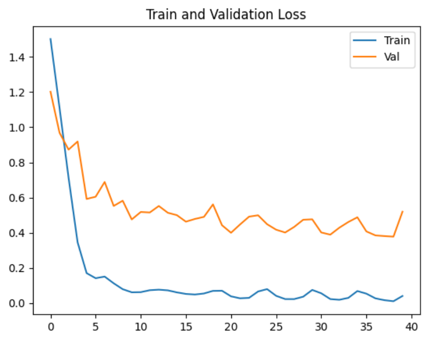


Figure 9: Training using Face dataset

	precision	recall	f1-score
0	0.86	0.86	0.86
1	0.93	0.84	0.88
2	0.90	0.84	0.87
3	0.93	0.95	0.94
4	0.84	0.89	0.86
5	0.93	0.92	0.92
6	0.88	0.87	0.88
accuracy	0.89	0.89	0.89
macro avg	0.90	0.88	0.89
weighted avg	0.898	0.897	0.897

Table 5: GoogLeNet Model results with Face Dataset

From 9 5 we can clearly see that Googlenet on FaceDataset **outperforms the vanilla dataset**, although similar trend can as can be observed as in 3 that the validation loss becomes stagnant after 20 epochs.

4.5.2 Mouth Dataset Results

From 10 6, the results of GoogleNet trained on Mouth dataset are at par with the 4.5.1 and **outperforms the vanilla dataset**.
This experiment shows that our feature selection leads to dominant training and noise reduction.

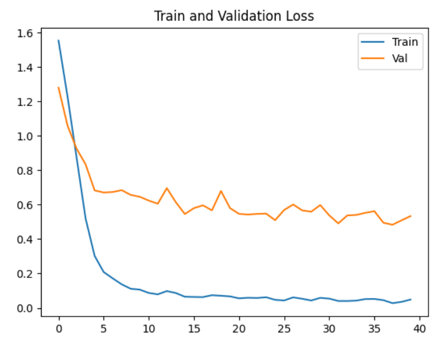


Figure 10: Training using Mouth dataset

	precision	recall	f1-score
0	0.89	0.86	0.88
1	0.86	0.88	0.87
2	0.89	0.82	0.86
3	0.93	0.96	0.94
4	0.87	0.86	0.87
5	0.90	0.94	0.92
6	0.86	0.90	0.88
accuracy	0.89	0.89	0.89
macro avg	0.89	0.89	0.89
weighted avg	0.899	0.899	0.899

Table 6: GoogLeNet Model results with Mouth Dataset

5 Conclusion

In this work, we learned how to work with deep convolutional neural networks applied to an emotion classification problem. The experiment conducted helped to understand the importance of choosing good parameters to the final results. For instance, one might prefer to look at a higher batch size, higher learning rate, less augmentations and a less deeper network architecture and fine tune pre-trained models if the intention is to increase the training speed, of course being cautious to not jeopardize the model's performance. Other parameters like filters size, pooling size, dropout can help the network to focus on regions in a certain way, helping with the domain of the problem application and generalization. We also learnt that feature selection plays an important part in deep learning problems, and reducing the suitable noise from the datasets is an important step.

References

[1] François Chollet. Xception: Deep learning with depthwise separable convolutions, 2017.

[2] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. Imagenet classification with deep convolutional neural networks. *Commun. ACM*, 60(6):84–90, May 2017.

[3] Christos Sagonas, Georgios Tzimiropoulos, Stefanos Zafeiriou, and Maja Pantic. 300 faces in-the-wild challenge: The first facial landmark localization challenge. In *2013 IEEE International Conference on Computer Vision Workshops*, pages 397–403, 2013.

[4] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott E. Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. *CoRR*, abs/1409.4842, 2014.

A Learned Filters

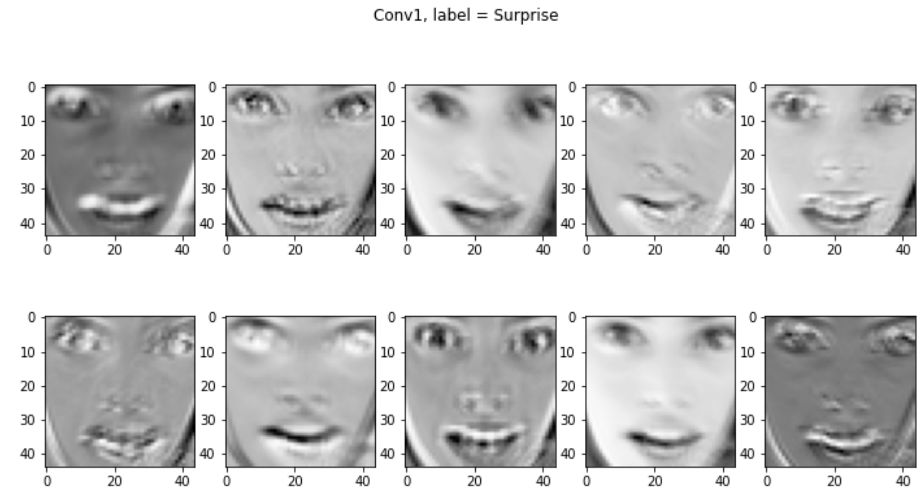


Figure 11: Activations from first convolution layer of CNN 1

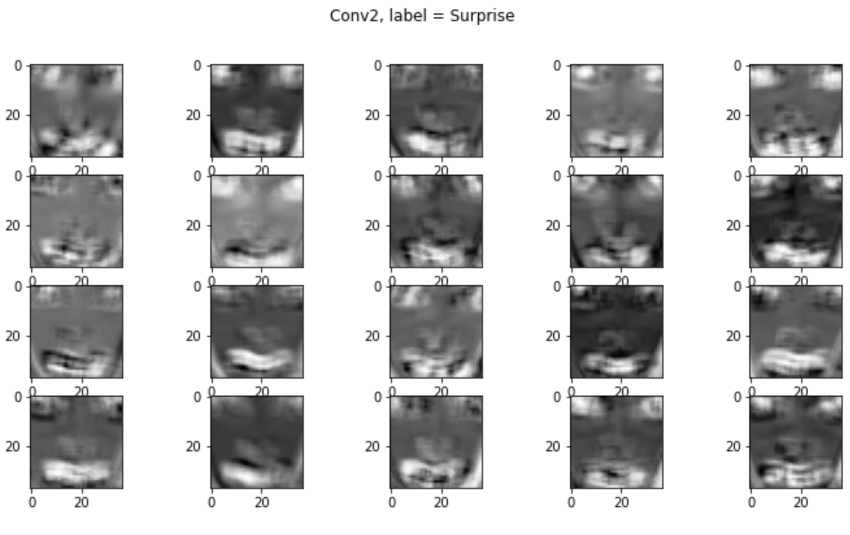


Figure 12: Activations from second convolution layer of CNN 1

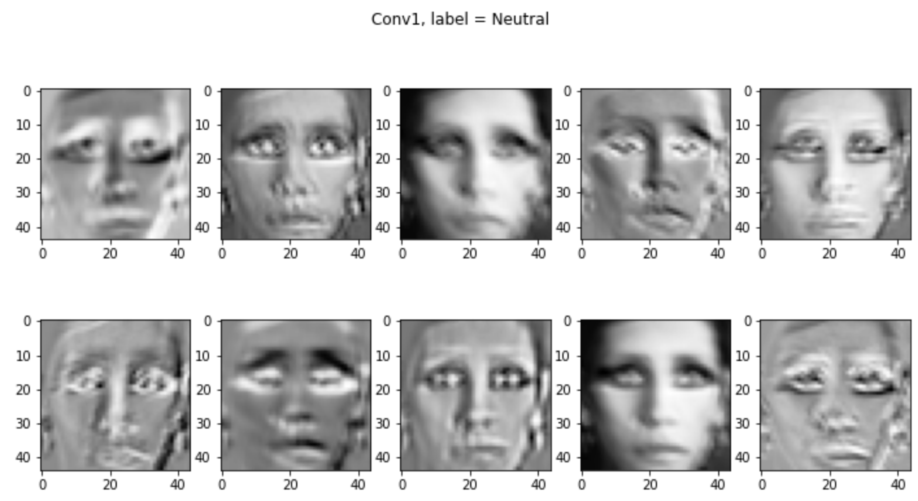


Figure 13: Activations from first convolution layer of CNN 1

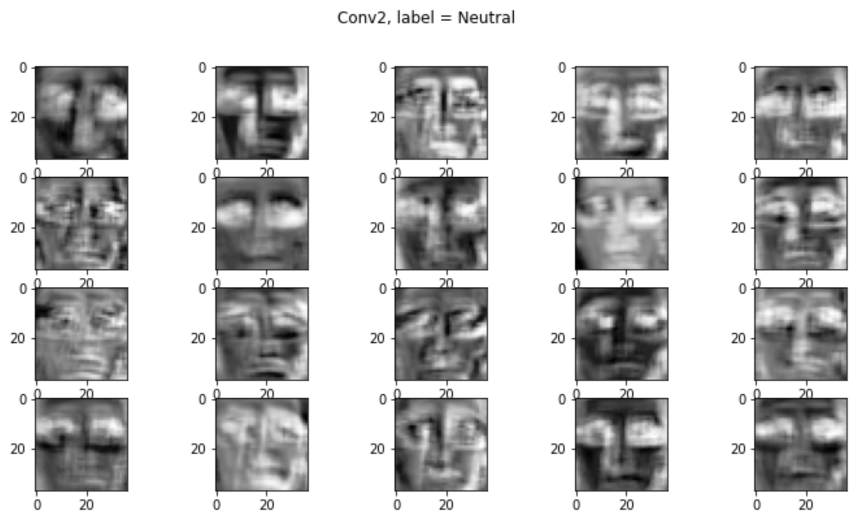


Figure 14: Activations from second convolution layer of CNN 1

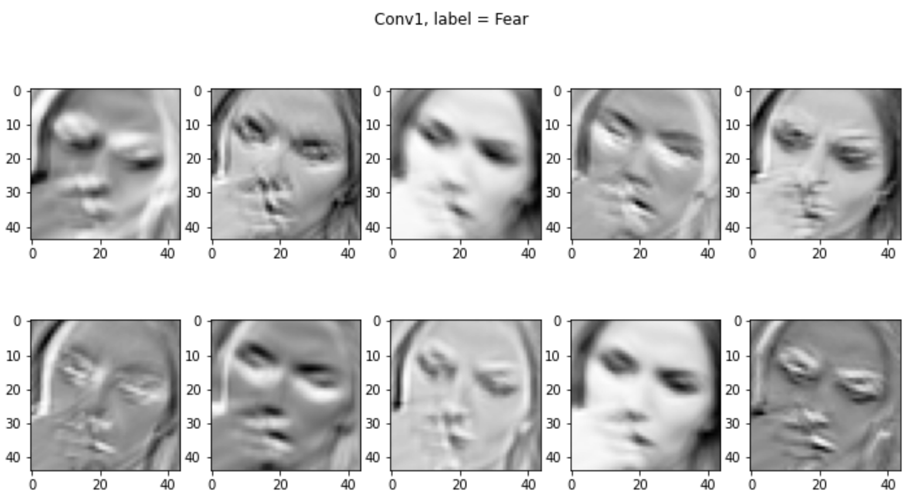


Figure 15: Activations from first convolution layer of CNN 1

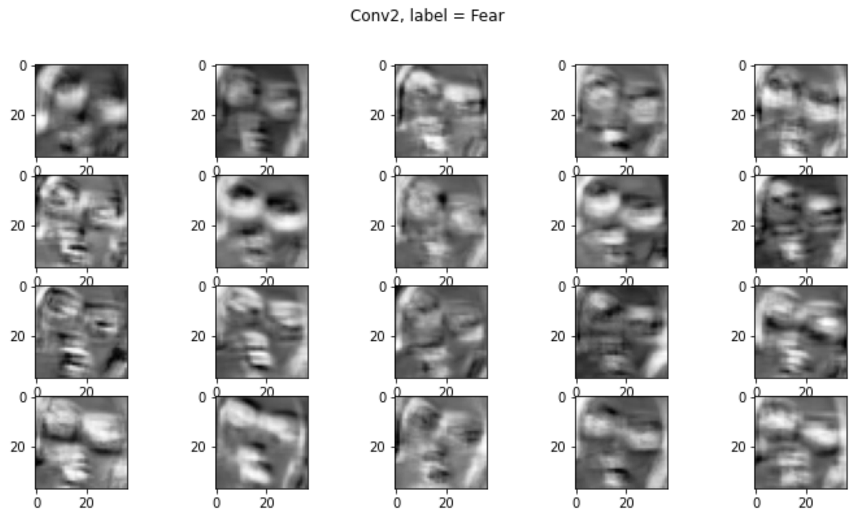


Figure 16: Activations from second convolution layer of CNN 1