Human Emotion Detection using Convolutional Neural Networks

Chetanraj Kadam
Department of Computer Science
Syracuse University

***Abstract*— In this project, I explored the problem of detection of human emotion by observing facial expressions from images. For this purpose I used Convolutional Neural Networks (CNNs) to extract image features and detect emotion from seven key human emotions: Angry, Disgust, Fear, Happy, Sad, Surprise, and Neutral. I used dataset from Kaggle (Facial Expression Recognition Challenge) [1] which is by far largest dataset for this task. My aim is to address this task with understanding of how different CNN architectures perform in expressions detection. For this I implemented Shallow CNN model and Deep CNN model. Also to leverage transfer learning with state-of-the-art CNN architectures, I used VGG-16 with pretrained model on ImageNet. The accuracy I got with best performing deep model is 65.23% which can be ranked in top 10% in leadership board for this challenge.**

I. INTRODUCTION

Understanding human emotion is one of the important areas of research with wide variety of applications. It is noted in many studies that more than 90% of communication of humans is through nonverbal channels [2]. Human emotion recognition is actually combination of multiple things like facial expressions, speed and tone of voice, body movements. In this project, the focus is only on facial expressions as it most important factor of all.

Recognizing facial movements proved to be important in number of applications ranging from smart human-machine interaction to augmented or virtual reality. Interaction with day-to-day devices around us can be made more friendly and fast if devices made capable of recognizing person's motive without explicitly telling it. This also has applications in target advertising on social media platforms to advertise products based on person's mood. One of the important applications can be seen in security and surveillance cameras which can have power to detect objectionable behaviors. New age interactive gaming is also big area application of emotion recognition.

Detection of human emotions is actually subjective and ambiguous task. The person may not be feeling the same thing as we detected. Also emotions are varied and have no fixed criteria and measurement. Still we can focus on key human emotion to get overall broad idea of person's emotion and which will still very useful. The main six key humans described by ranchers which are constant across all cultures [3] are Angry, Disgust, Fear, Happiness, Sadness and Surprise.

Many of the old facial expression recognition models were based on standard machine learning techniques such as Bayesian classifiers and Support Vector Machines. But the use of Convolutional Neural Networks in the recent past [4], proved to breakthrough in image classification tasks. Its success has extended to all sorts of image processing and computer vision problems including recognition of facial expressions from images. The main reason for choice of this topic is after so many studies and advanced architecture this problem is still far from excellent results.

II. PROBLEM & DATA DESCRIPTION

A. Problem Description

The problem I am addressing is about extracting expressions from images of human face and classifying that into one of the seven emotion categories. These emotions are enough for representing key state of minds in human and added 'Neutral' emotion for the purpose of classification in addition to six basic emotions [3]. The problem is not about just face detection or recognition; it is about detecting expressions on those faces. The task seems simple but actually a complex because it involves small nuances of structure and parts of faces. Model needs to focus on important parts of human face called Action Units (AU) [5] such as Eyes, Cheek and Lips. It needs to weight these action units by their nature in image and construct meaningful relation between them.

There are various stages involved in this task. First stage is detecting faces from given image. Then second stage is of feature extraction from faces from high level features to low level features. And at final stage classify those into appropriate category.

The problem I am addressing in this project is consists of three main objectives:

- First is to solve the problem of detection of human emotion from observing person's face. For this I am going to implement CNN models from scratch.
- Second is to gain deep understanding of CNN architectures and performance with varying architectures, layers and depth. I am going to implement two CNN models: Shallow model and Deep model. This will not only demonstrate their performance on this task but also underline importance of depth in layers for complex tasks.
- Third objective is to demonstrate transfer learning with state of the art architecture trained on massive datasets. For this I am using VGG16 pre-trained CNN architecture which is trained on ImageNet dataset. I want to benchmark performance of my models against pretrained models and research the reasons of difference between performances.

I am going to optimize implemented models with different techniques and also achieve deep understanding regarding neural networks work in classifying images.

*B. **Data Description***

The dataset I am using is from Kaggle (Facial Expression Recognition Challenge). The reason for choice of this dataset is that it is having images containing only faces unlike ImageNet. This dataset is as of now biggest labeled dataset available of human facial expressions. Other important aspect of this dataset is that image examples are uniformly distributed over age and gender. Also contains good number of examples from different number of races and ethnicities.

This dataset has following attributes:

- 35,887 example images
- 48x48 pixel grayscale format
- Labeled with seven different emotions. Sample images from the dataset are as follows:

Fig. 1. Sample images from dataset with 7 different emotion labels

After performing initial check on data I noted few points that may affect prediction accuracy of models. Some images are not positioned at center and in some images face is hidden by hands or hair. Many images got watermark on them and some images got glasses. These factors make this task more challenging as not all facial features can be captured equally.

III. APPROACH

To solve problem of human emotion detection from images I choose Convolutional Neural Networks (CNNs) because of it is proved to be most successful for tasks related to visual recognition than any other method. To implement CNN I carried out detailed study of function of each layers in it. Following is short summary of each layer I used in implemented models:

- Input Layer

This layer takes image input in form raw pixels data and its size depends on height, width and dimension of image. In our task width and height will be 48 each and dimension will be 1 as our images are grayscale images.

- Convolutional Layer

This layer convolves on input layer with filter size of F and computes output for neurons based on dot product between their weights and small region they are connected to its previous layer. There are other parameters it take such as Stride which determines by how much amount window will slide over image. Also padding determines how many layers of 0 to add on boundaries of image. In my experiment I'm using default stride which is of 1 pixel and padding I am using padding to make output same length as of input.

- ReLU Layer

This layer applies activation function to input received from previous layer by using thresholding to zeros using max (0, x). In each of the model I used ReLU layers.

- Pooling Layer

This layer makes representation of image smaller and manageable. Basically it performs down sampling on image along height and width. There are different pooling techniques. For my implemented models I am using MAX pooling.

- Batch Normalization:

I used batch normalization layer in architectures to make these network easier and faster to train. This layer forces each input to a unit of Gaussian output. In following model architectures layer is used after max pooling and fully connected layer.

- Fully Connected Layer

These layers contain neurons which are connected to each neuron in its previous layer. The main purpose FC layer is to classify. These layers aggregate all spatial features it received from above layers and computes class scores. These are generally two or three layers and last layer is giving output vector. For all CNN models I am using Softmax activation in last layer to generate class probabilities.

As mentioned in problem description section I am going to implement mainly 3 CNN architectures. These are as following:

A. *Shallow CNN Model*

I designed shallow model with just one convolutional layer and is used as baseline model. The architecture of this model can be seen in following figure:

Shallow CNN Model
Input Image (48 x 48 x 1)
Convolution (3 x 3), 32 - ReLU
MaxPooling (2 x 2)
Flatten
FC, 512 - ReLU
FC, 7 - Softmax
Output

Fig. 2. Architecture of Shallow CNN Model

This model did not use any regularization or batch normalization. The basic purpose of this model is to perform one layer convolution on input image and give best out possible to it. This model severs as base to highlight importance of deep layers. This model needed proper hyperparameter tuning to give somewhat good results. This is also because of the fact of shallowness of its architecture.

B. *Deep CNN Model*

To address the task of emotion detection by processing facial expression and to experiment effect of deep layers I designed this deep CNN model. The full design architecture of this model is as in Fig.3:

Total trainable parameters for this model are 4.4 million. I experimented with different number of convolutional layers and max pooling layers. I observed the fact that when convolutional and pooling layers used in pair they tend to perform better. So in this model I used 4 pairs of convolutional and pooling layers. In addition I also used batch normalization after each convolutional layer. For handling of classification I used 2 layers of fully connected neurons and 1 last layer to give output with Softmax activation.

Deep CNN Model
Input Image (48 x 48 x 1)
Convolution (3 x 3), 64 - ReLU - Batch Norm
MaxPooling (2 x 2)
Convolution (5 x 5), 128 - ReLU - Batch Norm
MaxPooling (2 x 2)
Convolution (3 x 3), 512 - ReLU - Batch Norm
MaxPooling (2 x 2)
Convolution (3 x 3), 512 - ReLU - Batch Norm
MaxPooling (2 x 2)
Flatten
FC, 256 – ReLU – Batch Norm
FC, 512 – ReLU – Batch Norm
FC, 7 - Softmax
Output

Fig. 3. Architecture of 8 Layer Deep CNN Model

The most important aspect of these deep models that performance not equally proportional to adding number of layers. It is definitely give you much better result than shallow but adding more layers on top of it not always generated great results.

C. *Transfer Learning: VGG-16 Model*

The main motivation behind using transfer learning is understand use of large state of architecture in our own task. I studied various pretrained architectures like LeNet, AlexNet, VGG, ResNet and Inception. The choice of VGG16 is because of the fact of its simple homogenous architecture. The architecture of this model is as following:

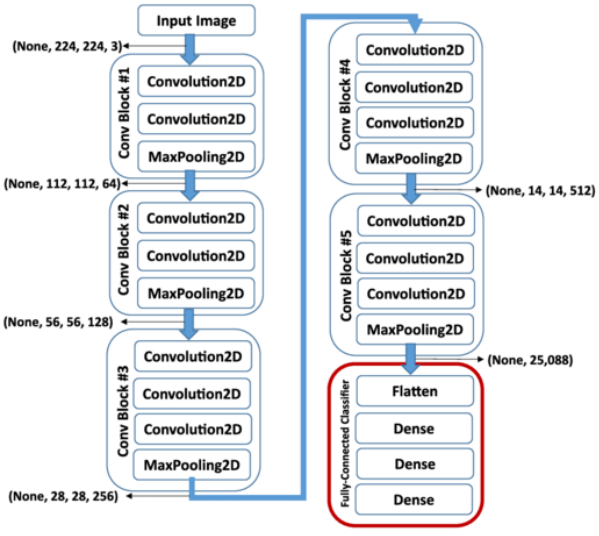

Fig. 4. VGG-16 Model Architecture [7]

The architecture is very similar to our models and hence it is easy to understand and compare its performance with other 2 models. It has 16 layers of convolutional and fully connected layers.

The approach I followed to demonstrate this is to use this model as a feature extractor and Fine-tuned.

1. **Feature Extractor:**
   To use it only to extract image features we only need its convolutional layers. So I first removed its fully connected layers as they were trained to classify images in 1000 class categories. In our task we need only 7 classes. Then I added 2 new fully connected layers with respectively 512 and 256 neurons. On top of that I added last FC layer to classify into 7 classes. Then I trained these 3 layers with our dataset. After this I used whole network to predict on our dataset.

2. **Fine-tuned:**
   This is process of adjusting pretrained weights on new dataset by continuing backpropagation. In this I first started to experiment with freezing top convolutional layers and train some below convolutional layers. This is due to the fact that top layers contain generic features from images like edges or colors but later layers become more specific. I did not have satisfying results with this strategy so then I trained whole network according to facial expression dataset.

In order to compare VGG-16 with other two I used exact same dataset with same preprocessing. I experimented with this model both as a feature extractor and fine-tuned. More about experiments discussed in results section.

IV. DATA

The dataset obtained from Kaggle is already divided in Training set (28,709 examples), Validation set (3, 589 examples) and Test set (3, 589 examples). Grayscale images represented in 48 x 48 grid of int values in range (0-255). So to visualize images I converted some of samples from pixel format to actual images.

To analyze dataset for distribution of emotion categories I calculated percentage distribution of these categories across dataset. The class distribution can be found in Fig.:

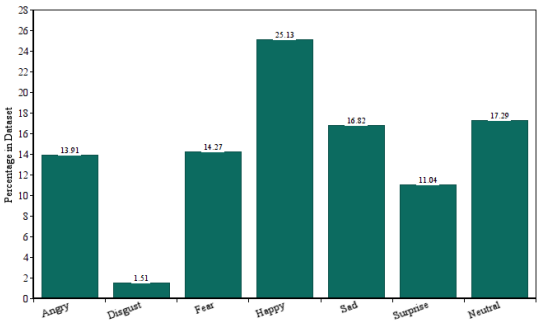

Fig. 5. Percentage distribution of emotion categories

From this figure, we can observe that 'Disgust' category is much underrepresented in the dataset. This played somewhat role in results so I later experimented with it in data augmentation stage. Dataset is not very much cleaned so I decided to perform data pre-processing before giving as input to models as following:

A. Mean Centering and Normalization:

Image data is in pixel format with non-negative values in range (0 - 255). I performed the process of standardization in which I first mean centered pixel data and then normalized it. This is to reduce some of extra features in images such as excessive lightning or brightness. It can be seen in following image transformation Fig.:

Fig. 6. Image transformation after mean centering

B. Data Augmentation

As dataset is relatively small and 'Disgust' category is underrepresented I performed data augmentation. This process I performed after implementing all models to improve their accuracy. I performed it by flipping images horizontally as seen in following Fig.:

Fig. 7. Flipped image of original image in Fig.6

I first performed data augmentation for 'Disgust' category but it did not improve my accuracy much. So I performed augmentation for whole training data which doubled my training size. This proved successful and accuracy of all of models increased by around 2%. This is because of the fact that expressions on face are irrespective of angles of photo taken and can be considered as new train example.

V. RESULTS

In this project to implement designed architectures and to train them on facial expression recognition dataset I used following hardware setup:

- NVidia (GeForce GTX 1080Ti) GPU: provided with CUDNN a GPU accelerated library for deep neural nets.
- 4 CPUs
- 16 GB Memory
- Ubuntu 16

To run experiments training time was varied between three of the models. Shallow network was comparatively very quick to train because of less number of layers and trainable parameters. On the other hand Deep model took time around 2-3 hours to run 50 epochs. And VGG16 model takes less time when I only trained fully connected layers whereas when fine-tuned whole network it takes around 3-4 hours. The fact that it is already trained is helped in comparatively lower time. Otherwise we know that these state of the art architectures take weeks to train on large datasets.

- ***Hyperparameter Tuning:***

The above described model architectures are the best performing model I got in those particular categories. To improve performance results of each of the models I did experiments with following hyperparameters:

- Number of Conv2D layers
- Number of FC layers
- Number of filters: [32, 64, 128, 512]
- Size of filters: [3 x 3, 5 x 5]
- Number of epochs: [20, 40, 50]
- Dropout: [0.10 – 0.30]
- Batch sizes: [50, 128]
- Learning Rate

I performed several incremental results to finally settle with satisfactory results for each models.

- ***Model Results:***

I have used Accuracy as evaluation metric for each of the model. I evaluated performance of each model with help of Training Accuracy, Training Loss and Validation accuracy while training on training dataset. Then when evaluated on test dataset I evaluated those on Test Accuracy and Loss. Following are comparison of 3 models on each of these metrics:

	Accuracy		
Model	**Training**	**Validation**	**Test**
Shallow CNN	87.95	52.24	52.02
Deep CNN	93.57	66.15	**65.23**
VGG16 Feature Extractor	77.48	48.04	49.51
VGG16 Fine-tuned	98.88	66.29	65

Table 1. Accuracy results of CNN model

For shallow model, I initially obtained accuracy below 50% which very low in terms of the task we are addressing. After carrying out hyperparameter tuning it improved little bit. Then I performed data augmentation and trained model with bigger dataset. Then accuracy of 52.02% achieved on test dataset.

Deep model I trained for 50 epochs initially with batch size of 128 which obtained me test accuracy of around 60%. Then using several tuning strategies especially adding dropout is proved successful is avoiding overfitting of model with training data. Because of this accuracy improved further and best model achieved accuracy of 65.23%. This quite satisfactory by looking number of classes model needs to classify.

For transfer learning with VGG16 I carried out number of experiments first as a feature extractor. With using model as it and not knowing anything about facial expression dataset it performed comparatively well. This is one of interesting fact of transfer learning even though model doesn't have any specific features for given dataset it is somewhat successful in classifying according to categories.

The results I got initially by fine tuning some of the lower layers are much lower than random guess. This also surprising as it was expected it will give better result than feature extractor. Accuracy achieved

initially with this experiment was around 24-25%. So when incrementing experiments I understood the fact that VGG16 like models will be used at its potential when whole network is fine-tuned with specific dataset. So after fine-tuning it achieved accuracy of 65% which is in par with our deep model.

Deep model achieved best accuracy amongst all three models we look its detailed performance while training and testing. Following is the plot of Deep model's training and validation accuracy over number of epoch while training:

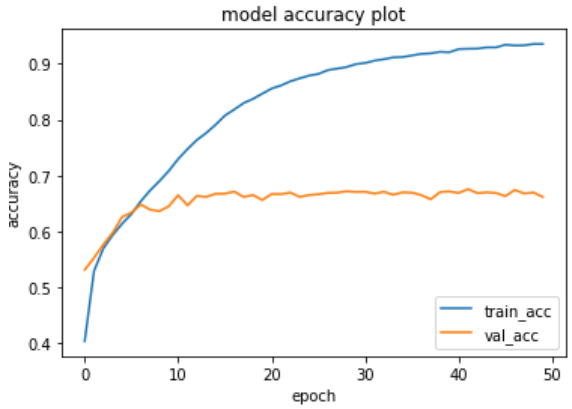

Fig. 8. Deep CNN model training accuracy plot

We can observe the fact that training accuracy is increasing with number of epochs but validation accuracy is stable after 10 epochs and having little variation in its performance.

To further analysis results of each model and make improvement in them I used confusion matrix as evaluation to look on which categories particular model is performing good and on which it is not. Following is confusion matrix of deep CNN model in fig. 9.

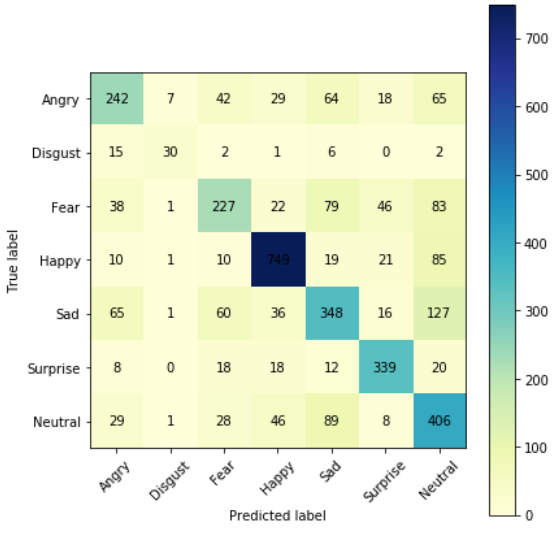

Fig. 9. Confusion matrix of Deep model

From confusion matrix we can observe that it is performed most accurately with predicting faces with 'Happy' category. The reason is in the fact that training dataset has most number of images of this category as seen in Fig. 5.

VI. DISCUSSION

The results obtained for each of the model can uncover some interesting facts in the task of learning from image details by CNN and performance variance with architectures.

So if we compare result of Shallow vs. Deep model we can state that having more number of layers and filters always lead to more accuracy. Shallow model was using just one convolutional layer as compare to four in Deep model. As convolutional layers increases the number of filters also increases. Convolutional layers stacked upon each other it works like particular set of filters assigned to local region but they capture different features. Earlier layers capture broad features such as dots, edges, lines whereas later layers focus on specific characteristics in images. So as in Shallow model has only one layer of convolution it cannot capture deep insights from image. For this purpose having number of convolutional layers helps which highlight importance deep layers in these kinds of task.

It is also interesting to compare result of VGG16 and Deep model as former has so many deep layers but little knowledge of dataset whereas later has limited number of deep layers but has full knowledge of dataset. VGG16 is basically trained ImageNet dataset which has all kinds of images and may contain some facial images but are not labeled as so. Still when it is fine-tune on FER dataset it is giving surprising result only by using weights trained from ImageNet and modified little with FER dataset. But if we look at 13 convolutional layers used by VGG16 and 4 convolutional layer used by Deep model then we can say Deep model gained enough details from dataset by using only 4 convolutional layers.

To analyzed performance on individual classes we can have look at confusion matrix result in Fig. 9. One interesting fact that can be observed is that although 'Disgust' category has very low number of images around 1.5%, still Deep model is not completely ignoring that category. It correctly classified more than half of its images and large numbers of other are classified as 'Angry'. This is because if we look at images from 'Angry' and 'Disgust' category, we can say it is difficult for humans also to classify them correctly as there is very little distinction between them. Similarly 'Fear' and 'Sad'

are when misclassified are mostly as 'Neutral'. The reason can be that both of these emotions contains little tension on forehead which can be if neglected by model then those images will be classified as 'Neutral' emotion.

If we see that emotion recognition by only observing person's face is quite ambiguous task. It is proved in some experiments human accuracy of classifying emotions from this dataset is around 60%. So if we compare our model's performance to that then the results are quite satisfactory. This underlines the facts that if images could be more clear and of higher resolution then model can perform much better than human.

The problem addressed in this project can have many meaningful implications for other tasks also. Accuracy achieved these kinds of task is not only measure of success of models. It is observed that achieving accuracy around 60% is simple but improving further from 60%-70% is complex task. For that we have to apply several design and experimenting strategies.

These results can be improved further with large dataset and high resolution images with little noise. In this task accuracy mostly affected by noises in images e.g watermarks on face. For future work on this project, I can use pretrained model which is trained on large dataset of similar kind such as VGG-Face which is trained on 2.6 million images of 2600 people. Although task addressed by VGG-Face is face detection, features captured by convolutional layers are of faces which are suitable for our task. Another future work can be combination of human sound and image to accurately detect emotion by using facial expression and tone/volume of person's speech.

VII. Conclusion

In this project I was able to successfully implement CNN models from scratch to detect human emotions from images. I was also able to use pretrained VGG-16 model for this task. The accuracy of 65.23% achieved by Deep model is best among all and can be ranked in top 10% in leadership board on this dataset. I was also able to compare performances of these models and draw interesting insights from that.

APPENDIX

After implementation submission I worked on two important aspects. First one is to visualization from code to include in this paper. This includes converting raw pixels to images for each category, output of data pre-processing, visualization of obtained results including tables, graphs, plots and confusion matrix.

Second thing I worked on is implementation of VGG-Face as described in future work. Unlike VGG-16 which is available in Keras [9] with pretrained weights, VGG-Face is not available in any such library. It needed to be downloaded from official site of Visual Geometry Group [8] from Oxford. After downloading I started implementing and training it but due to constraint of time and available resource setup I could not yet finish yet and so not included in the results.

References

[1] https://www.kaggle.com/c/challenges-in-representation-learning-facial-expression-recognition-challenge

[2] Albert Mehrabian. Silent Messages, University of California Los Angeles, 1971.

[3] P. Ekman and W. V. Friesen. Emotional facial action coding system. Unpublished manuscript, University of California at San Francisco, 1983.

[4] Krizhevsky, Alex, Ilya Sutskever, and Geoffrey E.Hinton."Imagenetclassificationwithdeepconvolutional neural networks." Advances in neural information processing systems. 2012.

[5] P.Ekman,W.Friesen,FacialActionCodingSystem:A Technique for the Measurement of Facial Movement, Consulting Psychologists Press, 1978

[6] Very Deep Convolutional Networks for Large- Scale Image Recognition, Karen Simonyan, Andrew Zisserman , 2015.

[7] Deep Convolutional Neural Networks with transfer learning for computer vision-based data-driven pavement distress detection, 2017.

[8] Visual Geometry Group, University of Oxford. http://www.robots.ox.ac.uk/~vgg/research/very_deep/

[9] Keras: The Python Deep Learning library, https://keras.io/