

University of Denver

Digital Commons @ DU

Electronic Theses and Dissertations

Graduate Studies

2020

Facial Action Unit Detection With Deep Convolutional Neural Networks

Siddhesh Padwal

Follow this and additional works at: <https://digitalcommons.du.edu/etd>

 Part of the [Artificial Intelligence and Robotics Commons](#), [Robotics Commons](#), and the [Theory and Algorithms Commons](#)

Facial Action Unit Detection with Deep Convolutional Neural Networks

A Thesis

Presented to

the Faculty of the

Daniel Felix Ritchie School of
Engineering and Computer Science
University of Denver

In Partial Fulfillment

of the Requirements for the Degree

Master of Science

by

Siddhesh Padwal

March 2020

Advisor: Mohammad Mahoor

Author: Siddhesh Padwal

Title: Facial Action Unit Detection with Deep Convolutional Neural Networks

Advisor: Mohammad Mahoor

Degree Date: March 2020

ABSTRACT

The facial features are the most important tool to understand an individual's state of mind. Automated recognition of facial expressions and particularly Facial Action Units defined by Facial Action Coding System (FACS) is challenging research problem in the field of computer vision and machine learning. Researchers are working on deep learning algorithms to improve state of the art in the area. Automated recognition of facial action units has many applications ranging from developmental psychology to human robot interface design where companies are using this technology to improve their consumer devices (like unlocking phone) and for entertainment like FaceApp. Recent studies suggest that detecting these facial features, which is a multi-label classification problem, can be solved using a problem transformation approach in which multi-label problems converted into single-label problem with BinaryRelevance classifier.

In this thesis, convolutional neural network is used as it can go substantially deeper, more accurate, though requires lots of data to train the algorithm. It usually results in a significant feature map obtained from each layer of the network. We introduce Modified DenseNet considering DenseNet as a baseline model. Averaging all the features obtained from each block of DenseNet gives importance to each level of features which can get lost during concatenating the layers in DenseNet and other state of the art classification models.

Detection of Facial Action Units (AUs) can be determined by selecting threshold for the probabilities obtained by training the Modified DenseNet model. Threshold selection can be done with the help of Matthew Correlation Coefficient. Using Matthew Correlation Coefficient, AU correlation can take into account which was missing for previous studies using BinaryRelevance classifier as it does not consider label's correlation because it treats every target variable independently. Modifying DenseNet model helped to improve results by reusing features and alleviating the vanishing-gradient problem.

We evaluated our proposed architecture on a competitive Facial Action Unit Detection task (EmotioNet) database which includes 950,000 images with annotated AUs. Modified DenseNet obtain significant improvements over the state-of-the-art methods on most of them by comparing with the accuracy and other metrics of evaluation and requiring less computation time as compared to problem transformation methods.

ACKNOWLEDGEMENT

I would like to express my sincere gratitude towards my thesis advisor, Dr. Mohammad Mahoor, for guiding me through every stage of my research work. His encouragement and guidance always motivated me to stay passionately indulged in the research. I was honored to work with him and gain a profoundly enriched experience.

I am very thankful to Professors Goncalo Martins, Yun-Bo Yi and Mohammad Matin for being a part of my Oral Defense Committee. Their valuable insights for my work helped me extensively in ameliorating the work.

I am very grateful to be a part of University of Denver and I am extremely thankful to the Department of Electrical and Computer Engineering at the University of Denver for providing me with the resources and a friendly environment to successfully complete my research work.

TABLE OF CONTENTS

ABSTRACT	ii
ACKNOWLEDGEMENT	iv
LIST OF TABLES	vii
LIST OF FIGURES	viii
ACRONYM	ix
CHAPTER 1: INTRODUCTION	1
1.1 Thesis Outline.....	9
CHAPTER 2: BACKGROUND AND RELATED WORK.....	11
2.1 Deep Learning	11
2.1.1 Neural Network Elements.....	11
2.1.2 Working	14
2.1.3 Activation Functions	15
2.1.4 Backpropagation.....	19
2.1.5 Loss Function	19
2.1.6 Optimizers.....	20
2.2 Types of Deep Learning algorithms	22
2.2.1 Supervised Learning	22
2.2.2 Unsupervised Learning:.....	23
2.2.3 Semi supervised Learning.....	23
2.3 Convolutional Neural Networks.....	24
2.3.1 Convolution Layer.....	25
2.3.2 Padding	26
2.3.3 Rectified Linear Unit (ReLU).....	26
2.3.4 Pooling Layer.....	27
2.3.5 Pre-Trained Networks:	28
2.4 Action Unit Detection with Deep Learning	34
2.4.1 Deep Net	35
2.4.2 DISFA: A Spontaneous Facial Action Intensity Database [26].....	36
2.4.3 Binary Relevance for Multi Label Learning [27]	38
CHAPTER 3: METHODOLOGY.....	41
3.1 Shortcoming of CNN state of the art model.....	41
3.2 Motivation to improve	42
3.3 Ideas to implement.....	43
3.4 EmotioNet Challenge [28]	45

3.4.1	Training data:	45
3.4.2	Challenge Phase:	46
3.4.3	Evaluation:	46
3.5	Approach to solve multi label problem.....	46
3.5.1	Pre-Processing.....	47
3.5.2	Modified DenseNet	49
3.5.3	Threshold Selection.....	54
CHAPTER 4: EXPERIMENTS		56
4.1	Dataset.....	56
4.2	Dataset Split	56
4.3	Model Framework	56
4.4	Pre-Processing	57
4.5	Network Setting and Training [35].....	58
4.5.1	Data and Batchsize	58
4.5.2	Activation function and Loss function	59
4.5.3	Learning Methods and Learning rate	60
4.5.4	Dropout.....	60
4.5.5	Number of Layers.....	60
4.5.6	Number of Epochs.....	60
4.6	Threshold Selection	61
4.7	Evaluation	61
4.8	Testing the data.....	62
CHAPTER 5: RESULTS AND DISCUSSION		64
5.1	VGG16.....	64
5.2	ResNet.....	65
5.3	DenseNet.....	65
5.4	Ensembling Methods	66
5.5	Modified Densenet.....	66
CHAPTER 6: CONCLUSION AND FUTURE WORK.....		68
BIBLIOGRAPHY.....		70

LIST OF TABLES

Table 2-1: Architecture of VGG16 (Source: arxiv)	29
Table 2-2: Architecture of ResNet (Source: arxiv)	30
Table 2-3: Architecture of DenseNet (Source: arxiv)	32
Table 3-1: Architecture of Modified DenseNet	53
Table 5-1: Accuracy for each action units (Accuracy on the test data).....	66
Table 6-1: Evaluation of models based on the metrics (Mean accuracy calculated by taking average of individual AU and F1 by using listed formula).....	69

LIST OF FIGURES

Figure 1-1: Six Universal Expressions (Source:twinklet8)	2
Figure 1-2: Facial Action Coding System (Source: researchgate).....	4
Figure 1-3: Robot based Autism Therapy (Source: lab).....	6
Figure 1-4: Ryan (Source: dreamface)	8
Figure 2-1: Concept of Neural Network (Source: Pathmind).....	12
Figure 2-2: 6 Neural Network Architecture (Source: Pathmind).....	12
Figure 2-3: Level of feature abstraction (Source: Pathmind)	13
Figure 2-4: Working of Neural Networks (Source: faculty.neu)	14
Figure 2-5: Logistic Activation Function (Source: towardsdatascience)	16
Figure 2-6: Tanh Activation Function (Source: towardsdatascience).....	17
Figure 2-7: ReLu Activation Function (Source: towardsdatascience)	18
Figure 2-8: Types of Deep Learning Algorithm (Source: Viblo)	24
Figure 2-9: Block Diagram of CCN (Source:medium)	25
Figure 2-10: Building Block for ResNet (Source: arxiv)	31
Figure 3-1: Modified DenseNet Block Diagram.....	43
Figure 3-2: Framework for the classification.....	47
Figure 3-3: Block Diagram of Modified DenseNet Model.....	49
Figure 4-1: Preprocessing using Haar Cascade (Source: Krasserm).....	58

ACRONYM

1. Action Units (AU)
2. Facial Action Coding System (FACS)
3. Convolutional Neural Network(CNN)
4. Matthew Correlation Coefficient (MCC)
5. Mean Squared Error(MSE)
6. Binary Cross Entropy (BCE)
7. Categorical Cross Entropy (CCE)
8. Artificial Neural Networks(ANN)

CHAPTER 1: INTRODUCTION

Facial expressions can be defined as changes in the muscles beneath the skin of the face. These movements which can be seen from the face convey the emotional state of an individual. These facial expressions can be voluntarily or involuntarily [1]. Voluntary facial expressions are often socially conditioned and follow a cortical route in the brain. On the other hand, involuntary facial expressions are believed to be natural and follow a subcortical route in the brain. The best example to explain this is smiling for the camera (voluntarily) and smiling at a joke (involuntarily). Thus, Facial expressions are a form of nonverbal communication. It is one of the means to understand an individual's emotion and state of the mind.

Charles Darwin [2] wrote in his 1872 book about the Expressions for the Emotions in Man or Animals that “facial expressions of emotion are universal, not learned differently in each culture.” The most notable research into the topic came from psychologist Paul Ekman, who pioneered research into emotion recognition in the 1960s. His team of researchers gave hand on pictures of faces showing different emotional expressions to test subjects.

The test subjects then had to determine the emotional states they saw in each photo, based on a predetermined list of possible emotions they had seen prior. Through

these studies with the help of the test subject's majority agreement about the emotional state of the photo, Expressions Ekman found to be universal includes those indicating happiness, disgust, anger, sadness, surprise and fear. From this study, the six basic emotions were proposed. In Figure 1-1, these 6 universal basic emotions are mentioned.

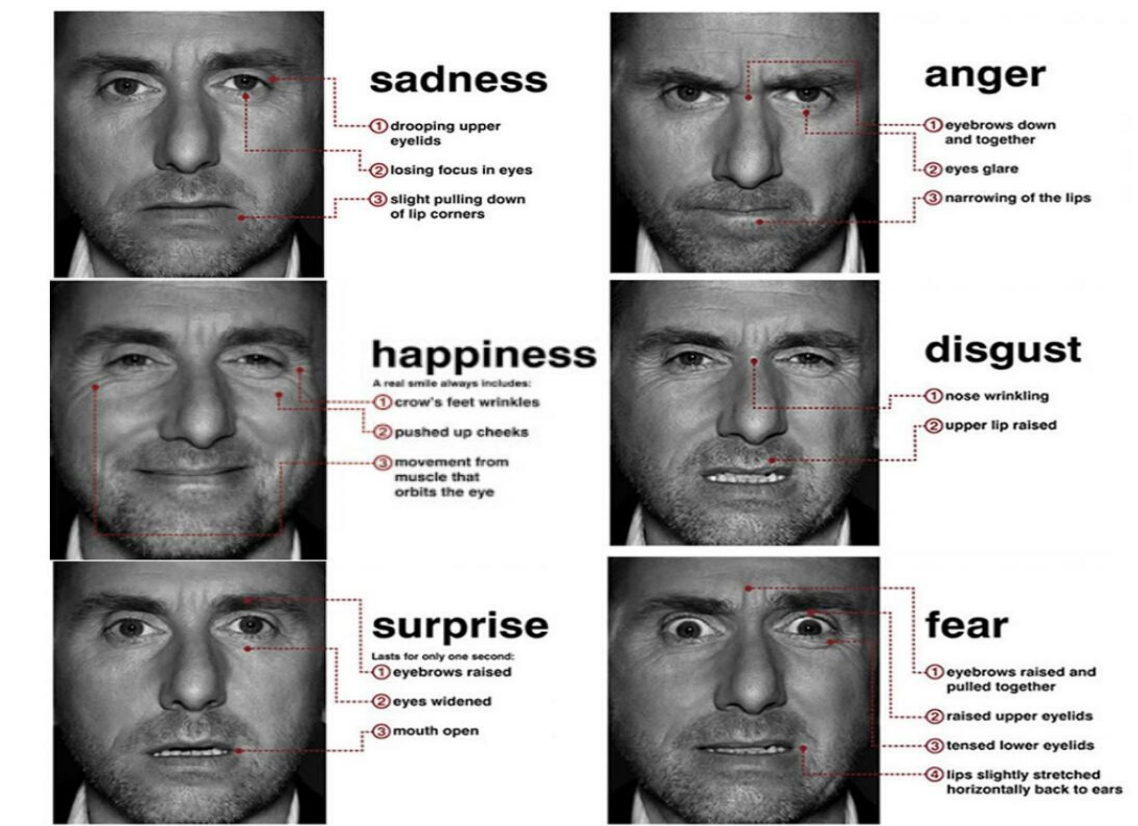


Figure 1-1: Six Universal Expressions (Source:[twinklet8](#))

However, guilt, jealousy, pride, shame which we genuinely feel these emotions, we do not express it clearly and it is difficult to identify such emotions and label them by easily glancing at any photos. Thus, you cannot just label these six emotions to any other state of an individual's expression. Sometimes humans go to mixed feelings which are difficult to

understand by just seeing them. It is important to pay attention to the minute details of the face and this is the reason why facial action units come into the picture [3].

The Facial Action Coding System (FACS) refers to a set of facial muscle movements that correspond to a displayed emotion. Ekman and Friesen developed the Facial Action Coding System (FACS) [4] for describing facial expression by action units. Of 44 FACS AUs that they defined, 30 AUs are anatomically related to contractions of specific facial muscles. 12 are Upper face and 18 are lower face. AUs can occur either singly or in combination. In Figure 1-2, 12 Upper face and 18 lower face action units are mentioned.

By keeping track of each action unit whether they are present or not on the face of individuals, we can reach the final conclusion about their emotions. The study of action units becomes really important in various fields after realizing few applications we stated below.

1. Counseling and determining client's medical state
2. During healthcare, determining patients feeling and comfort level about the treatment
3. In the case of autism, struggling to interpret expressions
4. In the case of e-learning, study the emotions and adjust the learning technique and presentation according to the style of learner
5. Determining fatigue in the case of driving and alerting in advance

6. When person is scared and withdrawing money, ATM machine not dispensing money

























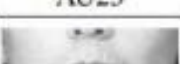
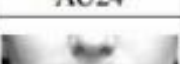



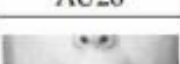
Upper Face Action Units					
AU1	AU2	AU4	AU5	AU6	AU7
					
Inner Brow Raiser	Outer Brow Raiser	Brow Lowerer	Upper Lid Raiser	Cheek Raiser	Lid Tightener
*AU41	*AU42	*AU43	AU44	AU45	AU46
					
Lip Droop	Slit	Eyes Closed	Squint	Blink	Wink
Lower Face Action Units					
AU9	AU10	AU11	AU12	AU13	AU14
					
Nose Wrinkler	Upper Lip Raiser	Nasolabial Deepener	Lip Corner Puller	Cheek Puffer	Dimpler
AU15	AU16	AU17	AU18	AU20	AU22
					
Lip Corner Depressor	Lower Lip Depressor	Chin Raiser	Lip Puckerer	Lip Stretcher	Lip Funneler
AU23	AU24	*AU25	*AU26	*AU27	AU28
					
Lip Tightener	Lip Pressor	Lips Parts	Jaw Drop	Mouth Stretch	Lip Suck

Figure 1-2: Facial Action Coding System (Source: [researchgate](#))

Social Robots

A social robot is an autonomous robot that interacts and communicates with humans using artificial intelligence to decide how to act on information received through cameras and other sensors. When the robot is communicating with humans, it is really

important to capture human's facial expressions to predict the emotions behind their actions or words so that the robot can respond to it accordingly. The important aspect of this implementation is when a person is interacting with a social robot, a person should feel much more like they are interacting with someone rather than something.

The ability to respond in ways that seem lifelike has been achieved with the study of social and emotional intelligence. Advancement in Artificial Intelligence (AI) has opened ways to interpret humans by designing algorithms that allow robots to recognize voices, faces and emotions; interpret speech and gestures; respond appropriately to complex verbal and nonverbal communications which includes Action unit detection.

There has been so much research going on around the world where interaction with humans enhances human life in a positive way. One of the examples of this interaction is that the researchers found that RUBI (social robot) learned to use information about the children's facial expressions to accurately predict their preferences for different activities. They have verified this with the number of human judges agreeing on the same prediction which social robot did.

The personal robot can be programmed to catch depression in human beings early by monitoring day to day changes in someone's behavior and help with simple interventions, like music and video, for people in need of social therapies. This can be really helpful to the people who are suffering from depression or similar kind of mental disease and could not tell the society about it or could not understand by themselves.

Socially assistive robots also show promise in providing therapy to children with autism. Research has found that engaging with therapy robots increases engagement, attention and novel behaviors (such as spontaneously imitating the robot) among children with autism. Figure 1-3 shows the analysis of behavior of children with ASDs with the help of robots [5].



Figure 1-3: Robot based Autism Therapy (Source: lab)

In our lab, under the guidance of Prof. Mahoor, a fully autonomous robot has been developed which can talk, listen, express, understand and remember things which we set for reminders. This social robot named Ryan, interacts with any age of the person and keeps the person engaged in the conversation by making people comfortable with the responses. Figure 1-4 shows the social robot named Ryan [6].

The **goal of AU detection** can be achieved by predicting AU labels correctly for each image of the facial expression. Human face with complex facial expression requires a better classifier so that it can catch each minute features to determine the correct expression. Improvements in computer hardware and network structure have given access to the researchers to build and train truly deep Convolutional Neural Networks (CNNs) for the betterment of the results. Various convolution neural networks applied and tested to get the best feature map for AU detection before reaching to the proposed Modified DenseNet model. We started with VGG16 [7], ResNet [8], Inception ResNet [9] and using ensemble method [12] to include both ResNet and Inception ResNet in one algorithm [11].

As testing all these models to the ‘EmotioNet’ dataset, we faced the shortcomings of the respective models which cause problems of vanishing gradient descent, too many parameters results into over-fitting of the model and the computational time required for training. Stochastic depth shortens ResNet by randomly dropping layers during training to allow better information and gradient flow. With the motivation of dropping layers idea and to create short paths from early layers to later layers, we found out that DenseNet model [10] performs really well as compared to other classifiers. We have discussed about it later in literature review.

Dense connections have a regularizing effect, which reduces overfitting on tasks with smaller training set sizes. With the DenseNet motivation, in this thesis, we propose a new Convolutional network to generate the best feature map for detecting the action unit. For the detection of AU properly we need each and every minute detail possibly. Thus, considering all the features we generate from each block of the Convolutional network is

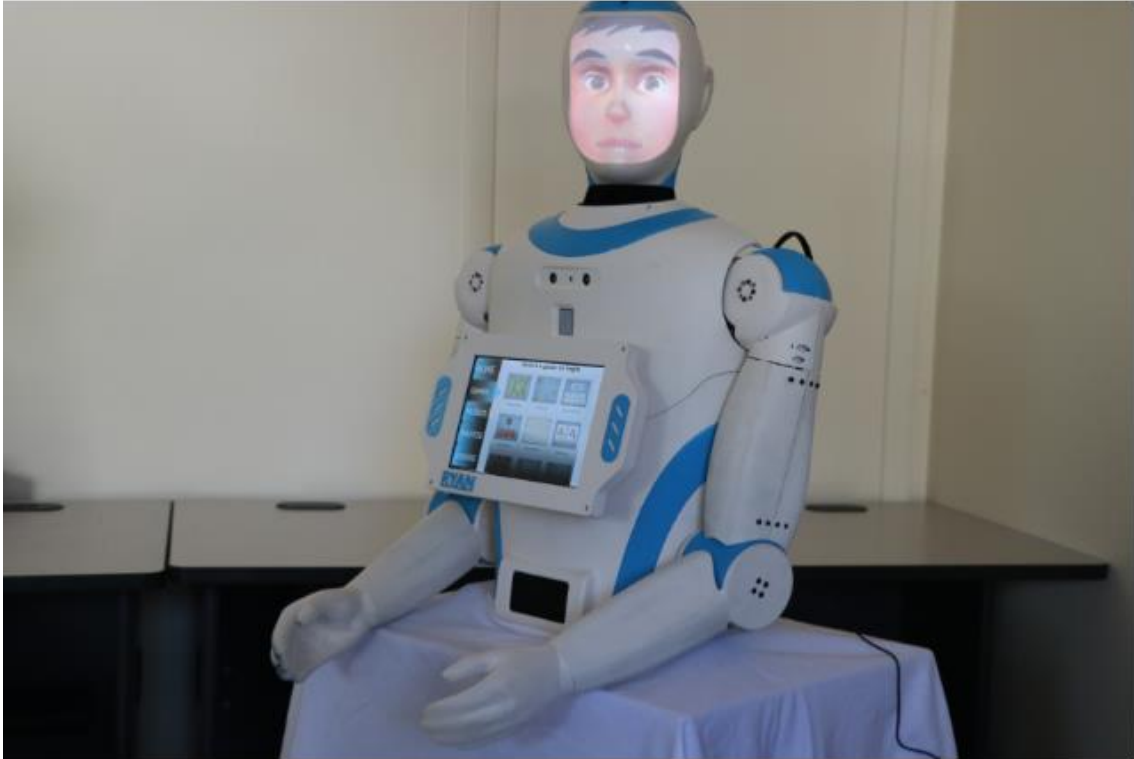


Figure 1-4: Ryan (Source: dreamface)

important. In DenseNet, feature maps get reduced across each transition layer between two dense blocks. With the network going deeper, dense blocks increase as a result of which feature maps reduced at each step of transition layers. To consider all the features, in my proposed Modified DenseNet model, Average of all the dense block output considered and created feature map to predict the AU detection.

After training the data with Modified DenseNet model, threshold selection method helped to determine whether AUs are present or not. Sigmoid activation function with binary cross entropy loss provides the probabilities for each of the AU's. Matthew Correlation Coefficient (MCC) method decides the threshold considering AU correlation

amongst each other. Comparing the threshold with the probabilities from the convolutional network, AUs are set and reset.

We evaluated proposed models on highly competitive ‘EmotioNet’ dataset and we significantly outperform the current state-of-the-art results on the basis of accuracy metric.

1.1 Thesis Outline

The overview of this thesis is as follows:

- **Chapter 2: Background and Related Work**

We discuss in depth how CNN works and all the previous works in the field of Classifying multi label problem with Binary Relevance approach and other Pre-trained models including DenseNet which is the motivation of our proposed network for multi label classification.

- **Chapter 3: Methodology**

This chapter discusses the approach of multi label classification problem we have implemented with the proposed Modified DenseNet model.

- **Chapter 4: Experiments**

To perform well on test data, this chapter describes the parameters and methods implemented to achieve best results.

- **Chapter 5: Results and Discussion**

This chapter presents the results obtained by performing different models and the proposed modified DenseNet model including comparison with the state of the art by evaluating performance metrics.

- **Chapter 6: Conclusions and Future Work**

This chapter summarizes the contributions of the work in the thesis and briefly discusses the different directions that can be explored to improve the performance of the task.

CHAPTER 2: BACKGROUND AND RELATED WORK

2.1 Deep Learning

Deep Learning is a field inspired by the structure and function of the brain called artificial neural networks (ANN). Deep learning is a process of learning structures extracting from the data which can be numerical, images, sound, text or time series. Structures are extracted through a number of successive layers which differ based on the dataset. Patterns can be learned for these structures by feeding data into the ANN and training them [13].

2.1.1 Neural Network Elements

Deep learning is all about networks composed of several layers, we should understand the working of these layers first. The layers are made of nodes. A node combines input from the data with a set of coefficients or weights, that either amplify or dampen that input. These weights get updated once you start training the network model. These inputs summed up after multiplying with weights. After summing up, inputs are passed through the activation function, which determines whether given node should be activated or not.

Figure 2-1 is a diagram of what one node might look like.

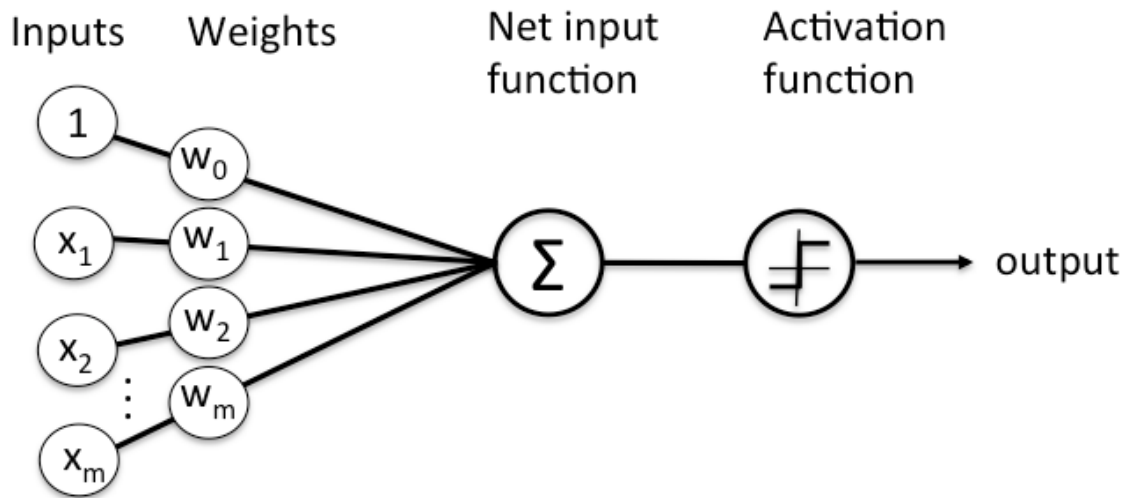


Figure 2-1: Concept of Neural Network (Source: Pathmind)

As we say ANN consists of stacked layers, these layers are stacked up as shows in Figure 2-2.

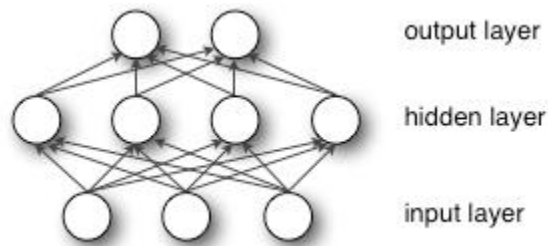


Figure 2-2: 6 Neural Network Architecture (Source: Pathmind)

In Recent years, the depth of the network is increasing which results in more hidden layers. ANN started with single input, output and hidden layer where nowadays networks are going deeper. The important reason behind this is the improvement in the predicting labels of the input. The advantage of multiple layers is that they can learn features at various levels of abstraction. Each layer trains on a distinct set of features based on the previous

Successive model layers learn deeper intermediate representations

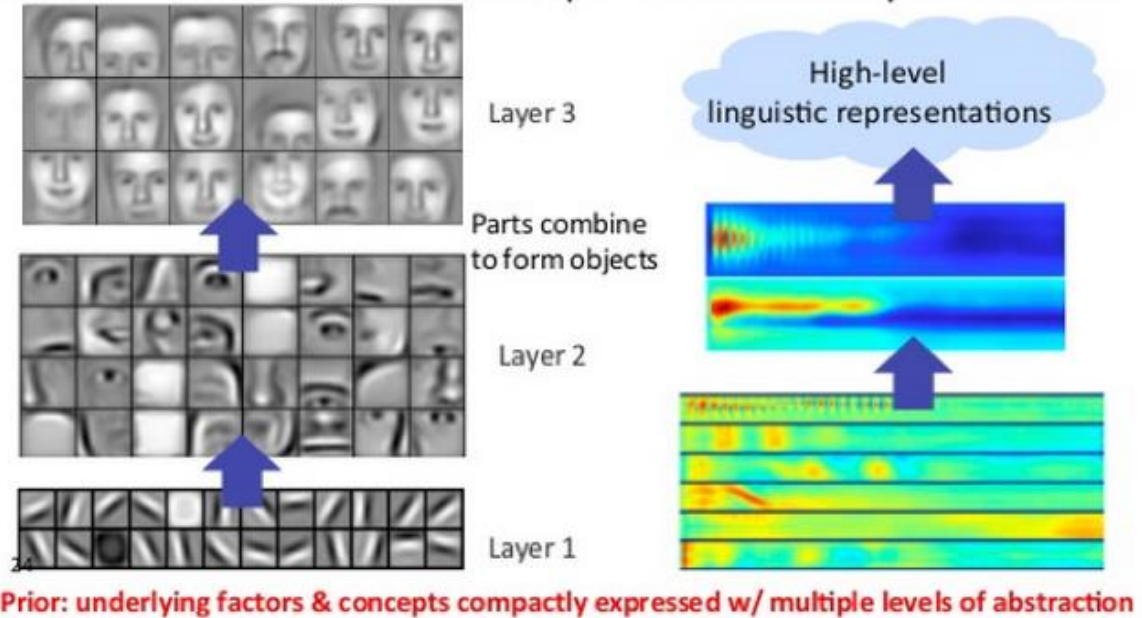


Figure 2-3: Level of feature abstraction (Source: Pathmind)

layer's output. The feature map improves after every layer adding up more information about the feature. In Figure 2-3, level of abstraction is explained with the example.

In Figure, you can see in layer 1, we are detecting basic features like edges. With these features adding information in layer 2 can determine shapes of different features like eyes, nose, etc. Combining this feature map by adding more layers we can predict really complex features like face from the given image. This is known as feature hierarchy in which adding more layers with more number of parameters can help to handle high dimensional datasets [14].

2.1.2 Working

To understand how neural networks work, we need to start with the integral element of the network which is layer.

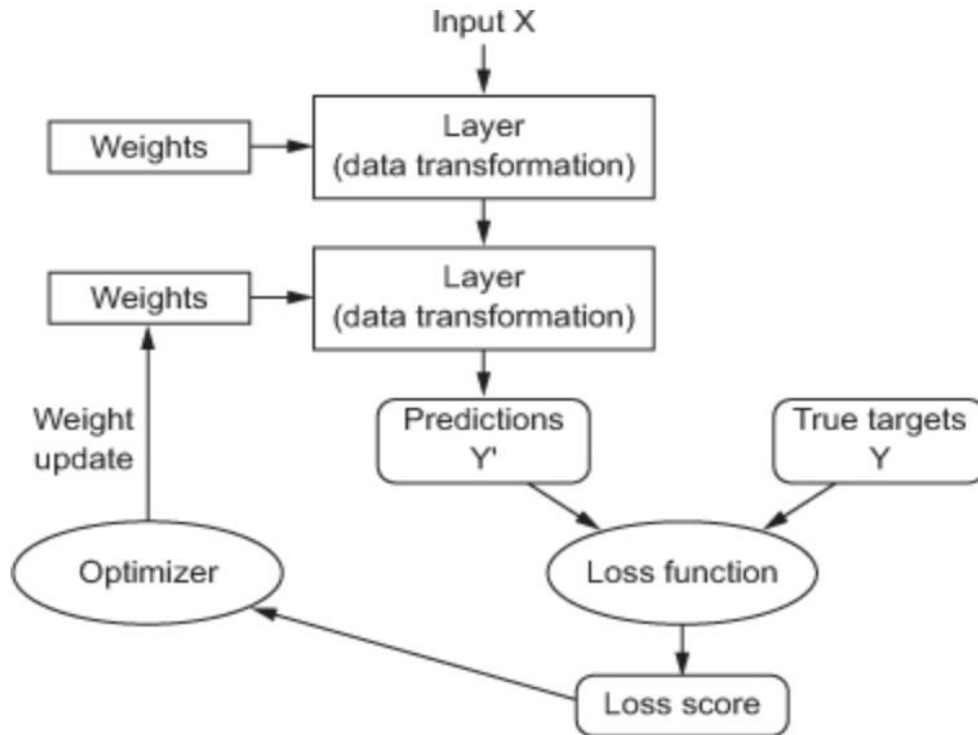


Figure 2-4: Working of Neural Networks (Source: faculty.neu)

These layers are learning essential information from the input data and storing all the information by updating its weights. In this context, learning means finding a set of values for the weights of all layers in a network, such that the network will correctly map example inputs to their associated targets. To see how well the weights are parameterized with respect to actual labels, we need some measure which is called as loss function. The loss function takes the predictions of the network and the true target and computes a

distance score, which determines how well the network is performing for given data points [15]. Figure 2-4 shows how different neural network components are connected to each other.

This distance score works as a feedback signal to the layers so that the value of given weights can be adjusted in a direction that will lower the loss score for the given data points. These weights get updated in the backward direction with optimizers which implement this using Backpropagation algorithm. This process can be carried out till our all parameters get the best values with low distance score. Initially, weights are randomly assigned due to which in the beginning we get output far from the actual targets with the large distance score. By repeating the backpropagation process through training the model results into weights with better value and minimal loss. This is how training the neural network model can actually predict the target we want to achieve.

2.1.3 Activation Functions

The activation function determines the complex relationship between the variables introducing non linearity and gives the output. Without activation function your input gets added similar to a linear regression model which can only address the linear relationship between variables which is not adequate to work with the 3d image data [16][17]. Commonly used activation functions are Logistic, tanh (hyperbolic tangent), ReLu (Rectified linear units)

2.1.3.1 Logistic activation function

It is a “S” shaped curve with equation

$$f(x) = \frac{1}{1+e^{-x}} \quad (1)$$

It ranges from 0 to 1. It is also referred to as Sigmoid activation function. The weighted sum of inputs is applied to it as an input. Figure 2-5 shows the Logistic Activation function Representation.

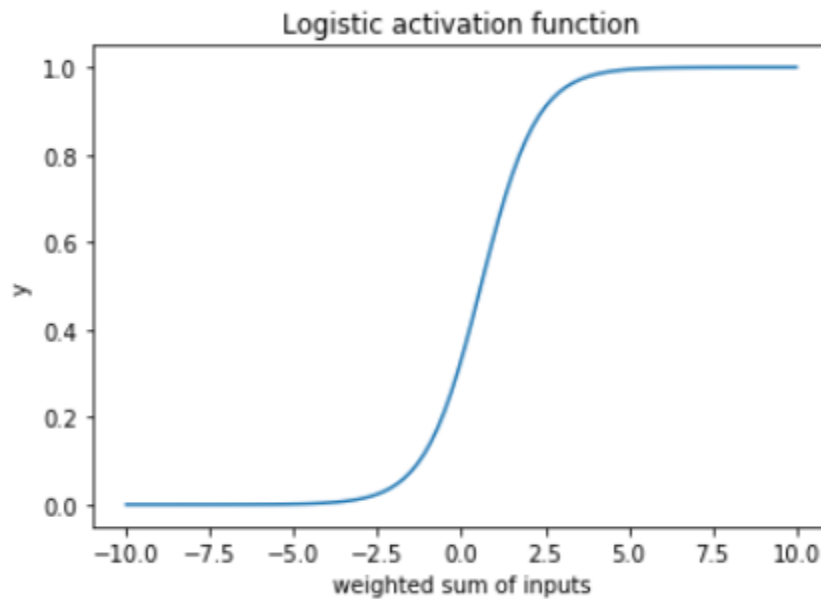


Figure 2-5: Logistic Activation Function (Source: towardsdatascience)

For a large positive input, it results in a large positive output which tends to fire and for large negative input, it results in a large negative output which tends not to fire.

2.1.3.2 Tanh (Hyperbolic tangent)

It is similar to logistic activation function with a mathematical equation

$$f(x) = 2 \times \text{logistic}(2x) - 1 \quad (2)$$

Figure 2-6 represents Tanh activation function graph.

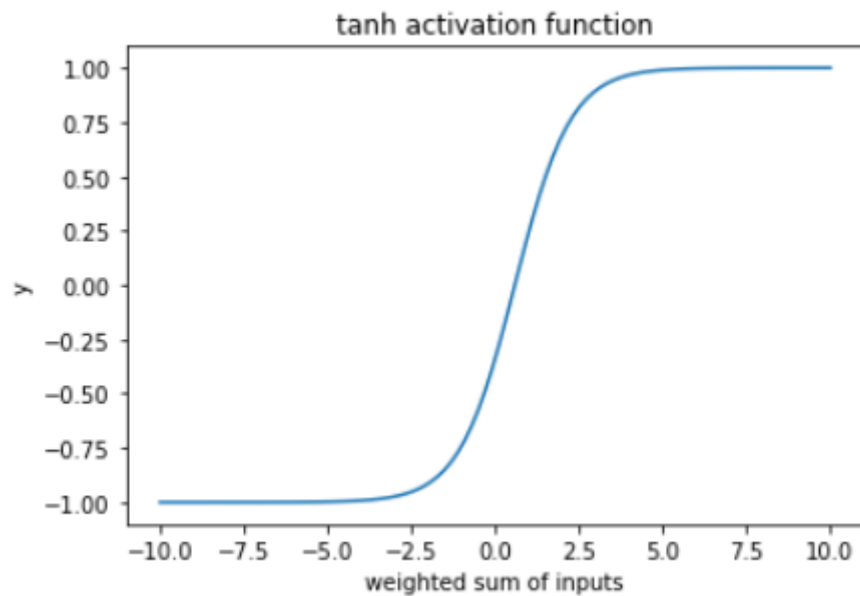


Figure 2-6: Tanh Activation Function (Source: towardsdatascience)

The output ranges from -1 to 1 and having an equal mass on both sides of zero-axis so it is zero centered function. So, tanh overcomes the non-zero centric issue of the logistic activation function. Hence optimization becomes comparatively easier than logistic and it is always preferred over logistic. But still, a tanh activated neuron may lead to saturation and cause vanishing gradient problems.

2.1.3.3 ReLu (Rectified linear units)

It is the most commonly used function because of its simplicity. It is defined as

$$f(x) = \max(0, \sum_{i=1}^n w_i x_i + b) \quad (3)$$

If the input is a positive number, the function returns the number itself and if the

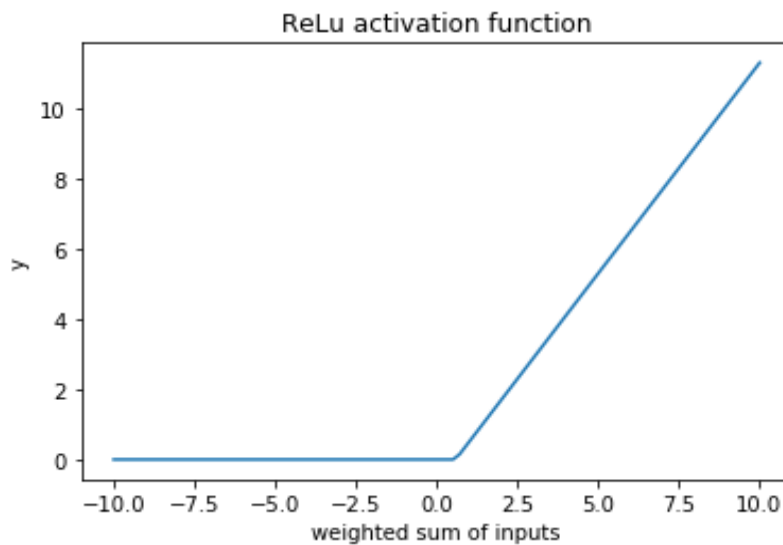


Figure 2-7: ReLu Activation Function (Source: towardsdatascience)

input is a negative number then the function returns 0. In Figure 2-7, ReLu graph representation is shown.

Advantages of ReLu activation function

1. Easy to compute.
2. Does not saturate for the positive value of the weighted sum of inputs.

Because of its simplicity, ReLu is used as a standard activation function in CNN.

2.1.4 Backpropagation

Backpropagation is the mechanism where loss score takes into account as we keep on training the model and update the weights in the backward to decrease the loss score and find the best values for each of the parameters. By doing this we extract best features to predict the labels perfectly in the end.

We change the parameters using optimization algorithms. A very popular optimization method is called gradient descent, which is useful for finding the minimum of a function. Gradient Descent is used to minimize the error which results in a low loss score. This function is also called as loss function.

2.1.5 Loss Function

Loss function helps in optimizing the parameters of the neural networks. The loss is calculated using loss function by matching the target value and predicted value by a model. Then we use the gradient descent method to update the weights of the neural network such that the loss is minimized. I have stated some of the popular loss functions below [18][19].

2.1.5.1 Mean Squared Error (MSE)

Mean squared error is calculated by taking the mean of squared differences between actual(target) and predicted values. MSE is used mostly whenever we must deal with the dataset of regression problems.

2.1.5.2 Binary Cross Entropy (BCE)

If you are using BCE loss function, you just need one output node to classify the data into two classes. The output value should be passed through a sigmoid activation function and the range of output is $(0 - 1)$. BCE is used mostly when we have a dataset with binary classification problems.

2.1.5.3 Categorical Cross Entropy (CCE)

If you are using CCE loss function, there must be the same number of output nodes as the classes. And the final layer output should be passed through a Softmax activation so that each node outputs a probability value between $(0-1)$. CCE is used mostly when we have a dataset with multi class problems.

2.1.6 Optimizers

Once we get the loss score after applying loss function, weights need to get updated to improve loss score further. Finding optimized value for the weights is the goal of the optimizers. Finding optimized values for weights ensure generalization of the model and help for the prediction on the data [20].

2.1.6.1 Gradient Descent

Gradient Descent is the most common algorithm to optimize the weights [21]. Gradient indicates the direction of increase. To find the minimum point in the valley we

need to go in the opposite direction of the gradient and update the weights to reduce loss values.

$$\theta = \theta - \eta \nabla J(\theta; x, y) \quad (4)$$

θ is the weight parameter, η is the learning rate and $\nabla J(\theta; x, y)$ is the gradient of weight parameter θ .

2.1.6.2 Adam — Adaptive Moment Estimation

Another method that calculates the individual adaptive learning rate for each parameter from estimates of first and second moments of the gradients [22].

Adam can be viewed as a combination of Adagrad (optimizer), which works well on sparse gradients and RMSprop (optimizer) which works well in online and nonstationary settings. Adam implements the exponential moving average of the gradients to scale the learning rate. Adam is computationally efficient and has very little memory requirements. Because of these factors I decided to use this optimizer in my project. Adam algorithm first updates the exponential moving averages of the gradient (m_t) and the squared gradient (v_t) which is the estimates of the first and second moment. Hyperparameters $\beta_1, \beta_2 \in [0, 1)$ control the exponential decay rates of these moving averages as shown below

$$m_t = \beta_1 m_{t-1} + (1 - \beta_1) g_t \quad (5)$$

$$v_t = \beta_2 v_{t-1} + (1 - \beta_2) g_t^2 \quad (6)$$

Moving averages are initialized as 0 leading to moment estimates that are biased around 0 especially during the initial timesteps. This initialization bias can be easily counteracted resulting in bias-corrected estimates

$$\widehat{m}_t = \frac{m_t}{1-\beta_1^t} \quad (7)$$

$$\widehat{v}_t = \frac{v_t}{1-\beta_2^t} \quad (8)$$

where \widehat{m}_t and \widehat{v}_t are bias corrected estimates of first and second moment respectively.

Finally, we update the parameter as shown below

$$\theta_{t+1} = \theta_t - \frac{\eta \widehat{m}_t}{\sqrt{\widehat{v}_t + \epsilon}} \quad (9)$$

2.2 Types of Deep Learning algorithms

After Learning through each neural network element and their working, It is really important to decide which type of learning is suitable for our classification problem, let's discuss the types of learning first. Figure 2-8 explains the types of Deep Learning algorithms.

2.2.1 Supervised Learning

Supervised Learning algorithms try to create mapping between input features and target prediction output so that it can predict the correct labels for the new data based on the learned relationships. In Supervised Learning, input feature is fed into the model with the labels and then training the model, loss function tries to minimize the error between

values they predicted (estimated value) and the actual label for the input. Convolutional neural networks, Recurrent neural networks can be used for this learning.

2.2.2 Unsupervised Learning:

In Unsupervised Learning, unlabeled data is trained to learn structure from which features can be extracted. There is no way to measure accuracy as the data is unlabeled. Clustering algorithm is the best example of this type of learning in which patterns can be detected from the structure learned during the training of the data. Identifying the pattern, data can be clustered into groups which help in deriving meaningful insights and describe the data better to the users.

These types of algorithms are useful in cases where the human expert doesn't know what to look for in the data.

2.2.3 Semi supervised Learning

Semi supervised learning is the middle ground between supervised learning and unsupervised learning, where both labeled and unlabeled images are used in the process.

In many practical situations, the cost to label is quite high, since it requires skilled human experts to do that. So, in the absence of labels, few images can be labeled and train the model as we do in Supervised Learning. This trained model is then trained on unlabeled data. These unlabeled data can be labelled with the help of trained models; this process is called Pseudo Labeling. Now full data which includes labeled data and pseudo labeled data can be trained with our model.

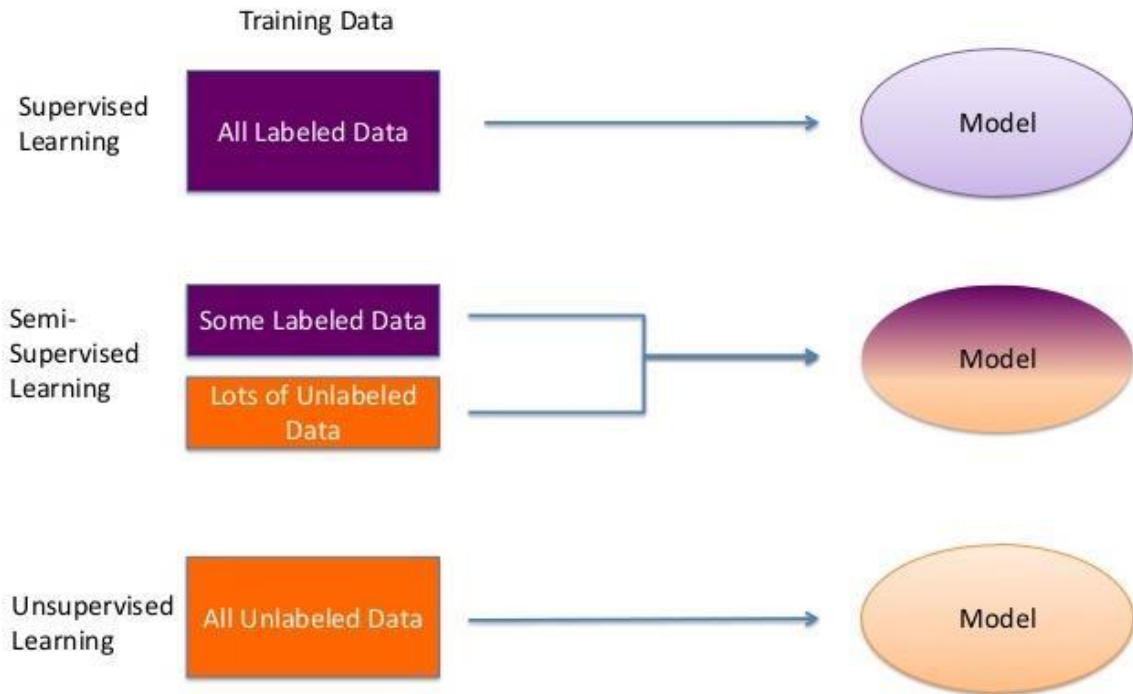


Figure 2-8: Types of Deep Learning Algorithm (Source: Viblo)

Our dataset consists of all the images of human faces and To work with the image dataset it is necessary to understand Convolutional Neural Network (CNN).

2.3 Convolutional Neural Networks

A Convolutional Neural Network (ConvNet/CNN) is a Deep Learning algorithm which can take in an input image, extract all the features from the image by processing it and this information updates all the parameters of the network to make a prediction on the unknown images. CNN takes care of feature engineering by learning through these weights [23].

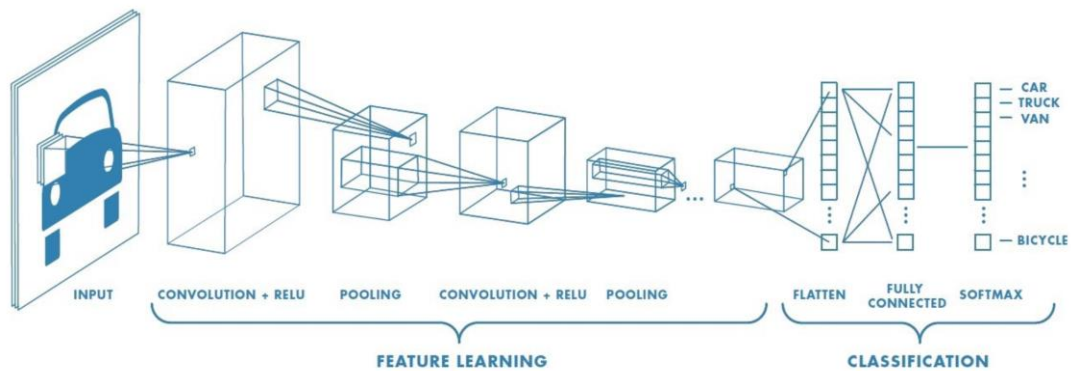


Figure 2-9: Block Diagram of CCN (Source:medium)

Figure 2-9 shows the block diagram of CNN consisting all the components. After training the model and by learning all the parameters with the given data, CNN can be useful to solve image classification problems. Layers in the CNN also called Convolution layer helps to learn features from the image which is why preprocessing required is less as compared to other neural networks.

A ConvNet can successfully capture the Spatial and Temporal dependencies in an image through the application of relevant filters. The architecture performs a better fitting to the image dataset due to the reduction in the number of parameters involved and reusability of weights.

2.3.1 Convolution Layer

Convolution is the first layer to extract features from an input image. Convolution preserves the relationship between pixels by learning image features using small squares of input data. Convolution is a mathematical operation that takes place between inputs such as image matrix and a filter or kernel.

The objective of the Convolution Operation is to extract the low-level features such as edges, from the input image. These low-level features serve as input to the next ConvNets layer which helps to gather all the information in one feature map. These low-level features when we give as input to the next convolution layer, we get high level features like shapes consisting more detailed information of the image. Adding Convolution layer helps to understand image more clearly, which is why we use ConvNet to work with complex 3D images. Convolution of an image with different filters can perform operations such as edge detection, blurring and sharpening the image by applying filters.

2.3.2 Padding

Sometimes filter does not fit perfectly to the input image because of different sizes of images. Padding helps to process those images with the following two ways:

1. Pad the picture with zeros (zero-padding) so that it fits.
2. Drop the part of the image where the filter did not fit. This is called valid padding which keeps only the valid part of the image.

2.3.3 Rectified Linear Unit (ReLU)

ReLU stands for Rectified Linear Unit for a non-linear operation.

The output is $f(x) = \max(0, x)$.

ReLU's purpose is to introduce non-linearity in our ConvNet. Since, the real world data are not linearly distributed, ReLU adds non linearity to detect all those non linear features.

2.3.4 Pooling Layer

Pooling layers section would reduce the number of parameters when the images are too large. Spatial pooling is also called subsampling or downsampling which reduces the dimensionality of each map but retains the important information. Spatial pooling can be of different types:

- Max Pooling
- Average Pooling
- Sum Pooling

The role of a fully connected layer in a CNN architecture. The objective of a fully connected layer is to take the results of the convolution/pooling process and use them to classify the image into a label. The output of convolution/pooling is flattened into a single vector of values, each representing a probability that a certain feature belongs to a label. For example, if the image is of a cat, features representing things like whiskers or fur should have high probabilities for the label "cat".

2.3.5 Pre-Trained Networks:

A pretrained network is a saved network that was previously trained on a large dataset, typically on a large-scale image-classification task. ImageNet is one of the large dataset consisting of more than 14 million images which consist of 20000 classes. Training model on this dataset surely helps to generalize that model on another dataset. Creating a model from the start is always a huge task but with the model trained on ImageNet can efficiently help to predict on other small data. There are so many Pre-trained well-known models already built on ImageNet dataset in the past years. Let's start exploring those to get an idea of the architecture and ideas used to build a model. The pre-trained models which we used for this project is as follows:

2.3.5.1 VGG16

VGG is a Convolutional Neural Network (CNN) architecture used for classification tasks [7]. All the CNNs have more or less similar architecture, stack of convolution and pooling layers at start and ending with fully connected and soft-max layers. The VGG architecture is also similar and is clearly explained in the following paper. VGG was the simplest but one of the deep neural networks prior to the state of the art model. It accepts input as 224*224 RGB image. VGG used 3*3 receptive fields as convolution layer which is why they could go deeper till 19 layers. Max Pooling is used to extract the features from

ConvNet Configuration					
A	A-LRN	B	C	D	E
11 weight layers	11 weight layers	13 weight layers	16 weight layers	16 weight layers	19 weight layers
input (224 × 224 RGB image)					
conv3-64	conv3-64 LRN	conv3-64 conv3-64	conv3-64 conv3-64	conv3-64 conv3-64	conv3-64 conv3-64
maxpool					
conv3-128	conv3-128	conv3-128 conv3-128	conv3-128 conv3-128	conv3-128 conv3-128	conv3-128 conv3-128
maxpool					
conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256 conv1-256	conv3-256 conv3-256 conv3-256	conv3-256 conv3-256 conv3-256 conv3-256
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 conv1-512	conv3-512 conv3-512 conv3-512	conv3-512 conv3-512 conv3-512 conv3-512
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 conv1-512	conv3-512 conv3-512 conv3-512	conv3-512 conv3-512 conv3-512 conv3-512
maxpool					
FC-4096					
FC-4096					
FC-1000					
soft-max					

Table 2-1: Architecture of VGG16 (Source: arxiv)

the image in between the two layers. Architecture ended with Fully connected layer of 1000 classes.

VGG really generalized well by securing first and second positions in the localization and classification tasks respectively in ImageNet challenge. Table 2-1 shows the Architecture of VGG16.

Problem: It has been noticed that after some depth, the performance degrades. This was one of the bottlenecks of VGG. They couldn't go as deep as wanted, because they started to lose generalization capability.

One of the problems ResNet solve is the famous known vanishing gradient. This is because when the network is too deep, the gradients from where the loss function is calculated easily shrink to zero after several applications of the chain rule. This result on the weights never updating its values and therefore, no learning is being performed.

layer name	output size	18-layer	34-layer	50-layer	101-layer	152-layer
conv1	112×112	7×7, 64, stride 2				
		3×3 max pool, stride 2				
conv2_x	56×56	$\begin{bmatrix} 3 \times 3, 64 \\ 3 \times 3, 64 \end{bmatrix} \times 2$	$\begin{bmatrix} 3 \times 3, 64 \\ 3 \times 3, 64 \end{bmatrix} \times 3$	$\begin{bmatrix} 1 \times 1, 64 \\ 3 \times 3, 64 \\ 1 \times 1, 256 \end{bmatrix} \times 3$	$\begin{bmatrix} 1 \times 1, 64 \\ 3 \times 3, 64 \\ 1 \times 1, 256 \end{bmatrix} \times 3$	$\begin{bmatrix} 1 \times 1, 64 \\ 3 \times 3, 64 \\ 1 \times 1, 256 \end{bmatrix} \times 3$
conv3_x	28×28	$\begin{bmatrix} 3 \times 3, 128 \\ 3 \times 3, 128 \end{bmatrix} \times 2$	$\begin{bmatrix} 3 \times 3, 128 \\ 3 \times 3, 128 \end{bmatrix} \times 4$	$\begin{bmatrix} 1 \times 1, 128 \\ 3 \times 3, 128 \\ 1 \times 1, 512 \end{bmatrix} \times 4$	$\begin{bmatrix} 1 \times 1, 128 \\ 3 \times 3, 128 \\ 1 \times 1, 512 \end{bmatrix} \times 4$	$\begin{bmatrix} 1 \times 1, 128 \\ 3 \times 3, 128 \\ 1 \times 1, 512 \end{bmatrix} \times 8$
conv4_x	14×14	$\begin{bmatrix} 3 \times 3, 256 \\ 3 \times 3, 256 \end{bmatrix} \times 2$	$\begin{bmatrix} 3 \times 3, 256 \\ 3 \times 3, 256 \end{bmatrix} \times 6$	$\begin{bmatrix} 1 \times 1, 256 \\ 3 \times 3, 256 \\ 1 \times 1, 1024 \end{bmatrix} \times 6$	$\begin{bmatrix} 1 \times 1, 256 \\ 3 \times 3, 256 \\ 1 \times 1, 1024 \end{bmatrix} \times 23$	$\begin{bmatrix} 1 \times 1, 256 \\ 3 \times 3, 256 \\ 1 \times 1, 1024 \end{bmatrix} \times 36$
conv5_x	7×7	$\begin{bmatrix} 3 \times 3, 512 \\ 3 \times 3, 512 \end{bmatrix} \times 2$	$\begin{bmatrix} 3 \times 3, 512 \\ 3 \times 3, 512 \end{bmatrix} \times 3$	$\begin{bmatrix} 1 \times 1, 512 \\ 3 \times 3, 512 \\ 1 \times 1, 2048 \end{bmatrix} \times 3$	$\begin{bmatrix} 1 \times 1, 512 \\ 3 \times 3, 512 \\ 1 \times 1, 2048 \end{bmatrix} \times 3$	$\begin{bmatrix} 1 \times 1, 512 \\ 3 \times 3, 512 \\ 1 \times 1, 2048 \end{bmatrix} \times 3$
	1×1	average pool, 1000-d fc, softmax				
FLOPs		1.8×10^9	3.6×10^9	3.8×10^9	7.6×10^9	11.3×10^9

Table 2-2: Architecture of ResNet (Source: arxiv)

2.3.5.2 ResNet [8]

Neural networks approximate a function really well. The logic behind improving results is to create an identity function where the output of a function becomes the input itself.

$$f(x) = x \quad (10)$$

Following this logic, we bypass the input to the first layer of the model to be the output of the last layer of the model, so that the network should be able to predict any function it was learning before with the input added to it.

$$f(x) + x = h(x) \quad (11)$$

With ResNets, they created skip connections between layers so that the gradients can flow directly through the skip connections backwards from later layers to initial layers. Using this skip connection, it becomes easier to track back the gradient avoiding vanishing gradient problem. Table 2-2 represents the architecture of ResNet model.

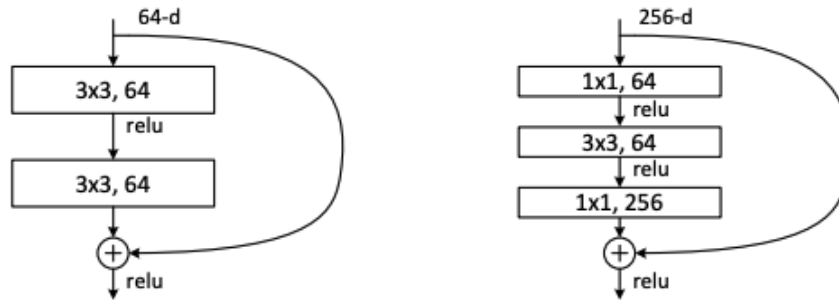


Figure 2-10: Building Block for ResNet (Source: arxiv)

To Discuss the architecture the first important component is the convolution layer.

- a) **Convolution 1:** The first step on the ResNet before entering the common layer behavior is a block called Conv1 consisting of a convolution + batch normalization + max pooling operation. We have explained working of convolution and max pooling layer before. Batch Normalization normalizes output by subtracting the batch mean and dividing by the standard deviation.

b) **ResNet Layers:** Resnet models can go deeper by increasing the number of blocks which consist of convolution layer and batch normalization with ReLU activation function. The identity shortcut connection between these blocks are created by the addition operator. Feature map generated from the previous layers gets added by the addition operator. Additional operator can be applied on the same size input which can be achieved by using 1x1 convolutions. Figure 2-10 shows the building block for ResNet model.

Layers	Output Size	DenseNet-121	DenseNet-169	DenseNet-201	DenseNet-264
Convolution	112 × 112	7 × 7 conv, stride 2			
Pooling	56 × 56	3 × 3 max pool, stride 2			
Dense Block (1)	56 × 56	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 6$	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 6$	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 6$	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 6$
Transition Layer (1)	56 × 56	1 × 1 conv			
	28 × 28	2 × 2 average pool, stride 2			
Dense Block (2)	28 × 28	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 12$	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 12$	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 12$	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 12$
Transition Layer (2)	28 × 28	1 × 1 conv			
	14 × 14	2 × 2 average pool, stride 2			
Dense Block (3)	14 × 14	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 24$	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 32$	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 48$	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 64$
Transition Layer (3)	14 × 14	1 × 1 conv			
	7 × 7	2 × 2 average pool, stride 2			
Dense Block (4)	7 × 7	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 16$	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 32$	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 32$	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 48$
Classification Layer	1 × 1	7 × 7 global average pool			
		1000D fully-connected, softmax			

Table 2-3: Architecture of DenseNet (Source: arxiv)

2.3.5.3 DenseNet [10]

In this paper, they have discussed the state of an art Convolutional Neural Network which performs best amongst all the other CNN's. Table 2-3 represents the Architecture of DenseNet model.

The problems arise with other CNNs when they go deeper. As the network goes deeper and deeper, connection between input layer and the output layer increases due to which, vanishing gradient problem occurs. In this problem, gradient becomes so small that after multiplying with the weights, it can vanish. The ResNet model solved this problem by adding skip connections but results in a large number of parameters and redundant feature map.

In DenseNet, each layer obtains additional inputs from all preceding layers and passes on its own feature-maps to all subsequent layers. Concatenation is used for connection between the two layers. Concatenation is the simple change made in the ResNet model which improved in all aspects of training the model. Feature reusability, computational efficiency and reduced complexity are the main advantages as compared to other models. By connecting this way DenseNets require fewer parameters than an equivalent traditional CNN, as there is no need to learn redundant feature maps.

Traditional feed-forward neural networks connect the output of the layer to the next layer after applying a composite of operations.

$$x_1 = H_1(x_{1-1}) \tag{12}$$

DenseNets make the first difference with concatenating all the feature maps.

The equation becomes:

$$x_1 = H_1([x_0, x_1, \dots, x_{1-1}]) \tag{13}$$

DenseNets are divided into DenseBlocks, where the dimensions of the feature maps remain constant within a block, but the number of filters changes between them. These

layers between them are called Transition Layers and take care of the downsampling applying batch normalization, a 1x1 convolution and a 2x2 pooling layer.

DenseNet also introduced the parameter growth rate which regulates how much information is added to the network each layer. In other words, it decides how much previous feature map should add to the next Dense Block as an input. To explain this, you can see every layer is adding to the previous volume these 32 new feature maps. Therefore we go from 64 to 256 after 6 layers. To reduce the feature map and bypass connection, Transition Block performs as 1x1 convolution with 128 filters. followed by a 2x2 pooling with a stride of 2, resulting in dividing the size of the volume and the number of feature maps in half.

The original idea about improving training efficiency by shorter connections between layers close to the input and those close to the output helped me to propose my Modified DenseNet model.

2.4 Action Unit Detection with Deep Learning

AU detection has been studied over a long period of time to understand human expression and emotions associated with it. To detect action units using algorithms, the best reference to have is the Facial key points (landmark points) [24]. Two types of features were usually used in landmark-based approaches. Landmark geometry features were obtained by measuring the normalized facial landmark distances and the angles of the Delaunay mask formed by the landmark points. On the other hand, landmark texture features were obtained by applying multiple orientation Gabor filters to the original images. These landmark points help to identify each feature of the face like nose, eyes, lips etc.

With this information it is easy to select features related to each action unit and then classify with the help of Supervised Vector Machine (SVM) and many other classifiers. SIFT algorithm also can be used to select such features.

Over the last few years, CNN has boosted the performance in many fields of detection and classification. With deep learning, it became easier to solve such complex problems with the large dataset. CNN can be used now for feature extraction and feature selection just by training a good deep classifier with a good amount of data.

2.4.1 Deep Net

As we discussed about the advantages of CNN, in this paper, the network consists of two parts [25]. The first part of the network is used to detect a number of facial key points to determine the features required to detect action units. Introducing CNN layers, we no longer require geometric methods to find out landmark points. In this paper, nine layers are dedicated to the detection of facial landmark points. They use three convolutional layers, two max pooling layers and two fully connected layers. They applied normalization, dropout, and rectified linear units (ReLU) at the end of each convolutional layer to enhance the results. They detected 66 landmark points with this first part of the network.

In the second part of the network they combined the landmark points obtained from the first part of the network and the images pre-processed required to detect action units. They used concatenation to connect two parts of the network and get the final prediction. To simplify, first part of the network concatenated with the output of the first fully connected layer of the second part of the network.

As for the structure of the deep net, they adopted GoogleNet to work with. To concatenate and add both the layers they had to change input size and number of filters according to the requirement. Detecting the action unit with this approach is really great to start with. Adapting to state of the art classifier can definitely improve the results for sure. The problem with this approach is that the first part of the network will always be fixed to determine landmark points and if the points do not match with the face of the images then it is really difficult to track back all the features we need. If the image has more than one face which is there in our dataset, it is going to be really difficult to match landmark points with the faces and extract features. Hence considering different datasets which has landmark points for every image and which can have automated facial expression recognition system can enhance the detection process.

2.4.2 DISFA: A Spontaneous Facial Action Intensity Database [26]

We have discussed different potential applications of AU detection include human-computer interaction, social robots, healthcare, marketing, biometrics, behavioral and neuroscience. With initial work in these domain, AU Detection technology is growing with different emerging applications. With improvement in different neural network models, to get better results it is critical to get well-labeled video and photos of facial behavior. To get the success of automated facial expression recognition, it is important to have publicly available databases to work on. Working with different database, will improve the feature map for respective action unit and it will help to detect action units more efficiently.

To annotate facial action units, a human expert manually labels every video frame which is very time consuming. To go through all the frames of videos and label them with different action units is tedious task to perform. To label graded changes in intensity, additional time would be required. Considering manually annotating facial action units is standard process, it is not ideal to rely on. Automated measurement is essential to the feasibility of real-time applications. DISFA is the database where action units are automatically annotated considering changes in intensity.

Almost all research into automated measurement has been directed at binary AU detection whether it is present or absent. But considering temporal envelope of intensity variation, it is possible to achieve more insights about different action units. DISFA contains approximately 130,000 annotated frames from 27 adult participants. For every video frame, the intensity of 12 action units was manually annotated on a six-point ordinal scale (0 and five increasing levels of intensity). The AUs chosen were among those most common in emotion expression and social interaction and that have been studied previously in computer vision and machine learning. Participants for data collection process viewed a 4-minute YouTube video clip to collect range of facial expressions of emotion.

AU intensity was coded for each video frame on a 0 (not present) to 5 (maximum intensity) ordinal scale. The advantage of this dataset is that they have set of landmark points (66 points) which are required for automated facial expression recognition. In my thesis, I am working on EmotioNet dataset which consist of images with 12 Action Units. But in future scope, DISFA is really one of the important datasets which will help me to

improve my detection results more considering changes in intensity. Let's discuss one of the approaches to work with multi label classification problem.

2.4.3 Binary Relevance for Multi Label Learning [27]

Multi Label classification problem is a generalization of multiclass classification where multiple labels are assigned for particular instance. In this paper they have approached this problem by decomposing the multi-label learning task into a number of independent binary learning tasks (one per class label).

The simplicity of this method is to start training independently on one binary classifier for each label. Given an unseen sample, the combined model then predicts all labels for this sample for which the respective classifiers predict a positive result. In this algorithm, binary classifiers need to train for one label irrespective of other existing labels. Due to its conceptual simplicity, binary relevance has attracted considerable attention in multi-label learning research.

Mathematical expression for the given algorithm is simplified as follows.

Let $X = R^d$ denote the d-dimensional instance space and let $Y = \{\lambda_1, \lambda_2, \dots, \lambda_q\}$ denote the label space, consisting of q class labels. For each multi label training example (x^i, y^i) , $x^i \in X$ is a d-dimensional feature vector $[x_1^i, x_2^i, \dots, x_d^i]$ and $y^i \in \{-1, +1\}^q$ is a q-bit binary vector $[y_1^i, y_2^i, \dots, y_q^i]$, with $y_j^i = +1$ (-1) indicating that y_j^i is a relevant (or irrelevant) label for x^i . It decomposes the multi-label learning problem into q independent binary learning problems. Each binary classification problem corresponds to one class label

in the label space Y . Then, as mentioned, binary classifier can be applied to the instances having the same label independently. To continue this process independently for each label can solve the multi label classification problem.

One potential weakness of binary relevance lies in its inability to exploit label correlations to improve the learning system's generalization ability. Therefore, a natural consideration is to attempt to provide binary relevance with label correlation exploitation abilities while retaining its linear modeling complexity w.r.t. the number of class labels.

Generally, representative strategies to provide binary relevance with label correlation exploitation abilities include the chaining structure assuming random label correlations, the stacking structure assuming full-order label correlations, and the controlling structure assuming pruned label correlations.

However, it is also noteworthy that some inherent properties of multi-label learning should be investigated in order to further enhance the generalization ability of binary relevance. There are two major issues classifying multi label problem which is not considered in this algorithm is as follows:

1. Class imbalance problem where there is a lack of equal number of images for each class.
2. Relative labeling-importance where the correlation between two labels cannot be prioritized.

Listed problems encouraged us to look in the classifier which can give us better action unit detection since classifier is the most important factor to start with.

With the mentioned literature review, our approach will be to locate the face and facial features in an image, derive a feature representation of the face, and then classify the presence or absence of a facial expression in that image using a really good fit classifier. Since this is a multi-label classification problem, it is really important to understand the Methodology to deal with this problem.

CHAPTER 3: METHODOLOGY

3.1 Shortcoming of CNN state of the art model

We went through all the history of CNN architecture from VGG16 to the state of the art DenseNet. In DenseNet we saw that connecting layers by concatenating requires fewer parameters than an equivalent traditional CNN, as there is no need to learn redundant feature maps. Furthermore, concatenating layers helps to solve vanishing gradient issues since each layer has direct access to the gradients from the loss function and the original input image.

In DenseNet architecture, the depth of a dense layer's output is dependent on the 'growth_rate' parameter. As every dense layer receives all the output of its previous layers, the input depth for the k th layer is $(k-1)*\text{growth_rate} + \text{input_depth_of_first_layer}$. So to add new information in the feature map it all depends on the growth rate parameter. Let's say the growth rate is kept 20 for the architecture, after 100 layers, the depth will be around 2000 which is huge. To compensate for this problem, DenseNet is using a Transition layer which is reducing the feature map into half whenever the transition layer is used in the architecture. Reducing the feature map results in losing some important features. Let's say

if we lose the edge of an eye at the lower level of feature abstraction, we will miss out on the high level of features because of the feature hierarchy.

So, it is really important to pick out the balanced value for growth rate to avoid such problems. You can see in the DenseNet architecture, after every Dense Block, they used Transition layer to reduce the depth of the feature map. Transition layer consists of 1*1 Convolution layer with batch normalization and Pooling layer after that. 1*1 Convolutional layer changes the dimensionality on the filter space whereas Pooling layer reduces the size of each feature map. Using these layers with pooling, transition layer achieved to reduce feature map into half. Reducing feature maps can be problematic as we discussed above about the important features. Missing features can affect probabilities predicted for each action unit which can also affect the prediction of the actual labels of the images.

Best value for Growth rate hyperparameter cannot be exactly determined and Thus we are dealing with the Tradeoff between new information and basic crucial features.

3.2 Motivation to improve

Considering the advantages of DenseNet which represents the state of the art, we should exhibit the idea of concatenation. To deal with vanishing gradient problems and to remove the redundant features, it is always helpful to create shorter connections between input and output. Shorter connection helps function to approximate well and update the parameter to predict near to the actual label.

Keeping in mind the cost of applying DenseNet, we need to improve our feature map avoiding the loss of basic features. Considering features which can get lost after applying transition layer is the prime target to improve the results and prediction on the dataset.

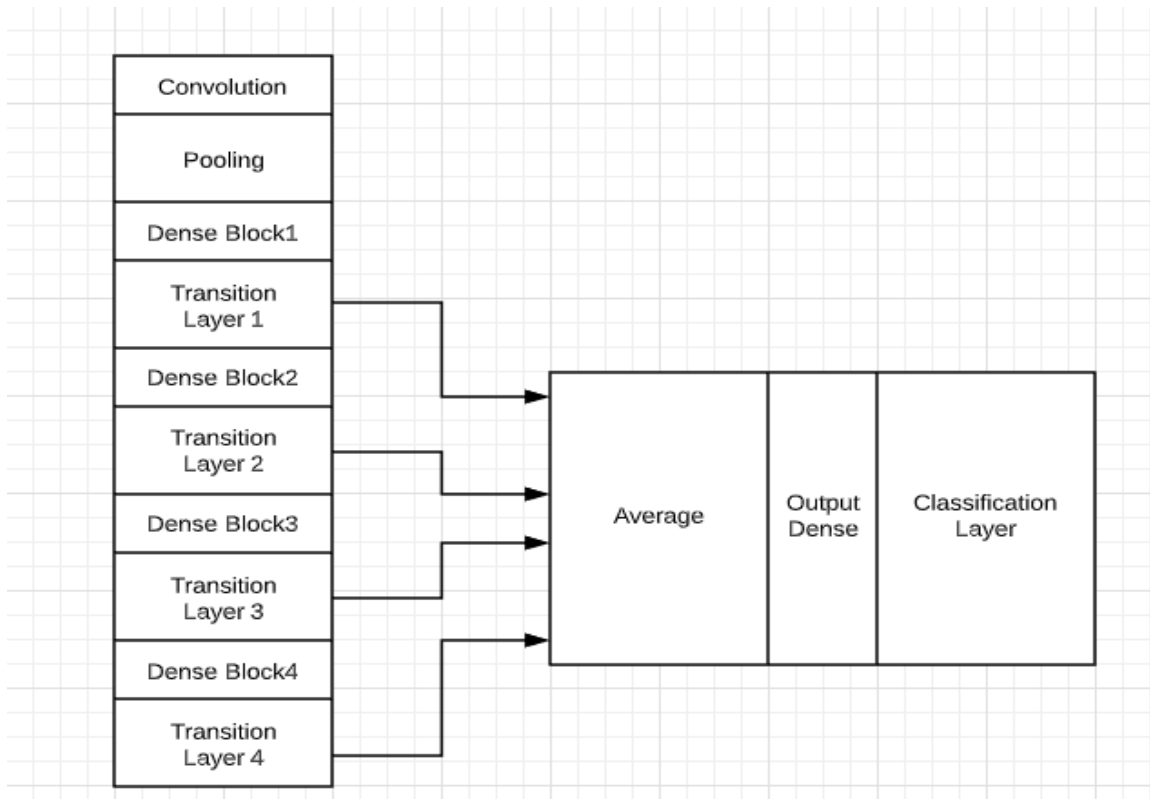


Figure 3-1: Modified DenseNet Block Diagram

3.3 Ideas to implement

In Table 3, DenseNet architecture, Dense Block and Transition block connected in a pair and it is repeated based on how many layers of network are you creating. In Dense block, convolutional layers used to extract features to create feature maps. These feature maps get reduced to half in transition layer and it is used as input for the next dense block.

Before applying dimensionality, reduction techniques using Pooling layer and 1*1 averaging layer, the output of two Convolutional layers combine together by connecting them using concatenation. The advantage of concatenation we have mentioned before, which is why we need to save this feature map before reducing the dimensions in the transition layer.

The idea to improve this is to make shorter connections from input to output and to consider all the feature maps derived from each concatenated layer. For that, we need to take out the output from each concatenated layer after every dense layer and combine it together. The architecture for the given ideas is shown below. Figure 3-1 shows the block diagram of proposed modified DenseNet idea.

Since I used DenseNet-169 as my baseline model, it has 4 layers in the network. Thus, output needs to be collected from the concatenated layer every after each Dense block.

These 4 inputs can combine by many approaches:

1. Addition:

By adding all the concatenated output, the feature map generated with the great information, but it increases dimensionality of the feature map. Increasing dimensions results in more numbers of parameters which need to be tuned and can get overfit if you do not have enough data.

2. Concatenation:

For Creating shorter connection, it is a really good idea to use Concatenation between the layers. But it did not improve the results by connecting in

concatenation as it did not actually perform anything on all the features. Connecting all features in concatenation did not help at all to improve results.

3. Average:

Averaging is the more efficient way to connect all the concatenated output. Averaging all the layers definitely consider all the output feature maps and combine together. It helped to keep the dimension of the feature remaining the same and consider all the inputs at the same time. Keeping the same number of parameters and with more feature information, averaging all the outputs really worked well to intact all the features and predict really well on the unknown dataset.

We implemented this idea in one of the competitions named EmotioNet challenge which is based on EmotioNet dataset.

3.4 EmotioNet Challenge [28]

EmotionNet challenge is based on the facial action unit detection. This challenge requires the identification of 12 action units (AUs). The AUs included in the challenge are: 1, 2, 4, 5, 6, 9, 12, 17, 20, 25, 26, 43.

3.4.1 Training data:

The EmotioNet database includes 950,000 images with annotated AUs. These were annotated with the algorithm described [29]. This dataset is used for training and validation datasets. For the Verification phase we got access to a server where we can implement our trained model on the unknown dataset.

3.4.2 Challenge Phase:

In the challenge phase, Participants connected to the server one final time to complete the final test on the test dataset. The test dataset used in this phase is different than the one in the verification phase. The results of this phase will be used to compute the final scores of the challenge.

3.4.3 Evaluation:

Identification accuracy of AUs measured using two criteria – accuracy and F-scores. These are the two evaluation metrics we used to test my algorithm in this thesis. The best Algorithms determined will then be classified based on the mean of the recognition of all AUs. Formally, these criteria are defined as follows. We finished 6th in this competition amongst all participants. The performance of my algorithm based on evaluation metrics is as follows:

<http://cbcs1.ece.ohio-state.edu/EmotionNetChallenge/index.html#2018results>

Let's discuss our approach to solve this multi label classification problem.

3.5 Approach to solve multi label problem

Focusing on our Problem Statement, AU detection considered a multi-label classification problem. The images in EmotioNet dataset are labeled for 12 Action Units. 12 Action units are as follows: 1, 2, 4, 5, 6, 9, 12, 17, 20, 25, 26, and 43. Framework.

Since we have a dataset with labeled data and we know which action units to determine amongst all, we chose Supervised Learning method to train the model and detect the AU's. Convolutional neural network is the most popular ANN for analyzing the images. Images can be classified with the help of a Convolutional layer consisting of filters which helps to detect patterns from the images. My proposed framework is shown in Figure 3-2. The overall procedure is composed of three steps.

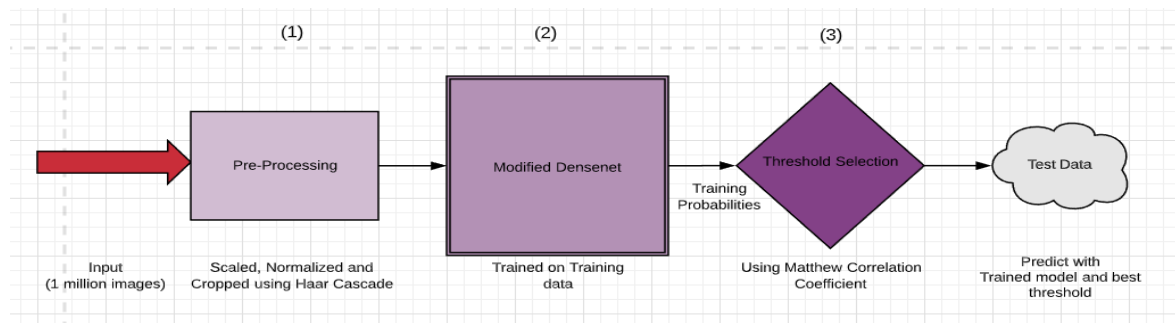


Figure 3-2: Framework for the classification

3.5.1 Pre-Processing

The images from ‘EmotiNet’ are of different sizes. There are a total of 950,000 images. It is important to pre-processed images before training the model on them. To detect AU labels, the important features are on the face and other details are redundant. There are a number of Pre- processing methods we can carry out to improve the data and we have applied few of those which are as follows:

3.5.1.1 Image Cropping

Object Detection using Haar feature-based cascade classifiers is an effective object detection method we used for image cropping [30].

Here we worked with face detection using Haar Cascade classifier where features of face region extracted from non-face region. To detect face regions, features are grouped into different stages of classifier and applied it one by one. While checking the face region, if a window fails the first stage of classifier, discard it. If it passes the first stage, apply the second stage of features and continue the process. The window which passes all stages is considered the Face region. I applied this algorithm to all the images and got nice face detected images to process further.

3.5.1.2 Image Scaling

Once we have cropped images detecting faces, image scaling is the next important step. To normalize the data, all the pixel values should be converted from $[0,255]$ to $[0,1]$. To achieve this all the pixel values are divided by 255, where 255 is the scaling factor to convert pixels to $[0,1]$.

3.5.1.3 Mean of input data

Mean of input data is calculated to normalize data in general where mean image obtained by taking the mean values for each pixel across all training examples. Observing this could give us insight into some underlying structure in the images.

3.5.1.4 Normalizing image inputs

Data normalization is an important step which ensures that all the pixels have a similar data distribution. This makes training the network with all data faster. Data normalization is done by subtracting the mean from each pixel and then dividing the result

by the standard deviation. After Scaling and normalizing the images, all the data feeds into convolutional neural networks as an input.

3.5.2 Modified DenseNet

Convolutional neural network is the classic approach we acquire to Classify 12 Action units by training all the image inputs. The Modified DenseNet classifier mentioned in our thesis is used to train the input images. Modified DenseNet is developed from Transfer Learning the DenseNet model which is the baseline model and motivation for the proposed method. The architecture of the Modified DenseNet is as shown in Figure 3-3.

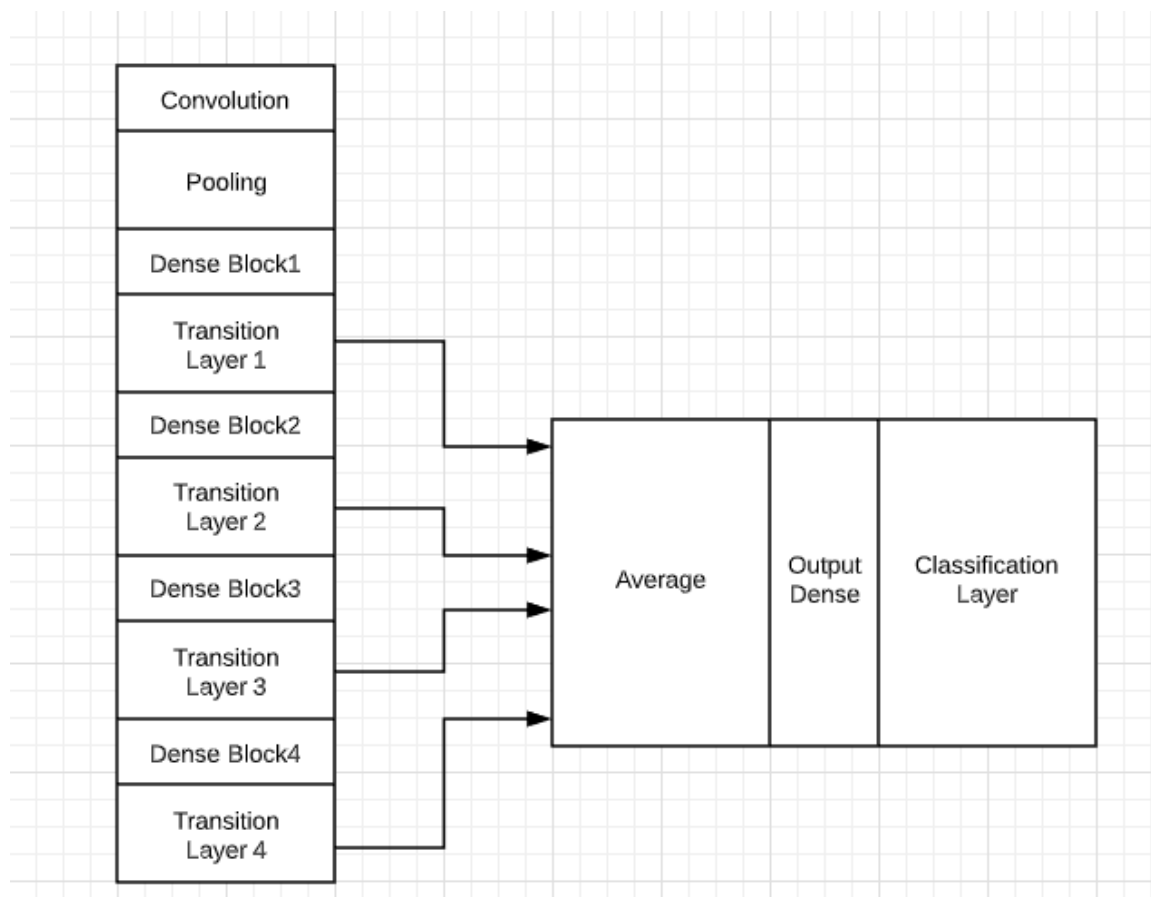


Figure 3-3: Block Diagram of Modified DenseNet Model

It is a popular approach in deep learning where pre-trained models are used as the starting point in computer vision. In transfer learning, we first train a base network on a base dataset, and then we reintroduce the learned features or transfer them to a second target network to be trained on a target dataset and task [31]. This process will tend to work if the features are general, meaning suitable to both base and target tasks, instead of specific to the base task.

- **Pre-trained Model Approach**

1. **Select Source Model.**

A pre-trained DenseNet model is chosen from available models. We applied other models like VGG16, ResNet, Inception-ResNet and Ensemble of both, but the DenseNet model gave the best result.

2. **Reuse Model**

The DenseNet pre-trained model used as the starting point for a model on the second task of interest.

DenseNet Layers:

Traditional feed-forward neural networks connect the output of the layer to the next layer after applying a composite of operations.

DenseBlocks, where the dimensions of the feature maps remain constant within a block, but the number of filters changes between them. These layers between them are called Transition Layers. They take care of the downsampling by applying batch normalization, a 1x1 convolution and a 2x2 pooling layers.

3. Modify Model

To modify the given DenseNet pre-trained model, we used Keras Functional model to add new layers and define my proposed model [32].

Keras Functional Models

Models are defined by creating instances of layers and connecting them directly to each other in pairs, then defining a Model that specifies the layers to act as the input and output to the model.

A. Defining Input

Unlike the Sequential model, we created and define a standalone Input layer that specifies the shape of input data. The input layer takes a shape argument that is a tuple that indicates the dimensionality of the input data. Since DenseNet is our baseline model which accepts input, we reshaped all the images to $224 \times 224 \times 3$ which is the shape of the DenseNet input data.

B. Connecting Layers

This is done by specifying where the input comes from when defining each new layer. The idea of taking the average of each dense layer is implemented by pulling out the Output of two concatenated dense layers and taking the average of them. Since the shape of each output of the dense layer is different, it is difficult to just add and take average of them. We had to apply transition layers to adapt to one shape for all the output and then add them to take average. Transition layer contains the same layers as it was in DenseNet which includes applying batch

normalization, a 1x1 convolution and a 2x2 pooling layer. Dense layer of 12 outputs is added in the end since we have to classify 12 AU's.

C. Creating the Model

We define the model by creating all of the model layers and connecting them together. Keras provides a Model class that you can use to create a model from your created layers. It requires only specific input and output layers.

After creating the functional model, all the images are trained with Modified DenseNet model which gives rise to the probabilities for the 12 respective AU's.

The Architecture of Modified DenseNet is shown in Table 3-1.

Layers	Output Size	Modified DenseNet
Convolution	112×112	7×7 conv, stride 2
Pooling	56×56	3×3 max pool, stride 2
Dense Block (1)	56×56	[1×1 conv, 3×3 conv]×6
Transition Layer (1)	56×56	1×1 conv
	28×28	2×2 average pool, stride 2
Dense Block (2)	28×28	[1×1 conv, 3×3 conv]×12

Transition Layer (2)	28×28	1×1 conv
	14×14	2×2 average pool, stride 2
Dense Block (3)	14×14	[1×1 conv, 3×3 conv]×32
Transition Layer (3)	14×14	1×1 conv
	7×7	2×2 average pool, stride 2
Dense Block (4)	7×7	[1×1 conv, 3×3 conv]×32
Average of concatenation layer from Dense Block (1), (2), (3) & (4)	(FC-128)	[1×1 average pooling, BatchNormalization Activation (Relu)]×4
Classification Layer	(FC-12)	12 classes, sigmoid

Table 3-1: Architecture of Modified DenseNet

3.5.3 Threshold Selection

Once we get the probabilities for each class from the classifier, the important task remains to select the threshold. Selection of threshold is important since it will decide which label should be 1 Or 0.

In Keras library, for binary classification, 0.5 is the threshold to decide the label. If the probabilities are above 0.5, label set to 1 and for the probabilities below 0.5, label set to 0. Since we are dealing with multi class labels, we cannot just set 0.5 as the threshold for all the classes. We need to consider all the possibilities between 0 and 1.

The Matthew correlation coefficient is used to achieve this as a measure of the multiclass classifications [33]. Matthew correlation coefficient takes into account true and false positives and negatives and is generally regarded as a balanced measure which can be used even if the classes are of very different sizes. MCC treats the actual class and the predicted class as two different variables and computes their correlation coefficient. The higher the correlation between actual and predicted values, the better the prediction. Formula use for computation is as follows:

$$\text{MCC} = \frac{TP \times TN - FP \times FN}{\sqrt{(TP+FP)(TP+FN)(TN+FP)(TN+FN)}} \quad (14)$$

The MCC is a correlation coefficient value between -1 and +1. A coefficient of +1 represents a perfect prediction, 0 an average random prediction and -1 an inverse prediction. MCC is useful to check the correlation between predicted label and actual label.

To set the threshold, we consider all the probabilities between 0 and 1 in the interval of 0.1 and set the best threshold (probability) for a particular action unit where MCC is close to +1(high). Best threshold achieved can be used for the prediction of the test images. Classification on the test image dataset is explored in the experiment below.

CHAPTER 4: EXPERIMENTS

4.1 Dataset

The EmotioNet database includes 950,000 images with annotated AUs. This dataset has been used to successfully train a variety of classifiers, including several deep networks.

4.2 Dataset Split

We split our dataset in 3 parts. 60% of the data is for training where we can run our Modified DenseNet model. 20% validation data where we can validate our trained model and the remaining 20% of data is for testing the model which states that how good our model is doing on unseen data. We used HDF5 file format to split and store data separately so that we can easily read this while doing a particular task [34].

4.3 Model Framework

There are many high level API's for building neural network models. We used Keras Library with Tensorflow as a back-end (neural network computation engine) for several reasons like it is user friendly, easy to learn and most importantly, it is easy for model building and deployment of the model.

Since there are around 1 millions of images to process, train, manipulate and store somewhere, we used ‘HDF5 library’ to deal with the large amount of data. The h5py package is a Pythonic interface to the HDF5 binary data format that helped us to preprocess our data and save by making hdf5 file which can be read again easily. Since Keras is pretty much flexible with Hdf5, it was easy to save and load the models we created for training and testing the data.

4.4 Pre-Processing

With the URL’s given, we downloaded all the images. Since we have different multi labels corresponding to each image, it was important to make sure that downloaded images are linked with the corresponding labels where HDF5 library came in the picture. With hdf5 file format it was easy to create dataset with the labels.

We pre-processed all the images by extracting facial landmarks using Haar Feature-based Cascade Classifiers. Face detection is the key part to avoid other unnecessary features using OpenCV library. Tracked faces were registered in hdf5 file after randomly cropped into 255×255 with 224×224 face images, as most neural networks acquire 224×224 inputs to the model. Once we got the images, we applied scaling and normalization methods to make sure we will get good results with our training process.

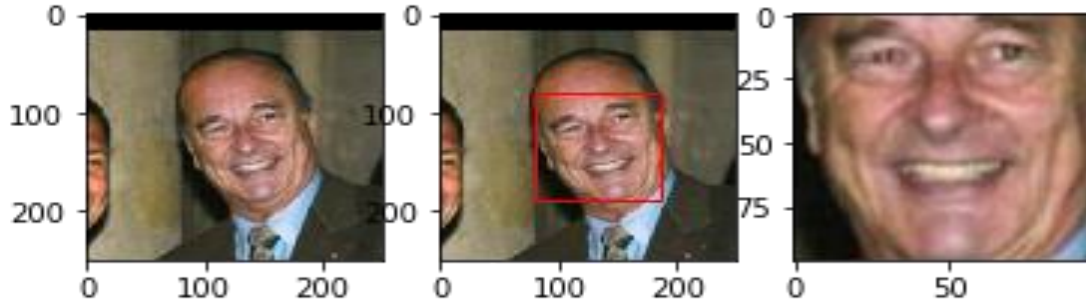


Figure 4-1: Preprocessing using Haar Cascade (Source: Krasserm)

These images then converted into numpy arrays. To normalize these images, the Mean subtraction method applied. The reason behind this is because in the process of training our network, we're going to be multiplying (weights) and adding to (biases) these initial inputs in order to cause activations that we then backpropagate with the gradients to train the model. We'd like in this process for each feature to have a similar range so that our gradients don't go out of control.

4.5 Network Setting and Training [35]

4.5.1 Data and Batchsize

On EmotionNet dataset, we train Modified DensNet model using batch size 32 for 10000 images respectively. Training of 1 million images is impossible at a time for 20 epochs because of the limited GPU space; we trained 10000 Images at a time with the batch size of 32. All input data is preprocessed to the same shape (224*224*3) which is the requirement of almost all the CNN models.

4.5.2 Activation function and Loss function

As we discussed before, activation function introduces non linearity into the model, we are using Sigmoid function for the last dense layer of this multilabel classification. Multilabel classification is done when your model needs to predict multiple classes as the output. For example, let's say you are training a neural network to predict the ingredients present in a picture of some food. There will be multiple ingredients we need to predict so there will be multiple 1's in Y.

For this we can't use softmax because softmax will always force only one class to become 1 and other classes to become 0. So instead we can simply keep sigmoid on all the output node values since we are trying to predict each class's individual probability. With the sigmoid activation function at the output layer neural network models the probability of a class c_j as a Bernoulli distribution.

$$P(c_j|x_i) = \frac{1}{1+\exp(-z_j)} \quad (15)$$

Now the probabilities of each class are independent from the other class probabilities. So we can select a threshold for each AU and can determine the AU is present or not. To make this work in Keras we need to compile the model. An important choice to make is the loss function. We use the binary cross entropy loss to penalize each output node independently, So that the output of the network will be independent Bernoulli distributions per label.

4.5.3 Learning Methods and Learning rate

Adam is an optimization algorithm that we used instead of the classical stochastic gradient descent algorithm since it is straightforward to implement and well suited for our problem which consists of large data. After experimenting with different learning rates, we fixed to 0.001 considering the change in the accuracy and overfitting problem.

4.5.4 Dropout

To avoid overfitting, we used one of the regularization techniques called Dropout where single model can be used to simulate having a large number of different network architectures by randomly dropping out neurons during training. We set dropout hyperparameter to 0.5 to achieve best results for such a large dataset.

4.5.5 Number of Layers

We used a pre-trained densenet model of 169 layers which consist of 4 convolutional blocks. After modifying the network with the proposed idea, the number of layers goes to 175 considering dropout layer.

4.5.6 Number of Epochs

Number of Epochs cannot be fixed and cannot be determined without running the experiment. We ran different numbers of epochs starting with 10, 15, 20 and 30. We fixed with 20 epochs considering all the factors like accuracy, overfitting and time complexity.

To summarize everything, Model is compiled with 0.001 learning rate and binary cross entropy loss function. After training and testing, keeping in mind about overfitting and underfitting decided to keep the last 30 layers trainable [36]. Model checkpoint used to save the information about the weights for particular experiments and find out the best weights for the greater accuracy on validation data. All the networks are trained using Adam optimizer and sigmoid activation function.

4.6 Threshold Selection

After training all the images, prediction probabilities for each action unit are received. Best threshold can be achieved using MCC by finding the correlation between label probabilities and actual labels.

4.7 Evaluation

Identification accuracy of AUs measured using two criteria – accuracy and F-scores [20]. Accuracy is a measurement of closeness to the true value. We have computed accuracy for each AU and Average accuracy is calculated by averaging all the single AU accuracies.

$$\text{accuracy}_i = \frac{\sum_i \text{true positives} + \sum_i \text{true negatives}}{\sum_i \text{total population}} \quad (16)$$

Where true positives are AU_i instances correctly identified in the test images, true negatives are images correctly labeled as not having AU_i active/present and the total population is the total number of test images.

The mean accuracy is

$$\text{Accuracy} = m^{-1} \sum_i \text{accuracy} \quad (17)$$

The standard deviation is

$$\sigma^2 = m^{-1} [(\sum_i \text{accuracy}) - \text{Accuracy}]^2 \quad (18)$$

where m is the number of AUs.

F-scores can be provided by each AU before computing the mean and standard deviation.

The F-score of AU_i is given by

$$F_{\beta_i} = (1 + \beta^2) \frac{\sum_i \text{precision} \cdot \sum_i \text{recall}}{\beta \cdot \sum_i \text{precision} + \sum_i \text{recall}} \quad (19)$$

Where Precision_i is the fraction of AU_i is correctly identified, Recall_i is the number of correct recognitions of AU_i over the actual number of images with AU_i active, and β defines the relative importance of precision over recall.

Labels of test data are predicted with the model we trained on the training data. The trained model is loaded again to predict probabilities for each action unit of the test images. These predicted probabilities then set to 1 or 0 by comparing the best threshold we got from Matthew correlation coefficient. The result we got with the Modified DenseNet model is best amongst all other state of the art models.

4.8 Testing the data

Labels of test data are predicted with the model we trained on the training data. The trained model is loaded again to predict probabilities for each action unit of the images [37]. These predicted probabilities then set to 1 or 0 by comparing the best threshold we

got from Matthew Correlation Coefficient. The result we got with the Modified DenseNet model is best amongst all other state of the art models.

CHAPTER 5: RESULTS AND DISCUSSION

F1 score is used as the primary criteria alongside average mean to evaluate large EmotioNet dataset. Moreover, as we were exploring which CNN structure performs the best for these 12 action units in the EmotioNet dataset, the F1 score and average mean of each action units are averaged. The averaged F1 score and classification rate are the final metrics to evaluate a certain pre-trained network and my proposed method.

$$\text{Final Metric} = 0.5(\text{accuracy} + \text{F1}) \quad (20)$$

We evaluate VGG16, Resnet, Inception Resnet, DenseNet, Ensembling Methods, Modified DenseNet with different loss functions, activation function, dropout and other factors. In the end we fixed on one framework for all the pretrained models after trying and testing to ensure a fair comparison between all pretrained models and our proposed model. We simply replace the training models keeping all the experiment settings exactly the same for all. All the models accept 224×224 input images.

5.1 VGG16

VGG16 model is the first experiment on Emotionet dataset we performed which gave decent results and gave me the idea of till what accuracy we can reach with this framework of fixed parameters.

VGG16 consists of 16 weight layers in the network. The width of the conv. layers (the number of channels) is rather small, starting from 64 in the first layer and then i

Increasing by a factor of 2 after each max-pooling layer, until it reaches 512. Three fully connected layers are added with the last sigmoid layer give us the probabilities for 12 AU's.

With VGG16, we got mean accuracy of 83 and F1 accuracy as 22.96. We accounted for a few problems like vanishing gradient problem, so we move forward with another model.

5.2 ResNet

ResNet 3×3 filters, Down-sampling with CNN layers with stride of 2, Global average pooling layer and a 1000-way fully-connected layer with Sigmoid in the end. We have not got any better results with ResNet as we got almost similar accuracy which is VGG16 has. The problems we figure out with ResNet are lots of parameters to train and gradient vanishing problem.

5.3 DenseNet

DenseNet consists of Dense Blocks, where the dimensions of the feature maps remain constant within a block, but the number of filters changes between them. Layers between them are called Transition Layers and take care of the downsampling applying batch normalization, a 1x1 convolution and a 2x2 pooling layer. With DenseNet accuracy increased to 85.5% mean accuracy with 33% F1 accuracy which was really impressive as

compared to the previous results. With improved results we decided to modify the DenseNet layer to improve accuracy further.

5.4 Ensembling Methods

In the Ensembling method, we averaged DenseNet model with ResNet model. The accuracy improved but the model was overfitting due to which it performed poorly on test data. Too many parameters result in overfitting the model on training data.

AU	DenseNet	Modified DenseNet
1	83.9	86.2
2	86.2	90.2
4	82.3	83.9
5	86.3	89.9
6	77.5	77.8
9	91.6	94.7
12	80.2	79.6
17	89.5	92.6
20	99	99.1
25	79.3	80.5
26	72.7	74.9
43	98.2	98.2

Table 5-1: Accuracy for each action units (Accuracy on the test data)

5.5 Modified DenseNet

Proposed model is developed by taking the entire concatenated layer which consists of features and we took the average of them. Modified Densenet model really worked well giving 87.5% mean accuracy with 36% F1 accuracy. This is the best accuracy we got on EmotioNet dataset after trying other pre-trained models. Comparison between DenseNet and Modified DenseNet AU wise is explained in Table 5-1.

CHAPTER 6: CONCLUSION AND FUTURE WORK

This project focused on detection of occurrence of each AU among the 12 AUs in the EmotioNet dataset using transfer learning. Feature extraction is conducted using a wide variety of models including VGG-16, ResNet-50, DenseNet, Ensembling of models and modified DenseNet model. Based on these CNN performances, the Modified DenseNet model really works well amongst all and gives the best probabilities for each AU's. With the best threshold calculated using MCC, we receive good accuracy on test data which is around 87.5. Final result of the evaluation metrics on DenseNet and Modified DenseNet is mentioned in Table 6-1.

A proposed extension of this project is to investigate pre-processing of Images. Images can be processed based on regions which depend on AU's needs to detect. Divide images into regions of AU present and feed into modified DenseNet model [38] [39].

Region based processing carried out considering AU's where Each AU represents a basic facial movement or expression change. Every facial action units depicts different movement in the facial feature such as eyebrows lower, cheek raiser, chin raiser and lip tighter, etc. Different emotions represent a combination of AU's. So for the different images we select different regions representing action units and then crop it before feeding into the model. Cropping the images through preprocessing will give more attention to the features which are representing these actual action units. Result of which, we will update

the weights of the network accordingly so that it gets easier for the predictions on test data. This preprocessing will surely help to improve the detection of action units.

LSTM layer can be added and train the model on a dataset consisting of images in sequence. Using LSTM layers, the images in sequence which are feeding to the network can be compared to see the changes in the features of the images as well as the labels. Comparing labels of different images, we can get the correlation between different action units so that when we are testing the unknown images, we can give attention to the group of different action units which show those respective features [40].

Training more datasets can improve the results acquired from the single dataset. Here we just used EmotioNet dataset. We can train with more dataset to improve the weights and therefore prediction of the network.

Metrics	DenseNet	Modified DenseNet
Mean Accuracy	85.5	87.5
F1	33	36.7

Table 6-1: Evaluation of models based on the metrics (Mean accuracy calculated by taking average of individual AU and F1 by using listed formula)

BIBLIOGRAPHY

1. https://en.wikipedia.org/wiki/Facial_expression
2. <https://thoughtcatalog.com/january-nelson/2018/06/list-of-emotions/>
3. Paul Ekman and Wallace V Friesen. Constants across cultures in the face and emotion. *Journal of personality and social psychology*, 17(2):124, 1971.
4. Ekman, P., and Rosenberg. *What the face reveals: Basic and applied studies of spontaneous expression using the Facial Action Coding System (FACS)*. Oxford University Press, USA, E. L. 1997.
5. <http://mohammadmahoor.com/autsm-robot-assist>
6. <http://dreamfacetech.com>
7. Karen Simonyan, Andrew Zisserman, “Very Deep Convolutional Networks for Large-Scale Image Recognition”, Apr, 2015.
8. Kaiming He, Xiangyu Zhang, Shaoqing Ren, Jian Sun, “Deep Residual Learning for Image Recognition”, Dec, 2015.
9. Christian Szegedy, Sergey Ioffe, Vincent Vanhoucke, Alex Alemi. “Inception-v4, Inception-ResNet and the Impact of Residual Connections on Learning”, Aug 2016.
10. Gao Huang, Zhuang Liu, Laurens van der Maaten, Kilian Q. Weinberger, “Densely Connected Convolutional Networks”, Jan 2018.
11. Grigorios Tsoumakas and Ioannis Katakis. *Multi label classification overview*. Aristotle University of Thessaloniki, Greece

12. Cheng Ju and Aurelien Bibaut and Mark J. van der Laan. "The Relative Performance of Ensemble Methods with Deep Convolutional Neural Networks for Image Classification", Apr 2017.
13. Ben bright Benuwa, Yong Zhao Zhan, Benjamin Ghansah, Dickson keddy wornyo. A Review of Deep Machine Learning, 2016.
14. J. Schmidhuber, "Deep learning in neural networks: An overview," Neural Networks, vol. 61, pp. 85-117, 2015.
15. Francois Chollet, "Deep Learning with Python", November 2017.
16. Chigozie Enyinna Nwankpa, Winifred Ijomah, Anthony Gachagan, and Stephen Marshall, "Activation Functions: Comparison of Trends in Practice and Research for Deep Learning", November, 2018.
17. Forest Agostinelli, Matthew Hoffman, Peter Sadowski, Pierre Baldi, "Learning Activation Functions to Improve Deep Neural Networks", December, 2014.
18. Katarzyna Janocha, Wojciech Marian Czarnecki, On Loss Functions for Deep Neural Networks in Classification, Feb, 2017.
19. Hang Zhao, Orazio Gallo, Iuri Frosio, Jan Kautz, Loss Functions for Neural Networks for Image Processing, Nov, 2015.
20. Dami Choi, Christopher J. Shallue, Zachary Nado, Jaehoon Lee, Chris J. Maddison, George E. Dahl, On Empirical Comparisons of Optimizers for Deep Learning Jan, 2020.
21. Sebastian Ruder, An overview of gradient descent optimization algorithms, June, 2017.

22. Diederik P. Kingma, Jimmy Ba, Adam: A Method for Stochastic Optimization, Jan, 2017.
23. Jiuxiang Gu, Zhenhua Wang, Jason Kuen, Lianyang Ma, Amir Shahroudy, Bing Shuai, Ting Liu, Xingxing Wang, Li Wang, Gang Wang, Jianfei Cai, Tsuhan Chen, “Recent Advances in Convolutional Neural Networks”, Oct, 2017.
24. C.P. Sumathi, T. Santhanam and M.Mahadevi, “AUTOMATIC FACIAL EXPRESSION ANALYSIS A SURVEY”, December 2012.
25. Fabian Benitez-Quiroz ; Yan Wang ; Aleix M. Martinez, “Recognition of Action Units in the Wild with Deep Nets and a New Global-Local Loss”, 2017.
26. S. Mohammad Mavadati, M. H. Mahoor, K. Bartlett, Jeff Cohn, “DISFA: A Spontaneous Facial Expressions Dataset”, IEEE Transactions on Affective Computing, April-June, vol. 4 no. 2, pp. 151-160, 2013
27. Zhang ML, Li YK, Liu XY, Geng X. Binary relevance for multi-label learning: an overview. 2018.
28. Fabian Benitez-Quiroz, Ramprakash Srinivasan, Qianli Feng, Yan Wang, Aleix M. Martinez, “EmotioNet Challenge: Recognition of facial expressions of emotion in the wild”, Mar, 2017.
29. C. F. Benitez-Quiroz, R. Srinivasan, and A. M. Martinez. Emotionet: An accurate, real-time algorithm for the automatic annotation of a million facial expressions in the wild. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pages 5562–5570, 2016.

30. Phillip Ian Wilson and Dr. John Fernandez. Facial Feature Detection Using Haar Classifiers. JCSC 21, 4 (April 2006)
31. Chuanqi Tan, Fuchun Sun, Tao Kong, Wenchang Zhang, Chao Yang, Chunfang Liu, “A Survey on Deep Transfer Learning”, Aug 2018.
32. Chollet, F., 2015. Keras.
33. Liu Y, Cheng J, Yan C, Wu X, Chen F. Research on the Matthews correlation coefficients metrics of personalized recommendation algorithm evaluation. 2015.
34. <https://www.h5py.org>
35. Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In Advances in neural information processing systems, pages 1097–1105, 2012.
36. N. Srivastava, G. E. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. JMLR, 2014.
37. R. K. Srivastava, K. Greff, and J. Schmidhuber. Training very deep networks. 2015.
38. Li, W.; Abtahi, F.; and Zhu, Z. Action unit detection with region adaptation, multi-labeling learning and optimal temporal fusing. In IEEE Conference on Computer Vision and Pattern Recognition, 6766–6775. IEEE. 2017.
39. Li, W.; Abtahi, F.; Zhu, Z.; and Yin, L. Eac-net: A region-based deep enhancing and cropping approach for facial action unit detection. In IEEE International Conference on Automatic Face & Gesture Recognition, 103–110. IEEE. 2017.

40. Razvan Pascanu, Caglar Gulcehre, Kyunghyun Cho, Yoshua Bengio, “How to Construct Deep Recurrent Neural Networks”, Apr 2014.