

CNN BASED SYSTEM CONTROLLER USING HAND GESTURES

*A project report submitted in partial fulfillment of the Academic requirements
for the award of the Degree of*

**BACHELOR OF ENGINEERING
IN
INFORMATION TECHNOLOGY**

By

M.Sai Kiran Kumar

(2451-16-737-046)

D.Sai Pranav

(2451-16-737-054)

G.Vamshi Krishna

(2451-16-737-060)

Under the guidance of

Mr. D. B. V. Ravi Sankar
Associate Professor, Dept. of IT
MVSR Engineering College
Nadargul, Hyderabad.



DEPARTMENT OF INFORMATION TECHNOLOGY
MATURI VENKATA SUBBA RAO ENGINEERING COLLEGE
(Affiliated to Osmania University, Hyderabad. Recognized by AICTE)
Nadargul, Saroornagar Mandal, Hyderabad-501510
2019-20

CERTIFICATE

This is to certify that the project entitled, “**CNN Based System Controller Using Hand Gestures**” is being submitted by **M.SAI KIRAN KUMAR[Roll No. 2451-16-737-046]**, **D.SAI PRANAV[Roll No. 2451-16-737-054]** and **G.VAMSHI KRISHNA[Roll No. 2451-16-737-060]** in partial fulfillment of the academic requirements for the award of the degree of Bachelor of Engineering in Information Technology to the Osmania University, is a record of bona fide work carried out by them under my guidance and supervision. The results obtained in the project have not been submitted to any other industry or institute for the award of any degree.

Internal Guide

Head of the Department

ACKNOWLEDGEMENTS

We with extreme jubilation and deepest gratitude, would like to thank our guide **Mr. D.B.V. Ravi Sankar, Associate Professor**, Department of Information Technology, MVSR Engineering College, for his constant encouragement and facilities provided to us to complete our project in time.

With immense pleasure, we record our deep sense of gratitude to our beloved Head of the department **V.Ashwini Kumar**, Department of Information Technology, MVSR Engineering College, for permitting us to carry out this project.

We are sincerely thankful to **Dr.G.Kanaka Durga**, Principal, M V S R Engineering College for her inspiration, continuous support and facilities provided us to do this project.

We would like to extend our gratitude to Dr.Ch.Samson, Professor, Department of Information Technology, M V S R Engineering College, for his valuable suggestions during project reviews and timely help during the course of the project.

We express, from the bottom of our heart, deepest gratitude to our parents and family for the support, dedication, comprehension and love.

Finally, we express our heartfelt thanks to each and everyone who directly and indirectly helped us in successful completion of this project work.

M.Sai kiran Kumar
(2451-16-737-046)

D.Sai Pranav
(2451-16-737-054)

G.Vamshi Krishna
(2451-16-737-060)

DEPARTMENT OF INFORMATION TECHNOLOGY

Vision :

To impart technical education producing competent and socially responsible engineering professionals in the field of Information Technology.

Mission:

M1. To make teaching learning process effective and stimulating.

M2. To provide adequate fundamental knowledge of sciences and Information Technology with positive attitude.

M3. To create an environment that enhances skills and technologies required for industry.

M4. To encourage creativity and innovation for solving real world problems.

M5. To cultivate professional ethics in students and inculcate a sense of responsibility towards society

Program Educational Objectives: After 3 to 4 years of graduation, graduates of the Information Technology program will:

- I. Apply knowledge of mathematics and Information Technology to analyze, design and implement solutions for real world problems in core or in multidisciplinary areas.
- II. Communicate effectively, work in a team, practice professional ethics and apply knowledge of computing technologies for societal development.
- III. Engage in Professional development or postgraduate education to be a life-long learner.

PROGRAM OUTCOMES (POs)

Engineering Graduates will be able to:

1. **Engineering knowledge:** Apply the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization to the solution of complex engineering problems.
2. **Problem analysis:** Identify, formulate, review research literature, and analyze complex engineering problems reaching substantiated conclusions using first principles of mathematics, natural sciences, and engineering sciences.
3. **Design/development of solutions:** Design solutions for complex engineering problems and design system components or processes that meet the specified needs with appropriate consideration for the public health and safety, and the cultural, societal, and environmental considerations.
4. **Conduct investigations of complex problems:** Use research-based knowledge and research methods including design of experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusions.

5. **Modern tool usage:** Create, select, and apply appropriate techniques, resources, and modern engineering and IT tools including prediction and modeling to complex engineering activities with an understanding of the limitations.
6. **The engineer and society:** Apply reasoning informed by the contextual knowledge to assess societal, health, safety, legal and cultural issues and the consequent responsibilities relevant to the professional engineering practice.
7. **Environment and sustainability:** Understand the impact of the professional engineering solutions in societal and environmental contexts, and demonstrate the knowledge of, and need for sustainable development.
8. **Ethics:** Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice.
9. **Individual and team work:** Function effectively as an individual, and as a member or leader in diverse teams, and in multidisciplinary settings.
10. **Communication:** Communicate effectively on complex engineering activities with the engineering community and with society at large, such as, being able to comprehend and write effective reports and design documentation, make effective presentations, and give and receive clear instructions.
11. **Project management and finance:** Demonstrate knowledge and understanding of the engineering and management principles and apply these to one's own work, as a member and leader in a team, to manage projects and in multidisciplinary environments.
12. **Life-long learning:** Recognize the need for, and have the preparation and ability to engage in independent and life-long learning in the broadest context of technological change.

Program Specific Outcomes (PSOs):

(13) **Hardware design:** An ability to analyze, design, simulate and implement computer hardware / software and use basic analog/digital circuits, VLSI design for various computing and communication system applications.

(14) **Software design:** An ability to analyze a problem, design algorithm, identify and define the computing requirements appropriate to its solution and implement the same.

MAIN PROJECT COURSE OUTCOMES:

- Demonstrate the ability to synthesize and apply the knowledge and skills acquired in the academic program to the real world problems.
- Evaluate different solutions based on economic and technical feasibility.
- Effectively plan a project and confidently perform all aspects of project management.
- Demonstrate effective written and oral communication skills

ABSTRACT

For handling various devices around us a lot of mechanical work need to be performed. To reduce this impact on the human beings we have designed a controller in which objects can be managed by hand gestures. Hand gestures are very common in daily life, it is used as a medium of communication between people to describe any aspect. Thus, we have used this technique to control the objects in between us. There are many other ways to control the devices but this method helps human to control the devices in a safe and comfortable user interface for diverse applications.

The main aim of the project is to take the input as a hand gesture and produce the output to identify the devices like light, fan, motor, doors etc.. to be controlled in the surroundings and to create the modern way of lifestyle to the people. To develop such a user interface we have used feed forward artificial neural networks such as Convolution Neural Network (CNN) with keras in python. This model is used to classify the hand gesture that has sent as input to the system and give a required output to control the device with the help of hand gesture.

CONTENTS

TOPIC	PAGE NO
1. Introduction	1
2. Literature Survey	5
3. Theory	8
4. Design	21
5. Implementation and Testing	31
6. Results and Discussion	36
7. Conclusion	37
8. Future Scope	38
9. References	39
Appendices	
A.1 List of Figures	41
A.2 Listing of Code	42

Chapter 1

Introduction

Image classification, in the context of machine vision, is the ability of software to identify objects, places, people, writing and actions in images. Computers can use machine vision technologies in combination with a camera and artificial intelligence software to achieve image recognition.

Image classification is a supervised learning problem: define a set of target classes (objects to identify in images), and train a model to recognize them using labelled example photos. Early computer vision models relied on raw pixel data as the input to the model. However, as shown in Figure 1.1, raw pixel data alone doesn't provide a sufficiently stable representation to encompass the myriad variations of an object as captured in an image. The position of the object, background behind the object, ambient lighting, camera angle, and camera focus all can produce fluctuation in raw pixel data; these differences are significant enough that they cannot be corrected for by taking weighted averages of pixel RGB values.

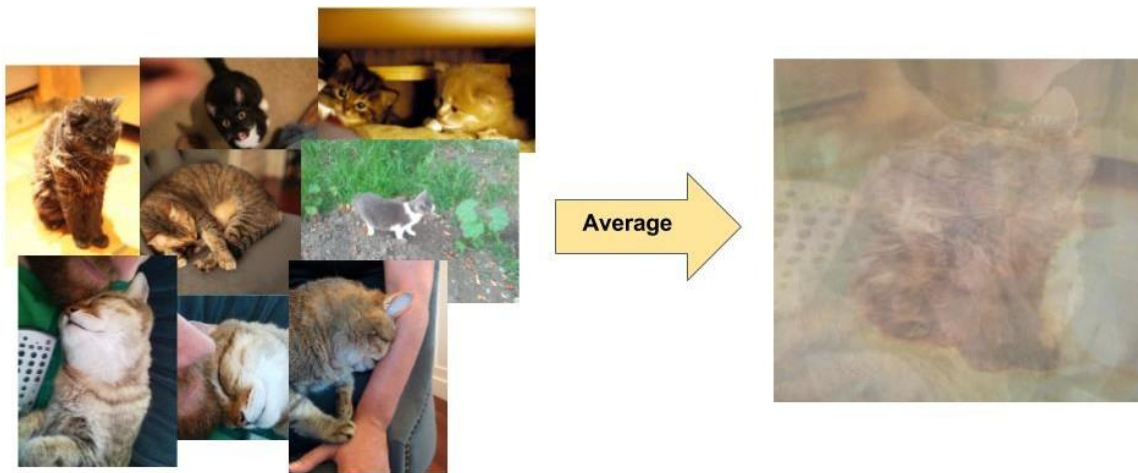


Figure 1.1. Left: Cats can be captured in a photo in a variety of poses, with different backdrops and lighting conditions. Right: averaging pixel data to account for this variety does not produce any meaningful information.

To model objects more flexibly, classic computer vision models added new features derived from pixel data, such as color histograms, textures, and shapes. The downside of this approach was that feature engineering became a real burden, as there were so many inputs to tweak. For a cat classifier, which colors were most relevant? How flexible should the shape definitions

be? Because features needed to be tuned so precisely, building robust models was quite challenging, and accuracy suffered.

While human and animal brains recognize objects with ease, computers have difficulty with the task. Software for image recognition requires deep machine learning algorithms. A breakthrough in building models for image classification came with the discovery that a convolutional neural network (CNN) could be used to progressively extract higher- and higher-level representations of the image content. Instead of pre-processing the data to derive features like textures and shapes, a CNN takes just the image's raw pixel data as input and "learns" how to extract these features, and ultimately infer what object they constitute.

As CNN is the best image classification approach to recognize an image so we have used this algorithm to recognize an hand gesture(action of an image) present in an image.

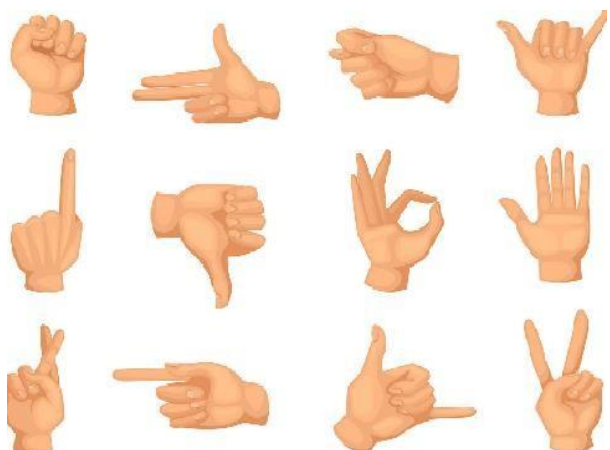


Fig1.2: Basic Hand gestures

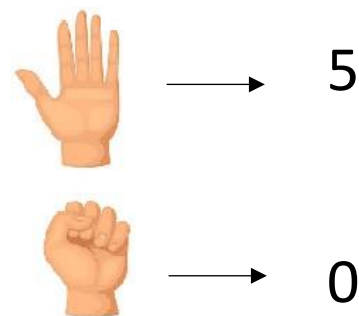


Fig1.3: Recognizing the gesture with label

Hand gestures are common way of communication among people. In the same context, Hand gestures provide a natural way for humans to interact with computers to perform a variety of different applications but there are several factors that can change the performance of hand gesture recognition algorithm. They are complexity of hand posture, size, structure of gesture and environment changes. Recent advances in deep Learning have significantly advanced the performance of image recognition systems. In particular, the deep convolutional neural network has demonstrated superior performance in image representation and classification, compared to conventional machine learning approaches.

Automatic hand gesture recognition using computer vision is important for various real-world applications such as Sign Language Recognition. Predominantly hand gesture recognition is come into existence for research in a sign language recognition due to wide spread of digital cameras.

In our project we have used hand gesture to control the devices in our surroundings. This way we can enhance the living style of people and reduce the work for people. Handling objects around us needs lot of mechanical work to be performed. It can help in building a safe and comfortable user interface for people.

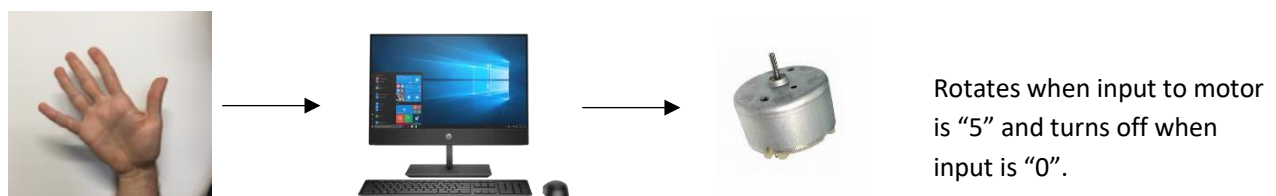


Fig 1.4: Brief idea of our project

In the above diagram as you can see the motor is controlled by giving gestures to the system. Similarly, we can control the home appliances in our home by using the above model.

To recognize the hand gesture we use Convolutional Neural Network algorithm of machine learning approaches. A specific kind of such a deep neural network is the convolutional network, which is commonly referred to as CNN or ConvNet. It's a deep, feed-forward artificial neural network. CNNs specifically are inspired by the biological visual cortex. The cortex has small regions of cells that are sensitive to the specific areas of the visual field. This idea was expanded by a captivating experiment done by Hubel and Wiesel in 1962 .

In this experiment, the researchers showed that some individual neurons in the brain activated or fired only in the presence of edges of a particular orientation like vertical or horizontal edges.

1.1 Problem Statement

There are many image recognition algorithms but they are not efficient and have less accuracy. The problem statement in this project is aimed “To recognize a hand gesture with very high accuracy using a CNN model”. The predicted hand gestures of this model are used to control the devices around us.

For the prototype purpose we have demonstrated with two labels:

1. 5 to ON the device
2. 0 to OFF the device.

1.2 Scope Of work

The main aim of the project is to provide modern way of lifestyle to people and provide safe, comfortable user interface for handling the devices. We have designed a CNN model to attain the modern way of living style to the people. The accuracy of the model is very high compared to the previously proposed models. The following are the deliverables that are expected of this project:

- Connecting the things around us with the help of software product.
- An algorithm which takes less time and performs less computations to recognize an image.
- It helps to improve the standard of living.
- It helps the person to operate anything from the long distances.
- It helps to operate any home appliances from anywhere.
- It provides safer environment to handle the devices.

1.3 Organizational Schematic of the project

This Report follows the following organizational Schematic:

- Chapter 2 Literature Survey :

This Chapter covers the aspect of looking at existing systems in the form of technical papers and analyzing those systems in order to generate motivation for the development new system.

- Chapter 3 Theory

The Fundamental Theory behind this project i.e. working principles, algorithms etc. and all the requirements that are necessary for the completion of the same are mentioned in this chapter.

- Chapter 4 Design Of CNN model

The essential architecture and training of a data set with example are discussed here.

- Chapter 5 Implementation and Testing of the System.

This chapter talks about the steps involved in the physical implementation of the system and the testing process adopted for the same.

- Chapter 6 Results and Discussion

This Chapter focuses on the results achieved by the system.

- Chapter 7 Conclusion

This Chapter talks about the overall realization of the proposed problem statement and the extent of the project on the whole in terms of applicability.

- Chapter 8 Future Scope

This Chapter talks about the extension of the applicability of the project in terms of additional features, increased computing power, reduced the size, development of an Integrated Circuit for the system etc.

Chapter 2

Literature Survey

Hand Gesture Recognition Using an Adapted Convolutional Neural Network with Data Augmentation

Ali A. Alani, Georgina Cosma, Aboozar Taherkhani, T.M McGinnity

Abstract:-

Hand gestures provide a natural way for humans to interact with computers to perform a variety of different applications. However, factors such as the complexity of hand gesture structures, differences in hand size, hand posture, and environmental illumination can influence the performance of hand gesture recognition algorithms. Recent advances in Deep Learning have significantly advanced the performance of image recognition systems. In particular, the Deep Convolutional Neural Network has demonstrated superior performance in image representation and classification, compared to conventional machine learning approaches. This paper proposes an Adapted Deep Convolutional Neural Network (ADCNN) suitable for hand gesture recognition tasks. Data augmentation is initially applied which shifts images both horizontally and vertically to an extent of 20% of the original dimensions randomly, in order to numerically increase the size of the dataset and to add the robustness needed for a deep learning approach. These images are input into the proposed ADCNN model which is empowered by the presence of network initialization (ReLU and Softmax) and L2 Regularization to eliminate the problem of data overfitting. With these modifications, the experimental results using the ADCNN model demonstrate that it is an effective method of increasing the performance of CNN for hand gesture recognition. The model was trained and tested using 3750 static hand gesture images, which incorporate variations in features such as scale, rotation, translation, illumination and noise. The proposed ADCNN was compared to a baseline Convolutional Neural Network and the results show that the proposed ADCNN achieved a classification recognition accuracy of 99.73%, and a 4% improvement over the baseline Convolutional Neural Network model (95.73%).

Published in: 2018 4th International Conference on Information Management (ICIM).

Date of Conference: 25-27 May 2018

Date Added to IEEE Xplore: 25 June 2018

INSPEC Accession Number: 17862325

DOI: 10.1109/INFOMAN.2018.8392660

Publisher: IEEE

From the above we have got an idea of convolutional neural network can be used to recognize the hand gesture but in this paper they used ADCNN with data augmentation and they used only static hand gestures to recognize an image. But in

our project we have used dynamic hand gestures in place of static hand gestures and the accuracy of this existing system is less compared to the project we have proposed.

HOME APPLIANCE IDENTIFICATION FOR NILM SYSTEMS BASED ON DEEP NEURAL NETWORKS

Deyvison de Paiva Penha and Adriana Rosa Garcez Castro

Abstract:-

This paper presents the proposal for the identification of residential equipment in non-intrusive load monitoring systems. The system is based on a Convolutional Neural Network to classify residential equipment. As inputs to the system, transient power signal data obtained at the time an equipment is connected in a residence is used. The methodology was developed using data from a public database (REED) that presents data collected at a low frequency (1 Hz). The results obtained in the test database indicate that the proposed system is able to carry out the identification task, and presented satisfactory results when compared with the results already presented in the literature for the problem in question.

Published in: 2018, Computer Science, International Journal of Artificial Intelligence & Applications

Date Added to IJAIA: March 2018 Volume no:2

DOI:10.5121/ijaia.2018.9206

Publisher: IJAIA

From the above paper we got an idea of connecting the home appliances to our system though the paper used to solve the problem in different context but we have taken the key point to connect the devices to our smartphone or computer to control the devices around us. The reduction and rationalization of electricity consumption are increasingly becoming priorities, not only for residential consumers, but also for electric power companies and government. Considering this concern, which is worldwide, research in Non-Intrusive Load Monitoring (NILM) has been emphasizing. A NILM system has as main objective to measure an aggregate load of a residence through a single sensor, placed in the central meter of the residence. From the aggregate load, measured over a period of time, it is possible, through specific software, to carry out an identification of the electric equipment in operation and obtain the individual consumption thereof, in addition to obtaining the operating hours of each equipment. This information can be used by residential consumers to take actions aimed at reducing and rationalizing their consumption, thus ensuring greater energy efficiency. Though above idea of author is

related to hardware and consumers electricity consumption but we got an idea of connecting the devices to computer to control them.

Hand Gesture Recognition with Convolution Neural Networks

Felix Zhan

Abstract:-

Hand gestures are the most common forms of communication and have great importance in our world. They can help in building safe and comfortable user interfaces for a multitude of applications. Various computer vision algorithms have employed color and depth camera for hand gesture recognition, but robust classification of gestures from different subjects is still challenging. I propose an algorithm for real-time hand gesture recognition using convolutional neural networks (CNNs). The proposed CNN achieves an average accuracy of 98.76% on the dataset comprising of 9 hand gestures and 500 images for each gesture.

Published in: 2019 IEEE 20th International Conference on Information Reuse and Integration for Data Science (IRI)

Date of Conference: 30 July-1 Aug. 2019

Date Added to IEEE *Xplore*: 19 September 2019

INSPEC Accession Number: 18995143

DOI: 10.1109/IRI.2019.00054

Publisher: IEEE

Conference Location: Los Angeles, CA, USA, USA

From the above paper we have got a greater insight of convolutional neural network model and how to improve our accuracy of the model. In this paper the author is using static images for recognition of hand gesture but we have used dynamic images to control the devices. The training of images of our project learned from this paper like how to train a static image in a CNN model.

Chapter 3

Theory

This chapter describes the requirements of software and hardware used in the project

PYCHARM IDE:

PyCharm is an integrated development environment (IDE) used in computer programming, specifically for the Python language. It is developed by the Czech company JetBrains.^[6] It provides code analysis, a graphical debugger, an integrated unit tester, integration with version control systems (VCSes), and supports web development with Django as well as Data Science with Anaconda.

3.1 Software requirements

3.1.1 Python

Python is a general-purpose interpreted, interactive, object-oriented, and high-level programming language. It was created by Guido van Rossum during 1985- 1990. Like Perl, Python source code is also available under the GNU General Public License (GPL).

Python is a high-level, interpreted, interactive and object-oriented scripting language. Python is designed to be highly readable. It uses English keywords frequently whereas other languages use punctuation, and it has fewer syntactical constructions than other languages.

- Python is Interpreted – Python is processed at runtime by the interpreter. You do not need to compile your program before executing it. This is similar to PERL and PHP.
- Python is Interactive – you can actually sit at a Python prompt and interact with the interpreter directly to write your programs.
- Python is Object-Oriented – Python supports Object-Oriented style or technique of programming that encapsulates code within objects.
- Python is a Beginner's Language – Python is a great language for the beginner-level programmers and supports the development of a wide range of applications from simple text processing to WWW browsers to games.

Python's features include

- Easy-to-learn – Python has few keywords, simple structure, and a clearly defined syntax. This allows the student to pick up the language quickly.
- Easy-to-read – Python code is more clearly defined and visible to the eyes.
- Easy-to-maintain – Python's source code is fairly easy-to-maintain.
- A broad standard library – Python's bulk of the library is very portable and cross-platform compatible with UNIX, Windows, and Macintosh.
- Interactive Mode – Python has support for an interactive mode which allows interactive testing and debugging of snippets of code.
- Portable – Python can run on a wide variety of hardware platforms and has the same interface on all platforms.
- Extendable – you can add low-level modules to the Python interpreter. These modules enable programmers to add to or customize their tools to be more efficient.
- Databases – Python provides interfaces to all major commercial databases.
- GUI Programming – Python supports GUI applications that can be created and ported to many system calls, libraries and windows systems, such as Windows MFC, Macintosh, and the X Window system of Unix.
- Scalable – Python provides a better structure and support for large programs than shell scripting

3.1.2 Tensor flow

TensorFlow is the second machine learning framework that Google created and used to design, build, and train deep learning models. You can use the TensorFlow library to do numerical computations, which in itself doesn't seem all too special, but these computations are done with data flow graphs. In these graphs, nodes represent mathematical operations, while the edges represent the data, which usually are multidimensional data arrays or tensors, that are communicated between these edges.

The name “TensorFlow” is derived from the operations which neural networks perform on multidimensional data arrays or tensors! It’s literally a flow of tensors.

Introducing Tensors

To understand tensors well, it’s good to have some working knowledge of linear algebra and vector calculus. You already read in the introduction that tensors are implemented in TensorFlow as multidimensional data arrays, but some more introduction is maybe needed in order to completely grasp tensors and their use in machine learning.

Plane Vectors

Before you go into plane vectors, it’s a good idea to shortly revise the concept of “vectors”; Vectors are special types of matrices, which are rectangular arrays of numbers. Because vectors are ordered collections of numbers, they are often seen as column matrices: they have just one column and a certain number of rows. In other terms, you could also consider vectors as scalar magnitudes that have been given a direction.

Remember: an example of a scalar is “5 meters” or “60 m/sec”, while a vector is, for example, “5 meters north” or “60 m/sec East”. The difference between these two is obviously that the vector has a direction. Nevertheless, these examples that you have seen up until now might seem far off from the vectors that you might encounter when you’re working with machine learning problems. This is normal; The length of a mathematical vector is a pure number: it is absolute. The direction, on the other hand, is relative: it is measured relative to some reference direction and has units of radians or degrees. You usually assume that the direction is positive and in counterclockwise rotation from the reference direction.

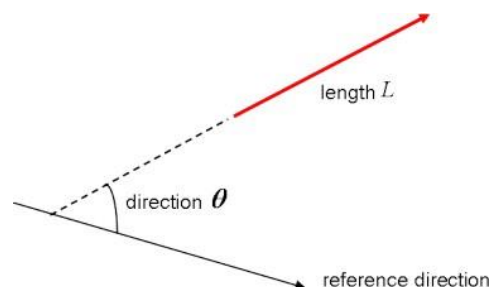


Fig:3.1 Vector direction

Visually, of course, you represent vectors as arrows, as you can see in the picture above. This means that you can consider vectors also as arrows that have direction and length. The

direction is indicated by the arrow's head, while the length is indicated by the length of the arrow.

Plane vectors are the most straightforward setup of tensors. They are much like regular vectors as you have seen above, with the sole difference that they find themselves in a vector space. To understand this better, let's start with an example: you have a vector that is 2×1 . This means that the vector belongs to the set of real numbers that come paired two at a time. Or, stated differently, they are part of two-space. In such cases, you can represent vectors on the coordinate (x,y) plane with arrows or rays.

Working from this coordinate plane in a standard position where vectors have their endpoint at the origin $(0,0)$, you can derive the x coordinate by looking at the first row of the vector, while you'll find the y coordinate in the second row. Of course, this standard position doesn't always need to be maintained: vectors can move parallel to themselves in the plane without experiencing changes.

Note that similarly, for vectors that are of size 3×1 , you talk about the three-space. You can represent the vector as a three-dimensional figure with arrows pointing to positions in the vectors pace: they are drawn on the standard x , y and z axes.

It's nice to have these vectors and to represent them on the coordinate plane, but in essence, you have these vectors so that you can perform operations on them and one thing that can help you in doing this is by expressing your vectors as bases or unit vectors.

Unit vectors are vectors with a magnitude of one. You'll often recognize the unit vector by a lowercase letter with a circumflex, or "hat". Unit vectors will come in convenient if you want to express a 2-D or 3-D vector as a sum of two or three orthogonal components, such as the x - and y -axes, or the z -axis.

And when you are talking about expressing one vector, for example, as sums of components, you'll see that you're talking about component vectors, which are two or more vectors whose sum is that given vector.

Tensors

Next to plane vectors, also covectors and linear operators are two other cases that all three together have one thing in common: they are specific cases of tensors. You still remember how a vector was characterized in the previous section as scalar magnitudes that have been given a direction. A tensor, then, is the mathematical representation of a physical entity that may be characterized by magnitude and *multiple* directions.

And, just like you represent a scalar with a single number and a vector with a sequence of three numbers in a 3-dimensional space, for example, a tensor can be represented by an array of 3^R numbers in a 3-dimensional space.

The “R” in this notation represents the rank of the tensor: this means that in a 3-dimensional space, a second-rank tensor can be represented by 3 to the power of 2 or 9 numbers. In an N-dimensional space, scalars will still require only one number, while vectors will require N numbers, and tensors will require N^R numbers. This explains why you often hear that scalars are tensors of rank 0: since they have no direction, you can represent them with one number.

With this in mind, it’s relatively easy to recognize scalars, vectors, and tensors and to set them apart: scalars can be represented by a single number, vectors by an ordered set of numbers, and tensors by an array of numbers.

What makes tensors so unique is the combination of components and basis vectors: basis vectors transform one way between reference frames and the components transform in just such a way as to keep the combination between components and basis vectors the same.

3.1.3 Numpy

As the name gives away, a NumPy array is a central data structure of the numpy library. The library’s name is short for “Numeric Python” or “Numerical Python”.

In other words, NumPy is a Python library that is the core library for scientific computing in Python. It contains a collection of tools and techniques that can be used to solve on a computer mathematical models of problems in Science and Engineering. One of these tools is a high-performance multidimensional array object that is a powerful data structure for efficient

computation of arrays and matrices. To work with these arrays, there's a vast amount of high-level mathematical functions operate on these matrices and arrays.

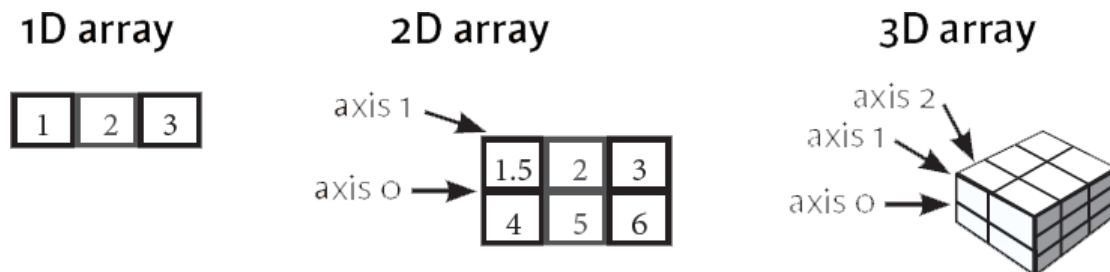


Fig:3.2 NumPy storage representation

The array that you see above is, as its name already suggested, a 2-dimensional array: you have rows and columns. The rows are indicated as the “axis 0”, while the columns are the “axis 1”. The number of the axis goes up accordingly with the number of the dimensions: in 3-D arrays, of which you have also seen an example in the previous code chunk, you’ll have an additional “axis 2”. Note that these axes are only valid for arrays that have at least 2 dimensions, as there is no point in having this for 1-D arrays; These axes will come in handy later when you’re manipulating the shape of your NumPy arrays.

The most important object defined in NumPy is an N-dimensional array type called **ndarray**. It describes the collection of items of the same type. Items in the collection can be accessed using a zero-based index.

Every item in an ndarray takes the same size of block in the memory. Each element in ndarray is an object of data-type object (called **dtype**).

Any item extracted from ndarray object (by slicing) is represented by a Python object of one of array scalar types. The following diagram shows a relationship between ndarray, data type object (dtype) and array scalar type –

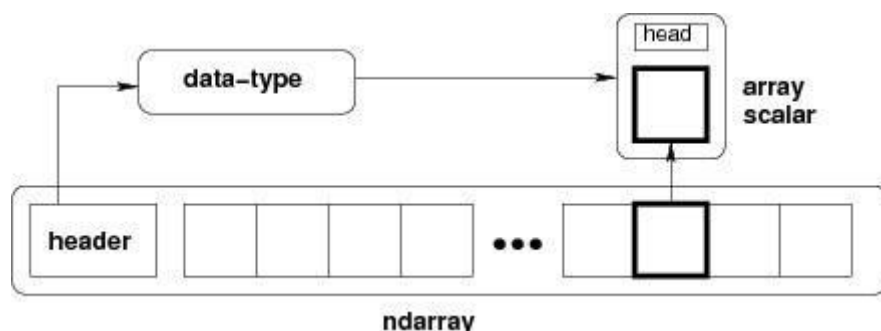


Fig:3.3 ndarray

An instance of ndarray class can be constructed by different array creation routines described later in the tutorial. The basic ndarray is created using an array function in NumPy as follows
—

`numpy.array` //Syntax to create an array using numpy

It creates an ndarray from any object exposing array interface, or from any method that returns an array.

3.1.4 Arduino IDE

Arduino is an open-source electronics platform based on easy-to-use hardware and software. Arduino boards are able to read inputs - light on a sensor, a finger on a button, or a Twitter message - and turn it into an output - activating a motor, turning on an LED, publishing something online. You can tell your board what to do by sending a set of instructions to the microcontroller on the board. To do so you use the Arduino programming language (based on Wiring), and the Arduino Software (IDE), based on Processing.

Over the years Arduino has been the brain of thousands of projects, from everyday objects to complex scientific instruments. A worldwide community of makers - students, hobbyists, artists, programmers, and professionals - has gathered around this open-source platform, their contributions have added up to an incredible amount of accessible knowledge that can be of great help to novices and experts alike.

Arduino was born at the Ivrea Interaction Design Institute as an easy tool for fast prototyping, aimed at students without a background in electronics and programming. As soon as it reached a wider community, the Arduino board started changing to adapt to new needs and challenges, differentiating its offer from simple 8-bit boards to products for IoT applications, wearable, 3D printing, and embedded environments. All Arduino boards are completely open-source, empowering users to build them independently and eventually adapt them to their particular needs. The software, too, is open-source, and it is growing through the contributions of users worldwide.

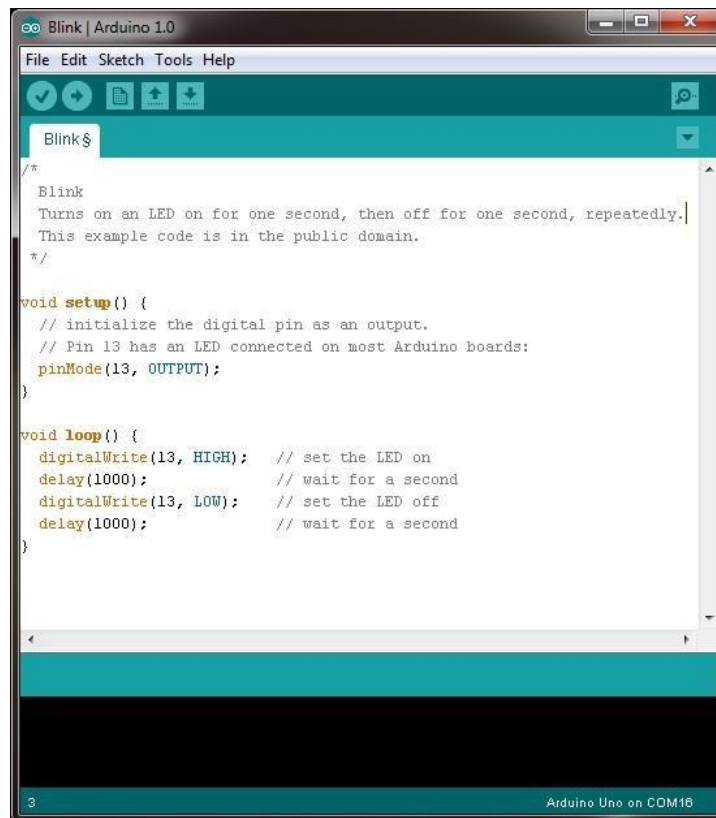


Fig 3.4 :Basic arduino code

3.2 Hardware Requirements

3.2.1 Arduino uno

The Arduino Mega 2560 is a microcontroller board based on the ATmega2560 (datasheet). It has 54 digital input/output pins (of which 14 can be used as PWM outputs), 16 analog inputs, 4 UARTs (hardware serial ports), a 16 MHz crystal oscillator, a USB connection, a power jack, an ICSP header, and a reset button. It contains everything needed to support the microcontroller; simply connect it to a computer with a USB cable or power it with a AC-to-DC adapter or battery to get started. The Mega is compatible with most shields designed for the Arduino Duemilanove or Diecimila.

Microcontroller:- ATmega2560

Operating Voltage:- 5V

Input Voltage (recommended):- 7-12V

Input Voltage (limits):- 6-20V

Digital I/O Pins:- 54 (of which 14 provide PWM output)

Analog Input Pins:- 16

DC Current per I/O Pin:- 40 mA

DC Current for 3.3V Pin:- 50 mA

Flash Memory:- 256 KB of which 8 KB used by bootloader

SRAM:- 8 KB

EEPROM:- 4 KB

Clock Speed:- 16 MHz



Fig3.5: Arduino ATmega2560

Power

The Arduino Mega can be powered via the USB connection or with an external power supply. The power source is selected automatically. External (non-USB) power can come either from an AC-to-DC adapter (wall-wart) or battery. The adapter can be connected by plugging a 2.1mm center-positive plug into the board's power jack. Leads from a battery can be inserted in the Gnd and Vin pin headers of the POWER connector. The board can operate on an external supply of 6 to 20 volts. If supplied with less than 7V, however, the 5V pin may supply less than five volts and the board may be unstable. If using more than 12V, the voltage regulator may overheat and damage the board. The recommended range is 7 to 12 volts. The Mega2560 differs from all preceding boards in that it does not use the FTDI USB-to-serial driver chip. Instead, it features the Atmega8U2 programmed as a USB-to-serial converter.

The power pins are as follows:

- VIN. The input voltage to the Arduino board when it's using an external power source (as opposed to 5 volts from the USB connection or other regulated power

source). You can supply voltage through this pin, or, if supplying voltage via the power jack, access it through this pin.

- 5V. The regulated power supply used to power the microcontroller and other components on the board. This can come either from VIN via an on-board regulator, or be supplied by USB or another regulated 5V supply.
- 3V3. A 3.3 volt supply generated by the on-board regulator. Maximum current draw is 50 mA.
- GND. Ground pins

Memory

The ATmega2560 has 256 KB of flash memory for storing code (of which 8 KB is used for the bootloader), 8 KB of SRAM and 4 KB of EEPROM (which can be read and written with the EEPROM library).

Input and Output

Each of the 54 digital pins on the Mega can be used as an input or output, using `pinMode()`, `digitalWrite()`, and `digitalRead()` functions. They operate at 5 volts. Each pin can provide or receive a maximum of 40 mA and has an internal pull-up resistor (disconnected by default) of 20-50 kOhms. In addition, some pins have specialized functions:

- Serial: 0 (RX) and 1 (TX); Serial 1: 19 (RX) and 18 (TX); Serial 2: 17 (RX) and 16 (TX); Serial 3: 15 (RX) and 14 (TX). Used to receive (RX) and transmit (TX) TTL serial data. Pins 0 and 1 are also connected to the corresponding pins of the ATmega8U2 USB-to-TTL Serial chip.
- External Interrupts: 2 (interrupt 0), 3 (interrupt 1), 18 (interrupt 5), 19 (interrupt 4), 20 (interrupt 3), and 21 (interrupt 2). These pins can be configured to trigger an interrupt on a low value, a rising or falling edge, or a change in value. See the `attachInterrupt()` function for details.
- PWM: 0 to 13. Provide 8-bit PWM output with the `analogWrite()` function.
- SPI: 50 (MISO), 51 (MOSI), 52 (SCK), 53 (SS). These pins support SPI communication using the SPI library. The SPI pins are also broken out on the ICSP header, which is physically compatible with the Uno, Duemilanove and Diecimila.

- LED: 13. There is a built-in LED connected to digital pin 13. When the pin is HIGH value, the LED is on, when the pin is LOW, it's off.
- I2C: 20 (SDA) and 21 (SCL). Support I2C (TWI) communication using the Wire library (documentation on the Wiring website). Note that these pins are not in the same location as the I2C pins on the Duemilanove or Diecimila.

The Mega2560 has 16 analog inputs, each of which provide 10 bits of resolution (i.e. 1024 different values). By default they measure from ground to 5 volts, though it is possible to change the upper end of their range using the AREF pin and `analogReference()` function. There are a couple of other pins on the board:

- AREF. Reference voltage for the analog inputs. Used with `analogReference()`.
- Reset. Bring this line LOW to reset the microcontroller. Typically used to add a reset button to shields which block the one on the board.

Communication

The Arduino Mega2560 has a number of facilities for communicating with a computer, another Arduino, or other microcontrollers. The ATmega2560 provides four hardware UARTs for TTL (5V) serial communication. An ATmega8U2 on the board channels one of these over USB and provides a virtual com port to software on the computer (Windows machines will need a .inf file, but OSX and Linux machines will recognize the board as a COM port automatically). The Arduino software includes a serial monitor which allows simple textual data to be sent to and from the board. The RX and TX LEDs on the board will flash when data is being transmitted via the ATmega8U2 chip and USB connection to the computer (but not for serial communication on pins 0 and 1). A `SoftwareSerial` library allows for serial communication on any of the Mega2560's digital pins. The ATmega2560 also supports I2C (TWI) and SPI communication. The Arduino software includes a `Wire` library to simplify use of the I2C bus; see the documentation on the Wiring website for details. For SPI communication, use the `SPI` library.

USB Overcurrent Protection

The Arduino Mega2560 has a resettable polyfuse that protects your computer's USB ports from shorts and overcurrent. Although most computers provide their own internal protection, the fuse provides an extra layer of protection. If more than 500 mA is

applied to the USB port, the fuse will automatically break the connection until the short or overload is removed.

USB 2.0 cable type A/B

Use it to connect Arduino Uno, Arduino Mega 2560, Arduino 101 or any board with the USB female A port of your computer.

Cable length is approximately 178cm. Cable color and shape may vary slightly from image as our stock rotates.

If you want to have a closer look to USB cables and standards check the USB cable pinouts referral page on pinouts.ru.



Fig:3.6 USB cable type A/B Standard USB 2.0 cable.

3.2.2 MOTOR

An **electric motor** is an electrical machine that converts electrical energy into mechanical energy. Most electric motors operate through the interaction between the motor's magnetic field and electric current in a wire winding to generate force in the form of torque applied on the motor's shaft. Electric motors can be powered by direct current (DC) sources, such as from batteries, motor vehicles or rectifiers, or by alternating current (AC) sources, such as a power grid, inverters or electrical generators. An electric generator is mechanically identical to an electric motor, but operates with a reversed flow of power, converting mechanical energy into electrical energy.. Electric motors are found in industrial fans, blowers and pumps, machine tools, household appliances, power tools and disk drives. Small motors may be found in electric watches.



Fig3.7: Electric Motor

3.3 Confusion Matrix

A confusion matrix is a table that is often used to **describe the performance of a classification model** (or "classifier") on a set of test data for which the true values are known. The confusion matrix itself is relatively simple to understand, but the related terminology can be confusing.

- **True Positives (TP):** These are cases in which both the predicted and actual values are yes.
- **True Negatives (TN):** These are the cases in which both the predicted and actual values are no.
- **False Positives (FP):** These are the cases in which the predicted value is yes and the actual value is no.
- **False Negatives (FN):** These are the cases in which the predicted value is no and the actual value is yes.

List of rates that are often computed from a confusion matrix for a binary classifier:

Total=TP+TN+FP+FN

- **Accuracy:** Overall, how often is the classifier correct?
 - $(TP+TN)/\text{total}$
- **Misclassification Rate:** Overall, how often is it wrong?
 - $(FP+FN)/\text{total}$
 - equivalent to 1 minus Accuracy
 - also known as "Error Rate"
- **True Positive Rate:** When it's actually yes, how often does it predict yes?
 - $TP/\text{actual yes} = 100/105 = 0.95$
 - also known as "Sensitivity" or "Recall"
- **False Positive Rate:** When it's actually no, how often does it predict yes?
 - $FP/\text{actual no} = 10/60 = 0.17$
- **True Negative Rate:** When it's actually no, how often does it predict no?
 - $TN/\text{actual no} = 50/60 = 0.83$
 - equivalent to 1 minus False Positive Rate
 - also known as "Specificity"
- **Precision:** When it predicts yes, how often is it correct?
 - $TP/\text{predicted yes} = 100/110 = 0.91$
- **Prevalence:** How often does the yes condition actually occur in our sample?
 - $\text{actual yes}/\text{total} = 105/165 = 0.64$

Chapter 4

Design

The block diagram in Fig 4.1 describe the overall design followed in this project.

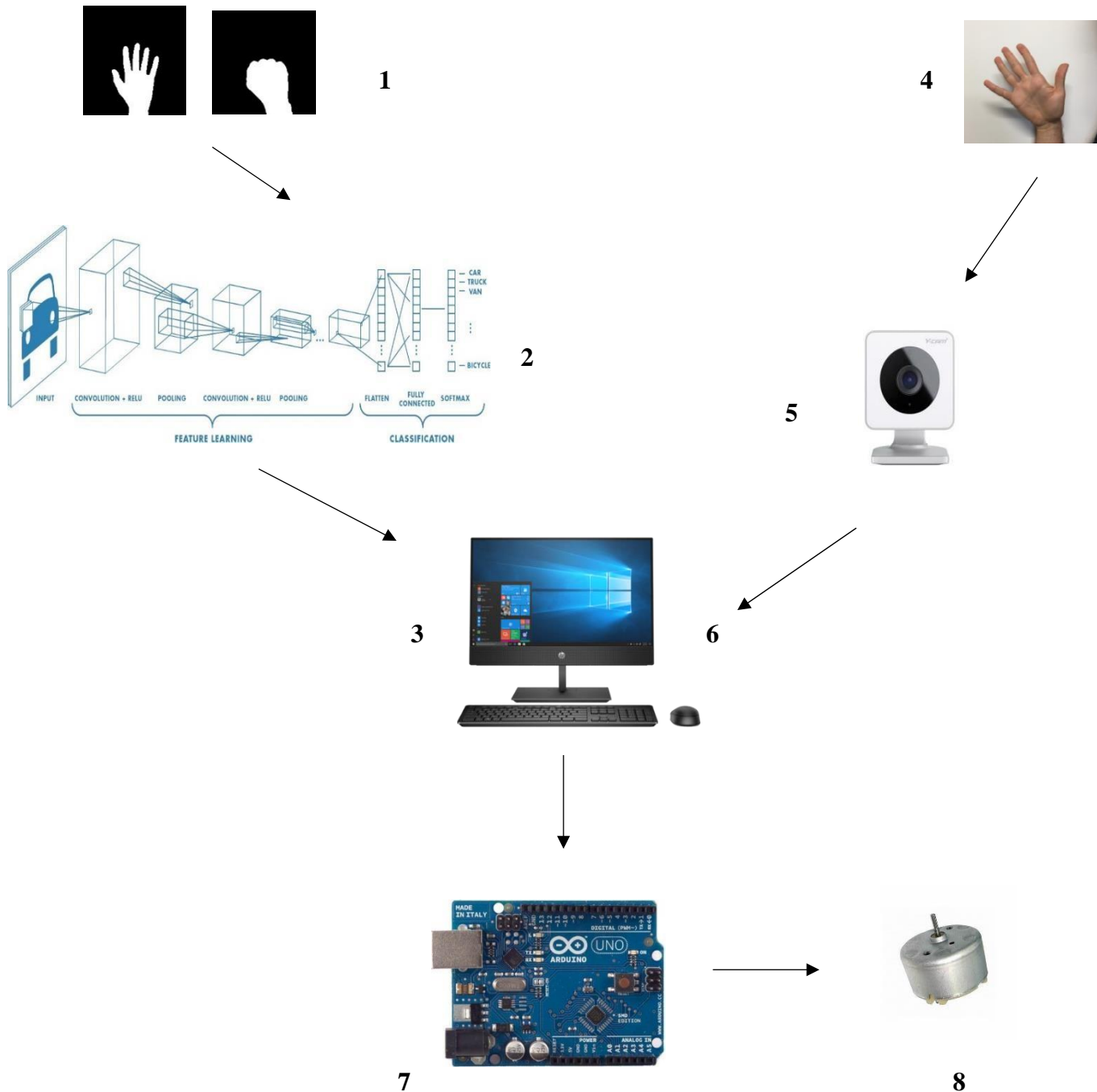


Fig4.1:Block Diagram

Image Storage by computer:

Color images require more storage space than grayscale images. Pixels in grayscale images need just one byte to indicate the intensity of gray needed to render the pixel on screen. It turns out that any color can be built using the correct combination of red, green, and blue. Thus, pixels in color images are represented by three values (r,g,b). The values indicate the intensity of red, green, and blue, respectively, needed to render the pixel on screen. The range of intensities is exactly the same as grayscale images - 0 means none of the color appears in the pixel and 255 indicates the highest level of the color is evident in the pixel. For example, the triple (128, 0, 128) would represent a medium purple while (255, 215, 0) represents gold.

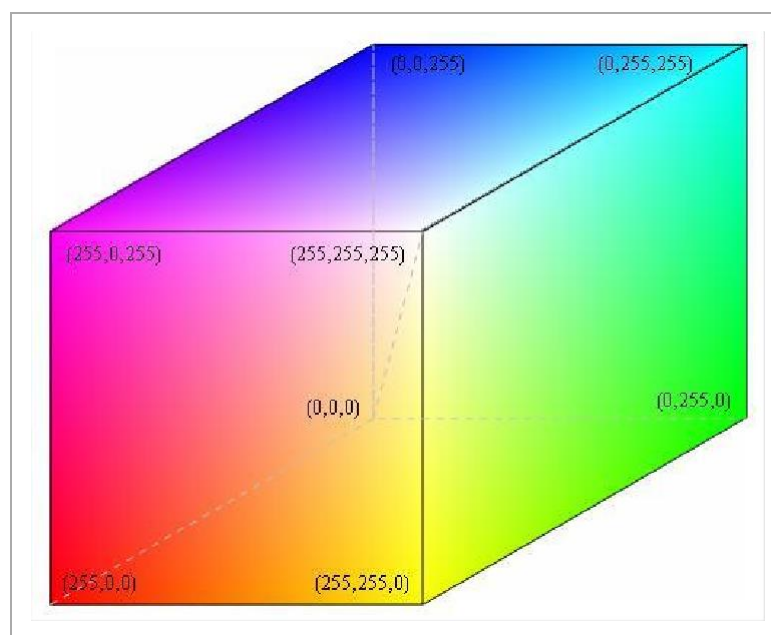


Fig4.2:3-dimensional space of pixel values

We can actually look at each of the red, green, and blue channels separately. The dimensions of the original image are 768 x 512 pixels. Since each pixel requires 3 bytes of information, we can store the image to disk in raw format using $768 \times 512 \times 3 = 1,179,648$ bytes. The bit stream would have length $1,179,648 \times 8 = 9,437,184$.

Gray Scale Images

The image at bottom is a thumbnail version of a *digital grayscale image*. The image is a rectangular tiling of fundamental elements called *pixels*. A pixel (short for **p**icture **e**lement) is a small block that represents the amount of gray intensity to be displayed for that particular portion of the image. For most images, pixel values are integers that range from 0 (black) to 255 (white). The 256 possible gray intensity values are shown below.

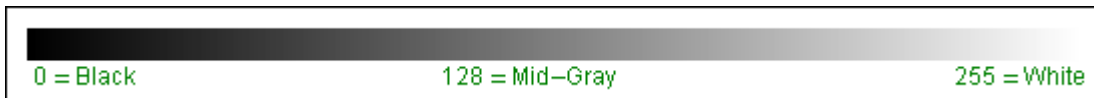


Fig4.3: The range of intensity values from 0 (black) to 255 (white).



Fig4.4: Thumbnail of a digital grayscale image.

Even if you view the full-size image, it is difficult to see the individual pixel intensities. This is the advantage of *high-resolution* images - more dots (or pixels) per inch (dpi) produces a finer image. Web applications often require images to have resolution 200dpi will printed matter such as books require 300dpi. The full-size image above is artificially enlarged - it is actually 2.016" x 3.024" or 508dpi. In order to get a better idea of pixel intensity values, we have taken the 15 x 15 pixel block that represents the bottom left-hand corner of the image and enlarged it. The images below show the enlarged block as well as their intensity values.

The fundamental unit on a computer is a *bit*. A bit (binary unit) takes either the value 0 or 1. The *byte* (the fundamental unit of storage on a PC) is composed of 8 bits. Since each bit takes on one of two values and 8 bits make a byte, we can use the multiplication principle to realize that there are $2^8 = 256$ possible bytes. We represent these bytes in base 2. For example, the intensity 125 can be written as

$$\begin{aligned}
 125 &= 64 + 32 + 16 + 8 + 4 + 1 \\
 &= 0 \times 128 + 1 \times 64 + 1 \times 32 + 1 \times 16 + 1 \times 8 + 1 \times 4 + 0 \times 2 + 1 \times 1 \\
 &= 0 \times 2^7 + 1 \times 2^6 + 1 \times 2^5 + 1 \times 2^4 + 1 \times 2^3 + 1 \times 2^2 + 0 \times 2^1 + 1 \times 2^0 \\
 &= 01111101_2
 \end{aligned}$$

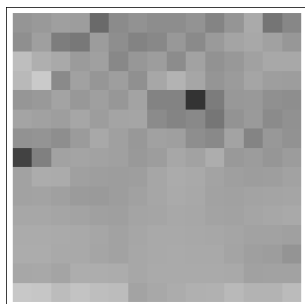


Fig4.5: The lower left 15 x 15 pixel portion of the image

```

150 154 160 157 106 140 147 142 141 147 132 150 171 117 136
144 159 125 121 157 143 132 136 153 138 155 164 169 162 152
190 175 169 155 161 136 152 158 141 162 147 153 161 168 169
185 203 139 161 151 159 145 167 179 167 150 155 165 159 158
151 153 163 152 160 152 164 131 131 51 124 152 154 145 143
164 162 158 167 157 164 166 139 132 138 119 148 154 139 146
147 148 143 155 169 160 152 161 159 143 138 163 132 152 146
66 129 163 165 163 161 154 157 167 162 174 153 156 151 156
162 173 172 161 158 158 159 167 171 169 164 159 158 159 162
163 164 161 155 155 158 161 167 171 168 162 162 163 164 166
167 167 165 163 160 160 164 166 169 168 164 163 165 167 170
172 171 170 170 166 163 166 170 169 168 167 165 163 163 160
173 172 170 169 166 163 167 169 170 170 170 165 160 157 148
167 168 165 173 173 172 167 170 170 171 171 169 162 163 162
200 198 189 196 191 188 163 168 172 177 177 186 180 180 188

```

Fig4.6: Pixel intensity values of the lower left 15 x 15 pixel portion of the image.

The American Standard Code for Information Exchange (ASCII) has assigned each of the 256 possible bytes to each keyboard character. And yes, there are 256 characters on a standard keyboard! If you are using a PC, open Notepad, enable NUMLOCK, hold down the ALT key and type 125 on the numeric keypad - you will see the ASCII character for the right brace } appear on screen.

If an image of size $M \times N$ pixels is stored in raw format on a web server or digital camera, then aside from some header information, the file consists of $M \times N \times 8$ bits (zeros and ones) where the rows of the image are concatenated to form one long *bit stream*. For our example image, the bit stream has length $768 \times 512 \times 8 = 3,145,728$ bits!

Training of Data Set Using CNN

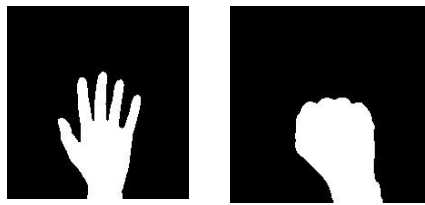
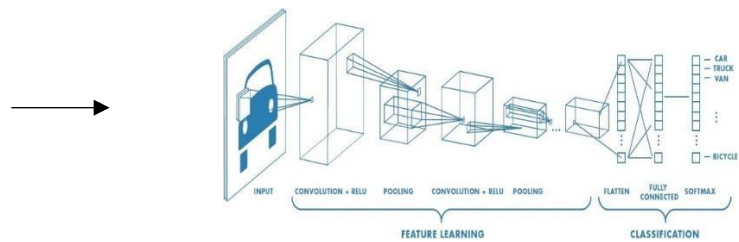


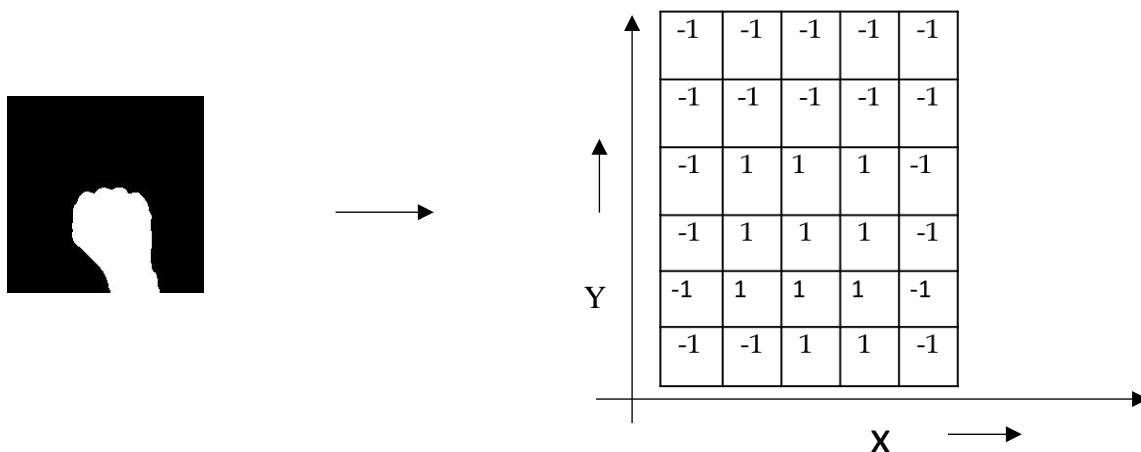
Fig4.7:Data Set



Models for each DataSet

As you have seen in the earlier topic how an image is stored in the computer's memory, it's in the form of pixel numbers. So to understand this model we have created cell values in such a way that it makes efficient to understand the a CNN algorithm.

We have defined two pixel values such that for an image, if an image of a hand is present then that region is positive 1, if not present we have given a value of -1. Although the value of -1 as a pixel value doesn't exist in the reality. To make clear the process of understanding the algorithm the -1 value is used.



In the above figure, we have taken the cell values of our image as -1 and 1 but internally the memory stores the pixel values according to the intensity of the colour in the gray scaled domain of 0 to 255.

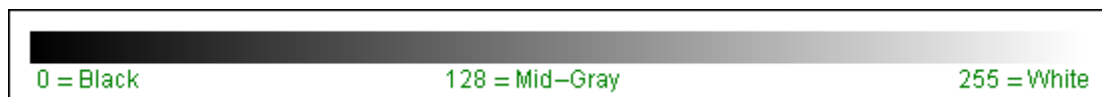


Fig4.8:Image intensity of gray scale image

And the above data set image is converted to gray scale image and then the pixel values of required intensity are assigned at that position.

Working Of CNN

How an image is stored in the memory is stated earlier but in this topic we deep dive and understand how Convolutional neural network make uses the image pixel values in the process of image recognition.

The CNN receives an input feature map: a three-dimensional matrix where the size of the first two dimensions corresponds to the length and width of the images in pixels. The size of the third dimension is 3 (corresponding to the 3 channels of a color image: red, green, and blue). The CNN comprises a stack of modules, each of which performs three operations.

CNN has 4 layers

- Convolution(Filtering the image)
- Relu Layer
- Pooling layers
- Fully connected layer

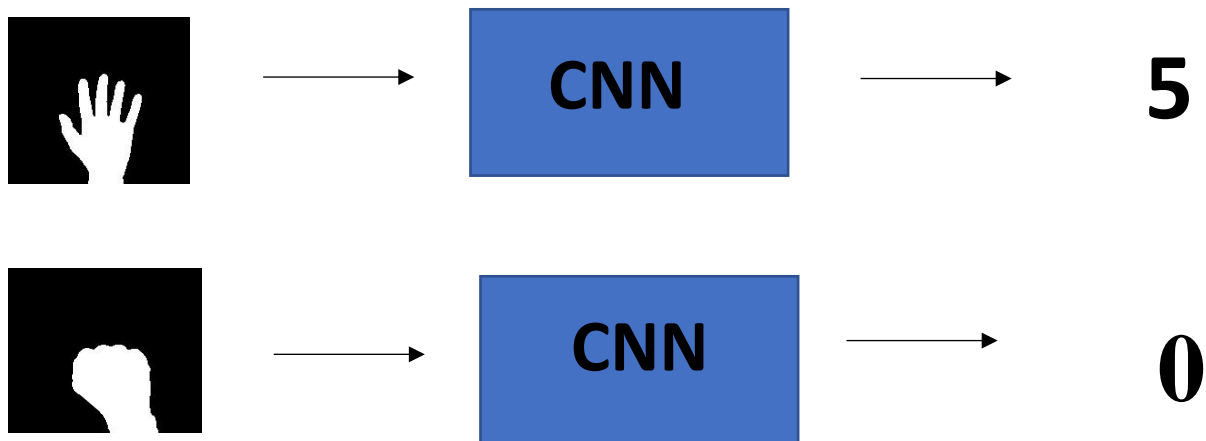


Fig4.9:Labeling of data sets

Convolutinal layer

The convolution layer computes the output of neurons that are connected to local regions or receptive fields in the input, each computing a dot product between their weights and a small receptive field to which they are connected to in the input volume. Each computation leads to extraction of a feature map from the input image. In other words, imagine you have an image represented as a 5x5 matrix of values, and you take a 3x3 matrix and slide that 3x3 window or kernel around the image. At each position of that matrix, you multiply the values of your 3x3 window by the values in the image that are currently being covered by the window. As a result, you'll get a single number that represents all the values in that window of the images. You use this layer to filtering: as the window moves over the image, you check for patterns in that section of the image. This works because of filters, which are multiplied by the values outputted by the convolution.

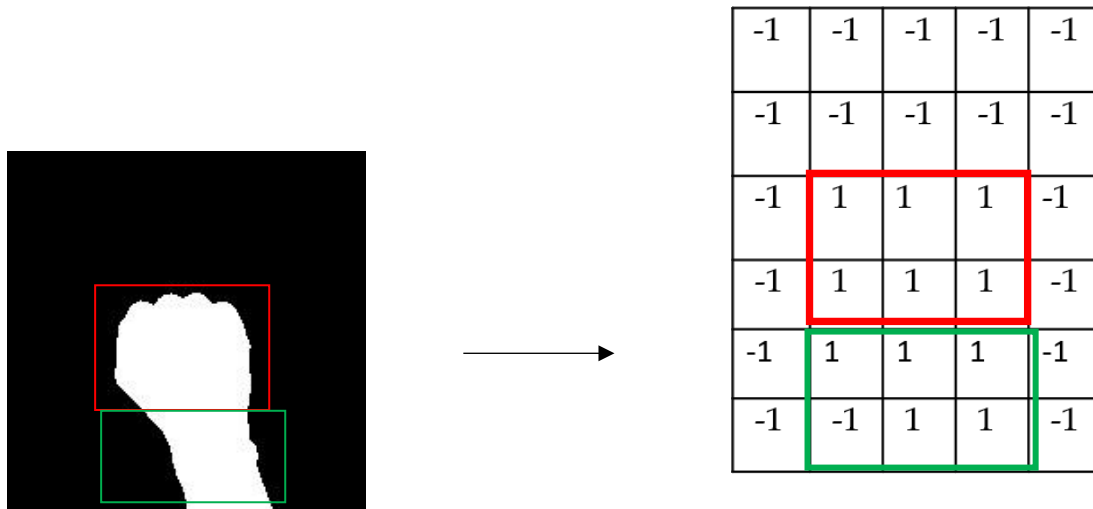


Fig4.10:Filtered images of a dataset

The algorithm first selects the regions where the image is present and then performs addition and multiplication with filtered images.

$$\begin{array}{|c|} \hline 1 \\ \hline \end{array}
 \begin{array}{|c|c|c|} \hline 1 & 1 & 1 \\ \hline 1 & 1 & 1 \\ \hline \end{array}
 \xrightarrow[6]{(1 \times 1 + 1 \times 1 + 1 \times 1 + 1 \times 1 + 1 \times 1 + 1 \times 1)} 1$$

$$\begin{array}{|c|} \hline 1 \\ \hline \end{array}
 \begin{array}{|c|c|c|} \hline 1 & 1 & 1 \\ \hline -1 & 1 & 1 \\ \hline \end{array}
 \rightarrow 0.66$$

By picking the region where image is spread the pixel values are multiplied by the filtered images and divided by the number of cells in filtered image to get a modified filtered image.

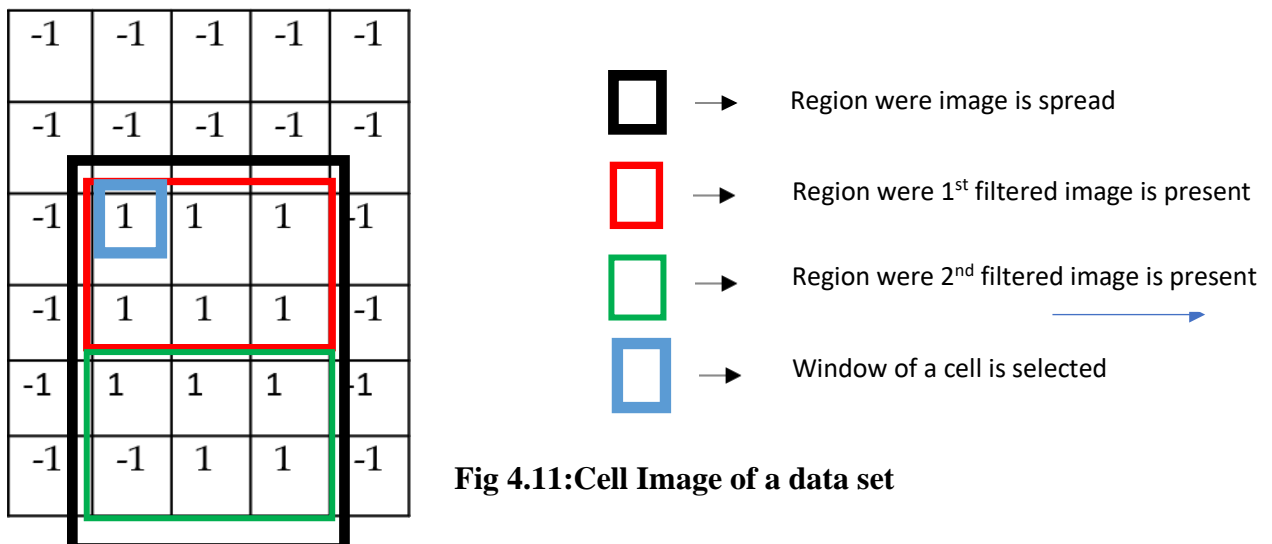


Fig 4.11:Cell Image of a data set

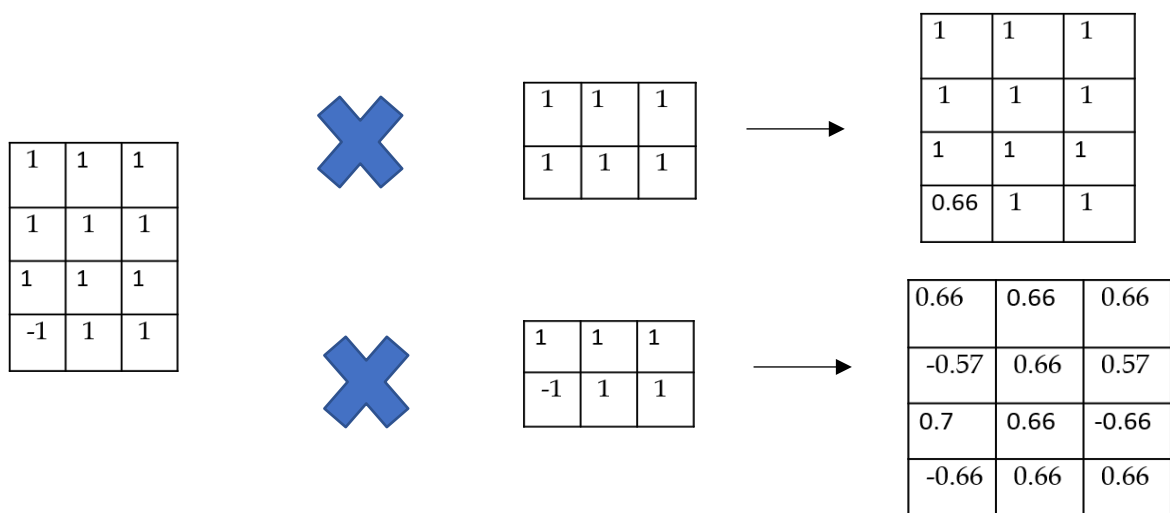
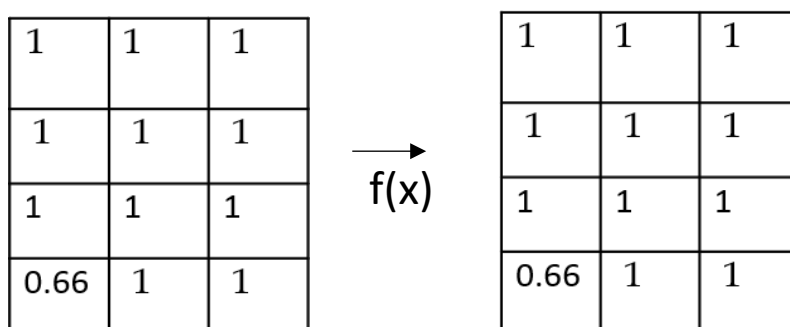
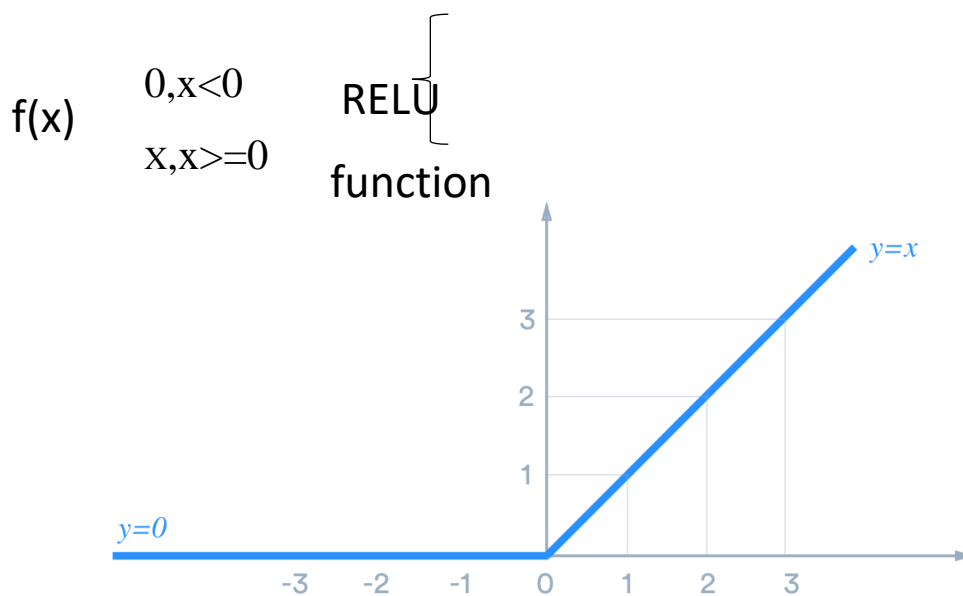


Fig4.12: output of convolution layer

Relu layer

Relu layer transform function only activates when an input is greater than 0 ,if the input value is less than 0 than output is 0.



0.66	0.66	0.66
-0.57	0.66	0.57
0.7	0.66	-0.66
-0.66	0.66	0.66

 $\xrightarrow{f(x)}$

0.66	0.66	0.66
0	0.66	0.57
0.7	0.66	0
0	0.66	0.66

Above cells are the output of the relu layer and the cells are sent to the next layer.

Pooling layer

The objective of subsampling is to get an input representation by reducing its dimensions, which helps in reducing overfitting. One of the techniques of subsampling is max pooling. With this technique, you select the highest pixel value from a region depending on its size. In other words, max pooling takes the largest value from the window of the image currently covered by the kernel. For example, you can have a max-pooling layer of size 2 x 2 will select the maximum pixel intensity value from 2 x 2 region. You're right to think that the pooling layer then works a lot like the convolution layer! You also take a kernel or a window and move it over the image; The only difference is the function that is applied to the kernel and the image window isn't linear.

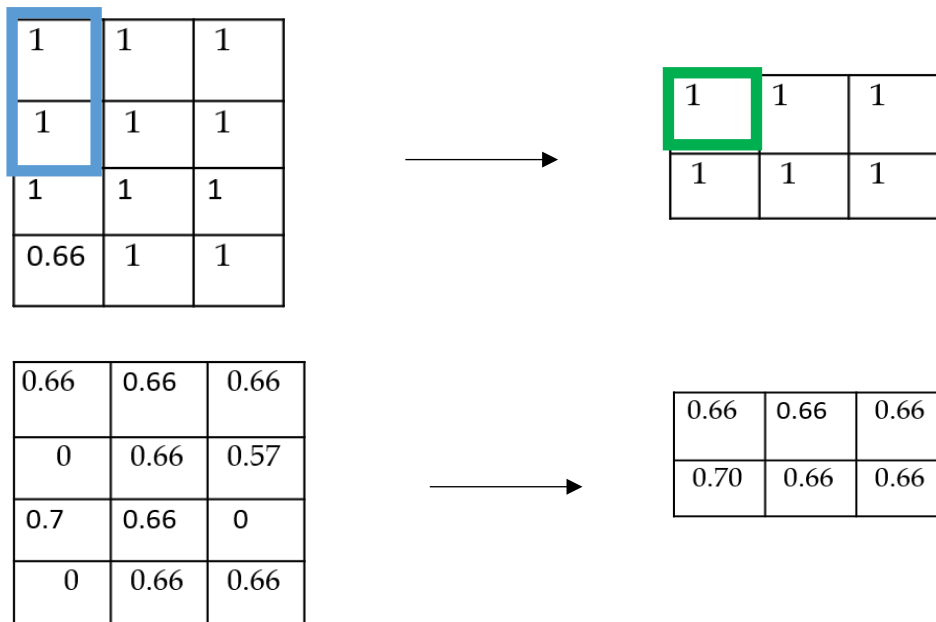


Fig4.13:output of pooling layer

Above cells are generated from the pooling layer and the size of the cells are reduced from 16 cells to 6 cells each. Like the dimension of the cell is reduced from 4X4 to 2X3.

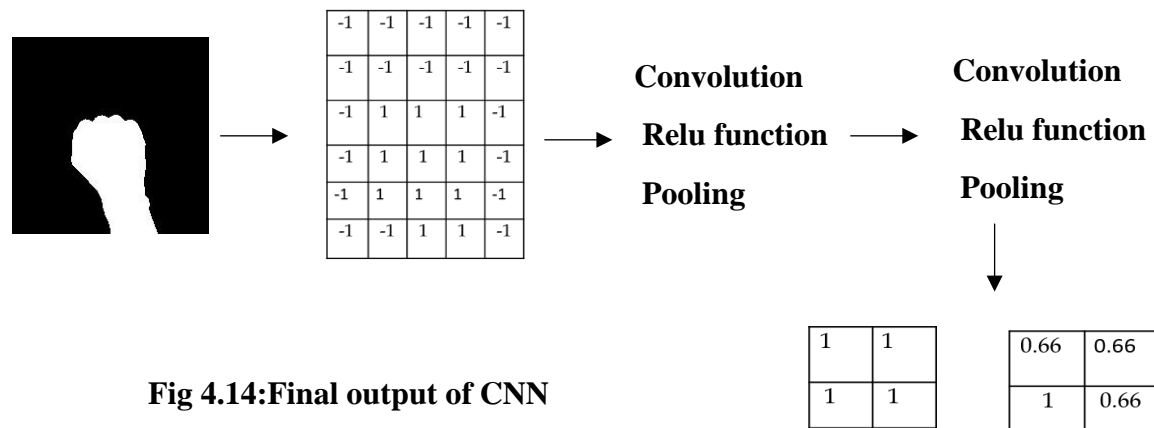
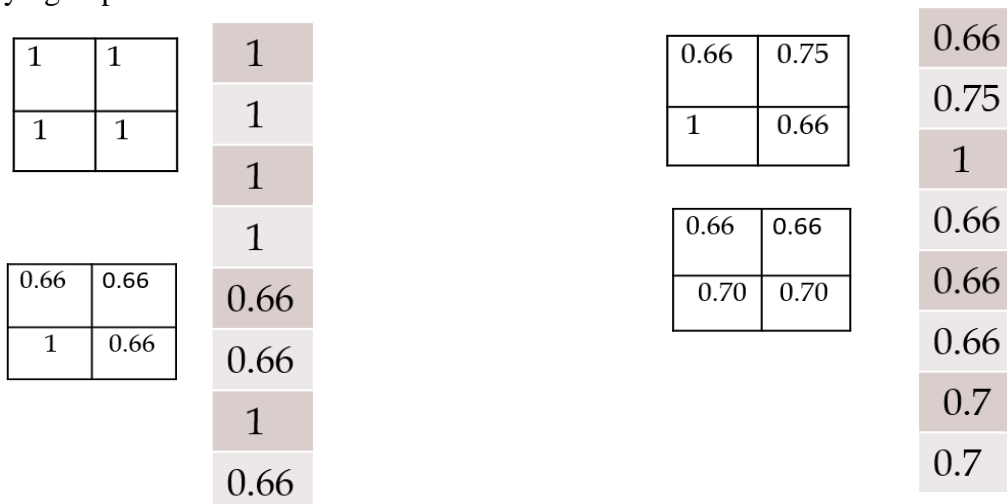


Fig 4.14: Final output of CNN

After performing multiple times of the convolution, relu and pooling layer, we reduce the spatial size of the original image. As you can see the original image is shrunk to a size from 6x5 to 2x2 (30 cells to 8 cells).

Fully connected Layer

At the end of a convolutional neural network are one or more fully connected layers (when two layers are "fully connected," every node in the first layer is connected to every node in the second layer). Their job is to perform classification based on the features extracted by the convolutions. Typically, the final fully connected layer contains a softmax activation function, which outputs a probability value from 0 to 1 for each of the classification labels the model is trying to predict.



Left side is the data obtained after training zero fingered data set and right side is five fingered data set.

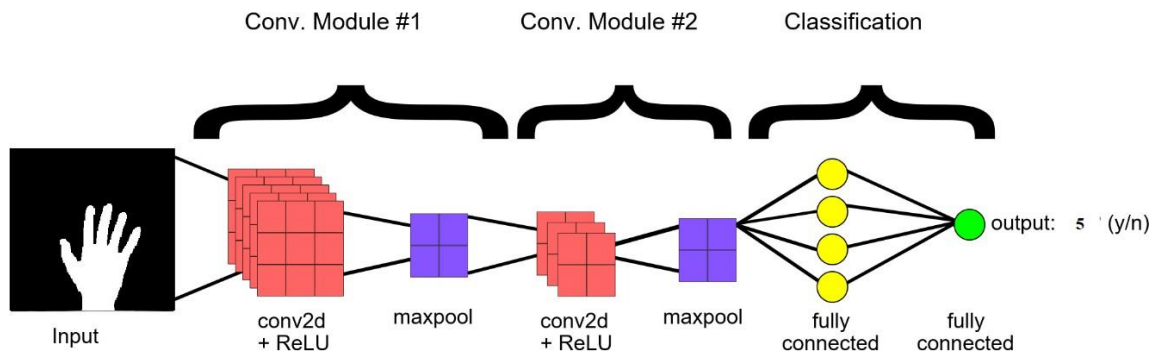


Fig4.15: Working of CNN model

The stack of cells obtained from the each image of a data set are averaged to create a model. And then the dynamic images model also generates a stack of values. By comparing these values of dynamic images data with the data set models we can predict the label of the dynamic image.



Fig4.16: Shows the dataset generated in system after processing the dynamic image.

The data generated after classifying the value of dynamic hand gesture using CNN it sends the data to the Arduino uno to control the device wired to it.



Fig4.17: Device is controlled using the output of the system.

If the output sent to the Arduino is “5” then the motor rotates, if the data sent is “0” then it halts.

Chapter 5

Implementation and Testing

This chapter describes the process of implementation of the CNN based system controller using hand gesture with the help of python scripts and hardware device.

Implementation

1. Data Set Retrieval

Accumulating the 4000 images to use as a data set in the project, which contains over set of 2000 images of five fingered and Fist fingered hand gesture. One among them is used to ON the device and other is used to Off the device.

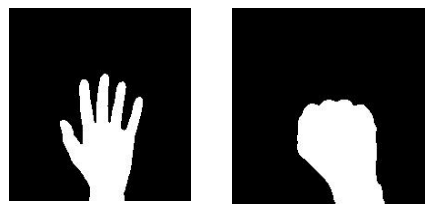


Fig:5.1 Dataset

Syntax for loading dataset and converting to grey scale images.

For Five fingered image:-

```
myFiveTrainImageFiles=glob.glob("C:/Users/saipranav/Desktop/dataset/fiveFingerTrainDataset/*.jpg")
```

Converting the dataset to a grey scale image by following line of code

```
myFiveTrainImages = [cv2.imread(img, 0) for img in myFiveTrainImageFiles]  
we pass zero to load greyscale image
```

For Zero fingered image:-

```
myZeroTrainImageFiles=glob.glob("C:/Users/saipranav/Desktop/dataset/zeroFingerTrainDataset/*.jpg")
```

```
myZeroTrainImageFiles.sort()
```

```
myZeroTrainImages = [cv2.imread(img, 0) for img in myZeroTrainImageFiles]
```

2.Training Data Set

Further the data set is divided into two parts, One is training data set and testing data set. Among them the trained data set is used for creating the CNN models.And the testing data set is used for estimating the accuracy of the model.

The trained data set are saved in the json model to create model weights for the cells,to assign labels to a specific image. These labels are used to predict the output of dynamic hand gesture.

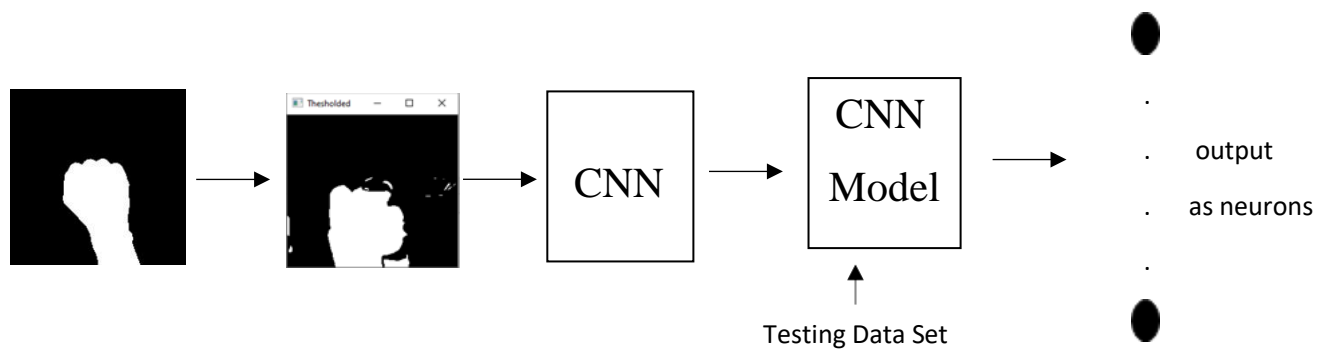


Fig5.2: Overview of training dataset

The above is the overview of the training model to generate a set of neurons for an image.

CNN model for all the set of images:-

```
model = baseline_model()
```

Fit the model

```
model.fit(x_train, y_train, validation_data=(x_test, y_test), epochs=10, batch_size=20, verbose=1)
```

Final evaluation of the model

```
scores = model.evaluate(x_test, y_test, verbose=0)
print("Baseline Error: %.2f%%" % (100 - scores[1] * 100))
```

Model stored as a json file:-

```
model_json = model.to_json();
with open("C:/Users/sai pranav/Desktop/trainedModel.json", "w") as jsonFile:
    jsonFile.write(model_json)
model.save_weights("C:/Users/satishkumar/Desktop/modelWeights.h5")
print("Saved model to disk")
```

3.Capturing the Dynamic Hand gesture

Here, the human hand gesture is captured and the model for that image is created to compare with the existing trained set of models. The hand gesture is captured with the help of the web cam.

Syntax for capturing the dynamic image

```
camera = cv2.VideoCapture(0)
```

The captured image is then transformed to grey scale image and then the model for the grey scale image is generated. With this model we can predict the type of hand gesture used.

4. Sending the Output of the CNN program to the Arduino

After recognizing the hand gesture form, we can ON or OFF the device to control the motor. The information produced from the Neural Network model is then sent to Arduino to control the device.



Fig 5.3 Arduino connection

We use pyserial package to transfer the data from the python script to Arduino.

Arduino has a program embedded to control the device. The above is the program stored in the processor of Arduino.

The following code is programmed in Arduino to control the device:-

```
void setup() {  
    // put your setup code here, to run once:  
    Serial.begin(9600);  
    pinMode(13,OUTPUT);  
}  
  
void loop() {  
    // put your main code here, to run repeatedly:  
    if(Serial.available())  
    {  
        switch(Serial.read())  
        { case '0':digitalWrite(13,LOW);  
          break;  
          case '5':digitalWrite(13,HIGH);  
            break;  
          default: break;  
        }  
    }  
}
```

5.Arduino connection with motor

The Arduino is connected to the motor through copper wires with the help of transistor, diode and resistor. These electronic objects are used to maintain the power supply.

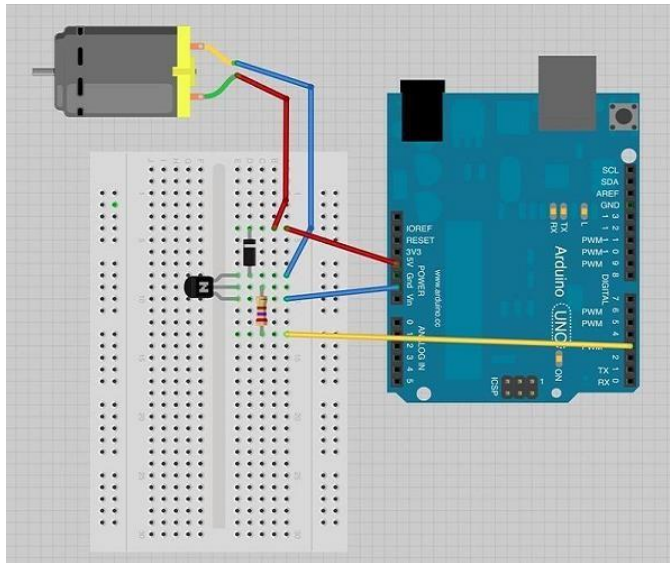


Fig5.4:Arduino to motor

You will need the following components –

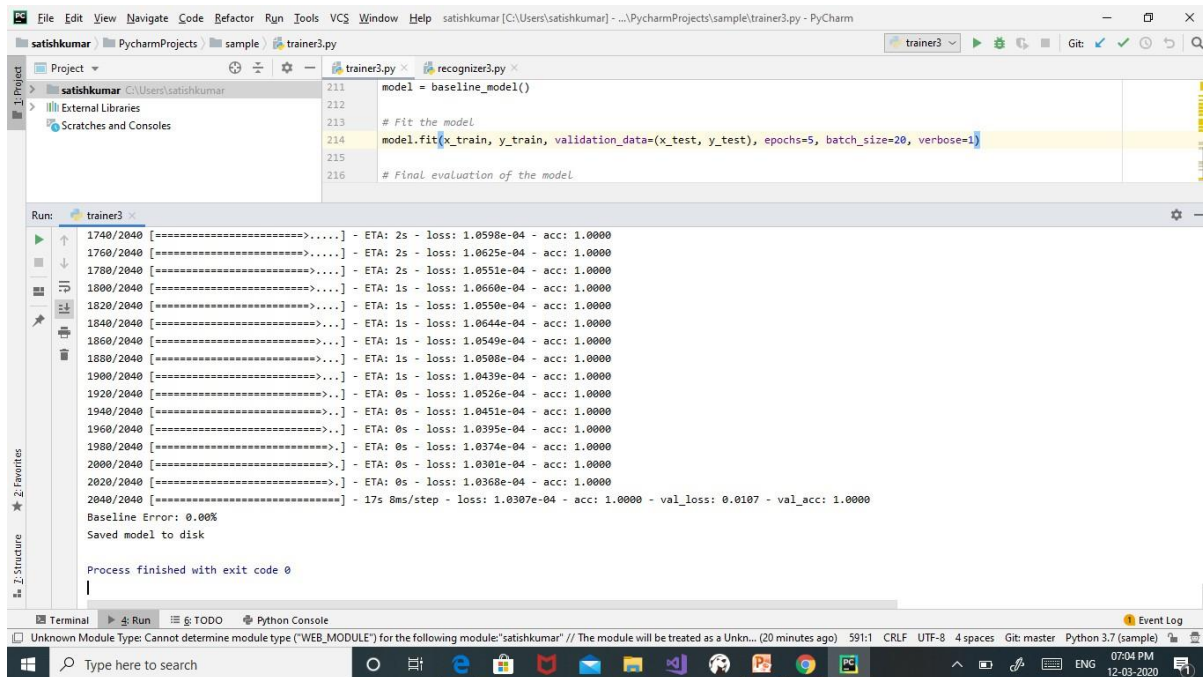
- 1x Arduino UNO board
- 1x PN2222 Transistor
- 1x Small 6V DC Motor
- 1x 1N4001 diode
- 1x 270 Ω Resistor

Connection Steps:

- Connect 5V and the ground of the IC to 5V and the ground of Arduino, respectively.
- Connect the motor to pins 2 and 3 of the IC.
- Connect IN1 of the IC to pin 8 of Arduino.
- Connect IN2 of the IC to pin 9 of Arduino.
- Connect EN1 of IC to pin 2 of Arduino.
- Connect SENS A pin of IC to the ground.
- Connect Arduino using Arduino USB cable and upload the program to Arduino using Arduino IDE software.
- Provide power to Arduino board using power supply, battery, or USB cable.

Testing:

After testing the data set we get the accuracy of our project using which we can conclude that how accurate is our model.



```
211 model = baseline_model()
212
213 # Fit the model
214 model.fit(x_train, y_train, validation_data=(x_test, y_test), epochs=5, batch_size=20, verbose=1)
215
216 # Final evaluation of the model
```

Run: trainer3

```
1740/2040 [=====>...] - ETA: 2s - loss: 1.0598e-04 - acc: 1.0000
1760/2040 [=====>...] - ETA: 2s - loss: 1.0625e-04 - acc: 1.0000
1780/2040 [=====>...] - ETA: 2s - loss: 1.0551e-04 - acc: 1.0000
1800/2040 [=====>...] - ETA: 1s - loss: 1.0660e-04 - acc: 1.0000
1820/2040 [=====>...] - ETA: 1s - loss: 1.0550e-04 - acc: 1.0000
1840/2040 [=====>...] - ETA: 1s - loss: 1.0644e-04 - acc: 1.0000
1860/2040 [=====>...] - ETA: 1s - loss: 1.0549e-04 - acc: 1.0000
1880/2040 [=====>...] - ETA: 1s - loss: 1.0508e-04 - acc: 1.0000
1900/2040 [=====>...] - ETA: 1s - loss: 1.0439e-04 - acc: 1.0000
1920/2040 [=====>...] - ETA: 0s - loss: 1.0526e-04 - acc: 1.0000
1940/2040 [=====>...] - ETA: 0s - loss: 1.0451e-04 - acc: 1.0000
1960/2040 [=====>...] - ETA: 0s - loss: 1.0395e-04 - acc: 1.0000
1980/2040 [=====>...] - ETA: 0s - loss: 1.0374e-04 - acc: 1.0000
2000/2040 [=====>...] - ETA: 0s - loss: 1.0301e-04 - acc: 1.0000
2020/2040 [=====>...] - ETA: 0s - loss: 1.0368e-04 - acc: 1.0000
2040/2040 [=====>...] - 17s 8ms/step - loss: 1.0307e-04 - acc: 1.0000 - val_loss: 0.0187 - val_acc: 1.0000
Baseline Error: 0.00%
Saved model to disk

Process finished with exit code 0
```

Fig5.5: Testing results

The above figure shows the testing dataset results of our CNN model which we have used in the project.

Chapter 6

Results and Discussion

This chapter presents the results of the CNN based system controller using hand gesture.

Results

When a five fingered hand gesture is shown:

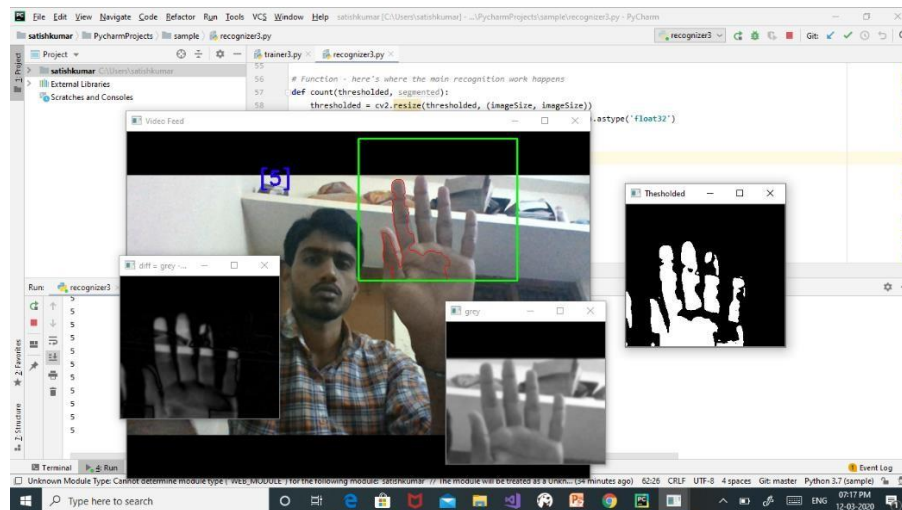


Fig6.1: Output of a five fingered hand gesture

The output of the model is “5” , which is sent to the Arduino using USB cable to control the device. The device is turned ON with this label. when an hand gesture of fist is given to model then the output is “0”.This label is used to OFF the device.

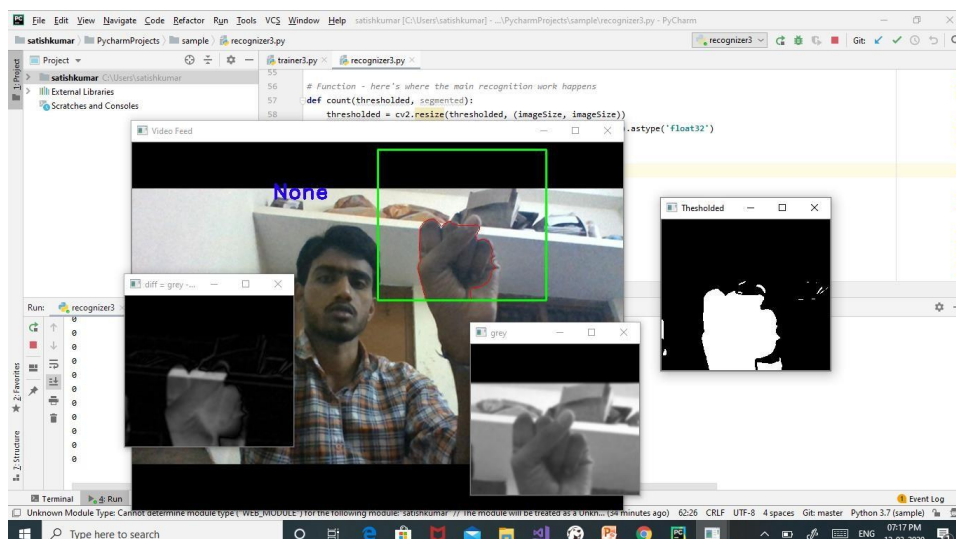


Fig6.2:Output of the fist figured hand gesture

Chapter 7

Conclusion

Automatic detection and classification of dynamic hand gestures in real-world systems intended for human computer interaction is challenging as:

- 1) there is a large diversity in how people perform gestures, making detection and classification difficult.
- 2) the system must work online in order to avoid noticeable lag between performing a gesture and its classification; in fact, a negative lag (classification before the gesture is finished) is desirable, as feedback to the user can then be truly instantaneous.

In this project, we address these challenges with a recurrent **convolutional neural network** that performs simultaneous detection and classification of dynamic hand gestures from multi-modal data. We employ connectionist temporal classification to train the network to predict class labels from in-progress gestures in unsegmented input streams. In order to validate our method, we introduce a new challenging multi-modal dynamic hand gesture dataset captured with depth, color. On this challenging dataset, our gesture recognition system achieves an accuracy of more than 95.8%, outperforms competing state-of-the-art algorithms, and approaches human accuracy of 99.4% . .

One of the problems in gesture recognition is dealing with the image background and the noise often present in the regions of interest, such as the hand region. The use of neural networks for color segmentation, followed by morphological operations and a polygonal approximation, presented excellent results as a way to separate the hand region from the background and to remove noise. This step is important because it removes image objects that are not relevant to the classification method, allowing the **convolutional neural network** to extract the most relevant gesture features through their convolution and pooling layers and, therefore, to increase network accuracy. The proposal to make a logical AND operation with the segmentation masks and the original images provided the relevant information of the palms and fingers. Thus, the proposed **CNN architectures** achieved high success rates at a relatively low computational cost. It was superior to methodologies mentioned in related works, confirming the robustness of the presented method. In addition, the proposed architectures reached accuracies very similar to the architectures already defined in the literature, although they are much simpler and have a lower computational cost. This is possible due to the proposed image processing methodology, in which unnecessary information is removed, allowing improved feature extraction by the **CNN**. The proposed methodology and **CNN** architecture open the door to a future implementation of gesture recognition in embedded devices with hardware limitations.

Chapter 8

Future Scope of Work

Hand gestures are more intuitive and hence have an advantage over other mediums of giving instructions (especially to computers). It can be used to enable speechless people to communicate verbally by directly converting the sign language to speech using gesture recognition.

The Future Scope of this project involves expanding to google cloud storage system, with any system we can send the inputs and control the devices. In this project, we are negotiating only two labels so that only one device can be handled with this application. To enforce more devices we can expand the classes to any number and control any number of devices with this application. So that it can have wide applications over Medical Field, Alternative computer interfaces, Entertainment Applications ,Automation systems and an easier life for the disabled. Using more sophisticated machine learning techniques we'll try to build a complete product as it is a Miniaturization of the whole system. And Running the device with least complexities.

The outcomes in this research are based on results that involve only sample datasets. It is necessary that additional datasets should be considered for the evaluation of different classification problems as the information growth in the recent technology is extending to heights beyond assumptions. Recent field of technology is growing and data are by nature dynamic. Hence, further classification of the entire system needs to be implemented right from the scratch since the results from the old process have become obsolete. The scope of future work can deal with Incremental learning, which stores the existing model and processes the new incoming data more efficiently. More specifically, the models with incremental learning can be used in categorization process to improve the following aspects in each type of problems.

When these enhancements are incorporated in the classification system, it would help further improve the performance and be useful for applications meant for the explicit classification system

Chapter 9

References

- [1] Ali A. Alani, Georgina Cosma, Aboozar Taherkhani, T.M Mc Ginnity “ Hand Gesture Recognition Using an Adapted Convolutional Neural Network with Data Augmentation” **Data of conference**-25 June 2018. **Published** -IEEE.
- [2] Deyvison de Paiva Penha and Adriana Rosa Garcez Castro. “HOME APPLIANCE IDENTIFICATION FOR NILM SYSTEMS BASED ON DEEP NEURAL NETWORKS” **Published in:** 2018, Computer Science, International Journal of Artificial Intelligence & Applications ;**Date Added to IJAIA:** March 2018 Volume no:2;**Publisher:** IJAIA
- [3] Felix Zhan “Hand Gesture Recognition with Convolution Neural Networks”. **Date of Conference:** 30 July-1 Aug. 2019;**Publisher:** IEEE
- [4] <https://www.datacamp.com/community/tutorials/convolutional-neural-networks-python>
- [5] https://www.tutorialspoint.com/python3/python_database_access.htm
- [6] <http://scikit-learn.org/stable/modules/tree.html#classification>
- [7] http://scikit-learn.org/stable/supervised_learning.html#supervised-learning
- [8] <https://www.edureka.co/blog/convolutional-neural-network/>
- [9] <https://developers.google.com/machine-learning/practica/image-classification>
- [10] <http://cs231n.stanford.edu/>
- [11] https://en.wikipedia.org/wiki/Visual_cortex
- [12] <https://developers.google.com/machine-learning/practica/image-classification>
- [13] https://www.youtube.com/playlist?list=PLZbbT5o_s2xq7LwI2y8_QtvuXZedL6tQU-deeplizard youtubechannel
- [14] J. J. LaViola Jr. An introduction to 3D gestural interfaces. In SIGGRAPH Course, 2014.
- [15] V. I. Pavlovic, R. Sharma, and T. S. Huang. Visual interpretation of hand gestures for human-computer interaction: A review. PAMI, 19:677–695, 1997.
- [16] S. B. Wang, A. Quattoni, L. Morency, D. Demirdjian, and T. Darrell. Hidden conditional random fields for gesture recognition. In CVPR, pages 1521–1527, 2006.
- [17] P. Trindade, J. Lobo, and J. Barreto. Hand gesture recognition using color and depth images enhanced with hand angular pose data. In IEEE Conf. on Multisensor Fusion and Integration for Intelligent Systems, pages 71–76, 2012.
- [18] M. Zobl, R. Nieschulz, M. Geiger, M. Lang, and G. Rigoll. Gesture components for natural interaction with in-car devices. In Gesture-Based Communication in Human Computer Interaction, pages 448–459. Springer, 2004.

- [19] F. Althoff, R. Lindl, and L. Walchshausl. Robust multimodal hand-and head gesture recognition for controlling automotive infotainment systems. In VDI-Tagung: Der Fahrer im 21. Jahrhundert, 2005.
- [20] F. Parada-Loira, E. Gonzalez-Agulla, and J. Alba-Castro. Hand gestures to control infotainment equipment in cars. In IEEE Intelligent Vehicles Symposium, pages 1–6, 2014.
- [21] D. C. Cireşan, U. Meier, J. Masci, L. M. Gambardella, and J. Schmidhuber. Flexible, high performance convolutional neural networks for image classification. In International Joint Conference on Artificial Intelligence, pages 1237–1242, 2011.
- [22] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In NIPS, pages 1097–1105. 2012.
- [23] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner. Gradientbased learning applied to document recognition. In Proceedings of the IEEE, pages 2278–2324, 1998.
- [24] P. Y. Simard, D. Steinkraus, and J. C. Platt. J.c.: Best practices for convolutional neural networks applied to visual document analysis. In Int. Conference on Document Analysis and Recognition, pages 958– 963, 2003.
- [25] D. Cireşan, U. Meier, and J. Schmidhuber. Multi-column deep neural networks for image classification. In CVPR, pages 3642–3649, 2012.
- [26] A. Karpathy, G. Toderici, S. Shetty, T. Leung, R. Sukthankar, and L. Fei-Fei. Large-scale video classification with convolutional neural networks. In CVPR, pages 1725–1732, 2014.
- [27] L. Pigou, S. Dieleman, P.-J. Kindermans, and B. Schrauwen. Sign language recognition using convolutional neural networks. In ECCVW, 2014.
- [28] K. Simonyan and A. Zisserman. Two-stream convolutional networks for action recognition in videos. In NIPS, pages 568–576, 2014.
- [29] P. Molchanov, S. Gupta, K. Kim, and K. Pulli. Multi-sensor System for Driver’s Hand-gesture Recognition. In AFGR, 2015.
- [30] S. Ioffe and C. Szegedy, “Batch normalization: Accelerating deep network training by reducing internal covariate shift,” in International Conference on Machine Learning, 2015, pp. 448–456.
- [31] Raymond Ahn, Justin Zhan, Using proxies for node immunization identification on large graphs, IEEE Access, Vol. 5, pp. 13046-13053, 2017.
- [32] Gary Blosser, Justin Zhan, Privacy preserving collaborative social network, International Conference on Information Security and Assurance, pp. 543-548, 2008.
- [33] Brittany Cozzens, Richard Huang, Maxwell Jay, Kyle Khembunjong, Sahan Paliskara, Felix Zhan, Mark Zhang, Shahab Tayeb, Signature Verification Using a Convolutional Neural Network, University of Nevada Las Vegas AEOP/STEM/REAP/RET Programs Technical Report, 2018.

Appendices

A.1 List of Figures

S.no	Figure number	Name of Figure
1	1.1	Example of cats image classification
2	1.2	Basic Hand gestures
3	1.3	Recognizing the gesture with label
4	1.4	Brief idea of our project
5	3.1	Vector direction
6	3.2	Numpy storage representaion
7	3.3	Ndarray
8	3.4	Basic arduino code
9	3.5	Arduino ATmega2560
10	3.6	USB cable
11	3.7	Electric motor
12	4.1	Block Diagram
13	4.2	3-dimensional space of pixel values
14	4.3	The range of intensity values from 0 (black) to 255 (white).
15	4.4	Thumbnail of a digital grayscale image.
16	4.5	The lower left 15 x 15 pixel portion of the image
17	4.6	Pixel intensity values of the lower left 15 x 15 pixel portion of the image.
18	4.7	Data Set Models for each image
19	4.8	Image intensity of gray scale image
20	4.9	Labeling data sets
21	4.10	Filtered images of a dataset
22	4.11	Cell Image of a data set
23	4.12	output of convolution layer
24	4.13	output of pooling layer
25	4.14	Final output of CNN
26	4.15	Working of CNN model
27	4.16	Dataset generated after processing the dynamic image
28	4.17	Device is controlled using the output of the system
29	5.1	Data set
30	5.2	Overview of training data set
31	5.3	Arduino connection
32	5.4	Arduino to motor
33	5.5	Testing results
34	6.1	Output of a five fingered hand gesture
35	6.2	Output of the fist figured hand gesture

A.2 Listing of Code

Trainer code:

```
import numpy
from keras.models import Sequential
from keras.models import model_from_json
from keras.layers import Dense
from keras.layers import Dropout
from keras.layers import Flatten
from keras.layers.convolutional import Conv2D
from keras.layers.convolutional import MaxPooling2D
from keras.utils import np_utils
import glob
import cv2
from sklearn.utils import shuffle
import os

# fix random seed for reproducibility
seed = 7
numpy.random.seed(seed)
imageSize = 50

# load training data for gesture 2
myFiveTrainImageFiles = glob.glob("C:/Users/sai
pranav/Desktop/dataset/fiveFingerTrainDataset/*.jpg")
myFiveTrainImageFiles.sort()
myFiveTrainImages = [cv2.imread(img, 0) for img in myFiveTrainImageFiles] # we pass
zero to load greyscale image

for i in range(0, len(myFiveTrainImages)):
    myFiveTrainImages[i] = cv2.resize(myFiveTrainImages[i], (imageSize, imageSize))
tn1 = numpy.asarray(myFiveTrainImages)

# load training data for gesture 1
myZeroTrainImageFiles = glob.glob("C:/Users/sai
pranav/Desktop/dataset/zeroFingerTrainDataset/*.jpg")
myZeroTrainImageFiles.sort()
myZeroTrainImages = [cv2.imread(img, 0) for img in myZeroTrainImageFiles]

for i in range(0, len(myZeroTrainImages)):
    myZeroTrainImages[i] = cv2.resize(myZeroTrainImages[i], (imageSize, imageSize))
tn2 = numpy.asarray(myZeroTrainImages)

# load training data for gesture 3
myGes3TrainImageFiles = glob.glob("C:/Users/sai
pranav/Desktop/dataset/gesture3Train/*.jpg")
myGes3TrainImageFiles.sort()
myGes3TrainImages = [cv2.imread(img, 0) for img in myGes3TrainImageFiles]

for i in range(0, len(myGes3TrainImages)):
```

```

    myGes3TrainImages[i] = cv2.resize(myGes3TrainImages[i], (imageSize, imageSize))
tn3 = numpy.asarray(myGes3TrainImages)

# load testing data for gesture 4 (RANDOM)
myGes4TrainImageFiles = glob.glob("C:/Users/sai
pranav/Desktop/dataset/gesture4Train/*.jpg")
myGes4TrainImageFiles.sort()
myGes4TrainImages = [cv2.imread(img, 0) for img in myGes4TrainImageFiles]

for i in range(0, len(myGes4TrainImages)):
    myGes4TrainImages[i] = cv2.resize(myGes4TrainImages[i], (imageSize, imageSize))
tn4 = numpy.asarray(myGes4TrainImages)

finalTrainImages = []
finalTrainImages.extend(myFiveTrainImages)
finalTrainImages.extend(myZeroTrainImages)
finalTrainImages.extend(myGes3TrainImages)
finalTrainImages.extend(myGes4TrainImages)

# load testing data for gesture 2 (STOP)
myFiveTestImageFiles = glob.glob("C:/Users/sai
pranav/Desktop/dataset/fiveFingerTestDataset/*.jpg")
myFiveTestImageFiles.sort()
myFiveTestImages = [cv2.imread(img, 0) for img in myFiveTestImageFiles]

for i in range(0, len(myFiveTestImages)):
    myFiveTestImages[i] = cv2.resize(myFiveTestImages[i], (imageSize, imageSize))
ts1 = numpy.asarray(myFiveTestImages)

# load testing data for gesture 1 (FIST)
myZeroTestImageFiles = glob.glob("C:/Users/sai
pranav/Desktop/dataset/zeroFingerTestDataset/*.jpg")
myZeroTestImageFiles.sort()
myZeroTestImages = [cv2.imread(img, 0) for img in myZeroTestImageFiles]

for i in range(0, len(myZeroTestImages)):
    myZeroTestImages[i] = cv2.resize(myZeroTestImages[i], (imageSize, imageSize))
ts2 = numpy.asarray(myZeroTestImages)

# load testing data for gesture 3 (OK)
myGes3TestImageFiles = glob.glob("C:/Users/sai
pranav/Desktop/dataset/gesture3Test/*.jpg")
myGes3TestImageFiles.sort()
myGes3TestImages = [cv2.imread(img, 0) for img in myGes3TestImageFiles]

for i in range(0, len(myGes3TestImages)):
    myGes3TestImages[i] = cv2.resize(myGes3TestImages[i], (imageSize, imageSize))
ts3 = numpy.asarray(myGes3TestImages)

```

```

# load testing data for gesture 4 (RANDOM)
myGes4TestImageFiles = glob.glob("C:/Users/sai
pranav/Desktop/dataset/gesture4Test/*.jpg")
myGes4TestImageFiles.sort()
myGes4TestImages = [cv2.imread(img, 0) for img in myGes4TestImageFiles]

for i in range(0, len(myGes4TestImages)):
    myGes4TestImages[i] = cv2.resize(myGes4TestImages[i], (imageSize, imageSize))
ts4 = numpy.asarray(myGes4TestImages)

finalTestImages = []
finalTestImages.extend(myFiveTestImages)
finalTestImages.extend(myZeroTestImages)
finalTestImages.extend(myGes3TestImages)
finalTestImages.extend(myGes4TestImages)

x_train = numpy.asarray(finalTrainImages)
x_test = numpy.asarray(finalTestImages)

# Now preparing the training and testing outputs

y_myFiveTrainImages = numpy.empty([tn1.shape[0]])
y_myZeroTrainImages = numpy.empty([tn2.shape[0]])
y_myGes3TrainImages = numpy.empty([tn3.shape[0]])
y_myGes4TrainImages = numpy.empty([tn4.shape[0]])

y_myFiveTestImages = numpy.empty([ts1.shape[0]])
y_myZeroTestImages = numpy.empty([ts2.shape[0]])
y_myGes3TestImages = numpy.empty([ts3.shape[0]])
y_myGes4TestImages = numpy.empty([ts4.shape[0]])

for j in range(0, tn1.shape[0]):
    y_myFiveTrainImages[j] = 5

for j in range(0, tn2.shape[0]):
    y_myZeroTrainImages[j] = 0

for j in range(0, tn3.shape[0]):
    y_myGes3TrainImages[j] = 1

for j in range(0, tn4.shape[0]):
    y_myGes4TrainImages[j] = 2

for j in range(0, ts1.shape[0]):
    y_myFiveTestImages[j] = 5

for j in range(0, ts2.shape[0]):
    y_myZeroTestImages[j] = 0

for j in range(0, ts3.shape[0]):

```

```

y_myGes3TestImages[j] = 1

for j in range(0, ts4.shape[0]):
    y_myGes4TestImages[j] = 2

y_train_temp = []
y_train_temp.extend(y_myFiveTrainImages)
y_train_temp.extend(y_myZeroTrainImages)
y_train_temp.extend(y_myGes3TrainImages)
y_train_temp.extend(y_myGes4TrainImages)
y_train = numpy.asarray(y_train_temp)

y_test_temp = []
y_test_temp.extend(y_myFiveTestImages)
y_test_temp.extend(y_myZeroTestImages)
y_test_temp.extend(y_myGes3TestImages)
y_test_temp.extend(y_myGes4TestImages)
y_test = numpy.asarray(y_test_temp)

print(x_train.shape)
# print(x_test.shape)

print(y_train.shape)
# print(y_test.shape)
print()
print(x_test.shape)
print(y_test.shape)
# shuffling the data
x_train, y_train = shuffle(x_train, y_train)
x_test, y_test = shuffle(x_test, y_test)

# flatten 50*50 images to a 2500 vector for each image
num_pixels = x_train.shape[1] * x_train.shape[2]
x_train = x_train.reshape(x_train.shape[0], 1, imageSize, imageSize).astype('float32')
x_test = x_test.reshape(x_test.shape[0], 1, imageSize, imageSize).astype('float32')

# normalize inputs from 0-255 to 0-1
x_train = x_train / 255
x_test = x_test / 255

# one hot encode outputs
y_train = np_utils.to_categorical(y_train)
y_test = np_utils.to_categorical(y_test)

num_classes = y_test.shape[1]
print("num_classes")
print(num_classes)

# define baseline model

```

```

def baseline_model():
    # create model
    model = Sequential()
    model.add(
        Conv2D(32, (5, 5), input_shape=(1, imageSize, imageSize),
data_format='channels_first', activation='relu'))
    model.add(MaxPooling2D(pool_size=(2, 2)))
    model.add(Dropout(0.2))
    model.add(Flatten())
    model.add(Dense(128, activation='relu'))
    model.add(Dense(num_classes, activation='softmax'))
    # Compile model
    model.compile(loss='categorical_crossentropy', optimizer='adam',
metrics=['accuracy'])
    return model

# build the model
model = baseline_model()

# Fit the model
model.fit(x_train, y_train, validation_data=(x_test, y_test), epochs=10, batch_size=20,
verbose=1)

# Final evaluation of the model
scores = model.evaluate(x_test, y_test, verbose=0)
print("Baseline Error: %.2f%%" % (100 - scores[1] * 100))

# Save the model
model_json = model.to_json();
with open("C:/Users/sai pranav/Desktop/trainedModel.json", "w") as jsonFile:
    jsonFile.write(model_json)
model.save_weights("C:/Users/satishkumar/Desktop/modelWeights.h5")
print("Saved model to disk")

```

Recognizer code:

```

import cv2
import imutils
import numpy as np
from sklearn.metrics import pairwise
import time
from keras.datasets import mnist
from keras.models import Sequential
from keras.models import model_from_json
from keras.layers import Dense
from keras.layers import Dropout
from keras.utils import np_utils
import tkinter
import winsound

```



```

import tkinter as tk
import glob

bg = None
global loaded_model
imageSize = 50

# Function - To find the running average over the background

# def executeThis():
#     print("Gesture 0")

def run_avg(image, accumWeight):
    global bg
    if bg is None:
        bg = image.copy().astype('float')
    return

    cv2.accumulateWeighted(image, bg, accumWeight)

# Function - To segment the region of hand in the image
def segment(image, threshold=30):
    global bg
    # find the absolute difference between background and current frame
    diff = cv2.absdiff(bg.astype('uint8'), image)
    cv2.imshow("diff = grey - bg", diff)
    cv2.imshow("grey", image)
    # threshold the diff image so that we get the foreground
    thresholded = cv2.threshold(diff, threshold, 255, cv2.THRESH_BINARY)[1]
    (cnts, _) = cv2.findContours(thresholded.copy(), cv2.RETR_EXTERNAL,
cv2.CHAIN_APPROX_SIMPLE)
    if len(cnts) == 0:
        return
    else:
        segmented = max(cnts, key=cv2.contourArea)
        return (thresholded, segmented)

# Function - here's where the main recognition work happens
def count(thresholded, segmented):
    thresholded = cv2.resize(thresholded, (imageSize, imageSize))
    thresholded = thresholded.reshape(1, 1, imageSize, imageSize).astype('float32')
    thresholded = thresholded / 255
    prob = loaded_model.predict(thresholded)
    if (max(prob[0]) > .99995):
        return loaded_model.predict_classes(thresholded)
    return

```

```

# Main function
if __name__ == "__main__":

    # load the structure of the model
    json_file = open('C:/Users/satishkumar/Desktop/trainedModel.json', 'r')
    loaded_model_json = json_file.read()
    json_file.close()
    loaded_model = model_from_json(loaded_model_json)

    # load weights into new model
    loaded_model.load_weights("C:/Users/satishkumar/Desktop/modelWeights.h5")
    print("\n\n\nLoaded model from disk\n\n\n")
    loaded_model.compile(loss='categorical_crossentropy', optimizer='adam',
metrics=['accuracy'])

    accumWeight = 0.5

    # get the reference to the webcam
    camera = cv2.VideoCapture(0)

    # region of interest (ROI) coordinates
    top, right, bottom, left = 10, 350, 225, 590

    # initialize num of frames
    num_frames = 0

    # calibration indicator
    calibrated = False
    #
    # window = tkinter.Tk()
    # keep looping, until interrupted
    while (True):
        # get the current frame
        (grabbed, frame) = camera.read()

        # resize the frame
        frame = imutils.resize(frame, width=700)

        # flip the frame so that it is not the mirror view
        frame = cv2.flip(frame, 1)

        # clone the frame
        clone = frame.copy()

        # get the height and width of the frame
        (height, width) = frame.shape[:2]

        # get the ROI

```

```

roi = frame[top:bottom, right:left]

# convert the roi to grayscale and blur it
gray = cv2.cvtColor(roi, cv2.COLOR_BGR2GRAY)
gray = cv2.GaussianBlur(gray, (7, 7), 0)

# to get the background, keep looking till a threshold is reached
# so that our weighted average model gets calibrated
if num_frames < 30:
    run_avg(gray, accumWeight)
    if num_frames == 1:
        print(">>> Please wait! Program is calibrating the background...")
    elif num_frames == 29:
        print(">>> Calibration successfull.... ")
else:
    # segment the hand region
    hand = segment(gray)
    time.sleep(.2)

    # check whether hand region is segmented
    if hand is not None:

        (thresholded, segmented) = hand
        # draw the segmented region and display the frame
        cv2.drawContours(clone, [segmented + (right, top)], -1, (0, 0, 255))

        # count the number of fingers
        fingers = count(thresholded, segmented)

        # print(fingers)
        if (fingers == 5):
            cv2.putText(clone, "[5]", (200, 80), cv2.FONT_HERSHEY_DUPLEX, 1, (255,
0, 35), 2)
        else:
            # print(fingers)
            cv2.putText(clone, str(fingers), (200, 80), cv2.FONT_HERSHEY_DUPLEX, 1,
(255, 0, 35), 2)
            # show the thresholded image

            cv2.imshow("Thesholded", thresholded)

        # draw the segmented hand
        cv2.rectangle(clone, (left, top), (right, bottom), (0, 255, 0), 2)

        # increment the number of frames
        num_frames += 1

        # display the frame with segmented hand
        cv2.imshow("Video Feed", clone)

```

```

# observe the keypress by the user
keypress = cv2.waitKey(1) & 0xFF

# if the user has pressed "q", then stop looping
if keypress == ord("q"):
    break

# free up memory
camera.release()
cv2.destroyAllWindows()

```

Arduino code:

```

void setup() {
    // put your setup code here, to run once:
    Serial.begin(9600);
    pinMode(13,OUTPUT);
}

void loop() {
    // put your main code here, to run repeatedly:
    if(Serial.available())
    {
        switch(Serial.read())
        { case '0':digitalWrite(13,LOW);
          break;
          case '5':digitalWrite(13,HIGH);
          break;
          default: break;
        }
    }
}

```