Facial Emotion Recognition using Convolutional Neural Networks

Sudarsini Tekkam Gnanasekar, Oke Obuareghe, Noor Abid, Uvesh Shelat and Ankit Prajapati
ENEL 645 Project
Department of Electrical and Software Engineering, University of Calgary, Canada

***Abstract*—Facial expression recognition for humans in static images and video frames using machine learning algorithms and convolutional neural network models is a highly challenging task due to the lighting conditions, posture, background/noise, and other factors like computational complexity, speed, processing time involved in detecting the expression in the image. To address this issue, we propose an effective technique using a smaller Convolutional Neural Network (CNN) Model which is comparable to the previous works in Facial emotion recognition in achieving a good prediction accuracy. The model consists of 4 convolutional layers and a max pooling layer after each convolutional layer trained for 100 epochs and can obtain a training accuracy of 75% and a test accuracy of 62% in predicting the 7 different emotions using the fer2013 dataset. We expect that this model developed for this Facial Emotion Recognition System can be used as an alternative to other CNN models having many layers, which increases the training time and complexity. Facial emotion recognitions becoming more and more advanced every year. Facial Emotion recognition provides benefits to many fields such healthcare, and security.**

I. Introduction

Emotions are quite imperative for making decision and interaction with others. Facial expression is one of the most important ways for humans to display their emotions. There are seven basic kinds of emotions such as anger, happy, sad, surprise, neutral are introduced. A better recognition of human expression will be an effective way of communication [1]. In recent years, the automatic Facial Emotion Recognition (FER) has been developed with the help of Artificial Intelligence [2]. For real time facial recognition, deep learning model and CNN algorithm are profound choices. Deep learning is imperative for features extraction using pattern detection and pattern classification from sample images. The information extracted from feature extraction will be useful for real time facial emotion recognition [2].

II. Related work

Recently, the importance of emotion recognition has increased at the business level. Hence, the related field in academic has also enhanced. The list of such studies is provided below. Yu & Zhang introduced a facial detection system for its Facial Expression Recognition (FER) which is based on convolutional neural networks. In this system, the distribution of images is done in different ways in training sequence and each test image result is the average voting of all distributed sample responses[4].

Kim, Roh and Lee [5] presented a deviation in the architecture of the network. The parameters are trained using weights of the model in different database and then the re – training is applied. The 61.6% accuracy was acquired for static facial emotion recognition using EmotiW2015 dataset for solving seven-class problem, a categorical classification.

Ranganathan et al. provided the emoFBVP database for emotion recognition using multimodal recordings [6]. In this database, 23 different emotions are displayed by subject and each emotion has different intensity. After that, Convolutional Deep Belief Model (CDBN) was used for emotion detection using his own dataset. The recognition with better accuracy was achieved for low intensity or precise emotions expression. Our model will be training the FER2013 dataset using CNN to detect 7 distinct emotions which could be deployed in an application for security or healthcare purposes.

III. Materials and Methods

A. Dataset collection

We started out by importing all necessary libraries. The dataset, FER2013 was downloaded from Kaggle [7]and imported into our Jupyter notebook and inspected. The dataset has 35,887 images which are classified into 7 emotions. The file contains 3 columns:

Emotion: A digit ranging from 0 to 6 representing the emotion illustrated on the corresponding photo as follows: 0 - 'Angry', 1 - 'Disgust', 2 - 'Fear', 3 - 'Happy', 4 - 'Sad', 5 - 'Surprise', 6 - 'Neutral'. Pixels: A string of 2,304 numbers, they are the pixel intensity values of our image. Usage: This column indicates the set which a particular image belongs to (Training, PublicTest, and PrivateTest).

B. Data Preprocessing

As mentioned in the Usage column, the dataset came segregated into a set for Training which was 80% of the entire dataset, 10% of the total set for Validation (PrivateTest), and the remaining 10% for Test (PulicTest). Each image pixels which are the features were stored in a list, same for their corresponding emotion labels, depending on their sets, i.e X_train and Y_train for the Training set, X_val and Y_val for the Validation set, and X_test and Y_test for the Test set. The distribution of data are as follows: Number of images in Training set: 28709 Number of images in Validation set: 3589 Number of images in Test set: 3589 The lists were converted into numpy arrays of float32 precision, and the labels were One-hot encoded using the to_categorical() function.

The datasets were then standardized by subtracting its mean and dividing its value by the standard deviation of that pixel, across all the feature sets.

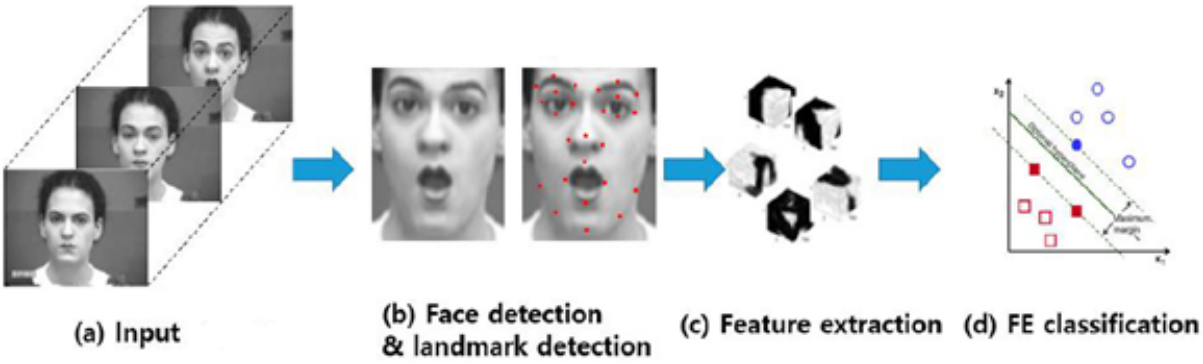

Fig. 1. Facial Emotion Recognition Procedure[3]

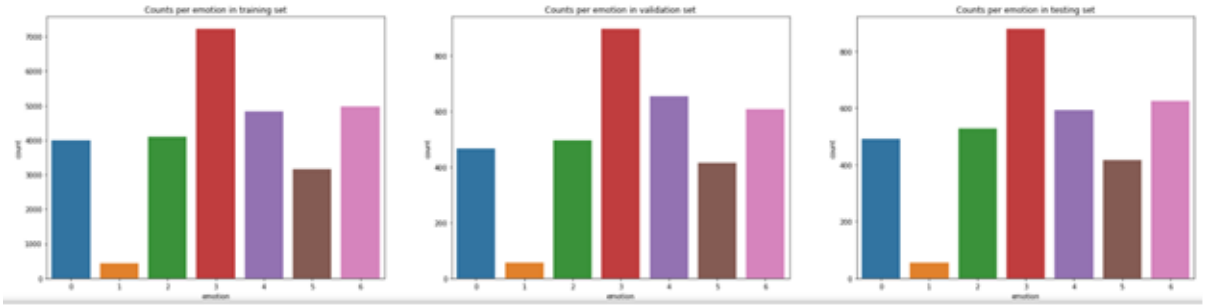

Fig. 2. Class Distribution among each sets[8]

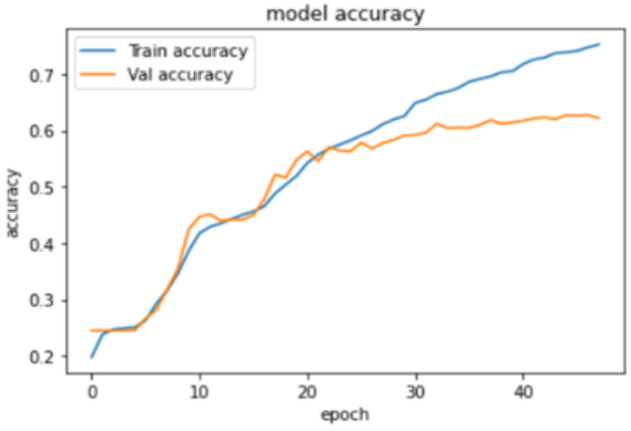

Fig. 3. Plot of Training and Validation Accuracy for 4 layers and 100 epochs

C. Model Definition

We used Keras Sequential Convolutional Network having linear stack of layers to construct our neural network. This network architecture consists of following components:

- Convolutional Layers: These layers are the building blocks of our network. They contain filters or kernels, and compute dot product between their weights and the small regions to which they are linked. This is how these layers learn certain features from these images.
- Activation functions: are those functions which are applied to the outputs of all layers in the network. In this project, we will resort to the use of two functions— Relu and Softmax for the output layer.
- Pooling Layers: These layers will downsample the operation along the dimensions. This helps reduce the spatial data of the feature maps and minimize the processing power that is required.
- Dense layers: These layers are present at the end of a C.N.N. They take in all the feature data generated by the convolution layers and do the decision making.
- Dropout Layers: randomly turns off a few neurons in the network to prevent overfitting.

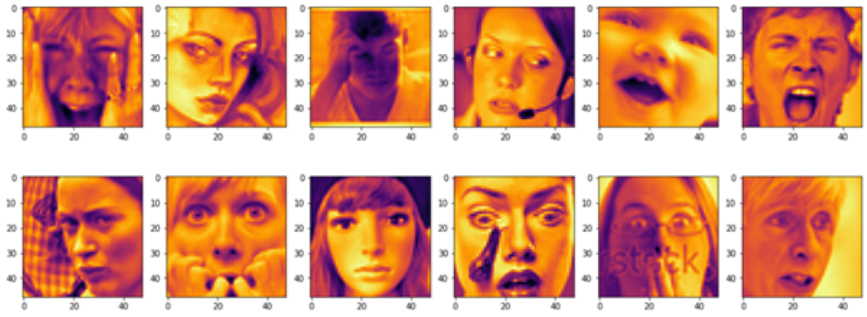

Fig. 4. Grad-Cam result

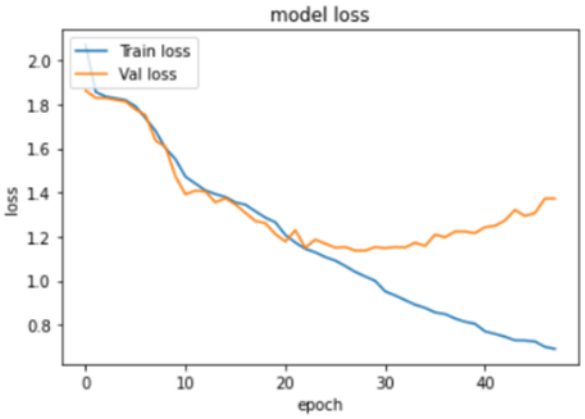

Fig. 5. Plot of Training and Validation loss for 4 layers and 100 epochs

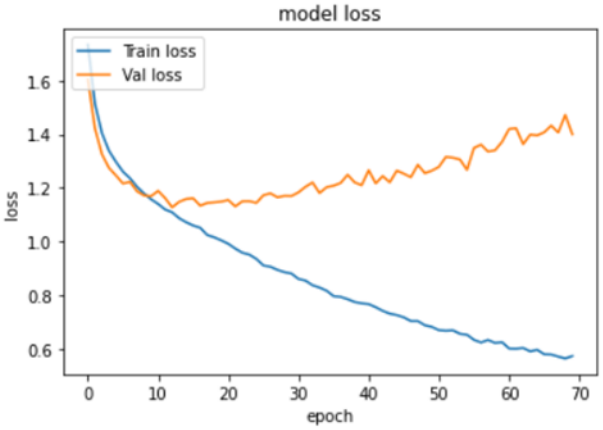

Fig. 7. Plot of Training and Validation loss for for 3 layers and 70 epochs

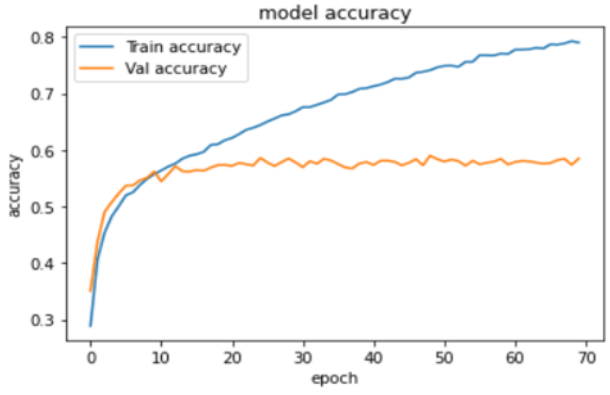

Fig. 6. Plot of Training and Validation loss for 3 layers and 70 epochs

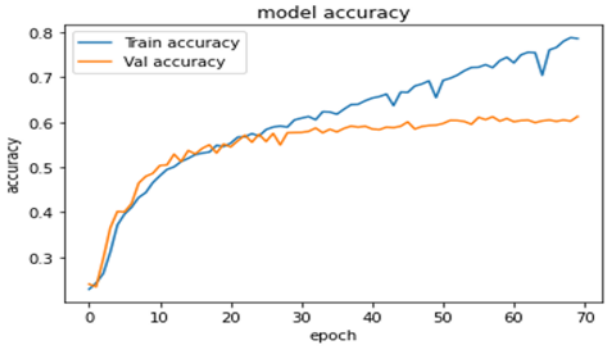

Fig. 8. Plot of Training and Validation loss for 5 layers and 70 epochs[9]

- Batch Normalization: normalizes the output of a previous activation layer by subtracting the batch mean and dividing by the batch standard deviation. This speeds up the training process.

We then compile the model architecture using the Adam optimizer. For a multiclass classification task such as this, we have utilized categorical_crossentropy as our loss function. The image of our model architecture can be found here[10]

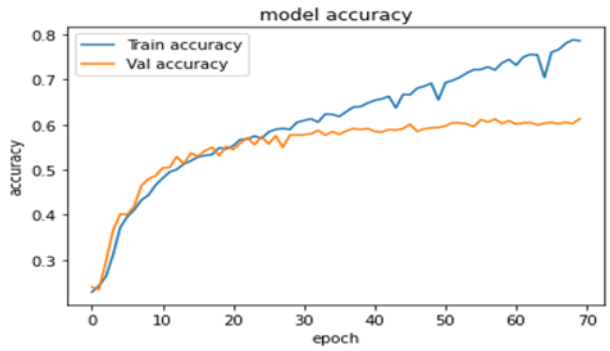

Fig. 9. Plot of Training and Validation loss for 5 layers and 70 epochs[9]

D. Define Callbacks

Callback are functions which are called after every epoch during the training process. We have adopted the following callback functions

- LearningRateScheduler: Before an iteration, this function gets the updated learning rate from a schedule we defined that divides the learning rate by 2 if the epoch index is not 0 and completely divisible by 10.
- EarlyStopping: While training a neural network, sometimes the progress stalls and we stop seeing any improvement in the validation loss. Majority of the time, this means that the network won't converge any further and there is no point in continuing the training process. This function waits for a specified number of epochs ("patience") and terminates the training if no change in the parameter is found. For this we set our patience value at 20, to monitor the validation loss parameter.
- ModelCheckpoint: Training neural networks generally take a lot of time, and anything can happen during this period that may result in loss of all the variables and weights. Creating checkpoints is a good habit as it saves your model after every epoch. In case your training stops you can load the checkpoint and resume the process.

IV. Experiments Results

After defining our model and callbacks, we proceeded to training our model on the Training set and validating using the validation set.

For this project we conducted more than one experiment. The best model that we got had a training accuracy of 75%, validation accuracy 62% and testing accuracy of 62%. In this model, we used 4 convolution layers and set our number of epochs to 100, and our batch size to 64, meaning our training data would pass through the model 100 times in batches of 64 images. We also included regularization parameter, kernel_regularizer of 0.01 and learning rate of 0.001. The model trained for 48 epochs and was terminated by EarlyStopping due to a lack of improvement in the validation loss after 20 different iterations. We considered this model to be the best because we got the highest testing accuracy of 62% on the unseen test data. Figure 3 shows the plot of Training accuracy and Validation accuracy during training and Figure 5 shows the plot of Training loss and Validation loss.

To interpret our model, we have used Gradient-weighted Class Activation Mapping (Grad-Cam). The method is an improvement over previous methods in versatility and accuracy. It is complex but, luckily, the output is intuitive. From a high-level, we take an image as input and create a model that is cut off at the layer for which we want to create a Grad-CAM heat-map. We can see that in figure 4, we have the Grad-CAM heat-maps for the last convolutional layer in our model. Grad-CAM heat-maps identify erroneous areas of emphasis by our models [11][12].

V. Discussion

In the second experiment, three convolutional layers were used to train the model. It was trained for 25 epochs. The accuracy rate was 66%, the validation accuracy was 57%, and the testing accuracy was 57%. From the result we can see that the model is not overfitting, but we did not select this model as our best model since the testing accuracy was comparatively lower than our best model.

In the third experiment, three convolutional layers also were used to train the model. It was trained for 70 epochs. The accuracy rate was 79%, the validation accuracy was 58%, and the testing accuracy was 58%. Here the model was defiantly overfitting and the testing accuracy was also lower than our best model. To reduce overfitting we tried to increase the number of layers to five convolution layers in the next experiment.

In the fourth experiment, we tried with 5 convolution layers and for 70 epochs but the training accuracy that we achieved was 78%, validation accuracy 61%, and testing accuracy was again 61%. We can notice here that we have less overfitting than experiment 3, so adding more convolution layers did help with reduce overfitting.

There can be different ways to increase the test accuracy. We already used regularization in our model. In the future work we can try increasing our training set and /or emotions classes. We can also use other model architectures like RestNet, VGG16, and EfficientNet to train our model. For overfitting, we can also try adding more data, use data augmentation or reduce model architecture complexity. In the future, we can try the suggested methodologies above to improve our model accuracies.

VI. Conclusion

The overall goal for this project was to detect human emotions in real-time using a host of technologies and frameworks. In addition to this, deep learning was introduced and applied in order to improve analyzing as well as recognizing human facial emotions. The Kaggle face recognition database (Facial Emotion Recognition 2013) was used in this project to detect seven emotion classes comprising of Happy, Sad, Surprise, Neutral, Fear, Disgust and Anger. After training our

model, we checked for posed facial emotions and spontaneous emotions, where we observed that the model is not always accurate for spontaneous emotions. The model we trained worked effectively in uneven lighting, different backgrounds and horizontal and vertical head rotation of approximately 20-25We conducted 4 experiments. The best model that we got had a training accuracy of 75%, validation accuracy 62% and testing accuracy of 62%. The results shows that the model can recognize emotions quite efficiently.

REFERENCES

[1] A. Hassouneh, A. Mutawa, and M. Murugappan, "Development of a real-time emotion recognition system using facial expressions and eeg based on machine learning and deep neural network methods," *Informatics in Medicine Unlocked*, vol. 20, p. 100372, 2020.

[2] M. A. Adil, "Facial emotion detection using convolutional neural networks," 2021.

[3] B. C. Ko, "A brief review of facial emotion recognition based on visual information," *sensors*, vol. 18, no. 2, p. 401, 2018.

[4] Z. Yu and C. Zhang, "Image based static facial expression recognition with multiple deep network learning," in *Proceedings of the 2015 ACM on international conference on multimodal interaction*, 2015, pp. 435–442.

[5] B.-K. Kim, J. Roh, S.-Y. Dong, and S.-Y. Lee, "Hierarchical committee of deep convolutional neural networks for robust facial expression recognition," *Journal on Multimodal User Interfaces*, vol. 10, no. 2, pp. 173–189, 2016.

[6] H. Ranganathan, S. Chakraborty, and S. Panchanathan, "Multimodal emotion recognition using deep learning architectures," in *2016 IEEE Winter Conference on Applications of Computer Vision (WACV)*. IEEE, 2016, pp. 1–9.

[7] "Kaggle," https://www.kaggle.com/datasets/deadskull7/fer2013, accessed: 2022-04-04.

[8] "Enel645 project group 2," https://github.com/oobuareghe/ENEL645-PROJECT_GROUP-2, accessed: 2022-04-04.

[9] "github facial emotion recognition code," https://github.com/rmsouza01/ENEL645/blob/master/JNotebooks/tutorial04_fully_connect, accessed: 2022-04-04.

[10] "Enel645 project group 2 fig," ENEL645-PROJECT_GROUP-2/convnet.pngatmainÂůoobuareghe/ENEL645-PROJECT_GROUP-2ÂůGitHub, accessed: 2022-04-04.

[11] "github alzheimer notebook," https://github.com/rmsouza01/ENEL645/blob/master/JNotebooks/alzheimer_classificiation.ipynb, accessed: 2022-04-04.

[12] "keras," https://keras.io/examples/vision/grad_cam/#:~:text=The%20Grad%2DCAM%20algorithm, accessed: 2022-04-04.