Real-Time Facial Emotion Recognition

Ritu Agrawal, Nikhil Kashid

ABSTRACT

Intrigued by the overwhelming popularity of Deep Learning and the pace at which it is prevailing, this paper depicts an attempt of creating a deep learning model for real time implementation of facial expression recognition. There has already been amazing work since the origin of Artificial Intelligence and Deep Learning which spawned many different models. The main goal of the project is to classify changing facial emotions of a person and segregating them into one of the seven categories: Happy, Sad, Neutral, Surprise, Anger, Fear and Disgust. This paper includes experiments and trials based on different models (CNN, Residual Network, VGG19) and the model with maximum accuracy is CNN with 3 convolutional layers, 100 epochs, 256 batch size, the optimizer being Adam optimizer with feeding preprocessed data using image processing and data augmentation which lead to obtaining a test accuracy around 67% on the FER2013 dataset.

1 INTRODUCTION

Active research is going on in Facial expression recognition domain in the past few years. Major application areas of it includes: Making cars safer and personalized, Facial Emotion Detection in Interviews, Testing for Video Games and sociable robots. The recognition of facial expressions is not an easy problem for machine learning methods, since facial features can differ significantly in the way they show their expressions. Recognizing human expressions is a key area of research as the ability to judge one's emotions can give access to a plethora of opportunities and applications such as friendly human- computer interaction, better targeted advertisements and market research. While there are gazillions of experimentations to carry out deep learning with human faces, in this paper we will be focusing on one area of this genre – Live recognition of facial expression. In this paper, we have trained three models: CNN with three convolutional layers, prebuilt models such as VGG19 and Residual Network. We have compared test accuracies of each model and got to learn that CNN stands out in these models with training accuracy of 67% and testing accuracy of 64%.

2 METHODOLOGY

2.1 DATA ACQUISITION

For training the model, FER2013 dataset has been used which is from Kaggle's Facial Expression Recognition Challenge. The dataset consists of 35,887 image pixels, a size that will suffice for making a relevant model. The pixels when converted to images, looks like the image below revealing that the data has grayscale images.

Figure 1 Example images from the FER2013 dataset. These images depict expressions, namely anger, disgust, fear, happiness, sadness, surprise, neutral.

The data consists of 48x48 pixel grayscale images of faces. The data categorizes each face based on the emotion shown in the facial emotions in to one of seven categories (0=Angry, 1=Disgust, 2=Fear, 3=Happy, 4=Sad, 5=Surprise, 6=Neutral).

The dataset consists of two columns, "emotion" and "pixels". The "emotion" column contains a numeric

code ranging from 0 to 6, inclusive, for the emotion that is present in the image.

2.2 BLOCK DIAGRAM

The model build is a CNN (Convolution Neural Network) with layers namely: conv2d, Batch normalization, max pooling, dense, flatten and activation.

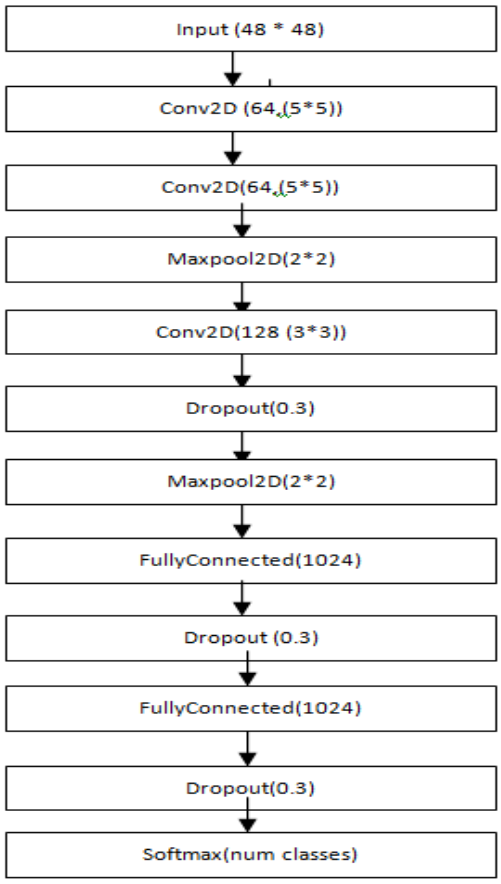


Fig2: CNN Layers

2.3 Convolutional Neural Network Model

Quoting it from Wikipedia-Convolutional Neural Networks are regularized versions of multilayer perceptrons. Multilayer perceptrons usually refer to fully connected networks, that is, each neuron in one layer is connected to all neurons in the next layer. The "fully-connectedness" of these networks make them prone to overfitting data. Typical ways of regularization include adding some form of magnitude measurement of weights to the loss function. However, CNNs take a different approach towards regularization: they take advantage of the hierarchical pattern in data and assemble more complex patterns using smaller and simpler patterns. Therefore, on the scale of connectedness and complexity, CNNs are on the lower extreme."

CNN is a class of deep, feed-forward artificial neural networks that has been applied to analyzing visual imagery. CNN is popular because it learns the filters by itself and these filters reduce the dimensionality of the data being fed to the system.

Layers in CNN model built:

• INPUT layer- The data is being input in the form of 48 x 48 pixels for a grayscale image specified as (48, 48, 1), where 1 tells that the image is grayscale.

• CONV layer will compute the output of neurons that are connected to local regions in the input, each computing a dot product between their weights and a small region they are connected to in the input volume.

• RELU layer will apply an elementwise activation function, such as the max (0, x) thresholding at zero.

• POOL layer will perform a down sampling operation along the spatial dimensions (width, height)

• FC (i.e. fully-connected) layer will compute the class scores.

We implemented a 6 convolution layered neural network for visualizing the internal working from layer-to-layer.

Visualizing intermediate activations consists in displaying the feature maps that are output by various convolutions and pooling layers in a network, given a certain input. This gives a view into how an input is decomposed unto the different filters learned by the network. Each channel encodes relatively independent features, so the proper way to visualize these feature maps is by independently plotting the contents of every channel, as a 2D image.
The 1st layer acts as collection of various edge detectors. At that stage, the activations are still

Layer (type)	Output Shape	Param #
conv2d_1 (Conv2D)	(None, 44, 44, 64)	1664
conv2d_2 (Conv2D)	(None, 40, 40, 64)	102464
batch_normalization_1 (Batch	(None, 40, 40, 64)	256
max_pooling2d_1 (MaxPooling2	(None, 20, 20, 64)	0
conv2d_3 (Conv2D)	(None, 18, 18, 128)	73856
conv2d_4 (Conv2D)	(None, 16, 16, 128)	147584
batch_normalization_2 (Batch	(None, 16, 16, 128)	512
max_pooling2d_2 (MaxPooling2	(None, 8, 8, 128)	0
conv2d_5 (Conv2D)	(None, 6, 6, 256)	295168
conv2d_6 (Conv2D)	(None, 4, 4, 256)	590080
batch_normalization_3 (Batch	(None, 4, 4, 256)	1024
max_pooling2d_3 (MaxPooling2	(None, 2, 2, 256)	0
flatten_1 (Flatten)	(None, 1024)	0
dense_1 (Dense)	(None, 1024)	1049600
batch_normalization_4 (Batch	(None, 1024)	4096
activation_1 (Activation)	(None, 1024)	0
dropout_1 (Dropout)	(None, 1024)	0
dense_2 (Dense)	(None, 1024)	1049600
batch_normalization_5 (Batch	(None, 1024)	4096
activation_2 (Activation)	(None, 1024)	0
dropout_2 (Dropout)	(None, 1024)	0
dense_3 (Dense)	(None, 7)	7175

Total params: 3,327,175
Trainable params: 3,322,183
Non-trainable params: 4,992

retaining almost all of the information present in the initial picture.

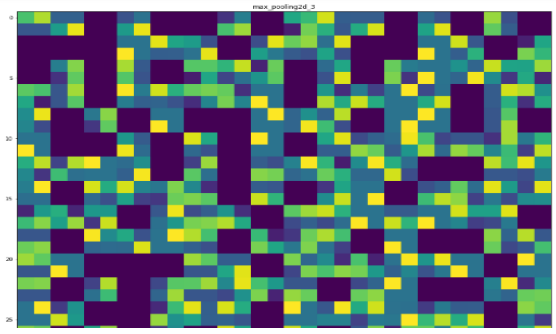


Figure 3 Showing the 12th layer in our CNN model

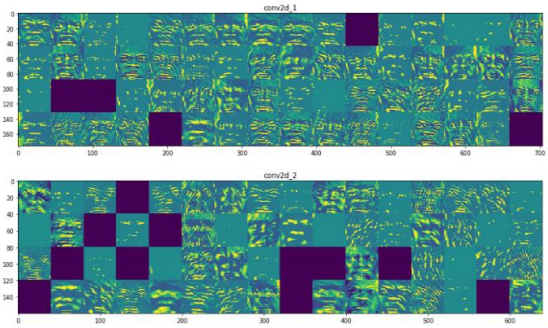


Figure 2 The first two layers in CNN model

As we go higher-up, the activations become increasingly abstract and less visually interpretable. Higher-up presentations carry increasingly less info about the visual contents of the image, and increasingly more info related to the class of the image.

The activations of layers higher-up carry less and less information about the specific input being seen, and more and more information about the target. A deep neural network effectively acts as an information distillation pipeline, with raw data going in, and getting repeatedly transformed so that irrelevant information gets filtered out while useful information get magnified and refined.

Thus, In this paper, we consider the task of predicting basic emotions from single images and live video stream using CNNs. We note that it is straight-forward to adapt image-based methods to support image sequences by integrating per-frame results using graphical models. The conclusions drawn in this paper are thus relevant for sequence based facial emotion recognition as well.

2.4 VGG19 MODEL

It has 19 layers with multiple 3X3 kernel-sized filters one after another. With a given receptive field(the effective area size of input image on which output depends), multiple stacked smaller size kernel is better than the one with a larger size kernel because multiple non-linear layers increases the depth of the network which enables it to learn more complex features, and that too at a lower cost. We followed following steps to train the data using VGG19.

Mechanism followed for VGG19

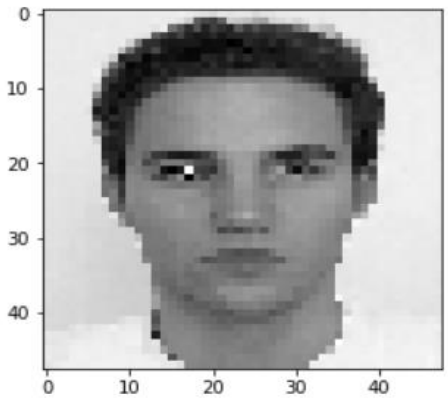
Building VGG Network: In this step we have stored image data in ‘network’ variable using ’input_data’ function. Conv_2d function is used after this to create convolution input and it produces tensor output. Output from conv_2d is then fed to ‘max_pooling’ (max_pool_2d) layer and it returns tensor output. This tensor output is then used by ‘fully_connected’ layer. This layer creates variable called ‘weightmatrix’ which is multiplied by inputs to produce required output tensor.

We have used RELU as an activation function for this model.

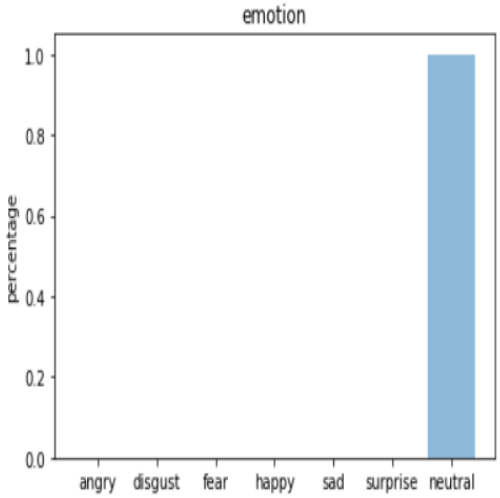
Training and evaluating the model: Built model is trained using Deep Neural Network (DNN from TFLearn). After training the model we checked the

accuracy of the model. Test accuracy of the model was 57%.

Plotted graph of the custom image:
Using trained model we validated the accuracy of it on a custom image. We got the fairly good results and



Custom image (Neutral Expressions)



Graph of the expression of the custom image

Detecting expressions by feeding a video or webcam:

Conclusion: We used VGG19 model and got a training accuracy of 99.3 % and a testing accuracy of 57.0 %. VGG19 has been a really effective model and also has won the Imagenet competition but with our data we can observe that it clearly tends to overfit as there is humongous difference between training and testing accuracy.

2.5 RESEDUAL Network

As deeper networks always outperform the less deep ones, residual networks are highly deep neural networks. With a residual block, however, a skip function reduces the number of times a linear function is used to achieve an output. A skip function creates what is known as a residual block.
As per what we have seen so far, increasing the depth should increase the accuracy of the network, as long as over-fitting is taken care of. But the problem with increased depth is that the signal required changing the weights, which arises from the end of the network by comparing ground-truth and prediction becomes very small at the earlier layers, because of increased depth. It essentially means that earlier layers are almost negligible learned. This is called vanishing gradient. The second problem with training the deeper networks is, performing the optimization on huge parameter space and therefore naively adding the layers leading to higher training error. Residual networks allow training of such deep networks by constructing the network through modules called residual models.

TFLearn Library

TFLearn is a modular and transparent deep learning library built on top of Tensorflow. It was designed to provide a higher-level API to Tensorflow in order to facilitate and speed-up experimentations, while remaining fully transparent and compatible with it.
TFLearn features include:
Easy-to-use and understand high-level API for implementing deep neural networks, with tutorial and examples. Fast prototyping through highly modular built-in neural network layers, regularizers, optimizers, metrics. Full transparency over tensorflow. All functions are built over tensors and can be used independently of TFLearn. Powerful helper functions to train any TensorFlow graph, with support of multiple inputs, outputs and optimizers. Easy and beautiful graph visualization, with details about weights, gradients, activations and more. Effortless device placement for using multiple CPU/GPU.

Conclusion:
The accuracies we have got in this model are:
Training Accuracy: 0.9160
Testing Accuracy: 0.5781

RESNET have the concept of residual blocks which helps in resolving the degradation problem as we increase the number of layers in the neural net. Here, we see that there is overfitting problem as the there is vast difference between training and testing accuracy.

3.0 EXPERIMENTS

As a part of experimentation we have implemented CNN model using three convolutional layers. We have fixed image size of (48X48) and we taking help of RELU as an activation function. Initially, we have trained our model with 100 epochs. However we have observed that training accuracy of the model increases gradually with increased number of the epochs. Hence we trained the same model again on 200 epochs and we got training accuracy of 67%. In order to increase the testing accuracy, we added batch normalization in every convolution layer. Batch normalization is used to normalize output of previous activation layer by subtracting the batch mean and dividing by the batch standard deviation. This gives required stability to a neural network. Adding batch normalization further improved the test accuracy to up to 65%. We fine-tuned hyper-parameters of the model and finally got training accuracy of 67%.

Figure 4 Training and validation accuracy

Figure 5 Training and validation loss

In order to check the model we tried our model by making prediction of a custom image out of test set. We got Figure 7 as an output for the test.

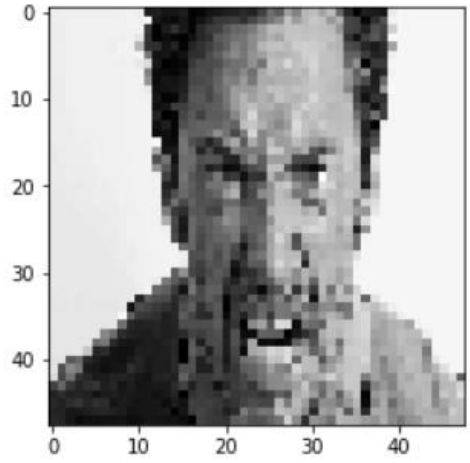


Figure 6 Sample image for testing

And we can observe the result predicted by the model in the Figure 8.

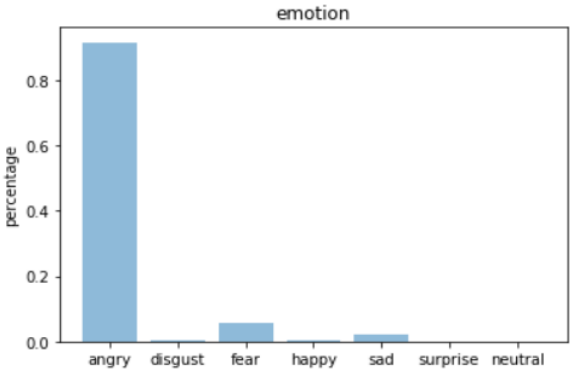


Figure 7 Prediction of model on sample image

Out of the many test we conducted it seems that model is able to predict well for expressions such as happy, neutral, angry, surprise. The model however struggles to predict disgust. It is because the training set didn't have enough image for the model to learn. We can see from the confusion matrix in Figure 9 the model performance.

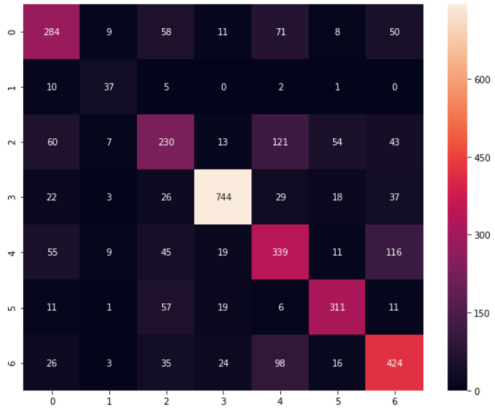


Figure 8 Confusion Matrix for prediction of various expressions [Index: 0 – Angry 1 – Disgust 2 – Fear 3 – Happy 4 – Sad 5 – Surprise 6 - Neutral]

We are using OpenCV model for live streaming and to capture live frames from the webcam. In first step individual frames will be pre-processed and cropped. In next stage, cropped image will get converted in to grayscale and adaptive histogram equalization technique CLAHE will be applied. CLAHE (Contrast Limited Adaptive Histogram Equalization) used in OpenCV divides the image into tiles of specified pixels . It also applies histogram equalization of tiles. Hence got resulting single or multiple bounding boxes are fed into the trained CNN model for expression recognition.

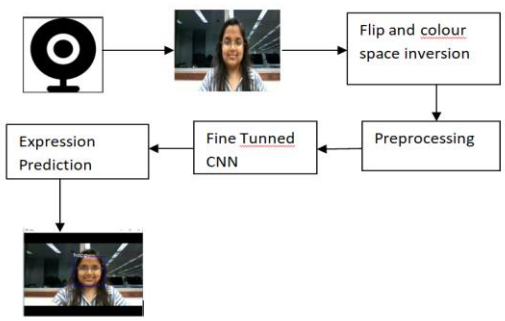


Figure 9 Sequential block diagram of Real Time Expression Detection and Classification

4.0 CONCLUSION

We implemented a real time facial expression recognition system successfully on fer2013 dataset. We successfully implemented a real time facial expression recognition system using three models. Convolutional Neural Network (CNN) performs best between CNN, VGG19 and ResNet and we got 67.09% accuracy for CNN.

Accuracy over test data using VGG19 model over 200 epochs is: 57%

Accuracy over test data using ResNet model over 100 epochs is: 64.14%

Future scope would be to create a model which could differentiate between subtle facial expression changes and to display the model prediction percentage in real time.

5.0 REFERENCES

[1] Deep convolutional network for image recognition.

https://arxiv.org/pdf/1409.1556.pdf

[2] Keras

https://www.pyimagesearch.com/2017/03/20/imagenet-vggnet-resnet-inception-xception-keras/

[3] TFLearn layers

http://tflearn.org/layers/conv/

[4] Residual Network & VGG

https://cv-tricks.com/cnn/understand-resnet-alexnet-vgg-inception/

[5] For face detection
https://www.youtube.com/watch?v=PmZ29Vta7Vc

[6] Residual Network

https://github.com/TFLearn/TFLearn/blob/master/examples/images/residual_network_cifar10.py

[7] OpenCV

https://towardsdatascience.com/tagged/opencv

[8] For VGG19 and Residual Network https://github.com/TFLearn/TFLearn/tree/master/examples/images

[9] For TFLearn

http://tflearn.org/tutorials/

[10] Article on Facial Emotion Recognition using CNN http://sefiks.com/2018/01/01/facial-expression-recognition-with-keras/

[11] Reference document for VGG19 CNN https://www.mathworks.com/help/deeplearning/ref/vgg19.html;jsessionid=ccf9599bd865b423281a56299a68

[12] YouTube Video lecture for face detection https://www.youtube.com/watch?v=PmZ29Vta7Vc

[13] Facial Emotion Recognition using CNN http://sefiks.com/2018/01/01/facial-expression-recognition-with-keras/

[14] Github:VGG19

https://github.com/TFLearn/TFLearn/blob/master/examples/images/VGG19.py

[15] Recognizing Facial Expressions Using Deep Learning

http://cs231n.stanford.edu/reports/2017/pdfs/224.pdf

[16] Gihub- Residual Network

https://github.com/TFLearn/TFLearn/blob/master/examples/images/residual_network_cifar10.py