DYNAMIC HAND TRACKING AND GESTURE DETECTION SYSTEM FOR MAN-MACHINE INTERACTION

A PROJECT REPORT

Submitted by

ESHAAN BAJPAI [RA1611003030675]
PRIYAL PATNEY [RA1611003030654]
ADITYA NAGORI [RA1611003030648]
ANKIT MISHRA [RA1611003030652]

Under the guidance of

Mr. R. Manikandan

(Asst. Professor, Department of Computer Science & Engineering)

in partial fulfillment for the award of the degree

of

BACHELOR OF TECHNOLOGY

in

COMPUTER SCIENCE & ENGINEERING

of

FACULTY OF ENGINEERING AND TECHNOLOGY

S.R.M. Nagar, Kattankulathur, Kancheepuram District

MAY 2020

SRM INSTITUTE OF SCIENCE AND TECHNOLOGY

(Under Section 3 of UGC Act, 1956)

BONAFIDE CERTIFICATE

Certified that this project report titled "**DYNAMIC HAND TRACKING AND GESTURE DETECTION SYSTEM FOR MAN-MACHINE INTERACTION** " is the bonafide work of "**ESHAAN BAJPAI [RA1611003030675], PRIYAL PATNEY [RA1611003030654],ADITYA NAGORI[RA1611003030648] ,ANKIT MISHRA [RA1611003030652],** ", who carried out the project work under my supervision. Certified further, that to the best of my knowledge the work reported herein does not form any other project report or dissertation on the basis of which a degree or award was conferred on an earlier occasion on this or any other candidate.

SIGNATURE

Mr. R. Manikandan
GUIDE
Asst. Professor
Dept. of Computer Science & Engineering

SIGNATURE

Dr. Rajendra Prasad Mahapatra
HEAD OF THE DEPARTMENT
Dept. of Computer Science & Engineering

Signature of the Internal Examiner

Signature of the External Examiner

ABSTRACT

Pattern recognition and Gesture recognition are the growing fields of research. Being a significant part in non-verbal communication hand gestures are playing vital role in our daily life. Hand Gesture recognition system provides us an innovative, natural, user friendly way of interaction with the computer which is more familiar to the human beings. Gesture Recognition has a wide area of application including man-machine interaction, automated sign language translation, immersive game technology, home automation etc. Gesture recognition can be conducted with techniques from computer vision and image processing. This project introduces a hand gesture recognition system to recognize real time gesture in unstrained environments. Efforts should be made to adapt computers to our natural means of communication: Speech and body language. A simple and fast algorithm using orientation histograms will be developed. It will recognize a subset of ML static hand gestures. The pattern recognition system will be using a transform that converts an image into a feature vector, which will be compared with the feature vectors of a training set of gestures. The final system will be a Neural Network implementation. To implement this approach, we will utilize a simple web cam which is working on 20 fps with 7 mega pixel intensity. On having the input sequence of images through web cam it uses some preprocessing steps for removal of background noise and employs K-means clustering for segmenting the hand object from rest of the background, so that only segmented significant cluster or hand object is to be processed in order to calculate shape based features. This proposed implemented algorithm will try to attain approximate recognition rate around 80-97

ACKNOWLEDGEMENTS

I would like to express my deepest gratitude to my guide, Dr. R. Manikandan his valuable guidance, consistent encouragement, personal caring, timely help and providing me with an excellent atmosphere for doing research. All through the work, in spite of his busy schedule, he has extended cheerful and cordial support to me for completing this research work.

Author

TABLE OF CONTENTS

CHAPTER 1

INTRODUCTION

1.1 Project Overview

Hand gesture recognition is of great importance for man-machine interaction. It has massive applications all around the world in the form of entertainment, utility, gaming industry etc. Sign language recognition is an active field of research. Researchers from all over the globe have written number of articles, papers and journals on Human-Computer Interaction (HCI). Different technologies and different models have been used over decades to make the process of man-machine interaction smooth and perfect. Machine learning and deep learning models like CNN, VGG19, and RESNET etc. have been used for gesture recognition. Despite lots of previous work, traditional vision based hand gesture recognition models have failed to perform with constant quality output. Because of the nature of optic sensor, its surrounding conditions and other factors, no significant outcome is seen. We have decided to build a deep neural network that can classify hand images with excellent accuracy and give fast results. Our model is trained on ASL dataset which consist of 87000 images. These images are collection of alphabets and hand signals for deaf people. We aim to predict the hand gestures made by deaf people in front of a camera or through web interface with 98 percentage accuracy using state of the art programming and pre-built models. We are using 50 layered model that can predict 29 different classes of images. It is trained on one of the best GPUs provided by Google over cloud. The architecture of our model is so well engineered that almost any medium mobile or web application can give outputs. The dataset has 87000 images which is already a huge amount. Above that, using augmentation techniques, we have trained our model with more than sufficient input data. Our model is scalable and optimized which is the key objective as very less number of significant application/ services have been developed upon this idea.

1.2 American Sign Language

American Sign Language is a complete natural language which has same linguistic properties as spoken language and is used by people who have impaired hearing or people who are deaf. All the communication is done via facial expression and hand gestures. It is used in America and in most parts of Canada. ASL is most closely related to sign language of France that is FSL. ASL is considered creole language of FSL. Creole language means a stable natural language which is developed by mixing or simplifying from other language. It originated in 1800s in the American School of Deaf. Since then, a lot of schools for deaf have made this language compulsory for its students. Despite the wide usage of ASL, no accurate count about the number of people using it is recorded officially. But few surveys have given an estimate range between 250,000 to 500,000 people including all men, women and children. Unfortunately, people using ASL face the stigma due to false belief of others that oral language is superior.

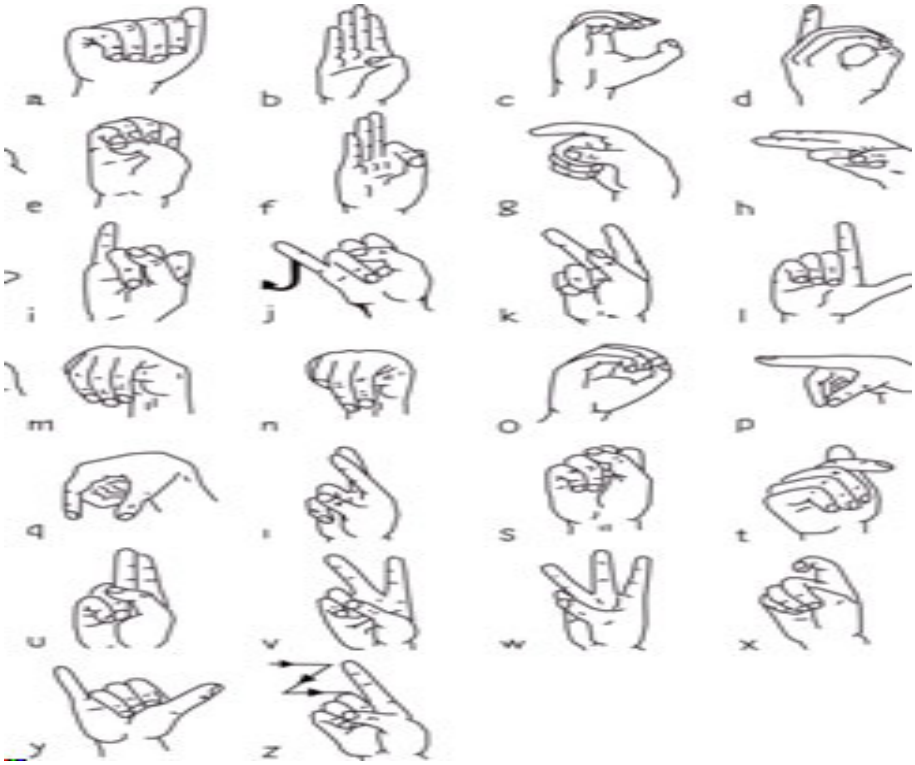


Figure 1.1: ASL

CHAPTER 2

LITERATURE SURVEY

2.1 Introduction

The most critical of all in today's life is communication- reading and writing. The way of communication in which any type of body movement is involved called Gestures. Gesture recognition is the mathematical interpretation by a computing device. Gestures are always expressive, meaningful body motions involving physical movements of the fingers, hands, arms, head, face, or body. There are many kinds of expressions of human movements, the common one is the expression of gestures.

Basically, Gesture is non-vocal way of communication which uses hand motion, different postures of body, face expressions. Gesture recognition is basically based on computer vision has gradually become a hot research direction in the field of human-computer interaction.

Sign language is the most expressive way for the hearing impaired, recognizer must be able to recognize continuous sign vocabularies in real-time. Gesture recognition based on attitude sensor is an emerging field of pattern recognition research. Experimental investigation proves the performance and high accuracy of any proposed device.

Normal way of using hand gesture recognition system is when we give some commands to our system by hand gestures the machine first captures our command as an image then compare this with database and if any image found in database then task assigned to that will be performed.

Normal way of using hand gesture recognition system is when we give some commands to our system by hand gestures the machine first captures our command as an image then compare this with database and if any image found in database then task assigned to that will be performed.

The most critical of all in today's life is communication- reading and writing. They feel communication medium difficult because they cannot access the computer. In some paper they have used Braille script for reading and writing purpose, which cannot be interpreted by the existing computers. The six fingers represent the six dots in the Braille.

Few papers have focused on Human Computer Interaction (HCI). In some paper they have used

Braille script for reading and writing purpose, which cannot be interpreted by the existing computers.

The six fingers represent the six dots in the Braille. A smart camera can be defined as a vision system which produces a high-level understanding of the imaged scene and generate application specific data to be used in system. Mono-vision based skin color segmentation techniques are used for segmenting the hand form a complex image sequence. The standard histogram features along with various geometrical features are extracted.

Some papers they have used End-Point problem to determine the end points in a gesture input sequence.

2.2 Literature Review

1. There are mainly two types of methods available in hand gesture recognition systems. The first one is based upon the computer vision and rely mainly upon the images which are fed to the model through the webcam or manually inserted as an input to the model, and the other type involve the use of some types of sensors which are made specifically for the data collection purposes. One such sensor is known as attitude sensor, this sensor collects the data in large amounts. The detection in second one takes place until a relevant information is extracted out from the provided output. In the end the generated data gets classified upon the features of input characteristics and further undergo post processing. Finally this implementation is mainly based upon the use of sensors.

2. Hand gesture is nothing but a mathematical functions when reorganized by the computer. In this paper they have implemented the computer vision base approach, this approach is fairly easy when compared to its other counterparts. They have trained the CNN classifier. This is judging the hands gesture based upon the pattern recognition, they have completely avoided the segmentation based upon the color of the skin.

3. A real-time gesture recognition system with continuous input from the user end using the basic webcam of the computer or the device upon which the application is executed. The main mechanism behind the working of this model is the use of input stream and then implementation of analysis happens upon this input based upon the various gyroscopic orientation of the gesture. In this the method used for the detection of the perfect end-point of the gesture us End-point problem.

4. Hand gesture recognition is considered as a more intuitive and proficient tool for human computer interaction. Gestures are expressive body motions to emphasize or help to express a thought or feeling Sign language recognition and gesture-based control are the two dominant applications that are used. Other applications include virtual prototyping, sign language analysis and medical training.

5. The vector feature is obtained by the tangent angle of the motion path of the palm. The forward-backward algorithm and Viterbi algorithm is used to solve the category of optimal trajectory sequence. The parameter model is instructed by using the Baum-Welch algorithm and lastly, to carry out model feature fusion to realize dynamic gesture recognition, D-S evidence theory is used.

6. Now-a-days 3 dimensional pictures are being used more frequently everywhere. In many papers, authors and researchers have described about 3D hand gesture recognition (HGR). These applications can be used in smart mobile devices such as head-mounted displays (HMDs), smartphones for Augmented reality /Virtual reality applications and also, they are using virtual 3D objects using depth sensing and hand tracking to enable the user interaction. Previously, in 3 dimensional HGR system, these sensors used low power and used convolutional neural network (CNN) which can be adopted to enhance the accuracy of the low-power stereo matching. These CNNs contain HGR in 6 layers. The CNN sensors can recognize the skin colour and skin texture to detect the hand accurately. The CNN model converts the image from 2D to 3D via some functions and have rendered the space and after doing some interactions they have obtained the output. [1]

7. Surface electromyography (EMG) sensor is heavily and frequently used in object motion recognition to identify low level hand movement by using the single channel surface EMG signal and in many papers, authors and researchers have explained about more use of the EMG sensors in different fields like signal acquisition system. This is a low-cost technique which uses a sensor network called surface EMGextensively used for hand motion recognition. The surface EMG is a sensor network which has a computer software and four (4) surface EMG sensors. They have extracted four set of domain feature from the raw EMG signals and after extraction these features are used to train the BPNN in MATLAB. The online motion of the hand recognition is done by these BPNN training data and the output of this system is the above average recognition accuracy. [12]

8. Blind people face many problems in their daily life. Most critical one is reading and writing. Communication is really difficult for them because they cannot access the computer to connect to the outside world. A real time embedded system is used to interact with an external environment, which may be any living organisms. In Braille script, six dots represent the six fingers. Some researchers have used microcontrollers in their paper which intakes the gestures and sends the input to the computer. The computer recognizes the gesture and prints it on the display as an output. The system is very cost effective as compared to other options available in the market. [9]

9. In this paper they use electrical impedance tomography (EIT). To produce High Accuracy Wearable Hand Gesture Recognition System, they have embedded electrodes to demonstrate the system interfaces and the forearm using a wrist wrap, which helps in measuring the inner conductivity distributions caused by bone and muscle movement of the forearm in real-time and passes the data to a deep learning neural network for gesture recognition. This system does this by measuring the bandwidth of 500 kHz and have measured the sensitivity in excess of 6.4 per frame. And this system also uses round robin subgrouping method to recognize the nineteen hand gestures and it achieved 98 percentage. [13]

10. Usage of gadgets has increased almost exponentially with the advancement of technology. So, to make intelligent machines we enable our machines to take the command by

recognizing the different hand gestures which we give as an input. After this, input is sent to databases in order to assign it some task. Now when we give some commands to the system by hand gestures, the machine first captures the command as an image then compare it with the database and if any result is found in the database then task assigned to that will be performed. [10]

11. Today, many industries are using the most common communication that is Human Machine interaction especially for Hand gesture recognition in the natural way. This technique enables the communication between humans and machine in easy way and does not need to use any extra device. In this paper, researchers are focusing mainly on the work done in the area of hand gesture recognition where the soft computing is based on methods like artificial neural network, fuzzy logic and genetic algorithms. They have also derived the hand detection methods in the form of preprocessed image for detecting the hand image.

12. Convolutional Neural Network represents a massive breakthrough in image recognition domain of machine learning. They're frequently used to analyse visual imagery and are frequently working behind the scenes in image classification. They are efficient and fast but they are not good enough for dynamic hand tracking. 3 layers of convolution along with pooling layers and modification of this type of neural network architecture does not increase the feature extraction and classification capacities of the network. Hence there is no significant improvement in accuracy, but only in the computational cost of the network. In a model of 3 convolutional layer, 2 max-pool layer, 2 dense layers and 1 flattening layer, the accuracy on imagenet database was computed around 52 percentage. This accuracy can be improved but by adding more layers which makes the coding part complex. [2]

2.3 Research Gap

Sign Language recognition is a challenging and demanding application among other human computer interaction applications[5]. Indian sign Language recognition is integrated research field of computer vision, image processing and pattern recognition. Indian sign language is the main mode of communication among deaf and dumb people of India.[3] [4] [11]

Many research work have been carried out on American sign language, British sign language, Japanese sign language and others countries sign language. Less research work has been carried out on Indian sign language[11]. Indian sign language changes from region to region as communication language[5]. However, in all large towns and cities across the Indian subcontinent, deaf people use sign languages which are not universal sign languages[5]. Further focus has been carried out to implement standardized Indian sign language by creating awareness among the sign language teachers. Due to communication gap, deaf and dumb community is isolated from the society. Due to lack of standardization of Indian sign language, research

work ISL interpretation started late in India[5]. There is need for computerized automatic sign language recognition system for hearing impairment and speechless people of India[11].
Sign Language Recognition is a very multifaceted task as task that uses hand shape recognition, gesture recognition, face and body parts detection and facial expression recognition as essential structure blocks[3] [4]. The problem of sign language recognition can be defined as the analysis of all components that form the language and the understanding of a single sign or a whole sequence of sign language communication. Static morphs of the hands called postures and together with hand movements called gestures. Hand gestures are most widely used as medium of sign language based communication framework among various forms of gestures[5]. With the help of advanced science and technology many techniques are developed by the researcher to make the deaf and dumb people communicate very fluently. More than 1 million of adults and 0.5 million children in India used Indian Sign Language[11].
So as to summarize everything we can say that following are the gaps for further development of Indian Sign Language recognition in the existing research literature:

- Indian sign language recognition should espouse data acquiring process in real time not. restricted to laboratory data. Further existing standard data sets are limited as it does not replicate real scenario.
- Established state of the art literature is mainly focused on alphabets/digits/static signs/ few words and sentences. Therefore research highlighting focus should be on dynamic signs and nonverbal kind of communication.
- Enhancement of various dynamic hand gesture recognition algorithms is needed to obtain more features to build up accurate system with large datasets. As review suggest that efficiency rate has not been elevated in real time environment.
- Systems should execute the recognition task in a user convenient and faster manner.

CHAPTER 3

MAJOR INPUTS

3.1 Neural Network

Neural networks is a system of neuron or neurons which could either be organic or can be artificial in their nature. A neural network is a series of algorithms that endeavours to recognize underlying relations in a data set through a process that mimics the way the human brain operates. So in regards to the previous statement, changing inputs in a neural network is feasible. The network is responsible for generating the best possible result without the need of redesigning the criteria for output. There is massive similarity in a computer's neural network and in the human brain's network. Neural networks helps us to classify and cluster. We can think of them as a classification and clustering layer on top of the data that you store and manage. They help to group unlabelled data according to similarities among the inputs, and they also classify the data when labelled dataset is available to train on. Feature extraction is also done using neural networks that are fed to other algorithms for classification and clustering, so we can think of neural net as a component of larger machine-learning application that involves algorithms for reinforcement learning, regression and classification.

3.1.1 Structure of Neural Network

Neural networks may be composed of several layers and not a single neuron or single layer. These can be dense or lightly dense in accordance to the need. The layer at the extreme left is called the input layer, and the neurons within the layer are called input neurons. The rightmost layer or better known as output layer contains the output neurons, or a single output neuron depending on the need of the model. The layer in the middle of input and output layer is called a hidden layer. This is because the neurons in this layer are neither outputs nor inputs. The term "hidden" perhaps sounds a little baffling. Designing the input-output layers of a network is mostly straightforward. Although designing hidden layer is not as easy as designing input–output layers. In particular, it is difficult to sum up the process of design for the hidden

layers with a few simple thumb rules. So now, people doing research on the same have developed many design heuristics for the hidden layers, which help people get the behaviour they want out of their neural networks. A layer of neuron is a row of those neuron-like switches that turn on or off again and again as the input is fed through the network. Each layer's output is the subsequent layer's input simultaneously. The first time someone hears the term, it looks thought it must have some deep mathematical or philosophical significance but it really means not an input or an output. There are models which have more complex applications of clustering and classifications which use networks with more hidden layer. The neurons are also called nodes in theoretical terminology. A neuron or node is the place where all the computation takes place, which is loosely patterned on a neuron in the human brain. This node or neuron fires when it encounters sufficient stimuli. The network is adjusted in such a way that comparison

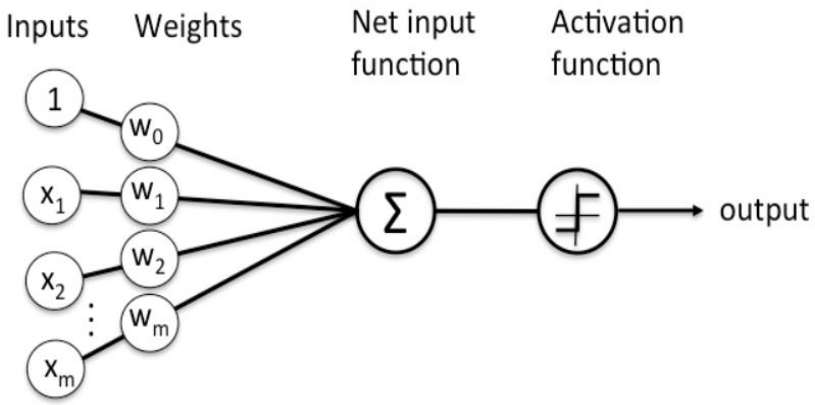


Figure 3.1: Neural Network

of the output and the target is done again and again until the network output matches the target. Many such input and target pairs are used typically. In today's time, neural networks are being trained to perform complex functions in various fields which are pattern recognition, speech, classification, identification, and control systems. Neural networks are being trained to solve problems that are difficult for conventional computers or human being.

The neuron is basically is a weighted average of input, then this sum is passed through an activation function to get an output. Activation function is just a function that is used to get the output of node. It also helps to normalize the output of any input in the range between 1 to -1. Activation function must should reduce the computation time because the neural network sometimes trained on thousands and millions of data points. Activation functions are necessary

because without them, weights and biases would have only transformations which are linear in nature.

Consider a neuron Y

$$Y = \sum(weight * input) + bias$$

Here, the neuron represented as Y can be anything for a neuron between -infinity to +infinity. So to bound the output and to get the desired prediction, Activation functions are used. These activation functions are mathematical in nature which are used to bound the output of the model in accordance to the requirement.

3.1.2 Activation Functions

1. **ReLU**: ReLU which is short for Rectified linear unit is widely used activation function which ranges from 0 to infinity. ReLU is mostly used function these days as it produces on positive outputs by negating or ignoring the negative side of the input. All negative values are effectively converted to zero. The conversion rate of ReLU is so fast that is sometimes responsible for underfitting. ReLU function can be mathematically writer as:

$$A(x) = max(0, x)$$

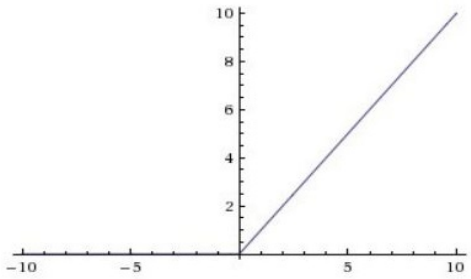


Figure 3.2: ReLU

2. **Leaky ReLU**: In ReLU, under fitting was a problem as all of the negative data was converted to zero and negative data was lost. Leaky ReLU solves this problem by converting the negative values into near zero but not zero values. This way, track of all input data is kept without loss of information. In the graph below, the slope is changed left of x=0

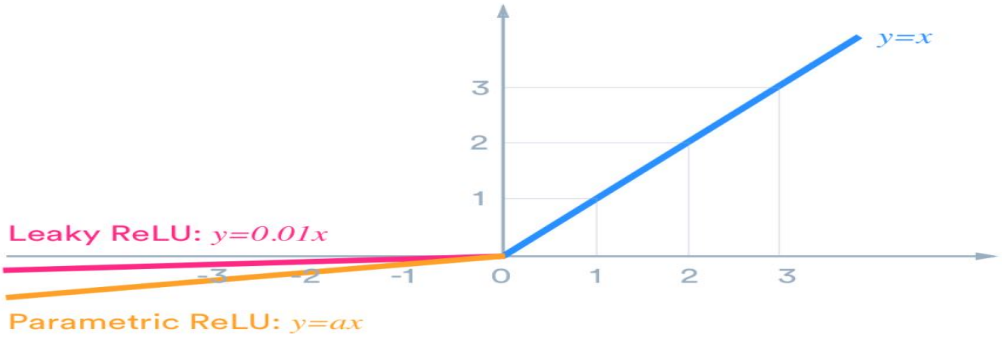


Figure 3.3: Leaky ReLU

and thus causing a leak which in turn extends the range of ReLU funcntion.

3. **Linear**: Linear, as the name suggest, is a simple straight line activation function. This function is directly proportional to the weighted sum of neurons or input. Linear functions give wide range of activations. They also give a line of a positive slope which may increase the firing rate as the input rate increases.

$$A = cx$$

Derivative with respect to x is c. This means that the gradient has no relationship with X.

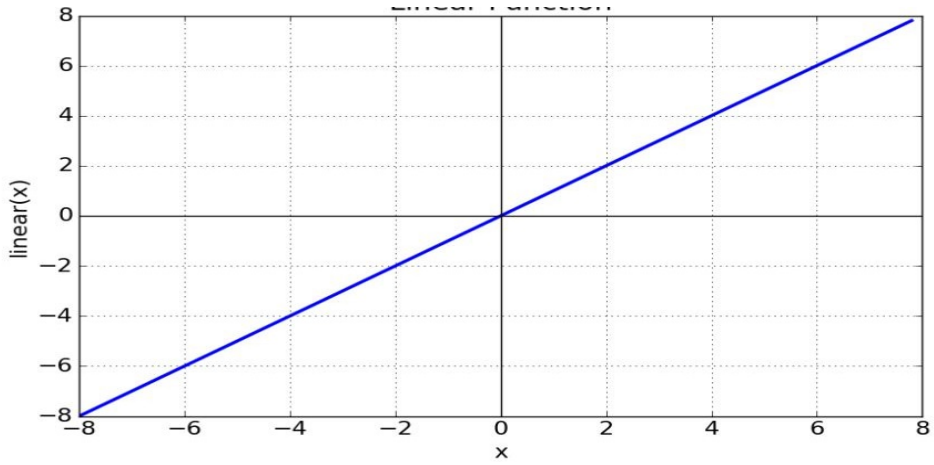


Figure 3.4: Linear Function

4. **Binary step**: This function is the most basic function in the pool of all activation functions. The dynamics of this functions is pretty basic. It is used to set threshold value for the required output. If the input is below the threshold, data will not be passed further. If the value is above the threshold, it will be passed further.
It can be explained in the following way:

Activation function A = "*activated*" if Y > threshold else *not*.
Alternatively, A = 1 if y> threshold, 0 otherwise

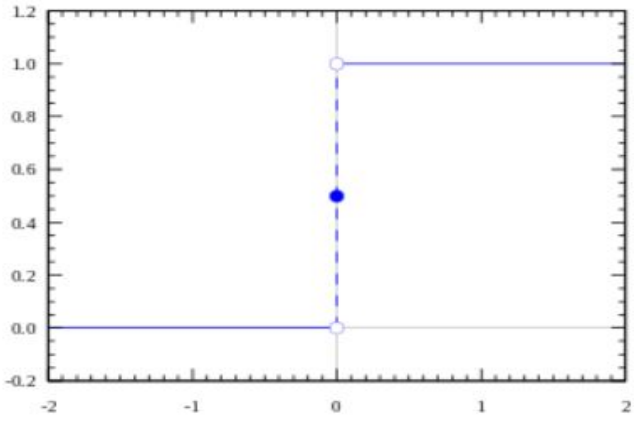
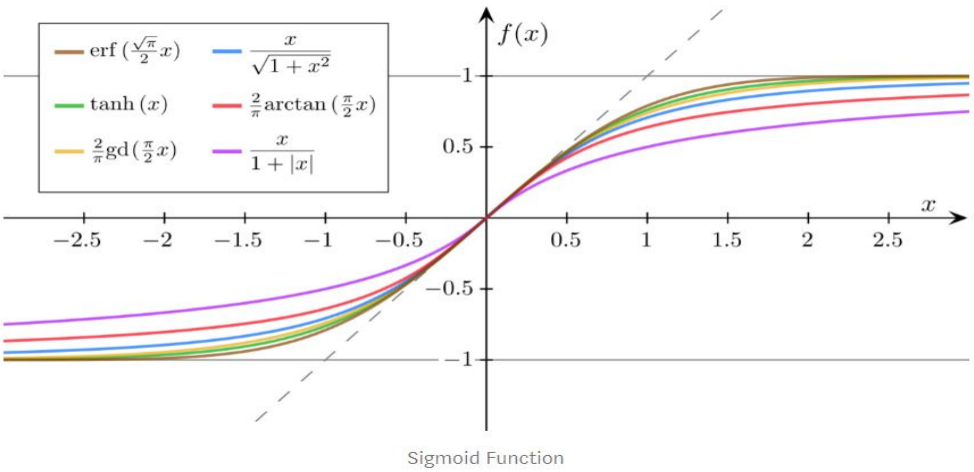


Figure 3.5: Binary Step

5. **Sigmoid**: Sigmoid functions are used for the basic neural network and implementations logistic regression. They are also the introductory activation units. Unfortunately, in advance machine and deep learning applications and models, sigmoid functions are not used due to its various drawbacks. Because of sigmoid function, more and more information is lost in dense neural networks. This problem cause unwanted data loss.



Sigmoid Function

Figure 3.6: Sigmoid

3.1.3 Neural computation and its Advantages

Hypothetical information processing performed by network of neurons is neural computation. It is affiliated with the philosophical tradition known as Computational theory of mind which is also referred to as computationalism. Few of its advantages are:

1. Neural Network can identify and recognize pattern in in the input data set. This is a powerful technique to harnessing the information in the data

2. Neural network is capable of high level multi-tasking with affecting the performance of the system

3. Neural Network have the ability to learn by themselves and perform tasks on unknown data set. This advantage makes it so useful

4. Neural networks can build informative models where more traditional model fail. Because of this, neural networks can handle very composite interactions

5. Data loss is not a concern while working with neural networks because they store data in their network itself and not in some database

6. Performance of neural networks is bottom-line as good as classical statistical model. It can be considered better on most problems

7. The network is capable of detecting faults and errors in a neuron on its own. Any piece of missing information can also be discovered

3.1.4 Limitation of Neural Computing

1. The most crucial setback is the neural network's impotence to explain the model it has built in a useful way. Analysts want to figure out why the model is developed as it is. Although neural networks get better answers, they have a hard time describing how it got there

2. It is tough to pull out rules from neural networks. This is mostly beneficial to those who have to explain their answer to others and to people who have been convoluted with artificial intelligence, particularly expert systems which are rule-based

3. Throwing data at a neural net and get a good answer is not remotely possible. We have to invest some hours trying to understand the problem or the outcome that we are trying to predict. We must be sure that the data we used to train the system is appropriate and is measured in a way that mirror the behaviour of the factors

4. Neural networks needs processors with parallel processing power. For the very reason, the dependency on hardware excellence is too much

5. In neural network, the difficulty of showing the problem to the networks real. ANNs can work with information which have numerical values. Shortcomings or problems have to be translated into numerical values before being introduced to ANN. The display method to be determined here will directly affect the performance of the network

6. A lot of time is required to train a model for a complex data set. Neural techniques require computer intensive processing or computation and will be slow on low end machines or machines without math coprocessors or support to cloud computing.

3.2 Tools

1. **ImageNet**: It is an image database organized according to the WordNet hierarchy, in which each node of the hierarchy is depicted by hundreds and thousands of images. It is an ongoing research effort to provide researchers around the world an easily accessible image database. As of now it has an average of over five hundred images per node. It has become a useful resource for researchers, educators, students and all of you who share our passion for images. In imagenet, there are more than 100,000 synsets in WordNet, majority of them are nouns (80,000+). It is aimed to provide on average 1000 images to illustrate each synset. Images of each concept are human-annotated and quality-controlled. ImageNet is inspired by a growing sentiment in the image and vision research field which is the need for more data. From the birth of the digital era and the availability of web-scale data exchanges, researchers in these fields have been working hard to design more and more sophisticated algorithms to index, organize, retrieve and annotate multimedia data. But at the end, good research needs good resource. This is the reason for us to use ImageNet. It is becoming a useful resource to the research community, as well as for anyone whose research and education would benefit from using a large image database.

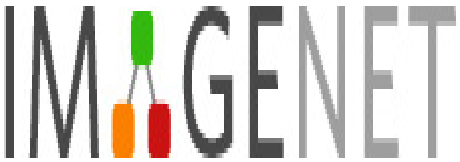


Figure 3.7: Imagenet

2. **Kaggle**: It is platform to compete with others in competitions which are based on machine learning tasks. It is a subsidiary of Google LLC. Kaggle is an online community of data scientists and machine learning practitioners. It allows users to find and publish data sets, explore and build models in a web-based data-science environment, work with other data scientists and machine learning engineers, and enter competitions to solve data science challenges. It has run hundreds of machine learning competitions since the company was founded It got its start in 2010 by offering machine learning competitions and now also offers a public data platform, a cloud-based workbench for data science, and Artificial Intelligence education. The competitions have resulted in many successful projects including furthering the state of the art in HIV research, chess ratings and traffic forecasting Everybody know about Codechef, Hackerrank etc., so kaggle is also like them, but the key difference is the competition are only related to machine learning, data science, deep learning or AI related. Mostly, the users are given some training and testing dataset to build some good machine learning models and when they public their kernel, others can review it and then gives you upvotes just like Quora. Kaggle provides with these services:

    - Machine learning competition
    - Kaggle kernels
    - Public dataset platform
    - Kaggle learn
    - Jobs board

Figure 3.8: kaggle

3. **Colab**: Colaboratory or better known as “Colab” is a product from Google Research. It allows anybody to write and execute python code through the browser. It is especially well suited to machine learning, data analysis and education purpose. They provide a free cloud service based on Jupyter Notebooks that supports free GPU. It is a great tool for improving your coding skills and also allows anyone and anybody to develop deep learning applications using popular libraries such as PyTorch, TensorFlow, Keras, and OpenCV. So in simplest terms, Google provides free GPU through colabs. This is very useful as accessing such tremendous processing power over internet is not usual. There is a limit to the file size and session duration. Although colab is ideal for everything from improving Python coding skills to working with deep learning libraries like PyTorch, Keras, TensorFlow, OpenCV and matplotlib etc. The code is executed in a virtual machine private to your account. Virtual machines are deleted when idle for a while, and have a maximum lifetime enforced by the Colab service It also gives us options to create notebooks in Colab, upload, store, share, mount your Google Drive and use whatever we

have got stored in drive, import favorite directories, upload personal Jupyter Notebooks, upload notebooks directly from GitHub, upload Kaggle files, download notebooks, and do just about anything and everything that a user might want to do.

Figure 3.9: colab

4. **Docker**: Docker is a popular open-source project which is based on Linux containers. It is a platform that works as a service products which uses OS-level virtualization to deliver software in packages known as containers. These containers are isolated from one another. They bundle their own software, libraries and configuration files. They can also communicate with each other through well-defined channels. Docker has some well-defined wrapper components which makes packaging application simple and easy. It separates the application from the infrastructure by packing all application system demands into a container. So the difference between docker and container is that docker is a technology or a tool developed to manage container implementations. Docker is designed to benefit both developers and system administrators making it a part of many DevOps or developer operations toolchains. Designers can concentrate on writing code without worrying about the system that it will finally be running on. Docker also allows developers to get a start by using any of the thousands of small codes that are already designed to run in a Docker container as a part of their application. Docker gives flexibility and likely reduces the number of systems needed because of its small footprint and low overhead for operational staff. Containers are thought of as necessitating three categories of software

    - Builder: It is a technology used for building containers
    - Engine: It is a technology used for running containers
    - Orchestration: It is a technology used for managing multiple containers

5. **Jupyter**: Project Jupyter is a non-profit organization. JupyterLab, which is web-based IDE for Jupyter notebooks, code, and data. It is flexible, it can configure and arrange the user interface to support a wide range of workflows in data science, scientific computing, and machine learning. JupyterLab is extensible. It is also modular. Developers can write plugins that add new components, features and integrate them with existing ones. So basically it is a blend of an IDE, server to run your projects which is called notebooks, either on developer's local computer or remotely at some other place through cloud. This notebook is a derivative project from the IPython, which used to have an IPython Notebook project itself. Jupyter name comes from the soul supported programming languages that it supports. These languages are Julia, Python, and R, hence Jupyter. It ships with the IPython kernel, which allows you to write your programs in Python, but there are currently over 100 other kernels that you can also use. It also has support for approximately

40 computer languages including java, JavaScript etc. Julia, python and R were the only languages it was built for originally. The jupyter notebooks contains both code as well as presentation elements such as images, calculations or both images and calculations together in one place. The jupyter notebooks are interpreted via virtual machines which are called kernels and they use memory of the computer running it. It can be used with Docker containers as well.

Figure 3.10: Jupyter

6. **Git**: Git is a version control system or VCS. There are many version control systems available. Some of them are CVS, SVN, Mercurial, Fossil etc. Git serves as the foundation for many services and products. Two of them are GitHub and GitLab, but git can be used without any other service. This means that Git can be used privately or publicly. So git is a distributed version-control system for stalking changes in source code during software development. It is designed to provide coordination in work among programmers and other users but it can be used to follow changes in any set of files too. Its goals consists of speed, data integrity, support for distributed and non-linear workflows. Git's user interface is akin to other VCSs. Git stores information in a very different way, and understanding these differences usually helps in avoiding confusion while using the software. Git stores its data like a series of snapshots of a scaled-down file-system. With Git, every time a programmer commit or save the progress of his/her project, Git simply takes a picture of what all your files look like at that moment and stores credentials of that snapshot. To provide more efficiency, files which have not changed doesn't get stored again, just a link to the previous identical file it has already stored. Git considers its data as more like a stream of snapshots.

3.3 Optimizer

3.3.1 ADAM

Adam is short for Adaptive Moment Estimation. Adam can be considered as a combination of RMSprop and Stochastic Gradient Descent with momentum [8]. It uses the squared gradients to improvise the learning rate like RMSprop and it takes edge of momentum by using moving average of the gradient instead of gradient itself like SGD with momentum. It calculates the adaptive learning rates for each parameter and works in the following way:

- Firstly, it computes the exponentially weighted average of past gradients
- Secondly, it calculates the exponentially weighted average of the squares of past gradients
- Thirdly, these averages have a bias towards zero and to counteract this a bias correction is applied to it
- At last, the parameters are revised using the information from the calculated averages

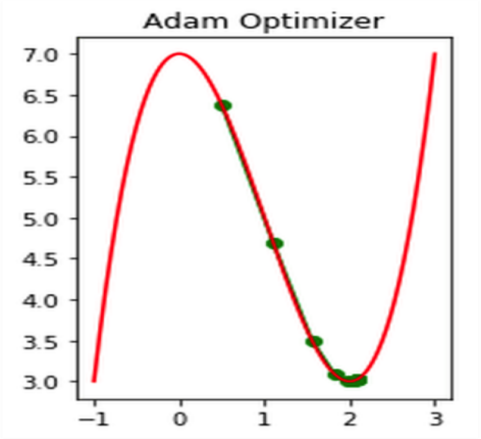


Figure 3.11: ADAM

Grasping the idea at the first time is pretty difficult. Gradient of the cost function of neural network can be considered a random variable, since it usually evaluated on some small random batch of data. The initial moment is mean, and next moment is uncentered variance. To approximate the moments, Adam utilizes exponentially moving averages, calculated on the gradient evaluated on a current mini-batch. Anticipated values of the estimators should equal to the parameter we are trying to estimate and as it happens, the parameter in this case is also

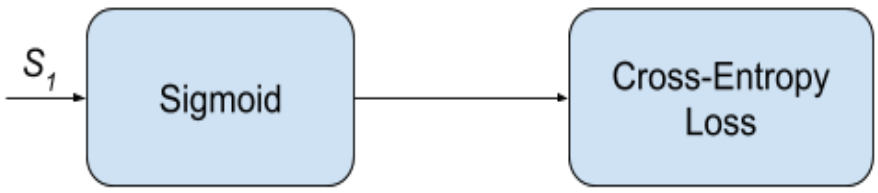
the expected value. If these properties remain true, it would mean that we have unbiased estimators. Now, we will see that these do not hold true for our moving averages. This is because we initialize averages with null value or zeros. The estimators are biased towards zero

3.4 Losses

3.4.1 Binary Cross-Entropy Loss

Also called Sigmoid Cross-Entropy loss. it's a Sigmoid activation plus a Cross-Entropy loss. Unlike Softmax loss it's independent for every vector component (class), meaning that the loss computed for each CNN output vector component isn't suffering from other component values. That's why it's used for multi-label classification, were the insight of a component belonging to a particular class shouldn't influence the choice for an additional class. It's called Binary Cross-Entropy Loss because it sets up a binary classification problem between C'=2 classes for each class in C, as explained above. So, when using this Loss, the formulation of Cross Entropy Loss for binary problems is usually used:

$$CE = -\sum_{i=1}^{C'=2} t_i log(f(s_i)) = -t_1 log(f(s_1)) - (1 - t_1) log(1 - f(s_1))$$



$$CE = -t_1 log(f(s_1)) - (1 - t_1) log(1 - f(s_1))$$

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

This would be the pipeline for everyone among the C classes. We set C independent binary

classification problems. Then we sum up the loss over the various binary problems: We sum up the gradients of each binary problem to backpropagate, and therefore the losses to watch the worldwide loss. s1 and t1 are the score and therefore the ground-truth label for the category C1, which is additionally the category

Ci in C. s2=1s1 and t2=1t1

are the score and therefore the ground-truth label of the category C2, which isn't a "class" in our original problem with C classes, but a category we create to line up the binary problem with C1=Ci

We will know it as a background class.

The loss is often expressed as:

Where t1=1 means the category C1=Ci is positive for this sample.

$$CE = \begin{cases} -log(f(s_1)) & if \quad t_1 = 1 \\ -log(1 - f(s_1)) & if \quad t_1 = 0 \end{cases}$$

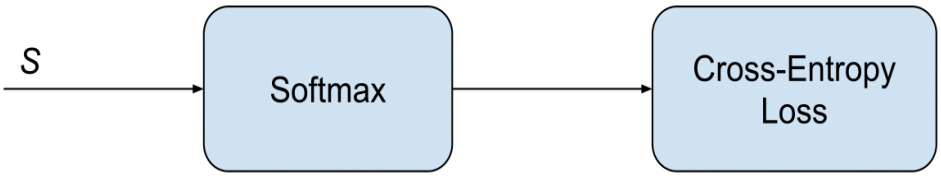
In this case, the activation function doesn't depend in many other classes in C more than C1=Ci. Therefore the gradient reference to the each score si in s will only depend upon the loss given by its binary problem. The gradient reference to the score si=s1 is often written as:

$$\frac{\partial}{\partial s_i}(CE(f(s_i)) = t_1(f(s_1) - 1) + (1 - t_1)f(s_1)$$

3.4.2 Categorical Cross-Entropy loss

Also called Softmax Loss. it's a Softmax activation plus a Cross-Entropy loss. If we use this loss, we'll train a CNN to output a probability over the C classes for every image. it's used for multi-class classification. Categorical cross-entropy is a loss function that is utilized for single label categorization. This is when only one section is suitable for each data subject. In other

words, an instance can belong to one class solely



$$f(s)_i = \frac{e^{s_i}}{\sum_j^C e^{s_j}} \quad CE = -\sum_i^C t_i log(f(s)_i)$$

In the specific (and usual) case of Multi-Class classification the labels are one-hot, so only the positive class Cp keeps its term within the loss. there's just one element of the Target vector t which isn't zero ti=tp. So discarding the weather of the summation which are zero thanks to target labels, we will write:

$$CE = -log\left(\frac{e^{s_p}}{\sum_j^C e^{s_j}}\right)$$

Where Sp is that the CNN score for the positive class.

Defined the loss, now we'll need to compute its gradient reference to the output neurons of the CNN so as to backpropagate it through internet and optimize the defined loss function tuning internet parameters. So we'd like to compute the gradient of CE Loss respect each CNN class score in s. Negative classes are giving the loss term as zero. However, the loss gradient respect those negative classes isn't cancelled, since the Softmax of the positive class also depends on the negative classes scores.
The gradient expression are going to be an equivalent for all C except for the ground truth class Cp, because the score of Cp(sp) is within the nominator.

After some calculus, the derivative reference to the positive class is:

And the derivative reference to the opposite (negative) classes is:

$$\frac{\partial}{\partial s_p}\left(-log\left(\frac{e^{s_p}}{\sum_j^C e^{s_j}}\right)\right) = \left(\frac{e^{s_p}}{\sum_j^C e^{s_j}} - 1\right)$$

$$\frac{\partial}{\partial s_n}\left(-log\left(\frac{e^{s_p}}{\sum_j^C e^{s_j}}\right)\right) = \left(\frac{e^{s_n}}{\sum_j^C e^{s_j}}\right)$$

Where sn is that the score of any negative class in C different from Cp.

CHAPTER 4

METHODOLOGY

In this chapter, the methods and the concepts used in the context of this probem domain will be discussed.

4.1 CNN

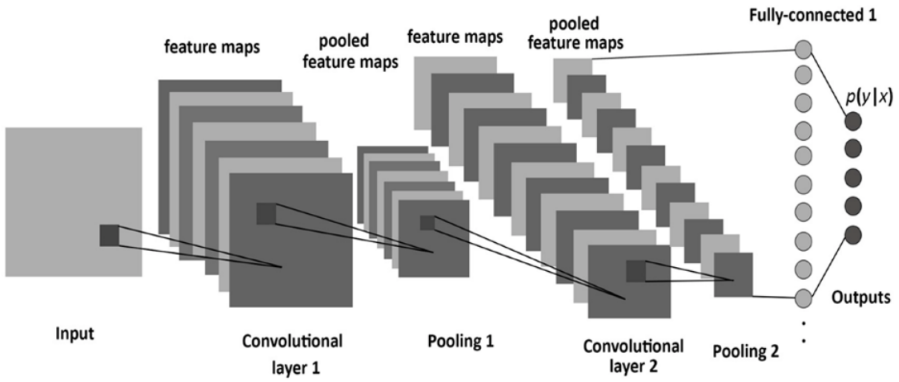


Figure 4.1: Architecture of CNN

CNNs are a fundamental example of deep learning, where a more sophisticated model pushes the evolution of computer science by offering systems that simulate differing types of biological human brain activity. Convolutional Neural Networks (CNN) [6] is one among the variants of neural networks used heavily within the field of Computer Vision. It derives its name from the sort of hidden layers it consists of. The hidden layers of a CNN typically comprises of convolutional layers, pooling layers, fully connected layers, and normalization layers. Here it simply implies that rather than using the conventional activation functions defined above, convolution and pooling functions are used as activation functions.

Convolutional Neural Networks (CNN) is one in all the variants of neural networks used heavily within the sphere of Computer Vision. It derives its name from the type of hidden layers it consists of. The hidden layers of a CNN typically carries with it convolutional layers, pooling

layers, fully connected layers, and normalization layers. Here it simply means instead of using the standard activation functions defined above, convolution and pooling functions are used as activation functions.

Convolution: Convolution operates on two signals (in 1D) or two images (in 2D): you can think of one as the "input" signal (or image), and the other (called the kernel) as a "filter" on the input image, producing an output image (so convolution takes two images as input and produces a third as output).

In layman terms it takes in an input signal and applies a filter over it, essentially multiplies the input signal with the kernel to get the modified signal. Mathematically, a convolution of two functions f and g is defined as which, is nothing but inner product of the input function and a kernel function.

$$(f * x)(i) = \sum_{j=1}^{m} g(i).f(i - j + m/2) \tag{4.1}$$

In case of Image processing, it's easier to visualise a kernel as sliding over a whole image and thus changing the worth of every pixel within the process.

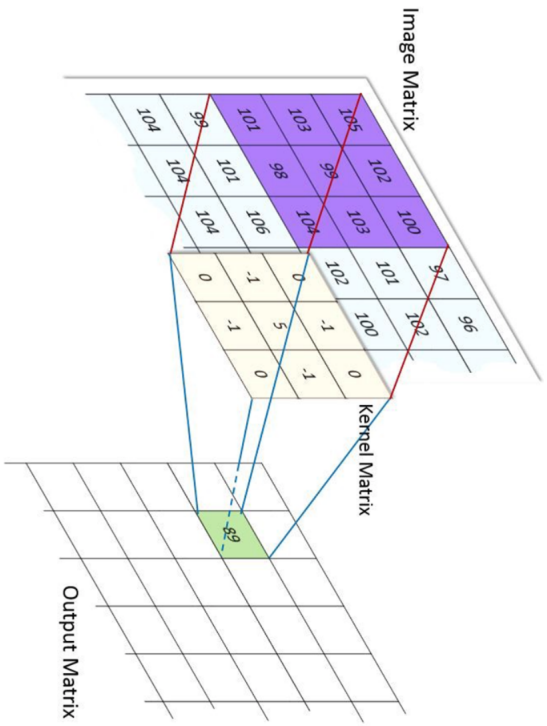


Figure 4.2: image input/output matrix

4.2 Computational Graphs

Neural networks are nothing but mathematical functions. This implies that after you taken off to use a neural network for a few task, your hypothesis is that there's some function which will approximate the observed behavior reasonably well. after we train a neural network we are attempting to search out one such reasonable approximation. Since these functions are often monstrously complex we use graphs to represent them instead of the quality formula notation. These graphs help us organize our considering the functions we taken off to make and it seems some graphs work far better than others for particular tasks. plenty of research and development within the neural network space is about inventing new architectures for these graphs, instead of inventing spanking new algorithms.

A computational graph could be a thanks to represent a math function within the language of graph theory. Recall the premise of graph theory: nodes are connected by edges, and everything within the graph is either a node or a foothold. In a computational graph nodes are either input values or functions for combining values. Each and every node in the following graph takes weights and biases as there integral part, as data flows through the network. Left most nodes behaves as input node and are weighted thereupon input values; these nodes from a function node are weighted by combining the weights of the inbound edges using the required function. For example, consider the relatively simple expression:

$$f(x, y, z) = (x + y) * (2) \tag{4.2}$$

this can be how we'd represent that function as as computational graph:

There are three input nodes, labeled X, Y, and Z. the 2 other nodes are function nodes. during a

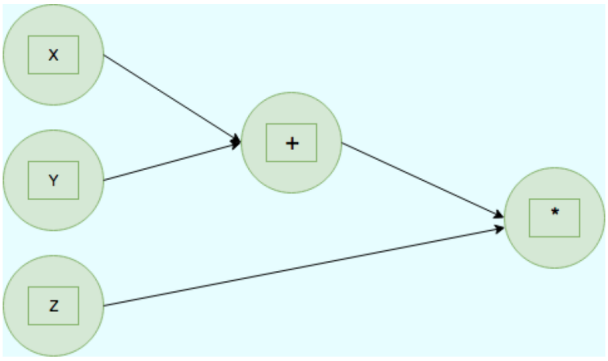


Figure 4.3: Computational graph

computational graph we generally compose many simple functions into a more complex function. we will do composition in notation additionally, but I hope you'll agree the subsequent isn't as clean because the graph above: In both of those notations we will compute the answers to every function separately, provided we do so within the correct order. Before we all know the solution to f(x, y, z) first we want the solution to g(x, y) then h(g(x, y), z).With the notation we resolve these dependencies by computing the deepest parenthetical first; in computational graphs we've got to attend until all the sides pointing into a node have a worth before computing the output value for that node. Let's study the instance for computing f(1, 2, 3).

$$f(x, y, z) = h(g(x, y), z)$$

$$g(i, j) = i + j$$

$$h(p, q) = p * q$$

Above is the example for what happens in the computation graph when we undergo the forward pass. Hence we can easily formulate the mathametical equation with the help of this forward pass. This methods holds true not only for this simple equations but much complex mathematical equations which are usually used in the actual neural networks in production, just the connections between each and every node becomes denser and denser based upon the level of complexity and the operators or operands which are used inside it.

Following is the example for a much complex equation undergoing the forward pass inside the computational graph:

Now after we have done the forward pass now is the time when the model is aware of the final equation we intend to put intend to utilize. After coming up with the equation is not enough, it is only the job half done. We have our final equation, we have to now optimize it so that it fits out data set and provide us with the output which we intend from it with minimum loss and maximum accuracy.

This step of optimization is also done with the help of this computational graph. In this we traverse through the nodes in the direction opposite to the forward pass which was originally used to find the equation. Since we are moving in the opposite direction it is known as backward pass.

In backward pass we now try to calculate the partial differentiation of the function formed by the nodes till that point, with respect to the other function it is equating to.

In calculus what we are doing by differentiation of the function is nothing but the finding the relationship between the two functions, how one function gets affected by making a small change in another function. Let us take an example for the same.

$$f(X) = Y^2$$

$$\frac{\partial f(x)}{\partial y} = 2 * y$$

From the equations above we can say that the attribute X is related to Y2 in the manner what we get after finding the differentiation of X with respect to Y2. So it's a safe bet to say that if we make any changes in the X it will impact the Y attribute by offsetting the graph by the factor of two units.

Let us understand the above approach by taking another example:

In this the above equation is found by the forward pass of the computation graph. Now if we

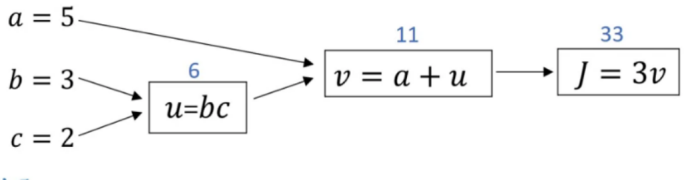


Figure 4.4: Example of a computation graph

start calculation of the partial differentiation we will come up with the following explanation. Since we have to work in the direction opposite to the forward pass we will start out journey of differentiation by finding the differentiation of the equation,

$$J = 3V$$

Let us assume the following assumptions:

$$v = 11 -> 11.001$$

$$J = 33 -> 33.003$$

The values 11.001 and 33.003 are just a little nudge to check the effects of change on function imparted on another function.

$$f(a) = 3a$$

$$\frac{\partial f(a)}{\partial a} = \frac{\partial f}{\partial a} = 3$$

$$hence : j = 3V$$

$$\frac{\partial j}{\partial v} = 3$$

Now, following the same approach we will move forward and get to our next equation, viz.

$$v = a + u$$

If we do the partial differentiation we will reach to the following equation,

$$\frac{\partial j}{\partial a} = \frac{\partial j}{\partial v} * \frac{\partial v}{\partial u} \tag{4.3}$$

$$\frac{\partial j}{\partial b} = \frac{\partial j}{\partial u} * \frac{\partial u}{\partial b} \tag{4.4}$$

We get the following results,

$$\partial a = 3, \partial u = 3, \partial b = 6, \partial v = 3, \partial c = 9$$

All these values are also known as slope of the function graph at that particular point. We try to minimize the loss in the output of the neural network model by finding the global minimum on the function graph. Hence these values of the partial derivatives helps us find that point in the function graph, as we compare the new updated values by the previous values and decide what we have to choose and what should be the set of new values which will give us the new minimum when compared to this current set.

This step of backward pass is also known as gradient descent, in neural networks as is a must to minimize the loss function for the particular set of values and cost function for all the dataset elements in general.

4.3 Residual Network

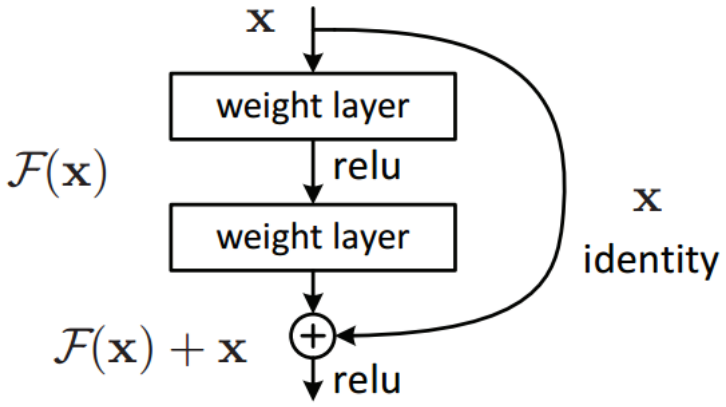


Figure 4.5: Single Residual Block

Neural networks are by default design are made to feed data into the layer by layer fashion, form vegetation cell in one layer to the vegetation cell of next layer of neurons it is in contact with. As this approach is benifitial and proved to be working very effitiantily for majority of the cases, but there is one problem which was discovered and rectified not long ago and is known as the problem of decay.

The neural networks proved to be an amazing piece of art when it comes to small number of layers. It is simple analogy that if you increase the number of data you want to be digested by the network then you have no other choice than to increase the number of layers involved. This might sounds like the problem can be solved by application and incorporation of this approach, but in reality the ground truth is completely different, the neural network has the tendency to loss its efficiency when the number of layers are increased.

Usually when you start to increase the number of layers involved the performance might seem to be in increasing order at first but this only last for a short amount of time. The efficiency of the neural network starts to get plateau at one point and when the number of layers increased further it eventually starts to get deteriorate. This problem of loss in the efficiency of the network due to the increase of the number of layers is known as the degrading problem.

So the above argument concludes that the accuracy of the neural network is not a function of the number of layers and we can also state the fact that shallower neural networks tends to learn better than their deeper counterparts[7]. This surely seem counter-intuitive but this is what is seen in practice and is popularly known as degradation problem.

This problem can be handeled by directly targeting what's the main root cause and why this happens? The main reason of this to occur is due to the large number of neural network layers the gradient tends to disappear as we go deeper into the deep neural networks. The most basic solution behind it is that we can solve it by feeding the output of the neurons not only just to the layer just next to it but to the neurons in the layer which are two to three hops away form it. This architecture tends to provide not only better results but are efficient in training deeper neural networks with large number of layers and with large size of data, all of this without using of any of that extra GPU or TPU power.
Let us understand this with the help of the mathematical equations as follows. Given a neural network block, whose input is x and we would like to learn the true distribution H(x). Let us denote the difference (or the residual) between this as.

$$R(x) = Output - Input = H(x) - x \tag{4.5}$$

After equating the above equation,

$$H(x) = R(x) + x \tag{4.6}$$

Our residual block is overall trying to learn the true output, H(x) and if you look closely at the image above, you will realize that since we have an identity connection coming due to x, the layers are actually trying to learn the residual, R(x). So to summarize, the layers in a traditional network are learning the true output (H(x))whereas the layers in a residual network are learning the residual (R(x)). Hence, the name: Residual Block.

4.4 Transfer Learning

Conventional machine learning and deep learning models, so far, have been traditionally designed to work in stand alone isolation. These models are trained to solve some tasks that are specific.The models have to be built and trained from scratch once the data distribution changes. Transfer learning is usually a scenario where what has been learned in one state is exploited to improve generalization in another state.

To understand transfer learning we need to understand the stark difference between the traditional concept of making and training machine learning models, and using a methodology following transfer learning concepts.

Traditional learning is task specific and is purely isolated , the type of dataset and training performed creates isolated models . No knowledge can be transferred from one model to another.

In transfer learning, you can use the retained knowledge (features, weights etc) from beforehand or previously trained models for training of newer models and even create solutions for problems that have less data for the newer task.

Let's try to understand the condescending explanation with the help of an illustration. Let's just assume our task is to classify objects in images within a subset domain of a restaurant.Let T1 be the mark to define the current task in its scope. For the given dataset we train the model to obtain the classification of restaurants .Traditional supervised learning ML algorithms fails when the data points or observation present in the dataset is lower or not sufficient enough for the model to learn well or generalise to a solution. Let task T2 be to classify or identify the objects in a park or a cafe. Theoretically we should be able to apply the model trained for T1, in this problem scope but practically the model faces some performance degradation, which is collectively and liberally termed as the model's bias towards the training or problem domain.

Transfer Learning thus enables us to apply knowledge gained from previously learned tasks and utilize them to different but contextually related tasks. Generally if we have sufficient enough data for task T1 the knowledge gained(weights, baises etc) can be transferred to task T2.

A formal definition of Transfer Learning. A domain, D, is defined as a two-element tuple consisting of feature space, x, and marginal probability, P(), where is a sample data point. Thus, we can represent the domain mathematically as

$$D = \{x, P(X)\}$$

Domain contains two things:

Feature space: x Marginal Distributions:

$$P(X), X = \{x_1, x_2, ...x_n\} \in X$$

The function can be denoted as P(y|x) from a probabilistic point of view, Y can be two label tuple.

For a given domain D, a task is defined by two components:

$$T = \{Y, P(Y|X)\} = \{Y, n\}; Y = \{y_1, y_2, y_3.....y_n\}, y_i \in y$$

A label space: Y

A predictive function n.

Learned from feature vector/label pairs, (xi, yi) For each feature vector in the domain, predicts its corresponding label:

$$n(x_i) = y_i$$

Transfer learning Strategies

- Inductive Transfer learning In this case, where the source and target domains are the same, yet the source and target tasks are different from one another.
- Unsupervised Transfer Learning it is similar to inductive transfer learning but with the scenario of having an unsupervised domain for the task.
- Transductive Transfer Learning: this is the scenario where there are similarities in the source and the target but the task domains are different to one another.

4.5 Model Evaluation Criteria

There are broadly two categories of model evaluation that are hideout and cross-validation. Set dataset is used in both of them to evaluate models Performance. It is not essentially a good practice to use the dataset that we trained on as a testing dataset because the model could completely remember the data set to perform well in evaluation criteria which could induce overfitting.

4.5.1 Holdout

In hideout evaluation the model is evaluated on a different dataset then it was trained on. Unbiased estimation of learning performance is provided as a result. Three subset of dataset is created randomly in this method:

- Training set A subset of dataset used for the training.
- Validation set A subset of dataset used for the model evaluation of the model during the training phase. Helps in fine tuning the model during the training phase.
- Test set it is a subset of a dataset which is unseen by the model and is used for the evaluation during the testing phase.

4.5.2 Cross-Validation

An independent dataset is used for the evaluation of the model apart from the dataset used during the training phase.
K-fold cross validation is the most common validation technique in which the original data observations are divided into k equal size subsamples. K is chosen by the user and is also can be considered as a hyperparameter. 5 to 10 is the usually the preferred value for k. The process is repeated k times and one of the k subset is used as the test/validation set and the other part that is the other k-1 sets are put together to create a training set.he error estimation is averaged over all k trials to get the total effectiveness of our model.

4.5.3 Model Evaluation Metrics

The quantification of our models performance is represented by models evaluation metrics. The general choice of matrices chosen depends on the different machine learning problem. Some of the supervised learning Model evaluation metrics are as follows.

Classification Accuracy

The most common evaluation matrix for model evaluation is Accuracy. It is the ratio of the total number of correct predictions made to all the predictions made.

Confusion Matrix

More detailed breakdown of correct and incorrect classification is provided in a confusion matrix. For any classification problem the diagonal elements present in the matrices are the

number of points for which the prediction was correct and anything apart from the diagonal element is a bad classification. Thus as a general rule, the higher the diagonal values indicates better model performance.

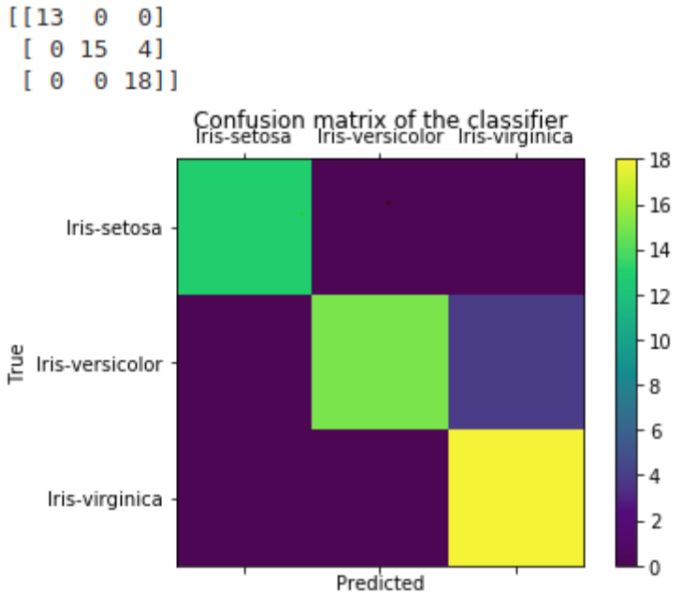


Figure 4.6: Confusion Matrix

Logarithmic Loss

The performance of classification models where the prediction input is a probability value between 0 and 1 is done using a logarithmic loss matrix. Smaller value of log loss is better as the function increases the predicted probability diverging from the actual value. A perfect model would have 0 logloss.

Area under Curve (AUC)

A quantified discriminate between positive and negative of a binary classifier is measured using the area under the ROC curve. In the example above, the AUC is relatively close to 1 and greater than 0.5. A perfect classifier will have the ROC curve go along the Y axis and then

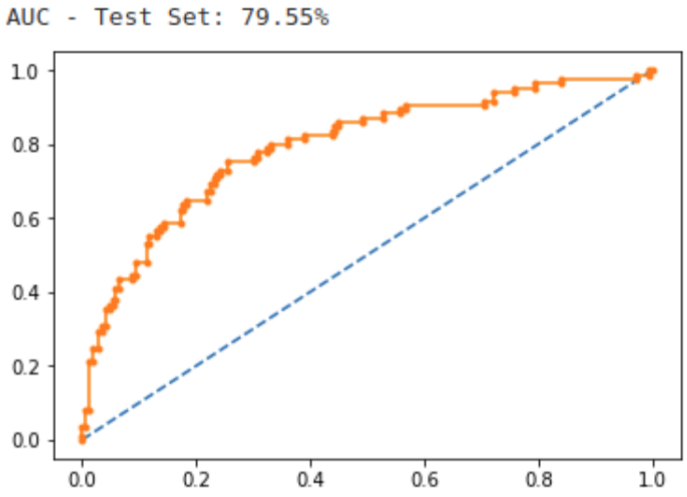


Figure 4.7: Area under Curve

along the X axis.

F-Measure

Test's accuracy for both precision and the recall is computed to measure the F-score or F-measure. Precision is the number of correct positives divided by the total predicted data observations. Number of correct positive results divided by the number of all relevant samples is the recall.

CHAPTER 5

PREPROCESSING

Datasets are usually some large files with a huge number of rows and columns. While that is generally true , it is not the case — data could be in so many different forms: Structured Tables, unstructured tables, json, Images, Audio files, Videos etc.

Machines don't always understand free text, image or video data as is, they only understand binary.

In any Machine Learning pipeline, Data Preprocessing is that step in which the data is transformed, Encoded or augmented to bring it to such a shape or state that now the machine can parse it. More likely the features of the data can now be easily interpreted by the sequence of computational steps or algorithms.

A dataset can be referred to as a collection of data objects, which are mostly called as records, points, vectors, events, cases, samples, observations, or entities. Data objects are explained by a number of features set , that observe the basic characteristics of an object. Features are usually called as variables, characteristics, fields, attributes, or dimensions, dims.

A feature is an individual quantifiable property or characteristic of a phenomenon being under observation.

Features can be of two types:

- Categorical : Features whose values are taken directly from a defined set of values. For example, days in a week : Monday, Tuesday, Wednesday, Thursday, Friday, Saturday, Sunday is a class because its value is always taken from this set. Another instance could be the Boolean set : True, False or 1 , 0
- Numerical : Features whose values are continuous or numeric valued. They are represented by inteteger and possess most of the properties of numbers. For instance, the number of days in a year.

Following are the steps of Data Preprocessing. One thing to note is, not all the steps are necessary for each problem, it is majorly dependent on the data we are working with but Generally they are :

- Data Quality Assessment
- Feature Aggregation
- Feature Sampling
- Dimensionality Reduction
- Feature Encoding

5.1 Data Quality Assessment

Data is often taken from different sources which are normally not too reliable and that too in different formats, thus it makes a challenge to consume the data for our machine learning.It is sometimes unrealistic to expect that the data will be perfect. Following are the methods to deal with inconsistencies in the dataset.

5.1.1 Missing values

Generally there are missing values in the dataset . It could have happened during data collection, or some validation issue, but regardless missing values must be corrected or taken into consideration.

- Removing rows with missing values :Simple and effective strategy. If a feature has a large number of missing values, then that feature itself can be dropped.
- Removing missing values :If only a considerable amount of values are missing, then we can usually run a normal interpolation method to fill in those values. However, the most common method of dealing with missing values is by filling them in with the statistical analysis generally mean, median or mode value of the feature in question.

5.2 Inconsistent values

Data can contain inconsistent values. Generally it is some trivial misplacing of values on some other fields. It could be due to human error or maybe the data was misinterpreted while being scanned from a handwritten format.

- It is therefore always a good practice to perform a data assessment like knowing what the data type and getting the knowledge of the shape data.

5.3 Duplicate values

A dataset could include data objects which are copies of one another. The term deduplication is often used to refer to the process of dealing with copies of objects.

- In most cases, the duplicates are removed so as to not give that particular data object an advantage or bias, when running machine learning algorithms.

5.4 Feature Aggregation

Feature Aggregations is the cumulation of values in order to put the data in a better perspective. These aggregate of values can be collected and created as a objects

- As a result memory is saved.
- Aggregations provides a high-level view of the data as the characteristics of groups or aggregates is more stable than individual data objects itself.

5.5 Feature Sampling

Sampling is a very general method for selecting a subset of the data that we are to analyze. Usually working with the entire dataset can turn out to be too computationally expensive considering the complexities. Using a sampling predefined algorithm can reduce the size of the dataset to a state where it can be used better.
Principle here is that the sampling should be done in such an order that the sample generated should have approx. the same characteristics of the original dataset, thus being a mathematical representative of the dataset. Thus having a good sample size and having a good sample strategy is important.
Simple Random Sampling dictates that there is an equal probability distribution of selecting any particular object. Following are the two main variations:

- Sampling without Replacement :the item selected is removed from the dataset and sample is created.
- Sampling with Replacement : The items are not removed from the original dataset and the sample is created.

An Imbalanced dataset is one in which the number of objects of a class(es) are significantly larger than another class(es), thus leading to an imbalance and creating rarer class(es).

5.6 Dimensionality Reduction

Most datasets in the world have a large number of features. Lets just consider the problem in context, image processing problem, we usually have to deal with millions of features and parameters generally known as dimensions.thus dimensionality reduction helps reduce the number of dimensions.
Mathematically , dimensions refers to the number of geometric planes on a coordinate system in which the dataset lies, which could be high to the point that it cannot be visualized with classical tools. Generally the More the number planes, the more complex dataset is.

5.7 The Curse of Dimensionality

This phenomenon simply refers that the task of analysing the data gets complicated as the dimensions of the dataset increases .
As the dimensionality increases, the number planes in the representation of data increases thus adding more and more sparsity to the data which is difficult to model and analyze.
Following are the major benefits of dimensionality reduction :

- Data representation and analysis algorithms work better if the dimensions of the dataset is lower.
- If Dimensions are reduced the visualization of the data set becomes easy and less computationally heavy.

5.8 Feature Encoding

The whole purpose of data preprocessing is to encode the data in such a state that a machine could understand it that is in binary format.

Feature Encoding is a process of encoding the data in such a way that the data remains true to the original meaning and can be easily understandable by a computer or a machine.

Following are the general norms of data encoding:

- Nominal : permutation of values like one-hot-encoding is generally encoding the data in one-to-one mapping.
- Ordinal : The chunks of data output blocks are created and separated in the output. Such as the concept of small, medium and large. Thus increasing the ordanality of the dataset.

5.9 Train / Validation / Test Split

After the data is being preprocessed the data needs to be in a specific format so that machine learning algorithms could understand it.

- Training data : This is a subset of a dataset in which the machine learning model is actually trained on. The dataset comprises the actual labels in case of a supervised learning model. This dataset is on which the training occurs and the model learns through it.
- Validation data : This is the part of our dataset which is used for the validation of model. It is usually a subset of the training dataset.
- Test data : This is the subset of the dataset which is used to test the hypothesis.
- Split Ratio : Data split ratio is a hyperparameter that needs to be tuned for case specific tasks. It justifies the split of the original dataset in training data, testing data and validation data.

CHAPTER 6

DETAIL MODEL DESIGN

Deep learning Model optimization is one of the toughest tasks in the implementation of machine learning algorithms. There is a whole branch of deep learning and machine learning theory dedicated to hyperparameter optimization and tuning. Normally, we think about model optimization as a process of regularly making changes in the code of the model in order to minimize the testing loss. However, machine learning model or deep learning model optimization often refers to fine tuning elements that live outside the model but that can heavily influence its behavior of the model.

Hyperparameters are nobs of settings that can be turned up and down to control the behavior of a machine learning algorithm. Theoretically, hyperparameters can be considered orthogonal to the deep or machine learning model itself in the sense that, although they live outside the model, there is a direct relationship between one another.

Generally ,the classification of what defines a hyperparameter is highly abstract and flexible. Sure, there are a lot of well created hyperparameters such as the number of hidden units or the learning rate of a model but there are also a multiple number of settings that can play the role of hyperparameters for specific models. In general, hyperparameters are very specific to the type of machine learning model under optimization. Usually, a setting is modeled as a hyperparameter because it is not correct to learn it from the training set. A classical example are settings that control the capacity of a model (the range of functions that the model can represent). If a deep learning model learns those changes directly from the training set, then it is likely to try to optimize for that dataset which will cause the model to overfit(poor generalization).

Validation dataset plays an important part in the hyperparameter tuning as it helps validating the model and reduces the overfitting of the model.

The number and classes of hyperparameters is specific for each model in machine learning and deep learning. There are some classical hyperparameters that can be used to do general optimization.

- Learning Rate: it is the process of moniraterning the learning progress of a model.
- Number of Hidden Units: A classical way of optimizing the model is selecting the number of hidden units in a model.
- Convolution Kernel Width: In convolutional Neural Networks(CNNs), the Kernel Width influences the number of parameters in a model which in turns, changes its capacity and helps fast processing of the image.

6.1 Model selection

6.1.1 ResNet50

Deep Convolutional neural networks are really great at identifying varying range of features from the images and arranging more layers on top of each other generally gives us better results so a but getting better accuracy is not as easy as stacking up layers on top each other.
With stacking up layers result in the arises of a problem of vanishing/exploding gradients these problems were heavily handled by multiple ways and enabled networks with tens of layers to converge but when deep neural networks start converging it is evident to see another problem of the accuracy getting on a plateau and then lowering rapidly and this is not caused by overfitting as one may guess and adding more layers to a suitable deep learning model just increased the factor of training error.
This problem is rectified by taking a superficial model and a deep model that was constructed with the layers from the shallow model and adding identity layers to it and accordingly the deeper model shouldn't have produced any higher training error than its counterpart as the added layers were just the identity layers.
The problem was solved by making a superficial model and a deep model which was constructed using the layers of the superficial model and adding unitary layers to it and respectively the deeper model should not have produced any significant inflation in the training error then its counterpart as the added layers are normal identity layer of superficial model. Authors of the Resnet[**?**] models addressed this problem by creating a shortcut connection that normally performs identity mapping this framework is called residual learning framework. Generally, the layers fit a residual mapping explicitly which is denoted by H(x) and another mapping

$$F(x) = H(x) - x$$

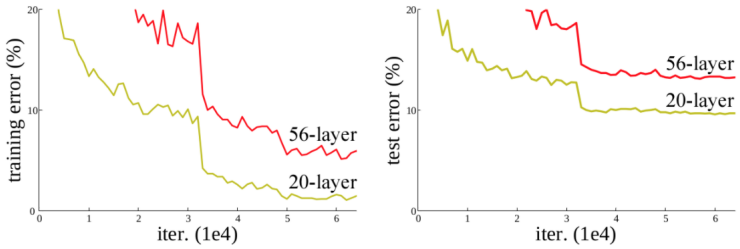


Figure 6.1: Comparison Matrix

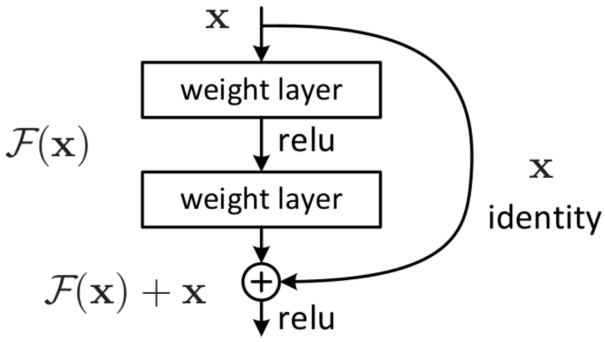


Figure 6.2: Single Residual Block

fit the nonlinear layers so the actual mapping becomes.

$$H(x) := f(x) + a$$

which can be seen in And as a result of shortcut identity mapping is that no additional parameters added to the modeland the computation is also remains in check.

6.2 Comparison

Figure is the demonstration of how a 34 layer pain model is compared against a 18 layer residual network ResNet. And the results are astounding the 18 layer residual network outperformed the 18 layer plain network. Thus stating the fact that increasing the layers does not necessarily increase the accuracy.

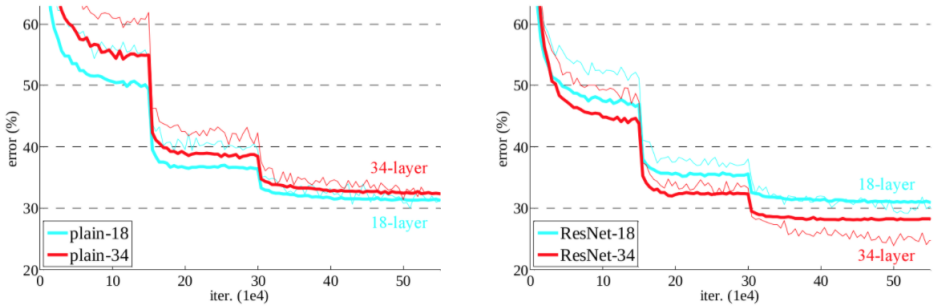


Figure 6.3: Comparison

6.3 Architecture

layer name	output size	18-layer	34-layer	50-layer	101-layer	152-layer
conv1	112×112	7×7, 64, stride 2				
conv2_x	56×56	3×3 max pool, stride 2				
		$\begin{bmatrix} 3\times3, 64 \\ 3\times3, 64 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 64 \\ 3\times3, 64 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$
conv3_x	28×28	$\begin{bmatrix} 3\times3, 128 \\ 3\times3, 128 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 128 \\ 3\times3, 128 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times8$
conv4_x	14×14	$\begin{bmatrix} 3\times3, 256 \\ 3\times3, 256 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 256 \\ 3\times3, 256 \end{bmatrix}\times6$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times6$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times23$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times36$
conv5_x	7×7	$\begin{bmatrix} 3\times3, 512 \\ 3\times3, 512 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 512 \\ 3\times3, 512 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$
	1×1	average pool, 1000-d fc, softmax				
FLOPs		1.8×10^9	3.6×10^9	3.8×10^9	7.6×10^9	11.3×10^9

Figure 6.4: Model Architecture

Now we are going to discuss about Resnet 50 and also the architecture for the above talked 18 and 34 layer ResNet is also given residual mapping and not shown for simplicity. There was a small change that was made for the ResNet 50 and above that before this the shortcut connections skipped two layers but now they skip three layers and also there was 1 * 1 convolution layers added that we are going to see in detail with the ResNet 50 Architecture. So as we can see in the table 1 the resnet 50 architecture contains the following element:

- A convoultion with a kernel size of 7 * 7 and 64 different kernels all with a stride of size 2 giving us 1 layer.
- Next we see max pooling with also a stride size of 2.
- In the next convolution there is a 1 * 1,64 kernel following this a 3 * 3,64 kernel and at last a 1 * 1,256 kernel, These three layers are repeated in total 3 time so giving us 9 layers in this step.

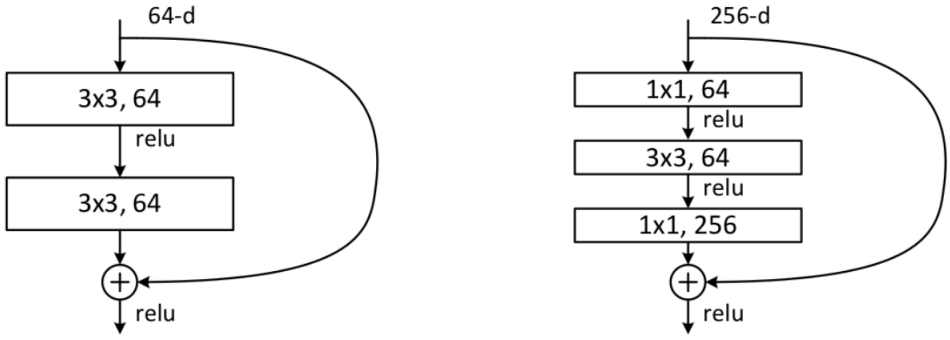


Figure 6.5: Different Residual Blocks

- Next we see kernel of 1 * 1,128 after that a kernel of 3 * 3,128 and at last a kernel of 1 * 1,512 this step was repeated 4 time so giving us 12 layers in this step.
- After that there is a kernal of 1 * 1,256 and two more kernels with 3 * 3,256 and 1 * 1,1024 and this is repeated 6 time giving us a total of 18 layers.
- And then again a 1 * 1,512 kernel with two more of 3 * 3,512 and 1 * 1,2048 and this was repeated 3 times giving us a total of 9 layers.
- After that we do a average pool and end it with a fully connected layer containing 1000 nodes and at the end a softmax function so this gives us 1 layer.

We don't actually count the activation functions and the max/ average pooling layers. so totaling this it gives us a 1 + 9 + 12 + 18 + 9 + 1 = 50 layers Deep Convolutional network. The Result were pretty good on the ImageNet validation set, The ResNet 50 model achieved a top-1 error rate of 20.47 percent and and achieved a top-5 error rate of 5.25 percent, This is reported for single model that consists of 50 layers not a ensemble of it. below is the table given if you want to compare it with other ResNets or with other models.

method	top-1 err.	top-5 err.
VGG [41] (ILSVRC'14)	-	8.43†
GoogLeNet [44] (ILSVRC'14)	-	7.89
VGG [41] (v5)	24.4	7.1
PReLU-net [13]	21.59	5.71
BN-inception [16]	21.99	5.81
ResNet-34 B	21.84	5.71
ResNet-34 C	21.53	5.60
ResNet-50	20.74	5.25
ResNet-101	19.87	4.60
ResNet-152	**19.38**	**4.49**

Figure 6.6: Performance of different models

CHAPTER 7

IMPLEMENTATION

7.1 Data Flow

According to our proposed model from hand gesture recognition system, holds different stages whose detailed elaboration is presented below. The execution of the hand gesture recognition model has also shown in Fig 1. Capture an image through available web cam or manually

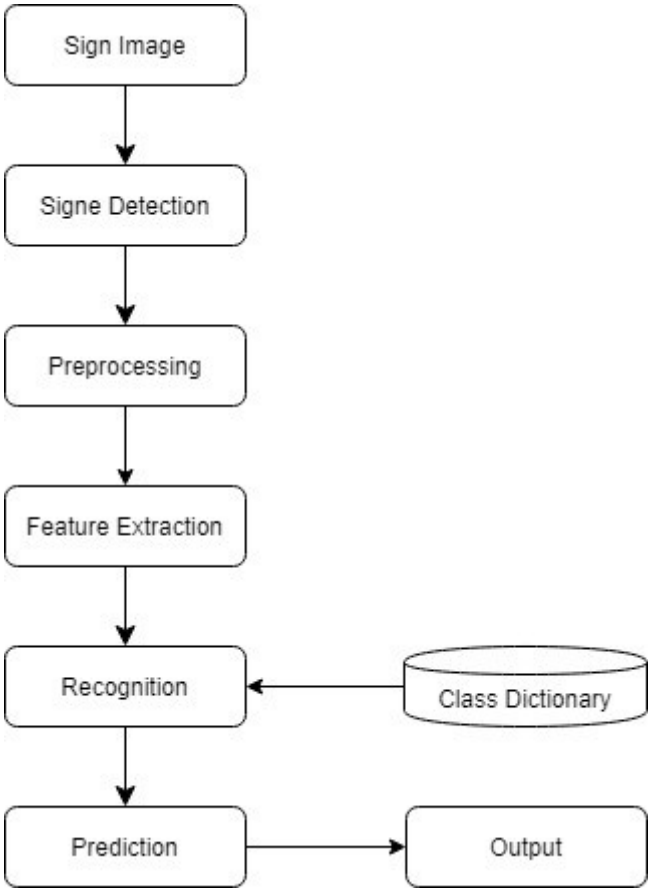


Figure 7.1: Flow-chart for hand gesture recognition

add the image of hand gesture. Frame extraction of the sign is done. After that sign image is transformed from RGB color space to inverted color space, hence making the hand detection based on skin color.

Once the hand sign is detected, now the image is converted to the monochrome color space,

and other steps explained under pre-processing heading.

Then for feature extraction all the parameters are taken into consideration and applied to the input image though pre-processing pipeline.

Finally the sign is extracted through the class dictionary which is maintained by our model. Hence prediction is made and the final output is rendered to the user interface through the services created.

7.2 Collecting the dataset

The asl data set is provided by a kaggle which comprises 29 classes of hand images belonging to x.

$$x \in \{a, b, c, d, e.....blank, space\}$$

The image shape is inconsistent and has a mean of (70.452,71.75). Each observation in the dataset is a 3 channel image i.e. the image is of RGB value. Collection of the data requires a kaggle api key to download the dataset. It is stored in kaggle.json file and the console downloading requires an API token.

```
!kaggle datasets download -d grassknoted/asl-alphabet
```

Figure 7.2: API Token

7.3 Preparation of dataset

All the classes of the ASL hand gestures are in seperate folder of namly A-hand, B-hand.... etc. These folders are needed to be extracted. For the preparation of data set all the files are collected in a single data frame object (df).

```
files_df = pd.DataFrame({
    'filename': File_name,
    'label': Image_labels
}).sample(frac=1, random_state=42).reset_index(drop=True)
```

Figure 7.3: Data set Preparation

7.4 Splitting the data set in train, validation and test subsets

This is achieved by using a module in python SKlearn.py and a ‘train_test_split()‘ function. The data set is divided into 70% percent training data set and 30% test data set. From the training data set another 10% split is generated for the validation data set. Because of the large

```
train_files, test_files, train_labels, test_labels = train_test_split(files_df['filename'].values,
                                                                      files_df['label'].values,
                                                                      test_size=0.3, random_state=42)
train_files, val_files, train_labels, val_labels = train_test_split(train_files,
                                                                    train_labels,
                                                                    test_size=0.1, random_state=42)
```

Figure 7.4: Train_test_split()

size of the dataset and for the fast and light processing of the images a generator is created for the creation of dataset.

```
image_generator = ImageDataGenerator(samplewise_center = True, samplewise_std_normalization = True, validation_split=val_frac)

train_gen = image_generator.flow_from_directory(path, target_size=target_size, batch_size=batch_size, shuffle=True, subset='training')
val_gen = image_generator.flow_from_directory(path, target_size=target_size, subset='validation')
```

Figure 7.5: Generator

There are 78300 images in the training dataset and 8700 images in the validation dataset.

7.5 Preparation of Image data generator

Images are resized to the shape of (1,64,64,3) , validation frequency is set to 0.1% and the batch size is set to 64.

```
train_gen = image_generator.flow_from_directory(path,
target_size=target_size, batch_size=batch_size, shuffle=True,
subset='training')
val_gen = image_generator.flow_from_directory(path,
target_size=target_size, subset='validation')
```

Figure 7.6: Preparation Of Image

7.6 Implementation of Model

Implementation of ResNet50 requires to define a self contained bottle neck. That would reduce the dimensions in the first two CONV layers and increase it again in the final CONV layer. For example the figure shows 2 residual modules stacked on top of each other. Finally, He

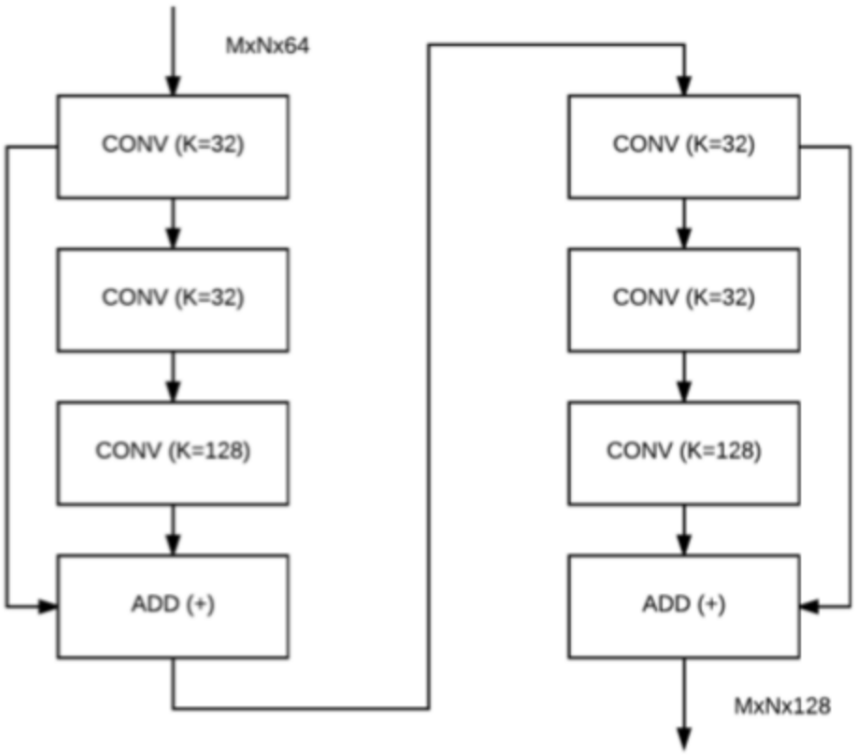


Figure 7.7: Convolution Block

et al.[] published a second paper on the residual module called Identity Mappings in Deep Residual Networks /citeresidual .It provided a per-activated residual model which is a much better version of a residual block; it simplifies the flow of gradients to propagate to any previous block without hindrance. We start with the series of (BN => RELU => CONV) * N layers instead of starting with the convolution weights. The next residual block gets the addition operation of previous residual modules. The overall architecture is similar to figure

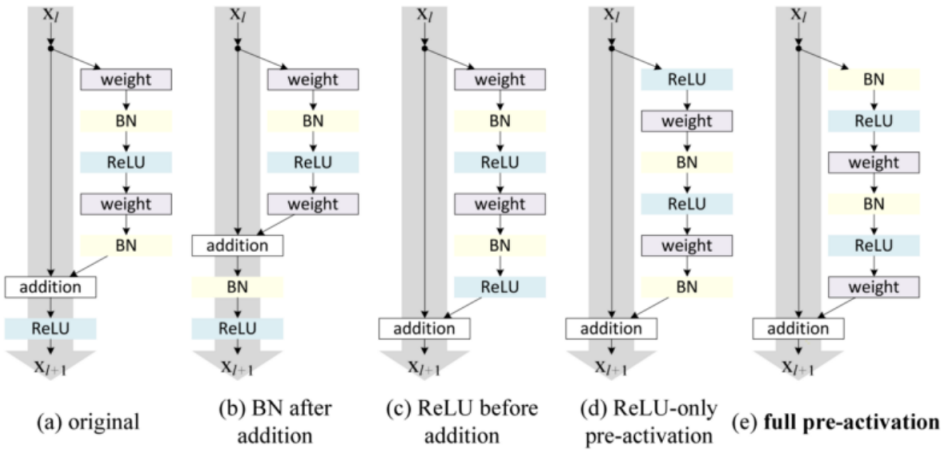


Figure 7.8: Architecture of ResNet

7.7 Defining the identity block

It comprises three identical convolutional layers with a stride size of (1,1) , kernel size of (1,1) and activation function used here is ReLU.

```
def identity_block(X,f,filters, stage, block):
  #defining name basis
  conv_name_base = 'res' +str(stage)+block+'_branch'
  bn_name_base = 'bn' +str(stage)+block+'_branch'

  #Retrive Filters
  F1,F2,F3 = filters

  X_shortcut = X

  X = Conv2D(filters=F1, kernel_size=(1,1), strides = (1,1), padding='valid', name = conv_name_base+'2a',kernel_initializer=glorot_uniform(seed=0))(X)
  X = BatchNormalization(axis = 3, name = bn_name_base+'2a')(X)
  X = Activation('relu')(X)

  X = Conv2D(filters=F2, kernel_size=(f,f), strides = (1,1), padding='same', name = conv_name_base+'2b',kernel_initializer=glorot_uniform(seed=0))(X)
  X = BatchNormalization(axis = 3, name = bn_name_base+'2b')(X)
  X = Activation('relu')(X)

  X = Conv2D(filters=F3, kernel_size=(1,1), strides = (1,1), padding='valid', name = conv_name_base+'2c',kernel_initializer=glorot_uniform(seed=0))(X)
  X = BatchNormalization(axis = 3, name = bn_name_base+'2c')(X)
  X = Add()([X, X_shortcut])
  X = Activation('relu')(X)

  return X
```

Figure 7.9: Identity Block

7.8 Defining convolutional block

Each Convolution block comprises four identical convolutional layers with the stride size of (1,1), kernel size of (1,1) and activation function as ReLU.

```
def identity_block(X,f,filters, stage, block):
  #defining name basis
  conv_name_base = 'res' +str(stage)+block+'_branch'
  bn_name_base = 'bn' +str(stage)+block+'_branch'

  #Retrive Filters
  F1,F2,F3 = filters

  X_shortcut = X

  X = Conv2D(filters=F1, kernel_size=(1,1), strides = (1,1), padding='valid', name = conv_name_base+'2a',kernel_initializer=glorot_uniform(seed=0))(X)
  X = BatchNormalization(axis = 3, name = bn_name_base+'2a')(X)
  X = Activation('relu')(X)

  X = Conv2D(filters=F2, kernel_size=(f,f), strides = (1,1), padding='same', name = conv_name_base+'2b',kernel_initializer=glorot_uniform(seed=0))(X)
  X = BatchNormalization(axis = 3, name = bn_name_base+'2b')(X)
  X = Activation('relu')(X)

  X = Conv2D(filters=F3, kernel_size=(1,1), strides = (1,1), padding='valid', name = conv_name_base+'2c',kernel_initializer=glorot_uniform(seed=0))(X)
  X = BatchNormalization(axis = 3, name = bn_name_base+'2c')(X)
  X = Add()([X, X_shortcut])
  X = Activation('relu')(X)

  return X
```

Figure 7.10: Convolutional Block

7.9 Residual networks

Neural networks are by default design are made to feed data into the layer by layer fashion, form vegetation cell in one layer to the vegetation cell of next layer of neurons it is in contact with. As this approach is beneficial and proved to be working very efficiently for majority of the cases, but there is one problem which was discovered and rectified not long ago and is known as the problem of decay.

The neural networks proved to be an amazing piece of art when it comes to small number of layers. It is simple analogy that if you increase the number of data you want to be digested by the network then you have no other choice than to increase the number of layers involved. This might sound like the problem can be solved by application and incorporation of this approach, but in reality the ground truth is completely different, the neural network has the tendency to loss its efficiency when the number of layers are increased.

Usually when you start to increase the number of layers involved the performance might seem to be in increasing order at first but this only last for a short amount of time. The efficiency of the neural network starts to get plateau at one point and when the number of layers increased further it eventually starts to get deteriorate. This problem of loss in the efficiency of the network due to the increase of the number of layers is known as the degrading problem.

So the above argument concludes that the accuracy of the neural network is not a function of the number of layers and we can also state the fact that shallower neural networks tends to learn better than their deeper counterparts.

This surely seem counter-intuitive but this is what is seen in practice and is popularly known as degradation problem.

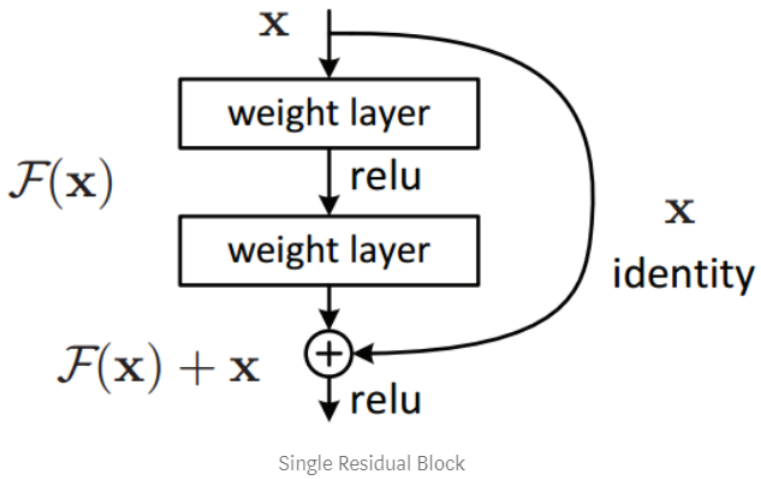


Figure 7.11: Single Residual Block

This problem can be handled by directly targeting what's the main root cause and why this happens? The main reason of this to occur is due to the large number of neural network layers the gradient tends to disappear as we go deeper into the deep neural networks.

The most basic solution behind it is that we can solve it by feeding the output of the neurons not only just to the layer just next to it but to the neurons in the layer which are two to three hops away from it. This architecture tends to provide not only better results but are efficient in training deeper neural networks with large number of layers and with large size of data, all of this without using of any of that extra GPU or TPU power.

Let us understand this with the help of the mathematical equations as follows.

Provided, in a neural network block with input x.Let the true distribution be H(x) and let the denotation of the difference / residual between this as R(x).

$$R(x) = Output - Input = H(x) - x$$

After equating the above equation,

$$H(x) = R(x) + x$$

In this the residual block is trying to fit the true output H(x) and upon close inspection of the equations above,we can note that since we have an identity connection which is induced due to x, the layers are actually trying to learn from identity connection as well as the residual

R(x). So the final deduction from the above equations is nothing but the fact that all layers in a traditional network are learning from the true output H(x) whereas all the layers in a residual network are learning from the residual R(x).

7.10 Defining ResNet50

The input layer shape is set to (1,64,64,3) and the padding is set to be zero in general until explicitly changed.
Stage one, comprises of one convolutional layer with 64 filters and and the kernel size is set to (7,7) , a batch normalization layer and the activation function is set to ReLU and finally a maxpool layer with a window size of (3,3)
Stage two, comprises a convolutional block with 64, 64 and 256 filters and it contains two additional identity blocks with 64,64,256 filters in stage 'a' and stage 'b'.
Stage three, comprises a convolutional block with 128,128 and 512 filters in three convolutional layers respectively. Followed by 3 identical identity blocks each having a filter size of 128,128 and 512 filters in each of the convolutional layers of an identity block.
Stage four, comprises a convolutional block with 256,256 and 1024 filters in each layer of convolution block respectively. Followed by four identity blocks each having 256,256 and 1024 number of filters in each of the convolutional layers respectively.
Stage Five, comprises a convolutional block with 512,512 and 2048 filters in each of the convolutional layers. Followed by two two identity blocks each having 512, 512 and 2048 filters in the convolutional layers of the identity block.
Then, the model after all five stages has an Average pooling layer followed by a flatten layer.
Finally, the model has a dense layer consisting of 29 neurons in the output layer signifying all the 29 classes of classification domain in this context.

7.11 Compilation of Model

Model is compiled with loss set to categorical cross entropy , the optimizer is Adam to reach the saturation and the matrix to check is the accuracy.

```
def ResNet50(input_shape = (64,64,3), classes = 29):
  X_input = Input(input_shape)

  #Zero padding
  X = ZeroPadding2D((3,3))(X_input)

  #stage 1
  X = Conv2D(64,(7,7),strides=(2,2), name = 'conv1', kernel_initializer=glorot_uniform(seed=0))(X)
  X = BatchNormalization(axis=3, name = 'bn_conv1')(X)
  X = Activation('relu')(X)
  X = MaxPooling2D((3,3), strides=(2,2))(X)

  #stage 2
  X = convolutional_block(X, f=3, filters=[64, 64, 256], stage = 2, block='a', s=1)
  X = identity_block(X, 3, [64,64,256], stage=2, block='b')
  X = identity_block(X,3,[64,64,256], stage = 2, block = 'c')

  #stage 3
  X = convolutional_block(X, f=3, filters=[128, 128, 512], stage = 3, block='a', s=2)
  X = identity_block(X, 3, filters=[128, 128, 512], stage=3, block='b')
  X = identity_block(X,3,filters=[128, 128, 512], stage = 3, block = 'c')
  X = identity_block(X,3,filters=[128, 128, 512], stage = 3, block = 'd')

  #stage 4
  X = convolutional_block(X, f=3, filters=[256, 256, 1024], stage = 4, block='a', s=2)
  X = identity_block(X, 3, filters=[256, 256, 1024], stage=4, block='b')
  X = identity_block(X,3,filters=[256, 256, 1024], stage = 4, block = 'c')
  X = identity_block(X,3,filters=[256, 256, 1024], stage = 4, block = 'd')
  X = identity_block(X,3,filters=[256, 256, 1024], stage = 4, block = 'f')

  #stage 5
  X = convolutional_block(X, f=3, filters=[512, 512, 2048], stage = 5, block='a', s=2)
  X = identity_block(X, 3, filters=[512, 512, 2048], stage=5, block='b')
  X = identity_block(X,3,filters=[512, 512, 2048], stage = 5, block = 'c')

  X = AveragePooling2D((2,2), name = 'avg_pool')(X)

  X = Flatten()(X)
  X = Dense(classes, activation='softmax', name='fc'+str(classes), kernel_initializer=glorot_uniform(seed=0))(X)

  model = Model(inputs = X_input, outputs = X, name='ResNet50')

  return model
```

Figure 7.12: Defining ResNet50

```
model.compile(loss='categorical_crossentropy', optimizer='adam', metrics=['accuracy'])

model.summary()
```

Figure 7.13: Compilation

7.12 Training of model on the data set

Training is done using the '.fit' function provided by the tensorflow library. The dataset is passed using a generator for easy computation.

```
history = model.fit_generator(train_gen,epochs=10, validation_data=val_gen)
```

Figure 7.14: Training

CHAPTER 8

EXPERIMENTAL RESULTS

In this section the discussion about the results we have achieved with the above implementation of the model system, with proper assessment and analysis of the input data set, which is ASL (American sign language) data set form kaggel.
This data set is a proper example of how academic data sets are, these type of data set are properly made for the research purposes. The main reason why this type of data set is mainly used for research is the way in which it is being curated. Let us discuss in some more detail.

8.1 Data Set

Our training data set is of academic data set type and is picked from kaggle. The main reason behind selection of this type of data is, as this is mainly made for research purposes and the speciality behind this data set is its properly curated and undergo extensive testing, so that there should not be any margin of error. This precaution is specially taken as the research works usually involved with the medical well being of human beings, hence when human life is on the line even an error of 0.01% can prove to be dangerous.
Contextually, our data set is a collection of images of alphabets in American Sign Language, separated in twenty-nine (29) folders which represent the different classes. The data set contains eighty-seven thousand (87,000) images which are 200x200 pixels. There are twenty-nine (29) classes, of which twenty-six (26) are for the letters A-Z and three (3) classes for NOTHING, DELETE and SPACE. These Nothing, space and delete classes are very helpful in real time applications and classification. The test data set contains minimum twenty-nine (29) images to encourage the use of real world test images.

8.2 RESULTS

The results of out system depends mainly upon the performance evaluation of our model during the time of training it. So to achieve the promised accuracy we have to fit our model for up to ten epoch. Following is the analysis of time and accuracy achieved by out system.

8.2.1 Analysis of accuracy

Epoch Wise details snapshot of our model is given below.

```
Epoch 1/10
1224/1224 [==============================]- 516s 422ms/step- loss: 0.8651- accuracy: 0.7676- val_loss: 0.4164- val_accuracy: 0.8723
Epoch 2/10
1224/1224 [==============================]- 492s 402ms/step- loss: 0.0839- accuracy: 0.9734- val_loss: 25.3385- val_accuracy: 0.0817
Epoch 3/10
1224/1224 [==============================]- 492s 402ms/step- loss: 0.0755- accuracy: 0.9774- val_loss: 0.7623- val_accuracy: 0.8362
Epoch 4/10
1224/1224 [==============================]- 492s 402ms/step- loss: 0.0489- accuracy: 0.9862- val_loss: 1.0171- val_accuracy: 0.8417
Epoch 5/10
1224/1224 [==============================]- 493s 402ms/step- loss: 0.0432- accuracy: 0.9878- val_loss: 0.5040- val_accuracy: 0.9161
Epoch 6/10
1224/1224 [==============================]- 493s 402ms/step- loss: 0.0681- accuracy: 0.9824- val_loss: 0.8043- val_accuracy: 0.8897
Epoch 7/10
1224/1224 [==============================]- 492s 402ms/step- loss: 0.0138- accuracy: 0.9962- val_loss: 0.6406- val_accuracy: 0.8941
Epoch 8/10
1224/1224 [==============================]- 492s 402ms/step- loss: 0.0399- accuracy: 0.9892- val_loss: 0.2183- val_accuracy: 0.9575
Epoch 9/10
1224/1224 [==============================]- 493s 403ms/step- loss: 0.0200- accuracy: 0.9940- val_loss: 0.2190- val_accuracy: 0.9464
Epoch 10/10
1224/1224 [==============================]- 492s 402ms/step- loss: 0.0393- accuracy: 0.9908- val_loss: 1.9646- val_accuracy: 0.9807
```

Figure 8.1: Model fit analysis

8.2.2 Statistics analysis

As we can see we have fairly achieved a respective accuracy over our data set model.
The above accuracy graphs depicts a clear view of how our model is performing per epoch. We see a instant hike in the accuracy at first epoch as this is the time our model is introduces to the data set and this shows the model is quick to adapt the data set. After this there is a gradual increase in the system from first epoch to third epoch of our validation set as compared to training data set. Now there is a gradual dip in the accuracy as we proceed towards the further iterations. The model has peaked at third iteration of the epoch. This descend in the

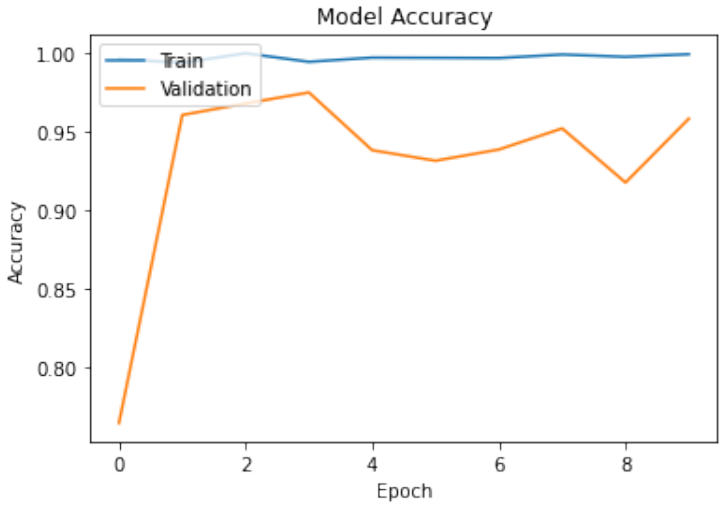


Figure 8.2: Accuracy graph

validation continues up till fifth epoch. From there on wards there is again a gradual increase in the accuracy, but this increase only lasts up to seventh epoch. Finally we see a sudden dip and a sudden hike after that till the last tenth epoch.

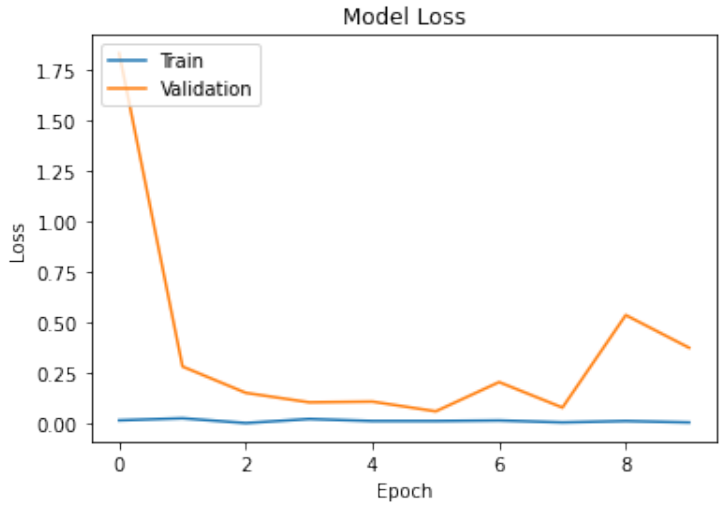


Figure 8.3: Loss graph

This loss graph provides a clear interpretation of the model loss, as we proceed over each and every epoch iteration. A quick overview, loss function is use to tell the difference between the model output and and the output we are expecting from it. Hence out model starts with a loss of 0.25 at its first epoch. After this the loss plateaued for the majority of the iterations. There is a sudden increase and a dip is seen in the last epoch and the last second epoch. There is a increase in the difference for the seventh epoch to ninth epoch and then it dips at final tenth epoch.

The final accuracy obtained is 98%

8.3 Sample Output

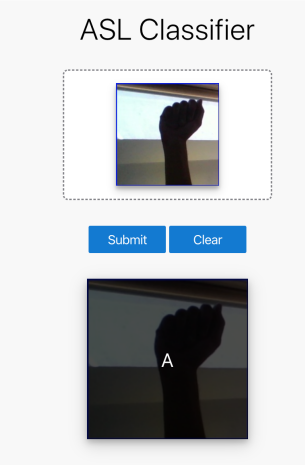


Figure 8.4: 'A' Sample Result

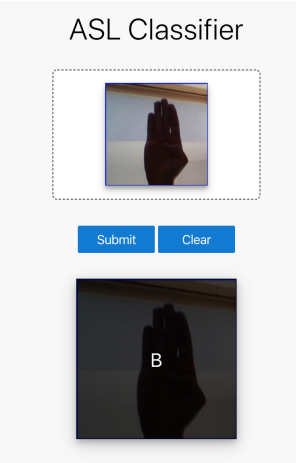


Figure 8.5: 'B' Sample Result

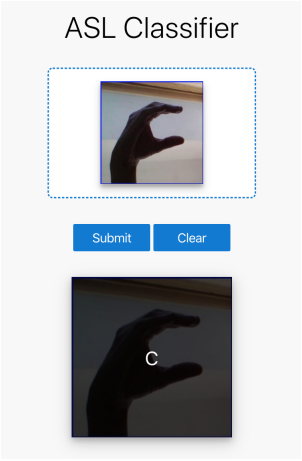


Figure 8.6: 'C' Sample Result

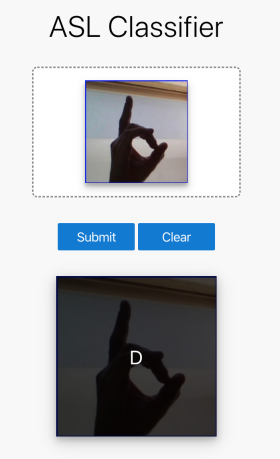


Figure 8.7: 'D' Sample Result

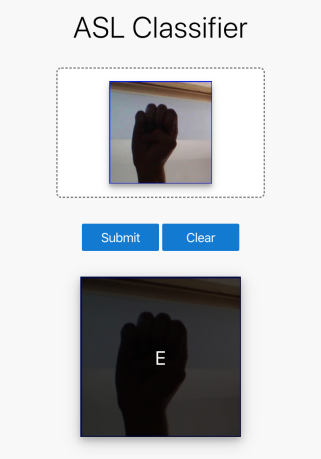


Figure 8.8: 'E' Sample Result

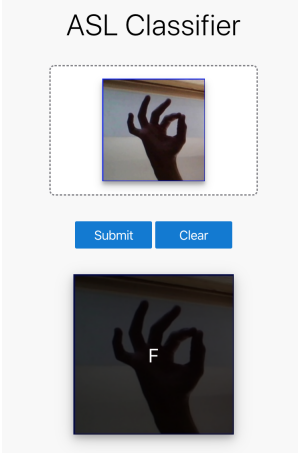


Figure 8.9: 'F' Sample Result

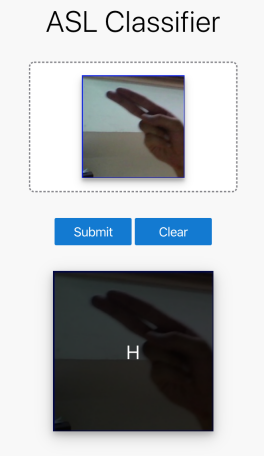


Figure 8.10: 'H' Sample Result

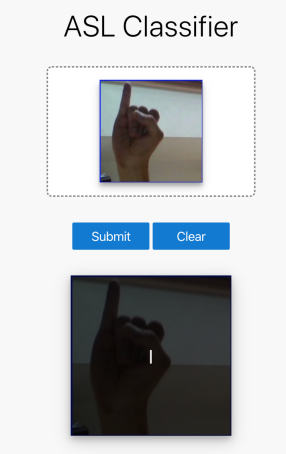


Figure 8.11: 'I' Sample Result

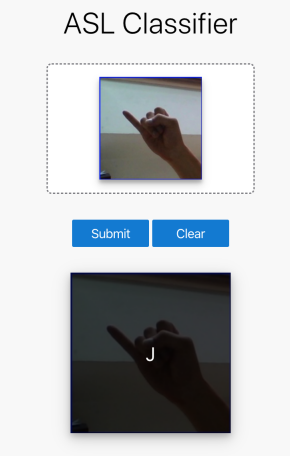


Figure 8.12: 'J' Sample Result

CHAPTER 9

CONCLUSION AND FUTURE WORKS

9.1 Conclusion

In this current era of technology, gesture is considered as antediluvian technique for conversation among human being. Now there already exists a wide verity of applications which are running directly or indirectly on computer systems which are processing upon wide verity of inputs ranging from inputs directly incorporated by the help of direct physical touches or inputs which are not taking help of any physical interaction at all and are fully dependent upon the gestures or actions which are preset to perform a particular action or set of tasks in case of robots.

Technologies involved in this field of gesture recognition has a very large impact on the industry of robotics in which robots control the virtual face recognition, computer applications like performing a particular task upon receiving a particular hand gesture or even controlling computer through hand gestures, artificial intelligence etc. They prove to be very helpful specially in case of physically challenged people as this mode of interaction is preferred more according to their needs.

The current system which we have implemented is having a user friendly interaction mode which is contrarian to most of the command line based systems and is having a descent prediction rate as we have encountered in former sections.

9.2 Future Works

Now after this we have to further work upon tackling one of the most common problem which is not only relevant in this model but equally affects all the systems which are dependent upon the pictures of hands for its training, and that is its high dependency upon background of the users hand skin color and the quality of the background in which the picture has been taken. This can be achieved by revamping the current set of algorithms for more robust and advance

set, as well as incorporating the HU set of moment invariant as an added layer of descriptors for the features of hand gestures. Addition of this added layer of descriptors on the feature set helps greatly in achieving a high accuracy and good performance of our model. But for this we have to change our current architecture of classifiers and approach with new set mechanisms.
We can also add more number of classes into our model for better understanding of the current language scenarios set as well as we can even increase the current scope of the languages it can understand.
The physical device which is currently being used can be improved too. Instead of taking pictures through the webcam, we can make use of devices with infrared capabilities as this will highly remove the dependency of background light and the analysis can be done in the three dimensional space with high accuracy and low failing rate ex. Kinetic.

REFERENCES

[1] Cao, J. and Chang, C.-Y. (2017). "Machine learning in nd signal processing." *Multidimensional Systems and Signal Processing*, 28(3), 791–793.

[2] Chaudhary, A., Raheja, J. L., Das, K., and Raheja, S. (2011). "A survey on hand gesture recognition in context of soft computing." *Communications in Computer and Information Science Advanced Computing*, 46–55.

[3] Dabre, K. and Dholay, S. (2014). "Machine learning model for sign language interpretation using webcam images." *2014 International Conference on Circuits, Systems, Communication and Information Technology Applications (CSCITA)*.

[4] De, O., Deb, P., Mukherjee, S., Nandy, S., Chakraborty, T., and Saha, S. (2016). "Computer vision based framework for digit recognition by hand gesture analysis." *2016 IEEE 7th Annual Information Technology, Electronics and Mobile Communication Conference (IEMCON)*.

[5] Ghotkar, A. and Kharate, G. (2015). "Dynamic hand gesture recognition for sign words and novel sentence interpretation algorithm for indian sign language using microsoft kinect sensor." *Journal of Pattern Recognition Research*, 10(1), 24–38.

[6] Goodfellow, I., Bengio, Y., and Courville, A. (2016). *Deep learning*. MIT press.

[7] He, K., Zhang, X., Ren, S., and Sun, J. (2016). "Deep residual learning for image recognition." *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

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

[9] Sharma, S., Jain, S., and Khushboo (2019). "A static hand gesture and face recognition system for blind people." *2019 6th International Conference on Signal Processing and Integrated Networks (SPIN)*.

[10] Sonkusare, J. S., Chopade, N. B., Sor, R., and Tade, S. L. (2015). "A review on hand gesture recognition system." *2015 International Conference on Computing Communication Control and Automation*.

[11] Wilcox, S. (2003). "The multimedia dictionary of american sign language: Learning lessons about language, technology, and business." *Sign Language Studies*, 3(4), 379–392.

[12] Wu, C., Yan, Y., Cao, Q., Fei, F., Yang, D., and Song, A. (2018). "A low cost surface emg sensor network for hand motion recognition." *2018 IEEE 1st International Conference on Micro/Nano Sensors for AI, Healthcare, and Robotics (NSENS)*.

[13] Wu, Y., Jiang, D., Duan, J., Liu, X., Bayford, R., and Demosthenous, A. (2018). "Towards a high accuracy wearable hand gesture recognition system using eit." *2018 IEEE International Symposium on Circuits and Systems (ISCAS)*.