National University of Computer & Emerging Sciences (FAST-NUCES)

Occluded Facial Expression Detection

Project Supervisor
Ms. Javeria Farooq
Project Co-Supervisor
Ms. Syeda Rubab Jaffar

Project Team

Abdul Mannan	**16K-3620**
Murtaza Multanwala	**16K-3618**
Muhammad Shehryar	**16K-3750**

Submitted in the partial fulfilment of the requirements for the degree of Bachelor of Science

Submitted on: 06/11, 2020

Department of Computer Science
National University of Computer & Emerging Sciences (FAST-NU)
Main Campus, Karachi
JUNE 2020

Project Supervisor	**Ms. Javeria Farooq**
Project Team	**16K-3620, 16K-3618, 16K-3750**
Submission Date	**11th, June 2020**

Supervisor Ms. Javeria Farooq

Head of Department

Dr. Muhammad Atif Tahir

Department of Computer Science

National University of Computer & Emerging Sciences (FAST-NU)

Main Campus, Karachi

JUNE 2020

ACKNOWLEDGEMENTS

It is indeed with a heart full of gratitude and joy that I would like to acknowledge the efforts of some people, without whom this accomplishment would not have been possible. First and foremost I would like to extend my profound gratitude to my Project Supervisor for her unwavering support and undeniable contribution to this project. The way she incredibly guided us through the journey was indeed at par with a true mentor. We owe her our success for sharing her pearls of wisdom with us during the course of this research. The supportive role played by our Co-Supervisor also integrated our efforts with her fine wisdom and guidance and made this come to shape. We thank our supervisors for the insight and expertise that they provided, which greatly assisted the research.

I am immensely grateful to FAST management for cooperating with us and facilitating us to the best of degree with the management and procurement of Data samples. I would also like to appreciate the Jury for their thoughtful analysis and healthy scrutiny of the research.

Last but not the least, special mentions for Sarmad Ali, Senior Business Analyst at Daraz, and Hassaan Malik, Project Manager at Olive Digital, for sharing ideas about target market and business application.

Abstract

Recognizing Facial Expressions is a challenging task and is a constant attraction for several researchers. Several researches have been performed in order to improve its accuracy. It has many applications and is being used in multiple fields nowadays such as from getting students feedback in classrooms to testing candidate's personality during an interview. The proposed research focuses on detection of occluded/non occluded facial expressions. The target of research work is to find out the best approach to detect and predict the occluded/non occluded facial expressions of multiple humans in sparse crowds. At first, faces are recognized from the frames, and afterward, expression classification is performed on them. In this research, different existing approaches of expression classification were studied. The first approach is to train the model beginning from the underlying layers (i.e. the Scratch approach) whereas the second approach is to train the model using transfer learning technique. The dataset utilized for training of the expression classification model is the FER2013 dataset provided by kaggle. Continuous experiments resulted in achieving 66% accuracy using the first model(Scratch approach) and 71% accuracy using the second model(Transfer learning approach). Models have been tested on a video feed obtained from a classroom to detect and predict the facial expressions of multiple students.

Table of Contents

1 Introduction

Recognition of facial expression has been researched by psychologists and computer scientists over the recent decades, as it is grasp promise to majority of applications, which includes behavior analysis, mental health analysis and human computer interaction. Although numerous facial expression identification methods have been proposed and built, majority of them are built in controlled surroundings. Mostly, the research is either on frontal faces or either without occlusion. Despite of numerous researches [1, 2] on different datasets, recognition of facial expression is still challenging because of partially occluded faces. It is not possible to detect the occluded facial expressions due to variation in human poses. The occlusion may occur due to hands, foods, scarf, glasses, food and additional objects that may block the faces in daily life. These objects can block any part of the face such as, eyes, part of cheek, mouth. The variation in occlusions due to different objects cannot completely covered by partial quantity of datasets and it will lead to the decrease in recognition accuracy.

To address the occlusion issue, we have proposed two different approaches:

1. Scratch method (Approach A): It proposes that the model has to be trained from initial levels and all the weights have to be adjusted sequentially.
2. Transfer learning method (Approach B): It proposes that a model that is trained to perform one task is finetuned to carry out your specific tasks.

The scratch approach requires much resources and time to train because all the weights have to be adjusted from initial stages. On the other hand, the transfer learning approach uses pretrained models to extract high level features from images enabling us to accurately predict facial expressions without experiencing the hectic details of training the entire model. In transfer learning approach, pretrained models are acquired such as VGG (Visual Geometry Group), Resnet, Inception-V3 or Mobilenet etc that are pretrained models of keras, each trained on imageNet. These models are themselves trained on millions of images for multi class classification. In transfer learning approach, we retrain some layers of the model to utilize it as we want and this saves a lot of time and resources, additionally, it gives us satisfying accuracy on the test sets.

Following any FER(Facial expression recognition) approach we need to hold a few constraints under observation which includes lighting variation, distance of subject from the camera, face not clearly visible hence difficult to detect correctly, and movement in position of the subject's face. Our main objective is to detect the occluded facial expressions. To recognize occluded faces, we have used MTCCN (Multi Task Cascaded Neural Network). Once we get the interest region(faces), they are passed to expression classification model to predict the expressions. Several existing FER approaches work on frontal faces. They have some limitations for example the face shall be clearly visible and free from occlusions. Furthermore they fail to detect slightly tilted faces.

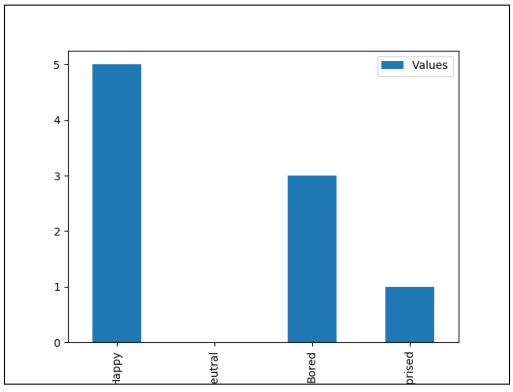

Figure 1: Graph obtained using prototype.

2 Related Study

Deep learning techniques have worked admirably in each field. It has gained much popularity in the field of facial expression recognition as well. The reason behind the success of deep learning techniques are their ability to perform feature engineering on it's own, due to which they can learn the most prominent and useful feature and patterns from the raw data. They are occasionally accurate enough that we don't have to perform occlusion detection or face reconstruction independently. Once provided with a face photograph, they can classify the expression easily. Many existing approaches attempt to conquer occlusions on a singular frontal face and don't work in real-time. The gap that can be found in existing approaches under occlusions is that majority of the methodologies doesn't work in real time and work on a single face only.

Octavio [1] proposed a light weight model keeping in mind the hardware constraints. He achieved 66% testing accuracy on the FER2013 dataset. His architecture works on frontal faces in real time. He used efficiently the correct regularization methods to increase processing speed in real time. He deleted the fully connected layer and inserted a depth wise separable convolution layer and residual modules.

Schwan, Justus Ghaleb, Esam Hortal, Enrique Asteriadis, Stylianos [2] utilized transfer learning approach on VGG-Face network. This approach works on single frontal face in real time. They got greater than 90% accuracy on labelled faces in wild and youtube faces dataset but failed to perform well under occlusions.

Duncan, D., Shine, G., English [3] used transfer learning technique on VGG S model and were able to score accuracy of 90.7% and 57.1% on train and test set respectively on a custom built dataset that contains images from CK+, Jaffe and images of five individuals posing different expressions. It works on multiple faces. It uses HAAR for face detection and limits the classification of expression to frontal faces only.

Mao Xu [4] constructed a modified model for FER. It utilized the working of two models to predict the facial expressions in occluded images. They used the MSRA CFW database for training the models. To train and test the models on occlusions, additionally used the KDEF , JAFFE and the Pain expressions (PICS). The model combines distinctive features obtained from two seperate CNNs having the same structure but trained on different training data. MSRA-CFW is used in training of one CNN while the same dataset with occlusions is used in training the next CNN. They were able to achieve 81.50% accuracy on the customized database.

Xia [5] uses the transfer learning approach on Inceptionv3 model. He used CK+(Extended Cohn Kanade) dataset. For training the last layer, a backpropagation algorithm and cross entropy cost function was used. Only the last layer is trained keeping the remaining parameters unchanged which dramatically reduces the training time. Their model achieved 97% classification accuracy.

Enrique [6] proposed a model which is currently able to run on a single frontal face in real time. The model was trained on FER2013 dataset and the evaluation and testing was carried on combined samples of FER2013 and Radboud Faces Database (RaFD) and achieved 63% accuracy.

Minaee, Shervin Abdolrashidi, Amirali [7] introduced an attentional convolutional network which focuses on feature-rich parts of the face. They added an attention mechanism to focus on important face regions. They used CK+, FER2013, Facial Expression Research Group Database (FERG) and (JAFFE) database for training and testing purposes. Classification accuracies of proposed architecture are 70.02% on FER2013, 92.8% on jaffe, 98% on CK+ and 99.3% on FERG.

Farhad [8] used 3d face reconstruction. It requires a lot of computation power. It works on a single person and in real time. It uses a BU3dfe dataset with synthesized occlusions. They used viola-jones trained cascades for face detection in picture sequences. The HSV skin color detection is also used in case of false face detection that detects faces on the basis of skin color. At first, facial landmarks are robustly identified on the face in the presence of occlusion. Secondly, the 3D pose of the face is identified. Then the feature vector with geometrical and texture features of the face are identified. Finally, the emotion is identified using a backpropagation neural network. These steps are repeated for each new frame in the sequence.

Zhao, X., Shi, X., Zhang, S. [9] uses a deep belief network. The datasets used are CK and JAFFE. This paper presented a new method for FER. They use a combination of MLP(Multi Layer Perceptron) and DBNs(Deep Belief Network). First they use DBN to extract some features from the image then those features act as an input for MLP.

MLP model performs facial expression classification. They obtained 98% on CK and 90.95% on JAFFE.

Georgescu [10] used synthetically occluded data to fine tune VGG-face and VGG-f. Li, Yong [11] proposed a CNN with attention mechanism also known as ACNN. ACNN's focus on the most important regions. These proposed ACNNs were evaluated on real and synthetically created occlusions on datasets namely FED-RO, RAF-DB and AffectNet.

Li Zhang [12] used MTCNN for the task of fruits detection. They used it because of its great and accurate real time performance so that it could be implemented in robots such as fruit picker. They modified the architecture of MTCNN by removing the landmark loss of the cascaded nets because MTCNN is made to detect faces but their task was to detect fruits. They were able to detect multiple fruits under occlusions with different lightening conditions and poses in real time. They used the technique of distinguishing fruits from the background which means it is a two class classification problem for which reason they used cross entropy loss.

Edmar Rezende [13] used ResNet50 for classifying malicious softwares. They used transfer learning on ResNet50. Gray scaled byteplot images are used to represent malwares. The first 49 layers of ResNet50 which are trained on imagenet dataset are freezed during training and only the last layer which is 1000 class fully connected layer is replaced with 25 class fully connected layer to adjust ResNet to perform classification on malware images. They achieved 98.62% accuracy on a dataset of 9,339 images of malwares from 25 different categories which means transfer learning was suitable for their task. They used categorical cross entropy loss function due to multi class classification.

Some of the popular datasets which are being used to train facial expression recognition models can be seen in figure 2:

- Japanese Female Facial Expression (JAFFE) database: It contains 213 images. This is a posed dataset by 10 japanese female subjects.
- Extended Cohn Kanade dataset (CK+): It contains 593 image sequences from a total of 123 subjects. It is a posed dataset.
- Fer2013: It contains 35,887 grayscale images. It is not a posed dataset and contains real life images.

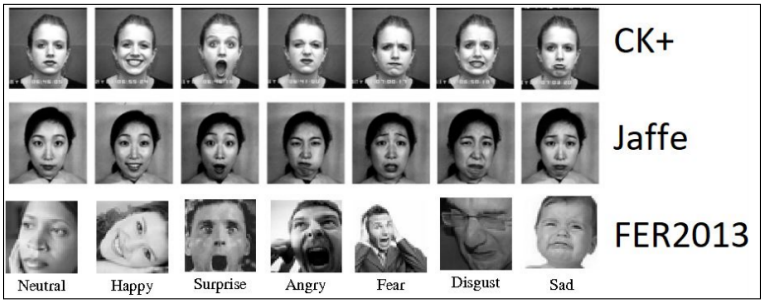

Figure 2: Sample Pictures of datasets.

3 Proposed Work

We propose a Multi Task Cascaded Convolution Neural Network (MTCNN) for occluded facial detection followed by expression classification model for expression classification with partial occlusion.

Figure 3 illustrates the architecture of the proposed approach. As can be seen in Fig. 3, the models are able to detect multiple faces from a distance and predict their expressions. The classroom images referred in the paper are taken from a tilted angle. The camera feed is obtained from a high quality camera. The frames are fed to MTCNN face detector which provides us with coordinates of faces. Each face is preprocessed and passed as an input to the expression classification model which predicts the facial expressions.

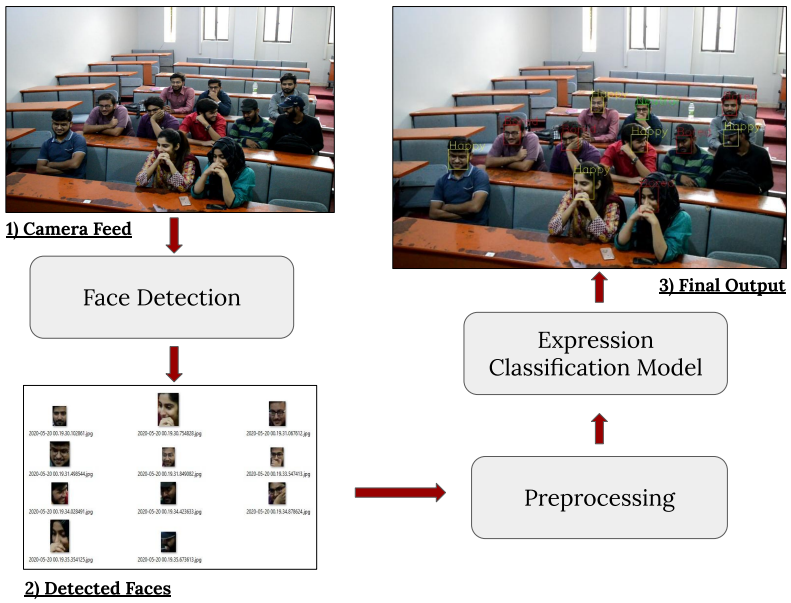

Figure 3: Work flow.

3.1 Face Detection

To achieve our objectives, an accurate and fast face detector was required. Most FER approaches use HAAR face detector for face detection. It is a lightweight detector used to detect faces in an image or video. It was proposed by Paul Viola and Michael Jones. The problem with HAAR is that it works on frontal faces and not on occluded or slightly tilted faces which is the main objective of our research.

For detecting occluded faces, we have used Multi Task Cascaded Convolutional Neural Network(MTCNN) [14]. It's performance is better than HAAR. It works on multiple faces in real time providing us with accurate results. MTCNN has three convolutional neural networks which provides us with accurate bounding boxes for the faces. This covers our required target i.e. occluded facial expression detection. The architecture can be seen in figure 4.

The frame is passed through the P-Net. It is the initial network and provides us with many bounding boxes for a single face. It has only three layers which are used for feature extraction and these features further act as input for the cascaded networks. To further reduce the bounding boxes, non maximum suppression is used which deletes the boxes with low confidences. P-Net's output becomes input of R-Net which further refines the bounding boxes and passes them to the final net i.e. O-Net. R-Net and O-Net both performs padding of out of bound boxes which helps to detect faces that are at the corner of frame. This network is the last stage and it minimizes the number of boxes to one box per face. It outputs a dictionary that contains coordinates of each face and also their facial landmarks. The changes in size of the cascaded nets is called as a pyramid architecture. Having three networks boosts the results as each improvises the detection. Once the face is obtained from the frame, it is passed to the expression classification model.

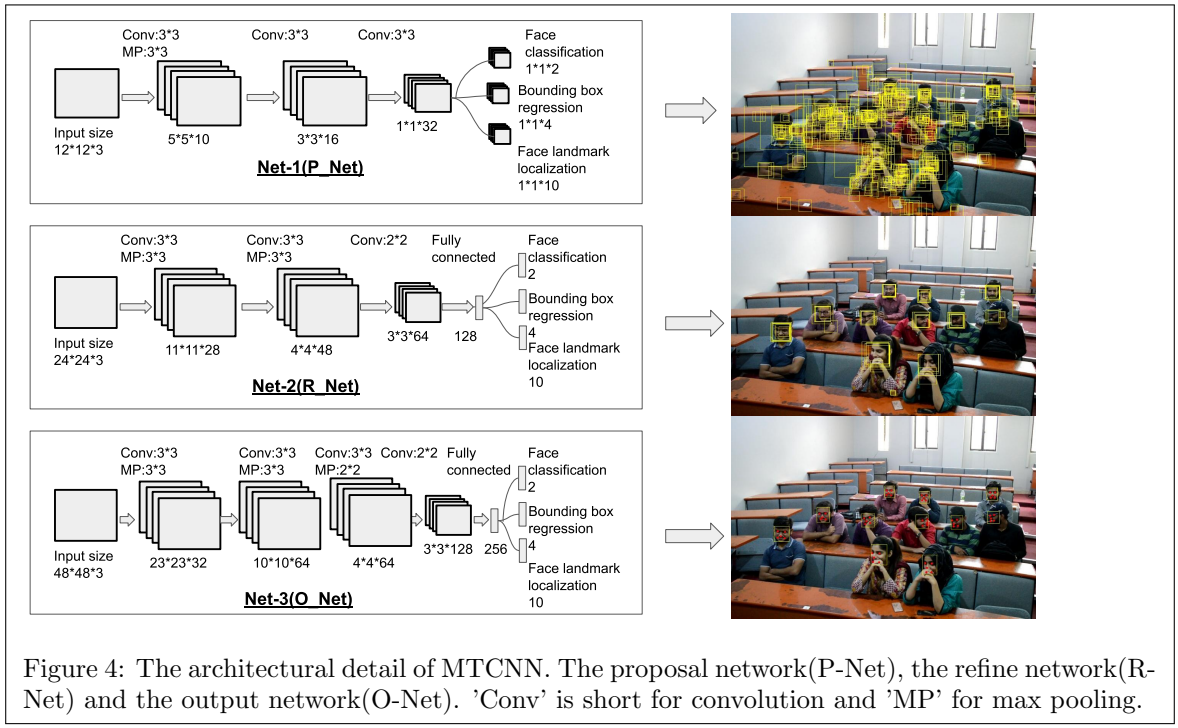

Figure 4: The architectural detail of MTCNN. The proposal network(P-Net), the refine network(R-Net) and the output network(O-Net). 'Conv' is short for convolution and 'MP' for max pooling.

3.2 Expression classification

Two different approaches were used to train the facial expression recognition model. The objective was to find out the best methodology in terms of accuracy. When the bounding boxes of faces are obtained, they are fed to the expression classification model for prediction of emotions. The predicted labels are shown along side each face in the frame as shown in figure 10.

3.2.1 Approach A model summary

The model architecture summary of approach 'A' is shown in figure 6. FER2013 dataset was used for training of the convolutional neural net, keeping the batch size to be 64, epochs to be 100 and the dimensions set to be (48x48) which is the actual size of FER2013. The dataset is distributed as 80%, 10% and 10% for training, validation and testing set respectively using hand-out approach. The loss function selected is categorical cross entropy which helps in calculating the loss by differentiating the distribution of the predicted output with the actual label. The categorical labels are changed over into the unique vector called a one-hot encoded vector which is then compared with the predicted result of the expression classification model that is likewise a vector and afterward the loss is calculated. In Multi-Class classification the labels are one-hot, so only the positive class keeps its term in the loss. For optimization, adam optimizer used. It is a well known optimizer. It ensures the benefits of both AdaGrad and RMSProp. Figure 6 shows the model summary of the model for approach 'A'. Figure 5 shows an output of a layer's feature map obtained using approach 'A' architecture.

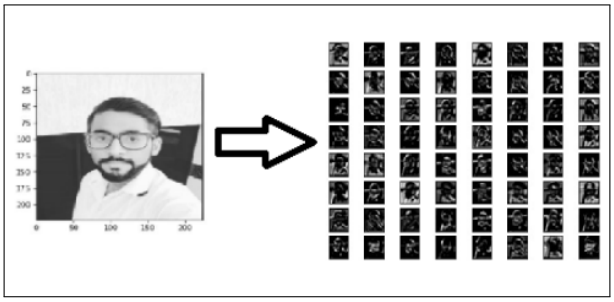

Figure 5: Visualizing feature map obtained by Approach A architecture.

Model: "sequential_1"

Layer (type)	Output Shape	Param #
conv2d_1 (Conv2D)	(None, 46, 46, 64)	640
conv2d_2 (Conv2D)	(None, 46, 46, 64)	36928
batch_normalization_1 (Batch	(None, 46, 46, 64)	256
max_pooling2d_1 (MaxPooling2	(None, 23, 23, 64)	0
dropout_1 (Dropout)	(None, 23, 23, 64)	0
conv2d_3 (Conv2D)	(None, 23, 23, 128)	73856
batch_normalization_2 (Batch	(None, 23, 23, 128)	512
conv2d_4 (Conv2D)	(None, 23, 23, 128)	147584
batch_normalization_3 (Batch	(None, 23, 23, 128)	512
max_pooling2d_2 (MaxPooling2	(None, 11, 11, 128)	0
dropout_2 (Dropout)	(None, 11, 11, 128)	0
conv2d_5 (Conv2D)	(None, 11, 11, 256)	295168
batch_normalization_4 (Batch	(None, 11, 11, 256)	1024
conv2d_6 (Conv2D)	(None, 11, 11, 256)	590080
batch_normalization_5 (Batch	(None, 11, 11, 256)	1024
max_pooling2d_3 (MaxPooling2	(None, 5, 5, 256)	0
dropout_3 (Dropout)	(None, 5, 5, 256)	0
conv2d_7 (Conv2D)	(None, 5, 5, 512)	1180160
batch_normalization_6 (Batch	(None, 5, 5, 512)	2048
conv2d_8 (Conv2D)	(None, 5, 5, 512)	2359808
batch_normalization_7 (Batch	(None, 5, 5, 512)	2048
max_pooling2d_4 (MaxPooling2	(None, 2, 2, 512)	0
dropout_4 (Dropout)	(None, 2, 2, 512)	0
flatten_1 (Flatten)	(None, 2048)	0
dense_1 (Dense)	(None, 512)	1049088
dropout_5 (Dropout)	(None, 512)	0
dense_2 (Dense)	(None, 256)	131328
dropout_6 (Dropout)	(None, 256)	0
dense_3 (Dense)	(None, 128)	32896
dropout_7 (Dropout)	(None, 128)	0
dense_4 (Dense)	(None, 7)	903

Total params: 5,905,863
Trainable params: 5,902,151
Non-trainable params: 3,712

Figure 6: Approach A model summary.

3.2.2 Approach B model summary

The architecture used to implement approach 'B' is ResNet50. Residual Network (ResNet) has a special feature named as skip connection which helps the model to be deeper and train faster without decreasing the model performance and accuracy. It is trained on imagenet. It contains 50 layers and can classify images into thousand different classes. Resnet originally is trained on imagenet so the top layer is excluded. The initial layers provides us with bottleneck features using which we can perform classification of expressions. The input shape is adjusted to be (197x197x3) that works on a rgb color channel. The weights used are of VGGFace. Initially, just the top layers are trained and the model is compiled then the model is recompiled for finetuning. This is done because ResNet is initially trained to perform classification on imageNet dataset and classifies objects into one of 1000 classes. Our task is to detect facial expressions so we have to replace that layer with one that categorizes object into required emotions as can be seen in table 2. Generally, the deep convolutional neural networks are sequential which means several layers are stacked on each other and as we proceed from the initial layers to the end of the model, it learns high-level features. Resnet implements residual learning. In simple terms, Residual can be thought of as subtraction of features which are learned from input of the layer. To achieve this, we use residual block as shown in figure 7 i.e. connecting input of ith layer to (i+x) th layer. Due to its well structured architecture, we get higher accuracies. Figure 8 shows an output of a layer's feature map obtained using approach 'B' architecture. It performs well for our proposed target i.e. to detect occluded facial expressions of multiple subjects in real time. The parameters used for training the model are discussed in next section.

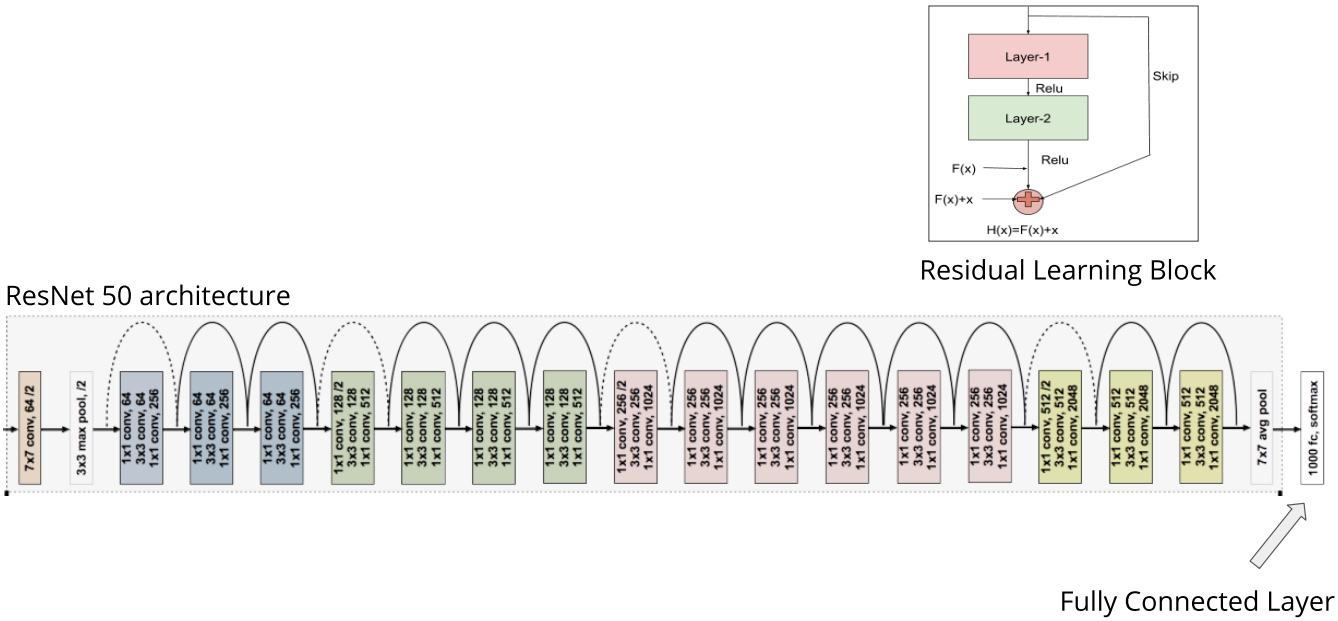

Figure 7: ResNet50 architecture.

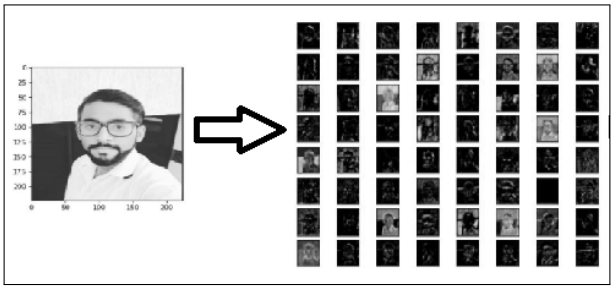

Figure 8: Visualizing feature map obtained by ResNet50.

3.2.3 Dataset

FER2013 contains 35,887 grayscaled images which are of size 48x48 pixels. The dataset is divided into seven emotions and the distribution can be seen in table 2. It is not a posed dataset and contains real life images. It covers pose variations and occlusions which is target of our study. It is publicly available on kaggle. Few of the images from the dataset is shown in figure 9. For simplicity, the emotions were mapped to four general emotions and can be seen in table 1.

Emotion	**Prototype label**
Happy	Happy
Neutral	Neutral
Fear, Surprise	Surprised
Angry, Sad, Disgust	Bored

Table 1: List of facial expressions in prototype.

Number of images	**Emotion**	**Label**
4593	Angry	0
547	Disgust	1
5121	Fear	2
8989	Happy	3
6077	Sad	4
4002	Surprise	5
6198	Neutral	6

Table 2: Dataset distribution.

Figure 9: FER2013 dataset.

Figure 10 shows the step by step image processing of prototype.

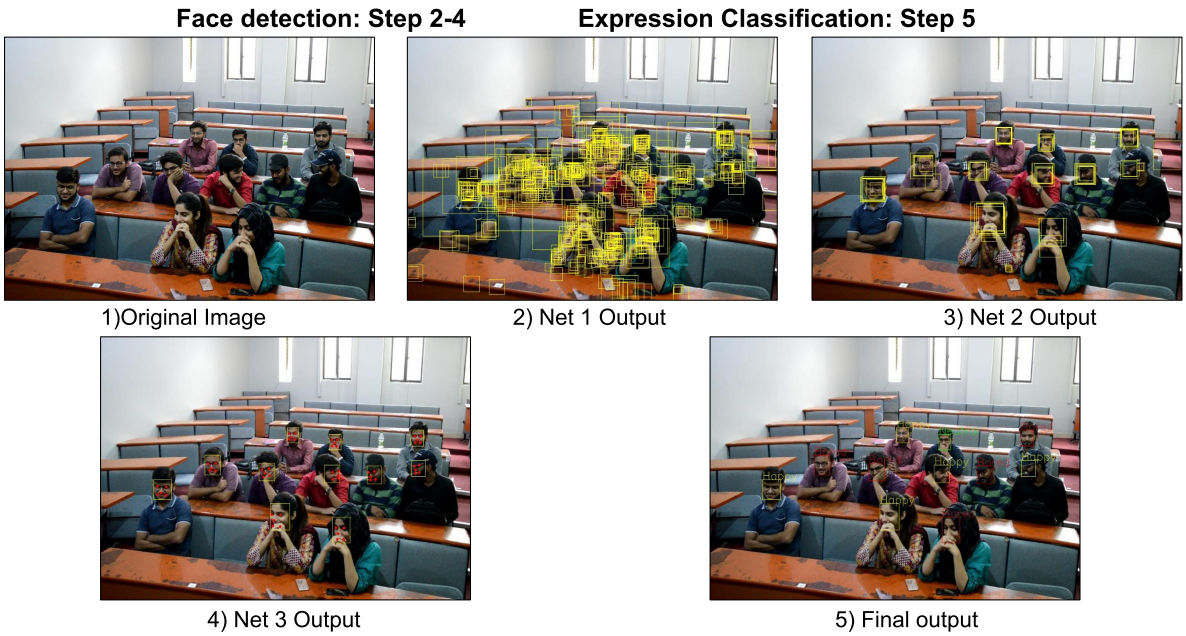

Figure 10: Working of occluded facial expression detection prototype.

4 Experimental Results

The primary focus of this study was on occluded facial expression detection of multiple faces in sparse crowd. The results mentioned in the paper are achieved using following parameters:

4.1 Loss Function

The loss function selected for the expression classification task is categorical cross entropy which is used for multi class classification. It is used to classify each sample to a single class by minimizing the difference between the probability distribution of predicted and target label.

$$-\sum_{c=1}^{M} y_{o,c} \log(p_{o,c})$$

Figure 11: Loss Function, where M is number of classes, p is predicted probability, y is the binary indicator 0 or 1 and c is class label

4.2 Optimizer

The optimizers used are adam and stochastic gradient descent. The momentum of stochastic GD is 0.9. Adam guarantees the benefits of both AdaGrad and RMSProp.

4.2.1 Learning Rate

The learning rate of adam is kept as 0.001 where as the learning rate of SGD is kept to be 0.0001. It is used to define the step size towards reaching the minimum of loss function.

4.2.2 Beta

The parameters beta1 and beta2 of adam optimizer are set to be 0.9 and 0.99 because it is good practice to keep it close to 1. Beta1 and beta2 are used to decay the running average of the gradient and square of the gradient respectively.

4.2.3 Epsilon

It is a parameter of adam optimizer used to prevent division by zero. It is set to be 1e-07 and 1e-08 in approach 'A' and 'B' respectively.

The top accuracies we achieved were 66% and 71% for approach A(Training CNN from scratch) and B(Transfer learning on ResNet50) respectively. Approach 'B' provides us with the best accuracy on FER2013. Approach 'A' provide us with good accuracy on test sets but they fail to perform well under real time scenarios. The feature extraction of transfer learning approach is much better than that of scratch approach and this is the reason that although the variation in testing accuracy is not much but the difference in real time is noticeable as can be seen in figure 12 and 13. Approach 'A' requires careful architectural implementation with proper adjustment of parameters. If the architecture is not properly adjusted, there is possibility of over fitting or under fitting. Although models used in transfer learning have a higher number of parameters than approach 'A', it takes less time to train, because we don't train all the weights in transfer learning method. The excessive amount of parameters in transfer learning techniques result in higher testing accuracies. We have selected approach 'B' model for our prototype.

Figure 12: Output of approach 'A' model.

Figure 13: Output of approach 'B' model.

5 Conclusion

The primary focus of this paper is on occluded facial expression recognition in sparse crowd video. For this purpose, We have opted for MTCNN for the task of occluded facial detection. We have chosen MTCNN because it runs accurately in real time which was our main objective, and doesn't lag like the other heavy weight models. For expression classification models we have followed two approaches 'A' and 'B', in which we train the model from scratch or use a pretrained model and fine tune it according to our task respectively. After conducting a detailed literature review and conducting several experiments, we conclude that transfer learning(Approach B) performs well in predicting the expressions. Also they take less time in training as compared with approach A. Transfer learning models are heavy, however when tuned properly, they provide good accuracy for the task of occluded facial expressions recognition. We achieved 66% and 71% accuracy on the FER2013 dataset using approaches A and B respectively. Our future work will be to make much accurate occluded facial expression detection.

References

[1] Octavio Arriaga, Matias Valdenegro-Toro, and Paul Plöger. "Real-time convolutional neural networks for emotion and gender classification". In: *arXiv preprint arXiv:1710.07557* (2017).

[2] Justus Schwan et al. "High-performance and lightweight real-time deep face emotion recognition". In: *2017 12th International Workshop on Semantic and Social Media Adaptation and Personalization (SMAP)*. IEEE. 2017, pp. 76–79.

[3] Dan Duncan, Gautam Shine, and Chris English. "Facial emotion recognition in real time". In: *Stanford University* (2016).

[4] Mao Xu et al. "Facial expression recognition based on transfer learning from deep convolutional networks". In: *2015 11th International Conference on Natural Computation (ICNC)*. IEEE. 2015, pp. 702–708.

[5] Xiao-Ling Xia, Cui Xu, and Bing Nan. "Facial expression recognition based on tensorflow platform". In: *ITM Web of Conferences*. Vol. 12. EDP Sciences. 2017, p. 01005.

[6] Enrique Correa et al. "Emotion recognition using deep convolutional neural networks". In: *Tech. Report IN4015* (2016).

[7] Shervin Minaee and Amirali Abdolrashidi. "Deep-emotion: Facial expression recognition using attentional convolutional network". In: *arXiv preprint arXiv:1902.01019* (2019).

[8] Farhad Goodarzi et al. "Real time facial expression recognition in the presence of rotation and partial occlusions". In: *Journal of Theoretical and Applied Information Technology* 96 (Feb. 2018), pp. 854–864.

[9] Xiaoming Zhao, Xugan Shi, and Shiqing Zhang. "Facial Expression Recognition via Deep Learning". In: *IETE Technical Review* 32 (Mar. 2015), pp. 1–9. DOI: 10.1080/02564602.2015.1017542.

[10] Mariana-Iuliana Georgescu and Radu Tudor Ionescu. "Recognizing Facial Expressions of Occluded Faces Using Convolutional Neural Networks". In: *International Conference on Neural Information Processing*. Springer. 2019, pp. 645–653.

[11] Yong Li et al. "Occlusion aware facial expression recognition using CNN with attention mechanism". In: *IEEE Transactions on Image Processing* 28.5 (2018), pp. 2439–2450.

[12] Li Zhang et al. "Multi-task cascaded convolutional networks based intelligent fruit detection for designing automated robot". In: *IEEE Access* 7 (2019), pp. 56028–56038.

[13] Edmar Rezende et al. "Malicious software classification using transfer learning of resnet-50 deep neural network". In: *2017 16th IEEE International Conference on Machine Learning and Applications (ICMLA)*. IEEE. 2017, pp. 1011–1014.

[14] Kaipeng Zhang et al. "Joint face detection and alignment using multitask cascaded convolutional networks". In: *IEEE Signal Processing Letters* 23.10 (2016), pp. 1499–1503.

1 Source Code

1.1 Approach 'A' model architecture

```
# Sequential Keras model
model_approachA= Sequential()

model_approachA.add(Conv2D(64, kernel_size=(3, 3), activation='relu', input_shape=(48, 48, 1),
                 data_format='channels_last', kernel_regularizer=l2(0.01)))

model_approachA.add(Conv2D(64, kernel_size=(3, 3), activation='relu', padding='same'))
model_approachA.add(BatchNormalization())

model_approachA.add(MaxPooling2D(pool_size=(2, 2), strides=(2, 2)))
model_approachA.add(Dropout(0.5))

model_approachA.add(Conv2D(128, kernel_size=(3, 3), activation='relu', padding='same'))
model_approachA.add(BatchNormalization())

model_approachA.add(Conv2D(128, kernel_size=(3, 3), activation='relu', padding='same'))
model_approachA.add(BatchNormalization())

model_approachA.add(MaxPooling2D(pool_size=(2, 2), strides=(2, 2)))
model_approachA.add(Dropout(0.5))

model_approachA.add(Conv2D(256, kernel_size=(3, 3), activation='relu', padding='same'))
model_approachA.add(BatchNormalization())

model_approachA.add(Conv2D(256, kernel_size=(3, 3), activation='relu', padding='same'))
model_approachA.add(BatchNormalization())

model_approachA.add(MaxPooling2D(pool_size=(2, 2), strides=(2, 2)))
model_approachA.add(Dropout(0.5))

model_approachA.add(Conv2D(512, kernel_size=(3, 3), activation='relu', padding='same'))
model_approachA.add(BatchNormalization())

model_approachA.add(Conv2D(512, kernel_size=(3, 3), activation='relu', padding='same'))
model_approachA.add(BatchNormalization())

model_approachA.add(MaxPooling2D(pool_size=(2, 2), strides=(2, 2)))
model_approachA.add(Dropout(0.5))

# Flattening Layers for Predicting Expressions.
model_approachA.add(Flatten())

# Adding set of Fully Connected Layers
model_approachA.add(Dense(512, activation='relu'))
model_approachA.add(Dropout(0.4))

model_approachA.add(Dense(256, activation='relu'))
model_approachA.add(Dropout(0.4))

model_approachA.add(Dense(128, activation='relu'))
model_approachA.add(Dropout(0.5))

# '7' units because of facial expressions
```

```
model_approachA.add(Dense(7, activation='softmax'))

model_approachA.summary()
```

1.2 Prototype code

```
#Necessary Imports to run the prototype
import cv2
import numpy as np
from time import sleep
from keras.models import load_model
from mtcnn.mtcnn import MTCNN
import pandas as pd
import matplotlib.pyplot as plt

image_width, image_height = 197, 197

# Labels
emotions = ['Anger', 'Disgust', 'Fear', 'Happiness', 'Sadness', 'Surprise', 'Neutral']

# The emotion map of prototype used in feedback graphs
EmotionList=["Happy","Neutral","Bored","Surprised"]

# Making dictionary for maintaining emotion count for feedback chart.
EmotionNames = dict.fromkeys(EmotionList, 0)

# Loading the saved model using keras
model_approachB = load_model('./ResNet-50.h5')

# Initializing MTCNN face detector
face_detector = MTCNN()

"""
Initilizing the camera feed
-> cv2.VideoCapture(1) for external webcam
-> cv2.VideoCapture(0) for internal camera
"""
live_feed = cv2.VideoCapture(1)

# Function to pre-process the face to be passed as input to the expression

# classification model
def preprocess_input(image):
    # Resizing images for the trained model
    image = cv2.resize(image, (image_width, image_height))
    ret_Image = np.empty((image_height, image_width, 3))
    ret_Image[:, :, 0] = image
    ret_Image[:, :, 1] = image
    ret_Image[:, :, 2] = image

    x = np.expand_dims(ret_Image, axis = 0)

    # Normalizing
    x -= 128.8006  # Mean of training set
    x /= 64.6497   # Standard deviation of training set
    return x
```

```
# Function to predict the emotion of face
def predict(emotion):
    prediction = model_approachB.predict(emotion)
    return prediction

# Opening Camera Feed
while(1):
    if not live_feed.isOpened():
        print('Error in opening camera')

        # wait for few seconds
        sleep(5)

    else:
        # Capturing frame by frame from the camera
        ret, feeded_frame = live_feed.read()

        # Convert frame to gray scale
        gray_frame = cv2.cvtColor(feeded_frame, cv2.COLOR_BGR2GRAY)
        rgb_frame = cv2.cvtColor(feeded_frame, cv2.COLOR_BGR2RGB)

        # Passing the frame to detector to detect faces
        bboxes = face_detector.detect_faces(rgb_frame)

        # Initializing variables
        prediction = None
        cord_x, cord_y = None, None

        # Looping through all the detected faces
        if len(bboxes) != 0:
            for i in bboxes:

                #Selecting each face one by one
                bboxes= list(i['box'])

                #Extracting face coordinates from the dictionary returned by MTCNN
                face_coordinates = int(bboxes[0]), int(bboxes[1]), int(bboxes[2]), int(bboxes[

                #Storing face coordinates into variables
                cord_x, cord_y, width, height = face_coordinates

                #Extracting faces from frames
                face_detected_gray_scale = gray_frame[cord_y: cord_y + height, cord_x: cord_x

                """
                Preprocessing the face for making it suitable for expression
                classification model
                """
                emotion = preprocess_input(face_detected_gray_scale)

                #Predicting the facial expression
                prediction = predict(emotion)

                #Getting the prediction with highest probability
                top_1_prediction = emotions[np.argmax(prediction)]

                #Mapping the labels to 4 general emotions
```

```
                if(top_1_prediction.__contains__("Happiness")):

                    #Making a bounding box on face
                    cv2.rectangle(rgb_frame, (cord_x, cord_y), (cord_x+width, cord_y+height),

                    #Putting the label of emotion
                    cv2.putText(rgb_frame,"Happy",(cord_x, cord_y),cv2.FONT_HERSHEY_SIMPLEX,
                                1, (255, 255, 0),1)
                    #Maintaining a dictionary containing emotion count
                    EmotionNames["Happy"]+=1

                if(top_1_prediction.__contains__("Anger") or
                   top_1_prediction.__contains__("Sadness") or
                   top_1_prediction.__contains__("Disgust")):
                    cv2.rectangle(rgb_frame, (cord_x, cord_y), (cord_x+width, cord_y+height),
                    cv2.putText(rgb_frame,"Sad",(cord_x, cord_y),cv2.FONT_HERSHEY_SIMPLEX,
                                1, (255, 0, 0),1)
                    EmotionNames["Bored"]+=1

                if(top_1_prediction.__contains__("Neutral")):
                    cv2.rectangle(rgb_frame, (cord_x, cord_y), (cord_x+width, cord_y+height),
                    cv2.putText(rgb_frame,"Neutral",(cord_x, cord_y),cv2.FONT_HERSHEY_SIMPLEX,
                                1, (0, 255, 0),1)
                    EmotionNames["Neutral"]+=1

                if(top_1_prediction.__contains__("Fear") or
                   top_1_prediction.__contains__("Surprise")):
                    cv2.rectangle(rgb_frame, (cord_x, cord_y), (cord_x+width, cord_y+height),
                    cv2.putText(rgb_frame,"Surprised",(cord_x, cord_y), cv2.FONT_HERSHEY_SIMPL
                                1, (0, 255, 255),1)
                    EmotionNames["Surprised"]+=1

            # Display the modified frame
            modified_frame = cv2.resize(rgb_frame, (800, 500))
            modified_frame = cv2.cvtColor(modified_frame, cv2.COLOR_RGB2BGR)
            cv2.imshow('Video', modified_frame)

            # Closing the window
            if cv2.waitKey(1) & 0xFF == ord('q'):
                break
    # Making a feedback graph from the emotion dictionary
    EmotionGraphValues=pd.DataFrame({"Expressions": [*EmotionNames.keys()],
                                     "Values":  [*EmotionNames.values()]})
    EmotionGraphValues.plot(kind="bar", x="Expressions", y="Values")

    plt.savefig("outputLiveCam.png")

    # Printing dictionary
    print(EmotionNames)

    # Releasing the camera
    live_feed.release()

    cv2.destroyAllWindows()
```