Facial Expression Classification using Deep Learning

*Course No: EEE 402 Group No: 01

Kazi Ahmed Akbar Munim
ID: 1906006
EEE, BUET
Dhaka, Bangladesh
1906006@eee.buet.ac.bd

Sabir Mahmud
ID: 1906032
EEE, BUET
Dhaka, Bangladesh
1906032@eee.buet.ac.bd

Nafis Faisal
ID: 1906033
EEE, BUET
Dhaka, Bangladesh
1906033@eee.buet.ac.bd

Mohammad Nafis Kamal
ID: 1906061
EEE, BUET
Dhaka, Bangladesh
1906061@eee.buet.ac.bd

***Abstract*—Facial Expression Recognition is an important field of Deep Learning Research, with articles pouring in every year. Our project implements one such paper titled "Facial Expression Recognition Using Multi-Branch Attention Convolutional Neural Network" by Yinggang He. In this paper, a multibranch attention convolutional network having three branches, three modules called DSC-CBAM in each branch with additional depthwise separable convolution layers was trained on the FER2013 Dataset to obtain an accuracy of 69.49%. The CBAM adds a channel and spatial attention mechanism to the network while the depthwise separable convolution layers reduce the number of parameters. Our best implementation of the model trained on Kaggle FER2013 provided a test accuracy of 71.55%.**

***Index Terms*—CBAM, Facial Expression Recognition, Depthwise Separable Convolution, Deep Learning**

I. Introduction

For this project, we were instructed to select a mother paper with high value, that implemented Deep Learning Algorithm in a efficient way. Our group, selected an IEEE Access paper titled 'Facial Expression Recognition Using Multi-Branch Attention Convolutional Neural Network' by Yinggang He from Jimei University, China [1].

The paper we selected describes a methodology that develops a multi-branch attention CNN for facial expression recognition, which combines the Convolutional Block Attention Module, Depthwise separable convolution, and a three branch structure. The Convolutional Block Attention module was exploited to enable a better extracting of features from facial expression images giving proper attention to the appropriate channels and spatial regions, and depthwise separable convolution was added to reduce the model parameters. The author of the paper, used three different datasets to verify the effectiveness of the model. The three datasets are titled 'FER-PLUS', 'FER2013', and 'CK+'. The recognition accuracy was 84.633%, 69.49%, and 99.39% respectively using the model developed in the paper. Our primary target is to implement the mother paper we choose. All the three datasets mentioned previously, are available in Kaggle. We selected 'FER2013' dataset for our project.

A secondary paper that came with its corresponding code, titled "Convolutional Neural Network Hyperparameters Optimization for Facial Emotion Recognition" by Adrian Vulpe-Grigoraşi and Ovidiu Grigore [2] was also implemented first, then the mother paper was implemented taking extensive aid from the code provided by the secondary paper.

II. Methodology

A. Implementing the secondary paper

The secondary paper came with the code. The model implemented by this paper followed the traditional structure of repeatedly combining similar blocks made of Convolution, followed by Max Pooling and Dropout. The paper optimized the hyperparameters to obtain a claimed accuracy of 72.16% on the "FER2013" dataset. We achieved a slightly lower validation accuracy after training and validating on the dataset we first used from Kaggle, not the ICML Dataset.

B. Implementing the mother paper

The mother paper used a model having numerous elements different from the secondary paper. While the secondary paper had no branching and only traditional convolution-maxpool-dropout structure, the mother paper contained Depthwise Separable Convolutions(DSC), Convolutional Block Attention Module(CBAM) and a three branch approach along with Global Average Pooling used at the end before sending to Dense layer for classification.

The mother paper used 'Categorical Cross-entropy' as the loss function. The initial learning rate was set at 0.001 with Adam as learning rate optimizer while the number of epochs was 300 with no early stopping. Data augmentation is said to have been used for CK+ dataset due to its small size but whether it has been used for FER2013 dataset is a bit unclear. We used augmentations as suggested in the paper – horizontal flip, vertical flip, rotate, random scaling and shearing. Augmentation sometimes left the image in a different size than the input size, hence all the images were resized after augmentation. The augmentations were done using albumentations library.

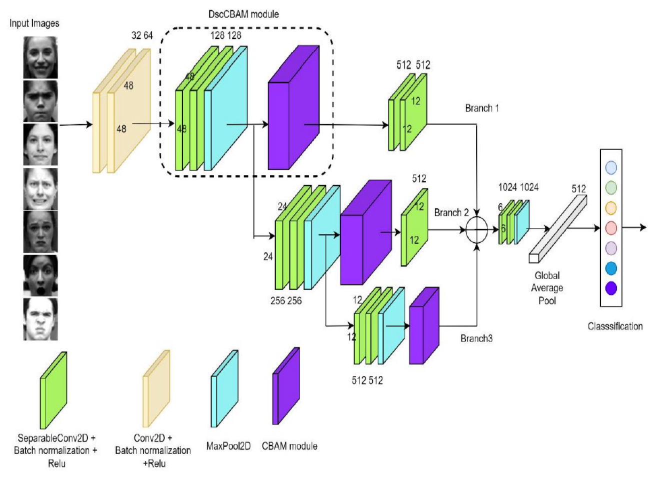
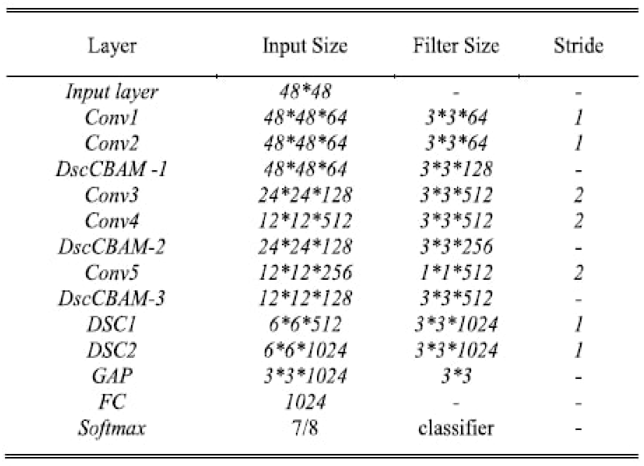


Fig. 1. Model Architecture



Layer	Input Size	Filter Size	Stride
Input layer	*48*48*	-	-
Conv1	*48*48*64*	*3*3*64*	*1*
Conv2	*48*48*64*	*3*3*64*	*1*
DscCBAM -1	*48*48*64*	*3*3*128*	-
Conv3	*24*24*128*	*3*3*512*	*2*
Conv4	*12*12*512*	*3*3*512*	*2*
DscCBAM-2	*24*24*128*	*3*3*256*	-
Conv5	*12*12*256*	*1*1*512*	*2*
DscCBAM-3	*12*12*128*	*3*3*512*	-
DSC1	*6*6*512*	*3*3*1024*	*1*
DSC2	*6*6*1024*	*3*3*1024*	*1*
GAP	*3*3*1024*	*3*3*	-
FC	*1024*	-	-
Softmax	7/8	classifier	-

Fig. 2. Whole Model Architecture with parameters

C. *Mother Paper Model*

A pictorial description of the model taken from the paper along with the whole architecture is shown in Fig. 1 and Fig. 2 respectively. (All the figures have been taken from the mother paper [1]) It is to be noted that the strides are also specified in Fig. 2. The model takes the input image and sends it through 2 blocks each consisting of 2D convolution followed by Batch Normalization followed by a 'ReLU' activation. The number of filters in the these two convolutional layers is unclear as they are declared to be 32 and 64 respectively in Fig. 1 while claimed to be 64 and 64 respectively in Fig.2. The former number was used in our model. Then this is fed into a DSC-CBAM module with 128 filters in the pointwise convolution layers. This model consists of 3 such modules in 3 branches, this being the first module in the first branch.

The DSC-CBAM module embeds a CBAM (Convolutional Block Attention Module) with DSC (Depthwise Separable Convolution) layers and Maxpooling. Each of these will be explained in detail later. The structure of the DSC-CBAM Module is shown in Fig. 3 and a comparison between DSC and normal convolution is depicted in Fig. 4.

The DSC-CBAM layer is followed by two DSC layers having 512 filters, concluding branch 1 of the 3 branch model. The output from the MaxPooling layer of the DSC-CBAM of branch 1 serves as the input of branch 2. Branch 2 starts with another DSC-CBAM layer, this one having 256 filters in its pointwise convolution layers. This is followed by a single DSC layer with 512 layers. The output of the MaxPooling inside the DSC-CBAM in branch 2 acts as the input of branch 3. Branch 3 starts with a DSC-CBAM module and it contains this module only. The module has 512 filters in its pointwise convolution layers. The 3 output tensors from the 3 branches are of same shape. They are added elementwise to get a merged output tensor. These are then sent through two more DSC layers each having 1024 filters in their pointwise convolution layers. Then instead of flattening, the model implements global average pooling to create one pixel from each of the 1024 channels (erroneously labelled as 512 in Fig. 1) and send them through a layer of 1024 fully connected neurons. The final output layer is another fully connected layer having 7 neurons (FER 2013 has 7 classes) having 'Softmax' activation function. It is to be noted that only the final layer has Softmax as activation function while the other layers have 'ReLU' as activation function.

The model also implements dropout layers at various locations but exactly where has not been specified in the paper.

D. *DSC-CBAM Module*

The DSC-CBAM Module consists of 2 layers of Depthwise Separable Convolution followed by a layer of MaxPooling. Then the maxpooled feature tensor is sent to the CBAM or Convolutional Block Attention module. Fig. 3 shows the structure of the DSC-CBAM Module. A brief description of the DSC and CBAM modules are as follows:

1) DSC or Depthwise Separable Convolution: The depthwise separable convolution is different from the normal 2D convolution. Fig. 4 depicts this difference. The Depthwise Separable Convolution splits the ordinary 2D convolution into a Depthwise Convolution followed by a Pointwise convolution. An ordinary 2D convolution having C_2 filters will create each of the C_2 channels after operating on all of the C_1 channels in its previous layer. Thus the number of kernels is $C_1 * C_2$ while the number of trainable parameters will be $kernel_{height} * kernel_{width} * C_1 * C_2$. The Depthwise Convolution instead creates one feature channel for each of the feature channels in the previous layer. Thus, the number of output feature channels remains the same as the number of input feature channels while the number of parameters becomes $kernel_{height} * kernel_{width} * C_1$. It is to be noted that C_1 is the number of channels in the input feature tensor, the depthwise convolution cannot create new channels or collapse the old ones. It can only create one channel for each of the channel in the input feature tensor. To increase or decrease

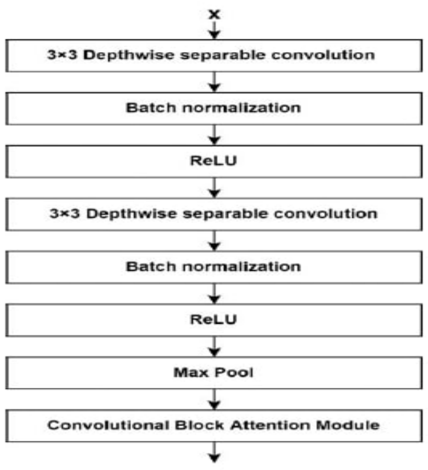

x
3×3 Depthwise separable convolution
Batch normalization
ReLU
3×3 Depthwise separable convolution
Batch normalization
ReLU
Max Pool
Convolutional Block Attention Module

Fig. 3. DSC-CBAM Module overview

the number of channels, the depthwise convolution, after a Batch Normalization and Activation layer, is followed by a pointwise convolution, which is an ordinary 2D convolution, but having kernel size (1,1). This is followed by another Batch Normalization and Activation layer thus creating the whole DSC module. This achieves what a normal convolution would do but with lower number of parameters and more normalization and activation. Fig. 5 shows this whole process pictorially.

2) CBAM or Convolutional Block Attention Module: The Convolutional Block Attention Module (CBAM) [4] is an improvement over the standard Squeeze-and-Excitation Net (SENet) [3]. Squeeze-and-Excitation Networks (SENet) are a type of neural network architecture designed to enhance feature recalibration and channel interdependencies. Introduced by Jie Hu et al. in 2017, SENets introduce two key operations: "squeeze" and "excitation." The "squeeze" operation compresses spatial information into channel-wise descriptors through global average pooling, reducing computational complexity. The "excitation" operation learns to emphasize informative features and suppress less relevant ones by adaptively recalibrating feature responses using learned weights. It does this by multiplying the features with the learned weights. Augmenting CNN models like ResNet, VGG etc with a SE Module significantly improves model performance. The following figure shows a visualization of the SENet structure augmented to a CNN model:

The Convolutional Block Attention Module (CBAM) improves over the standard Squeeze and Excitation Net (SENet) by adding a spatial attention mechanism along with the Channel Attention Mechanism developed in SENet. But it implements the Channel Attention in a manner different from the SENet approach too. The CBAM block has two modules sequentially: The Channel Attention Module (CAM) followed by the Spatial Attention Module (SAM).

In the Channel Attention Module (CAM), the input feature tensor F $\epsilon\ \mathbb{R}^{C\times H\times W}$, is taken and Global Max Pooling and

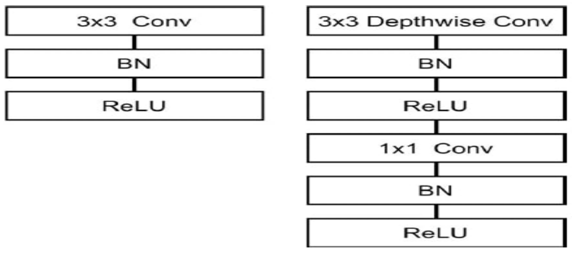


Fig. 4. Comparison between Depthwise Separable and Normal Convolution

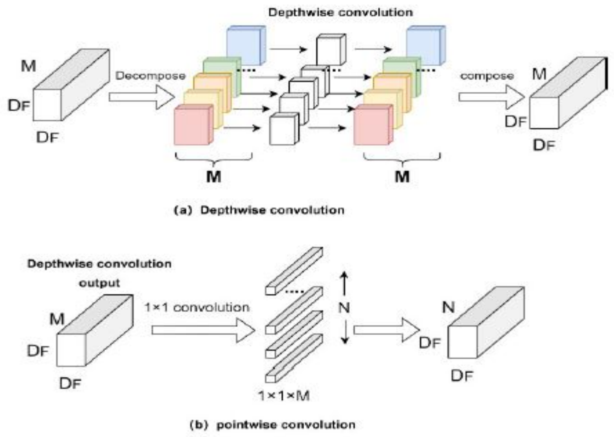


Fig. 5. DSC Module overview

Global Average Pooling are performed on it separately along the channel axis to obtain two output tensors having dimension C. Then these two feature tensors are passed through a shared MLP, which has three layers including input and output layer. Thus there are two separate weight vectors connecting them that act on these. Mathematically,

$$\begin{aligned} M_c(F) &= \sigma(\mathrm{MLP}(\mathrm{AvgPool}(F))) + \mathrm{MLP}(\mathrm{MaxPool}(F))) \\ &= \sigma(W_1(W_o(F_{avg})) + W_1(W_o(F_{max}))), \\ &\text{where } W_0\,\epsilon\,\mathbb{R}^{C/r\times C} \text{ and } W_1\,\epsilon\,\mathbb{R}^{C\times C/r} \end{aligned} \quad (1)$$

Here, σ denotes the sigmoid activation function. The MLP (Multilayer Perceptron) is shared for both inputs and the ReLU activation function is followed by the first input layer. An important thing to notice here is the reduction ratio, r, a positive integer. The hidden layer can reduce the number of neurons by r and the again increase to original number in the output layer. This reduces computational cost for r ¿ 1.

After passing through MLP, two sets of weights having length equal to number of channels in input feature tensor are found. These are added elementwise and passed through a sigmoid activation function to generate the final set of Channel Attention Module weights $M_c\ \epsilon\ \mathbb{R}^C$. These weights will be multiplied to corresponding channels of the input feature tensor to obtain the channel refined feature tensor F'.

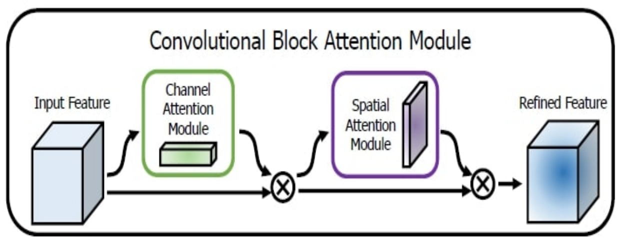


Fig. 6. The whole CBAM

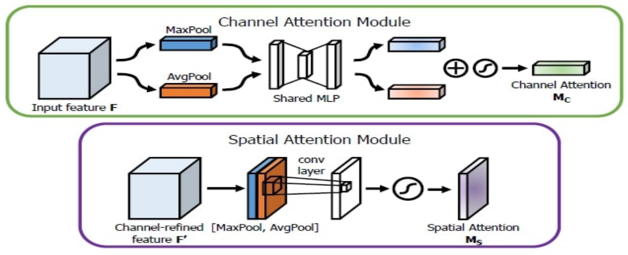


Fig. 7. CAM and SAM

In the Spatial Attention Module (SAM), the channel refined feature tensor undergoes MaxPooling and Average Pooling separately, but this time along the height and width axis, thus collapsing the channel axis. Two output tensors are obtained each having size 1 x H x W. The output is of size 2 x H x W. A 7 x 7 convolution acts on this to obtain a 1 x H x W sized output that is then passed through a sigmoid function to obtain the final spatial weight tensor $M_s \ \epsilon \ \mathbb{R}^{H \times W}$. Mathematically,

$$M_s(F) = \sigma(f^{7\times7}([\text{AvgPool(F);MaxPool(F)}])) \\ = \sigma(f^{7\times7}(F^s_{avg}; F^s_{max})) \quad (2)$$

After multiplying the spatial attention weight to each channel of the channel refined feature tensor F', the final output is obtained, which is the channel-refined and spatially refined feature tensor. Sequentially going through the Channel Attention Module and Spatial Attention Module produces the output of the Convolutional Block Attention Module. Mathematically,

$$F' = M_c(F) \otimes F,$$

$$F'' = M_s(F') \otimes F',$$

where $F$is the input feature tensor and

$\otimes$ is channelwise multiplication in the first line

and pixelwise multliplication in the second line

E. Dataset Description

The datasets used were two instances of the FER2013 dataset. The first dataset used is available in Kaggle. It contains 35887 grayscale images in two folders–train and validation. Each folder has 7 subfolders each for the 7 classes (angry, disgust, fear, happy, neutral, sad, surprised). The exact number of images in training dataset is as follows: Angry: 3993, Disgust: 436, Fear: 4103, Happy: 7164, Neutral: 4982, Sad: 4938, Suprised: 3205. So there is some class imbalance, especially the Disgust class has much less training data in comparison to the others. This will be referred to as the Kaggle dataset. It can be found in the link: https://www.kaggle.com/datasets/jonathanoheix/face-expression-recognition-dataset The second dataset is the original ICML Workshop Dataset on which the mother paper was trained and tested [5]. It contains 28709 images for training, 3589 images for validation and 3589 images for testing. Training data contains All test accuracies calculated are on the test data from this dataset.

III. Results

We tested various models. Some were trained on the first dataset and some were trained on the ICML Workshop FER2013 dataset with same parameters, some trained with augmentation and some without. The best model on FER2013 dataset uses auxiliary data along with ICML FER2013 for training [6]. All their results along with results from a few other papers are as follows for comparison:

Models	**Number of branches**	**Training dataset**	**Data Augmentation**	**Test Dataset**	**Test Accuracy**
Mother paper	3	ICML FER2013	Yes	ICML FER2013	69.49%
Our implementation of mother paper	3	Kaggle FER2013	No	ICML FER2013	56.06%
Our implementation of mother paper	3	Kaggle FER2013	Yes	ICML FER2013	71.55%
Implementation of mother paper model replacing DSC with Conv2D	1	Kaggle FER2013	Yes	ICML FER2013	41.46%
ResNet (untrained)	1	ICML FER2013	Yes	ICML FER2013	58.98%
Ensemble CNN [6]	3 models ensembled	ICML FER2013 + Auxiliary Data	Yes	ICML FER2013	75.8%

IV. Future Work

To further improve the accuracy of the model, changes can be made to the model architecture and the various hyperparameters e.g. reduction ratio, dropout percentage, number of filters can be varied. The exact location of the dropouts were not stated explicitly in the mother paper hence various locations may be experimented on. Several types of data augmentation was used, but other kinds of augmentations

V. Codes

The relevant codes along with links to the mother paper, secondary paper and a copy of the lab report have been uploaded to the github repository: https://github.com/Sifan69/EEE-402-Project-Group-1-FER2013

VI. Conclusion

This project aims to perform facial emotion recognition on FER2013 dataset using multi-branch attention CNN. It primarily focuses on implementing the model mentioned in a IEEE journal paper named "Facial Expression Recognition Using Multi-Branch Attention Convolutional Neural Network", which is the mother paper of this project. Depthwise separable convolution, multi branch structure and convolutional block attention journal are the highlighted parts of this CNN model. Although the model architecture showed promise in the original paper, achieving the same level of accuracy proved to be difficult in our implementation. Through repeated experimentation and analysis, we identified several factors contributing to this reduced accuracy, including variations in dataset characteristics, hyperparameter tuning, data augmentation and model structure properties. Using this knowledge, we have managed to reach 71.55% accuracy on test data while the original model claimed 69.49% accuracy while testing on the same dataset. However, there is always room for improvement by exploring alternative hyperparameter configurations, dataset augmentation techniques, and refining the model architecture to better suit our classification problem.

VII. Acknowledgment

This project was done as a part of the EEE 402 Artificial Intelligence and Machine Learning sessional course, under the supervision of Dr. Shaikh Anowarul Fattah and Shahed Ahmed, from Bangladesh University of Engineering and Technology. We are thankful to them for their helpful supervision and useful advice without which this project would not have been possible.

References

[1] Y. He, "Facial Expression Recognition Using Multi-Branch Attention Convolutional Neural Network," *IEEE Access*, vol. 11, pp. 1244-1253, 2023. DOI: 10.1109/ACCESS.2022.3233362.

[2] A. Vulpe-Grigoraşi and O. Grigore, "Convolutional Neural Network Hyperparameters optimization for Facial Emotion Recognition," in *2021 12th International Symposium on Advanced Topics in Electrical Engineering (ATEE)*, Bucharest, Romania, 2021, pp. 1-5. DOI: 10.1109/ATEE52255.2021.9425073.

[3] J. Hu, L. Shen, and G. Sun, "Squeeze-and-Excitation Networks," in *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2018, pp. 7132-7141. DOI: 10.1109/CVPR.2018.00745.

[4] S. Woo et al., "CBAM: Convolutional Block Attention Module," in *Computer Vision – ECCV 2018*, eds. V. Ferrari et al., Springer International Publishing, 2018, pp. 3-19. ISBN: 978-3-030-01234-2.

[5] I. J. Goodfellow et al., "Challenges in Representation Learning: A Report on Three Machine Learning Contests," in *Neural Information Processing*, eds. M. Lee et al., Springer Berlin Heidelberg, 2013, pp. 117-124. ISBN: 978-3-642-42051-1.

[6] A. Khanzada, C. Bai, and F. T. Celepcikay, "Facial Expression Recognition with Deep Learning," *CoRR*, vol. abs/2004.11823, 2020. url: https://arxiv.org/abs/2004.11823.

[7] CBAM code: https://github.com/kobiso/CBAM-tensorflow

[8] First Dataset used only for training: https://www.kaggle.com/datasets/jonathanoheix/face-expression-recognition-dataset

[9] ICML Challenge and dataset: https://www.kaggle.com/c/challenges-in-representation-learning-facial-expression-recognition-challenge