Facial Expression Classification using Convolutional Neural Networks

Utsav Murarka 2017B5A70854G

Piyush Agarwal 2017B4A80812G

Anany Shrey Jain 2017A4TS0920G

Rishabh Anand 2017A7PS0111G

***Abstract*—In this work, we try to create CNN based models for classifying facial expression images into one of the 7 emotions viz. Angry, Disgust, Fear, Happy, Sad, Surprise, Neutral. We also compare the various CNN architectures based on their performance on various evaluation metrics.**

Keywords—Image Classification, Convolutional Neural Networks, VGG-16, AlexNet, Expression Recognition

I. Introduction

For this work, we have been given a dataset consisting of 34,877 instances of 48x48 pixel grayscale images of different human faces along with 7 labels to categorise the emotion in each image. In this work, we will train a CNN to identify the emotions portrayed by various facial expressions and classify them into one of the seven categories. The usage of CNN in this work is motivated by the fact that they can exploit the spatial information in the images and learn to extract/capture the relevant features for making the correct classification. These features can then be passed on to the fully connected layers for final processing and classification. We will be first trying out VGG-16, AlexNet and then we will create our own model inspired from these architectures, tuned to achieve the best results. Data augmentation will be an important preprocessing step, as it will provide more varied data to the CNN and help prevent overfitting. Apart from Data augmentation, various other techniques like batch normalization and regularization will be used to acknowledge the problem of overfitting.

II. Our solution

A. Data processing

Given dataset is a 34,877 rows 2 columns CSV file, wherein first column represents the emotion i.e. Y and 2nd column is a string consisting of 2304 pixels. Firstly, we transform this string of 2304 pixels into a matrix of 48*48 for each data. We divided the dataset into training and validation in a ratio of 70:30. While splitting the data we ensure that distribution of data remains conserved in both the datasets for better performance.

Figure 1: Representation of various classes in data

Using a batch size of 64, we applied data augmentation with the following parameters:

- Rotated images between 0 to 10 degrees
- Zoomed images between 0 to 20%
- Horizontal flipping of images
- Shifting width between 0 to 10%
- Shifting height between 0 to 10%
- Adjusting brightness level from 70% to 130%

This helps in further diversifying our training set.

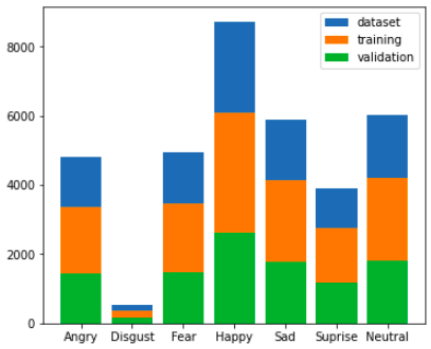

Figure 2: Stratification of the data

B. Description of model

Following parameters were kept for all the three models while training:

- **Initial learning rate** = *5e-4*, to ensure convergent behavior in loss function. We applied ReduceLROnPlateau which reduces learning rate by a factor of *0.4* if the validation accuracy does not increase for more than *2* epochs. This reduction in accuracy is repeated until a learning rate of *5e-6* is reached.
- **Epochs** = *35*, an epoch is considered complete when the whole training dataset is processed by the model. Multiple epochs may be required for completely training the model and achieving desired performance.
- **Optimizer:** Adam optimizers with parameters beta1 = *0.9*, beta2 = *0.999*, epsilon = *1e-07*
- **Loss function:** Sparse Categorical Cross Entropy Loss, because we desire classification into 1 of the 7 emotions
- **Activation Functions:** Swish activation is used for all the layers except the last, in the last layer we use SoftMax activation to favour our loss function. Swish activation is a special type of activation that utilises both the ReLU and sigmoid functions.
- **Regularizers:** It is a method to reduce overfitting in our network, we used L2 and dropout

VALUES OF HYPER-PARAMETERS ARE MENTIONED IN *ITALICS*

regularizers within our dense layers. L2 regularization (also known as Ridge Regression) adds a squared penalty term to the loss function, while dropout randomly prevents a selected amount of cells to be passed as input to the consecutive layers.

- **Padding:** Adding a boundary of zero pixel values to the image to prevent downsampling and loss of information from corners. Since we are working with deep convolutional neural networks, it can lead to huge reductions in the size of images. To prevent this phenomenon, the images are padded in a manner to preserve their dimensions.

Sequential Layers:

Following layers are used in our models:

- **Conv2D layer:** It is a convolutional 2D layer which takes in the following parameters: *input_size:*size of the input image(48,48,1)*, filters:* no. of filters in a layer*, kernel_size:* size of filters to be convoluted*, activation:* activation function to be applied*, strides:* pixels shift in applying filter
- **Batch Normalization Layer:** It is a layer added to normalize the inputs to a layer inorder to stabilize the learning process and achieve faster convergence thus reducing the number of epochs.
- **Pooling Layer:** It is a layer added to downsample the input so as to get only the important features passed to the next layer. Most commonly used poolings are average pooling and max pooling which take the average or max respectively of the pixel values in given pooling size. Since in our image we are focused on the emotions of the person which is the foreground of the image i.e. lighter parts of the image(high pixel value) it is favourable to use MaxPooling as our choice for the pooling layer.
- **Dense Layers:** These layers are added towards the end of the model, to finally learn from the features identified by the convolutional layers.

Models:

- **AlexNet[1]:** This architecture contains in total 8 layers, 5 Conv2D followed by 3 fully connected dense layers:
    - 5 Conv2D layers: with no. of filters *64, 128, 256, 512, 256* and using *swish* activation with each. Following each layer we have a Batch Normalization layer and the first four also have a max pooling layer too.
    - 3 Dense Layers: having output size 512,512, 7 respectively in each layer. First two layers use swish activation, L2 regularization followed by dropout. Last layer uses softmax activation.
- **VGG-16[2]:** This architecture has 16 layers in total, out of which, 13 are convolutional layers and 3 are fully connected dense layers. Some convolutional layers are followed by MaxPooling layers, and all hidden layers use ReLU activation:
    - 13 Conv2D layers: with no. of filters *32, 32, 32, 32, 64, 64, 64, 128, 128, 128, 256, 256, 256* and using *ReLU* activation with each. The 2nd, 4th, 7th, 10th and 13th layers are followed by Batch Normalization and MaxPooling with strides of 1x1.
    - 3 Dense Layers: with size 512,512, 7 respectively. First two layers use ReLU activation. Last layer uses softmax activation.
- **Our Custom Model:** This architecture contains 1,765,479 parameters out of which 1,764,519 are trainable and 960 are non-trainable. This models consists of the following layers:
    - 8 Conv2D layers with 32, 32, 64, 64, 128, 128, 256, 256 kernels of fixed size 3x3 fixed stride 1x1 and padding to preserve the dimensions. Every 2 Conv2D layers are followed by Batch Normalization Layer and MaxPooling Layer of size 2x2 .
    - 2 Dense layers with 256, 7 nodes containing L2 Regularization with weight decay of 0.005 and preceded by Dropout layers containing drop rate of 0.2.

C. EVALUATION

- **Accuracy:** This is the simplest metric for evaluating a classification model. It is the percentage of instances correctly classified out of the total number of instances in the test set.
- **Precision:** Precision is a measure of the number of instances of a particular class correctly classified as a percentage of the total number of instances classified to be in that class.
- **Recall:** Recall is the number of correct results divided by the number of results that should have been returned.
- **F1 Score:** It is the harmonic mean of Precision and Recall.
- **Confusion Matrix:** It is a 7x7 matrix where the (i,j) entry depicts the number of instances classified into class-i that were actually in class-j. Ideally all non-diagonal terms of the confusion matrix should be 0.

III. RESULTS AND ANALYSIS OF THE SOLUTION

As discussed in Section - II, we tried out 3 different architectures viz. VGG-16, AlexNet and a custom model. In order to determine the effectiveness of these models in solving real life problems, we need to compare them based on various metrics as discussed in section II Part C.
Figure 3 shows the loss curves and accuracy curves of all the three models for comparison.

	Accuracy	Precision†	Recall†	F1†
VGG-16	58.1%	0.57	0.58	0.55
AlexNet	64.1%	0.66	0.66	0.66
Custom	67.2%	0.67	0.67	0.67

† Weighted Average

Table 1 : Comparison of models based on Accuracy, Precision, Recall, F1

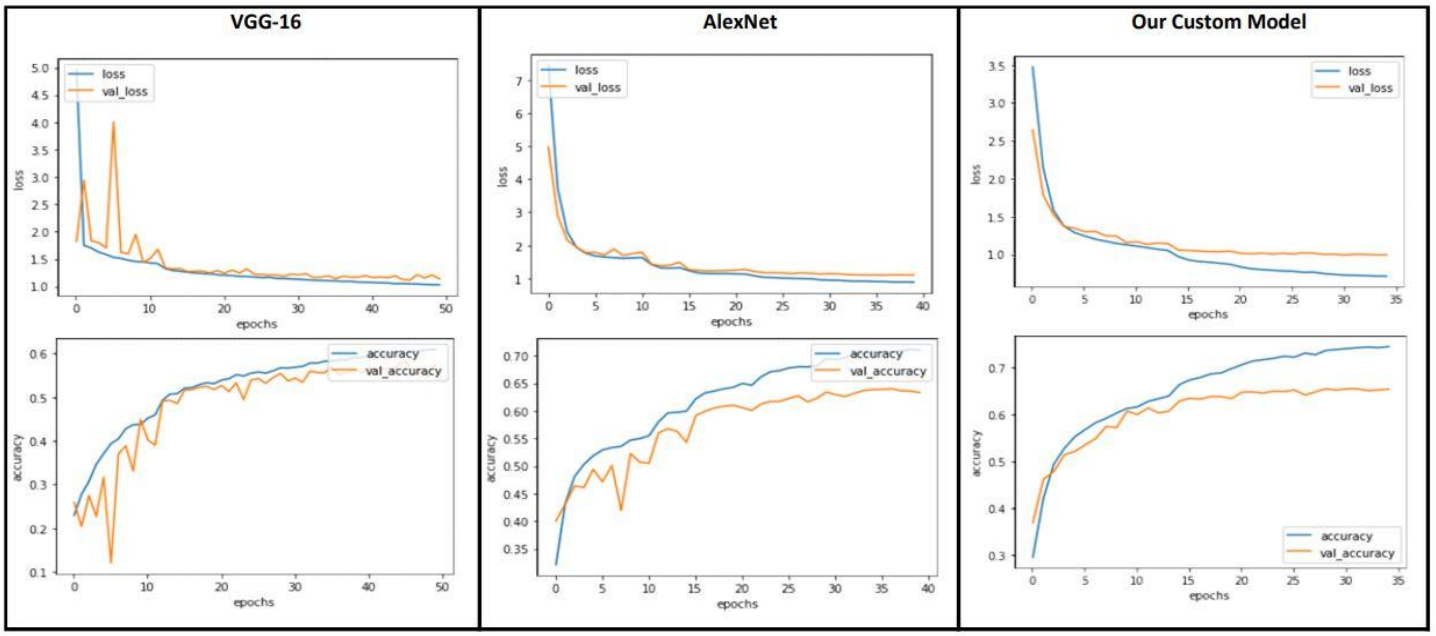

Figure 3: Loss and Accuracy curves for all models

As seen in Table 1, Our custom model is giving the best results in terms of Accuracy, Precision, Recall and F1 score. So, we further analyze the results obtained by the custom model.

Table 2 shows the precision, recall and F1 score of the individual classes. It can be noticed that the highest precision, recall and F1 score is of the "Happy" class. This can be associated with the fact that the class "Happy" has the highest representation of 25% in the data (as shown in Figure 1).

	precision	recall	f1-score	support
Angry	0.61	0.60	0.60	1448.00
Disgust	0.67	0.50	0.58	159.00
Fear	0.54	0.44	0.49	1488.00
Happy	0.87	0.87	0.87	2617.00
Sad	0.56	0.54	0.55	1771.00
Suprise	0.77	0.77	0.77	1175.00
Neutral	0.58	0.71	0.64	1809.00
accuracy	0.67	0.67	0.67	0.67
macro avg	0.66	0.63	0.64	10467.00
weighted avg	0.67	0.67	0.67	10467.00

Table 2: Precision, recall, F1 of each class for custom model

To further investigate the results, we also analyze the confusion matrix shown in Table 3.

	Angry	Disgust	Fear	Happy	Sad	Suprise	Neutral
Angry	864	15	108	47	187	28	199
Disgust	41	80	12	3	18	0	5
Fear	179	11	657	33	301	138	169
Happy	55	0	44	2278	48	45	147
Sad	155	8	202	49	958	21	378
Suprise	35	2	125	64	18	902	29
Neutral	97	3	67	135	180	35	1292

Table 3: Confusion Matrix for custom model

As seen in the confusion matrix, the majority of the instances are correctly classified, however there are a few classes which are misclassified more often than others. It can be noted that "Sad" is often confused with "Fear" and "Angry". Similarly "Neutral" is often confused with "Sad" and "Surprise" is often confused with "Fear".

IV. Conclusion

The classification of facial expressions on the FER2013 Dataset using our custom CNN based model yielded 67.2% accuracy. In this paper we tried 3 different approaches to model construction and highlighted the key differences between the same. The first approach used AlexNet reaching an accuracy of 64.1% while the second approach used VGG16 and reached an accuracy of 58.1%. In our third approach we built our own CNN after comparing the first two approaches and employing the swish activation function

which gave our final accuracy of 67.2%. This result is significant as the state of the art model which is an ensemble of modern day CNNs has a FER2013 test accuracy of 75.8% which relies on comprehensive data augmentation, face registration, additional data and extra features while our model relies on neither. Interestingly, human accuracy on the FER2013 dataset has been reported to be 65±5%, which puts our model in a better spot than most humans.

Furthermore, we believe that our model can be fine tuned to achieve even more accuracy and be relied upon to give real time facial emotion recognition speeds.

References

[1] Alex Krizhevsky, Ilya Sutskever, Geoffrey E. Hinton, "ImageNet Classification with Deep Convolutional Neural Networks", Advances in Neural Information Processing Systems 25 (NIPS 2012).

[2] Simonyan, Karen & Zisserman, Andrew. (2014). Very Deep Convolutional Networks for Large-Scale Image Recognition. arXiv 1409.1556.

[3] Christopher Pramerdorfer, Martin Kampel. (2016). "Facial Expression Recognition using Convolutional Neural Networks: State of the Art". arXiv 1612.02903.

[4] A. Mollahosseini, D. Chan, and M. H. Mahoor, "Going Deeper in Facial Expression Recognition using Deep Neural Networks," CoRR, vol. 1511, 2015.

[5] Amil Khanzada, Charles Bai, and Ferhat T Celepcikay, "Facial Expression Recognition with Deep Learning". arXiv:2004.11823