Department of Electrical and Computer Engineering
North South University

CSE499

Senior Design Project Proposal

Project Title:

Automated student review system from facial expression with computer vision and deep learning

Course Instructor: **A K M Bahalul Haque** (Abh3)

Team Members:

Sadman Chowdhury Siam - 1711172042

Abrar Faisal - 1712677642

Niazi Mahrab - 1711248042

Abstract:

Detecting Emotion from a face is nowadays a very popular topic and computer vision researchers are continuously working on perfecting this challenging task. Facial emotion are widely used in applications like Snapchat, Face apps, Cameras etc. to predict emotion on faces to detect smiles and many more. The task is very challenging as the facial features of a person highly varies from one to another. As Convolutional Neural Networks are capable of detecting such complicated features So, the idea of deep learning has been used to tackle solve this problem. And the results of training the images with such models are significantly better than the state-of-the-art methods like manually extracting features, dimensionality reduction algorithm etc. With the use of transfer learning the training can be more optimized and the training time can be substantially improved.

Contents

Chapter 1: Introduction

In this chapter, we are going to briefly discuss about the whole idea behind facial emotion detection, our project aim and objectives with our motivation for doing this project.

1.1 Facial Features:

Human face to face communication is very important for our daily cooperation and plays a vital role in our normal communication. Automatic recognition of facial key points plays an important role in artificial intelligence and robotics to allow the machine understand and extract information from human face. Human facial features [1] carry a lot of information about themselves. It includes emotion, age, state of eyes etc. This information would act as fundamental application of many other problems like facial expression classification, tracking faces in videos and also for medical diagnosis. So, detecting facial features fast and precisely has become a great challenge. Facial features vary greatly from person to person and even for a single individual, there is a high variation in the image because of the 3D pose, size, position, angle of view and different lighting condition. Another challenge is the detection of these features has to be very fast. If we want to use these features in mobile apps for a larger problem the computation complexity will increase. So, slower detection of the features will diminish the performance of the app and will fail to achieve user satisfaction

1.2 Detecting Emotion:

The facial emotion can be detected manually by using state-of-arts methods [2]. For example, using Gabor feature and probability graphic model to specify relationship between pixels and its neighbors. But with the essence of deep learning computer vision has reached its new landmark and problems like object detection, semantic segmentation and object classification has become much easier than before. Different deep learning architectures have been proposed for facial emotion detection with the combination of convolutional neural networks and multi layered perceptron networks.

Human face to face communication is very important for our daily cooperation and plays a vital role in our normal communication. Automatic recognition of facial expression plays an important role in artificial intelligence and robotics and thus it is a need of the generation. Human facial expression carries a lot of information about themselves. Every human in their life has different kinds of feelings. Sometimes they feel happy, sometimes they are sad, sometimes they are so frustrated. In our projects we can see six types of human emotions-anger, disgust, fear, happiness, sadness, and neutrality. The pioneering study on emotion messages revealed from human faces was from Darwin's work [3]. Then Ekman defined six basic emotions which are universally associated with distinct facial expressions [4]. Figure-1 will show us those emotions. So, based on these six emotions our computer will recognize about student's average emotional situation and finally give us a review.

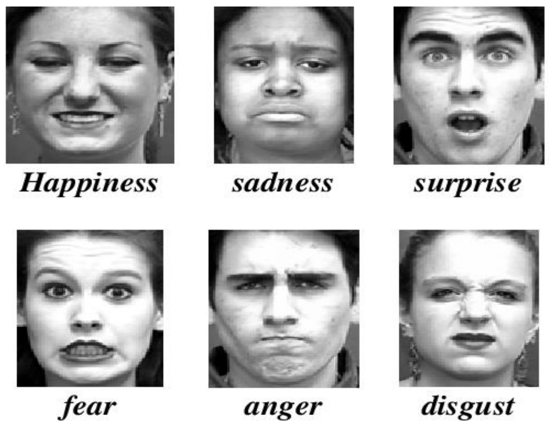


Figure 1: Feature extraction with CNN

1.3 Objective:

Previously there are many applications with this FER (facial emotion recognition) like human behavior understanding, detection of mental disorders, and synthetic human expressions etc. An overview of the early works in FER analysis can be found in [5]. So, our application is totally different from previous application. Our main goal is to make a student review system where we will get a review by computer about the mental conditions of students of a class. The Facial Action Coding System (FACS) is a human-observer-based system that has been developed to facilitate objective measurement of subtle changes in facial appearance caused by contractions of the facial muscles [6]. By this actions units FACS will be able to give us a description of all visibly discriminable expressions. As FACS indicates discrete and discernible facial movements and manipulations in accordance to the emotions of interest, digital image processing and analysis of visual facial features can allow for successful facial expression predictors to be trained. Recognition of facial expression by computer with high recognition rate is still a challenging and difficult task.

In this project we are using various deep learning methods to identify the emotional situation of students. We are going to use deep learning, neural network (convolutional neural network), transfer learning and computer vision. There are five stages in our whole projects. In figure 2 we have shown the steps. The approach is based on the assumption that a dataset face images corresponding to facial emotion is available to the system. After feature extraction from the images the network will be trained using back propagation. Facial expression recognition can be divided into two classes - geometrical feature-based approaches and appearance-based approaches [7]. The geometrical feature-based approaches represent geometric facial features which present the shapes and locations of facial components such as eyebrows, eyes, canthus, nose, mouth etc. These are also known as facial key points. We may have to face some experimental problems during the progress of our working because of the quality of images, illumination and some other disturbing factors.

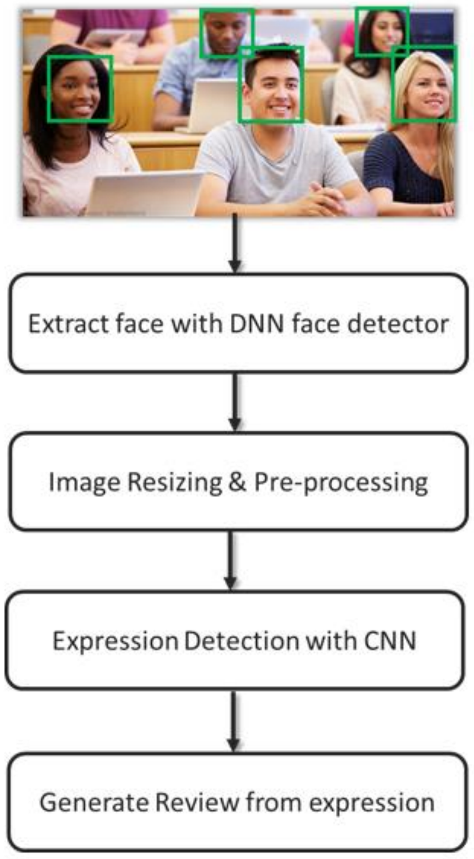

Extract face with DNN face detector
Image Resizing & Pre-processing
Expression Detection with CNN
Generate Review from expression

Figure 2: Pipeline of the whole structure

The appearance-based approaches states that the whole-face or specific regions in a face image are used for the feature extraction via optical flow or some kinds of filters. Some other approaches are able to discriminate expressions at a fine-grained level via the recognition of action units [8]. Some can fully automatically recognize expressions from image sequences. But some approaches still need to manually be labeled before passing as input to the network as we might not have enough labeled data for training and some feature points before the training procedure

1.4 Approaching the problem:

Facial Expression Recognition (FER) systems have been implemented in a multi-layer of ways and approaches. The majority of these approaches have been based on facial features analysis. Our first step is to perform image pre-processing. Illumination and contrast can vary in different images and hence introduce large impact on FER. [9] Pre-processing would be used to uniform shape and size of the input images as well as for removing unwanted noise in the background. We will use Video Capture library in OpenCV to capture image/video from cameras and use the information dynamic way. Open Source Computer Vision Library allows us to access the front camera in the device Additionally, input preprocessing is made before being used to predict facial emotion [10] In conventional expression recognition methods, they use cameras to capture images. From that image they extract enough information. Feature extraction transforms the original images and use some methods like Support Vector Machine (SVM), K Nearest Neighbor (KNN), Boosting, etc to build models using machine learning methods. But these models are not practical. But Convolutional Neural Networks (CNN) can automatically learn features in the original data. Also, Convolutional Neural Networks (CNN) have been used extensively in the recent years for computer vision and neural language processing [11]. Neural networks have been delivering simple and powerful solutions in areas relating to signal processing, artificial intelligence and computer vision. We have multi-layer perceptron networks in our project. In this project, we use a CNN with four convolutional blocks. Each layer will be composed of convolutions, max-pooling, and a Rectified Linear Unit (ReLU) activation function. A pre-trained Deep Neural Network model will be used to train the models. In the facial detection stage, the objective is to find the face so as to limit our Region of Interest (RoI) such that all further processing takes place from that Roil onwards. Interface’s 49-point landmark detector does an impressive job at localizing the points for the eyebrows and lips. The back-propagation neural network is a widely used neural network algorithm due to its simplicity. The back-propagation algorithm defines a systematic way to update the synaptic weights of multi-layer perceptron networks. Displaying the result is also important. To show our system to end user, we will design a simple user interface that will show the recognition result of facial emotion expression on the input image and display it on the screen. We are collecting data from different source like Kaggle dataset [12] for training data. We will use several algorithms to optimize the whole system and improve user experience.

1.5 Motivation:

Deep Learning is a hot topic in the field of Artificial Intelligence. With lots of data in our hand now it is possible to train amazing models that can provide good decisions. Modern applications of deep learning deals with data in form of images, audios, videos etc. Deep learning provides flexible models to deal with these data. Unlike non-parametric algorithms deep learning gradually learns to understand thing from past data. Some of the applications of deep learning includes in Self-Driving cars, health-care, Spam Detectors, game playing and many more.

From this project we aim to make a system that capture images through the front camera of laptop. Then processing the capture images to detect emotions of students in a classroom and describe overall students feeling in real time. We will exploit the power of deep learning technique to learn a facial emotion recognition (FER) model. The entire system will execute a real-time emotion detection which will help us to find out performance of students in a certain class [13]. From the result we will get several information about students, course and teacher. We will use bunch of new technology, libraries and algorithms. We aim to train a prediction model with high accuracy because robustness of a model is also an important aspect.

1.6 Project Overview:

We are going to actually make a student review system. Assume there are 20 students in a class and it takes long fifty minutes. We will snap a picture in every minute. Then our deep learning, computer vision is collecting, labeling and separating each photo and CNN (convolutional neural network) will give us the emotional react of every photo in every minute. Then every minute we get 20 photos of different students and CNN will give us a feedback of average of these 20 photos about their emotion. If most of the students are happy then it will give students are happy, if most are sad then it will give review that students are sad. Then in 50 minutes class, we get reviews in every minute. In 50 minutes, we get 50 reviews. Then finally after the class we will average these 50 reviews and we will get a result what is the situation of the class and that is the final review of class.

Chapter 2: Literature Overview

While working on this research project, we studied various research papers related to our project topic and picked a few papers on deep learning, neural network, CNN, transfer learning and computer vision. We will be reviewing these papers here.

2.1 Deep Learning

Deep Learning is a subfield of machine learning concerned with algorithms inspired by the structure and function of the brain called artificial neural networks. For example, in image processing, lower layers may identify edges, while higher layers may identify the concepts relevant to a human such as digits or letters or faces.

Figure 3: Deep Learning

In deep learning, a computer model learns to perform classification tasks directly from images, text, or sound. Deep learning models can achieve state-of-the-art accuracy, sometimes exceeding human-level performance. Models are trained by using a large set of labeled data and neural network architectures that contain many layers.

2.2 Overview

One of the hottest topics right now in tech is Artificial Intelligence. With so many data produced every minute we have now access to huge amount of resource. It has taken Artificial Intelligence to next level with the power of Machine Learning. In machine learning we apply the knowledge of statistics to train a computer to learn from data. Some of the powerful machine learning algorithms are Linear Regression, Support Vector Machine, Naïve Bayes etc. Although these algorithms work great but for large dataset like images, video, audio etc. they can't perform good enough. This gave birth to the new subfield of machine learning called Deep Learning. Deep Learning is concerned with algorithms that are inspired by the structure and function of the human brain called Artificial Neural Network. It has made learning algorithms much better and easier to use. With the discovery of several complicated architectures Deep Learning has brought revolutionary advances in the field of machine learning and AI. Below is a simple model of a neural network.

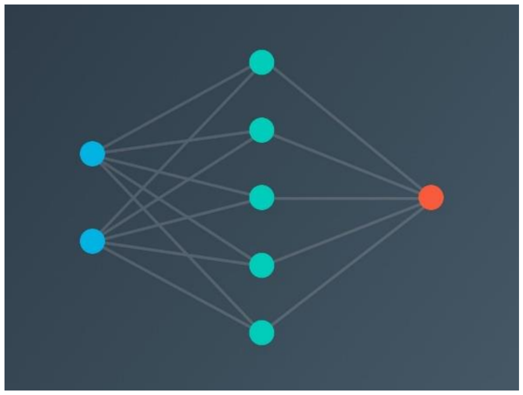


Figure 4: Neural Network

As we have access to millions of data so, it has become much easier for us to train models using deep learning. It is seen that the performance of deep learning algorithm is several folds better than the traditional machine learning algorithm. The more the data the better the prediction. With deep learning [14] we can design bigger models with heavy computational power. Before deep learning developers had to first extract features manually for sending to the classifier for training. But now with enough training Neural Network themselves can learn to extract features from the input data and perform classification tasks.

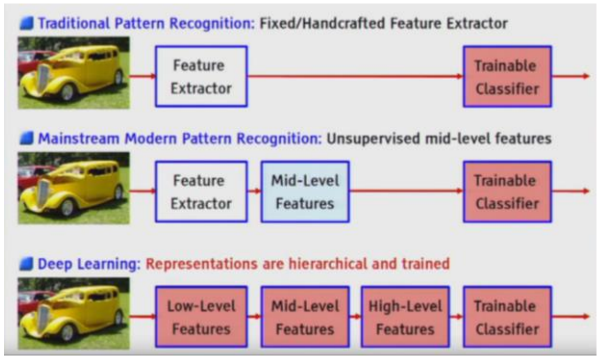

Traditional Pattern Recognition: Fixed/Handcrafted Feature Extractor
Feature Extractor
Trainable Classifier
Mainstream Modern Pattern Recognition: Unsupervised mid-level features
Feature Extractor
Mid-Level Features
Trainable Classifier
Deep Learning: Representations are hierarchical and trained
Low-Level Features
Mid-Level Features
High-Level Features
Trainable Classifier

Figure 5: Feature extraction example

2.3 Neural Network

2.3.1 Evolution of Neural Network

The investigation of the human mind is a great many years old. With the approach of current electronics, it was just normal to attempt to bridle this reasoning procedure. The initial move toward fake neural systems came in 1943 when Warren McCulloch, a neurophysiologist, and a youthful mathematician, Walter Pitts, composed a paper on how neurons may function. They displayed a straightforward neural system with electrical circuits.

In 1949, Donald Hebb reinforced the concept of neurons in his book, The Organization of Behavior. It pointed out that neural pathways are strengthened each time they are used. As computers became more advanced in the 1950's, it was finally possible to simulate a hypothetical neural network. The first step towards this was made by Nathanial Rochester from the IBM research laboratories. Unfortunately for him, the first attempt to do so failed. But later attempts were successful. It was during this time that traditional computing began to flower and, as it did, the emphasis in computing left the neural research in the background.

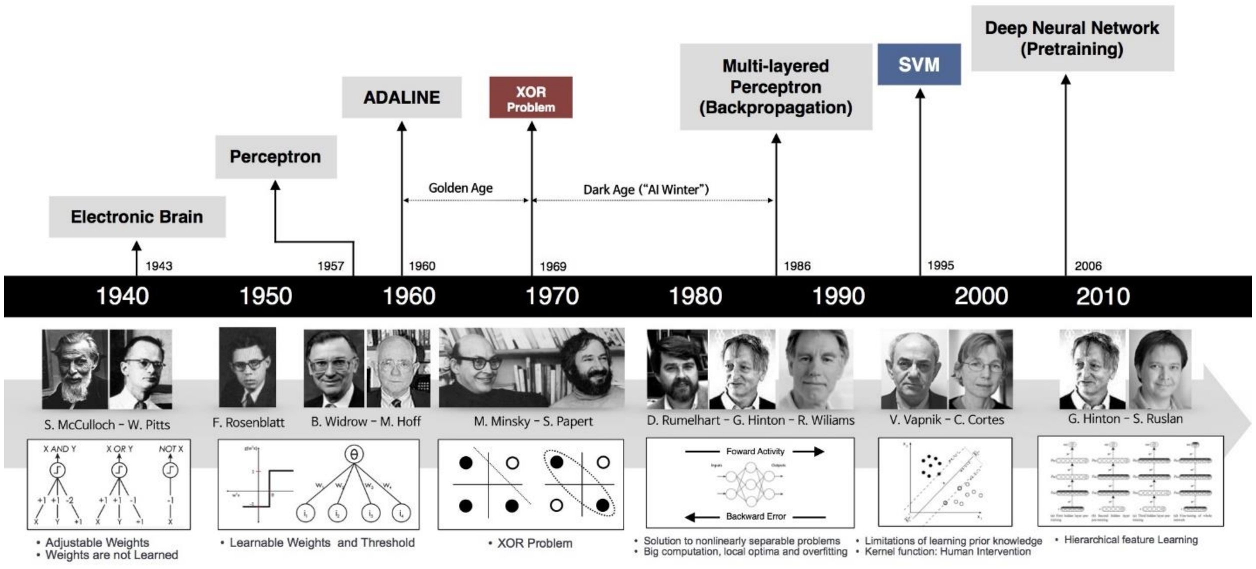

Electronic Brain
Perceptron
ADALINE
XOR Problem
Multi-layered Perceptron (Backpropagation)
SVM
Deep Neural Network (Pretraining)
Golden Age
Dark Age ("AI Winter")
1943
1957
1960
1969
1986
1995
2006
1940
1950
1960
1970
1980
1990
2000
2010
S. McCulloch – W. Pitts
F. Rosenblatt
B. Widrow – M. Hoff
M. Minsky – S. Papert
D. Rumelhart – G. Hinton – R. Wiliams
V. Vapnik – C. Cortes
G. Hinton – S. Ruslan
X AND Y
X OR Y
NOT X
Foward Activity
Backward Error
• Adjustable Weights
• Weights are not Learned
• Learnable Weights and Threshold
• XOR Problem
• Solution to nonlinearly separable problems
• Big computation, local optima and overfitting
• Limitations of learning prior knowledge
• Kernel function: Human Intervention
• Hierarchical feature Learning

Figure 6: History

Despite the later success of the neural network, traditional von Neumann architecture took over the computing scene, and neural research was left behind. Ironically, John von Neumann himself suggested the imitation of neural functions by using telegraph relays or vacuum tubes.

In the same time period, a paper was written that suggested there could not be an extension from the single layered neural network to a multiple layered neural network. In addition, many people in the field were using a learning function that was fundamentally flawed because it was not differentiable across the entire line. As a result, research and funding went drastically down.

The idea of a computer which programs itself is very appealing. If Microsoft's Windows 2000 could reprogram itself, it might be able to repair the thousands of bugs that the programming staff made. Such ideas were appealing but very difficult to implement. In addition, von Neumann architecture was gaining in popularity. There were a few advances in the field, but for the most part research was few and far between.

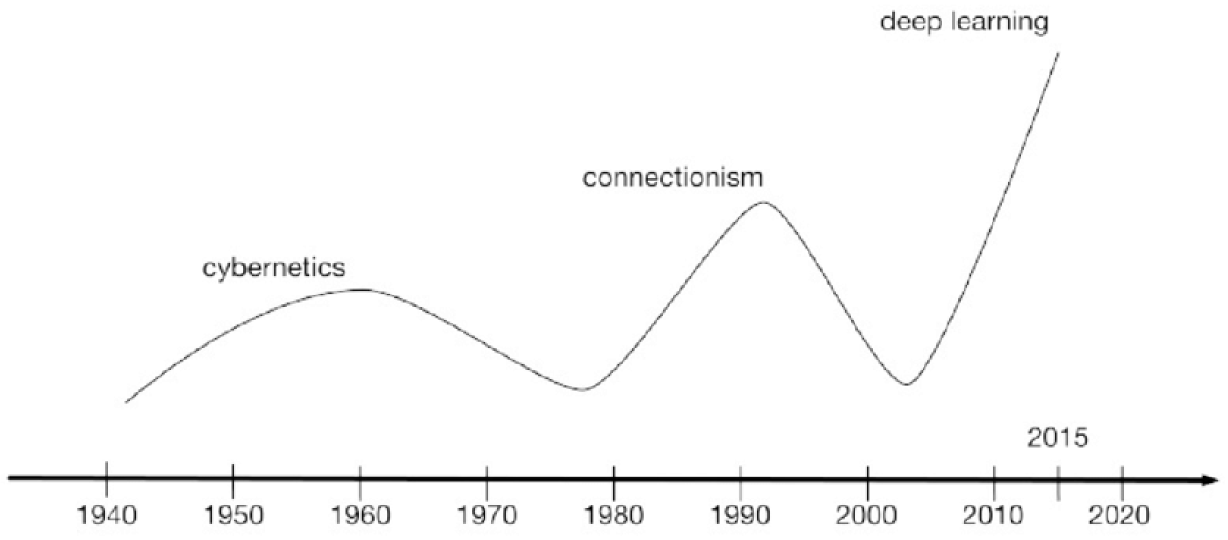


Figure 7: Historical wave of neural network

Several other steps have been taken to get us to where we are now; today, neural networks discussions are prevalent; the future is here! Currently most neural network development is simply proving that the principal works. This research is developing neural networks that, due to processing limitations, take weeks to learn.

2.3.2 Neural Networks Overview

Neural Networks are some combination of nodes and edges. Its architectures are inspired from the human neuron. These networks learn from observing the world instead of manually teaching them programmatically. Data are fed as input in the network of interconnected nodes for training. These individual nodes are called perceptron or artificial neurons. They are the basic unit of a neural network [15]. Each individual perceptron looks at input data and decides how to categorize the data. The figure below can provide us some intuition about the perceptron.
Several perceptions are combined together to build a complex model. Each of these works as different unit and their combined units then lead to a better model. By adding more layers and more nodes we can then build a network called deep neural network. These are extremely powerful for generating non-linear boundaries to provide better prediction.

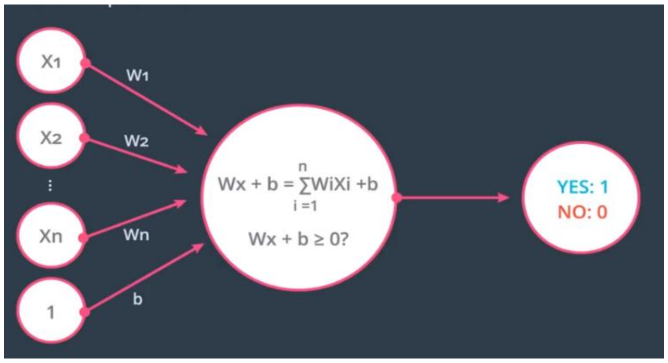


Figure 8: Perceptron

Weights and bias are used as parameters in the network to give predictions. These weights start out as random numbers. When inputs come into perceptron, they get multiplied by these weights that are assigned to a particular input. The network adjusts the weights based on any errors in categorization that results from the previous weights. This process of updating the weights from errors is call training a neural network. During the training process the network uses forward propagation through the perceptron to provide a prediction. Then it compares its prediction with the original labeled output and calculates an error. Then it backpropagates the error and uses algorithms like Gradient Descent to update the weights. In this way the network learns.

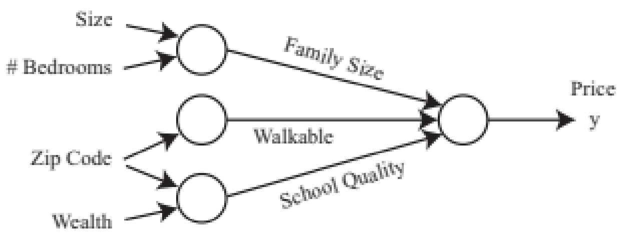


Figure 9: Neural Network

All the computation that make this networks work are based on mathematical theories of calculus, linear algebra, statistics and regression [16]. For instance, neural network uses dot product to feed forward the input through the network. It uses functions like cross entropy loss, soft-max etc. to compute the errors based on probability. Than it uses the concept of gradient from calculus to minimize the errors. Some other functions called Sigmoid, Relu, Tanh etc. are used as activation functions in neural network. Some key terms in the network are nodes, hidden layers, weights, bias, learning rate etc. These are also known as parameters and hyper parameters of the network. These are used for designing the architecture of the network and improving

2.3.3 Training Neural Networks

Neural Networks are some combination of nodes and edges. Its architectures are inspired from the human neuron. These networks learn from observing the world instead of manually teaching them programmatically. Data are fed as input in the network of interconnected nodes for training. These individual nodes are called perceptron or artificial neurons. They are the basic unit of a neural network [17]. Each individual perceptron looks at input data and decides how to categorize the data. The figure below can provide us some intuition about the perceptron.

Several perceptron is combine together to build a complex model. Each of these works as different unit and their combined units then lead to a better model. By adding more layers and more nodes we can then build a network called deep neural network. These are extremely powerful for generating non-linear boundaries to provide better prediction.

Weights and bias are used as parameters in the network to give predictions. These weights start out as random numbers. When inputs come into perceptron, they get multiplied by these weights that are assigned to a particular input. The network adjusts the weights based on any errors in categorization that results from the previous weights. This process of updating the weights from errors is call training a neural network. During the training process the network uses forward propagation through the perceptron to provide a prediction. Then it compares its prediction with the original labeled output and calculates an error. Then it backpropagates the error and uses algorithms like Gradient Descent to update the weights. In this way the network learns.

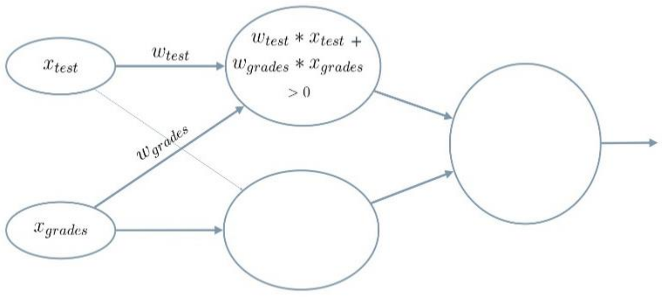


Figure 10: Combined Perceptron

All the computation that make this networks work are based on mathematical theories of calculus, linear algebra, statistics and regression [18]. For instance, neural network uses dot product to feed forward the input through the network. It uses functions like cross entropy loss, soft-max etc. to compute the errors based on probability. Than it uses the concept of gradient from calculus to minimize the errors. Some other functions called Sigmoid, Relu, Tanh etc. are used as activation functions in neural network. Some key terms in the network are nodes, hidden layers, weights, bias, learning rate etc. These are also known as parameters and hyper parameters of the network. These are used for designing the architecture of the network and improving training performance. Concepts of Drop-outs, Regularizations etc. are also used to improve efficiency and prevent problems of overfitting and underfitting of the network during training.

Different types of neural network design are now in use to tackle different types of problems. CNN are special type of neural network architecture that are used for working with image and video data to extract features form images. RNN is used in natural language processing, audio processing and for working with scenarios where we need to rely on previous state to predict future states like understanding sentiments, speech recognition etc. There are also many other deep architectures like R-CNN LeNet [19] that uses combination of different architectures to build a more complex network.

2.4 CNN:

CNN of convolutional neural networks are a special kind of architecture that are widely used when working with image data. CNNs were one of the key innovations that led to the deep neural network renaissance in computer vision. To understand CNN, we first need to know about how a computer interprets an image. In computer images are just collection or array of grid cells called pixels. Each pixel has a numerical value from 0-255. This value indicates the color channel of the image. Zero indicates the darkest and 255 indicates the lightest color. When we work with image data, we generally have huge number of inputs. If we have a 28x28 px grey-scaled image then the computer interprets the image as 2D array of 28X28 pixel values. We have (28 X 28) 784 inputs for processing such a small image. If we want to feed the input to a multi layered perceptron (MLP) we need to feed a vector of input size 784 to it. Now if we are working with color images than we will have to deal with 3 extra arrays of 28X28 arrays for each red, green and blue color channel. In such case the input size will increase many folds and working with such large number of inputs in MLP is very problematic. That's why CNN [20] was discovered to specifically working with large image inputs.

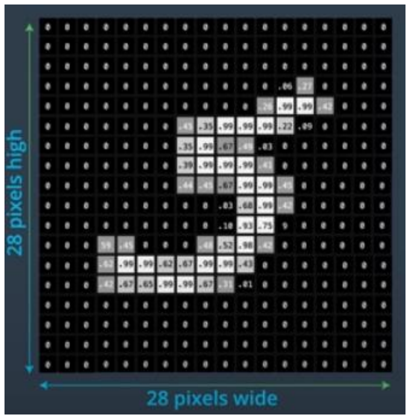


Figure 11: Image Visualization

Convolutional Networks also known as CONVNETS are neural networks that share their parameters across space. In that they can remember spatial information. In MLP inputs are looked at individually on the other hand CNN has the ability to look at the image as a whole i.e. the network can analyze groups of pixels at a time. The key to preserving this spatial information are called convolutional layer. A convolutional layer applies sets of convolutional filters also called kernels to input image. These filters can extract different features from the image like edges of objects in the image or different colors that distinguishes different classes of objects in the image. For example, look at the figure below. It is a visualization of the second layer of a CNN, how it is extracting patterns like circles and stripes from corresponding images.

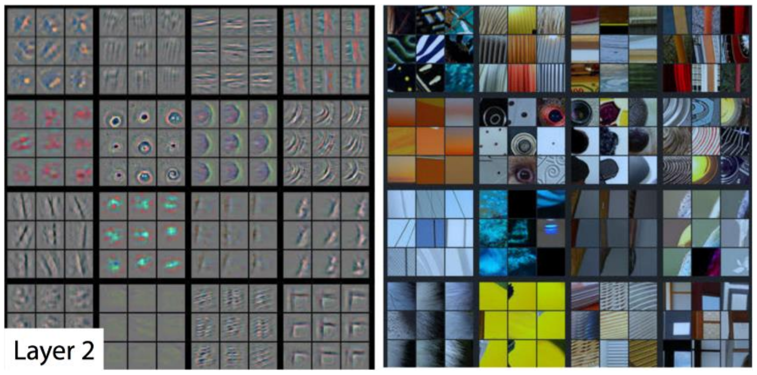


Figure 12: CNN Layer Visualization

In summary CNN are stack of convolutional that are used to extract patterns from images and pooling layers in between them to reduce dimensionality. [21] These layers are then combined with a fully connected layer to build the final architecture. As we get more deeper in the layer each individual feature maps in those layers function to extract more and more complicated patterns of the image.

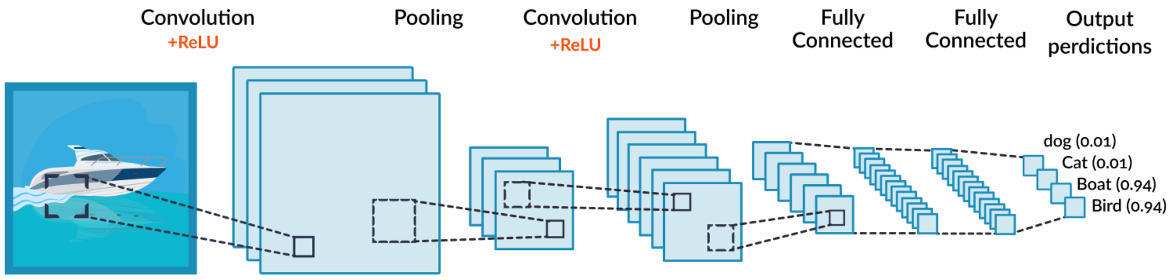


Figure 13: CNN Architecture

2.5 RNN:

The neural net architecture like the artificial neural network and CNN architecture uses the current input only. But these networks can't handle temporal dependencies or dependencies over time. RNN are neural nets that can handle these temporal dependencies. The R here means recurrent. These networks are called recurrent because with RNN we perform same task for each element in the input sequence. RNN are networks with loops in them which allows information to persist. It maintains an internal memory element also known as states. The memory element gives the network idea on the context. The principle behind the architecture is quite same as feed forward neural networks. But there are two main differences. The first is the way we define the input and output in the RNN. This could be one to one, one to many or many to many. Instead of training the network with single input and single output at each timestep, in RNN we train with sequences since previous input matter. The second input is because of the memory element. In RNN we have 2 separate input. One comes from the provided input and another comes from the memory element. This is why the output depends on both input and the previous input. The output can be expressed as a function of time i.e. y = F (x_t, x_t-1, x_t-2…., W). But there are several dependency issues in RNN. To tackle these advance structures were designed like LSTM cell.

2.5.1 LSTM:

Simple RNN has a problem called long-term dependency problem i.e. remembering information for long periods of time which is caused by gradient flow. LSTM has same chain like structure like RNN but the hidden layer architecture is quite different. LSTM uses a structure also called gates which gives it the ability to remove or add information to the cell state. Gate are a way to optionally let information through. It keeps track of memories separately. One act as Long term memory and another act as short term memory. So it has in total 3 inputs – Current event, LTM input and STM input. There are 4 different gates. The input from STM and the event enters in 1 gate also known as learn gate. The function can be represented as Nt = tanh(Wn[STM_t-1, Et] + bn). Some part of the input is then ignored by multiplying with a ignore factor I = G(Wi[STM_t-1, Et] +b)

Chapter 3: Convolutional Neural Network

3.1 Overview:

Convolutional Networks also known as CONVNETS are neural networks that share their parameters across space. In that they can remember spatial information. In MLP inputs are looked at individually on the other hand CNN has the ability to look at the image as a whole i.e. the network can analyze groups of pixels at a time. The key to preserving this spatial information are called convolutional layer. A convolutional layer applies sets of convolutional filters also called kernels to input image. These filters can extract different features from the image like edges of objects in the image or different colors that distinguishes different classes of objects in the image. For example, look at the figure below. It is a visualization of the second layer of a CNN, how it is extracting patterns like circles and stripes from corresponding images.

3.2 Working Methodology of CNN:

There are different types of neural network architectures, each has its own benefits. Neural network can represent powerful functions. For image related problem Convolutional Neural Network (CNN) is everywhere. Convolutional Neural Network (CNN) was designed to convert image data to output variable. Convolutional Neural Network (CNN) is specific for image which means that Convolutional Neural Network (CNN) gives some advantages over other neural networks like Feed-Forward- Neural Network, Artificial Neural Network etc. Images have some unique properties which we called spatial features. Suppose that we are working on a dataset where image size is 1000 * 1000, which means that we need 1 million neurons in input layer. If the image has three channels then input layer will be 3 million pixels, which is not practical. This leads us to Convolutional Neural Network. CNN can extract the feature of image and convert it to lower dimension without losing its original properties. Also CNN can develop internal representation of two dimensional image. All CNN follow a similar procedure as shown below:

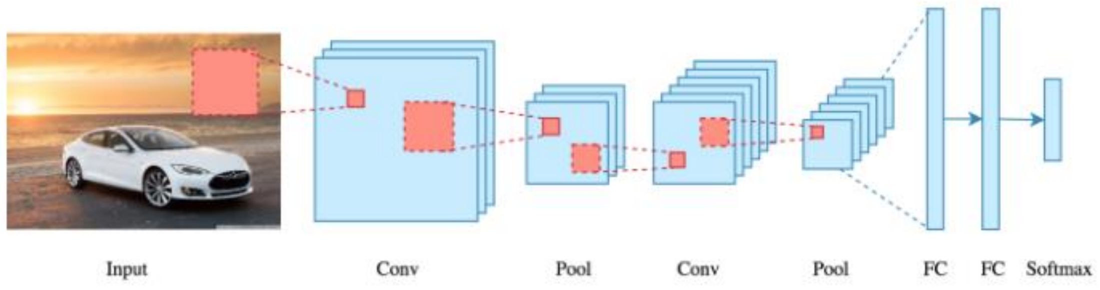
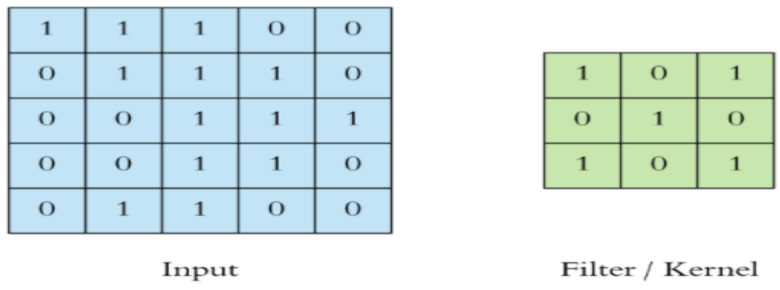


Figure 24: CNN Architecture

Every image is formed by combining vertical and horizontal edges. Convolution operation is used to detect edges by using some filters. Suppose that we have an input image of 5*5 dimension with a filter of 3*3 dimension. This is called 3*3 convolution because of filter's shape.



1	1	1	0	0
0	1	1	1	0
0	0	1	1	1
0	0	1	1	0
0	1	1	0	0

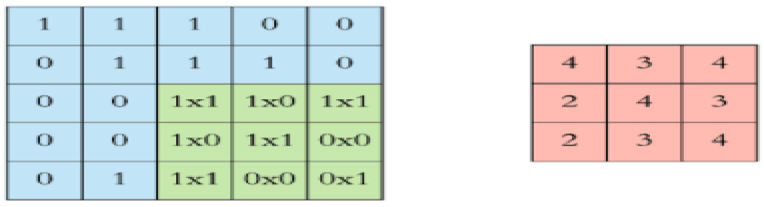
Input

1	0	1
0	1	0
1	0	1

Filter / Kernel

Figure 35: CNN Filter

We perform convolutional operation by overlay the filter over the input. At every location we scan the filter across the image to multiply matrix and sum the result. First of all, 3*3 filter matrix get multiplied with 3*3 size of image. Then we shift one column right up to end. After that we shift one row and so on. We will continue till get results in the feature map. After performing convolution, the output will be 3*3.



1	1	1	0	0
0	1	1	1	0
0	0	1x1	1x0	1x1
0	0	1x0	1x1	0x0
0	1	1x1	0x0	0x1

4	3	4
2	4	3
2	3	4

Figure 46: CNN Filter and Kernel

If we have n*n image size and f*f filter size then the result after convolution will be:

(n*n) * (f*f) = (n – f + 1) * (n – f + 1). This formula is only applicable for above situation.

In the above example, after each convolution operation we shift the filter one step. But it is not fixed. We can shift the filter two or more steps using stride. Stride determines how many steps we move in each row or column. By default, the value of stride is one. When we want less overlap then we denote bigger strides. Also stride reduce the size of the output or feature map.

If we have n*n image size and f*f filter size and stride =s then (n*n) * (f*f) = [(n – f + 1) /s] *[(n – f + 1)/s]

From the above equations we observe that the output size is smaller than the input. Sometimes we need to maintain the output dimensions as the input. Then we need a term called padding. Padding is a process used in CNN that maintain the dimension of the feature map as input otherwise the dimension of the feature map is reduced. After applying padding

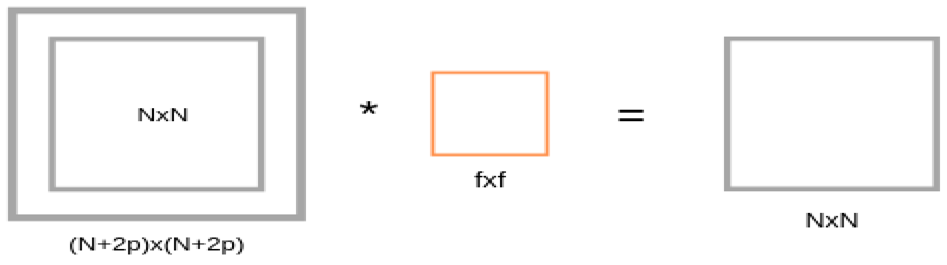


Figure 57: CNN Input and Output

Padding depends on the dimension of filter. When we apply padding then the input matrix will be (N+2p) * (N+2p). Also overlay filter over the input will give us output matrix which is same as the input size. So, when we apply padding in n*n image size and f*f filter size with stride =s, then output== [(n +2p– f + 1) /s] *[(n +2p– f + 1)/s].

3.3 Image Classification with Deep Learning:

3.3.1 Fully Connected Network Structure (MLP)

Computer recognize any grey scale image as an array. Greater values for each grey scale is called as pixel and pixel have numerical values. In MNIST database each image is 28*28 pixels. Computer recognize these images as 28*28 array. In a typical grey scale image white color is encoded as value of 255 and black as 0. Others are in between. These MNIST data set are gone through a preprocessing step. They have rescaled the images so that the pixel values are in arrange of 0 to 1 instead of 0 to 255.To rearrange pixel values from 0 to 255 to 0 to 1 we have to divide every pixel value by 255.This step is called normalization. It is a common use in many deep leering techniques. Normalization will help our algorithm to train better. Neural network relies on gradient calculation. Normalization ensures consistency while calculating gradient. With the normalize data we can classify image using Multilayer perceptron layer. Multilayer perceptron layer (MLP) takes only vector as input. For using MLP with images we have to convert image array into a vector. This process is known as flattening.

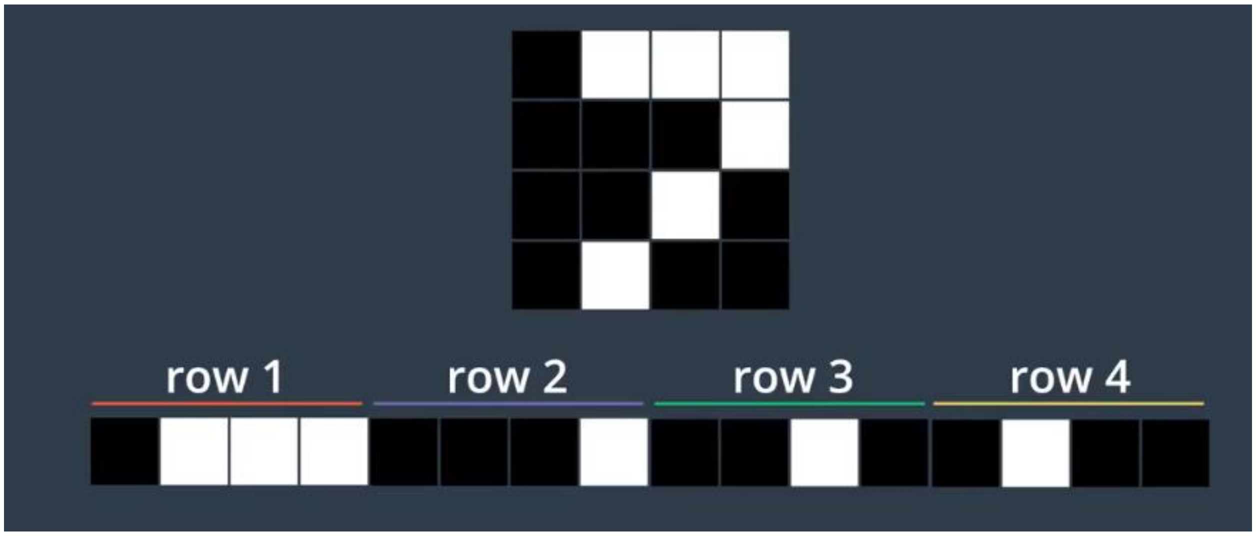


Figure 68: Feature extraction with CNN

In case of a 4*4 image we have a matrix with 16-pixel values. Instead of representing the image as 4 *4 matrix value we can convert this to a vector which has 16 entries.1st four entries are corresponding to row1 ,2nd four entries are corresponding to row2 and so on. After converting the image into a vector, we can the vector as a input of MyPlate normalizing the data, we will create a neural to train our data for discovering the patterns. When training will finish our network will looking at new data which is known as test data. In MNIST data set each is 28* 28 pixels. So, it will convert into a vector with 784 entries. It has 10 output class. We will give a vector with 784 entries in as input in MLP and it gives us 10 output class. The layers in between are known as hidden layer. How many hidden layers will be in between input and output layer is up to us? More hidden in layer means more ability to detect complex patterns. For small images one or two hidden layer work fine. It is better to avoid unnecessary complexity. Let's consider an example.

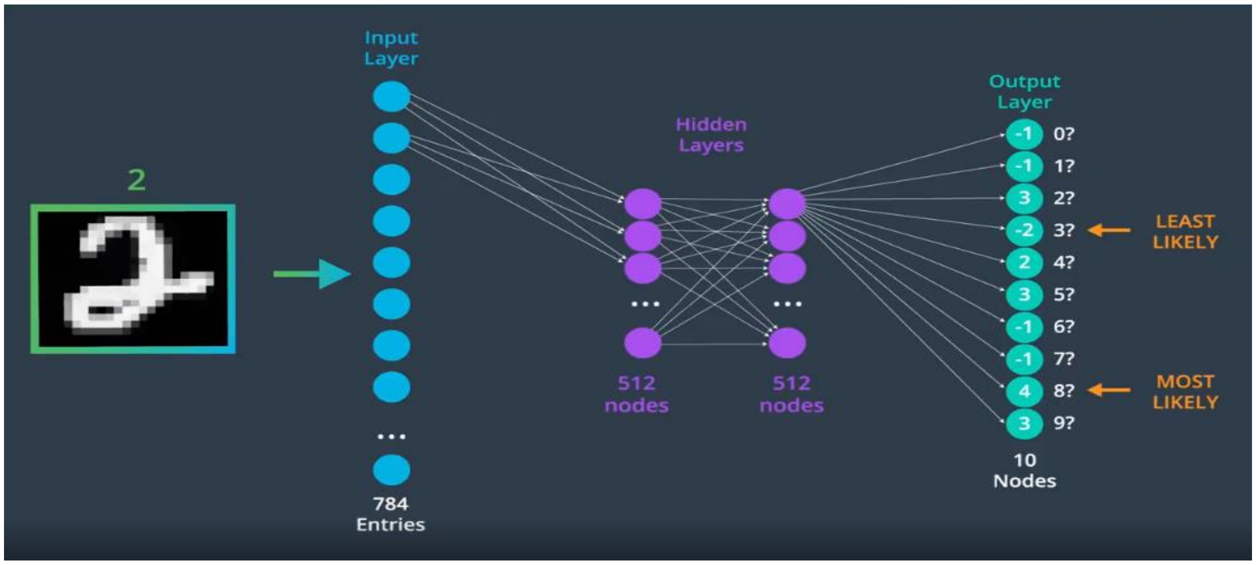


Figure 79: Neural Network Layers

Here the input image is hand written 2. We give this as input to MPL network and get 10 class scores output layer.4 is the highest value here and -2 is the lowest. The network believes that the input image is most likely to be 8 and least likely to be 3. But this is incorrect prediction because we know that the label is 2. We will tell our network to learn from this mistake. We measure any mistakes that network makes using loss function whose is to measure the difference between predicted and true class labels. Then using backpropagation, we can calculate the gradient of the loss with respect to the weights. In this way we quantify how bad a network is and find out which weights in the network are responsible for any errors. After that using a optimization function like gradient descent we can calculate better weight for the network. The standard method for minimizing the loss and optimizing the output value is called gradient descent. There are many ways to calculate gradient descent and each method has corresponding optimizer. By applying SoftMax function we can make our output into a probability. Applying categorical cross entropy loss function into output values we can get more accurate prediction. Cross entropy gets loss value by applying log function to output. Categorical cross entropy defines in such a way that the model predicts lower loss when prediction and true class label agree. It predicts higher loss when prediction and true class label disagree.

3.3.2 MLP VS CNN

So far, we have seen that MLP predict image classifier with higher accuracy in MNIST dataset. In MNIST data set all images are preprocessed and nicely distributed. But other data sets all images   will not nice, clean and preprocess. In the case of real-world messy data other structure like CNN will shine over MLP. In MLP, first we must convert the image into a vector with no special structure. CNN in contrast have the ability to work with multi-dimension data. MLP only use fully connected layers and only accepts vector as input. On the other hand, CNN use sparsely connected layers and accept matrices as input.

3.3.3 CNN for Image Classification

A Convolutional neural network is a special kind of neural network in that it can remember spatial information. Normal neural network looks at individual inputs. Convolutional neural network can look into input images as whole or analyze group of pixels at a time. The key to preserving the spatial information is convolutional layer. A convolutional layer applies a series of different filters also known as convolutional kernels to an input image. The resulting filter images has different appearances. The filters may have extracted   features like the edges of objects in that image or colors that distinguish the different classes of image.

To detect changes in intensity in an image, we will be using and creating specific image filters that look at groups of pixels and react to alternating patterns of dark/light pixels. These filters produce an output that shows edges of objects and differing textures. The convolutional layer is produced by applying a series of many different image filters, also known as convolutional kernels, to an input image.

We can control the behavior of convolutional layer by specifying the number of filters and the size of each filter. For instance, to increase the nodes in a convolutional layer, we can increase the number of filters. To increase the size of the detective patterns we could increase the size of filter. There are even more hypo-parameters. One of the hypo-parameters refers to the stride of the convolutional layer. The stride is just the amount by which filter slide over the image. When stride is 1 the input and output image size are same. By increasing the stride will decrease the size of the convolutional layer. Also applying padding, we could give the filter more space to move.

There is a common layer in CNN which we call pooling layer. Pooling layers take convolutional network as input. In CNN higher dimension means need to use more parameters which commit to overfitting. Thus, we need a method to reduce dimensionality. Pooling layers use in CNN to reduce the dimension. Pooling layers have two different types. One type is max-pooling layer. Max pooling layer take a stack of feature map as input. Applying max pooling will reduce size of the resultant image and also retain the image information.

There is another pooling called average pooling. Average pooling layer takes the average value of the features from the feature maps. So in a 2x2 window, this operation will see 4 pixel values, and return a single, average of those four values, as output. This kind of pooling is typically not used for image classification problems because max pooling is better at noticing the most important details about edges and other features in an image.

Pooling operations lost some image information. They throw away some pixel information in order to get a smaller, feature-level representation of an image. This works quite well in tasks like image classification, but it can cause some issues.

When we use CNN for facial recognition, we notice to some features like two eyes, a nose, and a mouth. And those pieces, together, form a complete face, A typical CNN that is trained to do facial recognition will identify these features to recognize face. But pooling throws away some information So, there has been research into classification methods that do not discard spatial information as in the pooling layers, and instead learn to spatial relationships between parts like between eyes, nose, and mouth. One such method, for learning spatial relationships between parts, is the capsule network.

3.3.4 Capsule Networks

Capsule Networks provide a way to detect parts of objects in an image and represent spatial relationships between those parts. Capsule networks are able to recognize the same object in different angle and different poses. Capsules are a collection of nodes, where each node contains information about a specific part; part properties like width, orientation, color, and so on. Capsule gives outputs a vector with some magnitude and orientation.

Magnitude (m) = the probability that a part exists; a value between 0 and 1.

Orientation (theta) = the state of the part properties.

These output vectors allow us to do routing math to build up a data tree that recognizes whole objects by combining all the nodes.

3.4 Advanced CNN Architectures for Object-Detection:

R-CNN:

R-CNN (Girshick et al., 2014) is short for "Region-based Convolutional Neural Networks". The main idea is composed of two steps. First, using selective search, it identifies a manageable number of bounding-box object region candidates. To bypass the problem of selecting a huge number of regions, Ross Girshick et al. proposed a method where we use selective search to extract just 2000 regions from the image and he called them region proposals. Therefore, now, instead of trying to classify a huge number of regions, we can just work with 2000 regions. These 2000 region proposals are generated using the selective search algorithm.

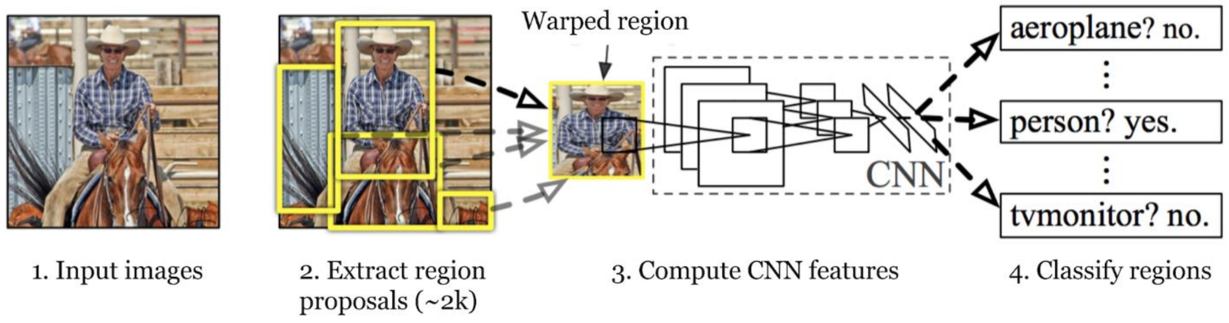


Figure 20: R-CNN Architecture

Selective Search:
1. Generate initial sub-segmentation, we generate many candidate regions.
2. Use greedy algorithm to recursively combine similar regions into larger ones.
3. Use the generated regions to produce the final candidate region proposals.

Problems with R-CNN

It still takes a huge amount of time to train the network as you would have to classify 2000 region proposals per image.

It cannot be implemented real time as it takes around 47 seconds for each test image.

The selective search algorithm is a fixed algorithm. Therefore, no learning is happening at that stage. This could lead to the generation of bad candidate region proposals.

YOLO:

YOLO (“You Only Look Once”) is an effective real-time object recognition algorithm. (YOLO) is a network that uses Deep Learning (DL) algorithms for object detection. YOLO performs object detection by classifying certain objects within the image and determining where they are located on it. YOLO is orders of magnitude faster (45 frames per second) than other object detection algorithms. The limitation of YOLO algorithm is that it struggles with small objects within the image, for example it might have difficulties in detecting a flock of birds. This is due to the spatial constraints of the algorithm.

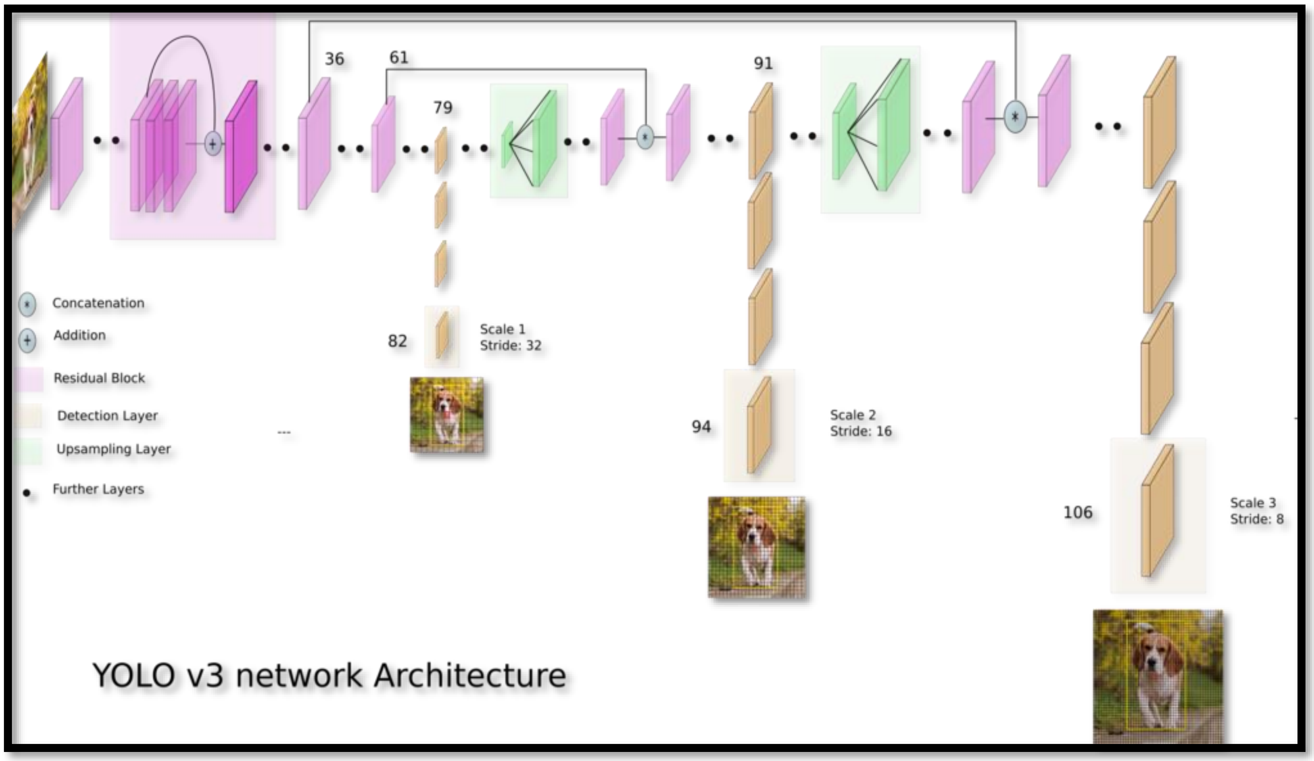


Figure 28: Yolo v3 Network Architecture

There are a few different algorithms for object detection and they can be split into two groups:

Algorithms based on classification. They are implemented in two stages. First, they select regions of interest in an image. Second, they classify these regions using convolutional neural networks. This solution can be slow because we have to run predictions for every selected region. A widely known example of this type of algorithm is the Region-based convolutional neural network (RCNN) and its cousins Fast-RCNN, Faster-RCNN and the latest addition to the family: Mask-RCNN. Another example is RetinaNet.

Algorithms based on regression – instead of selecting interesting parts of an image, they predict classes and bounding boxes for the whole image in one run of the algorithm. The two best known examples from this group are the YOLO (You Only Look Once) family algorithms and SSD (Single Shot Multibox Detector). They are commonly used for real-time object detection as, in general, they trade a bit of accuracy for large improvements in speed.

To understand the YOLO algorithm, it is necessary to establish what is actually being predicted. Ultimately, we aim to predict a class of an object and the bounding box specifying object location. Each bounding box can be described using four descriptors:

1. center of a bounding box (bxby)
2. width (bw)
3. height (bh)
4. value cis corresponding to a class of an object

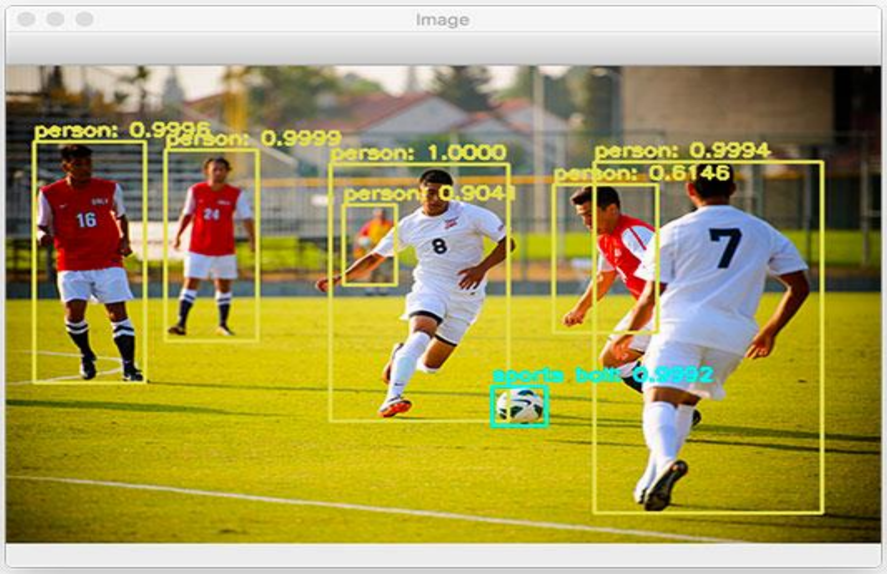


Figure 22: R-CNN

The Predictions Vector:

The first step to understanding YOLO is how it encodes its output. The input image is divided into an S x S grid of cells. For each object that is present on the image, one grid cell is said to be “responsible” for predicting it. That is the cell where the center of the object falls into.

Each grid cell predicts B bounding boxes as well as C class probabilities. The bounding box prediction has 5 components: (x, y, w, h, confidence). The (x, y) coordinates represent the center of the box, relative to the grid cell location (remember that, if the center of the box does not fall inside the grid cell, then this cell is not responsible for it). These coordinates are normalized to fall between 0 and 1. The (w, h) box dimensions are also normalized to [0, 1], relative to the image size. Let's look at an example:

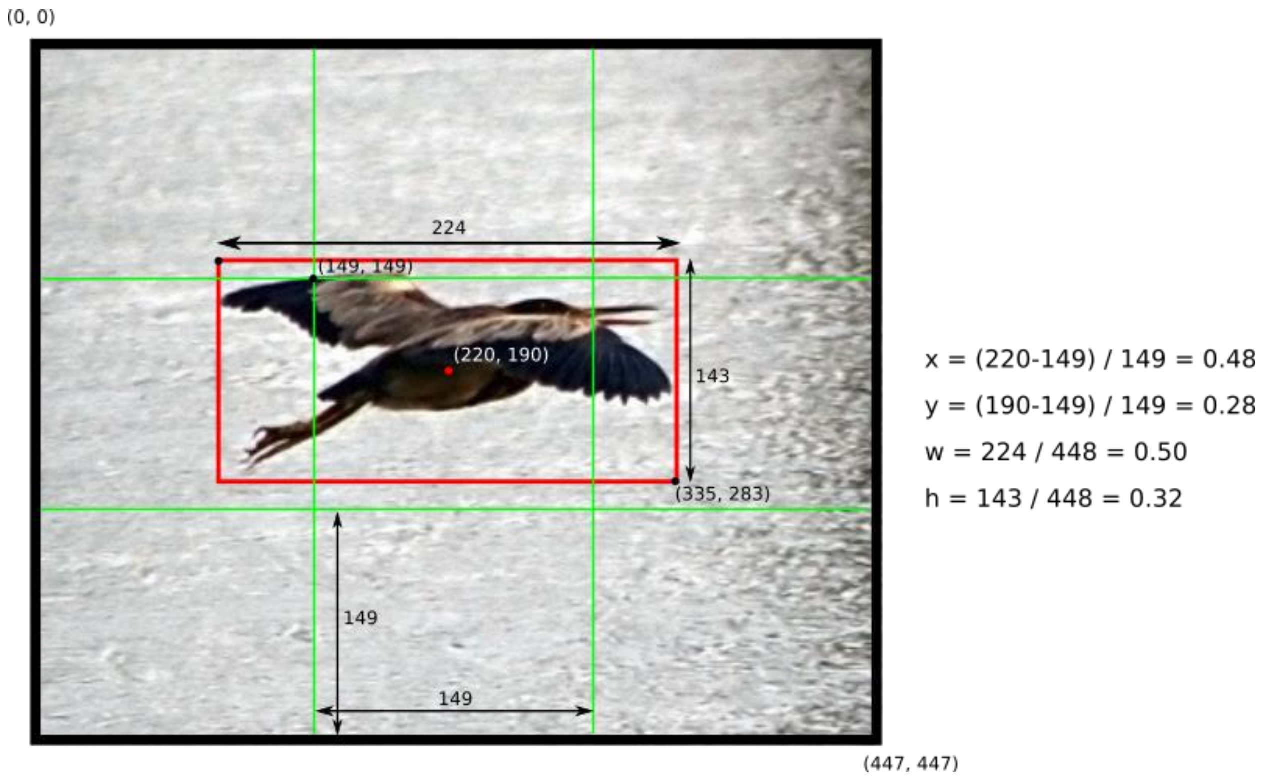


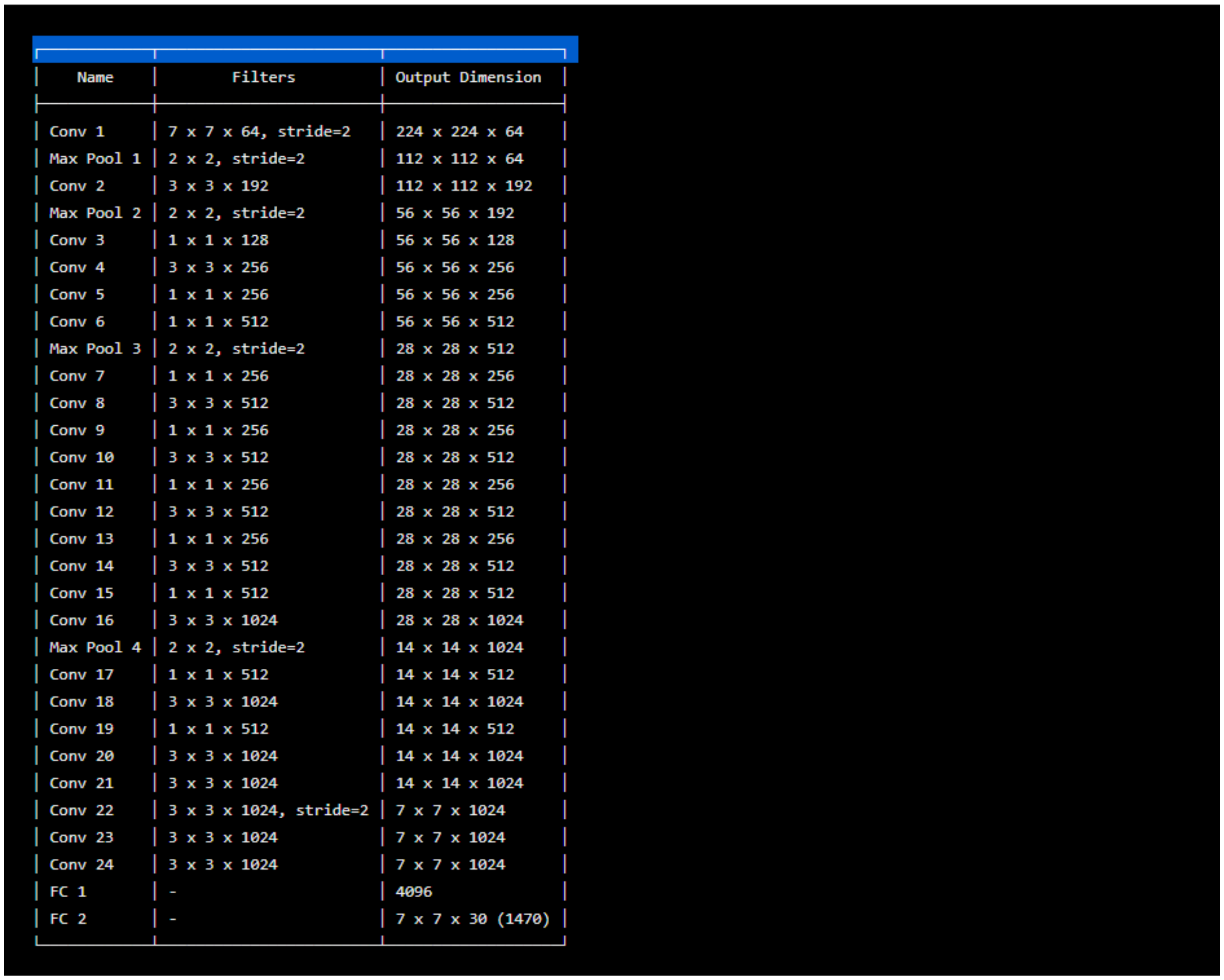
Figure 23: Prediction Vector

Example of how to calculate box coordinates in a 448x448 image with S=3. Note how the (x,y) coordinates are calculated relative to the center grid cell.

There is still one more component in the bounding box prediction, which is the confidence score. Formally we define confidence as PR(Object) * IOU(pred, truth) . If no object exists in that cell, the confidence score should be zero. Otherwise we want the confidence score to equal the intersection over union (IOU) between the predicted box and the ground truth.

The Network:

Once you understand how the predictions are encoded, the rest is easy. The network structure looks like a normal CNN, with convolutional and max pooling layers, followed by 2 fully connected layers in the end:



```
┌────────────┬─────────────────────────┬─────────────────────┐
│    Name    │         Filters         │ Output Dimension    │
├────────────┼─────────────────────────┼─────────────────────┤
│ Conv 1     │ 7 x 7 x 64, stride=2    │ 224 x 224 x 64      │
│ Max Pool 1 │ 2 x 2, stride=2         │ 112 x 112 x 64      │
│ Conv 2     │ 3 x 3 x 192             │ 112 x 112 x 192     │
│ Max Pool 2 │ 2 x 2, stride=2         │ 56 x 56 x 192       │
│ Conv 3     │ 1 x 1 x 128             │ 56 x 56 x 128       │
│ Conv 4     │ 3 x 3 x 256             │ 56 x 56 x 256       │
│ Conv 5     │ 1 x 1 x 256             │ 56 x 56 x 256       │
│ Conv 6     │ 1 x 1 x 512             │ 56 x 56 x 512       │
│ Max Pool 3 │ 2 x 2, stride=2         │ 28 x 28 x 512       │
│ Conv 7     │ 1 x 1 x 256             │ 28 x 28 x 256       │
│ Conv 8     │ 3 x 3 x 512             │ 28 x 28 x 512       │
│ Conv 9     │ 1 x 1 x 256             │ 28 x 28 x 256       │
│ Conv 10    │ 3 x 3 x 512             │ 28 x 28 x 512       │
│ Conv 11    │ 1 x 1 x 256             │ 28 x 28 x 256       │
│ Conv 12    │ 3 x 3 x 512             │ 28 x 28 x 512       │
│ Conv 13    │ 1 x 1 x 256             │ 28 x 28 x 256       │
│ Conv 14    │ 3 x 3 x 512             │ 28 x 28 x 512       │
│ Conv 15    │ 1 x 1 x 512             │ 28 x 28 x 512       │
│ Conv 16    │ 3 x 3 x 1024            │ 28 x 28 x 1024      │
│ Max Pool 4 │ 2 x 2, stride=2         │ 14 x 14 x 1024      │
│ Conv 17    │ 1 x 1 x 512             │ 14 x 14 x 512       │
│ Conv 18    │ 3 x 3 x 1024            │ 14 x 14 x 1024      │
│ Conv 19    │ 1 x 1 x 512             │ 14 x 14 x 512       │
│ Conv 20    │ 3 x 3 x 1024            │ 14 x 14 x 1024      │
│ Conv 21    │ 3 x 3 x 1024            │ 14 x 14 x 1024      │
│ Conv 22    │ 3 x 3 x 1024, stride=2  │ 7 x 7 x 1024        │
│ Conv 23    │ 3 x 3 x 1024            │ 7 x 7 x 1024        │
│ Conv 24    │ 3 x 3 x 1024            │ 7 x 7 x 1024        │
│ FC 1       │ -                       │ 4096                │
│ FC 2       │ -                       │ 7 x 7 x 30 (1470)   │
└────────────┴─────────────────────────┴─────────────────────┘
```

Figure 24: CNN network

Some comments about the architecture:

Note that the architecture was crafted for use in the Pascal VOC dataset, where the authors used S=7, B=2 and C=20. This explains why the final feature maps are 7x7, and also explains the size of the output (7x7x(2*5+20)). Use of this network with a different grid size or different number of classes might require tuning of the layer dimensions.

The authors mention that there is a fast version of YOLO, with fewer convolutional layers. The table above, however, display the full version.

The sequences of 1x1 reduction layers and 3x3 convolutional layers were inspired by the GoogLeNet (Inception) model

The final layer uses a linear activation function. All other layers use a leaky RELU (Φ(x) = x, if x>0; 0.1x otherwise)

If you are not familiar with convolutional networks, take a look at this great introduction

The Training:

The authors describe the training in the following way

First, pretrain the first 20 convolutional layers using the ImageNet 1000-class competition dataset, using a input size of 224x224

Then, increase the input resolution to 448x448

Train the full network for about 135 epochs using a batch size of 64, momentum of 0.9 and decay of 0.0005

Learning rate schedule: for the first epochs, the learning rate was slowly raised from 0.001 to 0.01. Train for about 75 epochs and then start decreasing it.

Use data augmentation with random scaling and translations, and randomly adjusting exposure and saturation.

Chapter 4: Transfer Learning

4.1 Transfer Learning:

Transfer learning is a research problem in machine learning (ML) that focuses on storing knowledge gained while solving one problem and applying it to a different but related problem. For example, knowledge gained while learning to recognize cars could apply when trying to recognize trucks. To deal with the lack of data, we make use of a technique called transfer-learning. Transfer learning is a method that allows us to use knowledge gained from other tasks in order tackle new but similar problems quickly and effectively.

In deep learning, transfer learning involves taking a pre-trained neural network and adapting the neural network to a new, different data set. Instead of training a neural network from scratch we can take a trained CNN model that can extract features with some level of accuracy and pass it on to a new deep learning model. Transfer learning involves several steps before it can successfully be used. First, we have to choose an architecture that is well trained on large datasets. Then we need do remove some part of the trained network in most cases the fully connected layer and then combine our own layer with it. Finally, the model can be used for training by freezing the parameters of the trained network i.e. we will only train the parameters of the modified layers that were added with the trained layers. This will reduce training time significantly and also improve accuracy.

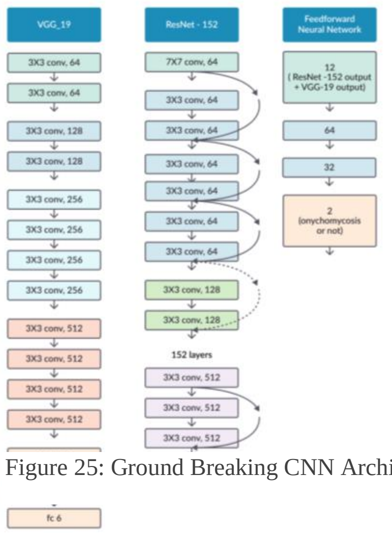


Figure 25: Ground Breaking CNN Architectures

There are quite a few ground-breaking CNN architectures that were trained under millions of data such as googles Inception architecture [22], ResNet, VGG, DenseNet etc. These models were trained on ImageNet dataset which contains more than 80 thousand images. So, using these architectures as a pre-trained network can provide better training result on small dataset.

4.2 Applying Transfer Learning:

Transfer learning has three main uses that professionals use for training. They are:

1. Feature extraction (train only the top-level of the network, the rest of the network remains fixed)
2. Finetuning (train the entire network end-to-end, start with pre-trained weights)
3. Training from scratch (train the entire network end-to-end, start from random weights)

As we mentioned earlier that transfer learning involves taking a pre-trained network and adapting the neural network to a new, different data set. The approach for using this technique is different for different use cases. Such as the size of the new data set and the similarity between the new data set and the original data set on which the network was previously trained on. There are four main cases:

1. new data set is small, new data is similar to original training data
2. new data set is small, new data is different from original training data
3. new data set is large, new data is similar to original training data
4. new data set is large, new data is different from original training data

Now what do we understand by small and large dataset is an important thing to get concerned with. A large data set might have over million images. On the other hand, a small data set could have a few thousand images. The dividing line between a large data set and small data set is somewhat subjective. Overfitting is a concern when using transfer learning with a small data set.

Again, if we consider the similarity between the new and the old dataset, we can say images of dogs and images of wolves can be considered similar. But images of set of flower images and dog images are totally different. Each of the above mentioned four transfer learning cases has its own approach. Let's look them one by one.

Case 1: Small Data Set, Similar Data:

If the new data set is small and similar to the original training data:

- slice off the end of the neural network
- add a new fully connected layer that matches the number of classes in the new data set
- randomize the weights of the new fully connected layer; freeze all the weights from the pre-trained network
- train the network to update the weights of the new fully connected layer

To avoid overfitting on the small data set, the weights of the original network will be held constant rather than re-training the weights.

Since the data sets are similar, images from each data set will have similar higher-level features. Therefore, most or all of the pre-trained neural network layers already contain relevant information about the new data set and should be kept.

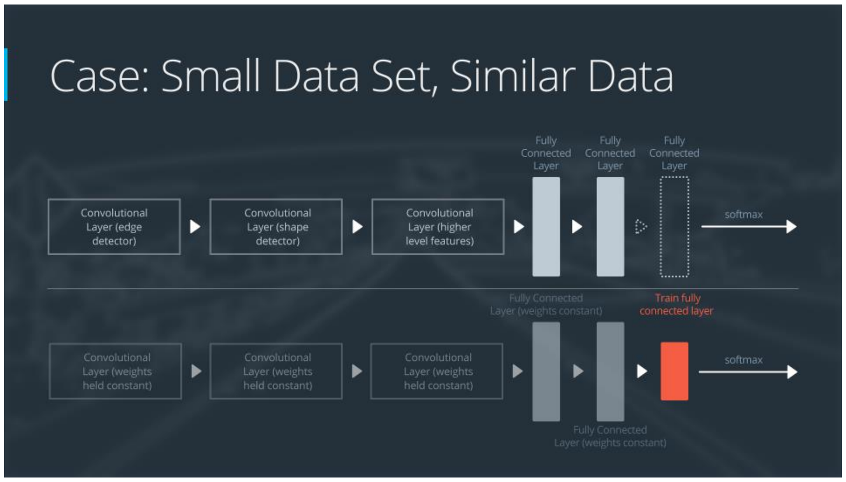

Case: Small Data Set, Similar Data
Fully Connected Layer
Fully Connected Layer
Fully Connected Layer
Convolutional Layer (edge detector)
Convolutional Layer (shape detector)
Convolutional Layer (higher level features)
softmax
Fully Connected Layer (weights constant)
Train fully connected layer
Convolutional Layer (weights held constant)
Convolutional Layer (weights held constant)
Convolutional Layer (weights held constant)
softmax
Fully Connected Layer (weights constant)

Figure 26: Feature extraction with CNN

Case 2: Small Data Set, Different Data:

If the new data set is small and different from the original training data:

- slice off most of the pre-trained layers near the beginning of the network
- add to the remaining pre-trained layers a new fully connected layer that matches the number of classes in the new data set
- randomize the weights of the new fully connected layer; freeze all the weights from the pre-trained network
- train the network to update the weights of the new fully connected layer

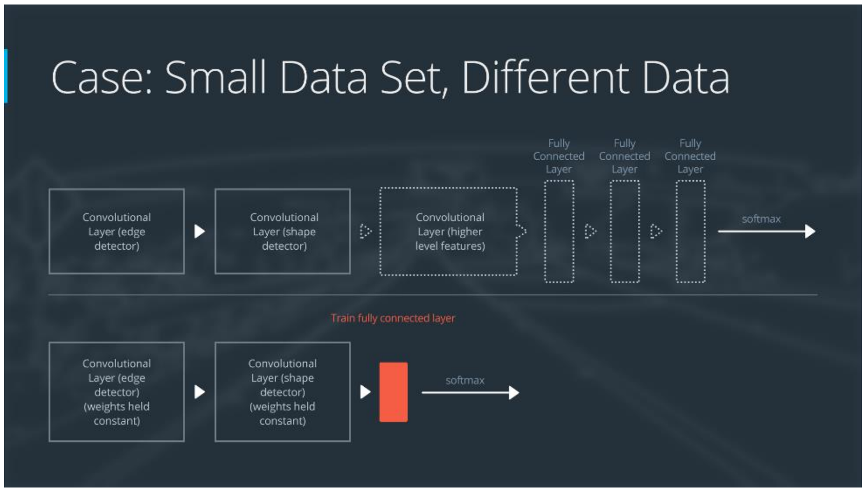

Case: Small Data Set, Different Data
Fully Connected Layer
Fully Connected Layer
Fully Connected Layer
Convolutional Layer (edge detector)
Convolutional Layer (shape detector)
Convolutional Layer (higher level features)
softmax
Train fully connected layer
Convolutional Layer (edge detector) (weights held constant)
Convolutional Layer (shape detector) (weights held constant)
softmax

Figure 27: Small Data Set

Because the data set is small, overfitting is still a concern. To combat overfitting, the weights of the original neural network will be held constant, like in the first case. But the original training set and the new data set do not share higher level features. In this case, the new network will only use the layers containing lower level features.

Case 3: Large Data Set, Similar Data:

If the new data set is large and similar to the original training data:

- remove the last fully connected layer and replace with a layer matching the number of classes in the new data set
- randomly initialize the weights in the new fully connected layer
- initialize the rest of the weights using the pre-trained weights
- re-train the entire neural network

Overfitting is not as much of a concern when training on a large data set; therefore, you can re-train all of the weights.

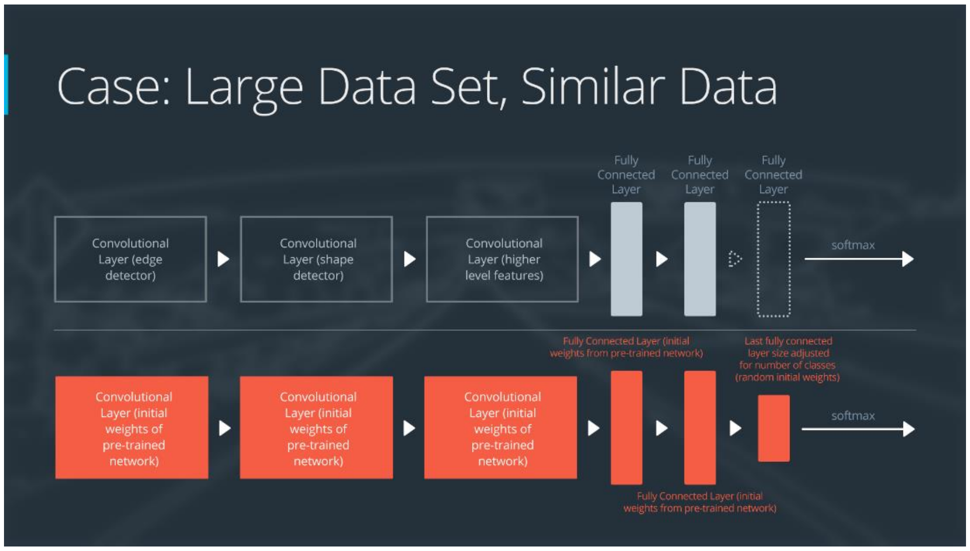

Case: Large Data Set, Similar Data
Convolutional Layer (edge detector)
Convolutional Layer (shape detector)
Convolutional Layer (higher level features)
Fully Connected Layer
Fully Connected Layer
Fully Connected Layer
softmax
Fully Connected Layer (initial weights from pre-trained network)
Last fully connected layer size adjusted for number of classes (random initial weights)
Convolutional Layer (initial weights of pre-trained network)
Convolutional Layer (initial weights of pre-trained network)
Convolutional Layer (initial weights of pre-trained network)
softmax
Fully Connected Layer (initial weights from pre-trained network)

Figure 28: Large Data Set, Similar Data

Because the original training set and the new data set share higher level features, the entire neural network is used as well.

Case 4: Large Data Set, Different Data:

If the new data set is large and different from the original training data:

- remove the last fully connected layer and replace with a layer matching the number of classes in the new data set
- retrain the network from scratch with randomly initialized weights
- alternatively, you could just use the same strategy as the "large and similar" data case

Even though the data set is different from the training data, initializing the weights from the pre-trained network might make training faster. So this case is exactly the same as the case with a large, similar data set.

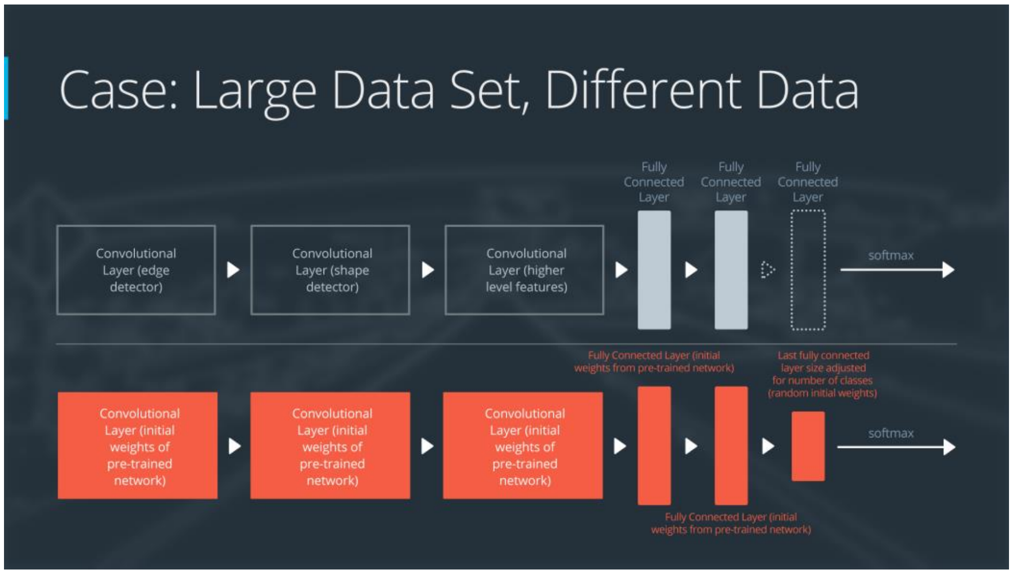

Case: Large Data Set, Different Data
Fully Connected Layer
Fully Connected Layer
Fully Connected Layer
Convolutional Layer (edge detector)
Convolutional Layer (shape detector)
Convolutional Layer (higher level features)
softmax
Fully Connected Layer (initial weights from pre-trained network)
Last fully connected layer size adjusted for number of classes (random initial weights)
Convolutional Layer (initial weights of pre-trained network)
Convolutional Layer (initial weights of pre-trained network)
Convolutional Layer (initial weights of pre-trained network)
softmax
Fully Connected Layer (initial weights from pre-trained network)

Figure 29: Large Data Set, Difference Data

If using the pre-trained network as a starting point does not produce a successful model, another option is to randomly initialize the convolutional neural network weights and train the network from scratch.

4.3 Pre-Trained CNN Architectures:

There are quite a few CNN Models that were trained over thousands of data and are able to extract useful features from images. These architectures are often used as pre-trained models in transfer learning. Let us go over few of these models.

AlexNet: It is a CNN architecture consists of eight layers: five convolutional layers and three fully-connected layers. AlexNet puts the network on two GPUs, which allows for building a larger network. AlexNet allows for multi-GPU training by putting half of the model’s neurons on one GPU and the other half on another GPU. Not only does this mean that a bigger model can be trained, but it also cuts down on the training time. Although most of the calculations are done in parallel, the GPUs communicate with each other in certain layers. The original research paper on AlexNet said that parallelizing the network decreased the classification error rate by 1.7% when compared to a neural network that used half as many neurons on one GPU.

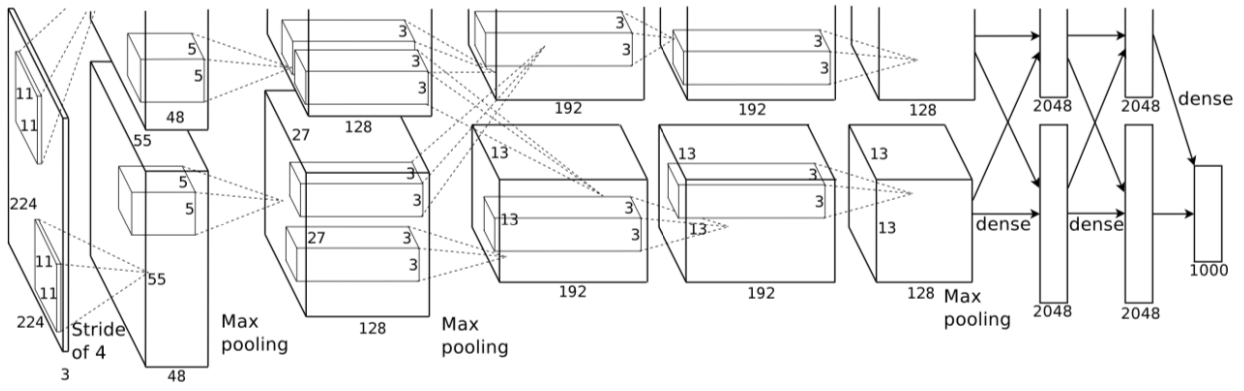


Figure 30: AlexNet

VGG: It came from the Visual Geometry Group from Oxford University [2]. It has a simple and

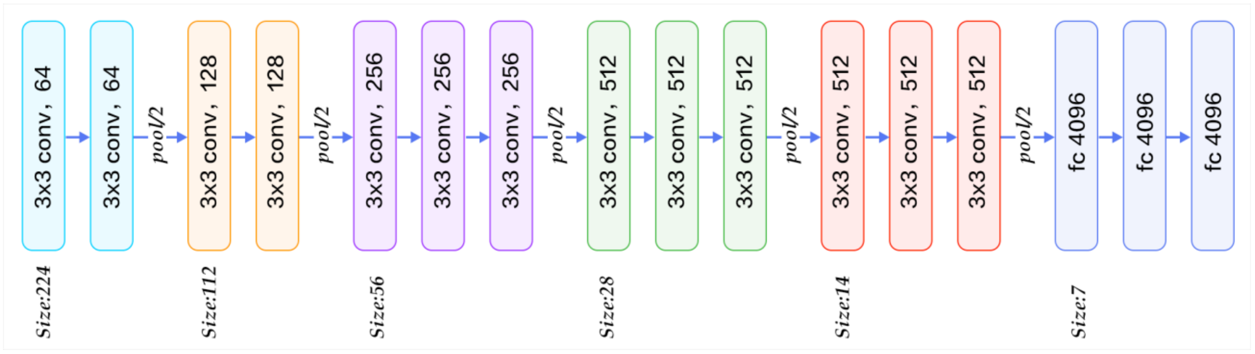


Figure 31: VGG

elegant architecture. The architecture is just a long sequence of 3 by 3 convolution layers broken up by 2 by 2 pooling layers and finished by some layers of fully connected layers. The strength of this network is its flexibility.

ResNet: ResNet was published in 2015 by Microsoft researcher that won the 2015 competition on ImageNet Dataset. It has a massive 152 layers. Its kind of like VGG where same structure is repeated again and again. The main idea was to add connections to the neural network that can skip layers that allows this long deep neural network to train properly. With this it can eliminate the vanishing gradient problem.

GoogLeNet: In 2014 google published its own network that proved to perform better than VGG in imagenet data. It is based on inception models. It creates a situation where the total number of parameters is very small. In an inception module instead of having a single convolution it includes composition of convolution of different filter size and average pooling and the output is the concatenation of all these compositions. GoogLeNet has 22 layers in its architecture.

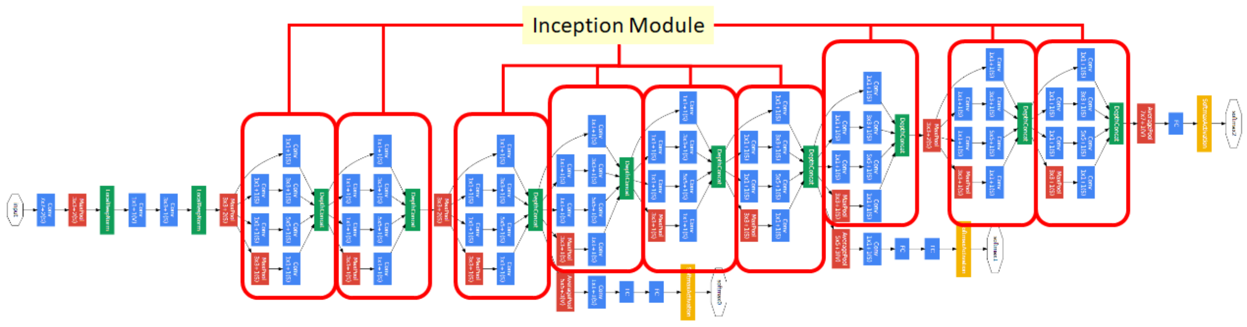


Figure 32: Inception

Chapter 5: Computer Vision

Computer Vision is a broad field of Computer Science where we work with algorithms that allows the computer to see the world the way we see them and teach it to understand color, depth, shapes and meaning from images or videos and interpret them. We want the computer to perceive and extract information from visual image data with the help of such algorithms. For example, edge detection, object detection, scene understanding etc. are some most common problems that are solved by computer vision algorithms. It allows the computer to understand what is in the image, who is in there and what is happening. Some applications of computer vision are Surveillance, building 3d models from medical images, motion capturing etc. It is also used in Optical Character Recognition (OCR) which is a technology used to convert scanned documents to text e.g. license plate readers [23], hand writing understanding. Another cool application of computer vision is facial recognition from images [24]. Almost all digital cameras that comes now a days have built in facial recognition tool. Even some cameras are able to understand smiling or blinking from the faces. Now with the advance of computer vision now face recognition is also used to unlock phones and securing one's privacy. Robotics and self-driving car widely use computer vision for object recognition, lane detection, pedestrian detection etc.

Figure 33: Smile shutter in Sony digital camera

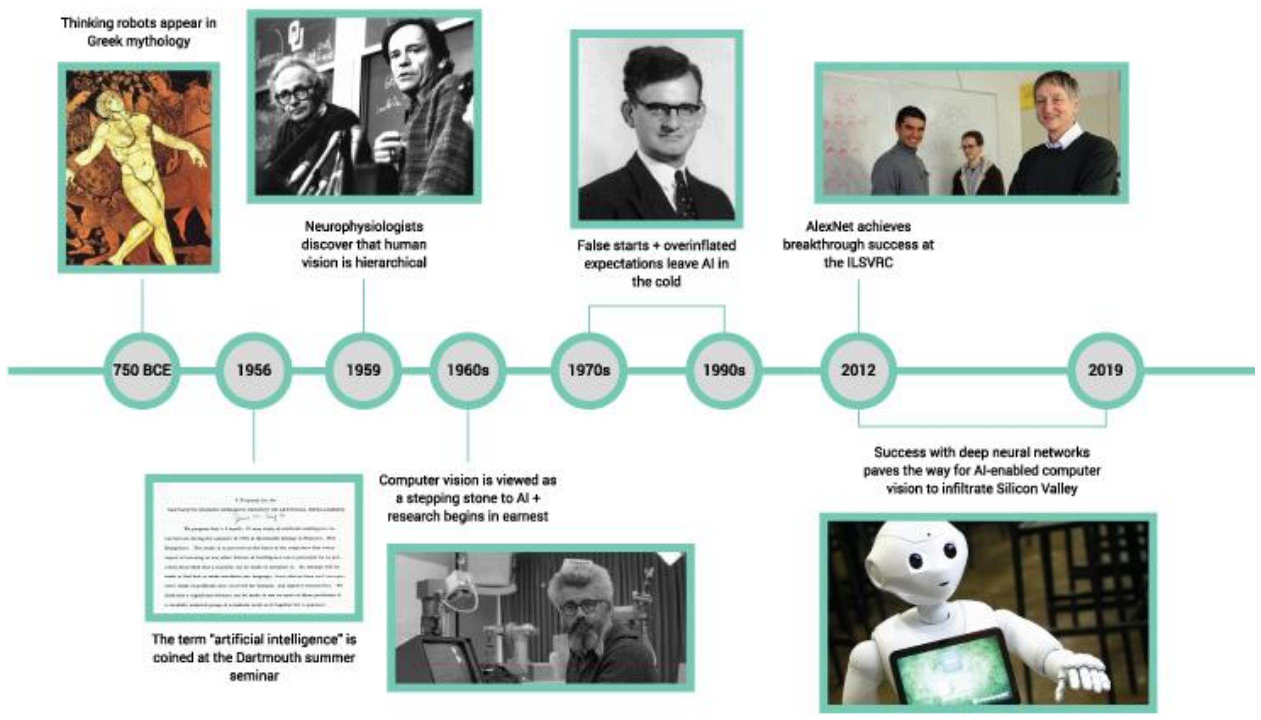
With the advancement of deep learning and the discovery of convolutional neural network computer vision has reached its new landmark. Now with powerful GPUs and millions of data it is now possible to combine deep learning approaches with computer vision to extract features from images and build highly accurate models. Edge detection has now become much easier and accurate. Some application includes image captioning where machine can generate caption by watching an image [25]. Another application is Vision based interaction that are highly used in gaming.

5.1 Background:

Computer vision is an interdisciplinary logical field that manages how Computers can increase significant level comprehension from advanced pictures or recordings. From the viewpoint of designing, it tries to comprehend and mechanize assignments that the human visual framework can. In the late 1960, computer vision began at universities which were pioneering artificial

intelligence. It was meant to mimic the human visual system, as a stepping stone to endowing robots with intelligent behavior. In 1966, it was believed that this could be achieved through a summer project, by attaching a camera to a computer

Figure 34: Smile shutter in Sony digital camera


Thinking robots appear in Greek mythology
Neurophysiologists discover that human vision is hierarchical
False starts + overinflated expectations leave AI in the cold
AlexNet achieves breakthrough success at the ILSVRC
750 BCE
1956
1959
1960s
1970s
1990s
2012
2019
The term "artificial intelligence" is coined at the Dartmouth summer seminar
Computer vision is viewed as a stepping stone to AI + research begins in earnest
Success with deep neural networks paves the way for AI-enabled computer vision to infiltrate Silicon Valley

and having it "describe what it saw". This kind of technology is the precursor to artificially intelligent image recognition. Before, any kind of image analysis had to be done manually, from x-rays to MRIs to hi-res space photography.

Just like animals, computers "see" the world differently from us humans: basically, they count the number of pixels, try to discern borders between objects by measuring shades of color, and estimate spatial relations between objects.

p into, there's been a rise in open-source projects such as ImageNet. ImageNet's mission is to create a large-scale image database that researchers can tap into in order to train and manufacture their algorithms.

The challenge is that in order for computers to index and catalogue these huge sets of data, they initially need to have some human input in terms tagging and classifying their 'training images'. Deep learning algorithms then use this information to create benchmarks to compare future images with, but need to be fed large quantities of training images, as many as 10s of millions

5.2 Application of Computer Vision:

Computer Vision is the art and science of perceiving information of the world around us through images. For example, in self driving cars computer vision can help the car understand about lane detection, pedestrian detection and other elements in the environment to navigate safely. Computer Vision has many use cases. Let us discuss some of its application.

Fixing Distortion of images:

Our real world is in 3D. But when we capture images with cameras, we get 2D images as output. This can cause distortion of objects in the image. Image distortion occurs when a camera looks at 3D objects in the real world and transforms them into a 2D image; this transformation isn't perfect. Distortion actually changes what the shape and size of these 3D objects appear to be. So, the first step in analyzing camera

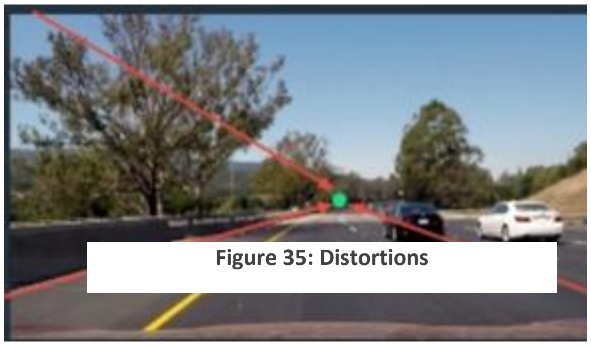


Figure 35: Distortions

images, is to undo this distortion so that we can get correct and useful information out of them. There are mainly two types of distortion of image:

1. Radial Distortion: Real cameras use curved lenses to form an image, and light rays often bend a little too much or too little at the edges of these lenses. This creates an effect that distorts the edges of images, so that lines or objects appear more or less curved than they actually are. This is called radial distortion, and it's the most common type of distortion.

2. Tangential Distortion: This occurs when a camera's lens is not aligned perfectly parallel to the imaging plane, where the camera film or sensor is. This makes an image look tilted so that some objects appear farther away or closer than they actually are.

Perspective transform:

A perspective transform maps the points in a given image to different, desired, image points with a new perspective. This is often needed to make sense of road lane by self-driving cars. Because objects appear smaller the farther away, they are from a viewpoint. Parallel lines seem to converge to a point. So, a perspective transform can be used for all kinds of different viewpoints and extract proper information.

Let's see how a simple text might look like after applying perspective transform on that image:

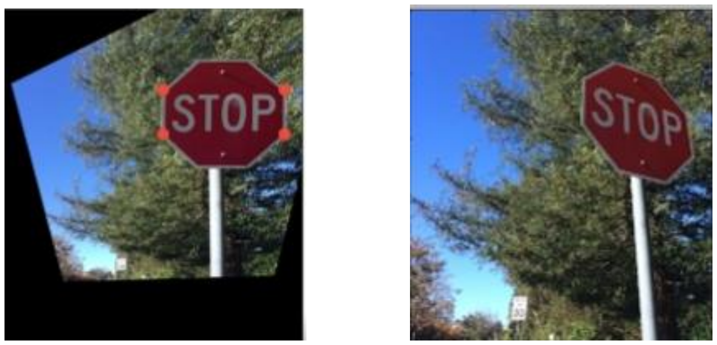


Figure 36: Perspective transform

There are many practical applications of computer vision:

Autonomous Vehicles — This is one of the most important applications of Computer vision where the self-driving cars need to gather information about their surroundings to decide how to behave.

Facial Recognition — This is also a very important application of computer vision where electronics use facial recognition technology to basically validate the identity of the user.

Image Search and Object Recognition — Now we could search objects in an image using image search. A very good example is google lens where we could search a particular object within the image by clicking the photo of the image and the computer vision algorithm will search through the catalogue of images and extract information out of the image.

Robotics — Most robotic machines, often in manufacturing, need to see their surroundings to perform the task at hand. In manufacturing machines may be used to inspect assembly tolerances by “looking at” them.

Chapter 6: Methodology

This section contains the methodology of our implemented system.

6.1 Data Preparation

We will now go through some data preprocessing techniques that are commonly used with image preprocessing. Preprocessing takes a lot of time in deep learning projects. It is difficult to make decision how to prepare image data for training. Different images have different shapes and sizes. There are lots of variations in images. For all of these variations we need to perform pre-processing on image data. First step of image pre-processing is to make the images of same size and aspect ratio. Most of the neural models assume that the shapes of input images are square. So we crop to select a square part of the image.

Image scaling to each image is important. There are varieties of image scaling like up-scaling and down-scaling. We use binary function for image scaling. Scaling improves the speed and accuracy of convergence.

Dataset Description:

We have used the Facial Expression Recognition 2013 (FER-2013) dataset for our work [26]. It is a free and open-source dataset that was published in International Conference on Machine Learning (ICML). Pierre-Luc Carrier and Aaron Courville generated the dataset by gathering search results from google of different human emotions based on feelings. Then the dataset was openly shared before ICML 2013 for a Kaggle competition. The dataset consists of 35,887 grayscale images, from which 28,709 labeled for training, 3589 images for validation, and 3,589 images for testing. The dataset contains 48-by-48-pixel grayscale images of faces, where each image is labeled as one of the seven facial emotions.

In Fig. 37. some sample images from our dataset have been shown with all the labeled expression that we have used in our work. The labels of categories indexed from (0 – 6) where 0: Angry (4593 images), 1: Disgust (547 images), 2: Fear (5121 images), 3: Happy (8989 images), 4: Sad (6077 images) , 5: Surprise (4002 images) , 6: Neutral (6198 images) .The dataset contains 48-by-48-pixel grayscale images of faces.

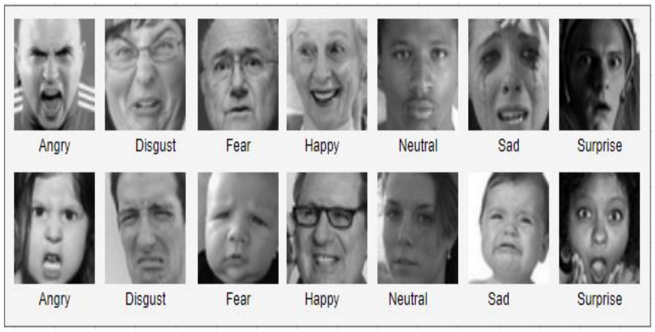


Figure 37: Dataset Images

Data Normalization:

Data normalization is an important and most crucial part in pre-processing. It ensures similar data distribution over every input parameter. It helps with the issue of propagating gradients. Also, it makes convergence faster while training the network. Data normalization is done by subtracting the mean from each pixel and then dividing the result by the standard deviation.

There are three most common techniques for finding the of value normalization:

- (x - x.min()) / (x.max() - x.min())
- 2*(x - x.min()) / (x.max() - x.min())
- (x - x.mean ()) / x.std()

The first one is for converting the value from 0 to 1. The second one is for values from -1 to 1. In the second approach the range center at zero. Third approach is the cleanest normalization. It is the only one that centers the mean at zero. It helps with the disappearing the gradients.

Dimension Reduction:

We can combine three RGB channels into a single gray-scale channel. This helps to reduce the dimensionality. We can speed up computation by reducing dimension of the dataset.

Data Augmentation:

It can be happened that the size of our dataset is not enough for performing classification tasks. We perform data augmentation in such cases. Data segmentation is a common task in pre-processing techniques. Data augmentation significantly boosts the accuracy of the model. Different types of data augmentation can possible. Basic ones are –

- Linear transformations
- Affine transformations

We say a Neural network is poorly trained when it can’t classify or detect the object from any angle or point of view. A poorly classified neural network thinks same images as distinct images. CNN can be invariant to translation, viewpoint, size or illumination. It can happen that our dataset has limited set of conditions. But target application has different types of conditions. Data augmentation helps us to overcome this problem. Data augmentation can also help to increase relevant data in our dataset. We do augmentation before we feed the data to the model.

The first option to perform data augmentation is offline data augmentation. This function is preferred for smaller dataset. After perform offline data augmentation, the size of the dataset increases by a factor equal to the number of transformations we perform.

The second option is online augmentation. This function is preferred for larger dataset.

Image Augmentation

When we design an algorithm to classify images, we have to handle a lot of irrelevant information. We really want to determine If an object in the image is present or not. The size or angle of the object doesn’t matter to us. We want that our algorithm learns invariant representation of image. We don’t want t our model to change its prediction based on the side of the image. This is called scale invariance. Likewise, we don’t want our angle of object to matter. This is called rotation invariance. If we shift the object of an image from left to right or right to left, it is still the same object. This is called translation invariance. Computer only sees array of pixel values as an image. If we want our CNN to be any of these invariances then we can manually add some images to our training network according to invariance. By doing this we have expanded the training set by augmented data. Data Augmentation helps us to avoid overfitting and getting better performance for testing dataset.

6.2 Model Design & Training

For model training, we have tried two approaches. First, we have tried using transfer learning. We have used VGG-16 architecture as our CNN model. It contains about 138 million [27] learning parameters. The architecture is simple and elegant, but the network is very deep. So, training from scratch would have taken a lot of time and required a huge dataset. For this reason, we have chosen to use transfer learning for training. Since VGG is a pre-trained architecture so, the CNN layers work very well as a feature extractor. So, we decided to freeze the learning parameters of the CNN layers and used them as the feature extractor for the images. Then we modified the fully connected layers of the architecture and added the final output layer with seven nodes. But this approach failed to give a good result. The validation loss was getting stuck. So, we had to reject this approach.

Next, we build a new CNN architecture from scratch to train the model. The architecture expects an input of size 48 by 48 greyscale images. So, it was an advantage as our dataset also had images of the same size, unlike VGG which expects to have input with image size of 224 by 224 pixels and three different color channels.

In Fig. 38, We have shown our CNN model architecture with layers and filter size. We have used filters of size 3 for all the convolutional layers. We have used stride size of 2 and applied same padding over the layers. The same padding allows to add extra padding to the image so that the whole image gets covered. We added batch normalization and pooling layers and used a dropout of 0.5 for most of the layers.

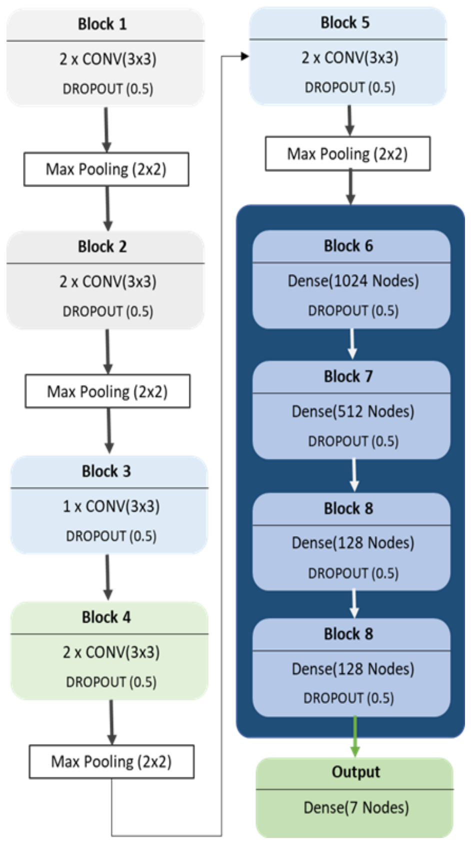


Figure 38: Model Design

The total number of learning parameters in our model is over 13 million. To train our model, we used batch size of 32 and trained the model for about 40 epochs. We have used the following function for calculating errors.

$$CCE = -\frac{1}{N}\sum_{i=0}^{N}\sum_{j=0}^{J} y_j \cdot \log(\hat{y}) + (1 - y_j) \cdot \log(1 - \hat{y}_j)$$

Equation 1

We have used equation (1), which is the categorical cross-entropy loss function, to calculating error. We have also used Adam optimizer to update weights with an initial learning rate of 0.003.

6.3 Generating Review

Our model can predict the facial expression from an image of a single face. But for generating reviews from an image with multiple faces, we have used computer vision. DNN Face Detector from OpenCV's Caffe model [28] is one of the most popular methods used to detect faces with high accuracy. So, we have used this DNN classifier to extract faces from images. Then for each face, we have used our model to generate expression. As we have seven facial expressions, we have assigned specific weights to each of them. The weights are shown in Table 1.

Table 1 shows the corresponding scores for each class of emotions. The weights here are being used to generate the overall score after the model predicts. These scores are assigned based on a general survey of student's behavior. In the survey, we have compared student's results and their response during the class. Students with excellent results tend to be very excited during lectures as they enjoy what they learn. And generally, attentive students are found to have neutral expressions during their class. So, happy and neutral expressions are assigned high scores.

On the other hand, some students remain unfocused during the class and focus on mobile phones or other topics. So, they often get surprised or remain in fear when asked questions, and many students who give negative reviews are found to have sad or dull expression during their class. Based on these general student behaviors, we have assigned the scores.

TABLE: EMOTION SCORE

Emotion	Score
Happy	1
Neutral	0.7
Surprise	0.6
Fear	0.5
Sad	0.4
Angry	0.25
Disgust	0

To generate the review, we have generated result based on an average calculation. We have calculated the sum from the scores generated from all the faces. Then we have taken the average of the scores. We have classified the scores into three labels. If the average review is over 0.6, we say it is positive, and if the score is below 0.4, we say the review is negative. If the review is above 0.4, and below 0.6, we have labeled the review as neutral. But during the class, a facial expression of an individual may change every minute. So, taking review from just a single image for each individual may cause a biased result. So, we took more than one image of the same scene with a specific time interval and then extracted the same individual faces for review. We have used the following approach.

Algorithm 1. Generate Average Review Pseudocode

```
1: review_count ← 0
2: repeat for every 5 munities up to 1 hour:
3:         temp_review ← 0
4:         Capture images and extract faces
5:         temp_review ← Score(extracted_face)
6:         review_count ← review_count + temp_review
7: end loop
8: final_review = Average(review_count)
```

In Algorithm 1, we have designed and approach that takes several images for every time frame at a specific gap during the class. Then it extracts faces from the images and generates a review score for each of the detected faces. For face extraction, we have used Caffe model's DNN face detector that comes with the OpenCV library. Then the algorithm repeats the same step several times and calculates the average score. This is how it generates the final score from the average of the calculated scores.

Chapter 7: Result analysis

7.1 Result of training with VGG Model:

As mentioned before that we first tried to train our data using VGG architecture, which performed quite well but not good enough. The input size of VGG is 224 X 224 X 3. But our dataset was provided in size 48X48 pixels with all grayscale images. So, the training loss initially was very high. As we trained for few 100 times over the input images, we noticed the network training loss was diminishing. But often it had few spikes in the loss. We weren't quite sure why this was happening. Below is the figure of the training loss of our initial architecture:

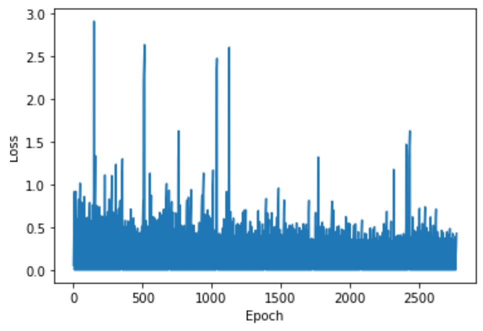


Figure 39: Overall Result

7.2 Result of training with our model:

In the 2nd try we designed another model to train our data. The model architecture is designed to look like a smaller version of the VGG16 model. So, we retrained the network from beginning. drastically and the performance of the model was far better than the previous one. We found almost 69% in our test accuracy. Let us visualize the result bellow:

.

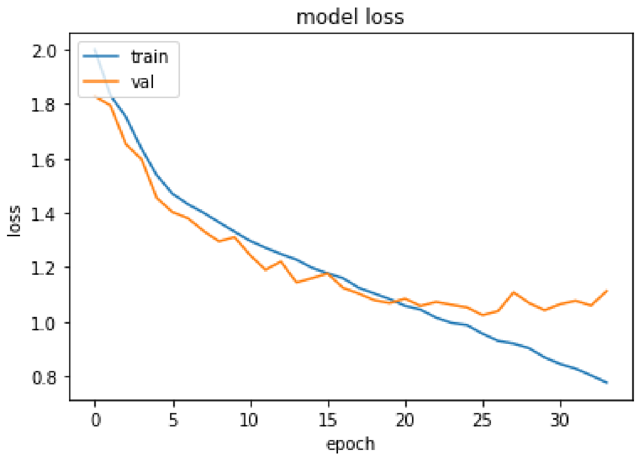


Fig. 40. Training loss Graph

Fig. 5 shows the training loss and the validation loss graph. For preventing overfitting, we used early stopping by monitoring the validation loss.

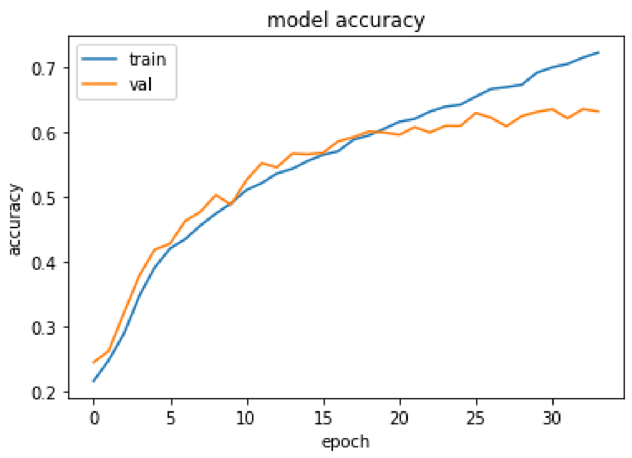
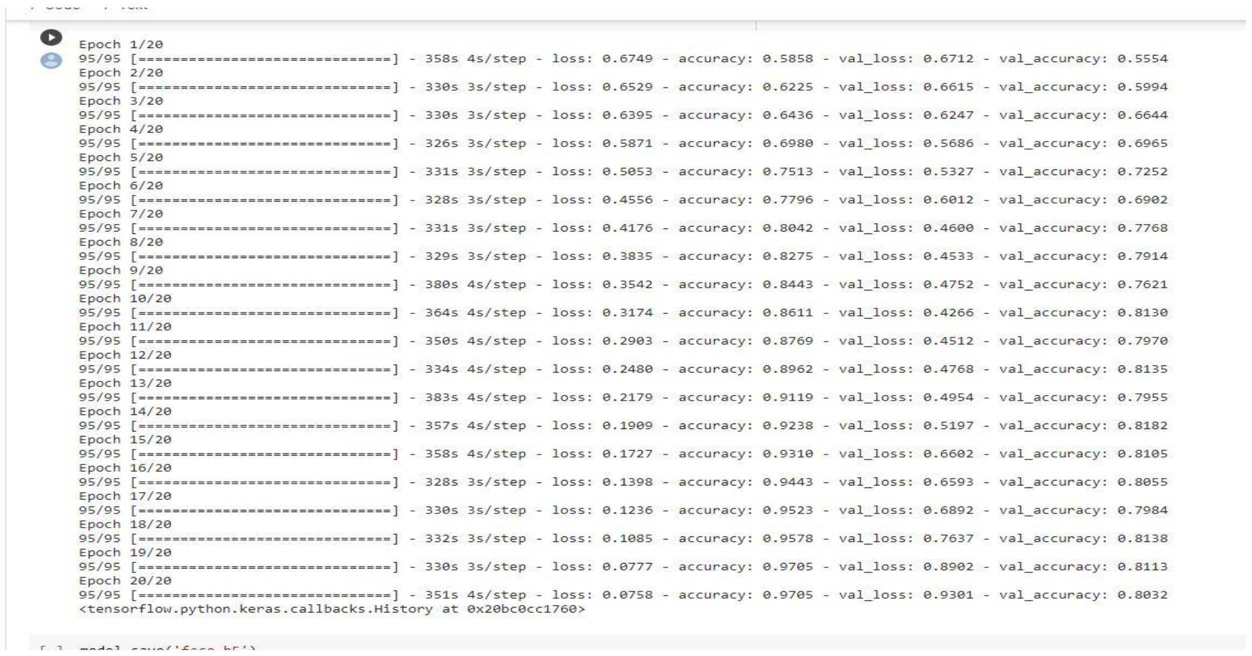


Fig. 41. Model Accuracy track

Fig. 18 shows the model accuracy graph generated while training the model. We have shown the training and validation accuracy in the graph.

Next to improve the accuracy we modified the data output classes. We distributed the six classes into two classes to design a new binary classification model. Then trained the new model.



```
Epoch 1/20
95/95 [==============================] - 358s 4s/step - loss: 0.6749 - accuracy: 0.5858 - val_loss: 0.6712 - val_accuracy: 0.5554
Epoch 2/20
95/95 [==============================] - 330s 3s/step - loss: 0.6529 - accuracy: 0.6225 - val_loss: 0.6615 - val_accuracy: 0.5994
Epoch 3/20
95/95 [==============================] - 330s 3s/step - loss: 0.6395 - accuracy: 0.6436 - val_loss: 0.6247 - val_accuracy: 0.6644
Epoch 4/20
95/95 [==============================] - 326s 3s/step - loss: 0.5871 - accuracy: 0.6980 - val_loss: 0.5686 - val_accuracy: 0.6965
Epoch 5/20
95/95 [==============================] - 331s 3s/step - loss: 0.5053 - accuracy: 0.7513 - val_loss: 0.5327 - val_accuracy: 0.7252
Epoch 6/20
95/95 [==============================] - 328s 3s/step - loss: 0.4556 - accuracy: 0.7796 - val_loss: 0.6012 - val_accuracy: 0.6902
Epoch 7/20
95/95 [==============================] - 331s 3s/step - loss: 0.4176 - accuracy: 0.8042 - val_loss: 0.4600 - val_accuracy: 0.7768
Epoch 8/20
95/95 [==============================] - 329s 3s/step - loss: 0.3835 - accuracy: 0.8275 - val_loss: 0.4533 - val_accuracy: 0.7914
Epoch 9/20
95/95 [==============================] - 380s 4s/step - loss: 0.3542 - accuracy: 0.8443 - val_loss: 0.4752 - val_accuracy: 0.7621
Epoch 10/20
95/95 [==============================] - 364s 4s/step - loss: 0.3174 - accuracy: 0.8611 - val_loss: 0.4266 - val_accuracy: 0.8130
Epoch 11/20
95/95 [==============================] - 350s 4s/step - loss: 0.2903 - accuracy: 0.8769 - val_loss: 0.4512 - val_accuracy: 0.7970
Epoch 12/20
95/95 [==============================] - 334s 4s/step - loss: 0.2480 - accuracy: 0.8962 - val_loss: 0.4768 - val_accuracy: 0.8135
Epoch 13/20
95/95 [==============================] - 383s 4s/step - loss: 0.2179 - accuracy: 0.9119 - val_loss: 0.4954 - val_accuracy: 0.7955
Epoch 14/20
95/95 [==============================] - 357s 4s/step - loss: 0.1909 - accuracy: 0.9238 - val_loss: 0.5197 - val_accuracy: 0.8182
Epoch 15/20
95/95 [==============================] - 358s 4s/step - loss: 0.1727 - accuracy: 0.9310 - val_loss: 0.6602 - val_accuracy: 0.8105
Epoch 16/20
95/95 [==============================] - 328s 3s/step - loss: 0.1398 - accuracy: 0.9443 - val_loss: 0.6593 - val_accuracy: 0.8055
Epoch 17/20
95/95 [==============================] - 330s 3s/step - loss: 0.1236 - accuracy: 0.9523 - val_loss: 0.6892 - val_accuracy: 0.7984
Epoch 18/20
95/95 [==============================] - 332s 3s/step - loss: 0.1085 - accuracy: 0.9578 - val_loss: 0.7637 - val_accuracy: 0.8138
Epoch 19/20
95/95 [==============================] - 330s 3s/step - loss: 0.0777 - accuracy: 0.9705 - val_loss: 0.8902 - val_accuracy: 0.8113
Epoch 20/20
95/95 [==============================] - 351s 4s/step - loss: 0.0758 - accuracy: 0.9705 - val_loss: 0.9301 - val_accuracy: 0.8032
<tensorflow.python.keras.callbacks.History at 0x20bc0cc1760>
```

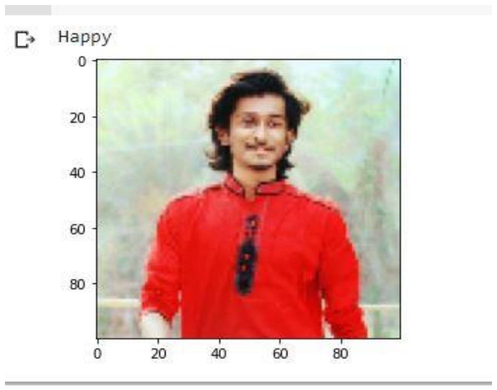
Figure 42: Results with each epoch

We got 55% validation accuracy and 58% train accuracy in our first epoch. We continued training for few more epochs and finally we were able to get 81% validation accuracy and 97% train accuracy from our model. To increase the accuracy, we converted the 6 emotion classes to be in two classes of each giving positive reaction and the other gives negative. The training result on the newly processed data is shown below:

Epochs	Accuracy	Loss Function	Processing Time(s)
1	0.5554	0.6712	358
2	0.5994	0.6615	330
3	0.6644	0.6247	330
4	0.6965	0.5686	326
5	0.7252	0.5327	331
6	0.6902	0.6012	328
7	0.7768	0.4600	331
8	0.7914	0.4533	329
9	0.7621	0.4752	380
10	0.8130	0.4266	364
11	0.7970	0.4512	350
12	0.8135	0.4768	334
13	0.7955	0.4954	383
14	0.8182	0.5197	357
15	0.8105	0.6602	358
16	0.8055	0.6539	328
17	0.7984	0.6892	330
18	0.8138	0.7637	332
19	0.8113	0.8902	330
20	0.8032	0.9301	351

Table 1: Showing accuracy with the help of epoch

The accuracy can be increased by augmentation and increasing epochs. By increasing epochs and adding augmentation we can increase our accuracy a little bit. From final output we can generate an average review of the facial expression. We could also use other features such as Histogram of Oriented Gradients (HOG) for feature extractions. These could help in getting better accuracy and improve the review prediction.



```
Happy
```

At the first when we start project, we got only 53% validation accuracy. This is our final result and we achieved 80% validation accuracy.

The following graph shows the relation between epoch and accuracy:

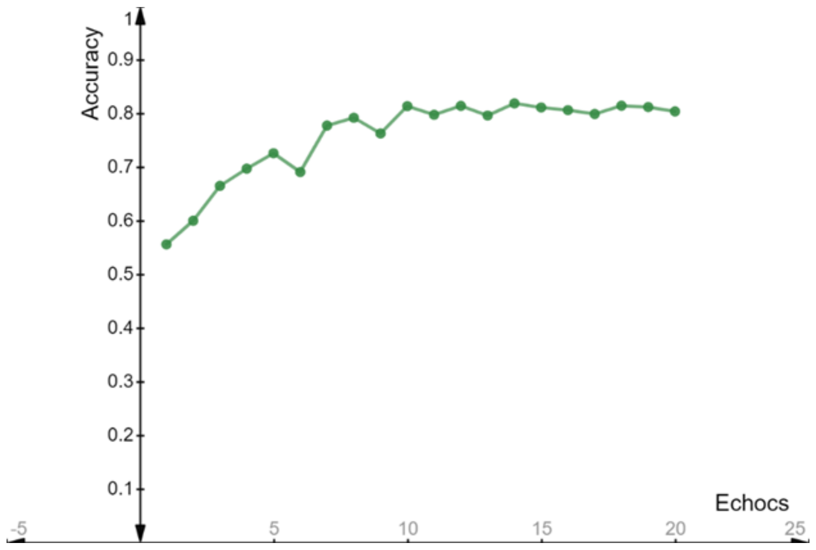


Figure 43: Epochs vs. accuracy graph

This graph shows us, accuracy is increasing in every epoch.

7.3 Confusion Matrix

The confusion matrix shows the ways in which your classification model is confused when it makes predictions. Our model got 8 correct predictions out of 10 test images from the new dataset.

We know, classification accuracy = correct predictions / total predictions * 100

= 8/10 *100 = 80%

error rate = (1 - (correct predictions / total predictions)) * 100

= (1- (8/10)) * 100 = 20%

This graph shows the relation between loss and iteration.

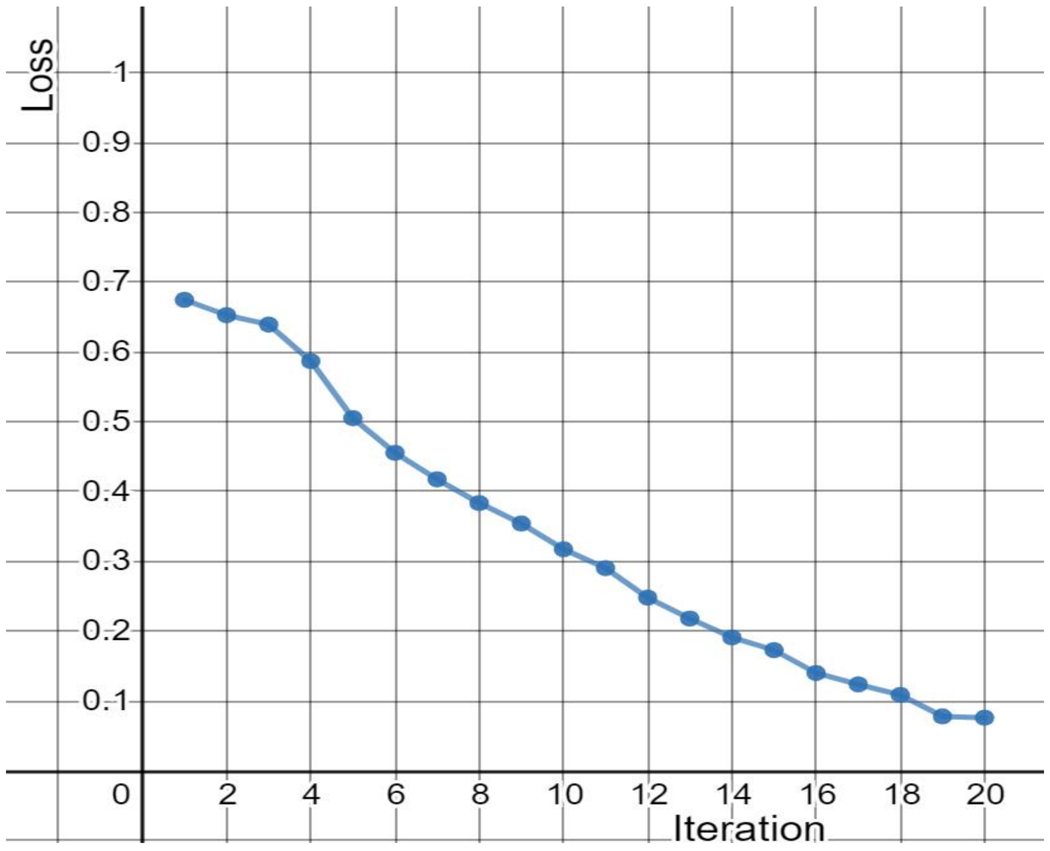


Figure 44: Loss vs. iteration graph

7.4 Conclusions

In our work, we have experimented with two models to predict facial expression from images of the face. We have assigned scores to the expression and designed an algorithm to generate reviews from that score. The main goal of our work was to design a system that can generate reviews. We focused less on tuning the accuracy of the models as there are already pre-trained architectures with high accuracy for predicting facial expressions. The system will help the faculties understand how students are responding during the class and help authorities get an idea about the interactivity of students during class.

7.5 Limitations & Future research

In this work, we focused on designing an algorithm that uses a trained model to extract facial expressions and generate reviews. With a better dataset of larger image size like 224 by 224, it is possible to train the data on pre-trained architectures like inception, ResNet, VGG etc. to improve the accuracy of the model. It is possible to experiment with other methods like Histogram of Oriented Gradients (HOG) for feature extractions. These might result in better accuracy and improve review prediction. This system can be incorporated with IoT devices that will be able to provide real-time reviews of the students during the class. Another approach that can also be experimented is to work with an image as a whole. Instead of working with an image of a single face separately, we could use the whole classroom image which will have faces of multiple students and use that image as the input to a Convolutional Neural Network to directly predict a review from that image. This will open many opportunities to experiment with more complicated architectures and use other methods like single-shot detectors to build better models. We could not collect enough data for testing these options. But with the power of CNN, we believe there are lots of scope for further research on this topic to improve and design better models to improve accuracy and generate reviews.

References:

[1] Y. l. Tian, T. Kanade, and J. F. Cohn, "Evaluation of Gabor-Wavelet-Based Facial Action Unit Recognition in Image Sequences of Increasing Complexity," in Proceedings of the Fifth IEEE International Conference on Automatic Face and Gesture Recognition, pp. 229-234, 2002.

[2] "How does a faculty evaluation system keep things fair? - Interfolio", Interfolio, [Online]. Available: https://www.interfolio.com/resources/blog/how-does-a-faculty-evaluation-system-keep-things-fair/. [Accessed: 19- Dec- 2020].

[3] Tan, Lianzhi & Zhang, Kaipeng & Wang, Kai & Zeng, Xiaoxing & Peng, Xiaojiang & Qiao, Yu. (2017). Group emotion recognition with individual facial emotion CNNs and global image based CNNs. 549-552. 10.1145/3136755.3143008.

[4] Zafar, Sahar & Ali, Fayyaz & Guriro, Subhash & Ali, Irfan & Khan, Asif & Zaidi, Adnan. (2019). Facial Expression Recognition with Histogram of Oriented Gradients using CNN. Indian Journal of Science and Technology. 12. 10.17485/ijst/2019/v12i24/145093.

[5] L. C. De Silva, T. Miyasato and R. Nakatsu, "Facial emotion recognition using multi-modal information," Proceedings of ICICS, 1997 International Conference on Information, Communications and Signal Processing. Theme: Trends in Information Systems Engineering and Wireless Multimedia Communications (Cat., Singapore, 1997, pp. 397-401 vol.1, doi: 10.1109/ICICS.1997.647126.

[6] Mao Xu, Wei Cheng, Qian Zhao, Li Ma and Fang Xu, "Facial expression recognition based on transfer learning from deep convolutional networks," 2015 11th International Conference on Natural Computation (ICNC), Zhangjiajie, 2015, pp. 702-708, doi: 10.1109/ICNC.2015.7378076

[7] Agrawal, Abhinav & Mittal, Namita. (2019). Using CNN for facial expression recognition: a study of the effects of kernel size and number of filters on accuracy. The Visual Computer. 36. 10.1007/s00371-019-01630-9.

[8] H. Jung et al., "Development of deep learning-based facial expression recognition system," 2015 21st Korea-Japan Joint Workshop on Frontiers of Computer Vision (FCV), Mokpo, 2015, pp. 1-4, doi: 10.1109/FCV.2015.7103729.

[9] V. Agarwal, "Face Detection Models: Which to Use and Why?", Medium, [Online]. Available: https://towardsdatascience.com/face-detection-models-which-to-use-and-why-d263e82c302c?gi=1d1c9add7285. [Accessed: 02- Dec- 2020].

[10] "fer2013", Kaggle.com, [Online]. Available: https://www.kaggle.com/deadskull7/fer2013. [Accessed: 06- Jan-2021].

[11] Simonyan, K., & Zisserman, A. (2014). Very Deep Convolutional Networks for Large-Scale Image Recognition*arXiv e-prints*, arXiv:1409.1556.

[12] "OpenCV: Deep Neural Network module", Docs.opencv.org. [Online]. Available: https://docs.opencv.org/4.0.0/d6/d0f/group__dnn.html. [Accessed: 11- Jan- 2021].

[13] Kumar, Alok & Jain, Renu. (2018). Faculty Evaluation System. Procedia Computer Science. 125. 533-541. 10.1016/j.procs.2017.12.069.

[14] A. Samal and P.A. Iyengar, "Automatic Recognition and Analysis of Human Faces and Facial Expressions: A Survey," Pattern Recognition, vol. 25, no. 1, pp. 65-77, 1992.

[15] P. Ekman and W.V. Friesen, Facial Action Coding System (FACS), Consulting Psychologists Press, 1978.

[16] [3] Y. l. Tian, T. Kanade, and J. F. Cohn, "Evaluation of Gabor-Wavelet-Based Facial Action Unit Recognition in Image Sequences of Increasing Complexity," in Proceedings of the Fifth IEEE International Conference on Automatic Face and Gesture Recognition, pp. 229-234, 2002.

[17] J. F. Cohn, A. J. Zlochower, J. J. Lien, and T. Kanade, "Feature-point tracking by optical flow discriminates subtle differences in facial expression," in Proceedings of the 3rd IEEE International Conference on Automatic Face and Gesture Recognition, pp. 396-401, 1998.

[18] M. Shin, M. Kim and D. Kwon, "Baseline CNN structure analysis for facial expression recognition", 2016 25th IEEE International Symposium on Robot and Human Interactive Communication (RO-MAN), 2016.

[19] G. Bradski, "The OpenCV Library," Dr. Dobb's Journal of Software Tools, 2000.

[20] Yu-Lun Liu, Yi-Tung Liao, Yen-Yu Lin, and Yung-Yu Chuang, "Deep video frame interpolation using cyclic frame generation," in Proc. AAAI, 2019, pp. 8794–8802.

[21] P. L. Carrier I. J. Goodfellow, D. Erhan et al., "Challenges in representation learning: A report on three machine learning contests," in International Conference on Neural Information Processing, pp. 117–124, 2013.

[22] Z. Zhang, Z. Yang, Y. Sun, Y. Wu and Y. Xing, "Lenet-5 Convolution Neural Network with Mish Activation Function and Fixed Memory Step Gradient Descent Method," 2019 16th International Computer Conference on Wavelet Active Media Technology and Information Processing, Chengdu, China, 2019, pp. 196-199, doi: 10.1109/ICCWAMTIP47768.2019.9067661.

[23] Kelvin Xu, Jimmy Ba, Ryan Kiros, Kyunghyun Cho, Aaron Courville, Ruslan Salakhutdinov, Richard Zemel, & Yoshua Bengio. (2015). Show, Attend and Tell: Neural Image Caption Generation with Visual Attention.

[24] J. F. Cohn, A. J. Zlochower, J. J. Lien, and T. Kanade, “Feature-point tracking by optical flow discriminates subtle differences in facial expression,” in Proceedings of the 3rd IEEE International Conference on Automatic Face and Gesture Recognition, pp. 396-401, 1998

[25] H. A. Rowley, S. Baluja and T. Kanade, "Neural network-based face detection," in IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 20, no. 1, pp. 23-38, Jan. 1998, doi: 10.1109/34.655647.

[26] Y. LeCun et al., 1989. Backpropagation Applied to Handwritten Zip Code Recognition, Neural Computation, 1(4), 541-551.

[27] Szegedy, Christian & Vanhoucke, Vincent & Ioffe, Sergey & Shlens, Jon & Wojna, ZB. (2016). Rethinking the Inception Architecture for Computer Vision. 10.1109/CVPR.2016.308.

[28] Lior Wolf, Tal Hassner and Itay Maoz Face Recognition in Unconstrained Videos with Matched Background Similarity. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR), 2011.