Facial Expression Recognition: Convolutional Attentional Masking Network and Ensemble Approach

by

Ibna Kowsar
17301130
Mashfiq Shahriar Zaman
17301167
Md. Fahmidur Rahman Sakib
17301196

A thesis submitted to the Department of Computer Science and Engineering
in partial fulfillment of the requirements for the degree of
B.Sc. in Computer Science and Engineering

Department of Computer Science and Engineering
Brac University
January 2021

Declaration

It is hereby declared that

1. The thesis submitted is our own original work while completing degree at Brac University.
2. The thesis does not contain material previously published or written by a third party, except where this is appropriately cited through full and accurate referencing.
3. The thesis does not contain material which has been accepted, or submitted, for any other degree or diploma at a university or other institution.
4. We have acknowledged all main sources of help.

Student's Full Name & Signature:

Ibna Kowsar
17301130

Mashfiq Shahriar Zaman
17301167

Md. Fahmidur Rahman Sakib
17301196

Approval

The thesis titled "Facial Expression Recognition: Convolutional Attentional Masking Network and Ensemble Deep Learning Model" submitted by

1. Ibna Kowsar (17301130)
2. Mashfiq Shahriar Zaman (17301167)
3. Md. Fahmidur Rahman Sakib (17301196)

Of fall, 2020 has been accepted as satisfactory in partial fulfillment of the requirement for the degree of Bachelor of Science in Computer Science and Engineering on January 11, 2021.

Examining Committee:

Supervisor:
(Member)

Md. Hasanul Kabir, PhD
Professor
Department of Computer Science and Engineering
Islamic University of Technology

Co-Supervisor:
(Member)

Rasif Ajwad
Lecturer
Department of Computer Science and Engineering
Brac University

Program Coordinator:
(Member)

Md. Golam Rabiul Alam, PhD
Associate Professor
Department of Computer Science and Engineering
Brac University

Head of Department:
(Chair)

Mahbubul Alam Majumdar, PhD
Professor and Dean, School of Data and Sciences
Department of Computer Science and Engineering
Brac University

Ethics Statement

We the members, hereby and sincerely declare that this thesis has been done based on the findings of our extensive research. All the materials which have been used are properly noted and cited in this report. This research work, neither in full nor any part has never been submitted by any other person to another university or any institution for the award of any degree or for any other purpose.

Abstract

Facial expression plays a significant role in human communication. The necessity of recognizing facial expression is increasing rapidly as it can be implemented in various important fields such as in human-computer interactions, medical care, autonomous transportation systems etc. The facial expression detection has been accomplished by the analysis of convolutional neural networks on the micromotors and action units. In this thesis, we have introduced a new variant of residual architecture named CAMnet which uses the split attentional module and the masking module mechanisms simultaneously. Also, the model performs better compared to other models without using any pretrained weights on small dataset like FER2013. Additionally, along with the CAMnet an ensemble model has been implemented and we have achieved 76.12% accuracy on the FER2013 test set.

Keywords: Facial Expression; Deep Learning; RAF; FER2013; CAMnet; Attention

Acknowledgement

Firstly, all praise to the Great Almighty for whom our thesis have been completed without any major interruption in this pandemic situation.
Secondly, to our supervisor Md. Hasanul Kabir and our co-supervisor Rasif Ajwad for their kind support and advice in our work. They helped us whenever we needed help.
Thirdly, We want to take a moment to thank our supportive friends who have been there to give us mental support in the hard times.
And finally to our parents without their throughout support it may not be possible. With their kind support and prayer we are now on the verge of our graduation.

Table of Contents

List of Figures

List of Tables

Nomenclature

The next list describes several symbols & abbreviation that will be later used within the body of the document

BAM Bottleneck Attention Module

CAMnet Convolutional Attentional Masking Network

CAU Convolutional Attentional Unit

CBAM Convolutional Block Attention Module

CNN Convolutional Neural Network

DSST Discrete Separable Shearlet Transform

DTAN Deep Temporal Appearance Network

DTGN Deep Temporal Geometry Network

FACS Facial Action Coding System

FER Facial Expression Recognition

HOG Histogram of Oriented Gradients

LBP Local Binary Pattern

LDP Local Directional Pattern

LSTM Long Short-Term Memory

MSE Mean Square Error

ResNet Residual Network

SeNet Squeeze and Excitation Network

SGD Stochastic Gradient Descent

SIFT Scale Invariant Feature Transform

SVM Support Vector Machine

Chapter 1

Introduction

Facial expressions have been considered as important elements in the process of human communications which helps to understand the inner states of others. Also, the most natural and instant indication about one's reactions can be obtained from the facial expression [1]. Humans can realize the facial expressions of others through vision and also understand the inner states through the analysis of the human brain. However, the method of attempting to analyze and recognize facial features from the visible perspective by a computer has been a challenging procedure and has been referred to as the automatic facial expression recognition system [2].

1.1 Background

As facial [3] expressions are dominant information channels in interaction, research on facial expressions have gaining a lot of popularities over the past decades not only in the fields of perceptual and cognitive science, but also in computer animations and Affective Computing Vision [4, 5].
Emotions not only play significant roles in human interaction but also in the way of computer usage. Moreover, a domain named Affective computing targets user feelings while the user interacts with computers and applications [5]. To increase the productivity and effectiveness of computers, Affective computing vision has been used to make computerized systems to concede human feelings. The purpose of emotion recognition is to systematically classify temporal states of emotion depending on the input data.
Facial emotions are the layout of various micromotor motions in the face [6]. Therefore, the facial movements can infer a person's emotional situation such as happiness, fear, anger, disgust, sadness and surprise. Moreover, the Facial Action Coding System (FACS) is the first broadly used approach to classify human expressions experimentally [6]. In the conventional approaches which have been used to detect facial expressions, the recognition process has three major segments, at first the face and facial landmark points have been detected, after that spatial and temporal features have been extracted from the detected facial points. In this part, various facial features extraction algorithms have been utilized namely HOG, SIFT, LBP, LDP, etc [7–10]. Lastly the facial expression has been classified. In the facial expression recognition step, some of the existing pre-trained facial expression classifiers have been implemented such as the SVM, random forest, AdaBoost and other classifier to classify the facial expressions from the extracted facial features.

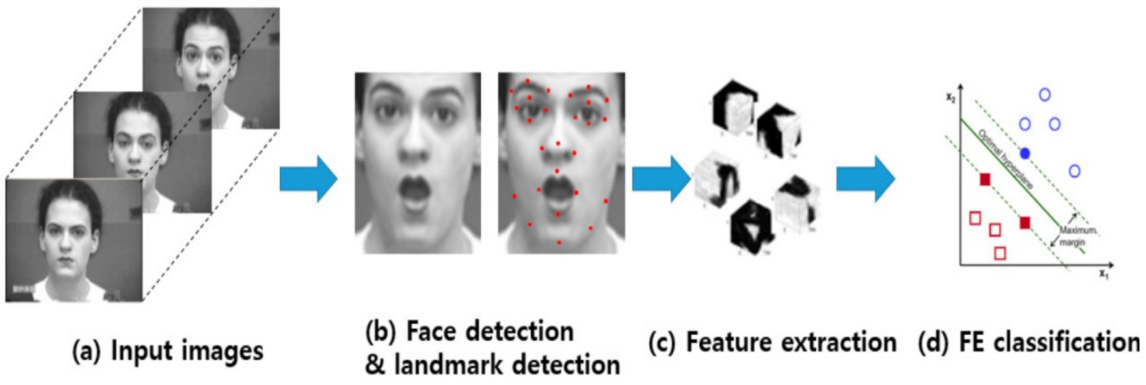


Figure 1.1: Traditional FE classification framework
Adapted from [3]

Meanwhile, in contrast to traditional approaches utilizing the handcrafted features, the emerging and newly developed deep learning models have achieved state-of-the-art detection accuracies. These models have outperformed the existing machine learning models because of the availability of the big data [11].Deep learning based facial expression recognition approaches have significantly reduced the preprocessing methodologies by introducing "end-to-end" learning in the pipelined based architecture directly from the input images [12] and also the face-physics dependent models. The convolutional neural network (CNN) has shown better accuracy in facial expression recognition among all other types of deep learning models. In the CNN based approach, at first the input images have been convolved through a collection of filters to produce feature maps. After that, each feature map has been combined to a fully connected layer and lastly the facial expression has been detected as belonging to a particular class of emotion among the 7 output emotions by implementing the softmax layer.

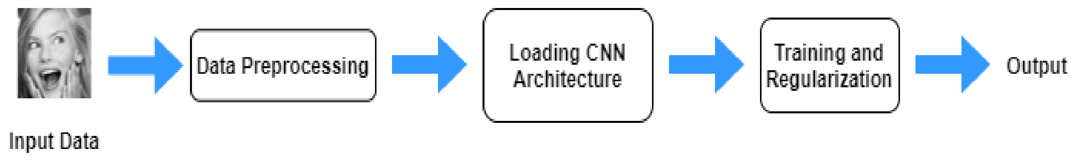


Figure 1.2: Deep Learning based FE classification framework

However, there are some difficulties in the process of facial expression recognition which need to be addressed. Firstly, the availability of mislabeling images in the dataset creates problems for the model while training and classifying the emotions of a particular class properly. Also, for the unlabeled data, traditional machine learning algorithms like Haar-cascade, HOG, LBP may not extract facial features accurately if they are not handcrafted properly. Also, the FER2013 is quite small in size and some of the classification classes do not have sufficient images to train the deep neural networks. Moreover, decoding compound expressions such as happily fearful which consists of the action units of both happy and fear, is a challenging task. Besides there are some similarities between the fear and sad classes as the AU of both classes overlaps. Lastly, the variations in the pose, occlusion and aging situations make the whole recognition procedure more difficult.

1.2 Problem Statement

As our study is focused on deep learning models, it is mandatory to have bigger dataset to get the expected performance from the model. However, all the existing dataset on FER is not big enough to train by using larger network architectures. Moreover, as the deep learning models are data hungry and also if the image size is not large enough we cannot go deeper using a architecture without loosing image details. Most of the existing work on FER has been done using simple network like VGG16 or even smaller model with depth of 5-10. Though using transfer learning is quite an advantage on deep learning field, it is not applicable for FER2013, RAF as they are not large enough and on the other hand, FER2013 has 48×48 size of images on its dataset and RAF has 100×100 size of images. Therefore, resizing and prepossessing is a challenging task here as resizing may result in block effect, false edge effect and undershoot-overshoot problem. As the existing Resnet50, SeNet, inception V3, VGG16, VGG19 architectures shows over-fitting problems indicating it is closely fitted to a particular set of dataset. Hyperparameter tuning methodologies need to be used to reduce the overfitting problem. Also, the state-of-the-art paper has achieved 76.85% accuracy using ensemble method.

1.3 Motivation

In the process of human communication, facial expressions have been considered as an important fundamental element. Furthermore, emotions not only play significant roles in human interaction but also in the way of computer usage. Moreover, a domain named Affective Computing utilizes the user's feelings while the user interacts with computers and applications. To increase the productivity and effectiveness of computers, Affective Computing Vision has been used to make computerized systems to concede human feelings. Depending on the emotion recognition process, the data, content and layout have been displayed to give the user a more pleasant scenario [6]. Besides, the recent works on facial expression recognition have been done by implementing the ResNet based models, VGG19, Inception architectures instead of using the Attentional mechanism. However, in the field of deep learning, Attention mechanism has achieved significant results compared to the existing architectures by emphasizing only on the important regions of the input. So, in this regard, we have been motivated to introduce a new variant of Residual network utilizing the concepts behind the Attentional module to increase the accuracy because of the increasing demand of facial expression recognition and the possibilities of outperforming the existing models.

1.4 Research Objective

As stated in problem statement, our main focus of this study is to come up with a new variant of architecture which will be able to go deeper and yet will achieve better performance without loosing much image details. Also, our focus is to use the idea of residual block and attention mechanism to merge them into to shallow network to make a deeper attentional module. The research is also focused on using different hyper-parameter tuning approach to get a desired output without using pretrained weights or any auxiliary data.

Therefore, in this research, we have focused on introducing a new variant of convolutional neural network architecture and at the same time to get a better performance on facial expression recognition. The major contributions of this thesis are stated as follows:

- **Novelty:** We have introduced a new variant of Residual Network named as Convolutional Attentional Masking Network (CAMnet).
- **Performance:** Compared to other existing non-pretrained model, we have achieved better result on emotion classification from facial expressions.
- **Accuracy:** Our ensemble model has achieved 76.12% accuracy on test set which is 0.7% less than current state-of-the-art.

The rest of the report has been organized in the following manner. Chapter 2 discusses the literature reviews and related work on FER. Chapter 3 describes the implemented dataset. Subsequently, in chapter 4, we have articulated implementation of existing model and our proposed method has been narrated in chapter 5. After that, our experimental setup have been stated in chapter 6. Finally, chapter 7 concludes our thesis.

Chapter 2

Literature Review and Related Work

2.1 Convolution Neural Network

A number of CNN models have been introduced gradually from the beginning of the year 1990. The one of the initial successful applications of CNNs is the LeNet-5 architecture [13]. It consists of 3 layers: convolution, pooling and lastly non-linearity. The fully connected layers have been used as the final classifier. After that, AlexNet [14] which belongs to the Deep CNNs that significantly popularized convolutional networks in computer vision. This network, compared to LeNet, was deeper consisting of about 60 millions of parameters and has implemented Relu as a non-linearity function and the method of overlapping Max Pooling and stacking the convolutional layers. Furthermore, VggNet [15] or VeryDeep has become popular for good performance because of the very deep architecture. This network has used much smaller 33 filters in each convolutional layer. After that, Inception or GoogLeNet [16] has a significant advantage in reducing the number of parameters as they use the idea of average pooling instead of using fully connected layers with convolutional layers. Lastly, the Residual Network [17], the main contribution of this architecture is to use the batch normalization method and adding shortcut connections to feed forward the network for training deeper architectures. Moreover, all the CNN architecture has some basic operations which have been illustrated in this chapter.

2.1.1 Layers

Convolutional Layer

All the convolutional layers have a number of filters and the parameters which need to be learned. Moreover, the dimensions of these filters are always smaller than the input filter. Each of the filters has been convoluted over the input image channels to compute an activation map [18]. The convolutional layers output has been calculated by stacking the activation maps of all filters. Moreover, as the convolutional layers are locally connected, it allows the architecture to learn filters which maximally respond to a local portion of the input. Also, from the convolutional operations between the filter, the input and the parameters of the filters the activation map has been created. It also shares the weight with the local position and this sharing

weights helps to decrease parameter numbers as well as for the efficient learning and better generalization.
In Equation 2.1,2.2, A^{m-1} represents an input image and computes an output image A^m by doing convolution over k channels. Also, W_{ok}^m is a matrix that parameterizes a special filter which can be used by a karnel to detect feature details from an image. Moreover, the matrices share the learned parameters to the networks forward connections.

$$A_o^{(m)} = g_m\Big(\sum_k W_{ok}^{(m)} * A_k^{(m-1)} + b_o^{(m)}\Big) \tag{2.1}$$

$$W_{ok} * A_k[s,t] = \sum_{p,q} A_k[s+p, t+q]W_{ok}[P-1-p, Q-1-q] \tag{2.2}$$

Pooling Layer

The pooling layer has been usually incorporated between two convolutional layers. It helps to reduce the number of parameters and furthermore computation complexities by implementing a down-sampling mechanism for the representation. This function can be max or average. The max pooling layer has been used frequently because of its better performance.
Equation 2.3 illustrates that an input image A^{m-1} can be convoluted by a pooling layers of size $p_m \times q_m$ with a stride of α,β for a k channel region image which results in an image A_m.

$$A_o^{(m)}[s,t] = k\cdot\Big(\sum_{p,q}(A_o^{(m-1)}[\alpha_m s + p, \beta_m t + q])^p\Big)^{\frac{1}{p}} \tag{2.3}$$

Linear or Fully Connected Layers

The linear layer can be defined as a function which has the capability of applying a linear transformation on the vertical input of dimension I and can output a vector dimension of O [18]. Moreover, the layer has a bias parameter, b which is shown in Equation 2.4, 2.5.

$$y = A \cdot x + b \tag{2.4}$$

$$y_i = \sum_{j=1}^{i}(A_{i,j}x_j) + b_i \tag{2.5}$$

The linear layer has been developed based on the fundamental processing unit of the brain which is known as the neuron. Furthermore, there are about 86 billion neurons and they are connected with a total number of $10^{14} - 10^{15}$ synapses. The neuron works in a systematic way such as receiving input signals and then produces output signals. Similarly, the linear layer is a simplified version of the dendrites of a group of neurons connected with the same inputs. For an activation function, sigmoid activation function has been used to imitate the 1-0 impulse carried away. Nevertheless, the activation function is the identity function which outputs the actual values.

2.1.2 Activation Functions

The neural networks have the ability to estimate any non-convex functions and the outcome of the non-linear activation functions. Moreover, these functions take a vector as an input and perform a fixed point-wise operation. Mainly, there are three types of activation functions, Binary, Linear and Non-Linear. For deep learning we only focus on Non-linear activation functions.
To control the gradient and learning rate of deep learning models there has been several activation functions introduced. The following section demonstrates the activation functions that are used in current research.

Sigmoid

The mathematical form of the sigmoid non-linear function has been given below in Equation 2.6

$$n = \sigma(m) = \frac{1}{1 + exp^{-m}} \tag{2.6}$$

This function flattens a real value between 0 to 1. When the neuron's activation becomes close to either 0 or 1, the gradient of the regions becomes almost 0. Furthermore, this back-propagation method has lackings at modifying the parameters of itself and the parameters of the previous neural layers [19]. Also, it converges slowly and it is not centered on zero.

Hyperbolic Tangent (TanH)

The TanH function follows the following mathematical term as of Equation 2.7

$$y = 2\sigma(2x) - 1 \tag{2.7}$$

This function flattens the input tensors real numbers between -1 and 1. Also, it has similar disadvantages like the sigmoid activation function [20].

Rectified Linear Unit (ReLu)

The ReLu function can be easily understood by observing the Equation 2.8

$$y = max(0, x) \tag{2.8}$$

Relu has been illustrated that it can accelerate the training. As the convergence of SGD can be accelerated greatly by using it, the Relu function is becoming a popular choice gradually to be used. Moreover, the performance of the function does not suffer from the vanishing or exploding gradient and the function implements reasonable operations rather than implementing the expensive exponentials. But Relu function has some disadvantages such as removing the negative information and does not perform well for all sorts of datasets and architectures [21]. The Relu activation function always returns 0 if the output is less than 0 else it gives identical output to the input.

Leaky ReLu

To reduce the "dying ReLu problem" Leaky Relu has been implemented. It will show a small negative slope having such values as 0.01 instead of the function being zero. Moreover, this function computes $f(x)$ in the following manner, Equation 2.9

$$f(x) = 1(x < 0)(\alpha x) + 1(x \geq 0)(x) \tag{2.9}$$

where α has been considered as a small constant [22].

Parametric Rectified Linear Unit (PReLU)

PReLU has been used to generalize the traditional rectified unit with a slope for the negative values which has been stated as functions in Equation 2.10, 2.11

$$f(y_i) = y_i,\ if\ y_i \geq 0 \tag{2.10}$$

$$f(y_i) = a_i y_i,\ if\ y_i \leq 0 \tag{2.11}$$

The y_i is the nonlinear activation's input on the i^{th} channel, the slope of the negative part is controlled by a_i. Moreover, the subscript i in a_i indicates the capabilities of varying the nonlinear activation on various channels. PReLus have been used for the linear layers to maintain the positive and negative responses of filters [23].

2.2 Existing Deep Learning Models

2.2.1 VGGNet

The authors have shown in this paper [15] that the accuracy of the convolutional network can be increased drastically in detecting large-scale images by increasing the depth of the networks. The authors have proposed a more accurate ConvNet architectures which have the capabilities to acquire impressive accuracy on ILSVRC classification and localisation tasks and can also perform well on image recognition datasets. Moreover, the described model has also shown magnificent performance while using as a part of simple pipelines. The inputs of the described ConvNets have a size of 224*224 RGB images. The most important advantage of using VGG-16 is that it is a much straightforward network. The conv layers are just 33 filters with a stride of 1 having the same padding. Moreover for all max pooling layers have 22 filters with a stride of 2.

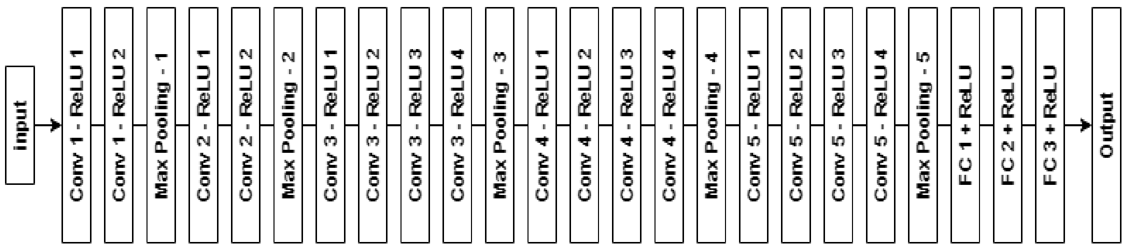


Figure 2.1: VGG19 Model Architecture

In the pre-processing step, the mean RGB value has been subtracted from each pixel of the training dataset. Subsequently, a stack of convolution layers have been followed by three Fully-Connected layers whereas the first two have 4096 channels in each. Also, the rectification non-linearity has been used in all the hidden layers. The authors have achieved remarkable accuracy for classifying large images using very deep convolutional networks consisting up to 19 weight layers. Though the main drawback of using the VGG-19 neural network it has implemented using 19 layes. It is understandable that the VGG-16 and VGG-19 achieves the same performance and their usability depends on the implemented situation. Vgg19 architecture have been shown in Figure 2.1. However, VGGNet has about 138 million parameters and its complexity is relatively higher than other existing architectures. Hence, the training proccess is very slow for VGGNet. Due to the depth and number of fully connected nodes, the weight of VGG16 is over 533 MB and for VGG19 it is over 574MB. As as result, implementing the weight of VGGNet is a challenging task.

2.2.2 ResNet

To reduce the difficulties of the training period, the layers have been explicitly redesigned to learn residual functions containing reference to the inputs and not to learn unreferenced functions [17]. Here, the desired underlying mapping has been $H(x$, and the stacked nonlinear layers fit the mapping of $G(x) = H(x)x$. As a result, the actual mapping has become $G(x) + x$. The equation $G(x) + x$ has been proposed to reduce the degradation problem by feeding forward to neural networks by creating the proposed shortcut connections. The skip connections have been implemented to achieve the identity mapping and also to add the outputs with the outputs of the stacked layers. The convolutional layers of the architecture mostly have 3×3 filters and the value for stride is 2. In the end of the network, a global average pooling layer and a n-way (class number) fully-connected layer and softmax have been added. Finally, the total number of weighted layers can be 18, 34, 50, 101, 152.
The important feature of ResNet is to train deep neural networks because of having skip connection and it can also generalize well. ResNet can also increase the speed of convergence. However, it has some disadvantage such as it only focuses on the depth of the architecture not the width, and also it learns all the features of each layer whether the feature is important or not and passes it to the next layer using residual block.

ResNet34

At first the inputs have been processed through the 7×7 Convolutional layer in the beginning and the total number of filters in the first layer will be 64. After that, pooling operation has been applied. Moreover, the resnet architecture has been created by stacking residual blocks. Here, each residual block has two 3×3 conv layers. Relu activation function has been applied between the two layers. The output from the second convolutional layer will be summed together with the unchanged input that has been entered to the residual block. Moreover, relu activation function has been applied on the summed value. Typically, the total number of filters has been doubled and downsample has been applied using a stride value of 2. Finally,

no fully connected layer at the end of architecture. ResNet34 model architecture have been illustrated in Figure 2.2.
The strength of ResNet34 is that it can easily reduce the vanishing gradient problem because of its skip connection. The main advantage of using ResNet 34 block is it only has 3.6×10^9 FLOPs indicating less computational complexities. However, it uses the idea of basic block and can not increase efficiency like other deeper network.

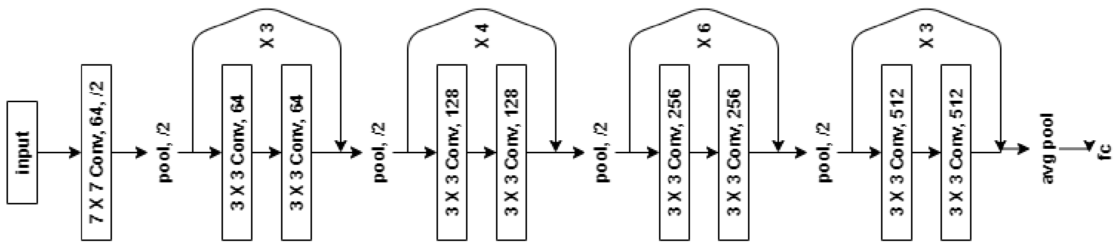


Figure 2.2: Resnet 34 Model Architecture

ResNet50 Block

In the case of ResNet 50 which is shown in Figure 2.3, the block has one 3×3 convolutional layer and also two 1×1 convolutional layers. The 1×1 convolutional layer which has 64 filters has been used to project to $28 \times 28 \times 64$. Sequentially, the 3×3 conv operates over only 64 feature maps. Lastly, the 1×1 conv, 256 filters projects back to 256 feature maps. Here, the "Bottleneck" layer has been used to improve the efficiency of the architecture.
ResNet50 overcomes the efficiency problem that has ResNet34 and it can go deeper as it uses the blottlenet block. However, adding skipped connections with bottleneck to the network creates dimension mismatch problem which needs to be checked in each layer. Also, its complexity is higher as it has FLOPs of 3.8×10^9 and the number of parameter 25.6 million.

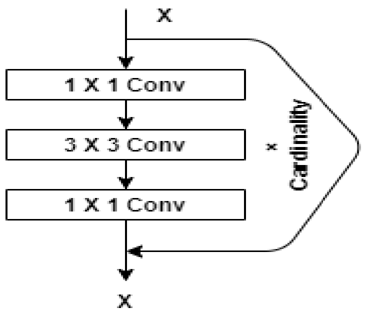


Figure 2.3: Resnet 50 Block Architecture

SeResNet Block

In SeResNet architecture which can be seen in Figure 2.4, after getting output from the residual block, global pooling function has been applied on the output. After that it has been passed through A fully connected layer and relu activation function has been applied. Furthermore, it has been passed through a fully connected layer and Sigmoid activation function has been applied. After that, it has been scaled to match with the height and width of the input and also this output and the unchanged input has been summed up together [24].

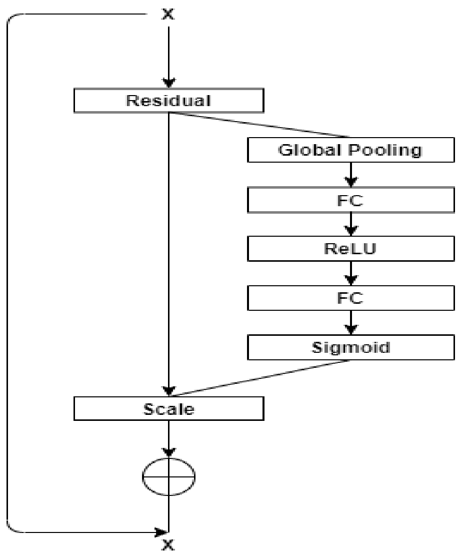


Figure 2.4: SeResNet Block Architecture

SeResNet has capability to increase channel interdependencies without increasing any computational cost. It has almost the same FLOPs as ResNet50 and other deeper residual network. However, it can only implemented on bottlenek block based architecture and therefore, it also has the same problem that is has to check dimension in each layer also batch norm based computational complexity still persists.

ResNext Block

ResNext block is a homogeneous neural network model which has reduced numbers of hyperparameters than the conventional ResNet. The block has a cardinality of 32 means that the same transformations have been applied for 32 times and the result has been summed up together at the end [25]. Furthermore, the network shares the same set of hyperparameters if the blocks produce the same dimensional features spatial maps and if the width of the block has been multiplied by a 2 then its spatial map will be downsampled by the same factor. The block follows the split-transform-merge strategy which has been visualised in Figure 2.5.
ResNext50 has improved the accuracy using lower number of parameter and lower complexity as it has FLOPs of 4.1×10^9 and total number of parameters 25 million. Moreover, the disadvantage of the architecture is that the complexity has been increased and as the width of each block is changed dimensionality mismatch need to be check for each cardinality.

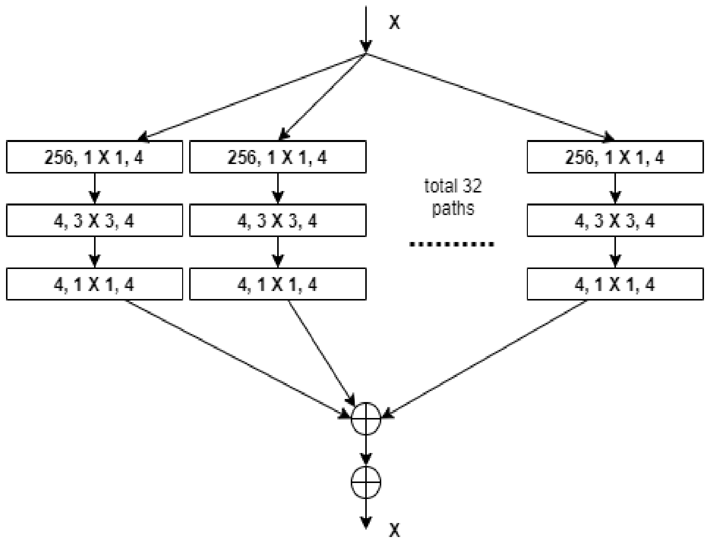


Figure 2.5: ResNext Block Architecture

ResNest Block

ResNest is a new variant of residual network which has been created by stacking the Split-Attention blocks [26] which has been shown in 2.6. The inputs have been divided into several feature map groups called cardinals. In each cardinal group the input has been splitted into multiple segments which contain one 1×1 conv layer and one 3×3 conv layer. Then the output has been passed through the split attention block. A combined representation for each cardinal group can be obtained by merging via an element-wise summation across multiple splits. The representations of the cardinal groups have been concatenated along the channel dimension. Furthermore, the output has been passed through a 1×1 conv layer because the output feature-map and the input do not have the same shape. This output and the input has been summed up together by using a shortcut connection.

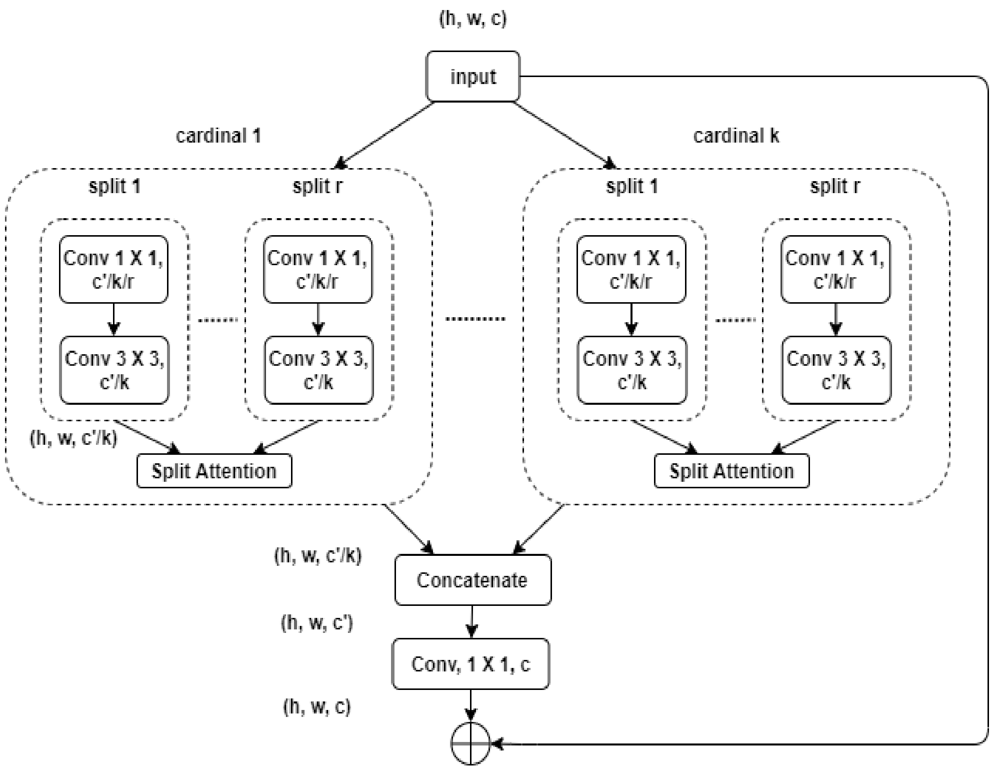


Figure 2.6: ResNest Block Architecture

In this architecture, the authors have addressed that, it can reduce the computional complexity of the architecture like SeResNet and doing the attentional operation to each individual groups. However, the total number of parameter is higher than ResNet50 and which is 27.5 million.

2.2.3 Inception

The authors in [16] have proposed the deep convolutional neural network architecture named Inception which can significantly improve the utilization of computing resources. The proper utilization of valuable computing resources has been achieved by crafting the design carefully and also allows increasing depth and width of the network while maintaining the computational budget constant. The primary intuition behind developing the Inception modes is to determine an optimal local sparse architecture. Moreover, the authors have restricted the filter sizes to 11, 33 and 55 to avoid the disadvantages of patch alignment. The layers consist of some occasional layers which have the ability to perform the max pooling operation having a stride value of 2 to halve the resolution. Moreover, the other main advantage of this model is the capability to allow an increasing number of units at each stage significantly without increasing the computational complexity significantly. The significant advantage of this method to achieve a desirable increase of computational capacities compared to shallower architectures. Although the similar type of performance can be achieved by utilizing a more expensive computational network of similar depth and width. Moreover, the proposed methodology demonstrates a stable evidence that starts moving to sparser architectures is possible to develop. Inception V3

Block Architecture has been illustrated in Figure 2.7.

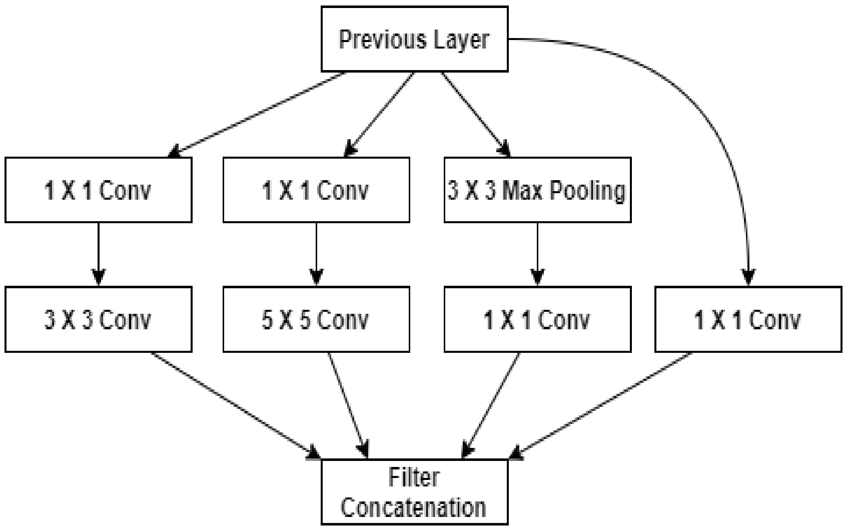


Figure 2.7: Inception V3 Block Architecture

Inception architecture attains efficiency by reducing the input image while getting important spatial data simultaneously. As the network has the capability of reducing the input image, it can also decrease the computational load. However, inception v4 has higher complexity than ResNet50.

2.3 Challenges with Training

2.3.1 Data Dependency

Deep learning achieves optimum performance whenever it gets enough quality data on its training set and this performance will increase as the data availability grows. However, if enough quality data isn't fed into a deep learning it cannot perform well. Therefore, if the dataset is not big enough, the trained model will be biased on training set and it will not be able to achieve desired classification or segmentation result in the test set. So, the model will be biased and overfitting will happen.

2.3.2 Overfitting

In a model when the performance is very high it learns all the patterns and noise of the training set but doesn't perform to that extent on unseen data because of the models biasness on the training set. The problem of overfitting occurs when the dataset is too small with respect to the depth of the model or the number of iterations. To handle the issue different techniques like data augmentation, early stopping and regularization of parameters can be used.

2.3.3 Gradient Vanishing

The problem occurs when many layers implementing certain activation functions have been added to the architecture of the neural networks and furthermore the gradients of the loss function approaches zero value. For example, while using the sigmoid function, it transforms the values into a small input space between 0 and 1. As a result, the chaanges in input becomes relatively small change in the output which results in the small change in the derivative.

2.4 Transfer Learning

2.4.1 Fine Tuning

This method involves the process of training of a pretrained network on a smaller dataset. Typically, the fully connected layers in the ending of the neural network architecture can be perceived as the classification layers and also a reduced learning rate has been adapted to the previously pretrained layers. In this way, the features have been adapted to the new dataset. It has been used to speed up the training and to overcome small dataset size.

2.4.2 Features Extraction

Extracting features is usually achieved from the network by forwarding examples and transforming the derived values into a non-redundant form of data. Moreover, transformations have been done to the images following some processes such as horizontal or vertical flip. After that the associated features of the example are aggregated by averaging or by stacking them. At last, a classifier (Supervised or Unsupervised) has been trained and tested on the features. Usually, the latter is a Support Vector Machine with a linear kernel or a deep neural network.

2.5 Optimization Algorithms

As the loss function of a CNN model is thoroughly non convex and is also fully derivable, gradient based optimization algorithms have been applied. But, the CNN models have been made using millions of parameters. As a result, only the first order derivatives have been used to compute because the second order derivatives are expensive in terms of memory and computational constraints.

2.5.1 Stochastic Gradient Descent (SGD)

In Equation 2.12, this optimization algorithm has been considered as one of the main optimization algorithms. It consists a few examples to compute the gradient of the parameters with respect to the loss function [27].

$$\theta_{t+1} = \theta_t - \lambda \cdot \nabla\theta_t L(f_{\theta_t}(x_i), y_i) \tag{2.12}$$

Though, this algorithm reaches good local minima, the parameters are randomly initialized. It has been said that the stochastic property of the algorithm, allowing the latter to optimize different loss functions and helps to reduce bad minima. Also, a lot of local minima are almost as accurate as the global minima.

2.5.2 Adam

Adam is a popular optimization algorithm because it achieves better results quickly. It has been used to update the weights of the network in an iterative based way in training data [28]. Implementing Adam has some advantages such as it is straightforward, computationally efficient and it requires less memory. The Adam optimizer has the following mathematical form as of Equation 2.13,2.14.

$$m_t = \beta_1 m_{t-1} + (1-\beta_1)g_t \tag{2.13}$$

$$v_t = \beta_2 v_{t-1} + (1-\beta_2)g_t^2 \tag{2.14}$$

This algorithm uses the benefits of both AdaGrad and RMSProp to converge faster.

2.5.3 AdaDelta

This algorithm [29] is the extension of Adagrad which has the ability to reduce the aggressive, monotonically decreasing learning rate and also restricts the window of accumulated past gradients to some fixed size w . This operation can be understood by the following Equation 2.15

$$E[g^2]_t = \gamma E[g^2]_{t-1} + (1-\gamma)g_t^2 \tag{2.15}$$

The variable w stores the sum of gradients recursively as a decaying average of all past squared gradients. By using the AdaDelta, the default learning rate needs not to be defined. The result $E[g^2]_t$ represents that it only depends on the previous and current gradient to calculate the gradient at time t and γ represents the momentum term.

2.5.4 RAdam

This optimization technique eliminates the problem of variance issue of the adaptive learning rate means the variance is quite large in the early stage of the model training. It has introduced a term to explicitly rectify the variance of the adaptive learning rate based on derivation [30]. Equation 2.16-2.21 shows how the optimization has been conducted.

$$g_t = \nabla_\theta f_t \theta_{t-1} \tag{2.16}$$

$$v_t = \frac{1}{\beta_2 v_{t-1}} + (1-\beta_2)g_t^2 \tag{2.17}$$

$$p_t = \beta_1 p_{t-1} + (1-\beta_1)g_t \tag{2.18}$$

$$p_t' = \frac{m_t}{1-\beta_1^t} \tag{2.19}$$

$$\rho_t = \rho_\infty - \frac{2t\beta_2^t}{1-\beta_2^t} \tag{2.20}$$

$$\rho_{\infty} = \frac{2}{1-\beta_2} - 1 \tag{2.21}$$

Here, the learning rate is small in the first few epochs of training which justifies the warmup heuristic. This method helps RAdam to rectify the variance problem.

2.6 Loss Functions

2.6.1 Mean Square Error Loss

MSE is defined as a multi class loss while train neural networks [31].

$$Loss(q,r) = \frac{1}{m}\sum_i |q_i - r_i|^2 \tag{2.22}$$

Here in 2.22, q is a vector of m prediction and moreover r is a binary vector full of 0 besides a 1 in terms of class dimension.

2.6.2 Cross Entropy Loss

It is also a multi class loss and it outperforms the MSE [32].

$$Loss(m,n) = -\sum_i n_i * log(\frac{exp(m_i)}{\sum_j exp(m_j)}) \tag{2.23}$$

In the Equation 2.23 m is defined as a vector of n predictions and n is a binary vector full of 0 besides a 1 in the corresponding class dimension. The Cross Entropy is better in the sense that MSE loss will eventually slow down learning on the other hand the Cross Entropy will not reduce the learning rate.

2.6.3 Loss Multi Label

The loss function in the Equation 2.24 has been created by adapting the Cross Entropy loss for multi-label classification [33].

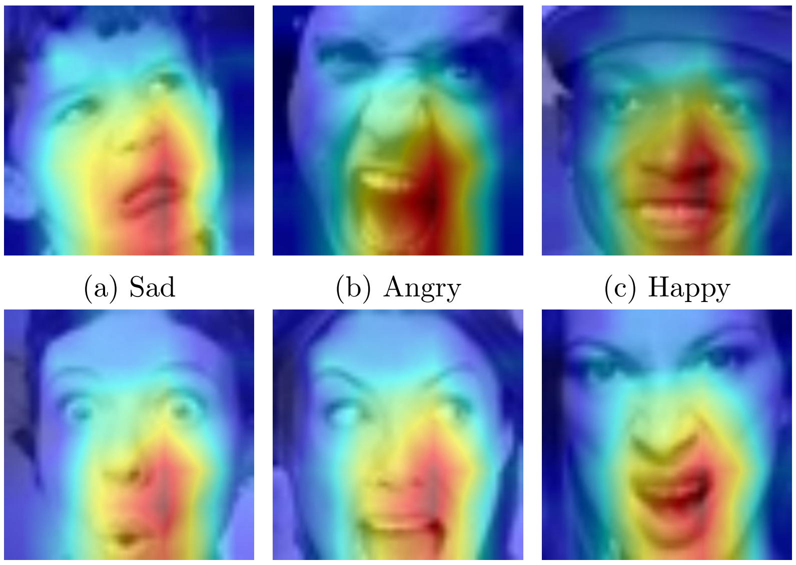
$$Loss(m,n) = -\sum_i n_i * log(\frac{exp(m_i)}{1+exp(m_i)}) + (1-n_i) * log(\frac{1}{1+exp(m_i)}) \tag{2.24}$$

2.7 Advanced Visualization Techniques

2.7.1 Gradient Based

In this approach, the output gradient with respect to the input has been implemented for completing the saliency maps [34]. To render the gradient as an image is a popular approach and it also conveys which image regions the current implemented classification mostly depends as shown in Figure 2.8. In this process, a network learns through manipulating the input using gradient descent to enhance the activations of certain nodes which have been selected from the hidden layers.

$$\frac{\partial S_c}{\partial I}|_{I_0} \tag{2.25}$$



(a) Sad (b) Angry (c) Happy

(d) Surprised (e) Fear (f) Disgust

Figure 2.8: Grad-CAM Visualization (CAMnet weights)

Here in the Equation 2.25, I_0 is the image, class is defined by c, and the class score function is denoted by $S_c(I)$. With respect to I at I_0, the heatmap can be calculated as absolute of the gradient of S_c.

2.8 Facial Expression with Deep Learning

The facial expressions have been considered as a salient feature of human language and interaction. Moreover, emotion recognition performs a significant role in various fields namely in human-computer association, automation, and human-robot communication. With the advancement of technology, the facial expression recognition (FER) has achieved immense popularity among the people. Since several works have been conducted based on this topic for detecting the seven basic emotions precisely.

A recent work [17] on facial expression recognition has described a fuzzy - CNN architecture which has the capability of improving the detection accuracy on FER and moreover decreasing processing time significantly. The authors have initially recognised the face and also the facial landmark points by implementing a classifier similar to haar. After that various preprocessing mechanisms such as resizing and normalization have been applied to achieve the dimension of a 48×48 pixel sized image which has contained mainly the details of the facial points. Furthermore, the kernel has been created by using the image and the kernel has been necessary for the convolution mechanism by implementing the formulas of the subtraction and the mean. As a result of this procedure, an output image has been generated which contains mainly the edges and points of the face and the whole procedure has been manifested as the K-means clustering mechanism. After that, the newly constructed image has been utilized as the beginning kernel for the CNN architecture. Additionally, the convolutional mechanism implements a multiplication operation for matrix and as a result the facial edges and landmark points have been extracted more precisely. Furthermore, the whole training process has been completed by implementing a CNN model to successfully extract facial landmark details and also to label the extracted data properly. But, the SVM classifier has been utilized in the place of a softmax function. Also, to reduce the difficulties of implementing the random kernel procedure, the authors have followed this process. At last, the SVM shows the decision by further processing the data which has been given by the fuzzy-CNN architecture.

The authors [18] of this paper have shown a deep neural network which has been created by establishing two different architectures. Initially, the first deep architecture has been named as the deep temporal appearance network (DTAN) which has the strength to focus only on the temporal appearance features from the given image sequence and furthermore the deep temporal geometry network (DTGN) has been utilized to get the temporal geometry features. Lastly, the described two deep networks have been concatenated to create a deep temporal appearance-geometry network (DTAGN) to increase the performance of facial expression recognition process significantly. Moreover, to increase the accuracy of the recognition process, the authors have introduced a joint fine-tuning approach by concatenating the two previously described architectures which have no similarities in their architectures.

In [35] authors have used a combination of CNN and LSTM to extract the emotion feature. Here, they have conducted the work to make a human robot interaction system. In the first place, they processed the image by using a CNN model to reduce the dimension of the data and extract features at the same time. As training would take higher order of time with the raw image data. Therefore, the processed feature output of CNN has been given as input to the LSTM model. Furthermore, the current performance has been further improved using a deeper neural network. However, using a deeper layer in CNN might create vanishing and exploding gradient problems. For this reason, a residual block has been used to make up the training error.

The authors [36] have described that though Facial Expression Recognition (FER) is an important process, the current FER methods fail to provide higher accuracy in real-time scenarios. The authors have developed a Hybrid Convolutional-Recurrent Neural Network model for detecting facial expressions in images. Their proposed model consists of Convolution layers which followed by Recurrent Neural Network. Moreover, this merged model can determine the association within facial images and the temporal dependencies in the images can be obtained by implementing recurrent networks. The authors have evaluated the proposed hybrid model based on the two general dataset.

In [37] an ensemble of multiple networks is used by initializing different CNN so that the network can learn the ensemble weights. However, ensemble of multiple randomly initialized networks can lead to diverse network classification. Therefore, two loss functions named optimal ensemble log likelihood loss and optimal ensemble hinge loss were defined to optimise the learning rate. Moreover, in their model architecture five convolutional layers and three stochastic pooling has been used instead of max pooling. However, in [38] shows that stochastic pooling shows poor response than max pooling while training and for large dataset it overfits. In that regard, the stochastic pooling might have been combined with any form of regulation such as weight decay, drop out, data augmentation, etc to make up overfitting while training deep neural networks.

The authors [39] have discussed that extracting emotional features is referred to as a crucial action for recognizing facial expression. Moreover, the existing methods have not fully examined the features of facial elements and movements of face muscles. The authors have proposed a model which can detect 'salient' distance features from face and the features have been obtained using patch-based 3D Gabor features. In addition, the model has shown notable improvement in correct recognition rate (CRR) because of taking the consideration of facial elements and also the motions of the muscles in the face. Here, the authors stated that facial movement features refer to the feature position and the changes in face due to the movements of facial elements and muscles. In this study, the authors suggested a model which has the capability of increasing the performance of FER by acquiring the facial movements in static images on distance features systematically. The previously mentioned distance features prevailed by taking 'salient' patch-based Gabor features and operating patch matching operations. Furthermore, it is shown in the paper, the patch-based Gabor features have performed very well in recognizing position, scale, orientation changes in a given data set. On the contrary, the recognition rate of the appearance based features is not significant as the appearance based features generate similar results even changes occur in facial movements and these features do not interpret the differences. The authors have gained desired results by merging patch-based Gabor features in the restricted matching area.

In [40] the authors have proposed a novel approach of detecting facial expression by utilizing the discrete separable shearlet transform (DSST) and normalized mutual information feature selection where the whole model is splitted into five operations. At first, the training and testing data sets have been trained. After that, DSST has been applied to the preprocessed images and the gathered information acts as the transformation coefficients. Then, the improved normalized mutual information feature selection has been applied to measure the optimal feature subset from the previous feature set. After implementing the linear discriminant analysis the selection of the feature space has been reduced. Finally, the expression has been recognized after using SVM. The author has suggested using DSST which is a multi-scale geometric framework instead of using Gabor wavelet transform because Gabor wavelet transform is not capable to represent high-dimensional features.

Chapter 3

Dataset

3.1 Facial Expression Recognition (FER2013)

This dataset [41] consists of 35,685 grayscale images of faces of facial expressions and the dimension is 48x48 pixels. The dataset has been organised using the search results of Google based on each individual emotion and synonyms of the emotion. Moreover, the images of this dataset are labeled based on the seven basic emotions such as happy, neutral, sad, angry, surprise, disgust, fear which is shown have been shown in table 3.1. Also, It has three different files of images namely public training, public test and private test. To illustrate, there are 28,709 labeled images in the training set, 3,589 labeled images in the public test set, and 3,589 images in the private test set. The following Figure 3.1 illustrates some sample of the FER2013 dataset.

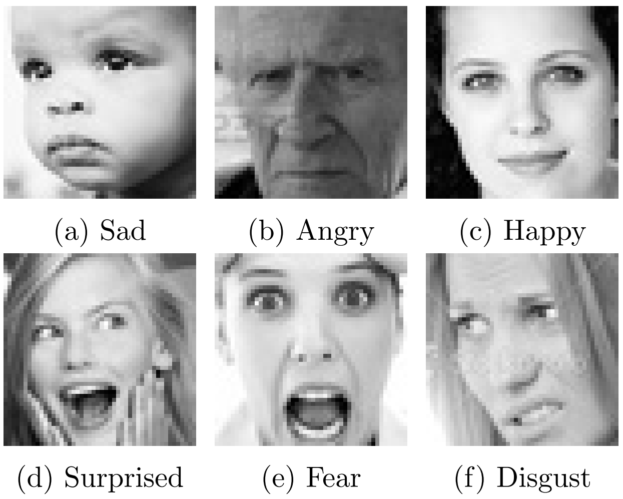

(a) Sad (b) Angry (c) Happy
(d) Surprised (e) Fear (f) Disgust

Figure 3.1: Sample images of FER2013 Dateset

	Happy	Angry	Disgust	Sad	Fear	Surprised	Neutral
Number of Images	8989	4953	547	6077	5121	4002	6198
Label in Dataset	3	0	1	4	2	5	6

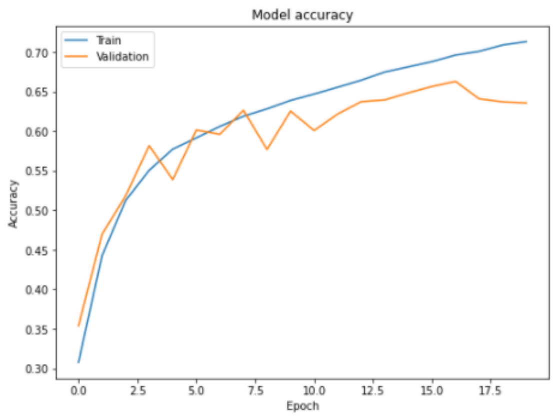
Table 3.1: Dataset Table of sample classes of FER2013

Chapter 4

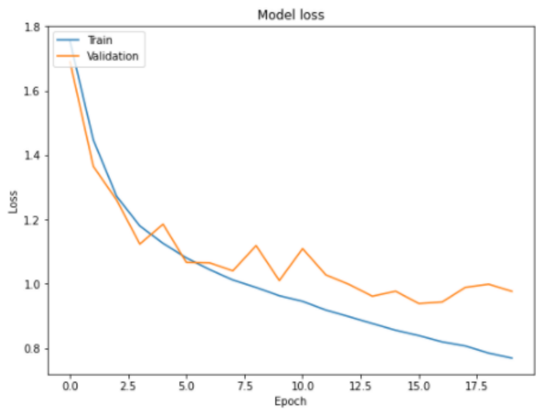
Implementation of Existing Model Architectures

4.1 Baseline Model

Here, we have used the FER dataset to train the model and we set the input dimension of the baseline model to $48 \times 48 \times 1$. At first, we created a sequential model and added a convolution layer having a dimension of $5 \times 5 \times 64$. After that, relu activation function has been used in each layer. We added a second convolutional layer having the same dimension and same value for both activation function and padding. Moreover, we have applied batch normalization function to reduce the amount of shifting values of the hidden units. Then we have applied max pooling and have added another 2 convolutional layers with the dimension of $5 \times 5 \times 128$. Then we have applied batch normalization and max pooling function having pool size of 2 in both dimensions. After adding the first four layers; two more convolutional layers with dimensions of $3 \times 3 \times 256$ have been added, also batch normalization. Moreover, a flattening layer has been added. We will use a dropout value of 0.2 and also add a dense layer having 7 nodes. Lastly, we have used the softmax activation function. After constructing the model, while compiling we have used categorical_crossentropy loss function value and adam optimizer. Our baseline model has gained 71.21% accuracy in training set and 64.47% accuracy in the validation set which has been shown in Figure 4.1.



(a) Baseline accuracy

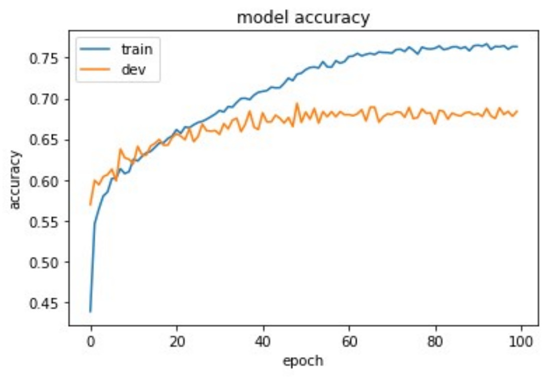


(b) Baseline loss

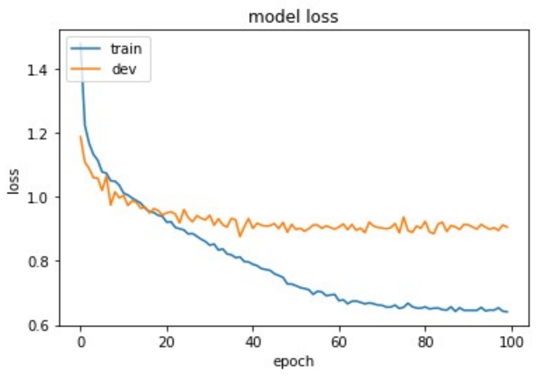
Figure 4.1: Accuracy and Loss Graph for Baseline Model Architecture

4.2 VGG 16

While implementing the VGG-16, we have used two various types of architecture. In the first implementation, we have not used any learned weight hyperparameters. The VGG-16 model has 16 weight layers and the input dimension is 224×224 RGB image. We have also implemented another variant of VGG-16 model using the learned weights. Also, we have replaced the output layer of VGG-16 with 3 fully connected layers having sizes of 512, 256, 128 respectively and a softmax output layer having 7 basic emotion classes. The batch size has been set to 64. Moreover, we have used the adam optimizer having a learning rate of 0.00001, beta_1 is 0.9 and also 0.999 as beta_2 value in both of the models' architecture.
To begin with, we have used these two types of models on the FER dataset. From the graphical representations of the Figure 4.2, the model of Figure 4.2 (a) has shown lower validation accuracy than the training accuracy indicating overfitting problem and the loss and accuracy become steady after completing 10 epochs. We have achieved 69.6% accuracy while using transfer learning.



(a) Vgg 16 accuracy with Transfer Learning

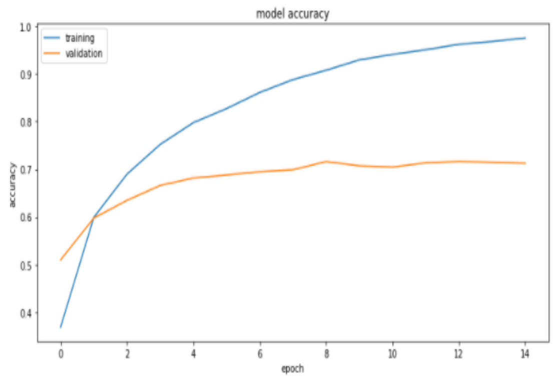


(b) Vgg 16 loss with Transfer Learning

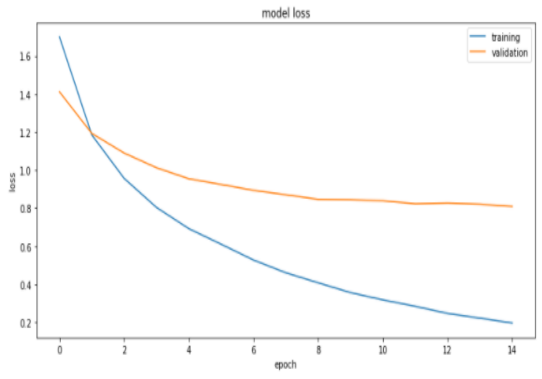
Figure 4.2: Accuracy and Loss Graph for Vgg 16 Transfer Learning

4.3 VGG 19

We have applied VGG-19 network architecture on the FER dataset. In the VGG-19 model, there are 19 weight layers. The input image size is 224×224 and the color channel is 3 indicating RGB images. In model 6, we have implemented VGG-19 architecture with ImageNet weight for transfer learning and we have maintained all pre trained layers frozen. Moreover, we have replaced the output layer of VGG-19 with 2 fully connected layers having sizes of 512 and 256 respectively and a softmax output layer having 7 basic emotion classes. Here, we have used FER



(a) Vgg 19 accuracy with Transfer Learning



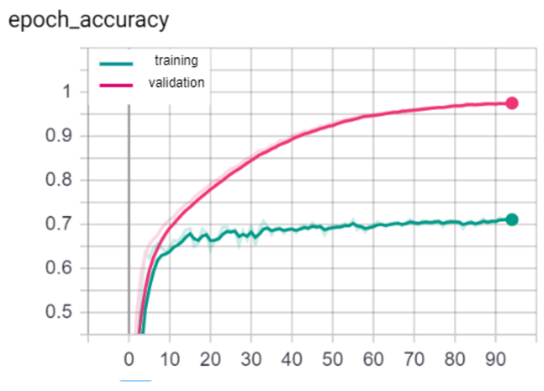
(b) Vgg 19 loss with Transfer Learning

Figure 4.3: Accuracy and Loss Graph for Vgg 19

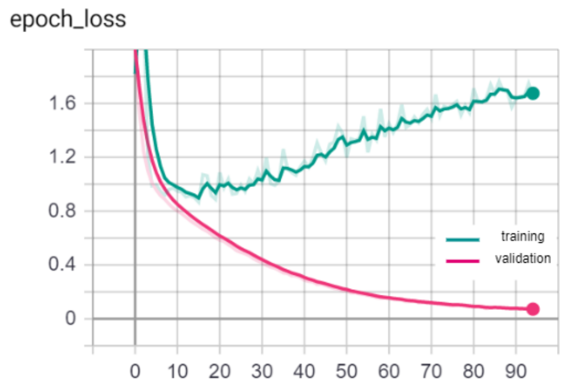
dataset to train the VGG-19 architecture and have applied the transfer learning methodologies in both models. It can be visualized from Figure 4.3 that both of the models have indicating overfitting problems. Doing hyperparameter tuning and transfer learning the model has achieved 94.04% accuracy on the training dataset and 70.03% validation accuracy.

4.4 SeNet-50

To begin with, Using a variant of residual network, SeNet which consists of 50 layers and we have applied SGD as the learning optimizer having a learning rate of 0.01 and batch size has been fixed to 128. Furthermore, the input image dimension is 197×197×3. We have trained this network using the FER dataset for 100 epochs which has been shown in Figure 4.4 and have achieved 71.48% validation accuracy.



(a) SeNet50 accuracy

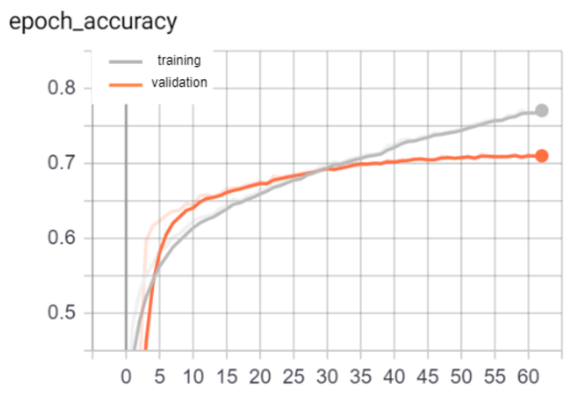


(b) SeNet50 loss

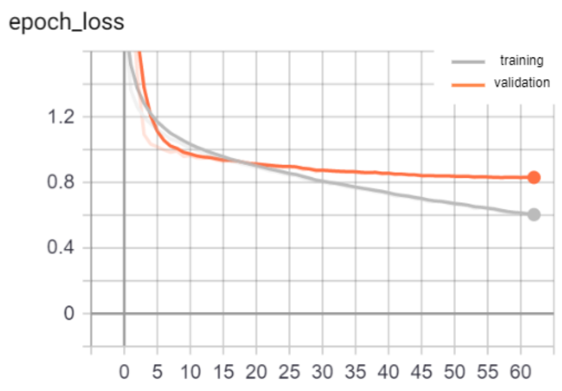
Figure 4.4: Accuracy and Loss Graph for SeNet50

4.5 Resnet50

The state-of-the-art paper which we have been focusing on has achieved 71.06% accuracy on validation set and 86.15% training accuracy on the FER dataset. However, by tuning hyperparameters we have achieved 72.11% accuracy on validation set and 82.44% training accuracy as represented on Figure 4.5. We have successfully increased the accuracy of both training and validation and most importantly we have reduced the overfitting problem which has been present in the existing state-of-the-art paper of ResNet50.



(a) Resnet50 accuracy

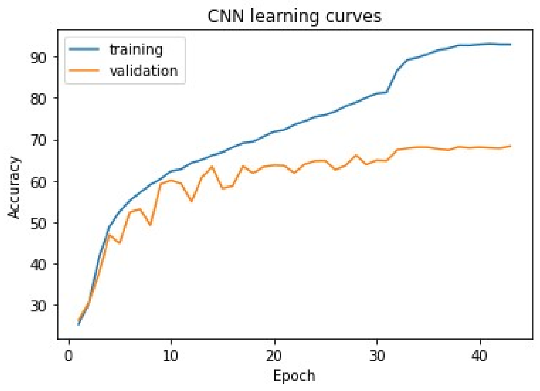


(b) Resnet50 loss

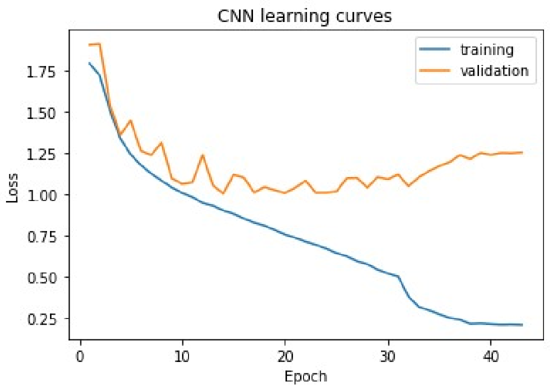
Figure 4.5: Accuracy and Loss Graph for Resnet50

4.6 SeResNext101

In Figure 4.6 displays the training and validation accuracy of the SeResNext101 model. The model has achieved an accuracy over 86.45% for the training set and over 70.85% accuracy for the validation set. It can be observed that the validation accuracy has become stable after completing 35 epochs. The graph has demonstrated the loss curve for the SeResNext101 architecture. From the graph, it can be seen that the loss curve for validation data set has been stable from 15 epochs to 30 epochs. After that, it has slightly increased.



(a) Seresnext 101 accuracy

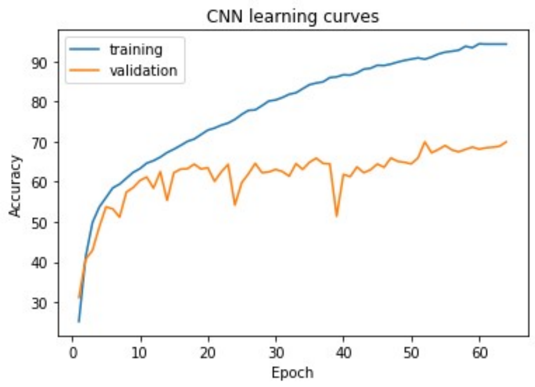


(b) Seresnext 101 loss

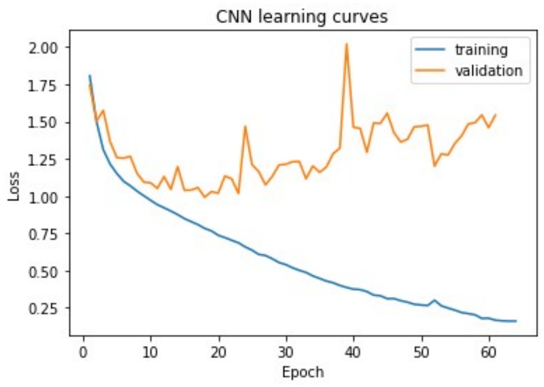
Figure 4.6: Accuracy and Loss Graph for Seresnext 101

4.7 SeResNet34

The graphical representation Figure 4.7 shows the training and validation accuracy of the SeResNet34 architecture. The overfitting problem is present as the training accuracy is 93.4% but the validation data set has gained 71.1% accuracy. Also, the graph shows the loss curve for both training and validation sets. However, the validation score is promisingly high compared to other models.



(a) Seresnet 34 accuracy

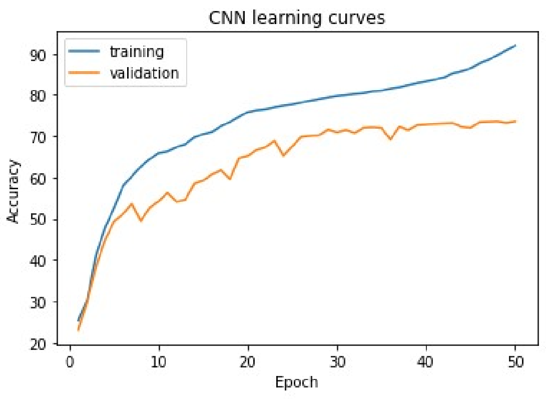


(b) Seresnet 34 loss

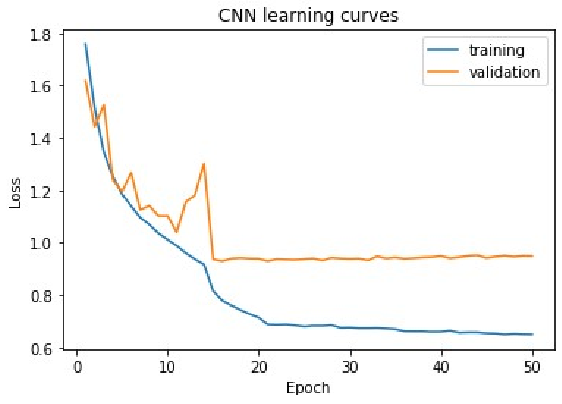
Figure 4.7: Accuracy and Loss Graph for Seresnet 34

4.8 Densenet121

This diagram of Figure 4.8 has shown the training and validation accuracy of the DenseNet. Here, the overfitting problem is small in extent.The training accuracy has been achieved over 90% and for the validation accuracy it has been about 70%. The validation accuracy has increased significantly from 10 epochs to 35 epochs. Moreover, the validation accuracy has become stable after passing 35 epochs. This graph has illustrated the loss curve for the both training and validation set of the DenseNet. Here, it can be seen that the loss curve of the validation set has become stable after completing 15 epochs and finally loss has been lower than 0.9.



(a) Densenet121 accuracy

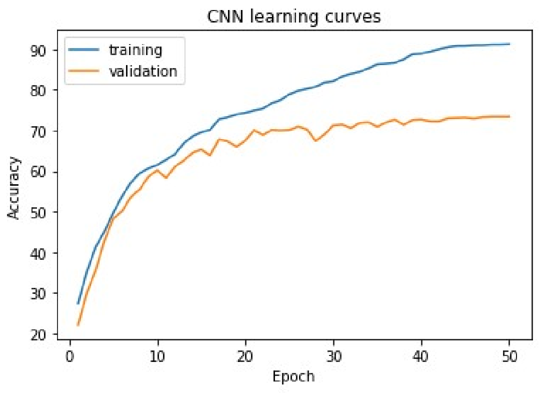


(b) Densenet121 loss

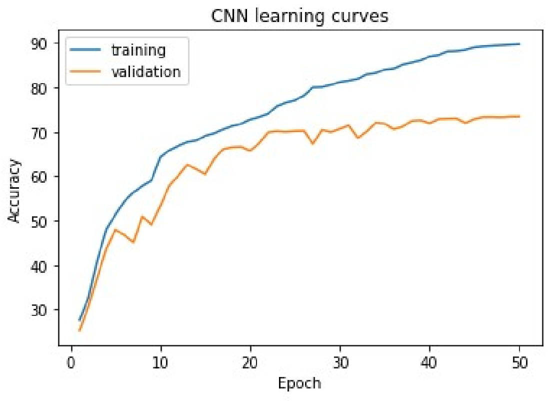
Figure 4.8: Accuracy and Loss Graph for Densenet121

4.9 BAM & CBAM

The BAM net has achieved training accuracy over 91.28% and validation accuracy about 73.42%. It can be seen from the graphical representation that, after 25 epochs the model has started showing overfitting. From 30 epochs to 50 epoch, the validation accuracy does not increase significantly; it mostly remains stable.



(a) BAM accuracy



(b) CBAM accuracy

Figure 4.9: Accuracy Graph for BAM & CBAM

Moreover, The graph has displayed the training and validation accuracy of CBAM net. From the graph, it can be seen that the training accuracy has been 89.75% and it has crossed over 73% for the validation accuracy which has been visualized in Figure 4.9. For the validation accuracy, the curve has become stable after completing 25 epochs.

Chapter 5

Proposed Method

In the existing residual network paper, the authors have proposed two variants of residual blocks while constructing the ResNet architecture. From the illustration of the basic block in Figure 5.1, we can see that there are two consecutive 3×3 Convolutional layers in each residual block. However, in our proposed architecture, we have used split-attentional block which has been named as CAU block instead of using the traditional 3×3 convolutional layers inside each basic block of ResNet34.

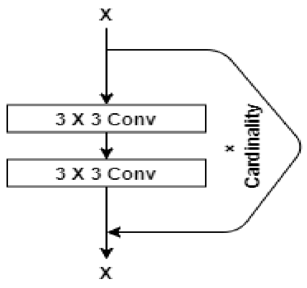


Figure 5.1: Resnet34 Block

After that, we have applied masking on the output of each layer's residual operation. Then, an element wise multiplication has been applied between the input and the output of the masking layer as shown in Figure 5.2. In Figure 5.3, we have illustrated our model's architecture and in the following section each block has been explained in detail.

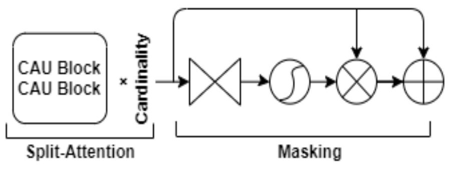


Figure 5.2: CAMnet Block

Attention mechanism has been used in our proposed architecture because it has the capability to focus on the important regions of the input. The rest of the parts of the input will not be considered while learning the features by the network. Also, the attention mechanism has the capability to differentiate the background and foreground of the input image by doing the operation shown in Equation 5.1 and as a result this mechanism reduces the computational complexities and can increase the performance significantly.

$$H_{i,c}(x) = (1 + M_{i,c}(x)) * F_{i,c}(x) \tag{5.1}$$

The value of $M(x)$ can be between 0 and 1, and when $M(x)$ is 0, $H(x)$ will estimate the original features $F(x)$. This is the procedure of attention residual learning. In the following sections, the architecture of our proposed model Convolutional Attentional Masking Network and its building blocks which are CAU block and Masking block have been stated sequentially.

5.1 Convolutional Attentional Masking Network

In the existing original paper, the authors have proposed two variants of residual blocks while constructing the ResNet architecture. From the illustration of the basic block and bottleneck block in chapter 2, in our proposed architecture, we have used split-attentional CAU block instead of using the traditional 3×3 conv layer inside each basicblock of ResNet34. After that, we have applied masking on the output of each layer. Then, element wise multiplication has been applied between the input and the output of the masking layer. In Figure 5.3, we have illustrated our models architecture and in the following section each block will be explained in detail.

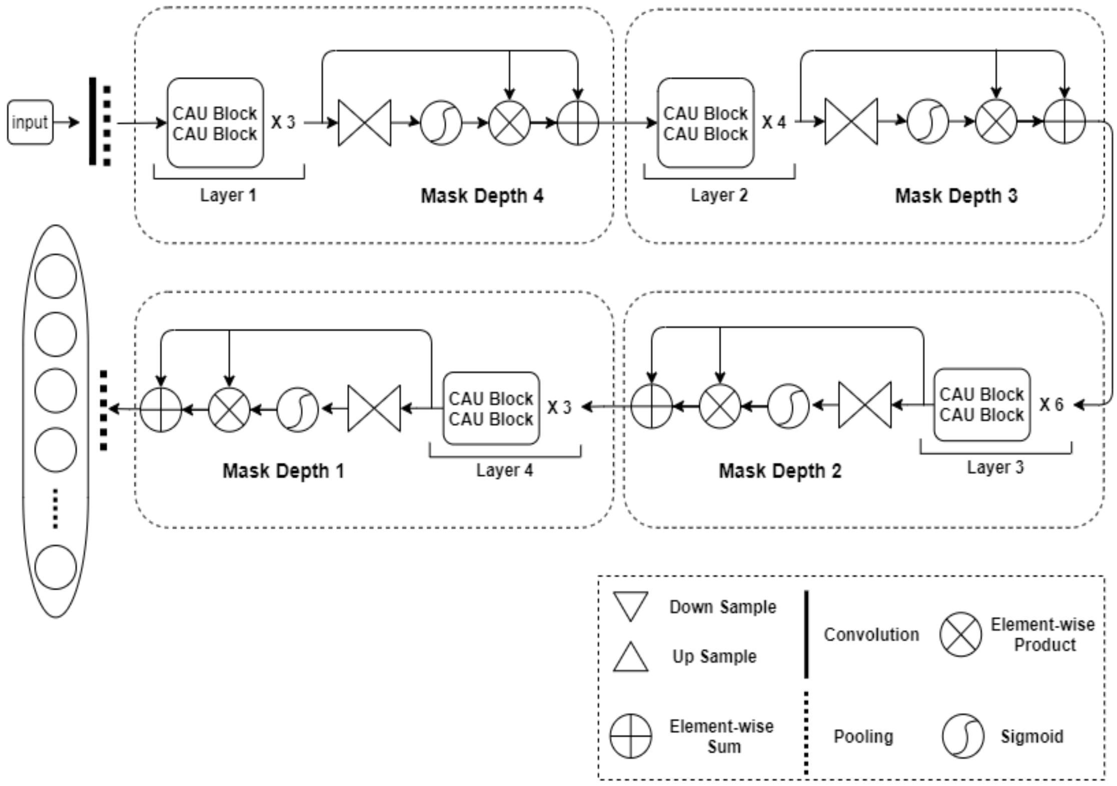


Figure 5.3: Convolutional Attentional Masking Network (CAMnet)

5.1.1 Convolutional Attention Unit (CAU)

The Split-Attention block has been considered as a computational unit which consists of a feature map group and split attention operations. This method enables feature-map attention across different feature-map groups.
The features can be divided into multiple groups and the number of feature map groups is specified by cardinality hyperparameter K. The resulting feature-map groups are considered as cardinal groups.

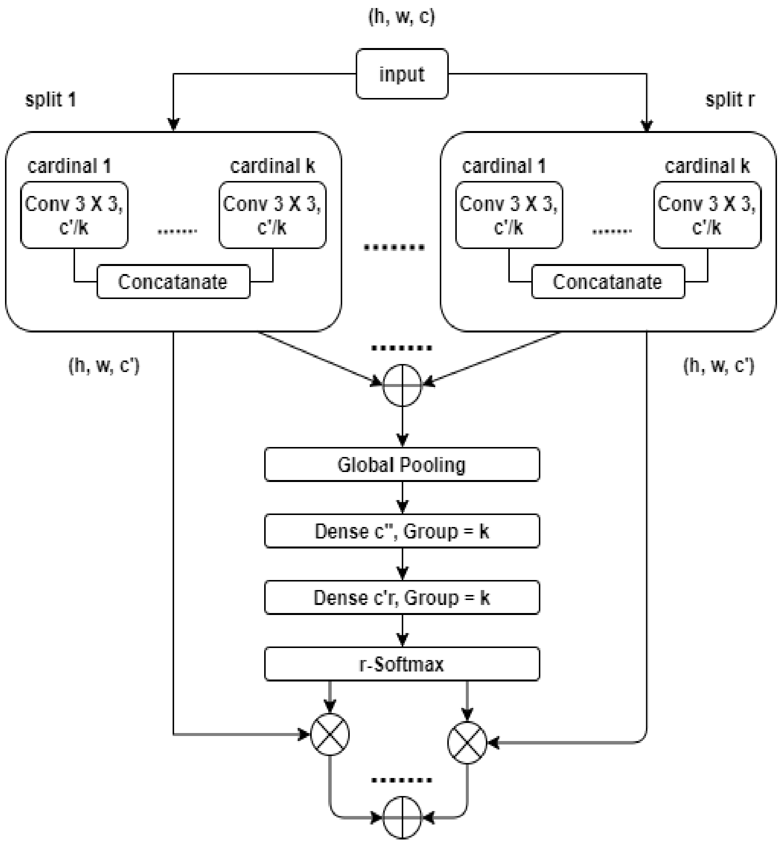


Figure 5.4: CAU Block

At first, as stated in Figure 5.4, the Convolution operation has been applied on the input features having the input channel value of 64, value of radix at 2, kernel size 3, padding and stride both having value of 1 and lastly the value of groups which consisting the number of cardinalities; 3,4,6 and 3 sequentially (ResNet34). After that, batch normalization and relu activation function have been applied. Subsequently, split operation has been applied on the result from the relu activation function consisting of two other parameters; number of filters divided by radix and dimension. All the splitted values have been summed up together and stored in a variable. Then, adaptive average pooling has been applied and the value has been passed through a fully connected layer. The features have been learned during this process. After that batch normalization and relu activation function have been applied.

Furthermore, the variable which has stored the summation of the splitted values, have been passed through another fully connected layer. Again the features have been learned in this step. After that, rsoftmax has been applied. Then, the split module has been applied again consisting of two parameters; number of filters have been divided by radix and dimension having value of 1.
In the end, element wise multiplication has been applied on the two previously calculated splitted values and summed together to store the value in a variable.

5.1.2 Masking Block

Masking Block is motivated by the attention mechanism from bottom up - top down (or the encoding - decoding) mechanism which has achieved many successes in estimating human posture and segmentation problems.
To begin with according to Figure 5.5, the Masking Block consists of two basic components: Encoder and Decoder. The Encoder, with the input feature block, learns the features at multiple levels and also pooling layers will be added to reduce the spatial dimension of the block features. Subsequently, the convolution blocks will quickly increase the depth of the feature to increase the ability of latent representation. After reaching the lowest resolution, the global feature block will be extended by the decoder block symmetrical to the Encoder block. The spatial size will be gradually increased through the convolutional layer to ensure the output size is equivalent to the input. Then a normalization function will be responsible for the normalization of output.
Block Masking Block has skip connection, inspired by U-net network, to combine the features in the Encoder block and Decoder block to increase localisation, here to increase the weighting of features.

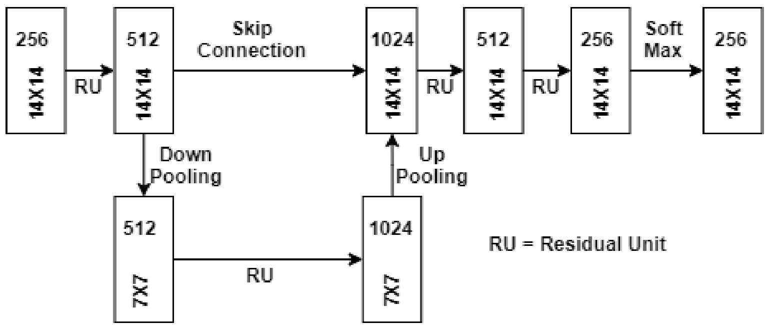


Figure 5.5: Masking depth 2

There are 4 different types of masking block based on the depth size and the depth of masking decreases as the model goes deeper. Firstly, for first layer the masking depth is set to 4 where the input filter size is 64 and the the generated output filter will be also 64. However, there will be some down-sampling and up-sampling operation to extract lower level and group specific image details. After that, for layer 2 we will have a masking block of depth 3 as shown in Figure 5.6 where the 2 down pooling operation will be done. In the first $28 \times 28 \times 128$ input tensor as residual operation will be done to do down sampling twice and their output on each level has been passed to each corresponding up pooling level as skip connection to pass the higher level image feature. In Figure 5.5, a masking block of depth 2 has been illustrated. To illustrate, we can see that the input tensors size is $14 \times 14 \times 256$ and after doing one residual operation a downpooling has been done. After that, before doing a up sampling another residual block has been added.

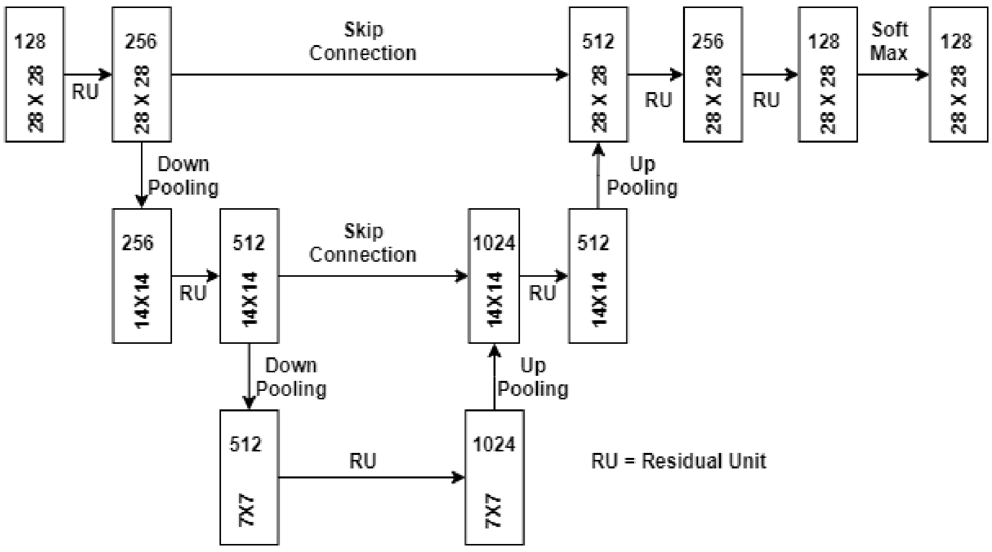


Figure 5.6: Masking depth 3

From the above illustration of the Figure 5.5 & 5.6 we can see that after doing down sampling and up sampling each block has been passed to a softmax function before passing it to the output. Softmax is mostly used and performs better for multi classification problem. Hence, we have used it here. Also, as softmax scales all the values between 0,1 it either scales close to 0 or close to 1. Therefore, the output tensor of the softmax results in a segmented image which focused on the important region that the block has learned. As a result, we can omit the unnecessary part while training and only focus on the important region while training our model with lower complexity.

To sum up, the inputs have been passed through the conv layer and then batch normalization has been applied. After that, relu activation function and max pooling have been applied. The output then feeds to the CAU block. After applying a masking operation on the output, element wise multiplication has been applied between the input and the output of the masking layer. Moreover, this split-merge-multiplication method has been executed for the next three layers. Finally, our proposed architecture ends with an average pooling layer and a flatten layer and the output has been passed through a 7 way fully connected layer.

Chapter 6

Experimental Analysis and Discussion

6.1 Experimental Setup

6.1.1 Data Pre-processing

For the FER2013 dataset, we have converted the csv dataset to png. Also, as the dataset is of 48×48 grayscale image. Hence, for the purpose of transfer learning and deep neural network training we have converted the images to RGB by cloning the grayscale pixel values in 3 color channel space. However, as the input samples are 48×48 it can be only trained with a shallow network if we want to avoid overfitting problems. With the current size of the images when the images will be fed to a deeper layer, the image details will be lost, vanishing gradient problem will arise and noises will be added to the cost function. Therefore, we tried to convert our dataset into a minimum size in which we will have maximum possible image details and will be able to train our dataset for transfer learning.
There are multiple algorithms to resize an image using interpolation techniques like bilinear, nearest, bicubic, area, lanczos, gaussian, mitchell cubic. For interpolating the pixels into a larger space, bilinear and lanczos performs better than other algorithms. However, bilinear gives some blurry effect and we lose image details on the edge. Hence, we have used lanczos and in Figure 6.1 both results are shown.

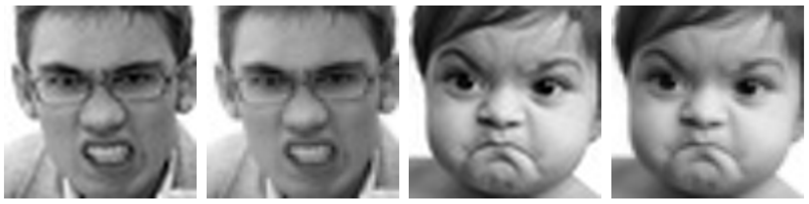

(a) Lanczos (b) Bilinear (c) Lanczos (d) Bilinear

Figure 6.1: Result image from resizing algorithm

6.1.2 Data Augmentation

In this work we are using different deep learning architectures like VGGNet, ResNet, Inception. For all the architectures the data augmentation in the training set has

rescale = 1/255
featurewise center = False
featurewise std normalization = False
rotation range = -30,30
width shifting range = 0.5
height shifting range = 0.5
horizontal flip = True

Table 6.1: Augmentation Parameter

been kept the same as shown in table 6.1. Also, using image data loader we have resized our dataset into 224×224×3.

6.1.3 Regularization Techniques

Deep neural networks contain the capability of memorizing any sort of data. During training, the models accuracy on the training set usually converges upward while its learning performance does not improves on the test set. This phenomenon has been known as overfitting.

Regularization L2

Overfitting problem is one the most common issue for small datset like FER2013. Therefore, the primary step to tune the model could be adding weight decay. It includes some bias to the cost function of each dimension which penalizes the parameters. Furthermore helps to generalize the test data which is the new data properly as stated in Equation 6.1.

$$Err(m,n) = Loss(m,n) + \sum_i \theta_i^2 \tag{6.1}$$

Here, in the equation the θ is a vector containing all the network parameters.

6.1.4 Dropout

As the deep learning architectures need to be trained over millions of parameters where each layer is connected. So, it is quite common that the huge amount of parameters can easily overfit to the training set. Therefore, one of the most used regularization techniques that has been used is Dropout [42]. It means randomly setting a portion of each of the layers activation to 0. When training, neurons need to learn better representations to keep the gradient stable. During testing, to compute the prediction and Dropout the neurons have been used.
Dropping out neurons means omitting neurons from the network as well as their weights. In dropout, the choice of units that needs to be dropped is random.

6.1.5 Normalization

To train a model and keep the models gradient in a stable condition all the data need to be scaled in a specific manner which is known as normalization. Normalization techniques are used to reduce exploding gradient problems. Moreover, a normalization is used mostly after performing each layers convolution operation. There are several normalization techniques like Batch Norm, Group Norm, Layer Norm, Instance Norm. However, in this thesis we only focused on Batch Norm and Group Norm to implement our work.

6.1.6 Early Stopping

This method mainly observes if a models performance if degrading such as the accuracy is not improving or the loss is increasing. So that it can stop the training after a certain number of epoch. Therefore, it helps to stop a models training if it start either overfitting or underfitting.

6.1.7 Training Set

By doing data augmentation we can create multiple views of a batch of images rather than having a single view that (i.e rotating, shifting, flipping images) has proven to have a positive effect in overfitting and the dataset can be easily enlarged. Here a snippet of our training augmentation has been added.

6.1.8 Validation Set

For the validation set we are keeping our sample as they are except normalizing the images with rescale $= \frac{1}{255}$.

6.1.9 Evaluation Criterion

In this thesis, we have evaluated our model's performance using the confusion matrix and the accuracy. The accuracy can derived from the sum of true predictions divided by the total classifications prediction as shown in the Equation 6.2.

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{6.2}$$

Here, TP is considered as true positive, TN is considered as true negative, FP is false positve and lastly FN is false negative.

In the confusion matrix, as we have a 7 class facial emotion recognition problem consisting of classes $TC_1, TC_2, ..., TC_7$, which has created a 7×7 confusion matrix. Here, the rows have represented the actual classes and the columns have represented the predicted classes. Moreover, the values which have situated at the diagonal positions of the matrix are the true positive TP values for the corresponding classes. These values represent the right decisions on the other hand the values which have situated at the off-diagonal position in the matrix represent the misclassified errors. From the table 6.2, it can be seen that PC12 in the following matrix shows that the actual emotion class TC_1 has been predicted as class TC_2.

		TC1	TC2	...	...	TCn
Actual Label	TC1	TP1	PC12	...	...	PC1n
	TC1	PC21	TP2	...	...	PC2n
	...	...	...	...	...	...
	...	...	...	...	...	...
	TCn	PCn1	PCn2	...	...	TPn
	Predicted Label					

*TC = true class, PC = predicted class, TP = true positive

Table 6.2: Confusion Matrix for FER2013 classes

6.2 Result and Evaluation

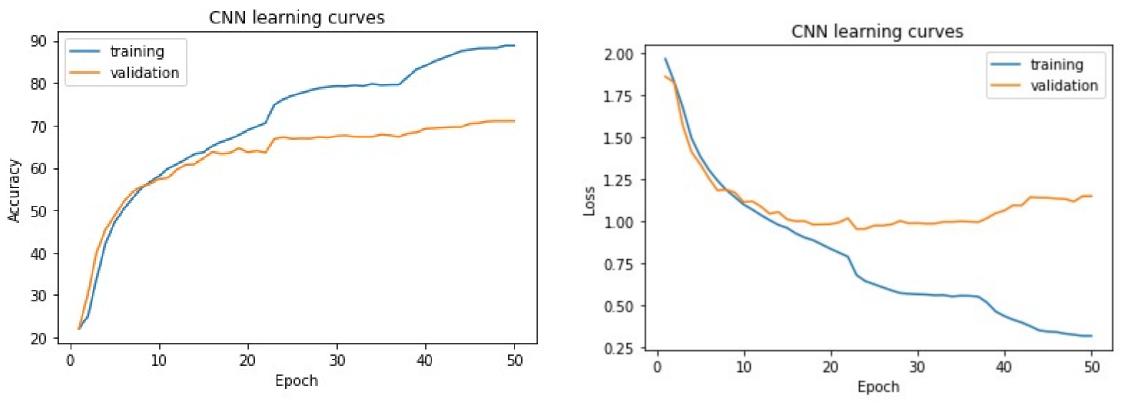
6.2.1 Implementation of CAMnet

We have implemented many optimizers and feature tuning approaches for hyper-parameter tuning. However, 4 types of optimizers shown in Table 6.3, having batch size 32, value of weight decay is 0.0001, num_workers is 4, momentum is set to 0.9 show well balanced result and we trained our model for 50 epochs in all the scenarios. Firstly, the AdaDelta optimizer has achieved 61.71% accuracy while setting the learning rate at 1.0. After that, we have achieved a slightly higher accuracy of 65.18 by using SGD and setting the learning rate value at 0.01. The Adam Optimizer has attained 67.35% accuracy and the learning rate has been set to 0.001. Finally, we have accomplished the highest accuracy which is 70.65% without using any pre-trained weights of ImageNet and by implementing the RAdam optimizer and also the learning rate value has been set to 0.0001.

Optimizer	Batch Size	Learning Rate	Accuracy
AdaDelta	32	1.0	61.71%
SGD	32	0.01	65.18%
Adam	32	0.001	67.35%
RAdam	32	0.0001	70.65%

Table 6.3: CAMnet feature tuning results

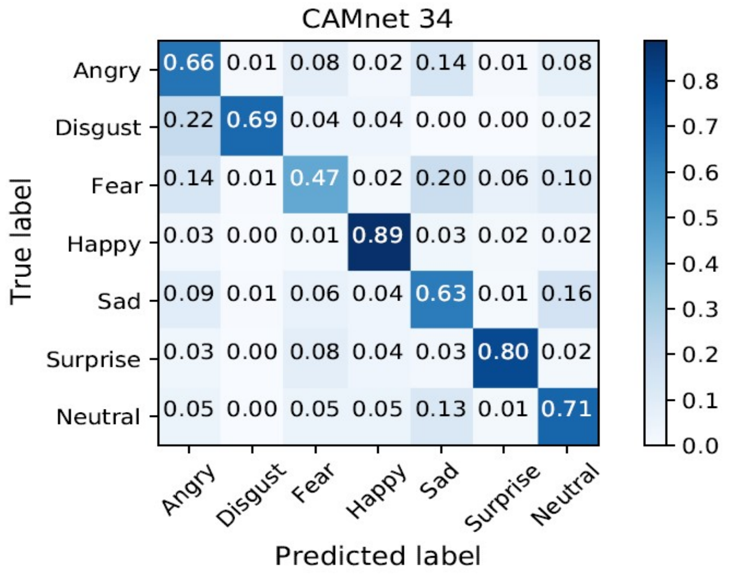
Our proposed CAMnet models performance is visualized by Figure 6.2 which has been trained from scratch without using any pretrained weight. Our model has acquired over 86.41% accuracy on the training set and over 70% accuracy on the validation set. We have used a dropout layer of 50% before final fully connected layer. Moreover, for dropout fine-tuning, 2 fully connected layer one before (512,128) and one after (128,7) of a dropout layer has also been tested. sequentially has also been tested. However, using only on 50% droput shows better result on validation set. Furthermore, the validation accuracy has increased significantly in the range of 15 epochs to 30 epochs. After 30 epochs the validation accuracy has become stable. The graph has shown the training and validation loss based on the epochs



(a) CAMnet accuracy (b) CAMnet loss

Figure 6.2: Learning curve for CAMnet with NPW

for the CAMNet. The loss curve for validation sets has become stable after passing 20 epochs and it has slightly increased after completing 40 epochs. From 20 epochs to 35 epochs the loss curver for validation has been stable. The following Figure 6.3 shows the performance evaluation based on the actual label and the predicted label of our CAMnet model.



	Angry	Disgust	Fear	Happy	Sad	Surprise	Neutral
Angry	0.66	0.01	0.08	0.02	0.14	0.01	0.08
Disgust	0.22	0.69	0.04	0.04	0.00	0.00	0.02
Fear	0.14	0.01	0.47	0.02	0.20	0.06	0.10
Happy	0.03	0.00	0.01	0.89	0.03	0.02	0.02
Sad	0.09	0.01	0.06	0.04	0.63	0.01	0.16
Surprise	0.03	0.00	0.08	0.04	0.03	0.80	0.02
Neutral	0.05	0.00	0.05	0.05	0.13	0.01	0.71

Figure 6.3: Confusion Matrix of CAMnet 34 (ours)

The Table 6.4 has exhibited the accuracy's of various architectures achieved without implementing the pretrained weight of the ImageNet and also our proposed model has outperformed the other existing models. The state-of-the-art Residual Masking Net architecture has achieved 68.18% accuracy not utilizing the weights of the ImageNet. Also, the ResNet34 has obtained an accuracy of 66.70%. Finally, our proposed CAMNet has acquired the highest accuracy of 70.65% which demonstrates better result than all the described models.

Architecture	Accuracy(NPW)
Residual Masking	68.18%
ResNet34	66.70%
CAMnet(ours)	70.65%

*NPW = No Pretrainrd Weight

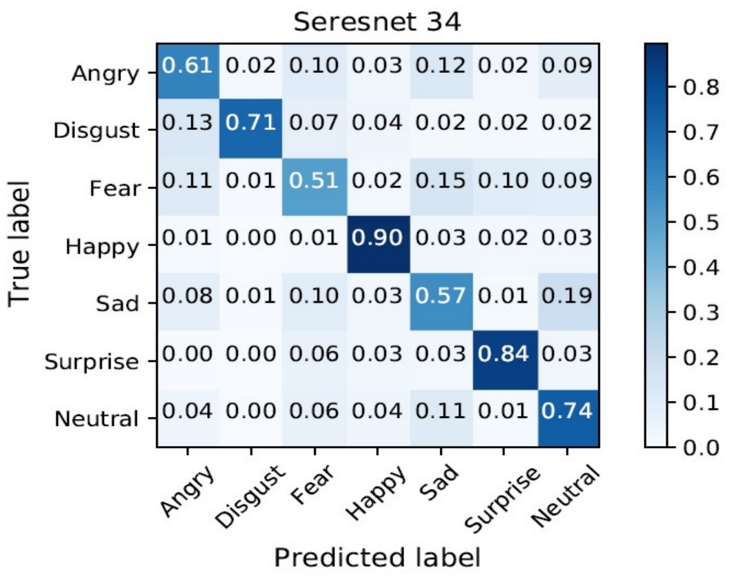
Table 6.4: Comparision with existing resnet34 based architecture

6.2.2 Implementation of Ensemble Model

The authors of the state-of-the-art architectures have achieved an accuracy of 76.82% by ensembling a total number of 7 architectures with pre-trained ImageNet weights. However, we have acquired 76.12% accuracy by ensembling our proposed CAMnet architecture along with 5 other architectures which has been shown in Table 6.5. Our ensemble model has implemented only 6 models and because of the memory constraint, our proposed CAMnet model has not been pre trained with the weights of ImageNet.

SeResNet34 Transfer Learning

We have utilized the SeResNet34 as our pre-trained model. The architecture of SeResNet34 uses bottleneck block and consists of 34 layers. We have replaced the softmax layer which has been set to output the 7 emotions. Also, we have used Radam Optimizer and have executed for 50 epochs having the learning rate of 0.0001 and the batch size has been set to 48. Finally, we have acquired 94.3% training accuracy and 71.1% accuracy on the FER2013 test set. Figure 6.4 illustrates confusion matrix of Seresnet 34 architecture.

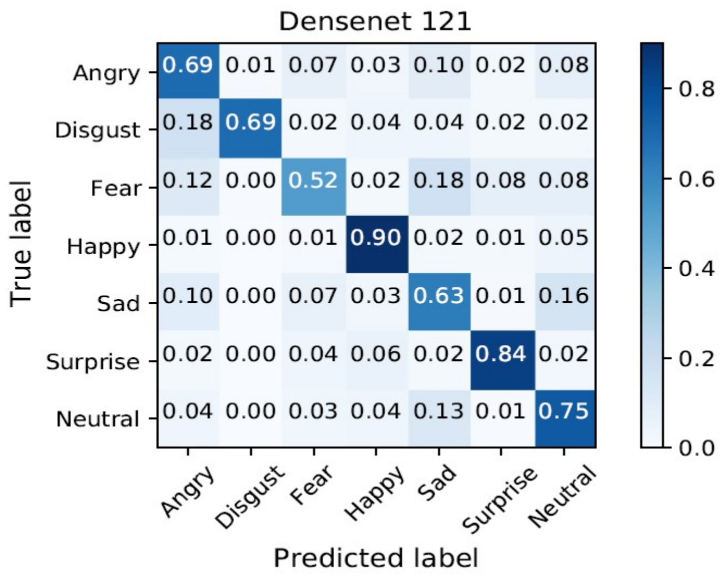


True label \ Predicted label	Angry	Disgust	Fear	Happy	Sad	Surprise	Neutral
Angry	0.61	0.02	0.10	0.03	0.12	0.02	0.09
Disgust	0.13	0.71	0.07	0.04	0.02	0.02	0.02
Fear	0.11	0.01	0.51	0.02	0.15	0.10	0.09
Happy	0.01	0.00	0.01	0.90	0.03	0.02	0.03
Sad	0.08	0.01	0.10	0.03	0.57	0.01	0.19
Surprise	0.00	0.00	0.06	0.03	0.03	0.84	0.03
Neutral	0.04	0.00	0.06	0.04	0.11	0.01	0.74

Figure 6.4: Confusion Matrix of Seresnet 34

DenseNet121 Transfer Learning

We have also used DenseNet121 as another pre-trained model. We have replaced the softmax layer which has been set to output the 7 emotions and have used Radam Optimizer. This architecture has been executed for 50 epochs having the learning rate of 0.0001 and we have achieved a training accuracy of 91.9% and also 73.59% accuracy on the validation sets. The confusion matrix of Densenet 121 yhas been shown in Figure 6.5.



	Angry	Disgust	Fear	Happy	Sad	Surprise	Neutral
Angry	0.69	0.01	0.07	0.03	0.10	0.02	0.08
Disgust	0.18	0.69	0.02	0.04	0.04	0.02	0.02
Fear	0.12	0.00	0.52	0.02	0.18	0.08	0.08
Happy	0.01	0.00	0.01	0.90	0.02	0.01	0.05
Sad	0.10	0.00	0.07	0.03	0.63	0.01	0.16
Surprise	0.02	0.00	0.04	0.06	0.02	0.84	0.02
Neutral	0.04	0.00	0.03	0.04	0.13	0.01	0.75

Figure 6.5: Confusion Matrix of Densenet 121

SeResNext101 Transfer Learning

Moreover we have implemented SeResNext101 as another pre-trained model. After that, we have replaced the actual output layer with a fully connected layer having a size of 1024 and lastly the softmax layer which has been set to output the 7 emotions. We have also used the ImageNet weights to train the model and have used Radam Optimizer and have executed for 43 epochs having the learning rate of 0.0001 and the batch size has been set to 48.Finally, we have acquired 92.92% training accuracy and 70.85% validation accuracy on the FER2013 data set. We can see the evaluation matrix of Seresnext 101 from Figure 6.6.

BAM and CBAM Transfer Learning

Finally, we have loaded BAM and CBAM architectures as our pre-trained models. For both of the architectures, we have replaced the actual output layer with a fully connected layer having a size of 2048 and a softmax layer which outputs to the 7 emotion classes. These both models have been executed for 50 epochs having the learning rate of 0.0001 and the CBAM model has achieved 89.75% training accuracy and the BAM model has achieved 91.28% training accuracy. Most importantly, we have achieved an accuracy of 73.42% for both of the models on the FER2013 test sets. Figure 6.7 and 6.8 respectively visualizes the confusion matrix of BAM and CBAM.

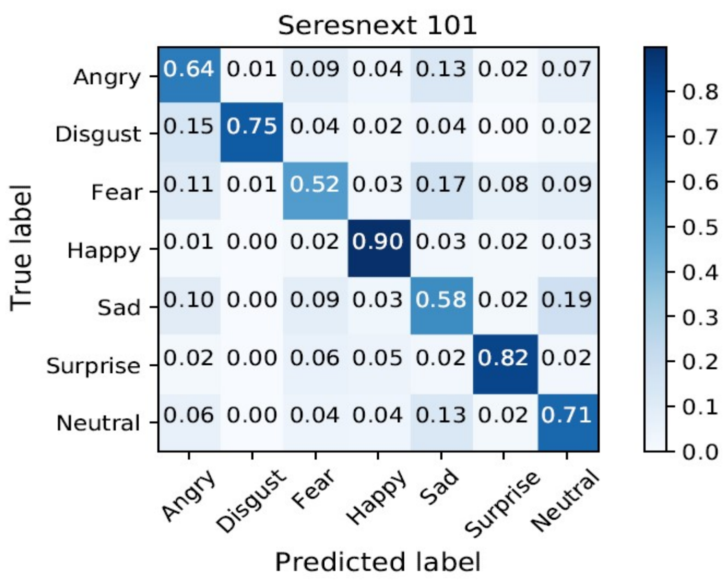


Figure 6.6: Confusion Matrix of Seresnext 101

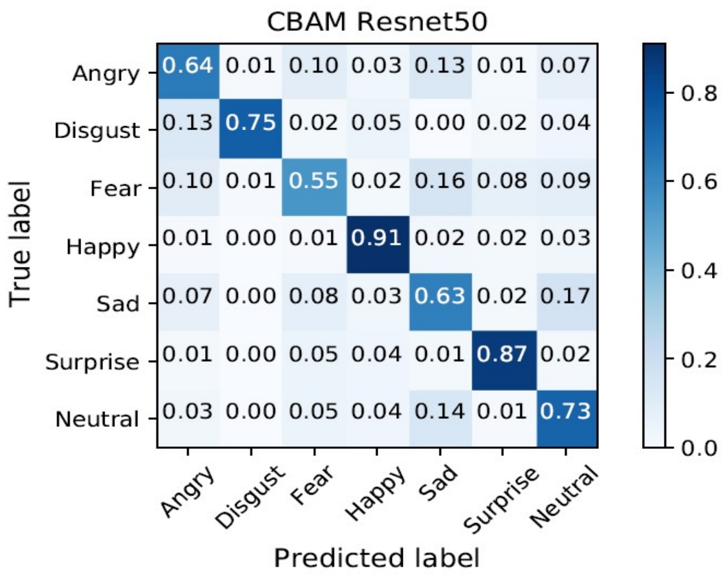


Figure 6.7: Confusion Matrix of CBAM Resnet 50

Test-Time Augmentation (TTA)

We have performed test data augmentation while validating each individual model's performance and our ensemble models' performance. The method is the same as the training time data augmentation but here the model predicts the result for each augmented image and takes the mean of the prediction probability as the result. Moreover, with TTA voting of our ensemble model, we have achieved 76.12% accuracy on our ensemble model which increases the result significantly from 74.93% to 76.12%. From Table **??**, we can see that our ensemble model outperforms all the previous proposed models except Pham et al. [43] which achieved an accuracy of

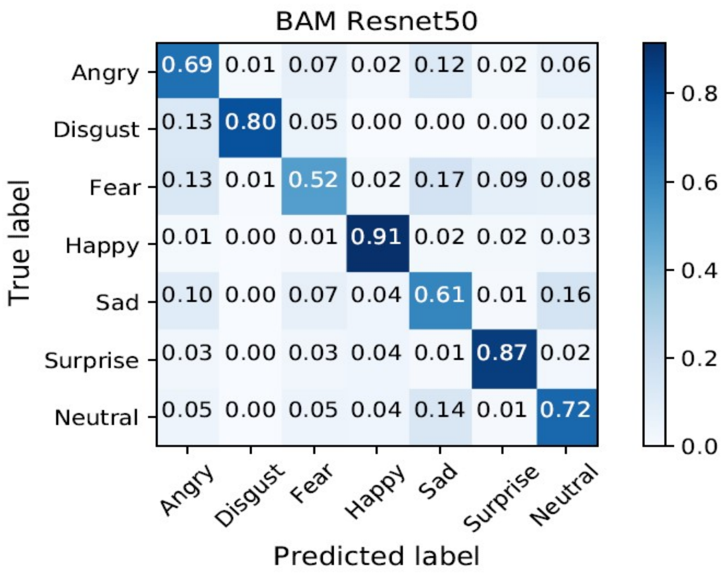


Figure 6.8: Confusion Matrix of BAM Resnet50

Model Name	Training Accuracy	Validation Accuracy	Epoch	Hyper-tuning	Dataset
SeResNet34	94.3%	71.1%	50	WPW	FER2013
CBAM_Resnet50	89.75%	73.42%	50	WPW	FER2013
SeResNextt101	92.92%	70.85%	43	WPW	FER2013
BAM_Resnet50	91.28%	73.42%	50	WPW	FER2013
Densenet121	91.9%	73.59%	50	WPW	FER2013
CAMnet	88.77%	70.65%	50	NPW	FER2013
Ensemble	-	76.12%	-	-	FER2013

*WPW = with pretrained weight; NPW = no pretrained weight

Table 6.5: Performance table of the trained architectures on FER2013

76.82% on validation set. However, we CAMnet model is far deeper than their proposed ResmaskingNet and yet it has shown lower overfitting problem.

Architecture	Accuracy
Vgg16	70.2%
VGG19	71.41%
BAM(ResNet50)	73.21%
CBAM(ResNet50)	73.37%
ResNet50	72.11%
SeResNet34	71.1%
SeResNet50	71.48%
DenseNet121	73.58%
Pramerdorfer et al. [44]	75.2%
Khanzada et al. [45]	75.8%
Pham et al. [43]	76.82%
CAMnet + Ensemble (ours)	76.12%

Table 6.6: Comparison with existing architecture

From Table **??**, we can see that our ensemble model outperforms all the previous proposed models expect Pham et al. [43] which achieved an accuracy of 76.82% on validation set. However, we CAMnet model is far deeper than their proposed ResmaskingNet and yet it has shown lower overfitting problem.

Chapter 7

Conclusion and Future Work

7.1 Conclusion

Facial expressions represent a person's emotions nonverbally. In this thesis, we have introduced a new variant of residual network named Convolutional Attentional Masking Network (CAMnet), our proposed architecture CAMnet shows improved performance without pretrained weight unlike current state-of-the-art models architecture. Moreover, our proposed architecture have successfully reduced overfitting issue on this dataset compared to other existing architecture. Most importantly, we have used only 6 architecture to make an ensemble model by which we have achieved 76.12% accuracy on test set.

7.2 Future Work

Putting aside what we have successfully achieved, there are some limitations that need to be addressed. The main barrier of this research is the constraint of the computational resources such as the limitations of the GPU and the cloud services. Therefore, in this research, we didn't use any pretrained weight of ImageNet dataset on our proposed model. Moreover, there are some matters which need some further improvements. In future studies, we would like to train our proposed CAMnet model on the ImageNet dataset. Also, we will use grid search like algorithms to find the best fitted set of hyperparameters in future which can be done using multiprocessor. After that, we would like to load the weight of the ImageNet dataset on our model to acquire a better performance. Also, we want to utilize auxiliary image data sets to improve the accuracy. Furthermore, different fine-tuning can be done and we will retrain our proposed model on data sets containing occluded images such as human faces wearing glasses.

Bibliography

[1] N. Hussain, H. Ujir, I. Hipiny, and J.-L. Minoi, "3d facial action units recognition for emotional expression," 12 2017.

[2] A. Kołakowska, A. Landowska, M. Szwoch, W. Szwoch, and M. Wróbel, "Emotion recognition and its applications," *Advances in Intelligent Systems and Computing*, vol. 300, pp. 51–62, 07 2014.

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

[4] T. Fang, x. Zhao, O. Ocegueda, S. Shah, and I. Kakadiaris, "3d facial expression recognition: A perspective on promises and challenges," pp. 603–610, 03 2011.

[5] C. Lisetti, "Affective computing -," *Pattern Anal. Appl.*, vol. 1, pp. 71–73, 03 1998.

[6] A. Kapoor and R. Picard, "Multimodal affect recognition in learning environments," pp. 677–682, 01 2005.

[7] T. Jabid, M. H. Kabir, and O. Chae, "Local directional pattern (ldp) for face recognition," in *2010 Digest of Technical Papers International Conference on Consumer Electronics (ICCE)*, pp. 329–330, 2010.

[8] D. G. Lowe, "Distinctive image features from scale-invariant keypoints," *International journal of computer vision*, vol. 60, no. 2, pp. 91–110, 2004.

[9] N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection," in *2005 IEEE computer society conference on computer vision and pattern recognition (CVPR'05)*, vol. 1, pp. 886–893, IEEE, 2005.

[10] T. Ahonen, A. Hadid, and M. Pietikainen, "Face description with local binary patterns: Application to face recognition," *IEEE transactions on pattern analysis and machine intelligence*, vol. 28, no. 12, pp. 2037–2041, 2006.

[11] S. Ebrahimi Kahou, V. Michalski, K. Konda, R. Memisevic, and C. Pal, "Recurrent neural networks for emotion recognition in video," in *Proceedings of the 2015 ACM on International Conference on Multimodal Interaction*, pp. 467–474, 2015.

[12] R. Walecki, O. Rudovic, V. Pavlovic, B. Schuller, and M. Pantic, "Deep structured learning for facial expression intensity estimation," *Image Vis. Comput*, vol. 259, pp. 143–154, 2017.

[13] Y. LeCun *et al.*, "Lenet-5, convolutional neural networks," *URL: http://yann. lecun. com/exdb/lenet*, vol. 20, no. 5, p. 14, 2015.

[14] F. N. Iandola, S. Han, M. W. Moskewicz, K. Ashraf, W. J. Dally, and K. Keutzer, "Squeezenet: Alexnet-level accuracy with 50x fewer parameters and¡ 0.5 mb model size," *arXiv preprint arXiv:1602.07360*, 2016.

[15] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," 2015.

[16] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 1–9, 2015.

[17] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.

[18] "cs231n convolutional neural networks for visual recognition."

[19] S. Sharma, "Activation functions in neural networks," Feb 2019.

[20] W. Research, "Tanh." https://reference.wolfram.com/language/ref/Tanh.html, 1996.

[21] J. Brownlee, "A gentle introduction to the rectified linear unit (relu)," Aug 2020.

[22] H. S, "Activation functions : Sigmoid, relu, leaky relu and softmax basics for neural networks and deep...," Feb 2019.

[23] L. Trottier, P. Gigu, B. Chaib-draa, *et al.*, "Parametric exponential linear unit for deep convolutional neural networks," in *2017 16th IEEE International Conference on Machine Learning and Applications (ICMLA)*, pp. 207–214, IEEE, 2017.

[24] J. Hu, L. Shen, and G. Sun, "Squeeze-and-excitation networks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 7132–7141, 2018.

[25] S. Xie, R. Girshick, P. Dollár, Z. Tu, and K. He, "Aggregated residual transformations for deep neural networks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 1492–1500, 2017.

[26] H. Zhang, C. Wu, Z. Zhang, Y. Zhu, Z. Zhang, H. Lin, Y. Sun, T. He, J. Mueller, R. Manmatha, *et al.*, "Resnest: Split-attention networks," *arXiv preprint arXiv:2004.08955*, 2020.

[27] L. Bottou, "Stochastic gradient descent tricks," in *Neural networks: Tricks of the trade*, pp. 421–436, Springer, 2012.

[28] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[29] M. D. Zeiler, "Adadelta: an adaptive learning rate method," *arXiv preprint arXiv:1212.5701*, 2012.

[30] L. Liu, H. Jiang, P. He, W. Chen, X. Liu, J. Gao, and J. Han, "On the variance of the adaptive learning rate and beyond," *arXiv preprint arXiv:1908.03265*, 2019.

[31] C. R. Rao, "Some comments on the minimum mean square error as a criterion of estimation.," tech. rep., PITTSBURGH UNIV PA INST FOR STATISTICS AND APPLICATIONS, 1980.

[32] P.-T. De Boer, D. P. Kroese, S. Mannor, and R. Y. Rubinstein, "A tutorial on the cross-entropy method," *Annals of operations research*, vol. 134, no. 1, pp. 19–67, 2005.

[33] K. Dembczyński, W. Waegeman, W. Cheng, and E. Hüllermeier, "On label dependence and loss minimization in multi-label classification," *Machine Learning*, vol. 88, no. 1-2, pp. 5–45, 2012.

[34] R. R. Selvaraju, M. Cogswell, A. Das, R. Vedantam, D. Parikh, and D. Batra, "Grad-cam: Visual explanations from deep networks via gradient-based localization," in *Proceedings of the IEEE international conference on computer vision*, pp. 618–626, 2017.

[35] T. S. Li, P. Kuo, T. Tsai, and P. Luan, "Cnn and lstm based facial expression analysis model for a humanoid robot," *IEEE Access*, vol. 7, pp. 93998–94011, 2019.

[36] N. Jain, D. S. Kumar, A. Kumar, P. Shamsolmoali, and M. Zareapoor, "Hybrid deep neural networks for face emotion recognition," *Pattern Recognition Letters*, 04 2018.

[37] Z. Yu and C. Zhang, "Image based static facial expression recognition with multiple deep network learning," pp. 435–442, 11 2015.

[38] M. Zeiler and R. Fergus, "Stochastic pooling for regularization of deep convolutional neural networks," *In: ICLR*, 01 2013.

[39] L. Zhang and D. Tjondronegoro, "Facial expression recognition using facial movement features," *IEEE Transactions on Affective Computing*, vol. 2, no. 4, pp. 219–229, 2011.

[40] Y. Lu, S. Wang, and W. Zhao, "Facial expression recognition based on discrete separable shearlet transform and feature selection," *Algorithms*, vol. 12, p. 11, 12 2018.

[41] I. J. Goodfellow, D. Erhan, P. L. Carrier, A. Courville, M. Mirza, B. Hamner, W. Cukierski, Y. Tang, D. Thaler, D.-H. Lee, Y. Zhou, C. Ramaiah, F. Feng, R. Li, X. Wang, D. Athanasakis, J. Shawe-Taylor, M. Milakov, J. Park, R. Ionescu, M. Popescu, C. Grozea, J. Bergstra, J. Xie, L. Romaszko, B. Xu, Z. Chuang, and Y. Bengio, "Challenges in representation learning: A report on three machine learning contests," 2013.

[42] A. Budhiraja, “Learning less to learn better-dropout in (deep) machine learning,” Mar 2018.

[43] L. P. . T. A. Tran, “Facial expression recognition using residual masking network,” 2020.

[44] C. Pramerdorfer and M. Kampel, “Facial expression recognition using convolutional neural networks: State of the art,” 2016.

[45] A. Khanzada, C. Bai, and F. T. Celepcikay, “Facial expression recognition with deep learning,” *arXiv preprint arXiv:2004.11823*, 2020.