Facial Emotion Recognition using CNN

Shrestha Malik : A53087000
COGS 260, Spring 2016
University of California, San Diego
shm039@eng.ucsd.edu

Shubham Saini : A53071422
COGS 260, Spring 2016
University of California, San Diego
ss1saini@eng.ucsd.edu

Abstract

In this project, we applied Convolution Neural Networks to predict emotions from the image of human faces. We used Kaggle Facial Emotion Recognition 2013 [11] dataset for training. The human accuracy on this dataset is 65%. We experimented with different models and hyper parameters to finally come up with a model that meets the human accuracy. Our best performing model also performs in the range of the top 4 models of the challenge. Further, we applied different visualisation techniques on this model.

1 Introduction

In the last decade the progress made by machine learning has been astounding, allowing it to grow in popularity both as a research field and as a basis for commercial applications. This work focuses on a specific domain of machine learning, namely artificial neural networks. We present one application of neural networks: Emotion Recognition from images of faces.

Defining a direct mapping between a high dimensional image and a class label is prevented by the little individual influence of each of the pixels on the label. In order to learn how to classify emotions, higher level facial features need to be extracted from the raw input images. These features can then be associated with emotions.

We present an approach based on Convolutional Neural Networks (CNN) for facial emotion recognition. The input into our system is an image; then, we use CNN to predict the facial expression label. We test our model on the Kaggle Facial Expression Recognition [11] dataset. We further try the model giving the highest accuracy on FER using our own collected data for live emotion recognition.

2 Motivation

Emotion recognition has multiple applications in entertainment. Gaming experiences can be enhanced if the action increases pace when the player is bored and slows down when the player is overwhelmed. A music phone application which shuffles songs according to the listeners face expression would provide a more personalized experience. In the context of social media, automatic emotion detection from images can be used for targeted advertisement or tagging of images, as well as status updates: the user can upload an image of themselves, and their status can change to reflect their emotional state.

A similar system can be used in a judicial setting to detect if a suspect is displaying genuine feelings or just pretending. The aim would be to exceed the human capability of detecting emotions, by spotting brief moments of weakness in which the individual is not maintaining its false, pretended pose.

3 Background

3.1 Neural Networks

Neural network can be seen as a graph, where the vertices are called nodes or neurons and edges are called synaptic connections. The synaptic connections have strengths (called weights) which adapt in the learning process. The networks are structured in layers, according to the connections between nodes. The layers can be grouped as follows: (1) The input layer storing the given data, (2) The hidden layers used to define a better representation of the data than the one given by the input, and (3) The output layer contains the output, after a pass through the network. Only required for nets used for discrimination.

3.2 Convolutional Neural Networks

Convolutional Neural Networks are very similar to ordinary Neural Networks. ConvNet architectures make the explicit assumption that the inputs are images, which allows us to encode certain properties into the architecture. These then make the forward function more efficient to implement and vastly reduce the amount of parameters in the network. These are a form of multi layer perceptrons (MLP) designed for minimal prepossessing, but do not have full connectivity between nodes thus avoiding the curse of dimensionality. The features that distinguish a CNN from an MLP are (1) 3D arrangement of neurons within a layer, (2) Local connectivity: CNNs exploit spatially local correlation by enforcing a local connectivity pattern between neurons of adjacent layers, and (3) Shared weights: In CNNs, each filter is replicated across the entire visual field.

A summary of types of layers and their features are as follows:

- **Convolutional Layer**: it is the core building block of a Convolutional Network, and its output volume can be interpreted as holding neurons arranged in a 3D volume. The CONV layer's parameters consist of a set of filters. Every filter is small spatially (along width and height), but extends through the full depth of the input volume.
- **Pooling Layer**: these are occasionally inserted between CONV laters. Its function is to progressively reduce the spatial size of the representation to reduce the amount of parameters and computation in the network, and hence to also control overfitting. The Pooling Layer operates independently on every depth slice of the input and resizes it spatially, using the MAX operation.
- **ReLU** Rectifier Linear Unit layer will apply an elementwise activation function, such as the max(0,x)max(0,x) thresholding at zero.
- **Fully-Connected** layer will compute the class scores, resulting in volume of size [1x1xn], where each of the n numbers correspond to a class score. As with ordinary Neural Networks and as the name implies, each neuron in this layer will be connected to all the numbers in the previous volume.

4 Dataset Used

The International Machine Learning Society as part of their 2013 workshop on Challenges in Representation Learning launched the Kaggle Competition for facial expression recognition[1]. The dataset was hand collected from Internet and screened by human.

The final data consisted of about 37,000 48 48 pixel gray-scale images of faces. The dataset is highly diverse in terms of gender, age and race. The images are processed in such a way that the faces are almost centered and each face occupies about the same amount of space in each image. The facial emotions have been categorized as: 0=Angry, 1=Disgust, 2=Fear, 3=Happy, 4=Sad, 5=Surprise, and 6=Neutral and some samples are given in the figure(1). As can be seen from the figure, all the classes are not uniformly distributed. Few classes are more dominant than others. There are about 29,000 training images, 4,000 validation images, and 4,000 images for testing.

The human accuracy on the dataset is 65 ± 5 %. The hosts also released the accuracy of a null model trained by them which was reported to be 60%.Top 4 submissions all used Convolution Neural Network.Few teams did not use feature learning and presented competitive models, but the performance

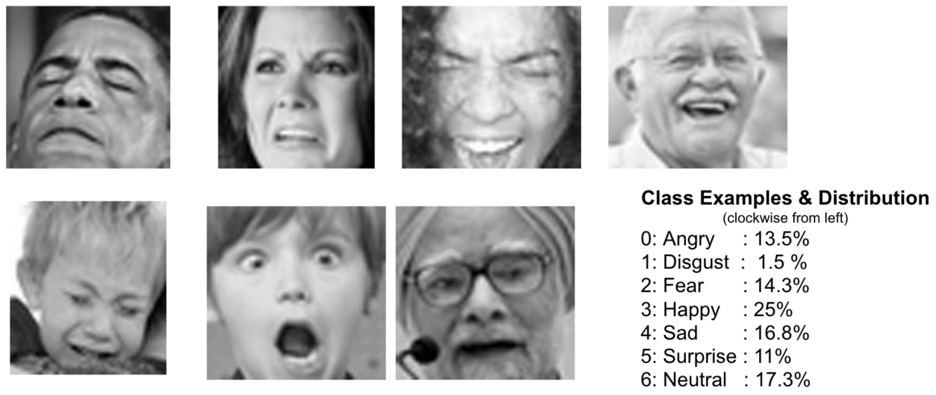


Figure 1: Sample Images for each class

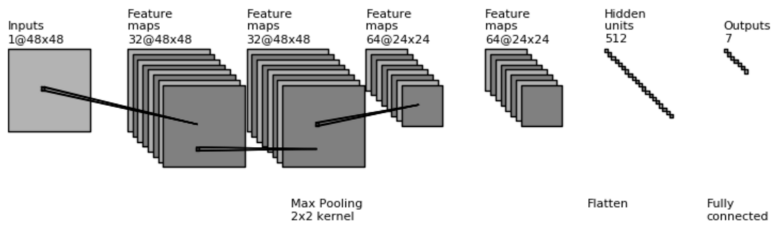


Figure 2: Model 0

was not as good as CNN models. The winning submission had an accuracy of 71.16% and it used SVM as the final classification layer, which was a novel approach. The accuracy Range of top 4 teams was between 67.48% - 71.16 %

5 Experimental Setup: Methods and Findings

The literature on facial emotion recognition, has shown that CNNs outperform most other feature based techniques. Even, the hybrid models that use convolution neural network along with features like HOG[12, 1] etc., do not perform better than the CNNs alone which shows that CNNS can learn all the features in the training set.

We present here 4 different models that we built and experimented with. For representation purposes, we define a block of neural network layers. Each block has 2 convolution layers activated via Relu(Rectified Linear Units), followed by a maxpool layer and a dropout layer. This is like the building block of our models. We add more blocks to build a deeper network. We used cross-entropy loss function and softmax for final classification. A very detailed architecture with all the hyperparameters have been mentioned in the Appendix.

As a pre-processing step we z-centered all the images with the mean of the training images. We also used data augmentation to introduce some translations in the training dataset. We used horizontal flipping.

First model, had 2 such blocks,one fully connected layer followed by a softmax layer, trained with mini-batch stochastic gradient descent(SGD). This model gave us an accuracy of 59.34%. This is referred as Model0, shown in figure(2), as it meets the null model accuracy suggested by the hosts.

We also used other training methods like RMSProp and Adagrad on this. We did not see any major difference in the accuracy achieved upon using these methods. Though, the training was faster in some cases it was not substantially faster and we decided to use SGD for the rest of the training

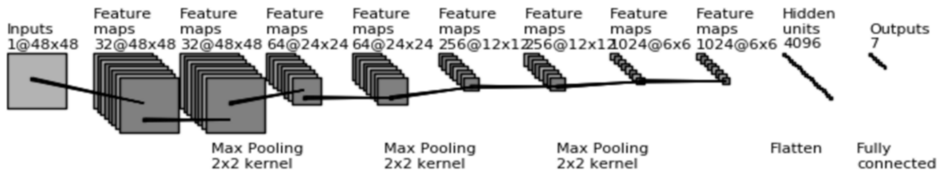


Figure 3: Model 1

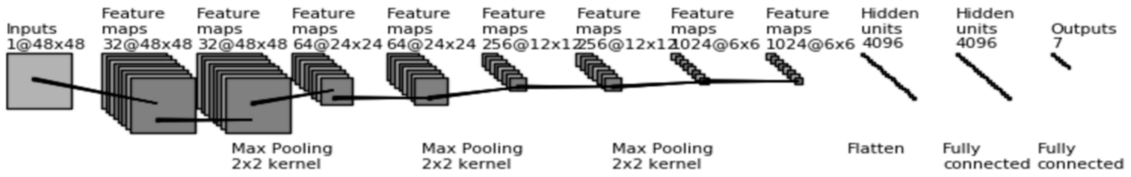


Figure 4: Model 2

too. We also experimented by adding batch normalisation and regularisation. This substantially increased the training time with not much increase in accuracy.

We now trained deeper models with more convolution layers. Our second model, referred as Model1, used four blocks of conv+maxpool as defined above, followed by a fully connected and softmax layer. We gradually increase the number of filters from one block to another. Each maxpool layer decreases the dimension of the input image by a scale of 4, so the number of filters should be choosen in such a way that maximum information from the previous block can be passed to the next. One way this can be done is by keeping the number of parameters to be same by increasing the number of filters.

Below, the figure(3) represents the second model that we built. This gave us a test accuracy of 63.7%.

Building on the above model, we add another fully connected layer before the softmax layer. So now there are 4 blocks and 2 fully connected layer followed by the softmax layer. The model,referred as Model 2, is shown in figure(4) and gave an accuracy of 65.4%. This is the model that does as well as the human in detecting emotions.

All these models were trained for over 100 epochs. We stopped training when we saw that the model had started to overfit.

We also explored the model suggested by Zhiding Yu and Cha Zhang in their paper[2]. The model, referred as Model3, we implemented is shown in the figure(5). Our implementation did not generalise as expected and gave an accuracy of 48.8%.

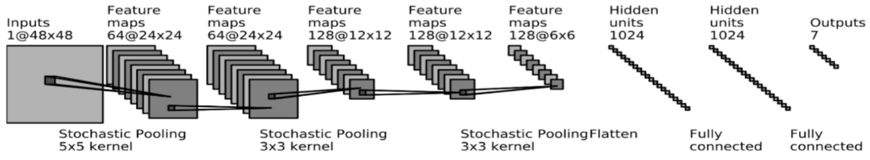
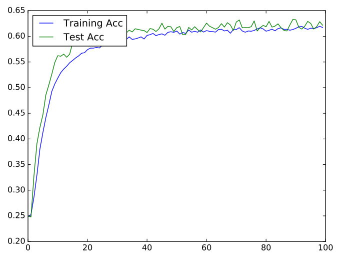
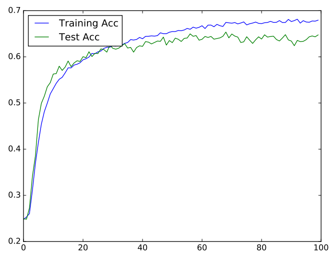


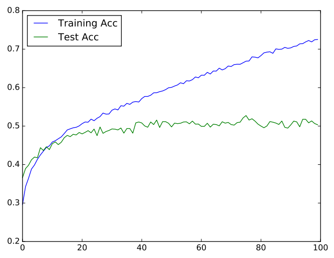
Figure 5: Model 3



(a) Model 1



(b) Model 2



(c) Model 3

Figure 6: Training and Test Accuracy

We also tried transfer learning for this project. We used the pre-trained weights of VGG net [10] and fine tuned the final layer of the VGG network. However, the accuracy was not good and the training time was huge.

We used Keras [8], a Theano [9] based CNN library for all our experiments.

6 Results and Visualisation

We trained 4 CNNs, as mentioned in the previous section. Fig 6 shows the training and test accuracy for all the trained networks, except the null model. Note that we show the accuracy for 100 epochs, after which over-fitting begins to kick in. Models 1 and 2, which are extensions of baseline Model 0, both gave higher accuracy than Model 3. Model 3 started overfitting after Epoch 20, when the training accuracy continued to increase with stagnant test accuracy. Table 1 summarizes the results of the models.

Model	Test Accuracy
0	59.34%
1	63.7%
2	65.4%
3	48.8%

For further investigations for our best performing model, we tried several network visualizations techniques. First, we tried t-SNE (pronounced tee-snee). t-Distributed Stochastic Neighbor Embedding (t-SNE) is a technique for dimensionality reduction that is particularly well suited for the visualization of high-dimensional datasets. It typically requires relatively low-dimensional data. Start by using an autoencoder to compress data into a low-dimensional space, then use t-SNE for mapping the compressed data to a 2D plane. The results of the t-SNE algorithm on our training set is given in Figure 7. Each point in the figure represents one image, and each color represents one class label.

As evident from the results, the images for each class are spread around and do not show very clear clustering pattern. However, each image is surrounded by images of the same class, forming micro clusters. For e.g. the aqua blue dots, do have more aqua dots around them as compared to other colours. So we see small clusters close to other clusters.

This was expected as the images are diverse in terms of the age, gender and race and emotion is local to something very characteristic of particular locations on face like eyes, eyebrows, mouth, teeth, etc. Moreover the human accuracy on the dataset is also around 65% which shows that its not very clear to categorise emotions.

We also tried to visualize the ConvNet filters. Specifically, we tried finding an input that maximizes a specific class. For this purpose, we maximize the activation of a specific output of the network, to get an image that looked like the class encoded by that output. Figure 8 presents an input that maximizes the probabilty of the 'Happy' class.

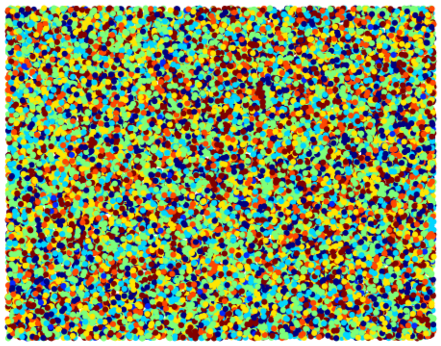


Figure 7: t-SNE output

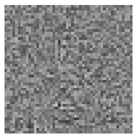


Figure 8: ConvNet Visualization

7 Future Scope

We read a lot of literature on this problem statement and learnt about many interesting techniques. The neural network suggested by Yichuan Tang[4], who won the Kaggle challenge, used L2 SVM loss for the final layer of classification. This approach has proven to do better than softmax for many applications. Also boosted deep belief networks as suggested in [3] looks like an interesting technique. The next step from recognising emotion will be generating facial expression given the emotion as suggested in the paper [5]

Acknowledgments

We would like to thank Prof. Zhuowen Tu for offering this very comprehensive course on Image Recognition using Neural Networks and guiding us throughout the lectures and project. We would also like to thank Apurva Pathak (TA) and Shuai Tang for helping us with ideas and suggestions whenever we got stuck, via Piazza and teaching hours. Finally, we would like to thank everyone who took the course with us for making the classes and assignments much more interesting with their participation.

References

[1] Ian J. Goodfellow1, Dumitru Erhan, et al *Challenges in Representation Learning: A report on three machine learning contests*.

[2] Zhiding Yu and Cha Zhang *Image based Static Facial Expression Recognition with Multiple Deep Network Learning,Microsoft Research.*

[3] Ping Liu, Shizhong Han and Zibo Meng Yan Tong *Facial Expression Recognition via a Boosted Deep Belief Network*.

[4] Yichuan Tang *Deep Learning using Linear Support Vector Machines*.

[5] Joshua M. Susskind1,2, Geoffrey E. Hinton2, Javier R.Movellan3 and Adam K. Anderson *Generating Facial Expressions with Deep Belief Nets* .

[6] Marian Stewart Bartlett, Gwen Littlewort, Mark Frank, Claudia Lainscsek, Ian Fasel, Javier Movellan *Recognizing Facial Expression: Machine Learning and Application to Spontaneous Behavior*

[7] Dan Duncan, Gautam Shine, Chris English *Facial Emotion Recognition in Real Time*

[8] Keras Documentation
`http://keras.io/`

[9] Theano: a CPU and GPU math expression compiler
`In Proceedings of the Python for Scientific Computing Conference (SciPy), June 2010.Oral Presentation.`

[10] VGG Keras and pre-trained weights
`https://gist.github.com/baraldilorenzo/07d7802847aaad0a35d3`

[11] Facial Emotion Recognition Dataset
`https://www.kaggle.com/c/challenges-in-representation-learning-facial-expression-r`

[12] Shima Alizadeh, Azar Fazel, Stanford University *Convolutional Neural Networks for Facial Expression Recognition*.

8 Appendix

Hyper parameters of models

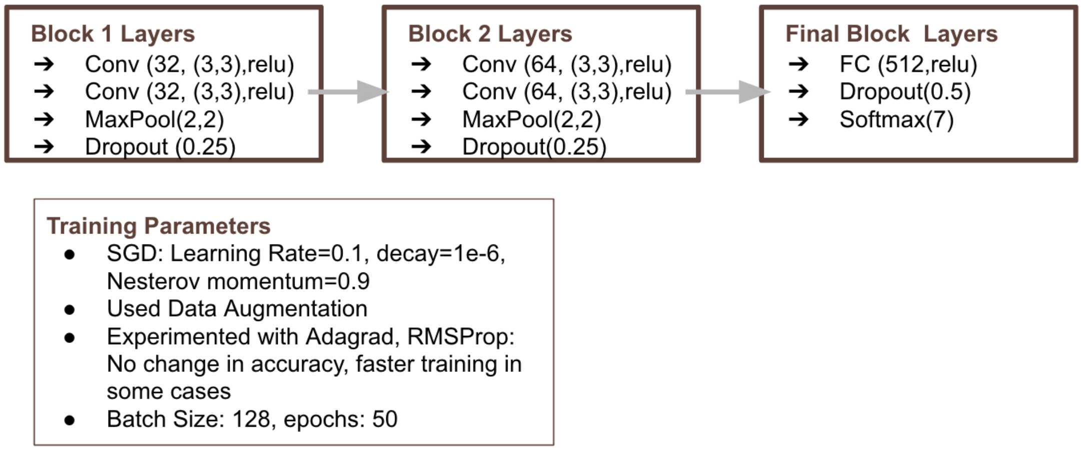


Figure 9: Model 0 Hyper parameters

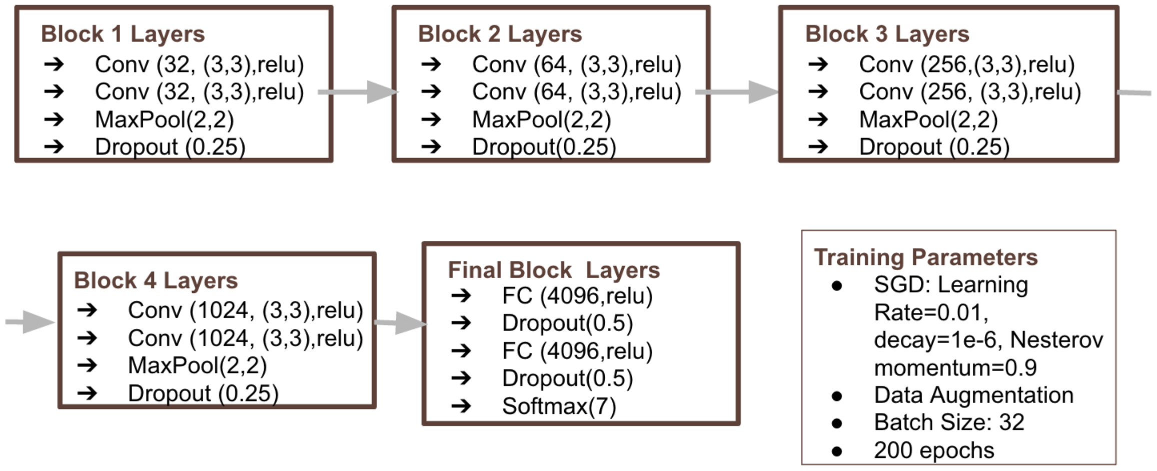


Figure 10: Model 1 Hyper parameters

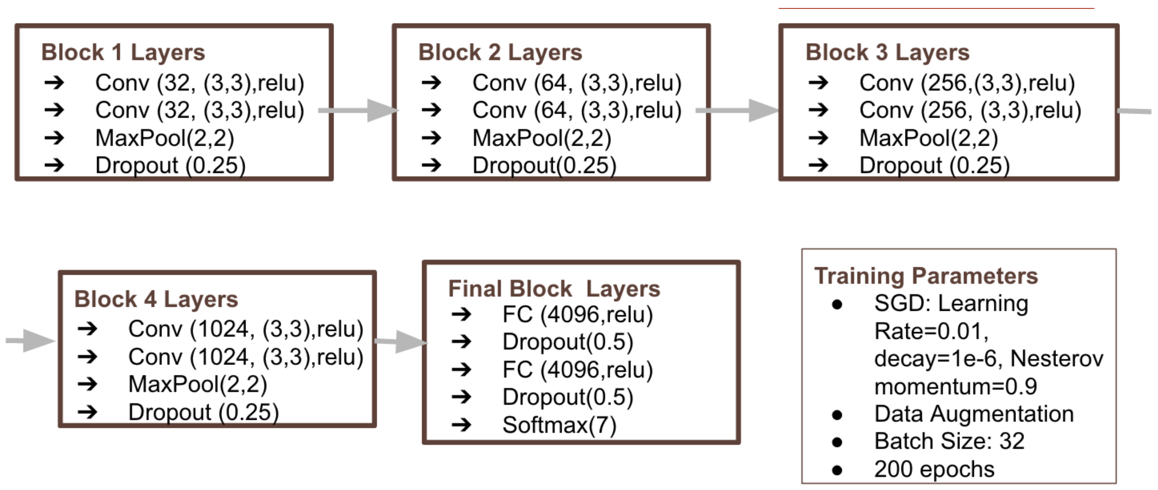


Figure 11: Model 2 Hyper parameters

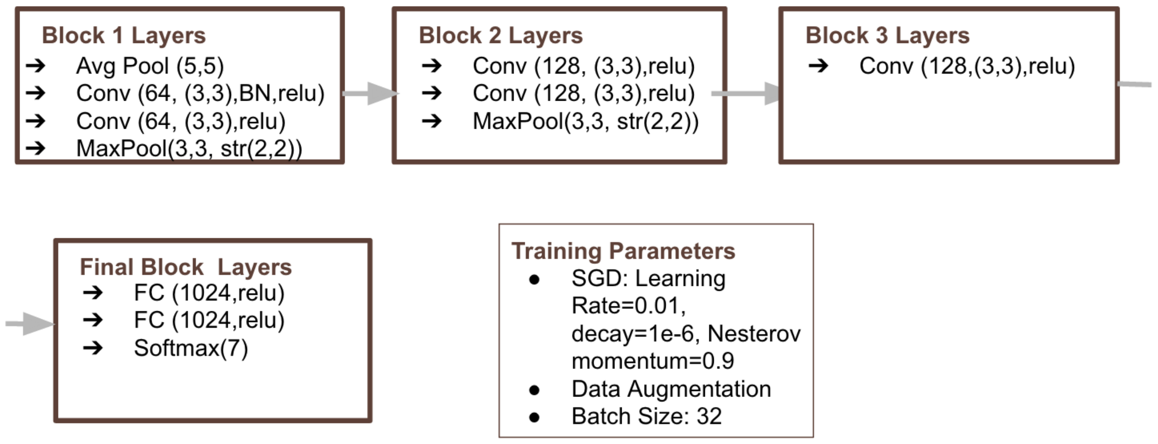


Figure 12: Model 3 Hyper parameters