CHALMERS
UNIVERSITY OF TECHNOLOGY

Classification of Facial Expressions

Project assignment: Part 3 in Spatial Statistics and Image Analysis, TMS016

SANTIAGO AMAYA
MAITREYA DAVE

Department of Mathematical Sciences
CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden 2020

Discussion and Results of Classification of facial expressions using various methods

SANTIAGO AMAYA
MAITREYA DAVE
Department of Mathematical Sciences
Chalmers University of Technology

Abstract

The idea behind this project is to understand how artificial neural networks, specifically convolutional neural networks work when tested on facial expressions. For an algorithm to successfully group similar expressions, it must be able to recognize the various arrangements of facial features (eyes, eyebrows, lips and more) in combination. The problem gets more difficult to solve when one these feature's may be hidden (example sunglasses) or the lighting in the image was inadequate (half pitch black half colour). If the photo is taken from a different angle it indicates a different perspective which is easy for humans to classify with just little-to-no context. Our aim is to study how different architectures perform and overcome such challenges, obtaining a comparison on at least 2 different data-sets.

Keywords: facial, expression, classification, pattern, recognition, neural, net, cnn, statistical, learning.

Contents

1 Data Sources **1**
- 1.1 Candidates . . . 1
  - 1.1.1 FER 2013 . . . 1
  - 1.1.2 Japanese Female Facial Expression (JAFFE) . . . 2
- 1.2 Selection . . . 3

2 Methods **4**
- 2.1 Prepossessing steps . . . 5
- 2.2 Our CNN architecture . . . 5
  - 2.2.1 Feature extraction . . . 5
  - 2.2.2 Classification . . . 6

3 Architectures and Results **7**
- 3.1 First Architecture: High over fit . . . 7
- 3.2 Second Architecture: Medium over fit . . . 8
- 3.3 Third Architecture: No overfit . . . 9
- 3.4 Test Set . . . 10
  - 3.4.1 Confusion Matrix . . . 11
  - 3.4.2 Some classified images . . . 11
  - 3.4.3 Visualizing the Process . . . 12
- 3.5 Testing on JAFFE Dataset . . . 15
  - 3.5.1 Training . . . 15
  - 3.5.2 Testing . . . 15

4 Conclusion **17**

Bibliography **17**

1

Data Sources

1.1 Candidates

Following are a list of data sources we have found and consider for use in our project.The databases are more classic in the sense that they are comprised of facial images of a number of subjects labelled with the correct expression. The last data-set, as numbered below, is considered a dummy data-set, where female models acted out different expressions in a controlled setting, and we will use it to benchmark the model that we create for the real data-sets. All data-sets focus mainly on discrete emotion classification.

1.1.1 FER 2013

Original Paper: [1]
Link To Dataset: Click Here

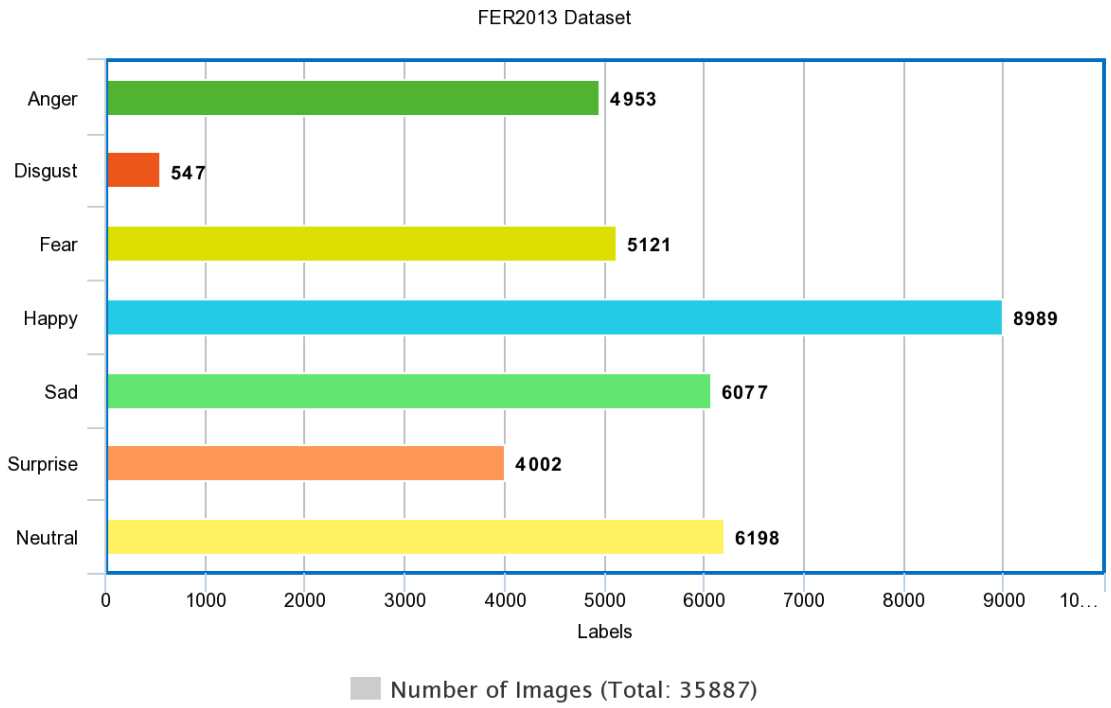

Figure 1.1: FER 2013 Dataset

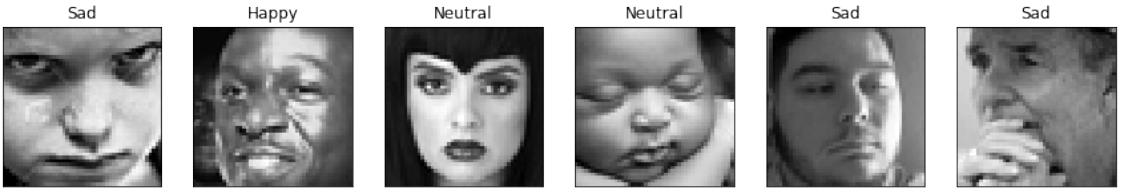

Figure 1.2: FER 2013 Random Sample

1.1.2 Japanese Female Facial Expression (JAFFE)

Original Paper: [3]
Link To Dataset: Click Here

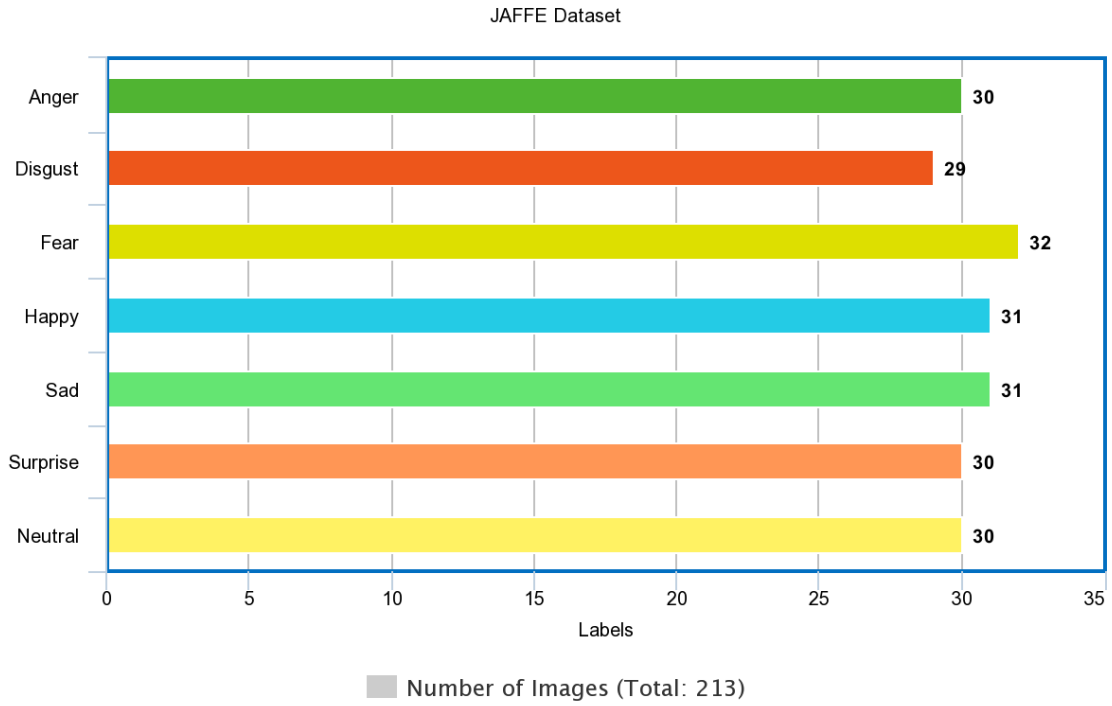

Figure 1.3: JAFFE Dataset

Figure 1.4: JAFFE Random Sample (each model has 3 or 4 poses for the same expression)

1.2 Selection

After a more detailed investigation of the data-sets, we decided to use the Facial Expression Recognition (FER) one, which we considered had better characteristics for the task in hand. First of all it is comprised of more than 30 thousand images, which is a size appropriate for a deep learning task (if not a bit small), which we have chosen as the method to identify expressions in face images. It also has fantastic variety, with images from different races, genders, and even movie characters, this makes the task more challenging and we consider it is an opportunity to fully explore this method. Finally, the format of the data-set (csv) was the easy to load and use in our preferred programming language (Python).

As a good practice in deep learning, one usually tries a new network architecture on a data-sets with which one can easily obtain comparable and well-known results. This is the case for instance of data-sets like MNIST that is used as a "first test". In our case, given the nature of our task, we decided to use the JAFFE data-set, which consists of 213 samples taken from 10 models, and can give us a starting point for the model.

2
Methods

For the task of expression recognition in images, we consider that the best approach is to use convolutional neural networks, a state of the art technique that has exploded in the last few years due to its outstanding results in the field. Neural networks as a whole allows us to obtain remarkable results that were not possible a decade ago, but convolutional networks specialize in image analysis by using combinations of different pixels to extract features through "filters" found using the usual back-propagation of neural networks. Depending on the number of layers, more complex features can be obtained, and thus we want to use at least 3 convolutional layers (with their respective pooling afterwards), in addition to some fully connected layers (for final classification) in order to obtain a model that allow us to classify the images in our data-sets according to the facial expressions in the labels.

We will also have a short phase of data-set visualization, and evaluation, in order to select those with highest quality and potential. It is possible that some prepossessing, balancing and fixing is in order.

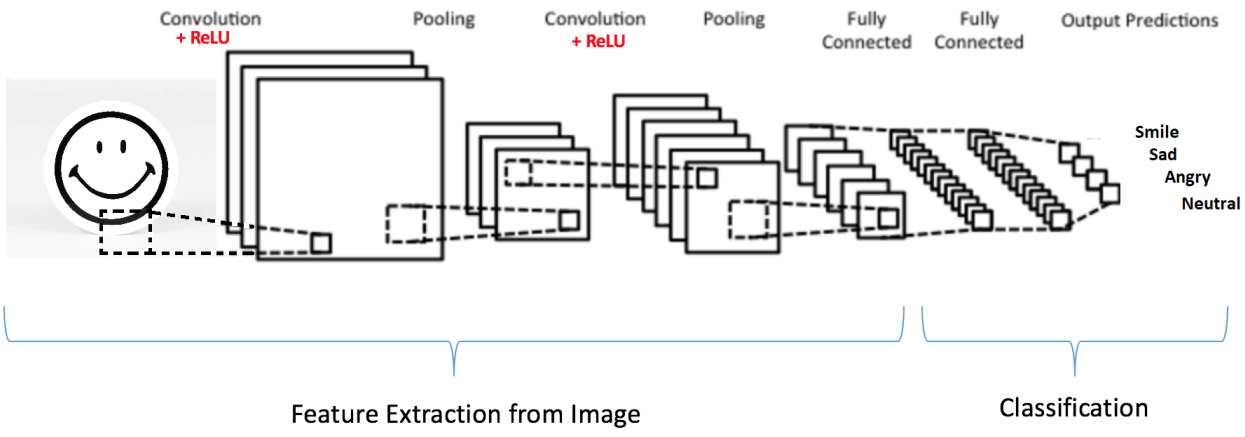

Figure 2.1: Preliminary diagram

2.1 Prepossessing steps

There are many options to follow in this step of the creation of the model, and it can improve the performance of the network if done correctly. For instance looking at figure 1.1 we can see that the images encase the faces nicely without much unnecessary background. This is not the case of the images in figure 1.2 so the first thing we did was crop the images in JAFFE data-set to avoid using information that does not contribute to the identification of gestures. Even in the first data-set we applied some cropping, obtaining a shape of 36x36 instead of 48x48, which gives an enclosing of just the necessary characteristics to recognize an expression. Apart from this, there are some data augmentation techniques, like random cropping, or horizontal flipping of the images, but in this case we did not consider them necessary. The only relevant reprocess we conducted on our data is the usual normalization of the pixel values, making the model more numerically stable and allowing it to run faster.

2.2 Our CNN architecture

As mentioned before, our main goal is to classify images from FER2013, so the architecture decisions are taken in consideration to this data only. The JAFFE data is only to be used as a benchmark of our model. We have divided our network in two sections, as shown in figure 2.1:

1. Feature extraction
2. Classification

2.2.1 Feature extraction

This section consists of blocks of grouped convolutional layers, followed by max pooling layers. The convolutional layers are self adjusting filters, that learn how to better extract relevant information from the previous layers in order to supply the classification section with the best possible combination of features from which to make predictions that minimize the Loss.

The base concept of our architecture consists of 3 blocks. The first one would act as an horizontal and vertical edge detector, an intensity normalization, a blurring of irrelevant information, a sharpening or smoothing of the image, and still much more. All this accomplished through the use of 16 filters in the first layer, followed by at least another two convolutional layers that would combine to obtain more complex iterations or "visualizations" of the original image. After this first block (as well as the others) a down sampling is performed using a Max pooling layer in order to reduce the sensitivity of the output feature map to the location of the input, giving our model “local translation invariance.” (a max pooling layer is the assumption that the more relevant information for our model is given by the maximum value inside the kernel box used for the pooling)

After down sampling the input to half its size, the second block would further try to recognize more complex shapes, like maybe eyes, brows, nose, and mouth, and

rearrange the important features across increasingly many channels. The channels in the hidden layers, are given by the number of filters we choose, which are stacked one after the other to generate a volume. The general idea of CNN is to take as an input an image with width and height greater than depth: maximum 3 channels (RGB), one in our case as our images are in gray scale, and output a volume with with and high smaller than the depth. At the end of the second block, another Max Pooling layer is applied to halve the width and height of output of the block.

Finally, the third block would group relevant combinations of features into different channels, in a compressed representation of the inputs. This block's output would also be down sampled and then flattened (if needed) to be passed as an input to the classification section of the model.

We applied Rectified Linear (Relu) activation function to the output of every convolutional layer, this is done in order to obtain non linear relationships.

2.2.2 Classification

This section consists of two fully connected (dense) layers which goal is to use the previously extracted features, and use them to classify each image into one of the 7 categories: Sad, Happy, Neutral, Angry, Disgust, Surprised, Fear. Before and after the first dense layer we added a dropout layer, which function is to drop some of the neurons with a given probability (0.4-0.5 in our case), in order to reduce over fitting of the model. We applied the Relu to the output of the first dense layer, and Sigmoid to the output of the second, which outputs a value between 0 and 1 that is used as the probability of belonging to a class.

The Loss function we used was the Categorical Cross Entropy Loss, and paired it with Stochastic Gradient descent as our optimization algorithm.

3

Architectures and Results

There is a saying with neural networks that say: "Start Over fitting and then work backwards", and we thought it would make sense to start with something like that: an architecture that would over fit the data set, and gradually decrease its complexity to obtain a good model and some intuition as well.

Following we describe the structural elements used in the different architectures, which have been chosen by investigating different famous networks like VGG16, or GoogleNet:

- Filter: we used a specific type of filter which has a size of 3x3, and is applied using a stride (step) of 1 in each direction, and a padding of 1. This filter is good to obtain the expected details and keeps the width and height sizes of the input.
- Max Pooling: the max pooling is an operation where we keep the maximum value of those inside the pooling box. We used a size of 2x2 with a stride of 2 in each direction and no padding, in order to reduce the width and height of the input by half.

In the next sections when we refer to filter of max pooling we mean this two motioned before, unless otherwise specified.

3.1 First Architecture: High over fit

Our first architecture had 9 convolutional layers in the feature extraction section, and 2 dense layers in the classification section. Next a brief description of the network

- Layer 1: takes as input the image with 1 channel and size 36x36, and applies 16 filters to generate a output volume of 16x36x36.
- Layer 2: takes as input the 16x36x36 volume from the previous layer and applies another 16 filters with the same characteristics as before.
- Layer 3: takes the new 16x36x36 volume and applies 32 filters, but now in this layer a max pooling is applied to reduce the size by 2 (not the depth). This is the end of the first block.
- Layer 4: from the previous block this layer takes a volume of 32x18x18, and applies 32 filters, keeping the volume of the same size but with more complex relationships between features.

- Layer 5: this layer is the same as the previous one, only increasing the complexity that can be obtained.
- Layer 6: this layer applies 64 filters, and a max pooling to obtain a volume of 64x9x9. This is the end of the second block
- Layer 7 & 8: these layers apply 64 filters to the input each.
- Layer 9: this layer applies 128 filters and a max pooling, obtaining a final volume of 128x4x4 for the feature extraction section.
- Layer 10: this layer takes and input vector of 2048 elements (so the volume of the previous block has to be flattened) and outputs a vector of 556 elements.This is the first dense layer of the classification section. Before and after this layer we applied a dropout of 0.5 in order to reduce overfitting.
- Layer 11: this is the final layer which outputs a vector of 7 elements that will be passed to a Softmax function to obtain the probabilities for belonging to each class.

According to this, the feature extraction section has about 4320 trainable parameters, and the classification has more than 1.1 million. With this we can see that the best approach is to reduce the number of elements resulting from the flattening of the final convolutional layer.

Now we can proceed to observe the performance of this architecture in the following plots which show the training and validation loss as well as the respective accuracies during 50 epochs of training.

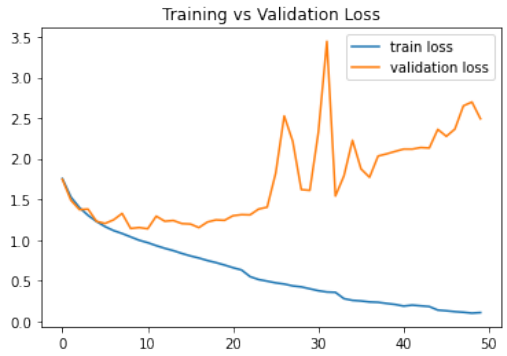

Figure 3.1: High overfit Loss

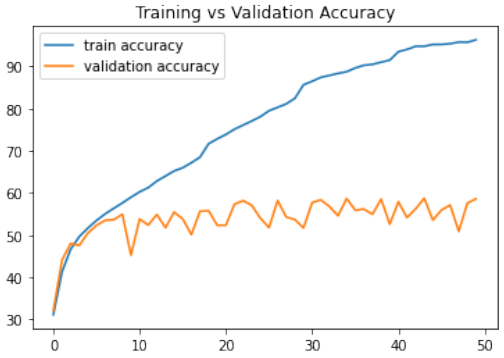

Figure 3.2: High overfit Accuracy

It is clear from Figure: 3.1 that the model starts overfitting at around 10 epochs and becomes very unstable after 20. The training accuracy showed in Figure: 3.2 goes up to more than 95% while the validation accuracy oscillates around 50% with a maximum of 56-57%. This model will thus be discarded.

3.2 Second Architecture: Medium over fit

Our second architecture had 6 convolutional layers and 2 dense layers:

- Layer 1: takes as input the image with 1 channel and size 36x36, and applies 16 filters to generate a output volume of 16x36x36.

- Layer 2: takes as input the 16x36x36 volume from the previous layer and applies 32 filters with the same characteristics as before.Then it applies a max pooling to obtain a 16x18x18 volume.
- Layer 3: this layer only applies 32 filters to extract more complex relationships.
- Layer 4: from the previous block this layer takes a volume of 32x18x18, and applies 64 filters, and a max pooling, obtaining a volume of 64x9x9.
- Layer 5: this layer is the same as the previous one, obtaining a volume of 64x4x4.
- Layer 6: this layer applies 128 filters, and a max pooling to obtain a volume of 128x2x2. This is the end of the feature selection phase.
- Layer 7: this layer takes the flattened output from the previous section which has 512 elements and outputs 56.
- Layer 8: this is the final layer which outputs a vector of 7 elements that will be passed to a Softmax function to obtain the probabilities for belonging to each class.

According to this, the feature extraction section has about 3360 trainable parameters, and the classification parameters now have been reduced to about 29 thousand.

Now we can proceed to observe the performance of this architecture in the following plots which show the training and validation loss as well as the respective accuracies during 50 epochs of training.

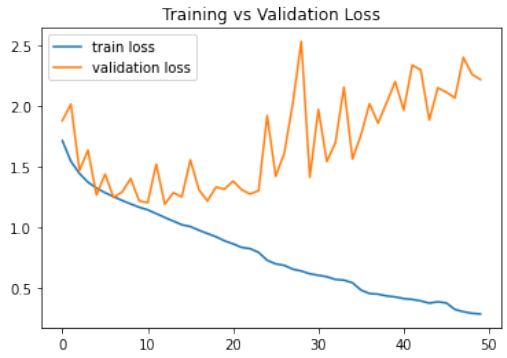

Figure 3.3: Med overfit Loss

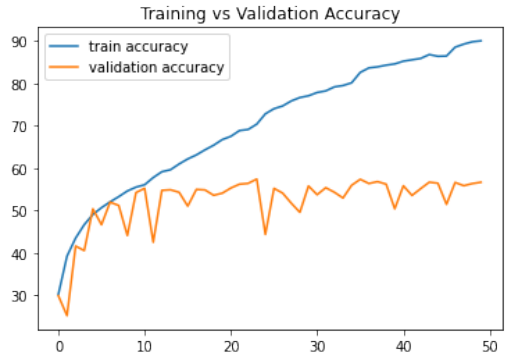

Figure 3.4: Med overfit Accuracy

It is clear now from the figures that the overfitting has been reduced, in Figure: 3.3 we can see that the validation loss starts increasing at around 20 epochs which is twice the number as before. The validation accuracy (Figure: 3.4) on the other hand does not seem to improve that much, although it seems more stable.

3.3 Third Architecture: No overfit

Finally, in our last architecture we reduce a little our feature extraction parameters but increase a bit the classification ones, this as an attempt to increase validation performance and still avoid overfitting. This architecture has 4 convolutional layers and 2 dense layers:

- Layer 1: takes as input the image with 1 channel and size 36x36, and applies 16 filters to generate a output volume of 16x36x36.

- Layer 2: takes as input the 16x36x36 volume from the previous layer and applies 32 filters with the same characteristics as before.Then it applies a max pooling to obtain a 32x18x18 volume.
- Layer 3: from the previous block this layer takes a volume of 32x18x18, and applies 64 filters, and a max pooling, obtaining a volume of 64x9x9.
- Layer 4: this layer is the same as the previous one, obtaining a volume of 64x4x4.
- Layer 5: this layer takes the flattened output from the previous section which has 1024 elements and outputs 256.
- Layer 6: this is the final layer which outputs a vector of 7 elements that will be passed to a Softmax function to obtain the probabilities for belonging to each class.

According to this, the feature extraction section has about 1760 trainable parameters, and the classification sections has about 265 thousand parameters.

Now we can proceed to observe the performance of this architecture in the following plots which show the training and validation loss as well as the respective accuracies during 50 epochs of training.

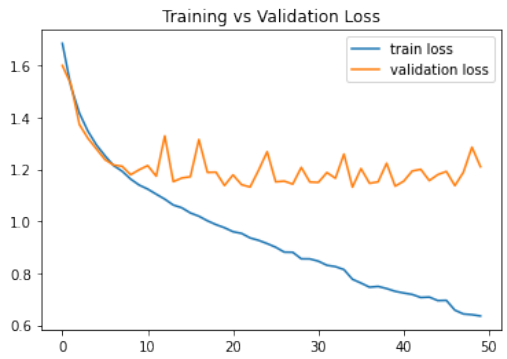

Figure 3.5: No overfit Loss

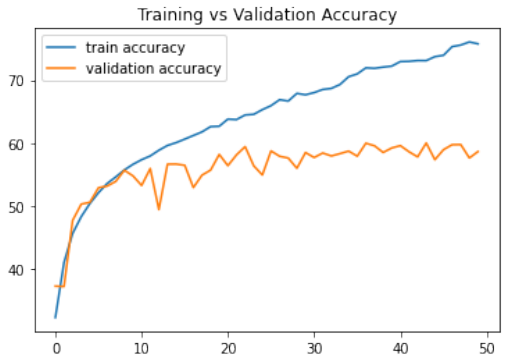

Figure 3.6: No overfit Accuracy

We can see in Figure: 3.5 that the model is finally not overfitting but the loss starts decreasing very slowly which means there is still room for improvement. In Figure: 3.6 we can clearly appreciate a better accuracy, which reaches 60% at some points.

Even though there is still room for improvement, obtaining an accuracy of 60% in this data set is a good achievement, specially for this kind of task. Without considering the heavy unbalance of some classes, a random guess would have probability of about 14%, which gives some context to our results.

Now that we have selected the architecture and trained our model we can perform some final selection in our separate testing set.

3.4 Test Set

As mentioned before we took a sample of 3500 images from the data and selected it as test set in order to prove our final model.

3.4.1 Confusion Matrix

In Figure: 3.7 we present the confusion matrix of the testing set

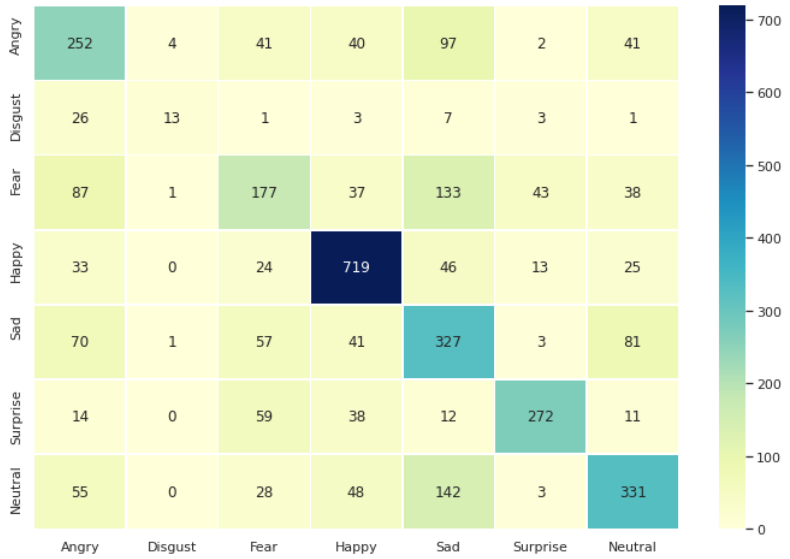

Figure 3.7: Classification of the separate test set

The "Happy" expression seems to be the easiest to classify, with an estimated identification probability of 77.64%, the overall estimated identification probability is $(252 + 13 + 177 + 719 + 327 + 272 + 331)/3500 = 59.74\%$

3.4.2 Some classified images

Figure 3.8: Predictions in the test set

3.4.3 Visualizing the Process

Here we preset a visual view of our CNN Architecture. We start by taking a random image from our test dataset, shown in Figure: 3.9.

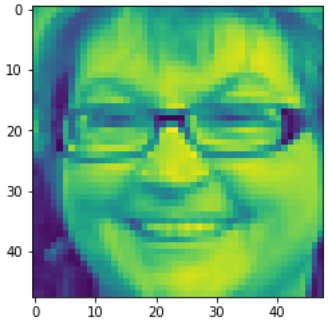

Figure 3.9: Test Image

All the filters used in our model are 3x3 Filters. In Figure: 3.10 we show the filters generated by our first convolution layer.

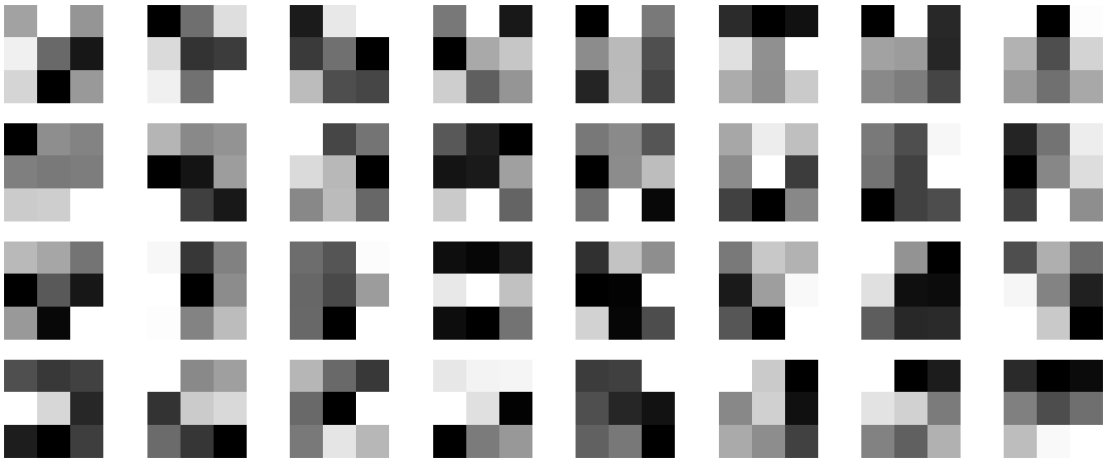

Figure 3.10: Filters of Convolution Layer 1

Figure: 3.10, represents 32, 3*3 filters. Each of these filters will then be convoluted with the input image to extract features from the image. As one can see a 3*3 filter is basically a 9 blocks of different grayscale values arranged in a square matrix of size 3. The grayscale intensity of the each block shall decide, which part of the input image shall be made brighter or darker.
After convoluting the filters and the input image what we get is a feature map, presented in Figure: 3.11 for layer 1.

Figure 3.11: Feature Map from Convolution Layer 1

Figure: 3.11, gives us an insight in what our CNN Model see's the image as after passing it through our first model.
Since in our model the number of filters we proceed to next year, hence we restrict ourselves to displaying only 32 feature maps from each convolution layer.

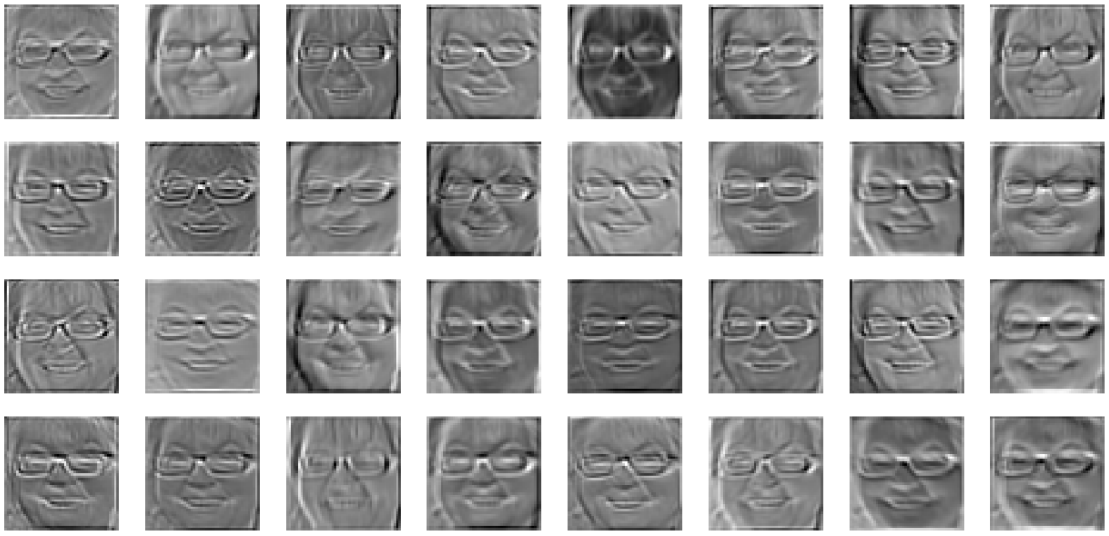

Figure 3.12: Feature Map from Convolution Layer 2

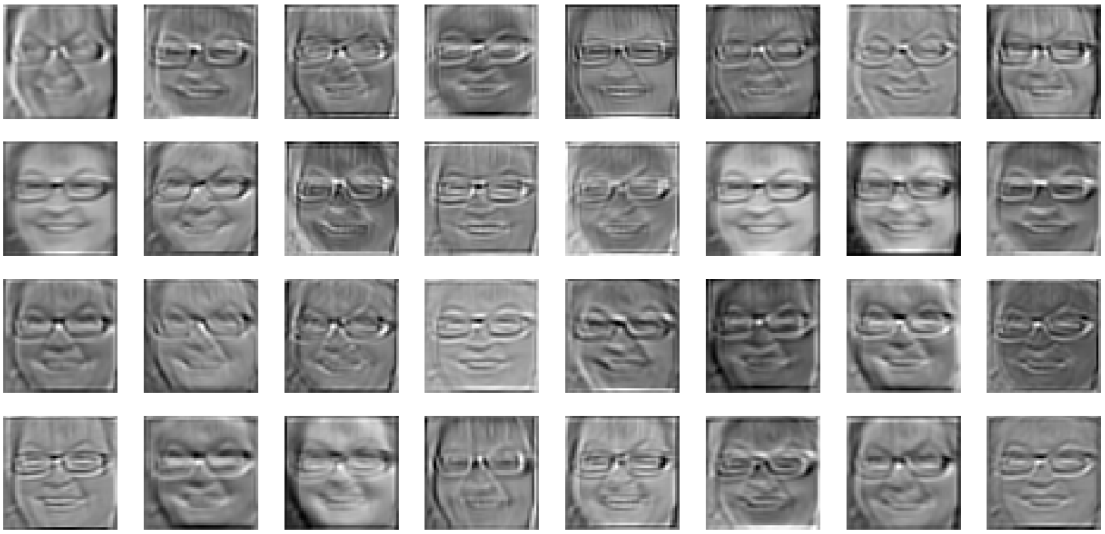

Figure 3.13: Feature Map from Convolution Layer 3

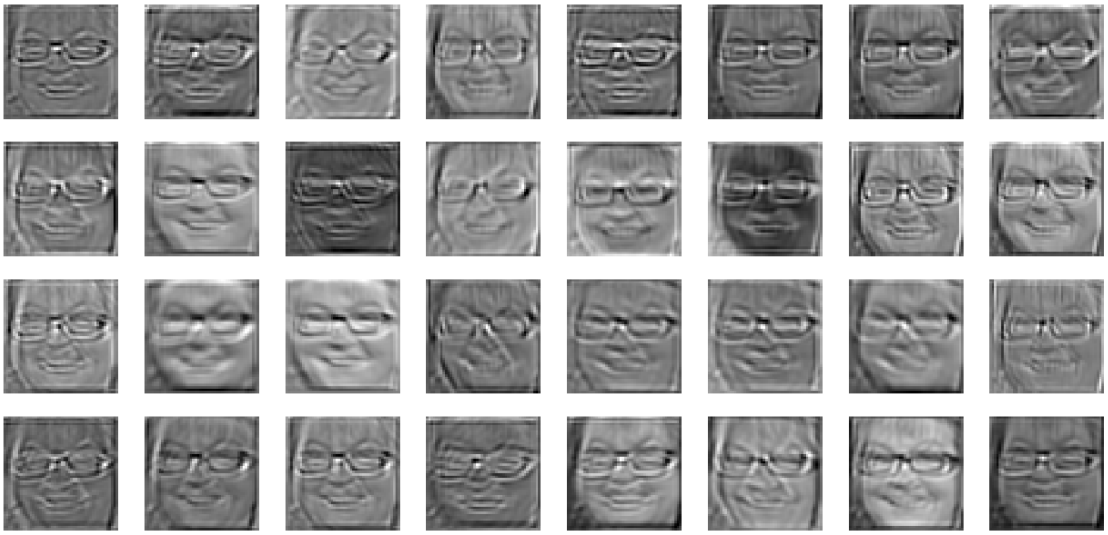

Figure 3.14: Feature Map from Convolution Layer 4

As we can see from Figure: 3.12, 3.13, 3.14, as we advance the CNN extracts more intricate features. The final work left todo is to pass the final feature map from convolution layer 4 to the linear layer, which is a simple fully connected neural network. In the end the output from the Linear Layer is passed to softmax function for calculating the probabilities.

3.5 Testing on JAFFE Dataset

To check the performance of our best model i.e. Third Architecture we decided to train it on the JAFFE Dataset. The dataset contains the same 7 emotions to be classified as the FER2013 data. There are about 213 samples available of 256 * 256 static gray-scale images.

3.5.1 Training

All the parameters used for training the model were same as used for our "Third Architecture".

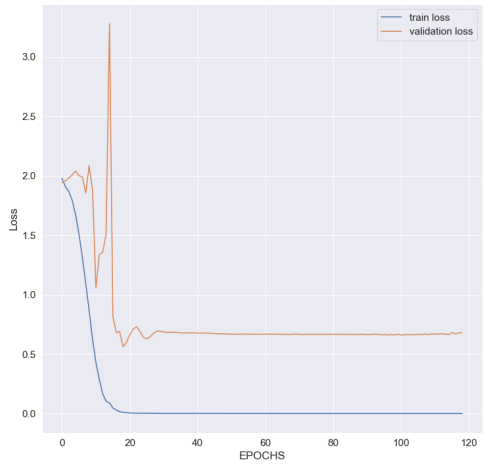

Figure 3.15: Loss Scores

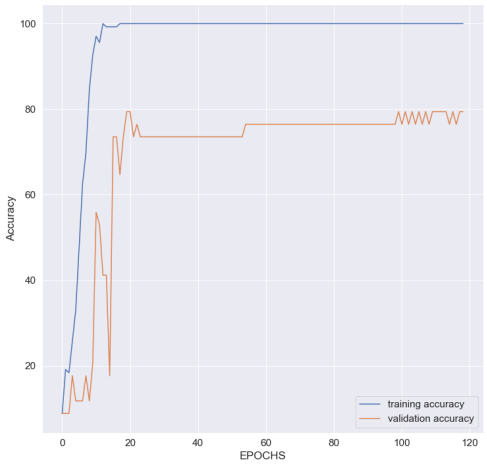

Figure 3.16: Accuracy Scores

As one can see, the model stabilizes the validation loss around 0.6 and has an validation accuracy of approx 80% while the training loss and accuracy is approximately 0 and 100% respectively. The model has a total of 206,439 trainable parameters. Certainly there is no overfitting in this model, one can clearly see the validation loss and training loss minimize and stabilize together. As we can see there are certain spikes in the both the plots above but these are due to the optimizer function trying to find the correct direction tot traverse in. As we can see that within a few epochs it improves itself.

3.5.2 Testing

The test set comprised of 43 samples i.e. about 20% of the total dataset.
In Figure: 3.17 we present the confusion matrix of the testing set. The test accuracy for our model is around 88%.

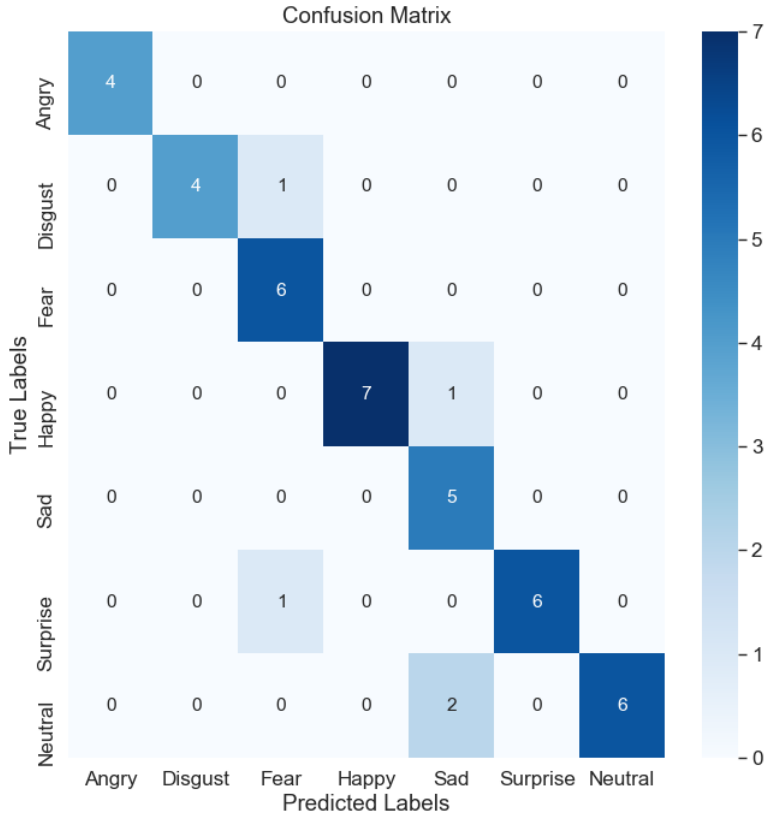

Figure 3.17: Confusion Matrix for JAFFE Test Set

4

Conclusion

1. The potential of CNN for image analysis is clearly observable, but it is also a tool which has to be very well understood in order to use it in the most effective way. Good knowledge of other image analysis techniques is practically a must, since that allows a more refined tuning of the network architecture.
2. Exploration of what the network is doing internally is necessary in order to understand how and why is it working correctly or incorrectly. In this sense, as we showed previously, one can visualize the obtained filters in each layer, as well as the outputs of different convolutional layers. the interpretation of this visualizations depends on experience and knowledge, and can help make informed decisions about the model.
3. By comparing the results obtained in the two selected data-sets, we can see a clear difference in complexity. There are many factors in this, like the size, the number of subjects, the artificiality of the data etc.
4. We consider that obtaining a 60% accuracy on such a complex data-set is quite good, at least as a starting approach, after which one could apply more advanced state of the art techniques, and also add some more prepossessing, or even more data. Considering the paper [1], we can see that the accuracy we obtained is pretty decent for a CNN model. The winner of the kaggle competition had an accuracy of 71.162% ([2]) but his model performance was far better. To know more one can read his paper [4] in which he describes how changing the softmax layer with L2-SVM provides these significant performance improvements.
5. Further investigation of why some classes cause more trouble than others in the complex data-set would be of value. We can see that the disgust class is greatly misclassified with only 24% accuracy. There is an overlap with some images of neutral, angry and disgust classes that could be investigated in further work.

Bibliography

[1] I. J. Goodfellow, D. Erhan, P. L. Carrier], A. Courville, M. Mirza, B. Hamner, W. Cukierski, Y. Tang, D. Thaler, D.-H. Lee, Y. Zhou, C. Ramaiah, F. Feng, R. Li, X. Wang, D. Athanasakis, J. Shawe-Taylor, M. Milakov, J. Park, R. Ionescu, M. Popescu, C. Grozea, J. Bergstra, J. Xie, L. Romaszko, B. Xu, Z. Chuang, and Y. Bengio. Challenges in representation learning: A report on three machine learning contests. *Neural Networks*, 64:59 – 63, 2015. Special Issue on "Deep Learning of Representations".

[2] Kaggle. Fer2013 kaggle challenge results. `https://www.kaggle.com/c/challenges-in-representation-learning-facial-expression-recognition-challenge/leaderboard`, 2013.

[3] M. Lyons, S. Akamatsu, M. Kamachi, and J. Gyoba. Coding facial expressions with gabor wavelets. In *Proceedings Third IEEE International Conference on Automatic Face and Gesture Recognition*, pages 200–205, 1998.

[4] Y. Tang. Deep learning using support vector machines. *CoRR*, abs/1306.0239, 2013.