Emotion Detection from Face using Deep Learning

Master of Engineering

in

Software System Engineering

University of Regina

By

Oisharya Dhar

200441728

Regina, Saskatchewan

August 29, 2022

Contents

1 Introduction

When people interact with each other, they judge the other person by their emotional state and according to their facial expressions the conversation takes turns. In the case of human-computer interaction, this facial expression can have a very big impact to develop a conversation between humans and computers with the correct approach. Because of this reason, automatic detection of human emotion is gaining popularity day by day as it can open the doors for analyzing human effective behavior in various disciplines such as psychology, linguistics, computer science, and other related disciplines. Mainly the automatic detection of human emotion can have three distinctive stages, face detection, facial expression data extraction, and finally emotion detection or recognition. As face detection is not a newly developed method in the machine learning area, this project aims at the last two parts of the recognition system. And in this project, I dived into the aspect of transfer learning, especially the Machine Learning part. For this emotion detection model, I have used DenseNet as transfer learning along with general Machine learning techniques. As in this project I am using a transfer-learning technique, the model provides a system with immensely powerful processing capabilities with less training time and a wide range for detecting emotions. The whole workflow of this project can be given by,

- analyzing the dataset and loading data to feed to the machine learning model.
- build and train machine learning model
- analyze the result

2 Knowledge and data representation

In this section, I will describe the related work and the dataset that I have used in this project and the scope of this project in the real world.

2.1 Literature Review

Jaiswal et al. [1] proposed a deep learning-based supervised algorithm for detecting facial emotion from images. For this purpose, they have used The Convolutional Neural Network (CNN) and they have also divided the whole procedure into three parts: face detection, feature extraction, and emotion classification. They started their model using the buthe the the ilt-in Keras library in python with some modifications in the basic CNN architecture for image recognition. To improve the model Jaiswal et al. [1] made some changes to the CNN from the previous methods and this change increases the accuracy of the model as well. The dimension of the input layer of CNN was kept similar to the dimension of the input image. For the second step, which is the feature extraction, they have used two parallel deep learning models which are then concatenated for better accuracy. Both the models have the same input and the kernel size is also the same. After extracting the features, the outputs from both the sub-models (Model-A & Model-B) are flattened to turn into vectors, and then they were concatenated into one long vector matrix. This long vector matrix is then fed to the fully connected layer of CNN for the classification task. Apart from the output layer, the ReLU activation function was used in the whole architecture, and in the output layer the SoftMax activation function was used. Furthermore, they used a 0.2 dropout rate for reducing the overfitting of the model. They trained their model for seven different emotions (anger, happiness, disgust, sadness, neutral, fear, and surprise) and tested the whole architecture with two different datasets: Japanese Female Facial Emotion (JAFFE) [2] and the FERC-2013. The images in the JAFFE were preplanned and posed which means images were clean without deformation, on the other hand, the FERC-2013 was the dataset of random images clicked from the wild scenes. To improve the performance of the model they further added a second layer of the max-pooling layer. They have trained their model for 100 epochs with 20000 images. By altering the classic CNN architecture, they have reduced computational time, and accuracy was increased than the previous methods. They have used the confusion matrix to evaluate their model. Their proposed model shows

70.14% average accuracy for the FERC-2013 dataset and for JAFFE database they achieved an average accuracy of 98.65%.

This paper shows promising accuracy by using two types of datasets and how the change of classical architecture can bring a good fluctuation in the average accuracy. Moreover, in this paper, they have used dropout rate and cross-entropy loss which is beneficial for training a model using machine learning. The only drawback I think this paper has is that they could increase the accuracy for the FERC-2013 dataset by fine-tuning the model and for the JAFFE dataset they could have done data augmentation for more robust recognition.

2.2 Scope of the Project

The scope for emotion detection in modern world is immense. Some of the scope for this project is given below,

- Using machine learning or computer vision, some company develops their games such a way that it can detects emotions via the webcam during the game and change the strategy of the game and thus making the game more realistic to play.
- Facial expression recognition surely can help medical experts diagnosis of many diseases such as depression, and anxiety at the early stages.
- Different kinds of mobile and computer application is developing by adapting kids' emotions. And according to the child's emotions, the given task will change its difficulty.
- One of the most important aspects of facial expression recognition is that it can help special needed children to recognize other people's emotions and ease the problem of communication.
- Emotion recognition can also help us at an industrial level with the help of emotion recognition the company can measure the level of stress and anxiety of employees..

2.3 Dataset

For this project, the FER2013 dataset is being used which contains a total of 35,685 images of facial expressions mined from the internet including human and animated faces. All the images are of greyscale and 48x48 pixel size. Moreover, this dataset only represents up close faces image which is divided into train and test datasets. There is a total of seven classes in seven different facial expression dataset which represents seven different (chapter the pines, neutral, sadness, anger, surprise, disgust, and fear).

Figure 1: Representation of the dataset

2.4 Reason for choosing this project and this method

CNN is a well-established method in the case of classification and recognition tasks. But for multi-class classification, sometimes because of the number of layers many features get lost in the pathway(vanishing gradient). As a result, the accuracy of this kind of recognition of the multi-class model is quite low. Moreover, the newly proposed method with many deep layers

along with some modifications shows promising results in this field. In this project, I tried to use the new modified deep neural network to solve the problem of gradient vanishing and also utilize the classification accuracy of the classic CNN model. On the other perspective, I choose this topic to make a bridge between human-computer interaction. Even for any simpler work such as face detection, emotion recognition along with face detection can help many researchers.

3 Approach, Techniques, and Algorithms

The methodology of this project can be divided into two parts, dataset acquisition, and the Machine Learning model. The deep neural-based model contains different kinds of layers and functions such as a convolutional layer, fully connected layers as a dense layer, pooling layer, and so on. A brief overview of these functions is given below,

- Convolutional Layer: It is a part of Convolutional Neural Network(CNN) architecture that works as a filter that strides through the whole input image (striding depending on filter size) and generates a feature map. As it is a filter the output size using this filter is much smaller than the input one.

- Dense layer or Fully Connected Layer is where each current neuron or node gets the input from all the neurons of the adjacent previous layer (flatten). Here, all the nodes are fully connected and thus the name comes.

- Average pooling is the way of taking the average value from each patch of the feature map where the pooling kernel is set.

- Activation function: 1. Rectified Linear Unit (ReLU) [3] is generally used as the activation function for all the convolutional layers and dense layer apart from the output layer. This activation function give the model non-linearity by setting all the negative values to zero and preserves only positive values. And thus it help with noise

deduction of data.

2. SoftMax [4] activation function is generally used in the output layer in machine learning model. With this function the output is also normalized by converting the inputs from weighted sum values into probabilities(range from zero to one).

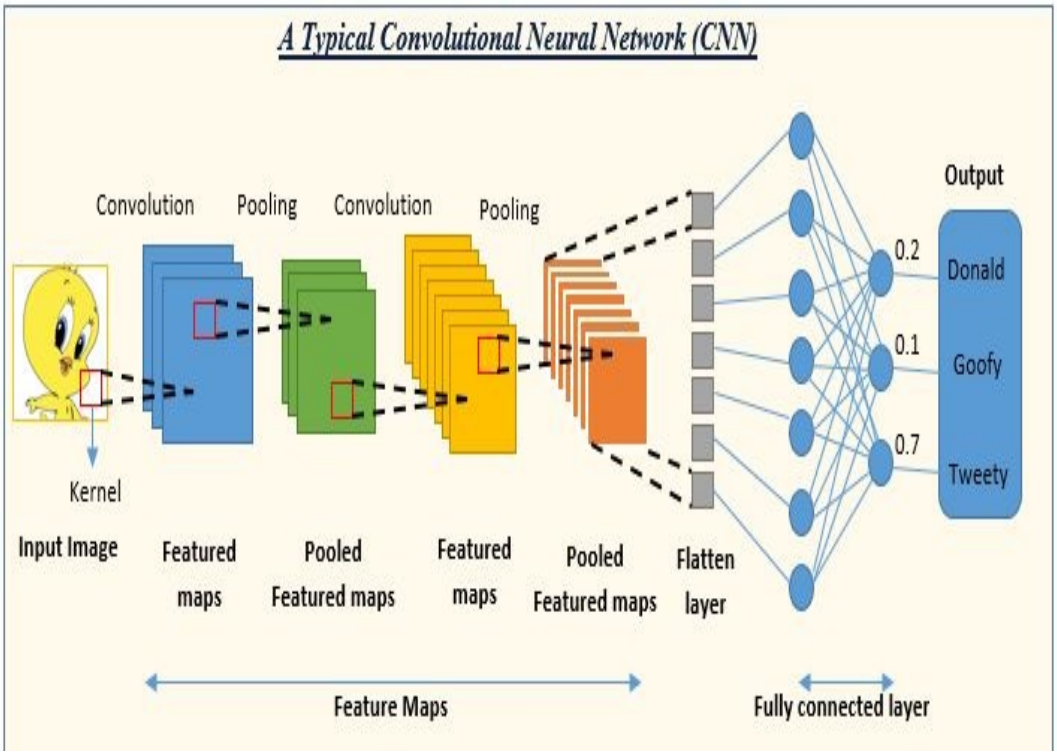

Figure 2: A Flow diagram of CNN model [5]

Figure 2 shows the diagram of the classic CNN network along with the convolutional layer, pooling layer, and dense(fully connected layer).

- Adam optimizer is an adaptive learning method. It computes individual learning values for different parameters.

- LearningRateSchedular: It is a built-in function of Keras which helps to set the learning rate automatically. Here in the final model as call back this function is used.

- Categorical cross-entropy is loss measure function that measure the loss by using the performance of a classification model whose output is a probability value between 0

and 1.

- DenseNet169 is a densely connected CNN model which has 169 layers. Unlike the traditional CNN, DenseNet uses a skip connection between the layers. That means a current layer of this model is fed with not only the output of its previous layer but also with all the feature maps generated by all past layers are connected to its previous. What makes this DenseNet special is before feeding to a layer all the input feature map is concatenated. Moreover, as there are many layers the feature map size must be changed but to concatenate all the feature maps it is important to have the same size of feature map. For this reason, DenseNet introduced transition layers where the feature map size is changed to the size of the input.

4 A Structural diagram and explanation

In this project, the model can be divided into two parts: Feature Extractor and Classifier. From the figure3, this is very clear that the first part of the whole model is the feature extractor which is the DenseNet169. DenseNet169 as the name goes has 169 dense layers where each layer is connected to its all precedent layers. Moreover, this DesnseNet169 is pre-trained on imagenet to increase accuracy.

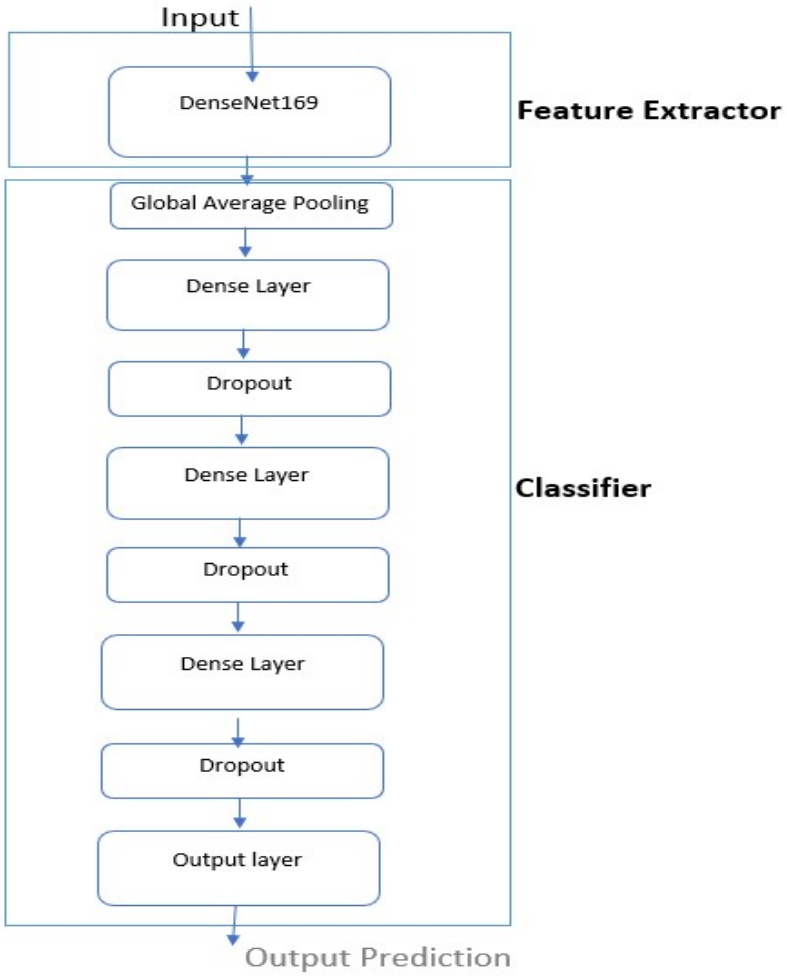

Figure 3: Structural Diagram of the model

The second part of the model is the classifier which is a classic CNN network. The output of the feature extractor works as the input of this network. Firstly, the output of the extractor goes through the pooling layer to pool all the average values and make the feature map trainable for this model. In this model, dropout is used, where randomly a fixed number of neurons are deleted from each dense layer. This dropout makes the model more robust for classification. Finally for activation functions other than the output layer, in each layer, ReLU is used. In the output layer, the SoftMax activation function is used.

5 Implementation

5.1 Techniques

- Data set Understanding: The project started with understanding the dataset and its class.

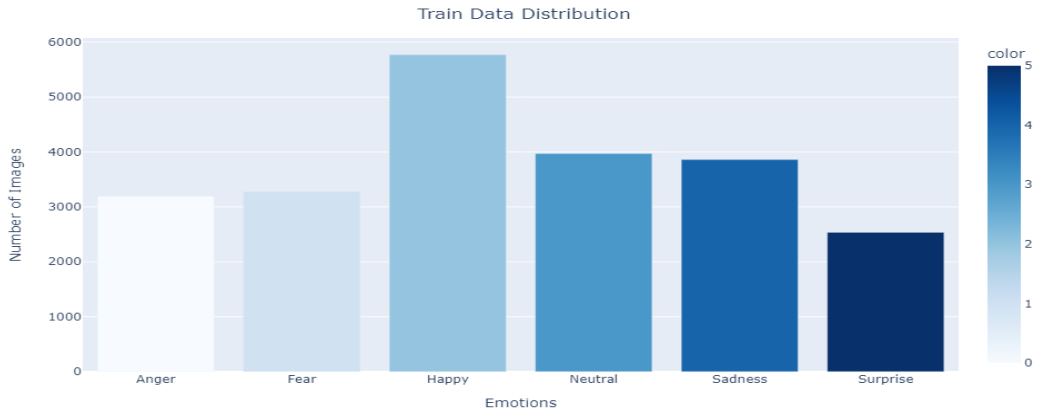

Figure 4: Distribution of Dataset

It is clear from this histogram that the most data is available for the happy emotion and the average number of the image in each class is around 3500.

- Data augmentation: As for any ML algorithm to work properly lots of data is needed. And face can be detected at any angle so for that reason the first task will be data augmentation.

```
# data augmentation for training, validation and testing
train_datagen = ImageDataGenerator(horizontal_flip=True,
                                   width_shift_range=0.1,
                                   height_shift_range=0.05,
                                   rescale = 1./255,
                                   validation_split = 0.2,
                                   preprocessing_function=new_preprocess
                                  )
test_datagen = ImageDataGenerator(rescale = 1./255,
                                  validation_split = 0.2,
                                  preprocessing_function=new_preprocess)
```

Figure 5: Methods used for data augmentation

From figure 6 it is clear that in this model for data augmenting I have used horizontal flip, width shift, height shift, and re-scaling. As a face can appear from any direction these modules are kept on. All the modules for data augmentation are the built-in function in Keras.

- DenseNet169 for feature extraction and Classic CNN for classification.

```
# feature extractor model using DenseNet169
def dense_feature_extractor(inputs):
    dense_feature_extractor = tf.keras.applications.DenseNet169(input_shape=(img_height ,img_width, 3),
                                                                include_top=False,
                                                                weights="imagenet")(inputs)

    return dense_feature_extractor
```

Figure 6: Feature Extraction using DenseNet

For extracting features, DenseNet169 is used and this is pre-trained on imagenet weights. For this extractor, the input is the augmented training data.

- Next Classic CNN is used for the classification where the dense layer is used for this purpose. In CNN the Convolutional layers are used for feature extracting and the

dense layer is used for classification. So for the classification part, the only dense layer is used. Figure?? shows the code for the classification part. Here, in the first three layers, ReLU activation function is used and in the output layer SoftMax activation function is used.

```
# classifier model using CNN
def cnn_classifier(inputs):
    x = tf.keras.layers.GlobalAveragePooling2D()(inputs)
    x = tf.keras.layers.Dense(256, activation="relu", kernel_regularizer = tf.keras.regularizers.l2(0.01))(x)
    x = tf.keras.layers.Dropout(0.3)(x)
    x = tf.keras.layers.Dense(1024, activation="relu", kernel_regularizer = tf.keras.regularizers.l2(0.01))(x)
    x = tf.keras.layers.Dropout(0.5)(x)
    x = tf.keras.layers.Dense(512, activation="relu", kernel_regularizer = tf.keras.regularizers.l2(0.01))(x)
    x = tf.keras.layers.Dropout(0.5) (x)
    x = tf.keras.layers.Dense(number_class, activation="softmax", name="classification")(x)

    return x
```

Figure 7: Classification Using Dense Layer

- For evaluation, this project will use validation set accuracy and loss. Accuracy is the quintessential classification metric. It is very easy to understand and for my model, I choose this metric to understand the result more clearly. The equation that uses to find out the accuracy of any model is based on the predicted results and ground truth,

$$Accuracy = \frac{TP + TN}{TP + FP + FN + TN} \tag{1}$$

TP represents the number of true positives, and FP represents the number of false positives. FN represents the number of false negatives. And finally, TN represents the true negative. This is one of the best strategies for a model to evaluate with a relatively small dataset.

5.2 Training

For the training of the model I have used the adam optimizer, and categorical cross-entropy loss and for evaluation, I have chosen the accuracy metric. Figure 8 shows the function that I have used for compiling the model.

```
# complaining model
def define_compile_model():

    inputs = tf.keras.layers.Input(shape=(img_height ,img_width,3))
    classification_output = final_model(inputs)
    model = tf.keras.Model(inputs=inputs, outputs = classification_output)

    model.compile(optimizer='adam',
                  loss='categorical_crossentropy',
                  metrics = ['accuracy'])

    return model
```

Figure 8: Compilation of the model

For training, I choose epoch 50 without any early stopping function. And at the end of the training, I save the model as ḣ5 file.

```
Epoch 49/50
707/707 [==============================] - 587s 830ms/step - loss: 0.8857 - accuracy: 0.7188 - val_loss: 1.0805 - val_accurac
y: 0.6411
Epoch 50/50
707/707 [==============================] - 587s 830ms/step - loss: 0.8885 - accuracy: 0.7206 - val_loss: 1.1730 - val_accurac
y: 0.6289
```

Figure 9: Training of the Model

6 Experiment and Result

For the experiment purpose, I have built three models to check which model gives a more precise result.

- Case 1: Classic CNN where the Convolutional layer is used for feature extraction and the dense layer is used for classification. There were a total of 7 layers in total with 5 hidden layers.
- Case 2: In case 2, the feature extractor part is replaced by DenseNet. An early stopping function is used on top of that where it monitored the validation loss with patience 6. the epochs stopped at 11.
- Case 3: For case 3 the early stopping function is removed from case 2. Other parameters remain the same.
- Case 4: Same as case 3 but with only three classes.

Table 1: Comparison between different cases

Number of Case	Accuracy
1	25%
2	63%
3	72%
4	93%

Table1 shows the comparison between the three models. And it is clear that case 3 produces the best result and case 1 the worst.

7 Discussion

To understand the result more deeply I have generated the precision, recall, and f1 score for all the classes. Firstly, I checked which number represents which class, and figure **??** shows that.

```
# printing label for each class
test_generator.class_indices
```

```
{'angry': 0, 'fearful': 1, 'happy': 2, 'neutral': 3, 'sad': 4, 'surprised': 5}
```

Figure 10: Class name along with class number

Now by generating the precision, recall, and F-1 score the more can be more accurately evaluated. class 2 which is 'happy' emotion produces the best classification accuracy of all other classes. With all this, the accuracy curve of the model is also displayed with all the classes.

	precision	recall	f1-score	support
0	0.50	0.62	0.56	958
1	0.51	0.35	0.41	1024
2	0.82	0.87	0.84	1774
3	0.51	0.66	0.58	1233
4	0.54	0.44	0.48	1247
5	0.80	0.71	0.75	831
accuracy			0.63	7067
macro avg	0.61	0.61	0.60	7067
weighted avg	0.63	0.63	0.62	7067

Figure 11: Evaluation score of each class

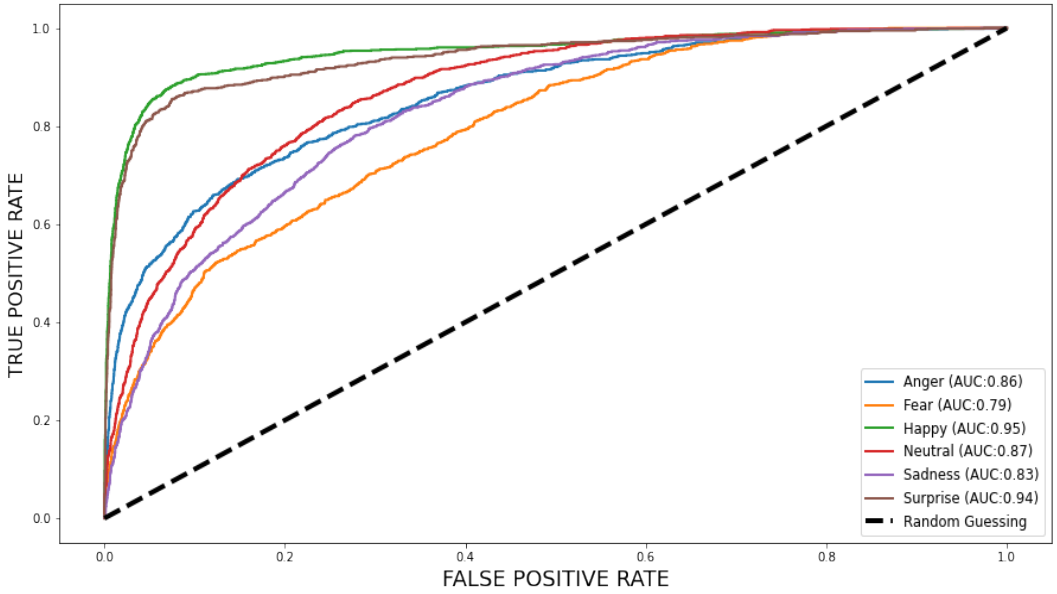

Figure 12: Accuracy graph of all class

7.1 Result

As the accuracy of different classes is different so is the prediction. In the case of prediction this model's accuracy is around 50% which means this model predicts three classes prediction

accurately(happy, fear, and sad) and the other three classes' prediction is wrong most of the time. The next two figures will show correctly recognise emotion and wrongly recognized emotion.

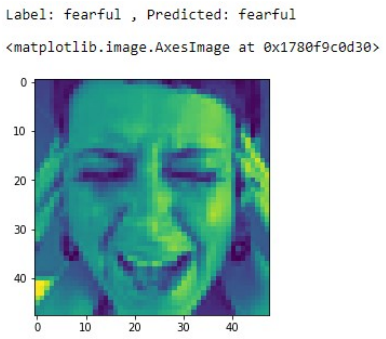

Figure 13: Recognition of 'fearful' emotion

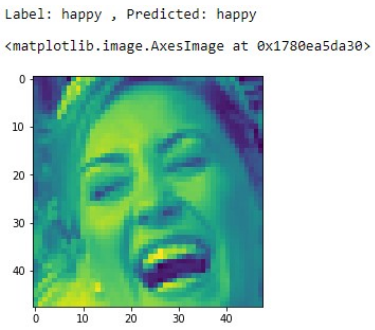

Figure 14: Recognition of 'happy' emotion

Figure 15: Recognition of 'sadness' emotion

Figure 16: predicted result

8 Conclusion

Though case 4 produces best result among all the experiments but I choose to go with case 3 as it has more classes and more emotions. The overall accuracy of this model is 72%. The accuracy is not that high but the number of classes is more. Moreover, using transfer learning this model was able to cut the training time while extracting more features. Last but not least this project give a comparison of classic CNN and transfer learning and their ability to extract feature with more and more layers.

9 Future work

In future, this project can be fine tuned more to improve the accuracy. Even different pre-trained model like VGG16 can be used to extract feature. Number of class can be used by training this model with different dataset of same manner. Moreover, the evaluation matrix can be changed to IOU matrix to estimate the accuracy of region of interest.

References

[1] A. Jaiswal, A. K. Raju, and S. Deb, "Facial emotion detection using deep learning," in *2020 International Conference for Emerging Technology (INCET)*, pp. 1–5, IEEE, 2020.

[2] P. Lucey, J. F. Cohn, T. Kanade, J. Saragih, Z. Ambadar, and I. Matthews, "The extended cohn-kanade dataset (ck+): A complete dataset for action unit and emotion-specified expression," in *2010 ieee computer society conference on computer vision and pattern recognition-workshops*, pp. 94–101, IEE, 2010.

[3] A. F. Agarap, "Deep learning using rectified linear units (relu)," *arXiv preprint arXiv:1803.08375*, 2018.

[4] M. Wang, S. Lu, D. Zhu, J. Lin, and Z. Wang, "A high-speed and low-complexity architecture for softmax function in deep learning," in *2018 IEEE asia pacific conference on circuits and systems (APCCAS)*, pp. 223–226, IEEE, 2018.

[5] S.-C. Ma, W. Chou, T.-W. Chien, J. C. Chow, Y.-T. Yeh, P.-H. Chou, H.-F. Lee, *et al.*, "An app for detecting bullying of nurses using convolutional neural networks and web-based computerized adaptive testing: development and usability study," *JMIR mHealth and uHealth*, vol. 8, no. 5, p. e16747, 2020.