Recognizing Facial Expression Using Deep Learning

Akib Ahmed Fahad [1] and Lamia Akter[1] and Anuska Prosad Paul[1] and
Maria Khondoker[1] and Md. Rakibul Haque[2]

***Abstract*— In this project we have implemented various deep learning methods such as Convolutional Neural Networks to identify six main human emotions that are anger, disgust, fear, happiness, sadness, surprise and lack thereof meaning neutrality. We have used two datasets, one is Kaggle's Facial Expression Recognition(FER) and another is Karolinska Directed Emotional Faces(KDEF) dataset. The architecture we have applied for our Convolutional Neural Networks were VGG-16 and ResNet50. The accuracy we have got using FER dataset was 64.3% and the accuracy we have got from KDEF dataset was 81.1% using VGG-16, 67.8% using ResNet50 architecture.**

I. INTRODUCTION

Understanding human emotions is a major area of research nowadays. Predicting human emotion has been a difficult problem which many people are trying to solve over a decade. As this area of work has lots of real life applications like friendly Human-Computer Interaction, better targeted advertising campaigns, systematic recruiting, improved recommender systems, so it can open a plenty of opportunities for us. We focus on visual recognition of facial expressions to identify the key seven human emotions, those are anger, disgust, fear, happiness, sadness, surprise, and neutrality. We have used deep learning to identify those seven primary human emotions. For achieving our goals, we have used two models of CNN architecture those are VGG-16 and ResNet50. The two datasets we have used is Kaggle's Facial Expression Recognition (FER) and Karolinska Directed Emotional Faces (KDEF) datasets. We have counted the accuracy of the training, validation, and test sets to evaluate our models' performance.

II. LITERATURE REVIEW

Savoiu & Wong [1] has used deep learning method was used for identifying the primary seven emotions of human, they are anger, disgust, fear, happiness, sadness, surprise and neutrality. They have used KDEF and FER datasets. They have used SVM model as their baseline. They also have implemented VGG-16 and ResNet50 to their convolutional neural networks separately. They used ensemble learning to combine both models. They also used transfer learning on KDEF dataset from training the models on FER. By using Support Vector Machine classifier as baseline, they got 31.8 percent accuracy. After that, they leveraged ensemble and transfer learning techniques to their datasets and then they got 67.2 percent accuracy and with transfer learning was 78.3 percent accuracy.

*This project is a part of Pattern Recognition Laboratory
[1]Student of United International University
[2] Faculty member of United International University

III. PRELIMINARIES

In this section we provide a brief description of the technical aspects of our project.

1) Convolutional Neural Networks (CNN): CNN is a special kind of neural network which is popular in image classification tasks among a variety of its other applications. An image can be represented as 2D matrices of a certain size (spatial dimension) which may be one or more in number stacked over each other (depth dimension). For example and RGB image of height and width 256 pixels respectively has a spatial dimension of 256×256 and its depth dimension is 3 (number of channels). When it is passed in the CNN as input, the convolution operation extracts nonlinear features by learning fixed portions of the image until it covers the entire image. This is done by computes the element wise multiplication between a fixed sized filter and equal sized portions of the input image obtained from each horizontal or vertical shifting of the portions (strides). The output is a feature map containing the extracted feature. The non linearity of the feature map is ensured by activation functions such as Rectified Linear Unit (ReLU). The feature map can also undergo a pooling operation where the most important information of it can be preserved while reducing its dimensionality subsequently. Finally a fully connected layer can carry out the final classification step while taking input of the final convolutional layer's feature map. CNNs are usually trained by backpropagation algorithm.

2) VGG-16: VGG-16 is a CNN model that consists of 16 layers (13 convolutional and 3 fully connected). The convolutional layers combined with max pooling layers are orgnized in 5 stacks as shown in Figure 1. The filters size of each convolutional layers is 3×3 and with a stride of 1 pixel. Similarly, the pooling window is of size 2×2 and the stride is set as 2 pixels. To retain the original spatial dimensions of the feature map padding is done appropriately. The number of filters in the initial stack is 64 and doubles in every subsequently layers until it reaches 512. Following the convolutional layers, there are 3 fully connected layers with 4096, 4996, and 1000 channels respectively. The last layer is the output layer with the soft-max activation function. The rest of the hidden layers have ReLU activation for non-linearity. With about 140 million parameters, VGG16 is a very expensive model to implement as it consumes a lot of time and memory.

3) ResNet: ResNet50 is another CNN model that consists of 50 layers (49 convolutional and 1 fully connected). The 48 convolution layers between the first convolution layer and

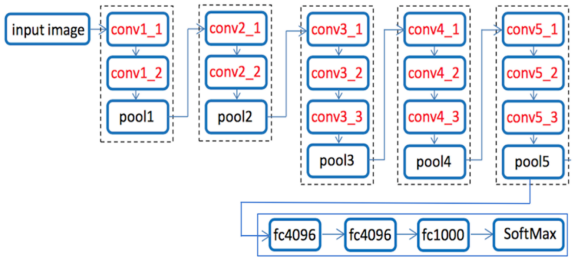

Fig. 1. VGG-16 Architecture Diagram

the last fully connected layers are divided in to 4 stages as shown in Figure 5. The initial convolution layer has 64 filters of size 7×7 and a max pooling layer of size 3×3, with a stride of 2 for both. In the following stages groups of 3 convolution layers are arranged with a skip connection between them. The sequence of convolutions are 1×1 for reducing dimensions of the input feature map, 3×3 for extracting non-linear features, and 1×1 for restoring dimensions of the output feature map. The number of filters are doubled in each consecutive stages starting from 64 upto 512 for 3×3 convolution, and from 256 to 2048 for 1×1 convolution. The purpose of the skip connections is identity mapping that enables the model to bypass a stack of three convolutional layer sequence. This helps to decrease the training time and avoiding overfitting. After the last stage an average pooling layer is used followed by a fully connected layer of 1000 neurons.

IV. DATASETS

We are using two datasets, one is Kaggle's Facial Expression Recognition (FER) and the other is Karolinska Directed Emotional Faces (KDEF) datasets. We wanted to choose those databases that not only provide a representative number of images, but also that contain data which is rather uniformly distributed across the race, ethnicity, and gender of the subjects, and with a relatively even distribution across the emotions of these subjects. Both the datasets are compatible with each other since they both have the same target values, i.e. the seven emotion values: afraid/fear, angry, disgusted, happy, neutral, sad, and surprised.
FER dataset contains a total of 35,887 images. The image format is 48 x 48 pixels (8-bit grayscale). The dataset is a CSV file with numeric representation of emotions and pixels value of images. It's a imbalanced dataset.

KDEF dataset contains a total of 4900 images where the image format is 562 x 762 (32-bit RGB), 72 x 72 dpi. It's an image folder of 70 individuals displaying 7 different expression from 5 different angles.. KDEF is a balanced dataset. Therefore we used both with a combined total of 40,787.

V. METHODS

In this section, We present the methodology of our system. As we have worked with 2 different datasets, so firstly we demonstrate the methodology for FER dataset. Next, we show the methodology we have used in the KDEF dataset.

A. Methodology for FER 2013 Dataset

Due to the imbalance of the FER 2013 dataset, we have implemented image augmentation and created a roughly balanced dataset. Using this new dataset, we created a training and testing set which we implemented on an architecture consisting of a convolutional and fully connected layers. In the following subsections we describe the methodology with necessary details.

1) Image Augmentation: Before applying image augmentation techniques, the conversion of the images into a 2D NumPy array was also done using the help of the nltk module in Python. Image augmentation was done using the Keras ImageDataGenerator module. Using different parameters, ten to be exact (namely rotation, height shift, width shift, horizontal flip, vertical flip, shearing, zooming, brightness changing, rescaling, and channel shifting), we transformed all the images (batch size = 547) with the disgust labels (which was in a very small proportion) ten times. The flow method was used to create an image Numpy Array Iterator which then created the augmented images for each sample in the batch upon calling the next() method. The result was an increase in the disgust class data to an amount that provides a roughly balanced class. Using a balanced dataset makes it useful to use accuracy as a suitable metric for comparison amongst providing other benefits. Also, it makes the model less biased with the other classes which allows it to classify more accurately.

Initially, we had only 547 images under the disgust label. Applying image augmentation 10 times resulted in further 5470 images. In total, now there is 6017 images of disgust class. Meanwhile, there is an addition of 5470 images which increases the total sample size to 41,357. The resulting dataset is described in Table I.

Class (Label)	Number of Images	Proportion
Angry (0)	4953	12.0%
Disgust (1)	6017	14.5%
Fear (2)	5121	12.4%
Happy (3)	8989	21.7%
Sad (4)	6077	14.7%
Surprise (5)	4002	9.67%
Neutral (6)	6198	15%

TABLE I
DATA DISTRIBUTION AFTER IMAGE AUGMENTATION OF THE FER 2013 DATASET

The data was then split into training, validation, and testing sets comprising of 60%, 20%, 20% of the total samples. One hot encoder was used to deal with the categorical values of the output classes.

B. Model Architecture

The spatial dimension of the input images were 48×48 with depth dimension of 1 since they were grayscale with one channel. The convolution layers were distributed among

5 blocks with 2, 3, 4, 4, and 4 convolution layers in each respective blocks. The number of filters of the convolution layers in each blocks were 64, 128, 256, 256, 512 sequentially, where each filter were of size 3×3. Each convolution layer was followed by batch normalization to standardize inputs for the next layer. At the end of each block there was a max pooling layer with a 2×2 window each. The droupout rate was set as 0.3. The final block was followed by an output layer of 7 neurons representing the 7 output classes. The hidden layers used ReLU activation function and the output layer used softmax function. The loss function used was categorical cross entropy with the optimizer Adam.

C. Methodology for KDEF Dataset

Since the KDEF dataset were perfectly balanced, image augmentation was unnecessary. For face detection in the images, we used Haar Cascade classifiers from OpenCV library that contains a huge collection of computer vision algorithms. We used the Haar Cascade Frontal Face detection algorithm on a subset of the KDEF dataset consists of only the frontal face images. The facial part of the images were cropped and resized to 300×300 dimensions. The spacial dimension of 3 was retained even after grayscaling. The data was split into training and test sets consisting of 80% and 20% of the samples respectively. Two similar sequential models were used where the images were taken as inputs. The only difference between the models was that one used the VGG-16 architecture and the other used ResNet50 architecture using Keras Applications. In both the models the top fully connected layers were excluded and transfer learning was employed using pre-trained weights from ImageNet. After the VGG-16 or ResNet50 layers 4 fully connected layers were used with 512, 256, 128, 64 filters sequentially. The dropout rate was set as 0.3 after the first fully connected layer and batch normalization was used adter the third fully connected layer. The activation function for these dense layers was ReLU. The output softmax layer had 7 nodes for the 7 output classes. The loss function used was categorical cross entropy and the optimizer used was Adam.

VI. RESULT ANALYSIS

In this section we present the accuracy we have got from FER and KDEF datasets. To assess the performance of our models, we used Accuracy which measures the proportion of true results amongst the evaluated set.

A. Results and analysis for FER Dataset

As FER is an imbalance dataset, achieving good prediction was difficult for us. We have pre-processed our dataset and managed to gain 64.3% accurate prediction.

The graph in 2 demonstrates the graphical representation of loss and accuracy on training and validation set. It shows that the model has overfitted and the validation loss exceeded the training loss after around 35 epochs.

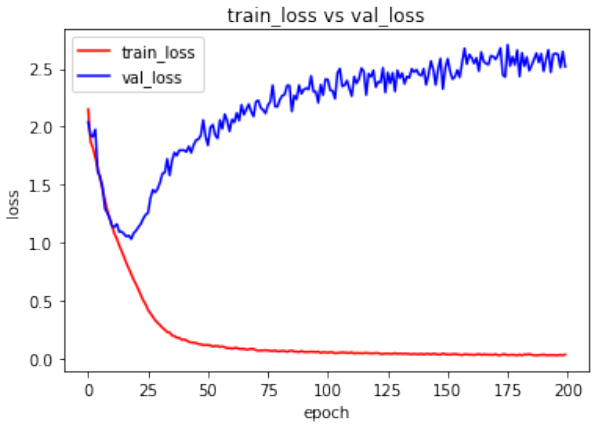

Fig. 2. Gained Loss on Training Data VS Validation Data

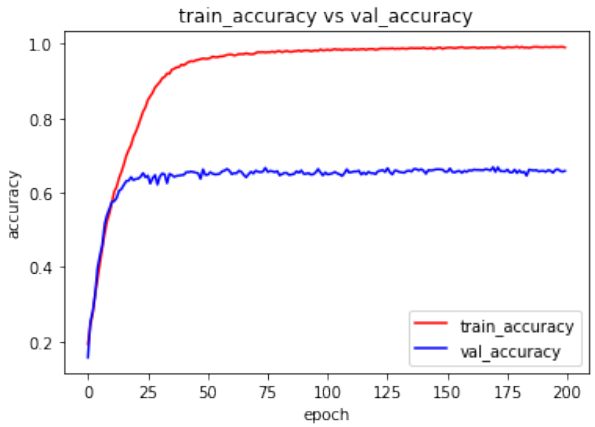

Fig. 3. Gained Accuracy on Training Data VS Validation Data

From this graph 4 we can see how much accurately our model was predicting different facial expression.

B. Results and analysis for KDEF Dataset

As we implemented two different Convulational Neural Network models, we got very different accuracy. We got more accurate prediction using VGG-16 architecture, the accuracy is 81.1%. ResNet50 achived 67.8% accuracy in our dataset. The discrepancy may have aroused due to the ResNet50 implementing skip connections.

Model	Accuracy
VGG-16	0.811
ResNet50	0.678

TABLE II
ACCURACY GAINED ON KDEF DATASET USING VGG-16, RESNET50

Our objective for this project was to achieve and accuracy over 60% in both the datasets which we have managed to achieve successfully. However, in case of the KDEF datasets further work needs to be done to incorporate all the other angles so that the entire dataset can be used.

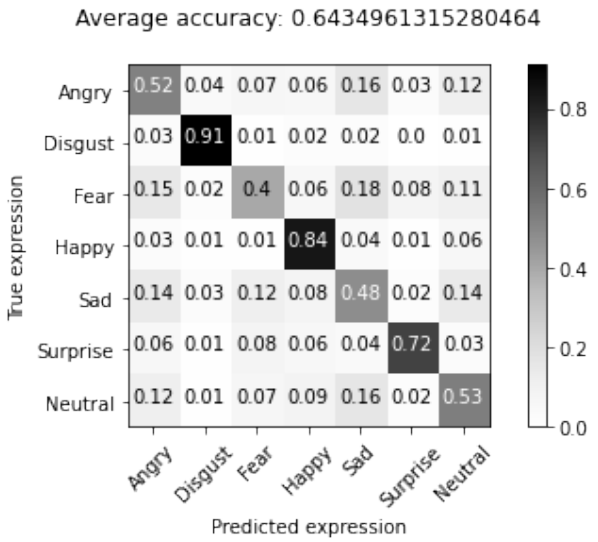

Fig. 4. Predicted Expression on FER Dataset

Future Work

In this section, we present the aspiration of expanding our project in future.

To further improvement of this work, we want to assess subject's micro-expressions meaning the expressions those last last less than half a second. Recognizing micro-expressions like confidence, credibility, reliability can be really helpful. As now we have worked with gray-scale images, next we want to recognize emotions from color images. Lastly, recognizing emotions from videos is another expansion we are hoping for our current project.

VII. Conclusions

For this project, we get to explore the VGG-16 and ResNet50 architecture for recognizing facial expression. Our initial objective was to achive atleast 60% of accuracy on both datasets. After implementing various methods and models such as image augmentation, VGG-16, ResNet50 architecture, we have reached our goal with better result. On FER we got 64.3% and on KDEF we got 67.8% accuracy using ResNet50, 81.1% using VGG-16.

Acknowledgment

We would like to thank Md Rakibul Haque Sir (faculty member, Department of CSE, United International University) for his mentorship, guidance, and support throughout this research project.

References

[1] Alexandru Savoiu and James Wong. Recognizing facial expressions using deep learning, 2017.

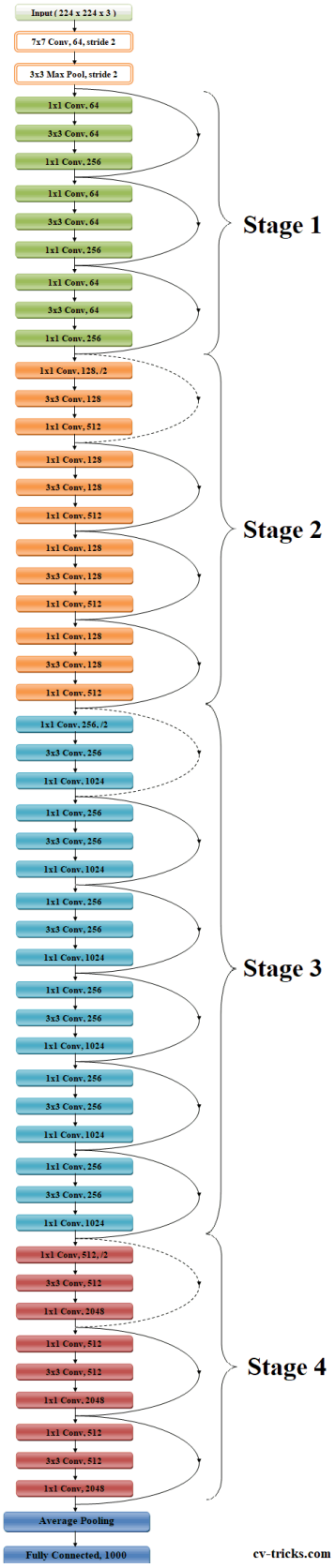

Fig. 5. ResNet50 Architecture Diagram