

YOGA POSE DETECTION

Mini Project submitted in partial fulfillment of the requirements for the award of

the degree of

BACHELOR OF TECHNOLOGY

IN

COMPUTER SCIENCE AND ENGINEERING

Submitted by

Dhruv Kumar Patwari (221710302021)

Mekala Muralidhar (221710302037)

Parimi Siri Chandana (221710302048)

P Sai Charan (221710302047)

Under the esteemed guidance of

Mr. Arif Mohammad Abdul



DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

GITAM

(Deemed to be University)

HYDERABAD

DECEMBER 2020

**DEPARTMENT OF COMPUTER SCIENCE AND
ENGINEERING
GITAM**

(Deemed to be University)



DECLARATION

We hereby declare that the mini project entitled “YOGA POSE DETECTION” is an original work done in the Department of Computer Science and Engineering, GITAM School of Technology, GITAM (Deemed to be University) submitted in partial fulfillment of the requirements for the award of the degree of B.Tech. in Computer Science and Engineering. The work has not been submitted to any other college or University for the award of any degree or diploma.

Date: 10 December 2020

**Dhruv Kumar Patwari (221710302021)
Mekala Muralidhar (221710302037)
Parimi Siri Chandana (221710302048)
P Sai Charan (221710302047)**

**DEPARTMENT OF COMPUTER SCIENCE AND
ENGINEERING**

GITAM

(Deemed to be University)



CERTIFICATE

This is to certify that Mini Project entitled “YOGA POSE DETECTION” is submitted by Dhruv Kumar Patwari (221710302021), Mekala Muralidhar (221710302037), Parimi Siri Chandana (221710302048), P Sai Charan (221710302047) in partial fulfillment of the requirements for the award of degree of Bachelor of Technology in Computer Science and Engineering. The Mini Project has been approved as it satisfies the academic requirements.

**Mr. Arif Mohammad
Abdul
Assistant Professor
Dept. of Computer
Science and Engineering**

ACKNOWLEDGMENT

Our Summer Internship would not have been successful without the help of several people. we would like to thank the personalities who were part of our seminar in numerous ways, those who gave us outstanding support from the birth of the seminar.

We are extremely thankful to our honorable Pro-Vice Chancellor, **Prof. N. Siva Prasad** for providing necessary infrastructure and resources for the accomplishment of our seminar.

We are highly indebted to **Prof. N. Seetharamaiah**, Principal, School of Technology, for his support during the tenure of the seminar.

We are very much obliged to our beloved **Prof. S. Phani Kumar**, Head of the Department of Computer Science & Engineering for providing the opportunity to undertake this seminar and encouragement in completion of this seminar.

We hereby wish to express our deep sense of gratitude to **Dr. S Aparna**, Assistant Professor, Department of Computer Science and Engineering, School of Technology and to **Mr. Arif Mohammad Abdul**, Assistant Professor, Department of Computer Science and Engineering, School of Technology for the esteemed guidance, moral support and invaluable advice provided by them for the success of the summer Internship.

We are also thankful to all the staff members of Computer Science and Engineering department who have cooperated in making our seminar a success. We would like to thank all our parents and friends who extended their help, encouragement and moral support either directly or indirectly in our seminar work.

Sincerely,

Dhruv Kumar Patwari (221710302021)
Mekala Muralidhar (221710302037)
Parimi Siri Chandana (221710302048)
P Sai Charan (221710302047)

ABSTRACT

Amid the raise in COVID-19 cases around the world, all the classes are forced to move online. Yoga classes are no exception. But correcting someone's yoga pose through video chat is not an easy task. Moreover, if that person continues to follow the same wrong posture, he/she is at a risk of serious injury.

Practice of yoga helps to balance both body and mind through a series of physical postures called asanas, breathing control and meditation. A computer-assisted self-learning system can be developed to help such elders, though improper training and the postures associated with it may harm the body's muscles and ligaments. To have a flawless system it is essential to classify asanas, and identify the one the practitioner is currently practicing, following which the system can offer the guidance necessary.

Building on the base paper we hope to make a complete system which will be able to let the user know if their yoga pose is correct. This can be used by yoga institutes to help their students during the at-home yoga classes.

TABLE OF CONTENTS

1. INTRODUCTION	1
1.1 BENEFITS OF PHYSICAL ACTIVITY	1
1.2 MOTIVATION	2
1.3 APPLICATIONS	2
1.4 ADVANTAGES.....	2
2. MACHINE LEARNING	3
2.1 INTRODUCTION.....	3
2.2 IMPORTANCE OF MACHINE LEARNING.....	3
2.3 USES OF MACHINE LEARNING.....	5
2.3.1 Virtual Personal Assistants	5
2.3.2 Product Recommendations	7
2.3.3 Online Fraud Detection.....	7
2.4 TYPES OF MACHINE LEARNING	8
2.4.1 Supervised Learning.....	8
2.4.2 Unsupervised Learning.....	9
2.4.3 Reinforcement Learning	9
2.4.4 Transfer Learning	10
3. DEEP LEARNING	12
3.1 INTRODUCTION.....	12
3.2 USES OF DEEP LEARNING	13
3.2.1 Translations	13
3.2.2 Adding Color To Black-And-White Images And Videos	13
3.2.3 Language Recognition.....	13
3.2.4 Autonomous Vehicles.....	14
3.2.5 Computer Vision.....	14
3.2.6 Text Generation	14
3.2.7 Deep-Learning Robots.....	14
3.3 RELATION BETWEEN DATA MINING, MACHINE LEARNING AND DEEP LEARNING	15
4. JAVASCRIPT.....	18
4.1 INTRODUCTION.....	18
4.2 CLIENT-SIDE JAVASCRIPT.....	18
4.3 ADVANTAGES OF JAVASCRIPT.....	19

4.4 LIMITATIONS OF JAVASCRIPT	19
4.5 P5.JS	20
4.5.1 Preferred Editors	20
4.5.2 Difference Between P5.js And JavaScript.....	20
4.5.3 Setting up p5.js	20
4.5.4 Environment	22
5. TENSORFLOW	24
5.1 INTRODUCTION.....	24
5.2 ARCHITECTURE	24
5.3 COMPONENTS OF TENSORFLOW	25
5.3.1 Tensor	25
5.3.2 Graphs.....	26
5.4 ABOUT TENSORFLOW.JS	26
5.4.1 Getting Started.....	26
5.4.2 CoreAPI.....	28
5.4.3 LayersAPI.....	29
5.4.4 Client-Side Superpowers	31
5.4.5 Server-Side Features.....	31
6. ML5.JS	33
6.1 INTRODUCTION.....	33
6.2 IMPLEMENTATION	33
7. POSENET	36
7.1 KEYWORD DEFINITION.....	36
7.2 POSE ESTIMATION.....	37
7.3 SINGLE-PERSON POSE ESTIMATION.....	38
7.3.1 Inputs	38
7.3.2 Outputs.....	39
8. DESIGN AND IMPLEMENTATION	42
8.1 ARCHITECTURE	42
8.2 DATA COLLECTION.....	42
8.2.1 Normalization	45
8.3 MODEL TRAINING	51
8.3.1 Neural Network	52
8.3.2 Activation Functions.....	54

8.3.3 Classifier Model Breakdown	59
8.4 UML MODELS.....	63
8.4.1 Use-Case Diagram	64
9. SYSTEM TESTING	66
9.1 INTRODUCTION TO TESTING.....	66
9.2 TESTING STRATEGIES	66
9.2.1 What is Verified in System Testing?.....	66
9.3 SOFTWARE TESTING HIERARCHY	67
9.4 DIFFERENT TYPES OF SYSTEM TESTING	68
9.5 WHAT TYPES OF SYSTEM TESTING SHOULD TESTERS USE?.....	69
10. RESULTS	70
CONCLUSION	72
REFERENCES	73

LIST OF FIGURES

Fig 2.1 Usage Of Machine Learning In Different Fields.....	4
Fig 2. 2 Uses of Machine learning	8
Fig 2. 3 Types of ML	8
Fig 3. 1 Deep Neural Network.....	13
Fig 3. 2 The Deep Learning Process	15
Fig 3. 3 Relation Between AI, ML, DL	16
Fig 3. 4 Process In Machine Learning And Deep Learning	16
Fig 4. 1 Linking P5.js Min File.....	21
Fig 4. 2 Linking P5.js Using URL	21
Fig 4. 3 Sample HTML Page	22
Fig 4. 4 P5.js Code Structure	23
Fig 5. 1 Importing Tensorflow.js Using Script Tag.....	27
Fig 5. 2 Adding Tensorflow.js using Yarn	27
Fig 5. 3 Adding Tensorflow.js using npm	27
Fig 5. 4 Importing Tensorflow.js	27
Fig 5. 5 Tensor	28
Fig 6. 1 HTML Code For Importing ML5.js.....	34
Fig 6. 2 The CDN link to ML5.js.....	34
Fig 6. 3 Output Of The Above Code.....	34
Fig 6. 4 ML5.JS Supports	35
Fig 7. 1 Posenet Output Keypoints.....	37
Fig 7. 2 Single Pose Estimation Example Code	39
Fig 7. 3 Example Output.....	40
Fig 7. 4 Single Pose Detection Algorithm	41
Fig 7. 5 Flow of the Current Project	41
Fig 8. 1 Project Architecture.....	42
Fig 8. 2 Data Collection Example.....	43
Fig 8. 3 Seventeen Key-Points Returned	44
Fig 8. 4 Section Of JSON File	45
Fig 8. 5 Summary of normalization techniques	45
Fig 8. 6 Scaling To A Range Formula	46
Fig 8. 7 Comparing A Raw Distribution Its Clipped Version	47
Fig 8. 8 Log Scaling Formula	47
Fig 8. 9 Comparing A Raw Distribution To Its Log.....	48
Fig 8. 10 Comparing A Raw Distribution To Its Z-Score Distribution	49
Fig 8. 11 Training On Un-Normalized Data.....	50
Fig 8. 12 Training on Normalized Data	51
Fig 8. 13 Classifier Model Architecture	52
Fig 8. 14 Sample Neural Network Architecture	53
Fig 8. 15 Sigmoid Function Formula.....	55
Fig 8. 16 Tanh Function Formula	56
Fig 8. 17 Softmax Function Formula.....	56
Fig 8. 18 Softsign Function Formula	57
Fig 8. 19 ReLU Function Formula.....	57

Fig 8. 20 ELU Representation	58
Fig 8. 21 Derivative or Gradient of ELU	59
Fig 8. 22 Layer Breakbown	59
Fig 8. 23 Classifier Model Sample JSON Output.....	60
Fig 8. 24 Pose Classified as Mountain.....	60
Fig 8. 25 Pose Classified as Warrior 2.....	61
Fig 8. 26 Pose Classified as Warrior 1.....	61
Fig 8. 27 Pose Classified as Downward Dog.....	62
Fig 8. 28 Pose Classified as Chair	62
Fig 8. 29 Pose Classified as Tree	63
Fig 8. 30 Use Case Diagram	64
Fig 8. 31 Sequence Diagram	65
Fig 9. 1 Flow Chart For Testing.....	67
Fig 10. 1 Project Flow.....,,,	70
Fig 10. 2 Landing Page	70
Fig 10. 3 Practice Page.....	71
Fig 10. 4 Congratulations Message.....	71

CHAPTER 1

INTRODUCTION

The COVID-19 pandemic is an unprecedented time all across the world. Worldwide, extensive social distancing policies are put into place, restricting people's daily activities and worldwide pleas from governments asking people to stay safe and stay at home. This, of course, means that most people will spend much of their time (if not all) at home.

These social distancing measures mean that people have far fewer opportunities to be physically active, especially if activities such as walking or cycling as transportation, or taking part in a leisurely activity (jogging, walking the dog, going to the gym) are being restricted. Furthermore, these drastic measures also make it so much easier to be sedentary at home for long periods. The impact of this physical inactivity may very likely be seen in many areas such as health and social care and the mental well-being of people all across the globe.

Although these social distancing measures are essential and needed in a time such as now, our bodies and minds still need physical activity and the many benefits thereof.

1.1 BENEFITS OF PHYSICAL ACTIVITY

There are many benefits of physical activity. These include Strengthening and maintaining your immune system strength - being less susceptible to infections, Reduces high blood pressure, Weight management, Reduces the risk of heart disease, reduces the risk of diabetes, reduces the risk of stroke, reduces the risk of certain cancers, Improves bone and muscle strength, Improves balance, Improves flexibility, Improves fitness, Improves mental health, Reduces the risk of depression, Reduces the risk of cognitive decline, Delays the onset of dementia, Improves overall feeling of well-being. In children, physical activity may, support healthy growth and development, reduce the risk of disease in later life, help in the development of fundamental movement skills.

1.2 MOTIVATION

Due to COVID-19 induced lockdown all over the world, fitness enthusiasts, wellness coaches, yoga instructors and everyone else turned to online classes to connect and continue training. But an online course isn't ideal for a yoga class for someone just starting with their practice. This might lead to practicing yoga with the wrong form. If this wrong form persists, then it increases the risk of injuries.

Having a tool which can be used in addition to the class which tells the student if their form is correct, will help students practice yoga with the right shape. This tool will be handy when a particular yoga class that takes place online has many students.

1.3 APPLICATIONS

This tool can be used for any fitness related tasks. Here it's used in the yoga domain, but the same model can easily be trained in a different field to estimate the pose. This can help an individual check if their form in a particular activity is correct.

Some of the possible applications include Checking the form in weight lifting, circuit training, cardio, yoga, cricket, golf, and many more such pose related activities can be checked using the model.

1.4 ADVANTAGES

Majority of the pose estimation software require a lot of hardware setup to be able to use their services. The present solution is run completely on the browser, which means that the user can access the solution from their laptops or phones, whenever they need it. This is also cheap to install and almost no friction in getting started.

The user can also train the model for their use-case if the existing asanas aren't to their liking. This also provides a fully customized model for the user.

CHAPTER 2

MACHINE LEARNING

2.1 INTRODUCTION

Over the past two decades Machine Learning has become one of the mainstays of information technology and with that, a rather central, albeit usually hidden, part of our life. With the ever-increasing amounts of data becoming available there is good reason to believe that smart data analysis will become even more pervasive as a necessary ingredient for technological progress.

Human designers often produce machines that do not work as well as desired in the environments in which they are used. In fact, certain characteristics of the working environment might not be completely known at design time. Machine learning methods can be used for on-the- job improvement of existing machine designs. The amount of knowledge available about certain tasks might be too large for explicit encoding by humans. Machines that learn this knowledge gradually might be able to capture more of it than humans would want to write down. Environments change over time. Machines that can adapt to a changing environment would reduce the need for constant redesign. New knowledge about tasks is constantly being discovered by humans. Vocabulary changes. There is a constant stream of new events in the world. Continuing redesign of AI systems to conform to new knowledge is impractical, but machine learning methods might be able to track much of it.

2.2 IMPORTANCE OF MACHINE LEARNING

Machine learning is a branch of artificial intelligence that aims at enabling machines to perform their jobs skillfully by using intelligent software. The statistical learning methods constitute the backbone of intelligent software that is used to develop machine intelligence. Because machine learning algorithms require data to learn, the discipline must have connection with the discipline of database. Similarly, there are familiar terms such as Knowledge Discovery from Data (KDD), data mining, and pattern recognition. One wonders how to view the big picture in which such connection is

illustrated.

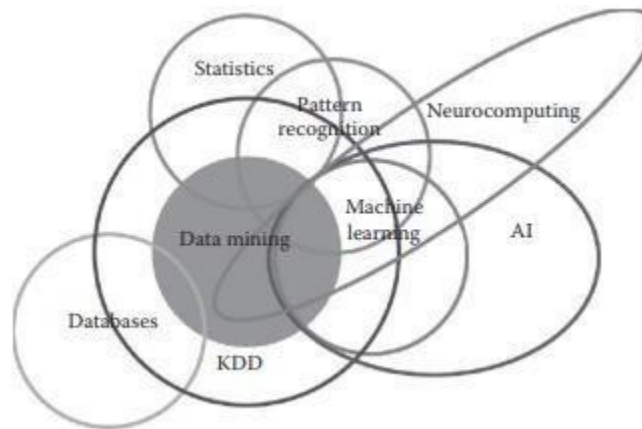


Fig 2.1 Usage Of Machine Learning In Different Fields

There are some tasks that humans perform effortlessly or with some efforts, but we are unable to explain how we perform them. For example, we can recognize the speech of our friends without much difficulty. If we are asked how we recognize the voices, the answer is very difficult for us to explain. Because of the lack of understanding of such phenomenon (speech recognition in this case), we cannot craft algorithms for such scenarios. Machine learning algorithms are helpful in bridging this gap of understanding.

The idea is very simple. We are not targeting to understand the underlying processes that help us learn. We write computer programs that will make machines learn and enable them to perform tasks, such as prediction. The goal of learning is to construct a model that takes the input and produces the desired result. Sometimes, we can understand the model, whereas, at other times, it can also be like a black box for us, the working of which cannot be intuitively explained. The model can be considered as an approximation of the process we want machines to mimic. In such a situation, it is possible that we obtain errors for some input, but most of the time, the model provides correct answers. Hence, another measure of performance (besides performance of metrics of speed and memory usage) of a machine learning algorithm will be the accuracy of results.

2.3 USES OF MACHINE LEARNING

Artificial Intelligence (AI) is everywhere. Possibility is that you are using it in one way or the other and you don't even know about it. One of the popular applications of AI is Machine Learning (ML), in which computers, software, and devices perform via cognition (very similar to human brain). Herein, we share few examples of machine learning that we use everyday and perhaps have no idea that they are driven by ML. These are some the uses and applications of ML

2.3.1 Virtual Personal Assistants

Siri, Alexa, Google Now are some of the popular examples of virtual personal assistants. As the name suggests, they assist in finding information, when asked over voice. All you need to do is activate them and ask “What is my schedule for today?”, “What are the flights from Germany to London”, or similar questions. For answering, your personal assistant looks out for the information, recalls your related queries, or send a command to other resources (like phone apps) to collect info. You can even instruct assistants for certain tasks like “Set an alarm for 6 AM next morning”, “Remind me to visit Visa Office day after tomorrow”.

Machine learning is an important part of these personal assistants as they collect and refine the information on the basis of your previous involvement with them. Later, this set of data utilized to render results that are tailored to your preferences.

Virtual Assistants are integrated to a variety of platforms. For example, Smart Speakers: Amazon Echo and Google Home, Smartphones: Samsung Bixby on Samsung S8, Mobile Apps: Google Allo.

2.3.1.1 Predictions While Commuting

Traffic Predictions: We all have been using GPS navigation services. While we do that, our current locations and velocities are being saved at a central server for managing traffic. This data is then used to build a map of current traffic. While this helps in preventing the traffic and does congestion analysis, the underlying problem is that there

are less number of cars that are equipped with GPS. Machine learning in such scenarios helps to estimate the regions where congestion can be found on the basis of daily experiences.

Online Transportation Networks: When booking a cab, the app estimates the price of the ride. When sharing these services, how do they minimize the detours? The answer is machine learning. Jeff Schneider, the engineering lead at Uber ATC reveals in a an interview that they use ML to define price surge hours by predicting the rider demand. In the entire cycle of the services, ML is playing a major role.

2.3.1.2 Social Media Services

From personalizing your news feed to better ads targeting, social media platforms are utilizing machine learning for their own and user benefits. Here are a few examples that you must be noticing, using, and loving in your social media accounts, without realizing that these wonderful features are nothing but the applications of ML.

People You May Know: Machine learning works on a simple concept: understanding with experiences. Facebook continuously notices the friends that you connect with, the profiles that you visit very often, your interests, workplace, or a group that you share with someone etc. On the basis of continuous learning, a list of Facebook users are suggested that you can become friends with.

Face Recognition: You upload a picture of you with a friend and Facebook instantly recognizes that friend. Facebook checks the poses and projections in the picture, notice the unique features, and then match them with the people in your friend list. The entire process at the backend is complicated and takes care of the precision factor but seems to be a simple application of ML at the front end.

Similar Pins: Machine learning is the core element of Computer Vision, which is a technique to extract useful information from images and videos. Pinterest uses computer vision to identify the objects (or pins) in the images and recommend similar pins accordingly.

2.3.1.3 Search Engine Result Refining

Google and other search engines use machine learning to improve the search results for you. Every time you execute a search, the algorithms at the backend keep a watch at how you respond to the results. If you open the top results and stay on the web page for long, the search engine assumes that the results it displayed were in accordance to the query. Similarly, if you reach the second or third page of the search results but do not open any of the results, the search engine estimates that the results served did not match requirement. This way, the algorithms working at the backend improve the search results.

2.3.2 Product Recommendations

You shopped for a product online few days back and then you keep receiving emails for shopping suggestions. If not this, then you might have noticed that the shopping website or the app recommends you some items that somehow matches with your taste. On the basis of your behavior with the website/app, past purchases, items liked or added to cart, brand preferences etc., the product recommendations are made.

2.3.3 Online Fraud Detection

Machine learning is proving its potential to make cyberspace a secure place and tracking monetary frauds online is one of its examples. For example: Paypal is using ML for protection against money laundering. The company uses a set of tools that helps them to compare millions of transactions taking place and distinguish between legitimate or illegitimate transactions taking place between the buyers and sellers.



Fig 2. 2 Uses of Machine learning

2.4 TYPES OF MACHINE LEARNING

There are 3 types of Machine learning which are widely used in today's world. These are as follows.

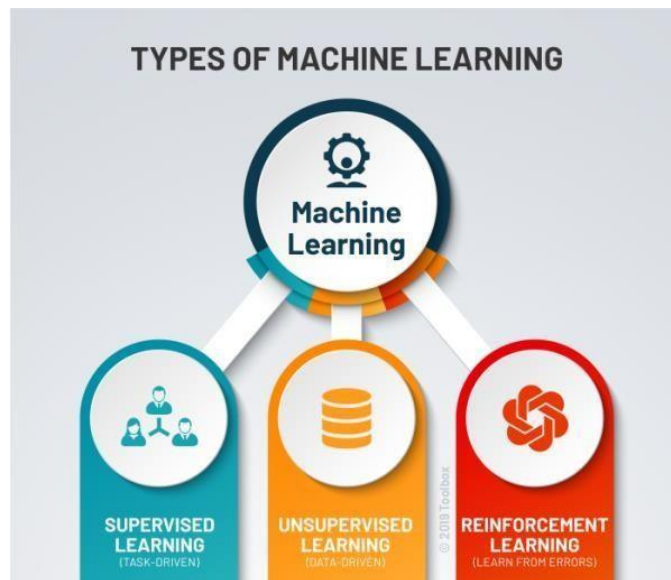


Fig 2. 3 Types of ML

2.4.1 Supervised Learning

Supervised learning is one of the most basic types of machine learning. In this type, the machine learning algorithm is trained on labeled data. Even though the data

needs to be labeled accurately for this method to work, supervised learning is extremely powerful when used in the right circumstances. In supervised learning, the ML algorithm is given a small training dataset to work with. This training dataset is a smaller part of the bigger dataset and serves to give the algorithm a basic idea of the problem, solution, and data points to be dealt with. The training dataset is also very similar to the final dataset in its characteristics and provides the algorithm with the labeled parameters required for the problem. The algorithm then finds relationships between the parameters given, essentially establishing a cause and effect relationship between the variables in the dataset. At the end of the training, the algorithm has an idea of how the data works and the relationship between the input and the output.

2.4.2 Unsupervised Learning

Unsupervised machine learning holds the advantage of being able to work with unlabeled data. This means that human labor is not required to make the dataset machine-readable, allowing much larger datasets to be worked on by the program. In supervised learning, the labels allow the algorithm to find the exact nature of the relationship between any two data points. However, unsupervised learning does not have labels to work off of, resulting in the creation of hidden structures. Relationships between data points are perceived by the algorithm in an abstract manner, with no input required from human beings. The creation of these hidden structures is what makes unsupervised learning algorithms versatile. Instead of a defined and set problem statement, unsupervised learning algorithms can adapt to the data by dynamically changing hidden structures. This offers more post-deployment development than supervised learning algorithms.

2.4.3 Reinforcement Learning

It directly takes inspiration from how human beings learn from data in their lives. It features an algorithm that improves upon itself and learns from new situations using a trial-and-error method. Favorable outputs are encouraged or ‘reinforced’, and non-favorable outputs are discouraged or ‘punished’. Based on the psychological concept of

conditioning, reinforcement learning works by putting the algorithm in a work environment with an interpreter and a reward system. In every iteration of the algorithm, the output result is given to the interpreter, which decides whether the outcome is favorable or not. In case of the program finding the correct solution, the interpreter reinforces the solution by providing a reward to the algorithm. If the outcome is not favorable, the algorithm is forced to reiterate until it finds a better result. In most cases, the reward system is directly tied to the effectiveness of the result. In typical reinforcement learning use-cases, such as finding the shortest route between two points on a map, the solution is not an absolute value. Instead, it takes on a score of effectiveness, expressed in a percentage value. The higher this percentage value is, the more reward is given to the algorithm. Thus, the program is trained to give the best possible solution for the best possible reward.

2.4.4 Transfer Learning

Transfer learning is the ability to combine a pre-trained model with custom training data. What this means is that you can leverage the functionality of a model and add your own samples without having to create everything from scratch.

For example, an algorithm has been trained with thousands of images to create an image classification model, and instead of creating your own, transfer learning allows you to combine new custom image samples with the pre-trained model to create a new image classifier. This feature makes it really fast and easy to have a more customized classifier.

2.4.4.1 Pre-Trained Models

There are a number of machine learning (ML) models that have already been trained by people all over the world and wrapped in an easy to use class. These are often based on the latest research being published. This is useful as we can focus on solving the problem rather than being fixated on how to train a model and collection terabytes of data for the same. Advantages of a pre-trained model are as follows.

No need to gather training data yourself - this can be very time consuming and costly to have data in the correct format and labelled so that a machine learning system can use it to learn from.

Rapidly prototype an idea with lower cost and time - often a pre-trained model may be good enough to do what you need to do so there is no point reinventing the wheel allowing you to concentrate on using the knowledge provided by the model to then implement that creative idea you had.

Use state of the art research - often these models are based on popular research so gives you exposure to such models and understand their performance in the real world.

Typically, easier to use and well documented due to the popularity of such models.

Some pre-trained models offer transfer learning capabilities. This essentially is the practice of transferring information learnt from one machine learning task to another similar example. For example, a model that was originally trained to recognize cats, could be retrained to recognize dogs, if you gave it new training data. This is often faster to train as you are not starting with a blank canvas. The model can use what it has already learnt to recognize cats to then recognize the new thing - dogs have eyes and ears too after all, so if it already knows how to find those features, we are halfway there. Re-train the model on your own data in a much faster way.

CHAPTER 3

DEEP LEARNING

3.1 INTRODUCTION

Deep learning algorithms run data through several “layers” of neural network algorithms, each of which passes a simplified representation of the data to the next layer. Most machine learning algorithms work well on datasets that have up to a few hundred features, or columns.

Basically deep learning is itself a subset of machine learning but in this case the machine learns in a way in which humans are supposed to learn. The structure of deep learning model is highly similar to a human brain with large number of neurons and nodes like neurons in human brain thus resulting in artificial neural network. In applying traditional machine learning algorithms we have to manually select input features from complex data set and then train them which becomes a very tedious job for ML scientist but in neural networks we don't have to manually select useful input features, there are various layers of neural networks for handling complexity of the data set and algorithm as well. In my recent project on human activity recognition , when we applied traditional machine learning algorithm like K-NN then we have to separately detect human and its activity also had to select impactful input parameters manually which became a very tedious task as data set was way too complex but the complexity dramatically reduced on applying artificial neural network, such is the power of deep learning. Yes it's correct that deep learning algorithms take lots of time for training sometimes even weeks as well but its execution on new data is so fast that its not even comparable with traditional ML algorithms. Deep learning has enabled Industrial Experts to overcome challenges which were impossible, a decades ago like Speech and Image recognition and Natural Language Processing. Majority of the Industries are currently depending on it , be it Journalism, Entertainment, Online Retail Store, Automobile, Banking and Finance, Healthcare, Manufacturing or even Digital Sector. Video recommendations, Mail Services, Self Driving cars, Intelligent Chat bots, Voice Assistants are just trending achievements of Deep Learning.

Furthermore, Deep learning can most profoundly be considered as future of Artificial. Intelligence due to constant rapid increase in amount of data as well as the gradual development in hardware field as well, resulting in better computational power.

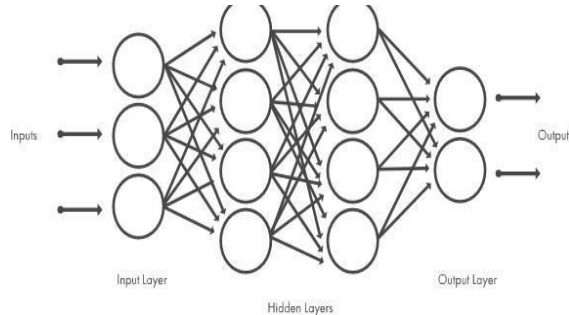


Fig 3. 1 Deep Neural Network

3.2 USES OF DEEP LEARNING

3.2.1 Translations

Although automatic machine translation isn't new, deep learning is helping enhance automatic translation of text by using stacked networks of neural networks and allowing translations from images.

3.2.2 Adding Color To Black-And-White Images And Videos

It is used to be a very time-consuming process where humans had to add color to black- and-white images and videos by hand can now be automatically done with deep-learning models.

3.2.3 Language Recognition

Deep learning machines are beginning to differentiate dialects of a language. A machine decides that someone is speaking English and then engages an AI that is learning to tell the differences between dialects. Once the dialect is determined, another AI will step in that specializes in that particular dialect. All of this happens without involvement from a human.

3.2.4 Autonomous Vehicles

There's not just one AI model at work as an autonomous vehicle drives down the street. Some deep-learning models specialize in streets signs while others are trained to recognize pedestrians. As a car navigates down the road, it can be informed by up to millions of individual AI models that allow the car to act.

3.2.5 Computer Vision

Deep learning has delivered super-human accuracy for image classification, object detection, image restoration and image segmentation—even handwritten digits can be recognized. Deep learning using enormous neural networks is teaching machines to automate the tasks performed by human visual systems.

3.2.6 Text Generation

The machines learn the punctuation, grammar and style of a piece of text and can use the model it developed to automatically create entirely new text with the proper spelling, grammar and style of the example text. Everything from Shakespeare to Wikipedia entries have been created.

3.2.7 Deep-Learning Robots

Deep-learning applications for robots are plentiful and powerful from an impressive deep-learning system that can teach a robot just by observing the actions of a human completing a task to a housekeeping robot that's provided with input from several other AIs in order to take action. Just like how a human brain processes input from past experiences, current input from senses and any additional data that is provided, deep-learning models will help robots execute tasks based on the input of many different AI opinions.

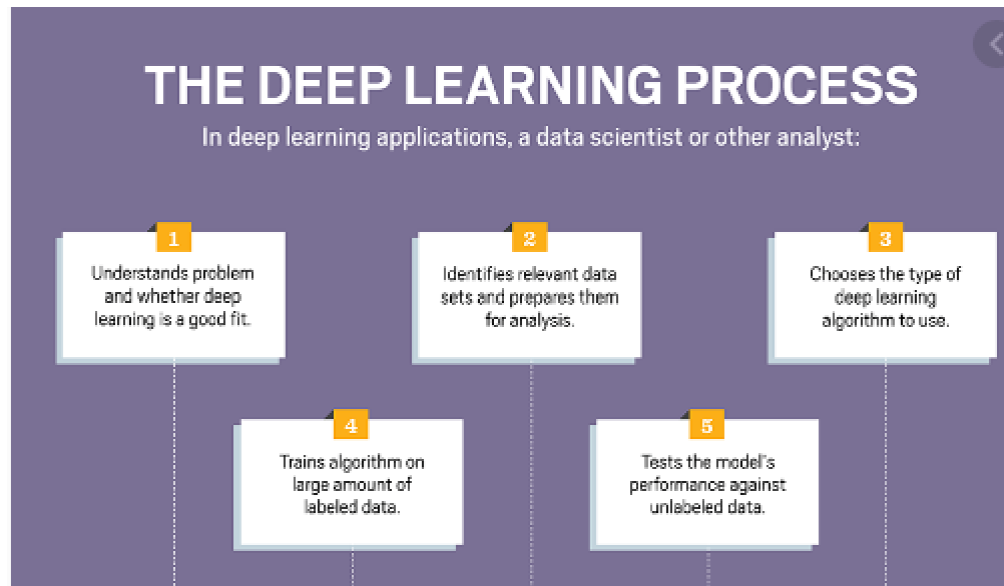


Fig 3. 2 The Deep Learning Process

3.3 RELATION BETWEEN DATA MINING, MACHINE LEARNING AND DEEP LEARNING

The deep learning, data mining and machine learning share a foundation in data science, and there certainly is overlap between the two. Data mining can use machine learning algorithms to improve the accuracy and depth of analysis, and vice-versa; machine learning can use mined data as its foundation, refining the dataset to achieve better results.

You could also argue that data mining and machine learning are similar in that they both seek to address the question of how we can learn from data. However, the way in which they achieve this end, and their applications, form the basis of some significant differences.

Machine Learning comprises of the ability of the machine to learn from trained data set and predict the outcome automatically. It is a subset of artificial intelligence.

Deep Learning is a subset of machine learning. It works in the same way on the machine just like how the human brain processes information. Like a brain can identify

the patterns by comparing it with previously memorized patterns, deep learning also uses this concept.

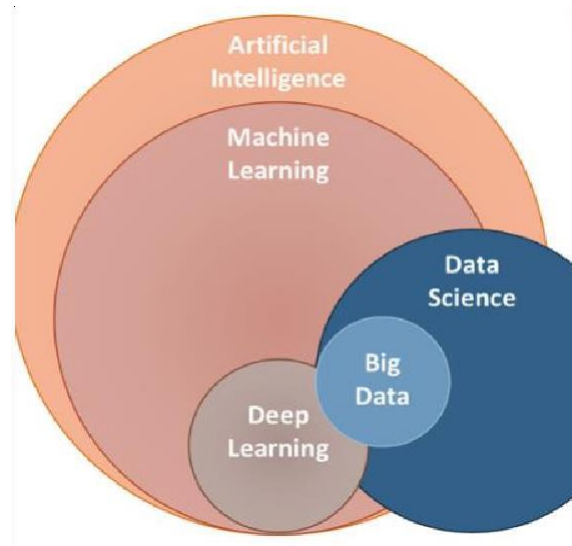


Fig 3. 3 Relation Between AI, ML, DL

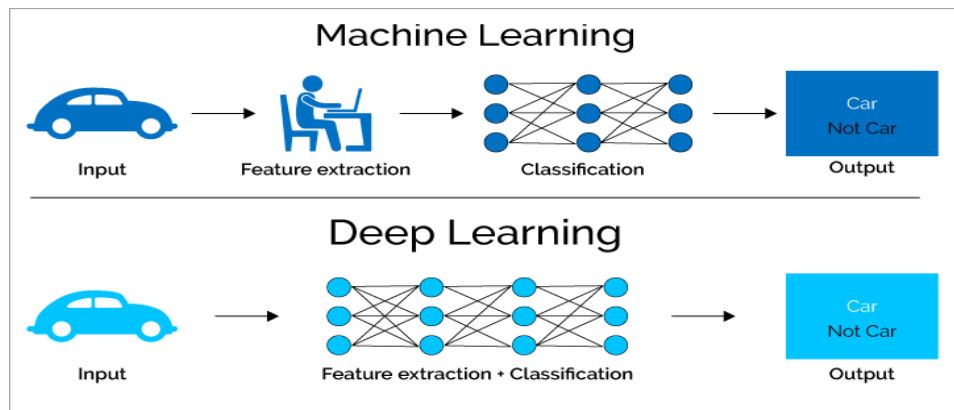


Fig 3. 4 Process In Machine Learning And Deep Learning

Deep learning can automatically find out the attributes from raw data while machine learning selects these features manually which further needs processing. It also employs artificial neural networks with many hidden layers, big data, and high computer resources.

Data Mining is a process of discovering hidden patterns and rules from the

existing data. It uses relatively simple rules such as association, correlation rules for the decision-making process, etc. Deep Learning is used for complex problem processing such as voice recognition etc. It uses Artificial Neural Networks with many hidden layers for processing. At times data mining also uses deep learning algorithms for processing the data.

CHAPTER 4

JAVASCRIPT

4.1 INTRODUCTION

JavaScript is a dynamic computer programming language. It is lightweight and most commonly used as a part of web pages, whose implementations allow client-side script to interact with the user and make dynamic pages. It is an interpreted programming language with object-oriented capabilities. JavaScript was first known as LiveScript, but Netscape changed its name to JavaScript, possibly because of the excitement being generated by Java.

JavaScript made its first appearance in Netscape 2.0 in 1995 with the name LiveScript. The general-purpose core of the language has been embedded in Netscape, Internet Explorer, and other web browsers.

The ECMA-262 Specification defined a standard version of the core JavaScript language. JavaScript is a lightweight, interpreted programming language. Designed for creating network-centric applications. Complementary to and integrated with Java. Complementary to and integrated with HTML. Open and cross-platform.

4.2 CLIENT-SIDE JAVASCRIPT

Client-side JavaScript is the most common form of the language. The script should be included in or referenced by an HTML document for the code to be interpreted by the browser.

It means that a web page need not be a static HTML, but can include programs that interact with the user, control the browser, and dynamically create HTML content.

The JavaScript client-side mechanism provides many advantages over traditional CGI server-side scripts. For example, you might use JavaScript to check if the user has entered a valid e-mail address in a form field.

The JavaScript code is executed when the user submits the form, and only if all the entries are valid, they would be submitted to the Web Server.

JavaScript can be used to trap user-initiated events such as button clicks, link navigation, and other actions that the user initiates explicitly or implicitly.

4.3 ADVANTAGES OF JAVASCRIPT

The merits of using JavaScript are:

Less server interaction: You can validate user input before sending the page off to the server. This saves server traffic, which means less load on your server.

Immediate feedback to the visitors: They don't have to wait for a page reload to see if they have forgotten to enter something.

Increased interactivity: You can create interfaces that react when the user hovers over them with a mouse or activates them via the keyboard.

Richer interfaces: You can use JavaScript to include such items as drag-and-drop components and sliders to give a Rich Interface to your site visitors.

4.4 LIMITATIONS OF JAVASCRIPT

We cannot treat JavaScript as a full-fledged programming language. It lacks the following important features: Client-side JavaScript does not allow the reading or writing of files. This has been kept for security reason. JavaScript cannot be used for networking applications because there is no such support available. JavaScript doesn't have any multithreading or multiprocessor capabilities. Once again, JavaScript is a lightweight, interpreted programming language that allows you to build interactivity into otherwise static HTML pages

4.5 P5.JS

p5.js is a JavaScript library used for creative coding. It is based on Processing which is a creative coding environment. The main focus of processing is to make it easy as possible for beginners to learn how to program interactive, graphical applications, to make a programming language more user-friendly by visualizing it. The advantage of using the JavaScript programming language is its broad availability and ubiquitous support: every web browser has a JavaScript interpreter built-in, which means that p5.js programs can be run on any web browser. Also, Processing is that language which emphasizes on feasibility for programmers to create software prototypes very quickly, to try out a new idea or see if something works. For this reason, Processing (and p5.js) programs are generally referred to as “sketches.”

4.5.1 Preferred Editors

The official documentation of p5.js suggests to use Bracket or Sublime and then include the JavaScript files, finally lead us to work like any other programming language. But the online p5.js Web Editor is the best alternative. It is based on a web-based programming environment.

4.5.2 Difference Between P5.js And JavaScript

JavaScript is a core language that provides all the features to build any functionalities into browsers. It can use Loop, Function, Conditional, DOM Manipulation, events, canvas, etc. Hence, by using it to develop and design any framework. p5.js is a library of JavaScript. P5.js is running on Pure JavaScript provides some functions that make JavaScript user life easy to draw in the web.

4.5.3 Setting up p5.js

To run p5.js in your computer you will need a text editor. You can use the code editor of your choice. Instructions for getting set up with Sublime Text 2 are included below, other good editor options include Brackets and Atom. If you are a screen reader

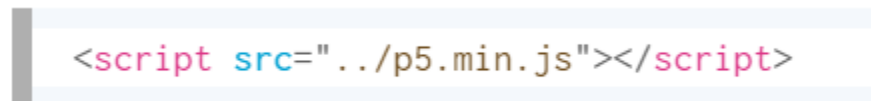
user and not using the p5 web editor, you may want to use Notepad++ or Eclipse.

4.5.3.1 Downloading A Copy Of The P5.Js Library

The easiest way to start is by using the empty example that comes with the p5.js complete download.

After download, you need to set up a local server. See instructions here . Run your local server within the downloaded folder and on your browser, go to <http://localhost:{your-port-num}/empty-example>.

If you look in index.html, you'll notice that it links to the file p5.js. If you would like to use the minified version (compressed for faster page loading), change the link to p5.min.js.



```
<script src="../p5.min.js"></script>
```

Fig 4. 1 Linking P5.js Min File

4.5.3.2 Using A Hosted Version Of The P5.Js Library

Alternatively, you can link to a p5.js file hosted online. All versions of p5.js are stored in a CDN (Content Delivery Network). You can find a history of these versions in the p5.js CDN. In this case you can change the link to:



```
<script src="https://cdn.jsdelivr.net/npm/p5@1.1.9/lib/p5.js"></script>
```

Fig 4. 2 Linking P5.js Using URL

A sample HTML page might look like this:

```
<html>
<head>
  <script src="https://cdn.jsdelivr.net/npm/p5@1.1.9/lib/p5.js"></script>
  <script src="sketch.js"></script>
</head>
<body>
  <main>
  </main>
</body>
</html>
```

Fig 4. 3 Sample HTML Page

4.5.4 Environment

Open Sublime. Go to the File menu and choose Open... and choose the folder that your html and js files are located in. On the left sidebar, you should find the folder name at the top, with a list of the files contained in the folder directly below.

Click on your sketch.js file, and it will open on the right where you can edit it. p5 starter code opened up in sublime editor."

Open the index.html file in your browser by double-clicking on it in your file manager or type: file:///the/file/path/to/your/html (or http://localhost:{your-port-num}/empty-example, if you are using a local server,) in the address bar to view your sketch.

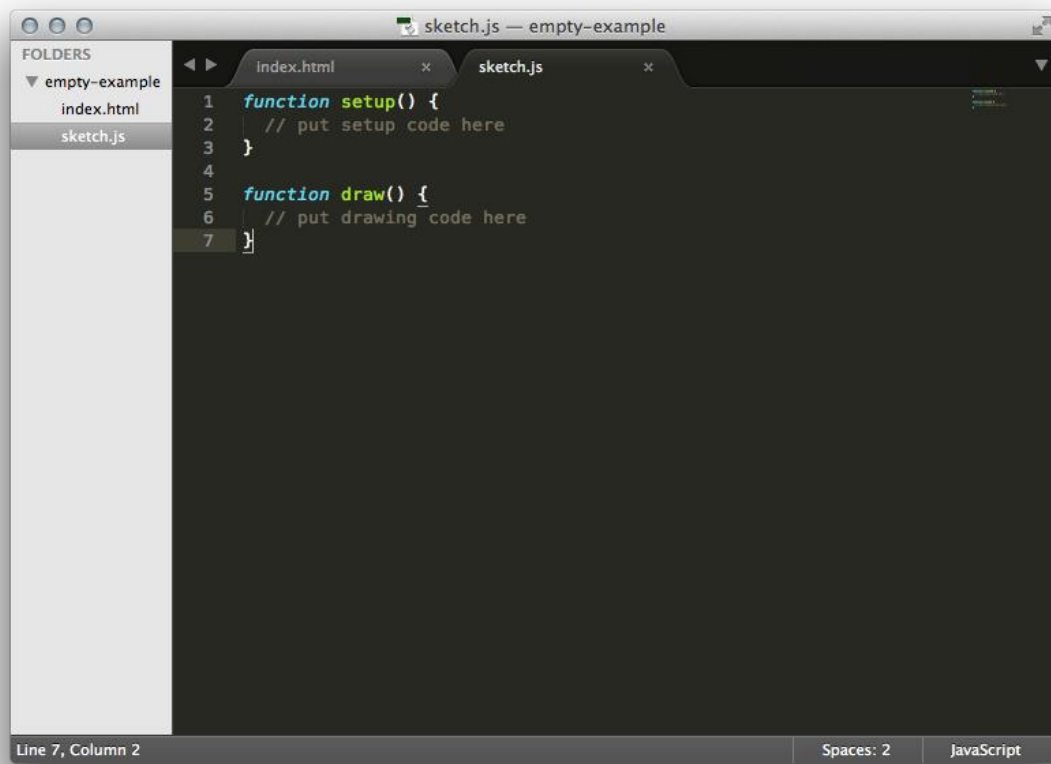


Fig 4. 4 P5.js Code Structure

CHAPTER 5

TENSORFLOW

5.1 INTRODUCTION

Currently, the most famous deep learning library in the world is Google's TensorFlow. Google product uses machine learning in all of its products to improve the search engine, translation, image captioning or recommendations.

To give a concrete example, Google users can experience a faster and more refined the search with AI. If the user types a keyword the search bar, Google provides a recommendation about what could be the next word.

Google wants to use machine learning to take advantage of their massive datasets to give users the best experience. Three different groups use machine learning – Researchers, Data scientists, Programmers.

They can all use the same toolset to collaborate with each other and improve their efficiency.

Google does not just have any data; they have the world's most massive computer, so Tensor Flow was built to scale. TensorFlow is a library developed by the Google Brain Team to accelerate machine learning and deep neural network research.

It was built to run on multiple CPUs or GPUs and even mobile operating systems, and it has several wrappers in several languages like Python, C++ or Java.

5.2 ARCHITECTURE

Tensorflow architecture works in three parts, Preprocessing the data, Build the model and Train and estimate the model.

It is called Tensorflow because it takes input as a multi-dimensional array, also known as **tensors**. You can construct a sort of **flowchart** of operations (called a Graph) that you want to perform on that input. The input goes in at one end, and then it flows

through this system of multiple operations and comes out the other end as output.

This is why it is called TensorFlow because the tensor goes in it flows through a list of operations, and then it comes out the other side. TensorFlow hardware, and software requirements can be classified into,

Development Phase: This is when you train the mode. Training is usually done on your Desktop or laptop.

Run Phase or Inference Phase: Once training is done Tensorflow can be run on many different platforms. You can run it on Desktop running Windows, macOS or Linux, Cloud as a web service or Mobile devices like iOS and Android.

You can train it on multiple machines then you can run it on a different machine, once you have the trained model.

5.3 COMPONENTS OF TENSORFLOW

5.3.1 Tensor

Tensorflow's name is directly derived from its core framework: **Tensor**. In Tensorflow, all the computations involve tensors. A tensor is a **vector** or **matrix** of n-dimensions that represents all types of data. All values in a tensor hold identical data type with a known (or partially known) **shape**. The shape of the data is the dimensionality of the matrix or array.

A tensor can be originated from the input data or the result of a computation. In TensorFlow, all the operations are conducted inside a **graph**. The graph is a set of computation that takes place successively. Each operation is called an **op node** and are connected to each other.

The graph outlines the ops and connections between the nodes. However, it does not display the values. The edge of the nodes is the tensor, i.e., a way to populate the operation with data.

5.3.2 Graphs

TensorFlow makes use of a graph framework. The graph gathers and describes all the series computations done during the training. The graph has lots of advantages. Like, it was done to run on multiple CPUs or GPUs and even mobile operating system, the portability of the graph allows to preserve the computations for immediate or later use. The graph can be saved to be executed in the future, All the computations in the graph are done by connecting tensors together. A tensor has a node and an edge. The node carries the mathematical operation and produces an endpoints output. The edges the edges explain the input/output relationships between nodes.

TensorFlow is the best library of all because it is built to be accessible for everyone. Tensorflow library incorporates different API to build at scale deep learning architecture like CNN or RNN. TensorFlow is based on graph computation; it allows the developer to visualize the construction of the neural network with Tensorboard. This tool is helpful to debug the program. Finally, Tensorflow is built to be deployed at scale. It runs on CPU and GPU.

5.4 ABOUT TENSORFLOW.JS

Tensorflow.js provides two things: The CoreAPI, which deals with the low level code and LayerAPI is built over the CoreAPI, and makes our lives easier by increasing the level of abstraction.

5.4.1 Getting Started

There are two main ways to get TensorFlow.js in your project:

5.4.1.1 Via <script> Tag

Add the following code to an HTML file:

```
<script src="https://cdn.jsdelivr.net/npm/@tensorflow/tfjs@0.12.0"> </script>
```

Fig 5. 1 Importing Tensorflow.js Using Script Tag

5.4.1.2 via NPM

Add TensorFlow.js to your project using yarn or npm.

```
yarn add @tensorflow/tfjs
```

Fig 5. 2 Adding Tensorflow.js using Yarn

```
npm install @tensorflow/tfjs
```

Fig 5. 3 Adding Tensorflow.js using npm

In your main js file:

```
import * as tf from '@tensorflow/tfjs';
```

Fig 5. 4 Importing Tensorflow.js

5.4.2 CoreAPI

5.4.2.1 Tensors



Fig 5. 5 Tensor

A scalar is a single number. For example, $x = 1$

A vector is an array of numbers. For example, $x = [1, 2]$

A matrix is a 2-D array ($[[1, 2], [3, 4], [5, 6]]$)

A tensor is a n-dimensional array with $n > 2$

TensorFlow.js has utility functions for common cases like Scalar, 1D, 2D, 3D and 4D tensors, as well a number of functions to initialize tensors in ways useful for machine learning.

5.4.2.2 Variables

Tensors are immutable data structures. That means their values can't be changed once they are set.

However, `tf.variable()` is introduced in TensorFlow.js. The real use case for `tf.variable()` is when we need to change the data frequently, such as when adjusting model weights in Machine Learning.

5.4.2.3 Operations

There are various operations in TensorFlow.js. In order to perform mathematical computation on Tensors, we use operations. Tensors are immutable, so all operations always return new Tensors and never modify input Tensors. So `tf.variable()` can be used to save memory.

5.4.2.4. Memory Management

Memory management is the key in Machine Learning/Deep Learning tasks because they are generally computationally expensive. TensorFlow.js provides two major ways to manage memory: `tf.dispose()` and `tf.tidy()`. They both typically do the same thing, but they do it in different ways.

`tf.tidy()` This executes the provided function `fn` and after it is run, cleans up all intermediate tensors allocated by `fn` except those returned by `fn`. `tf.tidy()` helps avoid memory leaks. In general, it wraps calls to operations in `tf.tidy()` for automatic memory cleanup.

`tf.dispose()` Disposes of any `tf.Tensors` found within the mentioned object.

5.4.3 LayersAPI

Layers are the primary building block for constructing an ML/DL Model. Each layer will typically perform some computation to transform its input to its output. Under the hood, every layer uses the CoreAPI of Tensorflow.js.

Layers will automatically take care of creating and initializing the various internal variables/weights they need to function. So, it makes life easier by increasing the level of abstraction.

TensorFlow.js is an open-source machine learning library that can run anywhere JavaScript can. It's based upon the original TensorFlow library written in Python and aims to re-create this developer experience and set of APIs for the JavaScript ecosystem.

Where can it be used? Given the portability of JavaScript this means we can now write in 1 language and perform machine learning across all of the following platforms with ease.

Client-side in the web browser using vanilla JavaScript

Server-side and even IoT devices like Raspberry Pi using Node.js

Desktop apps using Electron

Native mobile apps using React Native

TensorFlow.js also supports multiple backends within each of these environments (the actual hardware-based environments it can execute within such as the CPU or WebGL, for example. A "backend" in this context does not mean a server-side environment - the backend for execution could be client-side in WebGL, for instance) to ensure compatibility and also keep things running fast. Currently, TensorFlow.js supports:

WebGL execution on the device's graphics card (GPU) - this is the fastest way to execute larger models (over 3MB in size) with GPU acceleration.

Web Assembly (WASM) execution on CPU - to improve CPU performance across devices including older generation mobile phones, for example. This is better suited to smaller models (less than 3MB in size) which can execute faster on CPU with WASM than with WebGL due to the overhead of uploading content to a graphics processor.

CPU execution - the fallback should none of the other environments be available. This is the slowest of the three but is always there for you.

Note: You can choose to force one of these backends if you know what device you will be executing on, or you can let TensorFlow.js decide for you if you do not

specify this.

5.4.4 Client-Side Superpowers

Running TensorFlow.js in the web browser on the client machine can lead to several benefits that are worth considering.

5.4.4.1 Privacy

You can both train and classify data on the client machine without ever sending data to a 3rd party web server. There may be times where this may be a requirement to comply with local laws such as GDPR for example or when processing any data that the user may want to keep on their machine and not sent to a 3rd party.

5.4.4.2 Speed

As we do not have to send data to a remote server, inference (the act of classifying the data) can be faster. Even better, you have direct access to the device's sensors such as the camera, microphone, GPS, accelerometer and more should the user grant you access.

5.4.4.3 Reach And Scale

With one click anyone in the world can click a link you send them, open the web page in their browser, and utilize what you have made. No need for a complex server-side Linux setup with CUDA drivers and much more to use the machine learning system.

5.4.4.4 Cost

No servers mean the only thing you need to pay for is a CDN to host your HTML, CSS, JS, and model files. The cost of a CDN is much cheaper than keeping a server (potentially with a graphics card attached) running 24/7.

5.4.5 Server-Side Features

Leveraging the Node.js implementation of TensorFlow.js enables the following

features.

5.4.5.1 Full CUDA Support

On the server-side, for graphics card acceleration, one must install the NVIDIA CUDA drivers to enable TensorFlow to work with the graphics card (unlike in the browser which uses WebGL - no install needed). However, with full CUDA support, you can fully leverage the graphics card's lower level abilities which can lead to faster training and inference times. Performance is on parity with the Python TensorFlow implementation as they both share the same C++ backend.

5.4.5.2 Model Size

For cutting edge models from the research, you may be working with huge models, maybe gigabytes in size. These models can not currently be run in the web browser due to the limitations of memory usage per browser tab. To run these larger models, you can use Node.js on your server with the hardware specifications you require to run such a model efficiently.

5.4.5.3 IoT

Node.js is supported on popular single-board computers like the Raspberry Pi, which in turn means you can execute TensorFlow.js models on such devices too.

5.4.5.4 Speed

Node.js is written in JavaScript, which means that it benefits from just in time compilation. This means that you may often see performance gains when using Node.js as it will be optimized at runtime, especially for any preprocessing you may be doing. A great example of this can be seen in this case study which shows how Hugging Face used Node.js to get a 2x performance boost for their natural language processing model.

CHAPTER 6

ML5.JS

6.1 INTRODUCTION

The primary objective of ML5.js is to make ML user-friendly and accessible to a diverse audience. ML5.js adds another layer on top of TensorFlow.js to make it even simpler. The primary advantage of ML5.js is that it can be used in the browser itself without the requirement to install complicated dependencies. The funding and support to ML5.js has been given by the Google Education Grant at NYU's ITP/IMA programmed.

ML5.js is available as open-source software. It provides a high-level interface to the powerful Tensorflow.js library. If you have some exposure to Tensorflow.js, you might be aware that it is both easy to use and very powerful. As ML5.js has added another layer on Tensorflow.js, it should be even simpler.

6.2 IMPLEMENTATION

The browser-based approach is one of the important attributes of ML5.js. ML5.js requires no installation as it is browser based. To get started, you can right away open a code editor, start typing the code and execute it in a Web browser to see ML in action. Let's explore each of the steps to get started with ML5.js.

As a first step, write the following code to link to the ML5.js library from the browser, and to print the ML5 version in the console log window.

```

<!DOCTYPE html>
<html lang="en">
  <head>
    <title>Getting Started with ml5.js</title>
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <script src="https://unpkg.com/ml5@0.4.3/dist/ml5.min.js"></script>
  </head>
  <body>
    <script>
      console.log('ml5 version:', ml5.version);
    </script>
  </body>
</html>

```

Fig 6. 1 HTML Code For Importing ML5.js

```

<script src="https://unpkg.com/ml5@0.4.3/dist/ml5.min.js"></script>

```

Fig 6. 2 The CDN link to ML5.js

If you open the aforementioned code in a browser and look at the console log, you will get the version that will look something similar to what's given below:

```
ml5 version: 0.4.3
```

Fig 6. 3 Output Of The Above Code

ML5.js has support for the following categories:

The image, sound and text are used to apply data belonging to the respective type. The helper's category includes components such as Neural Networks and Feature Extractor, which can be used to perform transfer learning.

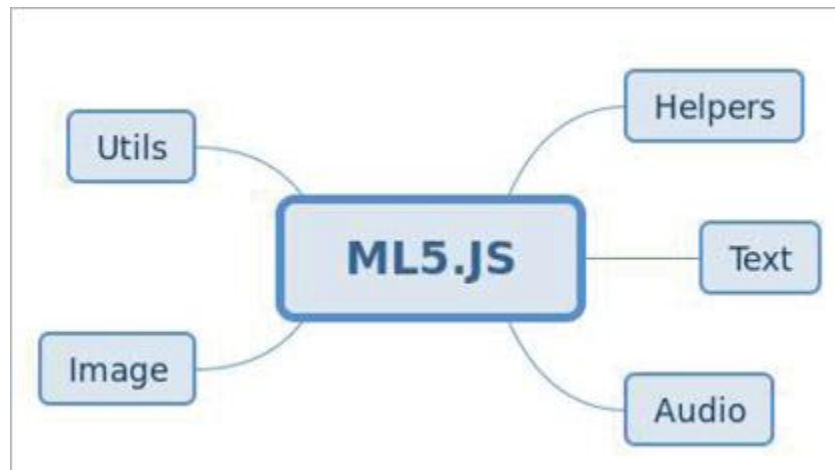


Fig 6. 4 ML5.JS Supports

CHAPTER 7

POSENET

PoseNet can be used to estimate either a single pose or multiple poses, meaning there is a version of the algorithm that can detect only one person in an image/video and one version that can detect multiple persons in an image/video. Why are there two versions? The single person pose detector is faster and simpler but requires only one subject present in the image (more on that later). We cover the single-pose one first because it's easier to follow.

At a high-level pose estimation happens in two phases: First, an input RGB image is fed through a convolutional neural network. Secondly, either a single-pose or multi-pose decoding algorithm is used to decode poses, pose confidence scores, keypoint positions, and keypoint confidence scores from the model outputs.

7.1 KEYWORD DEFINITION

Pose — at the highest level, PoseNet will return a pose object that contains a list of keypoints and an instance-level confidence score for each detected person.

Pose confidence score — this determines the overall confidence in the estimation of a pose. It ranges between 0.0 and 1.0. It can be used to hide poses that are not deemed strong enough.

Keypoint — a part of a person's pose that is estimated, such as the nose, right ear, left knee, right foot, etc. It contains both a position and a keypoint confidence score. PoseNet currently detects 17 keypoints illustrated in the following diagram:

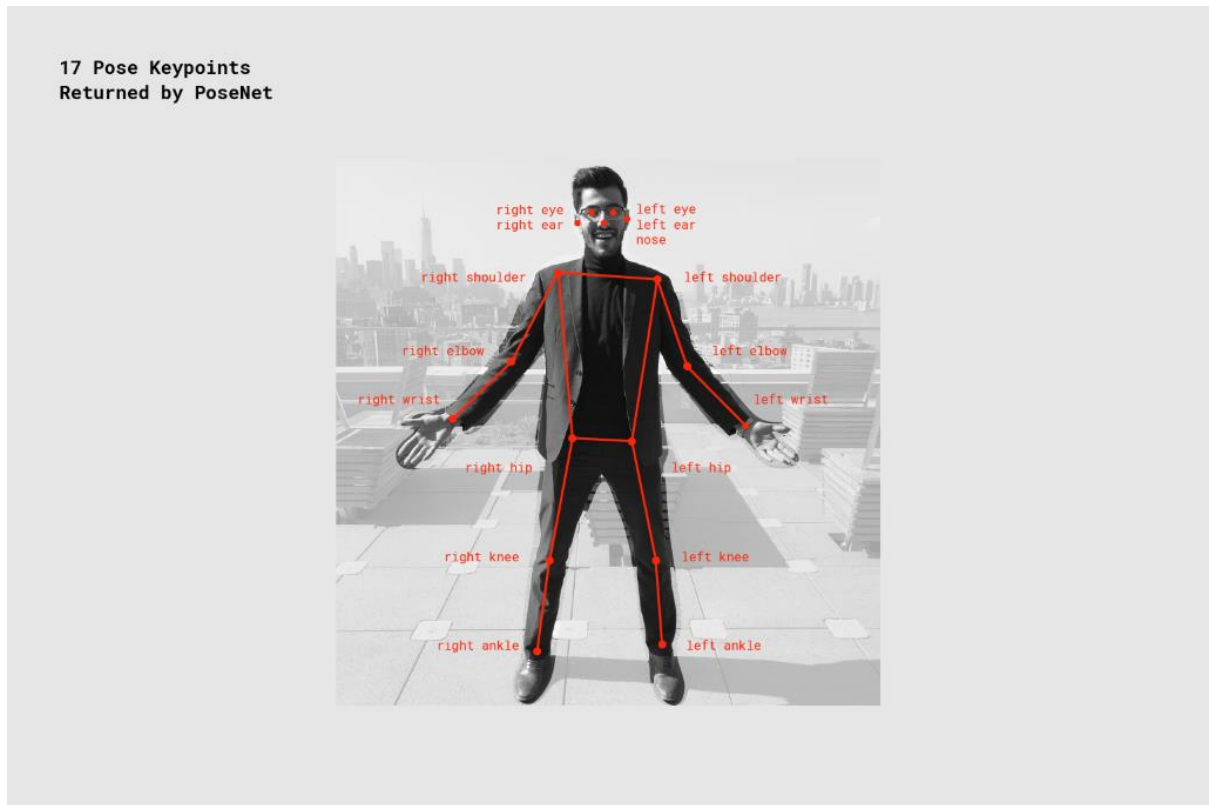


Fig 7. 1 Posenet Output Keypoints

Keypoint Confidence Score — this determines the confidence that an estimated keypoint position is accurate. It ranges between 0.0 and 1.0. It can be used to hide keypoints that are not deemed strong enough.

Keypoint Position — 2D x and y coordinates in the original input image where a keypoint has been detected.

7.2 POSE ESTIMATION

Pose estimation refers to computer vision techniques that detect human figures in images and videos, so that one could determine, for example, where someone's elbow shows up in an image. It is important to be aware of the fact that pose estimation merely estimates where key body joints are and does not recognize who is in an image or video.

The PoseNet model takes a processed camera image as the input and outputs information about keypoints. The keypoints detected are indexed by a part ID, with a

confidence score between 0.0 and 1.0. The confidence score indicates the probability that a keypoint exists in that position.

7.3 SINGLE-PERSON POSE ESTIMATION

As stated before, the single-pose estimation algorithm is the simpler and faster of the two. Its ideal use case is for when there is only one person centered in an input image or video. The disadvantage is that if there are multiple persons in an image, keypoints from both persons will likely be estimated as being part of the same single pose — meaning, for example, that person #1's left arm and person #2's right knee might be conflated by the algorithm as belonging to the same pose. If there is any likelihood that the input images will contain multiple persons, the multi-pose estimation algorithm should be used instead.

7.3.1 Inputs

Input image element — An html element that contains an image to predict poses for, such as a video or image tag. Importantly, the image or video element fed in should be square.

Image scale factor — A number between 0.2 and 1. Defaults to 0.50. What to scale the image by before feeding it through the network. Set this number lower to scale down the image and increase the speed when feeding through the network at the cost of accuracy.

Flip horizontal — Defaults to false. If the poses should be flipped/mirrored horizontally. This should be set to true for videos where the video is by default flipped horizontally (i.e. a webcam), and you want the poses to be returned in the proper orientation.

Output stride — Must be 32, 16, or 8. Defaults to 16. Internally, this parameter affects the height and width of the layers in the neural network. At a high level, it affects the accuracy and speed of the pose estimation. The lower the value of the output stride the higher the accuracy but slower the speed, the higher the value the faster the speed but

lower the accuracy. The best way to see the effect of the output stride on output quality is to play with the single-pose estimation demo.

7.3.2 Outputs

A pose, containing both a pose confidence score and an array of 17 **keypoints**.

Each keypoint contains a **keypoint position** and a **keypoint confidence score**. Again, all the keypoint positions have x and y coordinates in the input image space, and can be mapped directly onto the image.

This short code block shows how to use the single-pose estimation algorithm:

```
const imageScaleFactor = 0.50;
const flipHorizontal = false;
const outputStride = 16;

const imageElement = document.getElementById('cat');

// load the posenet model
const net = await posenet.load();

const pose = await net.estimateSinglePose(imageElement, scaleFactor,
flipHorizontal, outputStride);
```

Fig 7. 2 Single Pose Estimation Example Code

```
{
  "score": 0.32371445304906,
  "keypoints": [
    { // nose
      "position": {
        "x": 301.42237830162,
        "y": 177.69162777066
      },
      "score": 0.99799561500549
    },
    { // left eye
      "position": {
        "x": 326.05302262306,
        "y": 122.9596464932
      },
      "score": 0.99766051769257
    },
    { // right eye
      "position": {
        "x": 258.72196650505,
        "y": 127.51624706388
      },
      "score": 0.99926537275314
    },
    ...
  ]
}
```

Fig 7. 3 Example Output

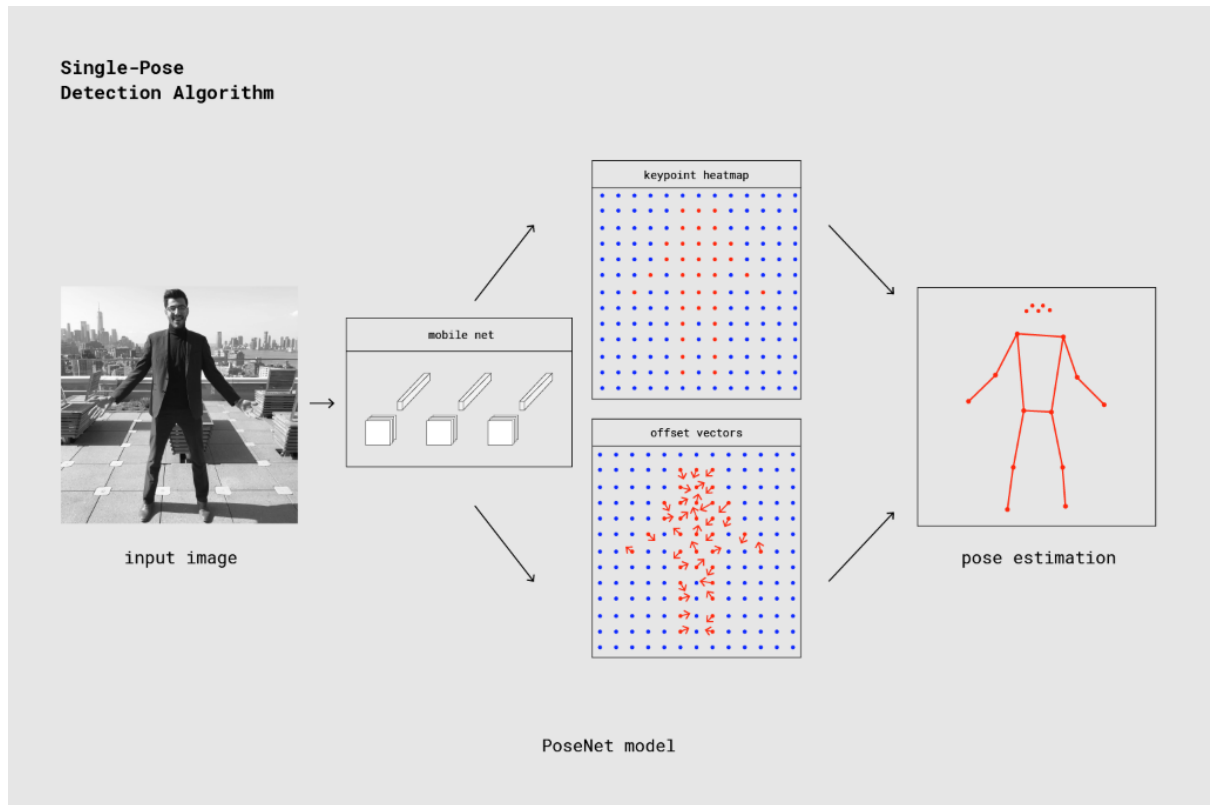


Fig 7. 4 Single Pose Detection Algorithm

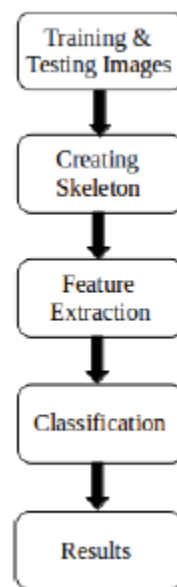


Fig 7. 5 Flow of the Current Project

CHAPTER 8

DESIGN AND IMPLEMENTATION

8.1 ARCHITECTURE

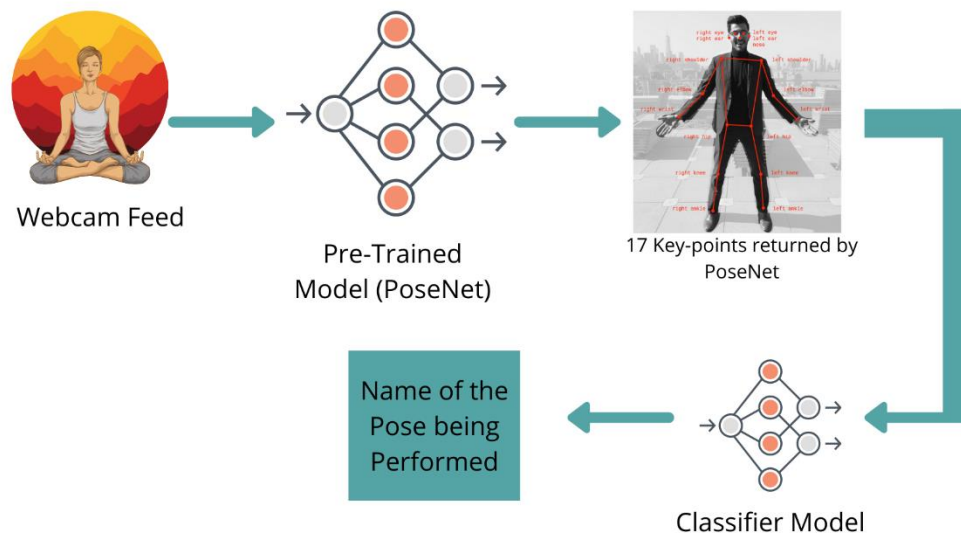


Fig 8. 1 Project Architecture

The webcam feed acts as the input given to the PoseNet model. Then the PoseNet model returns a json file containing 17 points in (x,y) form for each pose. These 17 points act as input for the classifier model. This model is made to classify the poses based on the 17 points received. The output which is received is the name of the pose that the model thinks is being performed.

8.2 DATA COLLECTION

The data collection process was unique as no pre-click images were used for this. Live webcam feed was used as a source of data. A person has to stand in front of the webcam and hold a particular pose for thirty seconds. Once that's completed, then they

hold another pose. This is continued until all the poses that the user wants to be classified are completed and then the classifier model is set to be trained.

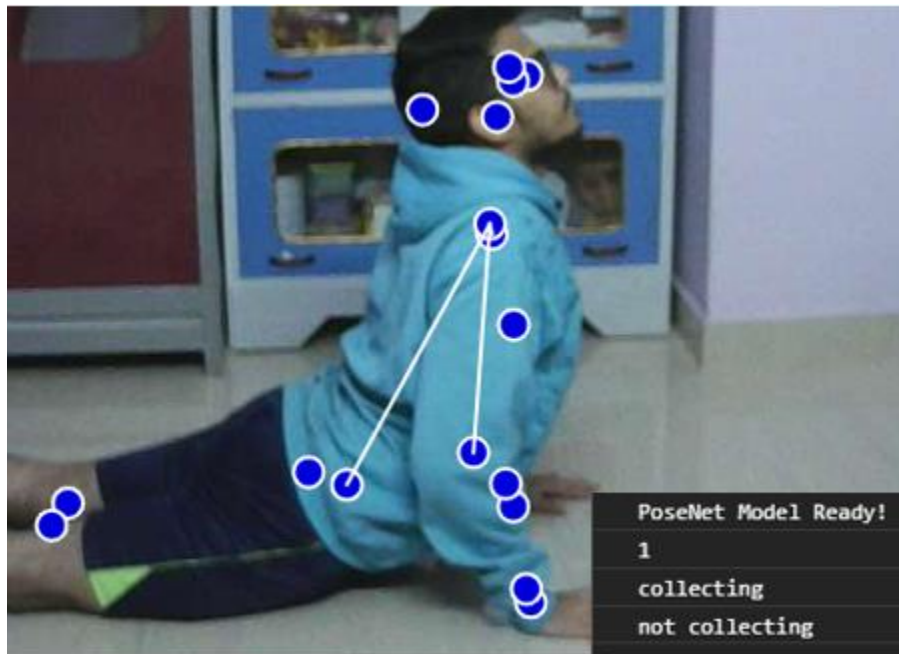


Fig 8. 2 Data Collection Example

In the above picture, 1 signifies the class label, and after 10s the model starts collecting data for 30 seconds and after which it stops collecting. A similar process can be followed for the other poses and after all the poses' data is collected, then the letter "s" can be pressed on the keyboard and a JSON file gets downloaded.

The data what is obtained from the live feed is passed through the PoseNet model. This model then returns 17 co-ordinates in a json file.

Id	Part
0	nose
1	leftEye
2	rightEye
3	leftEar
4	rightEar
5	leftShoulder
6	rightShoulder
7	leftElbow
8	rightElbow
9	leftWrist
10	rightWrist
11	leftHip
12	rightHip
13	leftKnee
14	rightKnee
15	leftAnkle
16	rightAnkle

Fig 8. 3 Seventeen Key-Points Returned

This json file then acts as input for the classifier model which learns to classify all the poses that the user had performed in front of the webcam.

```

{"data": [{"xs": {"input0": 327.45762518274853, "input1": 230.3505916483918, "input2": 340.70556164717345, "input3": 216.87945449561403,
"input4": 316.4760942068713, "input5": 217.83467729653995, "input6": 354.4341267665692, "input7": 224.23721521686156, "input8": 302.
6927997076023, "input9": 228.10777214303118, "input10": 378.5899808114035, "input11": 266.1833348562378, "input12": 279.92195114522417,
"input13": 264.4088465521442, "input14": 418.0738684819688, "input15": 223.5682794225146, "input16": 240.59800651803118, "input17": 228.
06286549707602, "input18": 460.3343536793372, "input19": 101.13039488913253, "input20": 202.777435124269, "input21": 101.86234085648147,
"input22": 292.6727735136452, "input23": 459.45967348927877, "input24": 372.3152336135477, "input25": 448.31745705409355, "input26": 434.
77183083576995, "input27": 472.2091633162768, "input28": 363.32065515350877, "input29": 550.4129355506823, "input30": 433.72685185185185,
"input31": 478.9686510112085, "input32": 433.1640244273879, "input33": 478.56877436647176}, {"ys": {"output0": "y"}}, {"xs": {"input0": 328.
3213404605263, "input1": 229.39989948830407, "input2": 341.165212292885, "input3": 214.7615893031189, "input4": 316.3912684271442,
"input5": 216.78096064814815, "input6": 354.87120035331384, "input7": 220.0738227948343, "input8": 302.60203460038986, "input9": 227.
95902244761209, "input10": 379.04487238060426, "input11": 265.5444764254386, "input12": 280.8453642787524, "input13": 264.16933174951265,
"input14": 418.47846613060426, "input15": 221.61199439571146, "input16": 238.95079495614033, "input17": 224.12107471369393, "input18": 205.
73657178971735, "input19": 99.50989126461988, "input20": 203.13240512305066, "input21": 100.51915813840154, "input22": 291.2635348135965,
"input23": 459.84367385477583, "input24": 364.94407133284597, "input25": 448.54977613304095, "input26": 337.65617385477583, "input27": 565.
0111933479532, "input28": 363.0197673001949, "input29": 561.5388949805068, "input30": 434.84192251461985, "input31": 341.62806103801165,
"input32": 425.5321408991228, "input33": 377.39221643518516}, {"ys": {"output0": "y"}}, {"xs": {"input0": 327.7118740862573, "input1": 229.

```

Fig 8. 4 Section Of JSON File

The obtained Data is first normalized as the raw data is relative to the canvas being used while the data was collected. This is why normalization is done so that the classification is faster and much more accurate.

8.2.1 Normalization

The goal of normalization is to transform features to be on a similar scale. This improves the performance and training stability of the model. Normalization Techniques at a Glance. Four common normalization techniques may be useful: scaling to a range, clipping, log scaling, z-score.

The following charts show the effect of each normalization technique on the distribution of the raw feature (price) on the left. The charts are based on the data set from 1985 Ward's Automotive Yearbook that is part of the UCI Machine Learning Repository under Automobile Data Set.

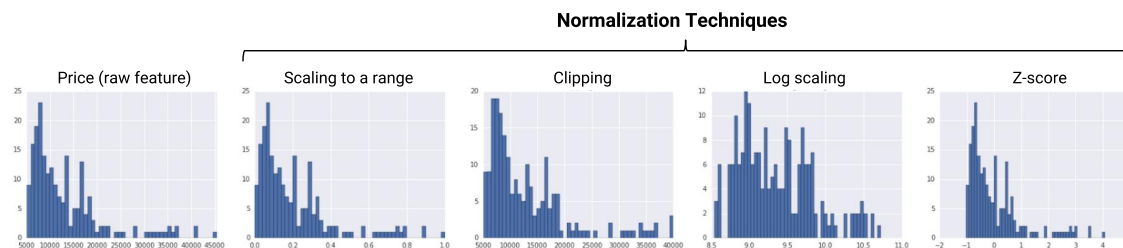


Fig 8. 5 Summary of normalization techniques

8.2.1.1 Scaling To A Range

Recall from MLCC that scaling means converting floating-point feature values from their natural range (for example, 100 to 900) into a standard range—usually 0 and 1 (or sometimes -1 to +1). Use the following simple formula to scale to a range:

$$x' = (x - x_{min}) / (x_{max} - x_{min})$$

Fig 8. 6 Scaling To A Range Formula

Scaling to a range is a good choice when both of the following conditions are met:
 You know the approximate upper and lower bounds on your data with few or no outliers.
 Your data is approximately uniformly distributed across that range.

A good example is age. Most age values falls between 0 and 90, and every part of the range has a substantial number of people.

In contrast, you would not use scaling on income, because only a few people have very high incomes. The upper bound of the linear scale for income would be very high, and most people would be squeezed into a small part of the scale.

8.2.1.2 Feature Clipping

If your data set contains extreme outliers, you might try feature clipping, which caps all feature values above (or below) a certain value to fixed value. For example, you could clip all temperature values above 40 to be exactly 40. You may apply feature clipping before or after other normalizations.

Formula: Set min / max values to avoid outliers.

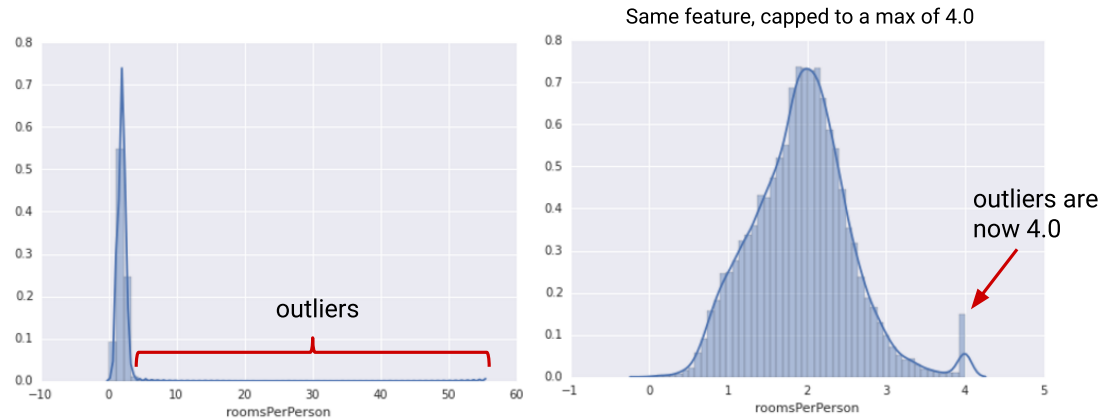


Fig 8. 7 Comparing A Raw Distribution Its Clipped Version

Another simple clipping strategy is to clip by z-score to $\pm N\sigma$ (for example, limit to $\pm 3\sigma$). Note that σ is the standard deviation.

8.2.1.3 Log Scaling

Log scaling computes the log of your values to compress a wide range to a narrow range.

$$x' = \log(x)$$

Fig 8. 8 Log Scaling Formula

Log scaling is helpful when a handful of your values have many points, while most other values have few points. This data distribution is known as the *power law* distribution. Movie ratings are a good example. In the chart below, most movies have very few ratings (the data in the tail), while a few have lots of ratings (the data in the head). Log scaling changes the distribution, helping to improve linear model performance.

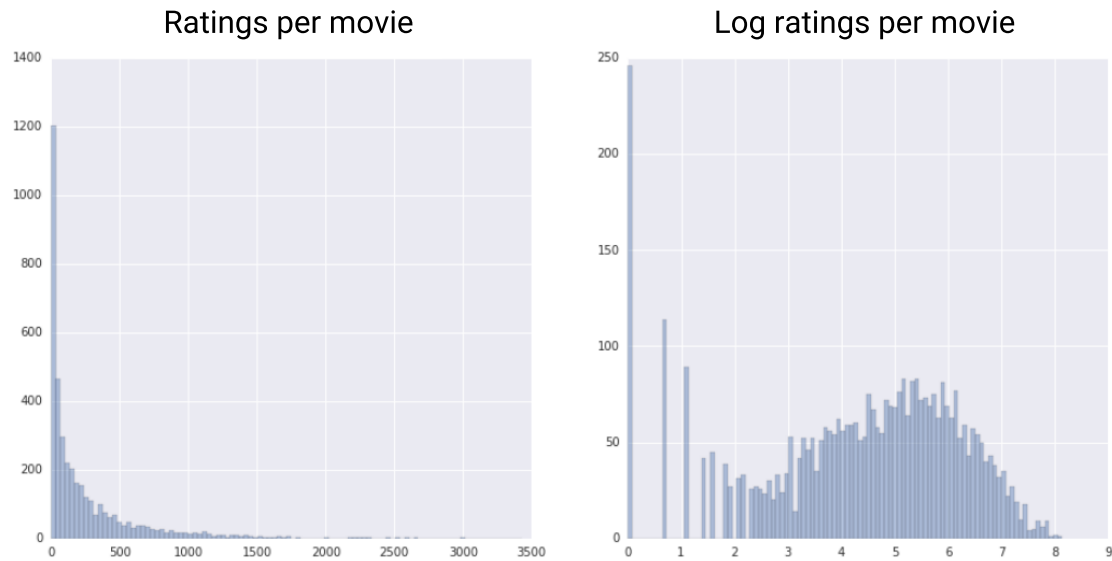


Fig 8. 9 Comparing A Raw Distribution To Its Log

8.2.1.4 Z-Score

Z-score is a variation of scaling that represents the number of standard deviations away from the mean. You would use z-score to ensure your feature distributions have mean = 0 and std = 1. It's useful when there are a few outliers, but not so extreme that you need clipping.

The formula for calculating the z-score of a point, x , is as follows:

$$x' = (x - \mu) / \sigma$$

Note: μ is the mean and σ is the standard deviation.

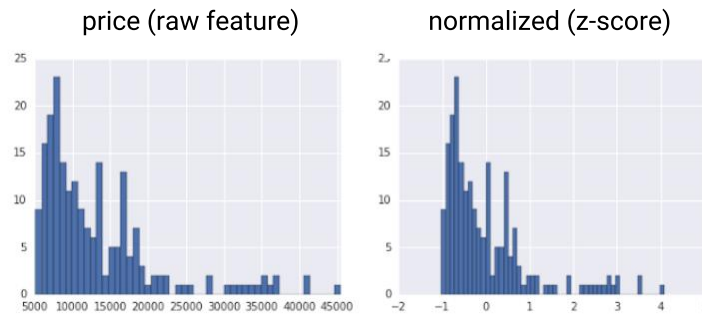


Fig 8. 10 Comparing A Raw Distribution To Its Z-Score Distribution

Notice that z-score squeezes raw values that have a range of ~40000 down into a range from roughly -1 to +4.

Suppose you're not sure whether the outliers truly are extreme. In this case, start with z-score unless you have feature values that you don't want the model to learn; for example, the values are the result of measurement error or a quirk.

8.2.1.5 Use Of Normalization In Project

In the current project, Scaling to a range normalization was used. The values in the data points received from the PoseNet model were brought to a range between 0 and 1. Only after this was the model trained.

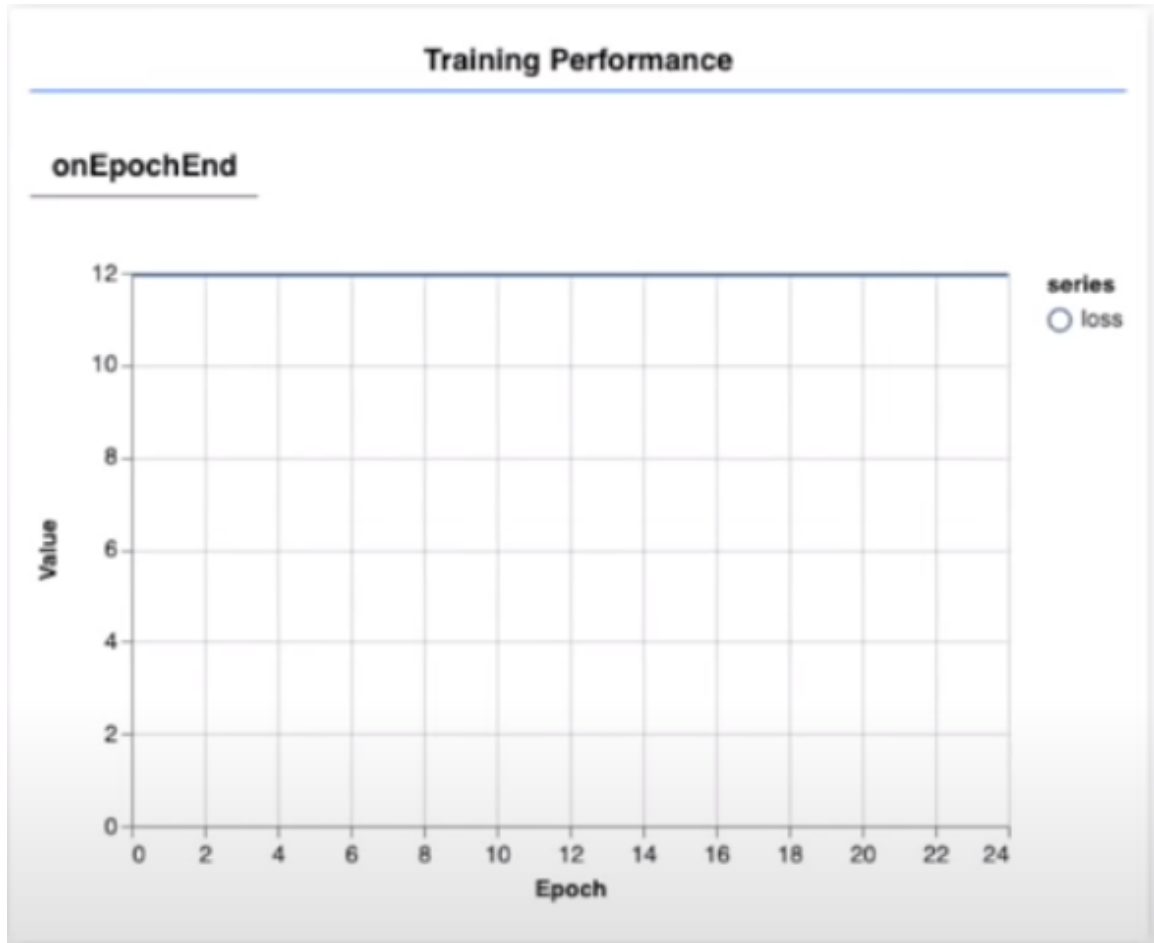


Fig 8. 11 Training On Un-Normalized Data

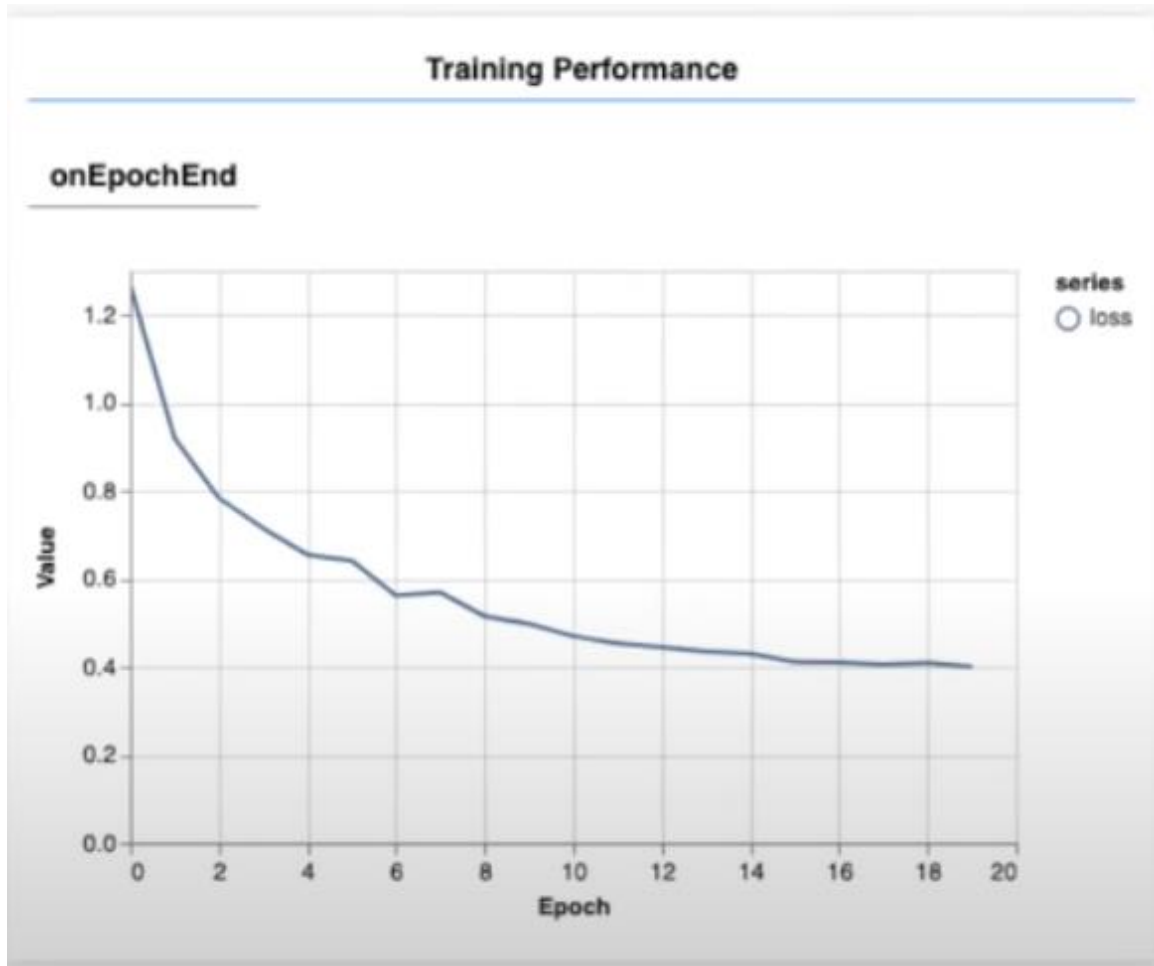


Fig 8. 12 Training on Normalized Data

The above two graphs act as examples for the difference caused when the data is normalized versus when the data is not normalized.

8.3 MODEL TRAINING

A Neural network with the following architecture was used as the classifier.

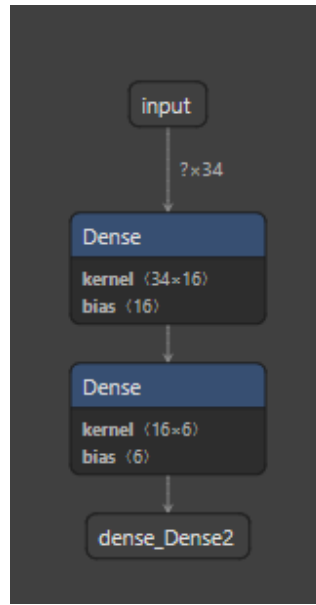


Fig 8. 13 Classifier Model Architecture

The Input in this model is the JSON that is returned by the PoseNet model at the time of collecting data. This is fed into the model after the data is normalised so that all the data is in the same form. As mentioned earlier, Normalization step is important to maintain consistency among data points.

8.3.1 Neural Network

A neural network is a series of algorithms that endeavors to recognize underlying relationships in a set of data through a process that mimics the way the human brain operates. In this sense, neural networks refer to systems of neurons, either organic or artificial in nature. Neural networks can adapt to changing input; so the network generates the best possible result without needing to redesign the output criteria. The concept of neural networks, which has its roots in artificial intelligence, is swiftly gaining popularity in the development of trading systems.

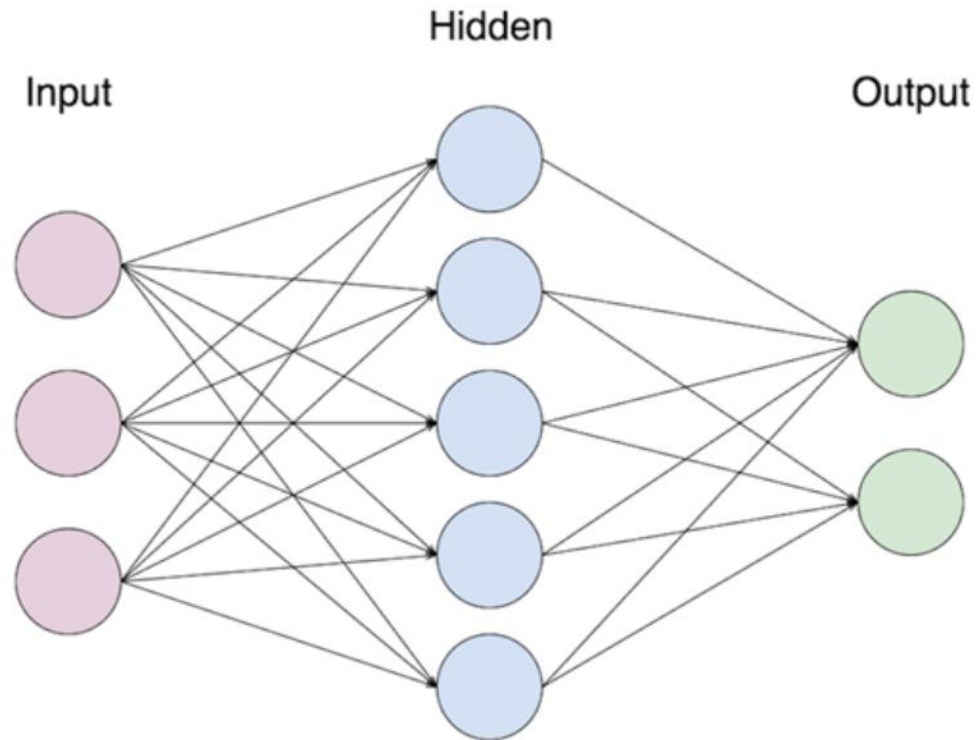


Fig 8. 14 Sample Neural Network Architecture

8.3.1.1 Components of Artificial Neural Networks

ANNs are comprised of three core layers or phases – an input layer, hidden layer/s, and an output layer.

Input Layer: The first layer is fed with the input, that is, raw data. It conveys the information from the outside world to the network. In this layer, no computation is performed – the nodes merely pass on the information to the hidden layer.

Hidden Layer: In this layer, the nodes lie hidden behind the input layer – they comprise the abstraction part in every neural network. All the computations on the features entered through the input layer occur in the hidden layer/s, and then, it transfers the result to the output layer.

Output Layer: This layer depicts the results of the computations performed by the network to the outer world.

Neural networks can be categorized into different types based on the activity of the hidden layer/s. For instance, in a simple neural network, the hidden units can construct their unique representation of the input. Here, the weights between the hidden and input units decide when each hidden unit is active.

Thus, by adjusting these weights, the hidden layer can choose what it should represent. Other architectures include the single layer and multilayer models. In a single layer, there's usually only an input and output layer – it lacks a hidden layer. Whereas, in a multilayer model, there is one or more than one hidden layer.

8.3.2 Activation Functions

As we mentioned earlier, ANNs are a crucial component of many structures that are helping revolutionize the world around us. But have you ever wondered, how do ANNs deliver state-of-the-art performance to find solutions to real-world problems?

The answer is – Activation Functions.

ANNs use activation functions (AFs) to perform complex computations in the hidden layers and then transfer the result to the output layer. The primary purpose of AFs is to introduce non-linear properties in the neural network.

They convert the linear input signals of a node into non-linear output signals to facilitate the learning of high order polynomials that go beyond one degree for deep networks. A unique aspect of AFs is that they are differentiable – this helps them function during the backpropagation of the neural networks.

If activation functions are not applied, the output signal would be a linear function, which is a polynomial of one degree. While it is easy to solve linear equations, they have a limited complexity quotient and hence, have less power to learn complex functional mappings from data. Thus, without AFs, a neural network would be a linear regression model with limited abilities.

This is certainly not what we want from a neural network. The task of neural networks is to compute highly complicated calculations. Furthermore, without AFs, neural networks cannot learn and model other complicated data, including images, speech, videos, audio, etc.

AFs help neural networks to make sense of complicated, high dimensional, and non-linear Big Data sets that have an intricate architecture – they contain multiple hidden layers in between the input and output layer.

8.3.2.1 Sigmoid Function

In an ANN, the sigmoid function is a non-linear AF used primarily in feedforward neural networks. It is a differentiable real function, defined for real input values, and containing positive derivatives everywhere with a specific degree of smoothness. The sigmoid function appears in the output layer of the deep learning models and is used for predicting probability-based outputs. The sigmoid function is represented as:

$$f(x) = \left(\frac{1}{(1 + \exp^{-x})} \right) \quad (1.4)$$

Fig 8. 15 Sigmoid Function Formula

Generally, the derivatives of the sigmoid function are applied to learning algorithms. The graph of the sigmoid function is ‘S’ shaped.

Some of the major drawbacks of the sigmoid function include gradient saturation, slow convergence, sharp damp gradients during backpropagation from within deeper hidden layers to the input layers, and non-zero centered output that causes the gradient updates to propagate in varying directions.

8.3.2.2 Hyperbolic Tangent Function (Tanh)

The hyperbolic tangent function, a.k.a., the tanh function, is another type of AF. It is a smoother, zero-centered function having a range between -1 to 1. As a result, the

output of the tanh function is represented by:

$$f(x_i) = \frac{\exp(x_i)}{\sum_j \exp(x_j)} \quad (1.12)$$

Fig 8. 16 Tanh Function Formula

The tanh function is much more extensively used than the sigmoid function since it delivers better training performance for multilayer neural networks. The biggest advantage of the tanh function is that it produces a zero-centered output, thereby supporting the backpropagation process. The tanh function has been mostly used in recurrent neural networks for natural language processing and speech recognition tasks.

However, the tanh function, too, has a limitation – just like the sigmoid function, it cannot solve the vanishing gradient problem. Also, the tanh function can only attain a gradient of 1 when the input value is 0 (x is zero). As a result, the function can produce some dead neurons during the computation process.

8.3.2.3 Softmax Function

The softmax function is another type of AF used in neural networks to compute probability distribution from a vector of real numbers. This function generates an output that ranges between values 0 and 1 and with the sum of the probabilities being equal to 1. The softmax function is represented as follows:

$$f(x_i) = \frac{\exp(x_i)}{\sum_j \exp(x_j)} \quad (1.12)$$

Fig 8. 17 Softmax Function Formula

This function is mainly used in multi-class models where it returns probabilities of each class, with the target class having the highest probability. It appears in almost all the output layers of the DL architecture where they are used. The primary difference between the sigmoid and softmax AF is that while the former is used in binary classification, the latter is used for multivariate classification.

8.3.2.4 Softsign Function

The softsign function is another AF that is used in neural network computing. Although it is primarily in regression computation problems, nowadays it is also used in DL based text-to-speech applications. It is a quadratic polynomial, represented by:

$$f(x) = \left(\frac{x}{|x| + 1} \right) \quad - (1.13)$$

Fig 8. 18 Softsign Function Formula

Here “x” equals the absolute value of the input.

The main difference between the softsign function and the tanh function is that unlike the tanh function that converges exponentially, the softsign function converges in a polynomial form.

8.3.2.5 Rectified Linear Unit (ReLU) Function

One of the most popular AFs in DL models, the rectified linear unit (ReLU) function, is a fast-learning AF that promises to deliver state-of-the-art performance with stellar results. Compared to other AFs like the sigmoid and tanh functions, the ReLU function offers much better performance and generalization in deep learning. The function is a nearly linear function that retains the properties of linear models, which makes them easy to optimize with gradient-descent methods.

The ReLU function performs a threshold operation on each input element where all values less than zero are set to zero. Thus, the ReLU is represented as:

$$f(x) = \max(0, x) = \begin{cases} x_i, & \text{if } x_i \geq 0 \\ 0, & \text{if } x_i < 0 \end{cases} \quad - (1.14)$$

Fig 8. 19 ReLU Function Formula

By rectifying the values of the inputs less than zero and setting them to zero, this

function eliminates the vanishing gradient problem observed in the earlier types of activation functions (sigmoid and tanh).

The most significant advantage of using the ReLU function in computation is that it guarantees faster computation – it does not compute exponentials and divisions, thereby boosting the overall computation speed. Another critical aspect of the ReLU function is that it introduces sparsity in the hidden units by squishing the values between zero to maximum.

8.3.2.6 Exponential Linear Units (ELUs) Function

The exponential linear units (ELUs) function is an AF that is also used to speed up the training of neural networks (just like ReLU function). The biggest advantage of the ELU function is that it can eliminate the vanishing gradient problem by using identity for positive values and by improving the learning characteristics of the model.

ELUs have negative values that push the mean unit activation closer to zero, thereby reducing computational complexity and improving the learning speed. The ELU is an excellent alternative to the ReLU – it decreases bias shifts by pushing mean activation towards zero during the training process.

The exponential linear unit function is represented as

$$f(x) = \begin{cases} x, & \text{if } x > 0 \\ \alpha \exp(x) - 1, & \text{if } x \leq 0 \end{cases} \quad - (1.22)$$

Fig 8. 20 ELU Representation

The derivative or gradient of the ELU equation is presented as:

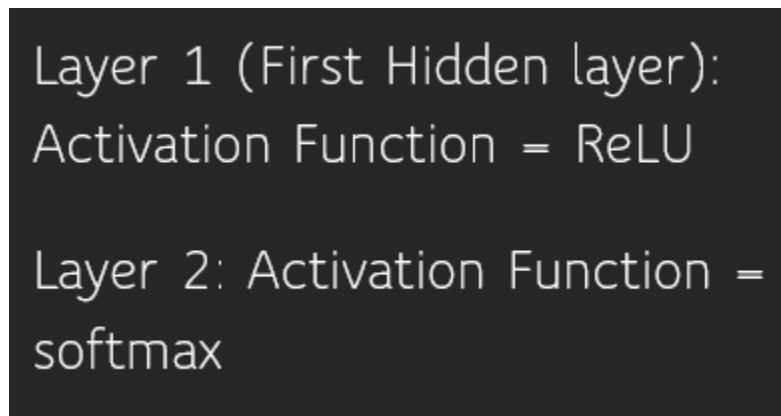
$$f' = \begin{cases} 1, & \text{if } x > 0 \\ f(x) + \alpha, & \text{if } x \leq 0 \end{cases} \quad - (1.23)$$

Fig 8. 21 Derivative or Gradient of ELU

Here “ α ” equals the ELU hyperparameter that controls the saturation point for negative net inputs, which is usually set to 1.0. However, the ELU function has a limitation – it is not zero-centered.

8.3.3 Classifier Model Breakdown

For the current project,



Layer 1 (First Hidden layer):
Activation Function = ReLU

Layer 2: Activation Function =
softmax

Fig 8. 22 Layer Breakdown

ReLU was used in the first layer and softmax in the second layer. Since it is a classification problem, Softmax functions are generally used in the final layer of the model.

The output of the above model is in the form of JSON containing 2 values, the result and the confidence.


```
▼ (4) [{...}, {...}, {...}, {...}, tensor: null]   
  ▶ 0: {label: "1", confidence: 0.9643369913101196}  
  ▶ 1: {label: "3", confidence: 0.02193737030029297}  
  ▶ 2: {label: "4", confidence: 0.009007570333778858}  
  ▶ 3: {label: "2", confidence: 0.004717966075986624}
```

Fig 8. 23 Classifier Model Sample JSON Output

In the above example the class with label “1” is the most likely and class with the label “2” is the least likely. The order of the labels changes based on the pose being performed.



Fig 8. 24 Pose Classified as Mountain



Fig 8. 25 Pose Classified as Warrior 2



Fig 8. 26 Pose Classified as Warrior 1



Fig 8. 27 Pose Classified as Downward Dog

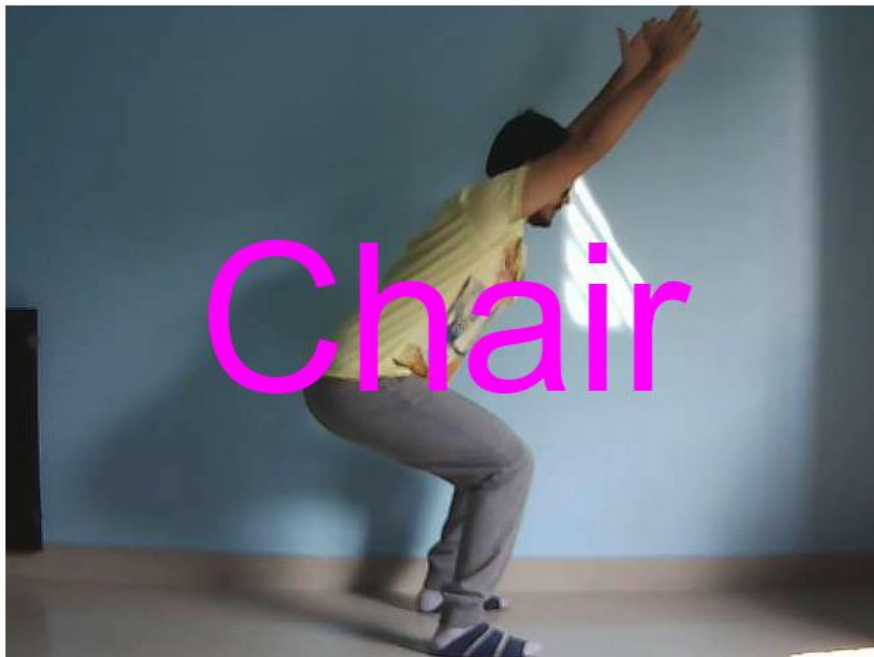


Fig 8. 28 Pose Classified as Chair



Fig 8. 29 Pose Classified as Tree

8.4 UML MODELS

UML is an acronym that stands for Unified Modeling Language. Simply put, UML is a modern approach to modeling and documenting software. In fact, it's one of the most popular business process modeling techniques.

It is based on diagrammatic representations of software components. As the old proverb says: "a picture is worth a thousand words". By using visual representations, we are able to better understand possible flaws or errors in software or business processes.

UML was created as a result of the chaos revolving around software development and documentation. In the 1990s, there were several different ways to represent and document software systems. The need arose for a more unified way to visually represent those systems and as a result, in 1994-1996, the UML was developed by three software engineers working at Rational Software. It was later adopted as the standard in 1997 and has remained the standard ever since, receiving only a few updates.

8.4.1 Use-Case Diagram

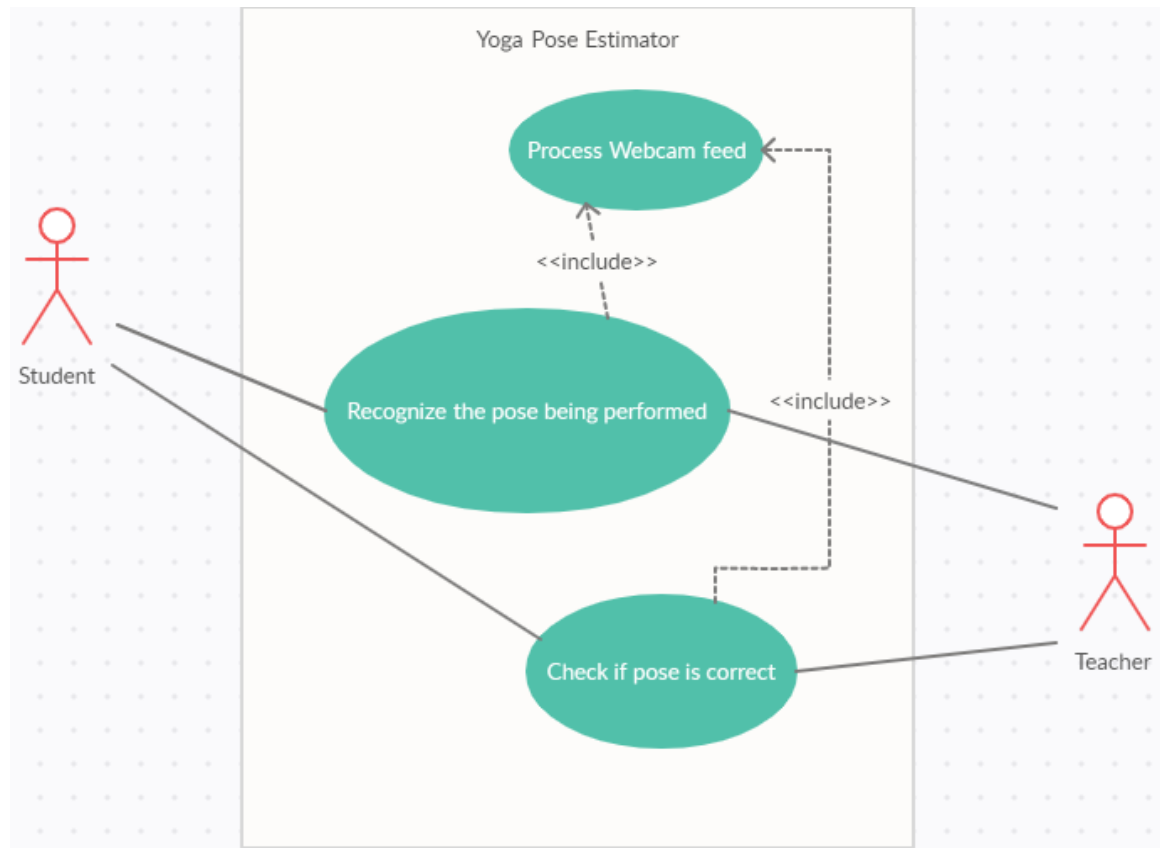


Fig 8. 30 Use Case Diagram

The Above Use-Case diagram is for the Pose Estimation Project. It shows how the users will be interacting with the project. The student and the teacher, both can check if the user is in the correct form or not. An alert is sent to the user if the pose is incorrect and the user can ask the teacher for further assistance to correct their posture.

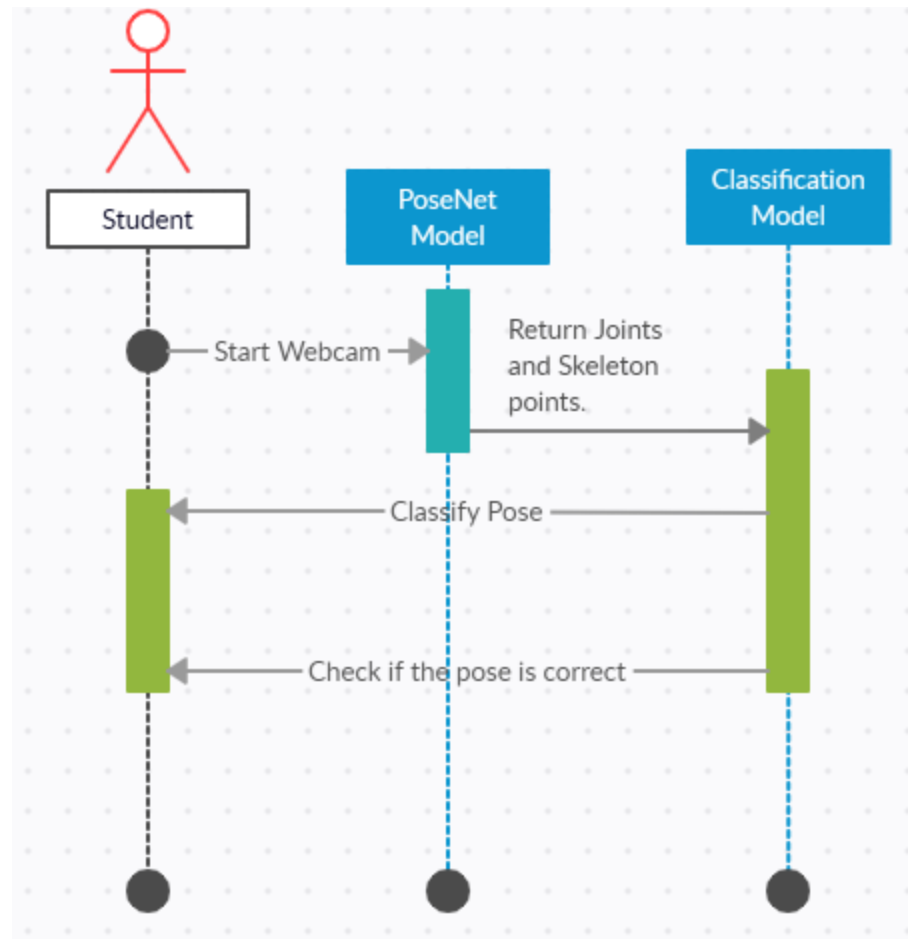


Fig 8. 31 Sequence Diagram

CHAPTER 9

SYSTEM TESTING

9.1 INTRODUCTION TO TESTING

System Testing is the testing of a complete and fully integrated software product. Usually, software is only one element of a larger computer-based system. Ultimately, software is interfaced with other software/hardware systems. System Testing is actually a series of different tests whose sole purpose is to exercise the full computer-based system.

9.2 TESTING STRATEGIES

System test falls under the **black box testing** category of software testing.

White box testing is the testing of the internal workings or code of a software application. In contrast, **black box** or System Testing is the opposite. System test involves the external workings of the software from the user's perspective.

9.2.1 What is Verified in System Testing?

System Testing involves testing the software code for following

Testing the fully integrated applications including external peripherals in order to check how components interact with one another and with the system as a whole. This is also called End to End testing scenario.

Verify thorough testing of every input in the application to check for desired outputs.

Testing of the user's experience with the application.

That is a very basic description of what is involved in system testing. You need to build detailed test cases and test suites that test each aspect of the application as seen from the outside without looking at the actual source code.

9.3 SOFTWARE TESTING HIERARCHY

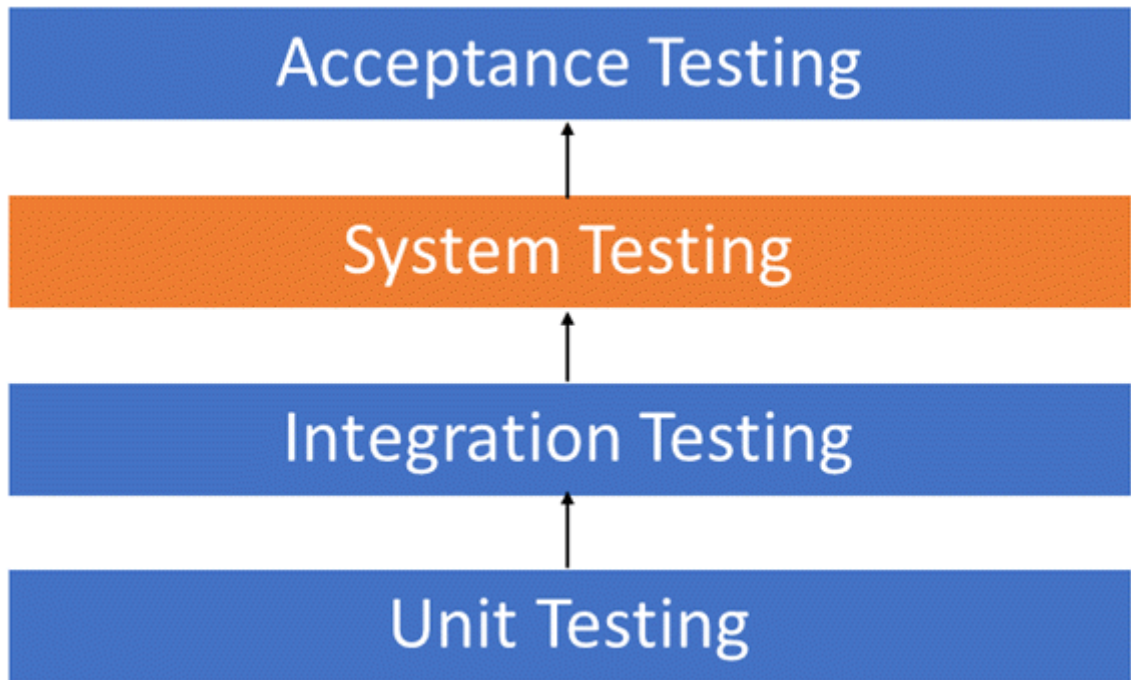


Fig 9. 1 Flow Chart For Testing

As with almost any software engineering process, software testing has a prescribed order in which things should be done. The following is a list of software testing categories arranged in chronological order. These are the steps taken to fully test new software in preparation for marketing it:

Unit testing - testing performed on each module or block of code during development. Unit Testing is normally done by the programmer who writes the code.

Integration testing - testing done before, during and after integration of a new module into the main software package. This involves testing of each individual code module. One piece of software can contain several modules which are often created by several different programmers. It is crucial to test each module's effect on the entire program model.

System testing - testing done by a professional testing agent on the completed software product before it is introduced to the market.

Acceptance testing - beta testing of the product done by the actual end users.

9.4 DIFFERENT TYPES OF SYSTEM TESTING

There are more than 50 types of System Testing. Below we have listed types of system testing a large software development company would typically use

Usability Testing - Usability Testing mainly focuses on the user's ease to use the application, flexibility in handling controls and ability of the system to meet its objectives

Load Testing - Load Testing is necessary to know that a software solution will perform under real-life loads.

Regression Testing - Regression Testing involves testing done to make sure none of the changes made over the course of the development process have caused new bugs. It also makes sure no old bugs appear from the addition of new software modules over time.

Recovery Testing - Recovery testing is done to demonstrate a software solution is reliable, trustworthy and can successfully recoup from possible crashes

Migration Testing - Migration testing is done to ensure that the software can be moved from older system infrastructures to current system infrastructures without any issues.

Functional Testing - Also known as functional completeness testing, Functional Testing involves trying to think of any possible missing functions. Testers might make a list of additional functionalities that a product could have to improve it during functional testing.

Hardware/Software Testing - IBM refers to Hardware/Software testing as "HW/SW Testing". This is when the tester focuses his/her attention on the interactions between the hardware and software during system testing.

9.5 WHAT TYPES OF SYSTEM TESTING SHOULD TESTERS USE?

There are over 50 different types of system testing. The specific types used by a tester depend on several variables. Those variables include:

Who the tester works for - This is a major factor in determining the types of system testing a tester will use. Methods used by large companies are different than that used by medium and small companies.

Time available for testing - Ultimately, all 50 testing types could be used. Time is often what limits us to using only the types that are most relevant for the software project.

Resources available to the tester - Of course some testers will not have the necessary resources to conduct a testing type. For example, if you are a tester working for a large software development firm, you are likely to have expensive automated testing software not available to others.

Software Tester's Education - There is a certain learning curve for each type of software testing available. To use some of the software involved, a tester has to learn how to use it.

Testing Budget - Money becomes a factor not just for smaller companies and individual software developers but large companies as well.

CHAPTER 10

RESULTS

After following the flow that's mentioned in Fig 10.1 A system is created which can be run entirely on the user's browser without the need of any additional hardware.

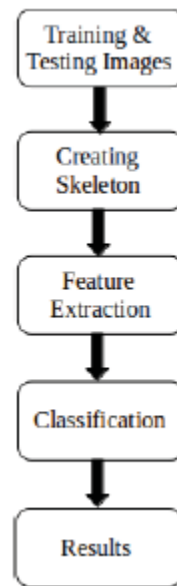


Fig 10. 1 Project Flow

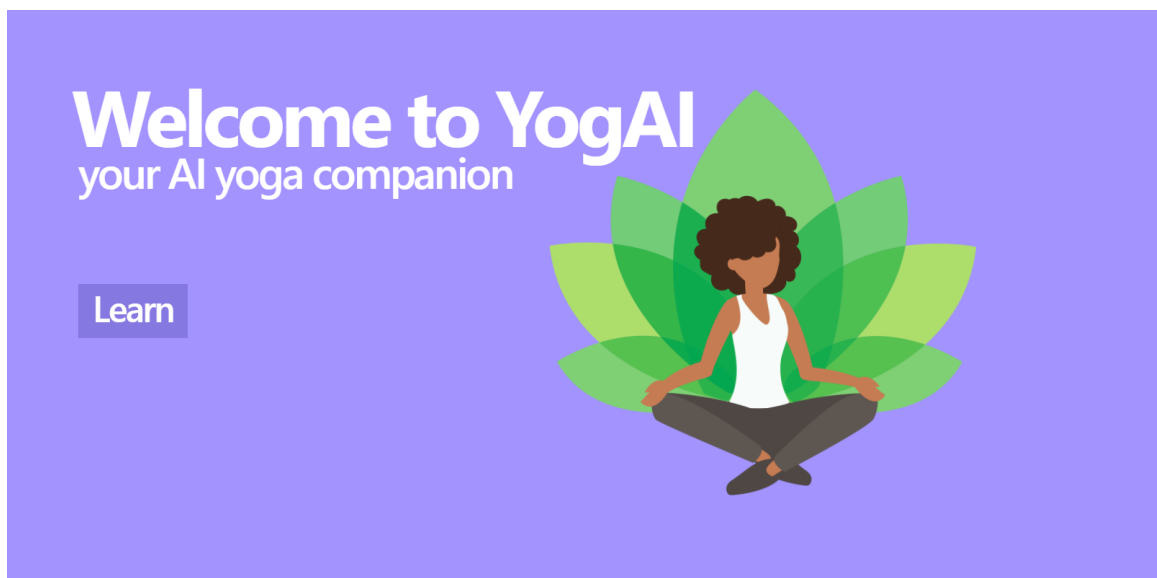


Fig 10. 2 Landing Page

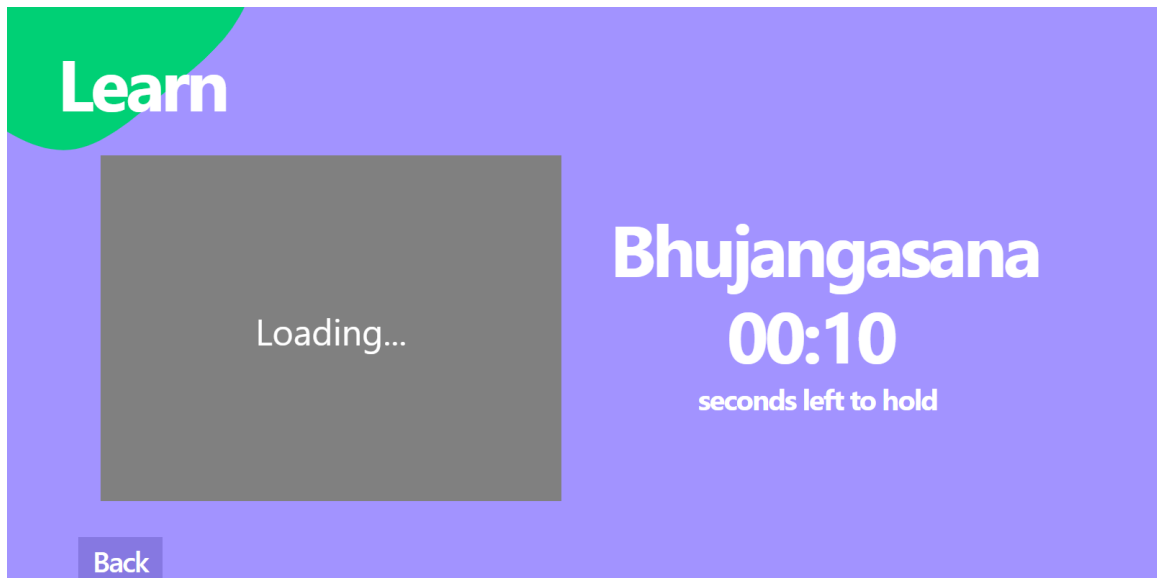


Fig 10. 3 Practice Page

Once the user clicks on “Learn” on the landing page of the website, the user is prompted to hold a pose for 10 seconds. If the user is successfully able to hold the pose with correct form, then a new pose is prompted. Otherwise the timer resets and the user has to hold it for 10 seconds again.

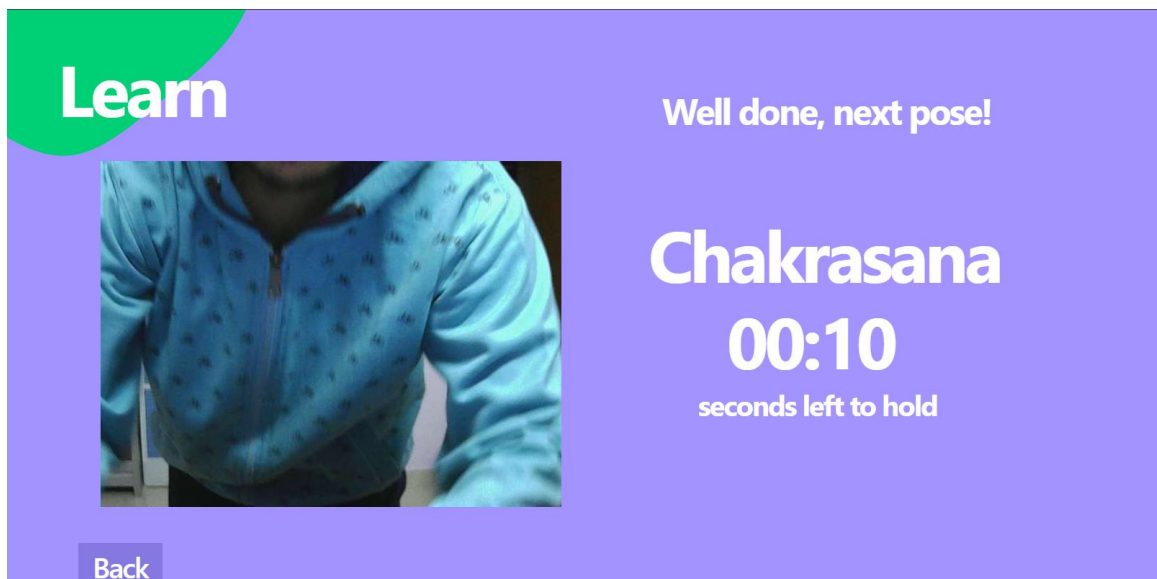


Fig 10. 4 Congratulations Message

CONCLUSION

Physical activity is vital to maintain good health. But, if the form is incorrect, it could lead to serious injuries. Practicing yoga at home for beginners could be dangerous if correct form is not followed. A lot of the popular form monitoring assistants require a lot of setup. This project aims to remove that barrier. This project runs entirely on the user's browser and can even be accessed on a mobile phone. The model can classify a pose and check if the form is correct. This can be used by the yoga teachers and the students to help with their yoga practice at home. Can be used by elderly people to practice yoga at home.

This can be further improved by using a much more robust dataset. The better the quality of the data, the better output can be expected. More number of poses and also better-quality data set can increase the accuracy of the model. A better way of finding wrong pose can be implemented in addition to the current model to make pose detection accessible to everyone working out from home. The current model is not limited to Yoga poses. It can be used in any field when the form of the person is important.

REFERENCES

- CODE AT (Version 1): <https://github.com/dhruvkp090/yogaPoseEstimator>
- CODE AT (Version 2): https://github.com/dhruvkp090/poseEstimator_v2
- https://physio-pedia.com/Physical_Activity_and_COVID-19#cite_note-4
- The Conversation. How to stay fit and active at home during the coronavirus self-isolation. Published on 25 March 2020. Available from <https://theconversation.com/how-to-stay-fit-and-active-at-home-during-the-coronavirus-self-isolation-134044>
- World Health Organisation. Be Active During COVID-19. Available from <https://www.who.int/news-room/q-a-detail/be-active-during-covid-19>
- Nieman DC, Henson DA, Austin MD, et al. Upper respiratory tract infection is reduced in physically fit and active adults. British Journal of Sports Medicine 2011;45:987-992.
- <https://p5js.org/get-started/>
- <https://www.geeksforgeeks.org/p5-js-introduction/>
- <https://learn.ml5js.org/#/>
- <https://ml5js.org/>
- <https://www.freecodecamp.org/news/get-to-know-tensorflow-js-in-7-minutes-afcd0dfd3d2f/>
- <https://www.smashingmagazine.com/2019/09/machine-learning-front-end-developers-tensorflowjs/>
- <https://codelabs.developers.google.com/codelabs/tensorflowjs-object-detection#1>
- <https://medium.com/tensorflow/real-time-human-pose-estimation-in-the-browser-with-tensorflow-js-7dd0bc881cd5>
- https://www.tensorflow.org/lite/models/pose_estimation/overview
- <https://github.com/tensorflow/tfjs-models/tree/master/posenet>
- <https://tallyfy.com/uml-diagram/>
- <https://github.com/cris-maillo>
- <https://github.com/CodingTrain/website>
- <https://thecodingtrain.com/>
- https://en.wikipedia.org/wiki/Software_testing
- <https://www.investopedia.com/terms/n/neuralnetwork.asp#:~:text=A%20neural%20network%20is%20a,organic%20or%20artificial%20in%20nature.>
- <https://towardsdatascience.com/understanding-neural-networks-19020b758230>
- A. Kendall, M. Grimes and R. Cipolla, "PoseNet: A Convolutional Network for Real-Time 6-DOF Camera Relocalization," 2015 IEEE International Conference on Computer Vision (ICCV), Santiago, 2015, pp. 2938-2946, doi: 10.1109/ICCV.2015.336.

•