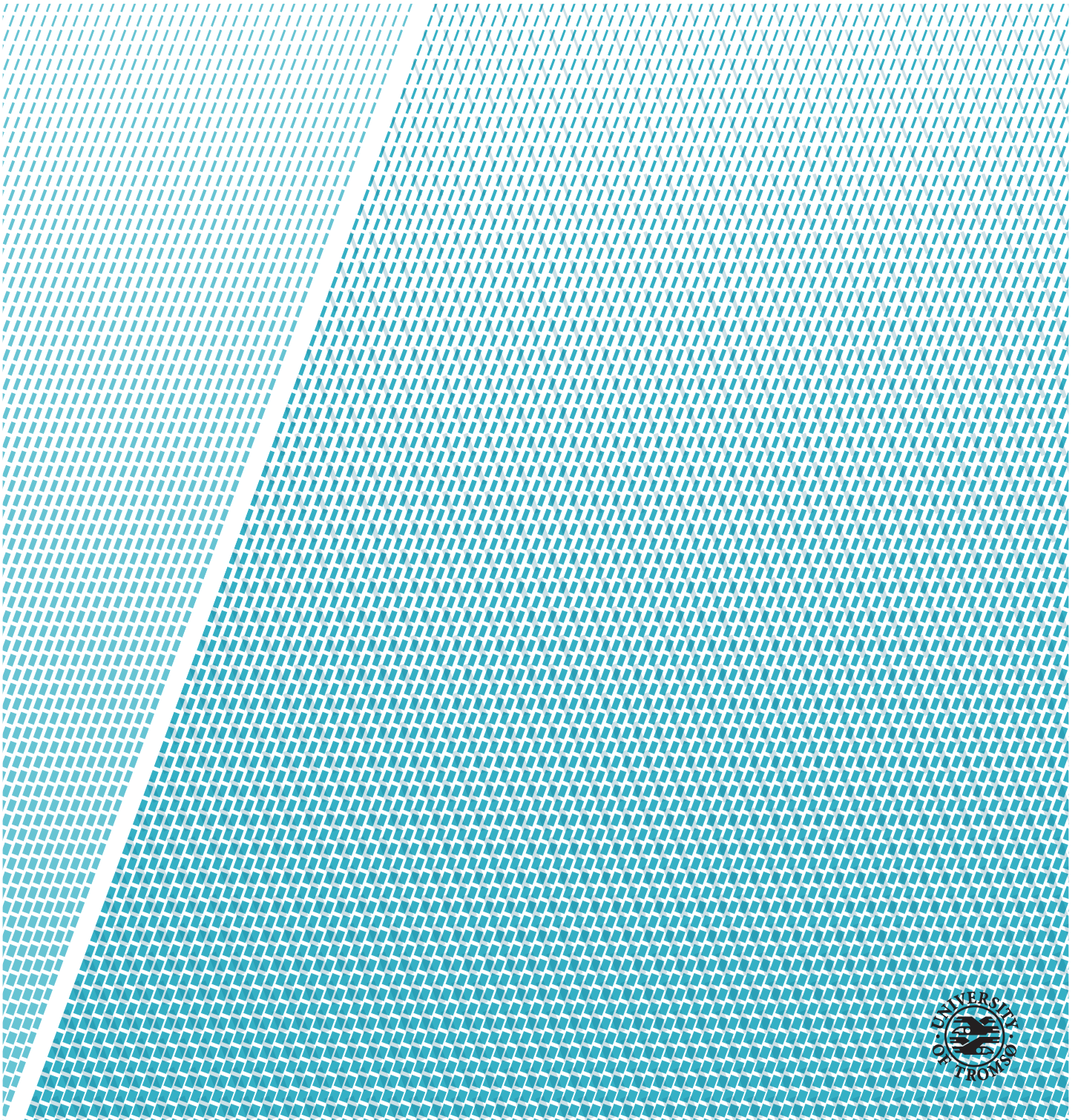


# Humanoid Robot handling Hand-Signs Recognition

---

**Mayuresh Amberkar**

*Master thesis in Computer Science ... August 2020*





# Abstract

Recent advancements in human-robot interaction have led to tremendous improvement for humanoid robots but still lacks social acceptance among people. Though verbal communication is the primary means of human-robot interaction, non-verbal communication that is proven to be an integral part of the human interactions is not widely used in humanoid robots. This thesis aims to achieve human-robot interaction via non-verbal communication, especially using hand-signs. It presents a prototype system that simulates hand-signs recognition in the NAO humanoid robot, and further an online questionnaire is used to examine people's opinion on the use of non-verbal communication to interact with a humanoid robot. The positive results derived from the study indicates people's willingness to use non-verbal communication as a means to communicate with humanoid robots, thus encouraging robot designers to use non-verbal communications for enhancing human-robot interaction.



# Acknowledgements

Although master's dissertations are said to be the work of an individual, I received enormous support and assistance from many people throughout the interim. Since the time I had been exploring various research topics until the very end, my supervisor, Professor Anne Håkansson, has provided invaluable inputs in every aspect of this thesis. I will be forever thankful for her constant guidance, expertise, investment in required resources, and availability in tough times to deliver this dissertation successfully. I would also thank my co-supervisor, Professor Randi Karlsen, especially to take time to evaluate my writing and provide different perspectives during my research.

I am heartfully thankful for the technical and administrative support provided by the Informatics Department at the University of Tromsø - particularly to Mr.Kai-Even Nilssen (Chief Engineer), Mr.Ken-Arne Jensen (Senior Engineer) and Mr.Jan Fuglesteg (Student Advisor) for their immediate assistance to any issues.

Besides, I would like to thank my colleague, Mr.Yigit Can Dundar, for sharing his inputs over my research, offering help when solving technical issues, and a healthy companionship at our workplaces. Finally, I would like to thank my friends and family to support me in this journey, especially my brother and sister-in-law, who motivated me to pursue a master's degree in the first place, and grateful to the God's grace in helping me to get through this entire journey!



# Contents

<b>Abstract</b>	<b>i</b>
<b>Acknowledgements</b>	<b>iii</b>
<b>List of Figures</b>	<b>vii</b>
<b>List of Tables</b>	<b>ix</b>
<b>1 Introduction</b>	<b>1</b>
1.1 Background and Motivation . . . . .	2
1.2 Problem . . . . .	3
1.3 Purpose . . . . .	4
1.4 Goals . . . . .	4
1.5 Methodology . . . . .	4
1.6 Contribution . . . . .	5
1.7 Delimitation . . . . .	6
1.8 Outline . . . . .	6
<b>2 Humanoid Robots and Importance of Hand-Signs</b>	<b>9</b>
2.1 Humanoid Robot . . . . .	9
2.1.1 Introduction to NAO humanoid robot . . . . .	10
2.1.2 Key components and Features of the NAO robot . . . . .	10
2.1.3 The NAOqi Framework and Choregraphe . . . . .	11
2.2 Importance of Hand-Signs in Non-Verbal Communication . . . . .	13
<b>3 Deep Learning Neural Networks</b>	<b>15</b>
3.1 Deep Learning . . . . .	15
3.2 Artificial Neural Network (ANN) . . . . .	16
3.3 Convolutional Neural Network (CNN) . . . . .	20
3.4 Training Deep Learning Neural Networks . . . . .	25
3.5 Related Work . . . . .	31
<b>4 Methodology</b>	<b>35</b>
4.1 Development Research . . . . .	35

4.2	Software Engineering . . . . .	37
4.3	Prototyping . . . . .	38
4.4	Research Evaluation . . . . .	39
<b>5</b>	<b>Prototype development of the Humanoid robot handling Hand-Signs</b>	<b>43</b>
5.1	System Requirements . . . . .	43
5.2	Design process of the prototype system . . . . .	45
5.3	Development of the NAO humanoid robot . . . . .	47
5.4	Development of the Integration Layer . . . . .	49
5.5	Development of Hand Signs Recognition Component (HSRC)	49
5.6	Establishing the Human-Robot Interaction . . . . .	51
<b>6</b>	<b>Implementation</b>	<b>55</b>
6.1	The NAO robot . . . . .	55
6.1.1	Programming NAO using Choregraphe . . . . .	56
6.2	Integration Layer . . . . .	58
6.3	Hand Signs Recognition Component (HSRC) . . . . .	59
6.3.1	Data Collection/Exploration . . . . .	60
6.3.2	Designing the model . . . . .	61
6.3.3	Evaluating and Refining the model . . . . .	63
<b>7</b>	<b>Evaluation and Results</b>	<b>65</b>
7.1	Accuracy of the prototype . . . . .	65
7.2	Responsiveness and User Acceptance of the prototype . . . . .	69
7.2.1	Summary of the results . . . . .	71
<b>8</b>	<b>Discussion</b>	<b>73</b>
<b>9</b>	<b>Conclusion and Future Scope</b>	<b>77</b>



# List of Figures

2.1	Sensors and Actuators of NAO robot, Source: Taken from [25]	11
2.2	Overview of the Choregraphe tool . . . . .	12
3.1	Structure of Biological Neuron, Source: Taken from [44] . .	17
3.2	Structure of Artificial Neuron, Source: Adapted from [45] . .	17
3.3	Softmax function for binary image-classification . . . . .	20
3.4	Matrix representations of a digital image, Source: Adapted from [51] . . . . .	21
3.5	Convolution in CNN, Source: Taken from [52] . . . . .	22
3.6	Convolution in CNN (Stride=2), Source: Taken from [53] . .	22
3.7	Examples of Feature Maps, Source: Taken from [54] . . . . .	23
3.8	Max pooling in CNN, Source: Taken from [56] . . . . .	24
3.9	Steps in CNN, Source: Adapted from [51] . . . . .	25
3.10	Underfitting vs Good fitting models, Source:[62] . . . . .	28
3.11	Overfitting vs Good fitting models, Source: Taken from [62]	29
3.12	Model Complexity Graph, Source: Taken from [67] . . . . .	31
4.1	Design Science Research (DSR) process model, Source: Taken from [75] . . . . .	36
4.2	Steps in Prototype Development . . . . .	39
5.1	Hand-Signs recognized by the prototype . . . . .	45
5.2	System design of the prototype . . . . .	48
5.3	Sequence diagram of establishing the human-robot interaction in prototype . . . . .	52
5.4	User showing the hand-sign to the NAO robot . . . . .	53
5.5	NAO robot performing a task on detecting the hand-sign . .	53
6.1	Choregraphe Behavior: 'Hand Signs Recognition' . . . . .	57
6.2	Parts of training a CNN model . . . . .	60
7.1	Accuracy of intermediate builds of CNN 1 . . . . .	66
7.2	Accuracy of intermediate builds of CNN 2 . . . . .	66
7.3	Accuracy and Loss of the best builds of CNN 1 . . . . .	68

7.4	Accuracy of intermediate builds of CNN 2 . . . . .	68
7.5	Responsiveness of the robot analyzed from questionnaire . .	70
7.6	Possible application areas for using non-verbal communication	71

# List of Tables

2.1	Main components of Choregraphe [33]	12
2.2	Common hand-signs and their meanings	14
6.1	Training Parameters and Accuracy of Baseline Model	62
7.1	Training Parameters and Accuracy of CNN 1	67
7.2	Training Parameters and Accuracy of CNN 2	67





# Introduction

A Humanoid Robot is a robot that resembles human's physical attributes like a head, upper torso, and legs. Moreover, it interacts with humans, other robots and environment, interprets the information and performs some actions using its sensors and actuators [1]. These robots were typically pre-programmed to perform specific tasks.

Recent advancements in humanoid robots have widened the application areas of humanoid robots typically to healthcare, education, research, and social care [1]. Healthcare practitioners appreciate the presence and help from advanced surgical robots. Humanoid robots acted as therapists have shown positive results to people suffering from depression, anxiety and anger [1]. In education and research, humanoid robots majorly serve as teaching assistants to teach various subjects (language, mathematics, nutrition) resulting in positive effects in learning, curiosity, creativity, knowledge and recall rate [1]. Humanoid robots can evoke a feeling of care and enhance social awareness [1]. Social robots or Socially Assistive Robots (SAR) have proven to be very useful among elderly and hospitality industries[2].

Humanoid robots need to communicate naturally to succeed in various such fields. Natural communication is multi-modal, with both verbal (text, speech) and non-verbal channels (signs, gestures, and other behaviours) [2]. Humans primarily interact using verbal communication. Therefore, verbal communication has been the first and principal form of communication robot designers used for delivering efficient interactions among humans and robots. But non-

verbal communication is often neglected that happens to be an integral part of human interactions since a long time. It augments and reinforces the verbal communication [3]. Thus, non-verbal communication has the potential to enhance the ability of robots to interact with humans and this thesis focuses on exploring this possibility to improve the human-robot interaction in humanoid robots.

Non-verbal communication is performed using eye contact, facial expressions, touch, posture, gestures and others. Amongst these, hand signs are the easiest to be controlled and one of the primary forms of non-verbal communication [4]. Though other studies focus on using non-verbal communication, this thesis aims to achieve human-robot interaction by establishing a non-verbal communication between a human and humanoid robot, mainly using hand-signs. It presents a prototype that includes the NAO humanoid robot from Softbank Robotics which can interact with its user using hand-signs [5]. Hand-signs recognition is the most significant aspect carried out via the Hand-Signs Recognition Component (HSRC) of the prototype. In simplest terms, the NAO robot captures an image via its camera and sends it to the HSRC which then, recognizes the hand-sign present in the received input image. After recognition, NAO performs specific tasks mapped to the detected hand-sign. For example, playing a favourite song after recognizing a 'Thumbs-Up' sign, or giving weather information after recognizing a 'OK' sign. NAO is programmed using its proprietary software suite called Choregraphe, and the Hand-Sign Recognition Component uses Deep learning techniques like Convolutional Neural Networks, that are described later in the thesis. The research in this thesis is interdisciplinary, spanning the areas of computer vision, robotics, artificial intelligence and deep learning.

## 1.1 Background and Motivation

The field of robotics has experienced enormous growth since the 20<sup>th</sup> century and expected to boom further in the near future, with estimates of humanoid robot markets reaching USD 5.5 billion by 2024 [6]. The applications of humanoid robots have expanded to newer domains of healthcare, education and home robots. However, in order to facilitate a higher use of robots in our daily lives, it is important that these robots offer a high degree of communication and interaction with humans [7]. Human-Robot Interaction (HRI) focuses exactly on this and aims to make modern-day robots more acceptable to humans.

Verbal and Nonverbal communication are two mediums for Human-Robot Interaction. With recent advancements in Natural Language Processing (NLP) [8],

verbal communication is widely used in humanoid robots for communication but non-verbal communication still needs more work. Nonverbal communication is the subtle yet effective act of responding or communicating without using any words [9]. It has played a significant role in human interaction for many centuries [10]. Different types of non-verbal communication include hand-signs and gestures, facial expressions, posture, eye contact and others [11]. Hand signs and gestures are forms of non-verbal communication in which a speaker uses hand movements when talking to others. It has been an integral part of the language that allows people to express their emotions and improve the level of communication among them. People from different cultures use hand signs and gestures when they talk. Even congenitally blind individuals, who have never seen anyone gesture, move their hands while talking. Thus, it highlights the robustness of hand signs and gestures in communication [3]. The project in this thesis focuses on these aspects to establish human-robot interaction using non-verbal communication via the hand-signs.

Recognizing the hand-signs is the most significant task in this thesis. It is achieved using advanced deep learning techniques, specifically the Convolutional Neural Network (CNN). Deep learning is a recent development of artificial intelligence and a sub-field of machine learning that involves many neural networks to produce an output without any human intervention [12, 13]. Modern advancements in deep learning research have led it to become an ideal choice for image classification problems, such as the hand-signs recognition in this research.

## 1.2 Problem

With the rapid developments in the field of humanoid robots [6], robot designers are always finding different ways to enhance human-robot interaction and improve its acceptance in the real world [7]. Human-Robot Interaction can be achieved via verbal or non-verbal means. Advancements in Natural Language Processing (NLP) and speech recognition has improved the verbal part [8], but non-verbal communication - though an integral part of human interactions is thinly incorporated in the actual world for humanoid robots [14]. Non-verbal signs such as hand-signs, facial expressions, postures, and others give additional information and meaning emphasizing verbal communication of an individual. Some studies estimate that around 70-80% of communication is non-verbal [15]. Indeed, it suggests that using non-verbal communication can help robot designers to improve the human-robot interaction for humanoid robots. The research work in this thesis explores the use of non-verbal communication in a humanoid robot to interact with people, especially using hand-signs.

### 1.3 Purpose

The purpose of this thesis is to achieve human-robot interaction via non-verbal communication in humanoid robots, especially using hand-signs. Humanoid robots struggle to attain natural interactions with people, making them less acceptable for usage. Understanding non-verbal communication would make the interactions more natural and thus, improve the human-robot interaction in humanoid robots. To achieve it, a Hand-Sign Recognition Component (HSRC) is developed to recognize the hand-signs shown to an NAO humanoid robot [5] by a human subject. The HSRC is loosely-coupled and can easily be disintegrated from the NAO robot providing greater flexibility and re-usability. The work in this research, with some extensions, can serve as an application to aid people with hearing disabilities.

### 1.4 Goals

This research explores the use of non-verbal communication in a humanoid robot by interpreting human sign language and establish interaction with humans. These two aspects - the Humanoid robot and Sign Language Interpretation, are the most significant aspects of this research implementation. The NAO humanoid robot actualises the former, and the Hand-Sign Recognition Component (HSRC) achieves the latter. The NAO robot establishes the interaction with humans via a task-based scenario. In this scenario, NAO captures the hand-sign via its camera, recognises the hand-sign with the help of HSRC, and finally performs a specific task assigned to the detected hand-sign. The HSRC uses a deep learning model employing Convolutional Neural Networks (CNN) that predicts the hand-sign in an input image. Further, this research also assesses people's opinion on using non-verbal communication to interact with a humanoid robot via online questionnaires.

### 1.5 Methodology

A research methodology represents the means, procedure, or technique used to carry the research in a logical, orderly and systematic way [16]. It offers a set of practices of analysing different methods, implying a set of principles and rules for managing the research project [16]. When a problem is studied, the researcher has a certain a priori assumptions affecting the way the research is perceived and its final result [16, 17]. Research methods ensure the results are trustworthy meaning they are valid independent of one's personal experiences[16]. There are two types of research methodologies to ensure validity-



Quantitative methods and Qualitative methods. Quantitative methods focus on understanding how to construct something, to build a thing, or understanding how it works. Whereas Qualitative methods focus on why to build something or what is its significance. Quantitative methods strive to formulate laws, theories or principles for a phenomenon, but Qualitative methods aim to observe and deepen our knowledge and understanding of that phenomenon [17].

The Hand-Sign Recognition Component and the humanoid robot are two primary components of the system developed in this project. Examining some of the challenges with both these components like lack of firm guidelines to develop deep learning models in the HSRC, and the software limitations of the NAO robot, this research applies **Qualitative research** methodologies rather than quantitative, to handle such a high degree of uncertainty in this scenario [18].

This research follows interpretivism philosophy and development research methods [19] to achieve its goal of establishing non-verbal communication in humanoid robots. It uses design science research strategies, more precisely - Software Engineering principles to build a prototype of a humanoid robot recognising different hand signs. This research uses inductive reasoning [17] based on the observations throughout the development phase and online questionnaires to assess people's opinion of using non-verbal communication to achieve HRI. Initially, data collection was planned by taking questionnaires and noting people's opinion of the developed prototype, but it was difficult, given the COVID-19 pandemic outbreak [20]. Instead, this research employs online questionnaires that contain a video demonstration of the prototype, followed by a list of questions to assess the research. Data collected by online questionnaires are analysed using statistics [17] that help to express the people's acceptance to use non-verbal communication in humanoid robots. Quality assurance of the research is achieved in terms of attaining validity, transferability and dependability [17]. Validity refers to if the system uses state-of-the-art knowledge (content validity) and various components to be consistently linked to each other (construct validity). Dependability corresponds to reliability of the research process and transferability is to create richer descriptions that become a database for other researchers [17, 21].

## 1.6 Contribution

This research presents a way for robot designers emphasizing the use of non-verbal communication (especially hand-signs) to establish human-robot interaction in humanoid robots. The contribution of this thesis is mainly two folds. The first being the prototype system developed in this thesis that demonstrates

a humanoid robot recognizing the user's hand-signs and performing useful actions based on the detected signs. The HSRC is loosely coupled and operates independently from the rest of the system, offering simple integration to other systems requiring just the signs recognition functionality. Thus, providing an immediate solution to people with speech impairments. This thesis implicitly provides a step-by-step procedure to develop advanced deep learning models for image classification problems and related research areas. The second contribution is to essentially highlight the potential of incorporating non-verbal communication to achieve Human-Robot Interaction in Humanoid Robots. This research adopts and showcases an unconventional way of collecting data using online questionnaires. Moreover, both of these contributions add to the existing literature in the fields of Computer Vision, Robotics and related research communities.

## 1.7 Delimitation

The system developed in this research achieves human-robot interaction via non-verbal communication using only hand-signs because hand-signs are adopted universally in conversations and are more robust than other forms of non-verbal communication. Though the system recognizes hand-signs of the user and performs most of the tasks without verbally communicating with the user, few of the basic commands like "Yes" or "No" still require verbal confirmation. The Hand-Signs Recognition Component of the system currently recognizes the following three hand-signs: 'Palms-Open', 'Thumbs-Up', and 'OK' sign. To add more hand-signs, one must produce an adequate number of images for every sign and may consider implementing image preprocessing techniques for better model performance. The research evaluation did not receive lot of participants, but the majority of responses are convincing to direct robot designers to adopt more to non-verbal communication features in humanoid robots.

## 1.8 Outline

Moving forward, this thesis follows the following structure: Chapter 2 and Chapter 3 provide a background on humanoid robots and theoretical understanding of deep learning concepts. Chapter 2 first describes the humanoid robots, introduces the NAO humanoid robot used in this research along with its features and software development kit. Later, it discusses the importance of hand-signs in non-verbal communication and different types of hand-signs commonly used in daily interactions.

Chapter 3 describes the fundamentals of deep learning along with different types of neural networks, reflects the theoretical concepts of how to train a deep neural networks, and further presents related works about hand signs recognition systems using deep learning.

Chapter 4 discusses the research methodologies used in this research. Qualitative research methodologies that adopt the Development research and Software Engineering methods are used as a means to conduct system development and research in this thesis. It also describes a reformed technique of online questionnaires used in collecting the data for research evaluation.

Chapter 5 describes the designing process of the prototype. It specifies the system requirements, illustrates the system architecture of the prototype system and how the human-interaction is established by the prototype in this thesis. Chapter 6 provides a detailed description on how each of the components of the prototype are developed.

Chapter 7 first presents the performance of the deep learning models developed by the HSRC and second compiles the results obtained from the online questionnaire revealing people's opinion on the developed prototype. Chapter 8 reflect over different phases in the entire research, the choices made, and findings from the research.

And Chapter 9 concisely summarizes the purpose of this research, contributions and impact of this research including suggestions for future work.



# /2

## Humanoid Robots and Importance of Hand-Signs

This section provides an overview of humanoid robot including the NAO robot [5] used in the thesis, the importance of non-verbal communication in human-robot interaction and fundamentals of developing deep learning models along with its theoretical concepts.

### 2.1 Humanoid Robot

A Humanoid robot is a robot that resembles a human's physical appearance (like having a head, torso, arms, or other body parts) that communicates with humans, interprets the collected information, and acts according to the user's input [1]. Every humanoid robot may not entirely look like a human. Some humanoids can only resemble a specific part of the human body - like the head and may miss the arms and legs. Humanoids resembling the male humans are called Androids, and those resembling the female humans are called Gynoids. Leonardo DaVinci developed the earliest form of a humanoid robot in 1495, which resembled an armoured knight who could stand, sit and walk like a human [22]. Traditionally humanoids were invented to provide better orthotics and prosthetics for humans [22], but nowadays they are being used as research tools, to carry out different tasks and play various roles in

our lives [22]. Modern-day humanoids act as personal assistants, receptionists, caretakers, entertainers, and assist humans in several types of activities. They have the highest potential to become the most useful industrial tool in the future. Humanoid robots are excelling in the medical industry, especially as companion robots [1]. Companion robots are a special kind of robots specifically designed for personal use at home. Ideally, they should communicate with humans naturally, perform a wide variety of tasks including daily chores, message delivery, home security, et cetera. Another type is the social robot that intends to interact with humans and other robots to accomplish an entire job function, like greeting or basic customer service. Some of the popular humanoid robots are Kuri - a home robot designed to interact in a family, Sophia - a first social robot to acquire national citizenship, Pepper - a humanoid robot used in many businesses and schools, NAO - an autonomous robot widely used for research, and many others [23]. This research uses the NAO robot as a tool to incorporate non-verbal communication to build the human-robot interaction with its user.

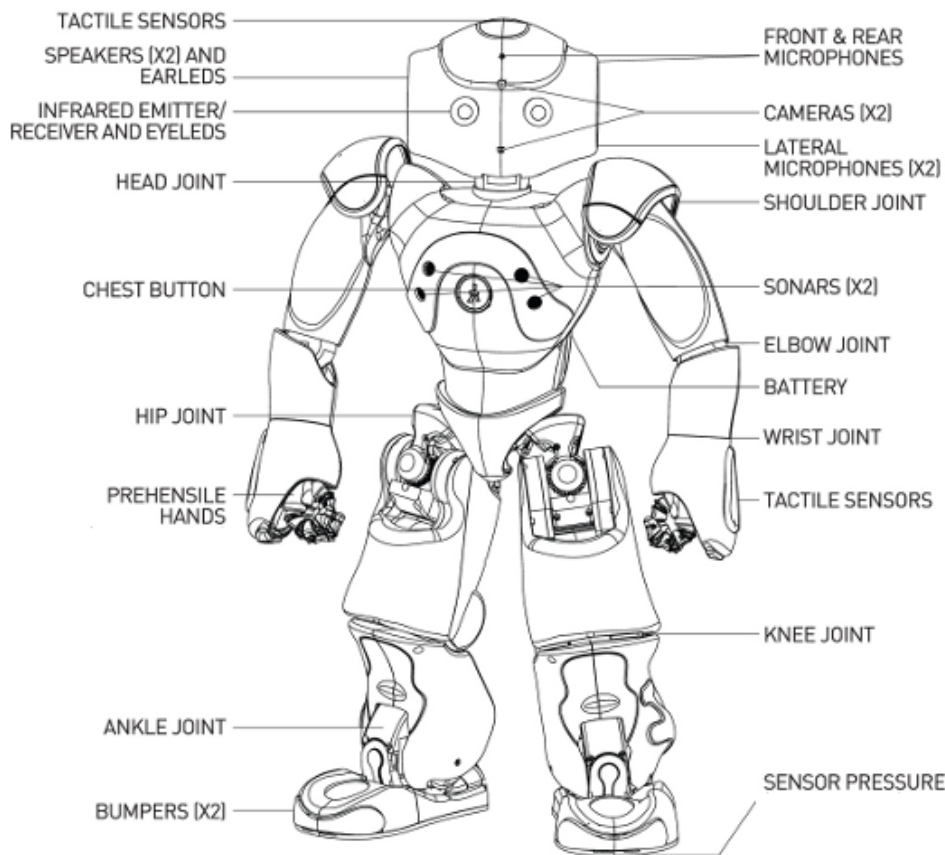
### **2.1.1 Introduction to NAO humanoid robot**

The NAO robot is an autonomous, programmable robot developed by Aldebaran robotics in 2006 [24], which was later acquired by Softbank Robotics in 2015. NAO is 56cm tall, having up to 25 degrees of freedom, with an ability to interact with humans in multiple languages, recognizes human faces, and other advanced features [5].

### **2.1.2 Key components and Features of the NAO robot**

NAO robot possesses various sensors and motors along with several programmable APIs operating on the Intel ATOM 1.6 GHz GPU processor, with 1 GB RAM, 2 GB flash memory, and 8 GB micro SDHC storage capabilities. It is accessible via Ethernet or Wi-Fi [25]. NAO has 62.5 Watt/hour battery providing about 1.5 hours of autonomy, depending on usage[25]. Among the many, the following features are of most relevance in this research:

1. NAO has two front cameras that can capture images with resolution from 160x120 up to 1280x960 [26]. None of these cameras is placed in the "eyes". One camera is on the forehead, and the other is placed at the "mouth". These cameras have 72.6° Field Of View (FOV) with 60.9° horizontal FOV and 47.6° vertical FOV [27].
2. NAO has four directional microphones and speakers offering multilingual language support to interact with humans [28]. Speech recognition mod-



**Figure 2.1:** Sensors and Actuators of NAO robot, Source: Taken from [25]

ules are easily configurable to produce audio outputs of the recognized signs and any other activities.

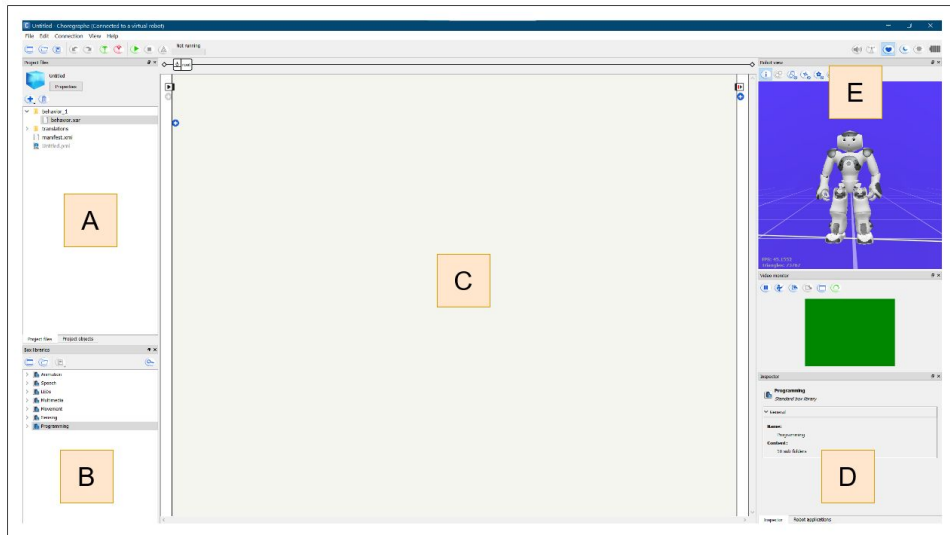
### 2.1.3 The NAOqi Framework and Choregraphe

NAOqi is the main framework based on a Linux based operating system offering cross-platform software development for the NAO robot [29]. It provides highly featured APIs in C++ and Python [30, 31]. NAOqi APIs are separated in different parts, each allowing access to a unique set of functionalities based on one's requirements. Below are the modules of more importance to the work in this thesis:

1. **NAOqi Core:** NAOqi Core contains API that is always available to run general-purpose tasks like network connections, memory, and event handling, etc [32].

2. **NAOqi Audio:** NAOqi Audio helps to set up an audio communication and interaction with the user in multiple languages [32].
3. **NAOqi Vision:** NAOqi Vision allows us to use the cameras to capture images or record videos [32].

Choregraphe is a multi-platform desktop application using the NAOqi framework that allows us to create applications and monitor the NAO robot without writing a single line of code [27]. One could create animations, configure different behaviours and dialogues, add a customized module in Python, and test everything on the robot using Choregraphe. The Figure 2.2 shows an overview of the main window of the Choregraphe tool that contains many panels annotated by different letters in the image. The Table 2.1 provides a brief description of these panels [33].



**Figure 2.2:** Overview of the Choregraphe tool

**Table 2.1:** Main components of Choregraphe [33]

Component Label	Name of the panel	Description
A	Project Content panel	Displays the properties of the project and all the files belonging to the current project
B	Box Libraries panel	Displays the list of programmable modules and behaviours available in the NAO robot
C	Flow Diagram panel	Displays the behaviours and their interconnections that are currently used in the project
D	Robot View panel	Displays a 3D view of the robot Choregraphe is currently connected
E	Pose Library panel	Displays specific poses for the NAO when creating a behaviour



## 2.2 Importance of Hand-Signs in Non-Verbal Communication

With all the recent advancements in creating a humanoid robot, it requires high-quality interaction with humans to be more acceptable in regular use [7]. The field of Human-Robot Interaction studies exactly these interactions between humans and robots.




The field of Human-Robot Interaction (HRI) is dedicated to understanding, designing, and evaluating robotic systems for human use. The HRI problem is to understand and shape the interactions between humans and robots [34, 35]. HRI helps to understand and perceive human's behaviour, encouraging the robots to collaborate with humans in different scenarios. Information exchange is an intrinsic part of an interaction [34]. The primary medium for information exchange is via verbal and nonverbal communication. Verbal communication includes speech and natural language and using visual displays (graphical user interfaces). Nonverbal communication includes gestures (hand and facial gestures), physical interaction, and haptics (use of human's sense of touch) [34]. Advancements in Natural Language Processing and speech recognition achieve verbal communication, but non-verbal communication gets neglected, even being an integral part of human interactions [14]. This research focuses on using non-verbal communication to improve human-robot interaction in humanoid robots.

Speech and gestures commonly form the building blocks of human interaction [36]. The former is a verbal and orderly means of how humans communicate while the latter is non-verbal means in which bodily actions reinforce particular aspects of the communication [37]. Typically, gestures refer to arm, hand or head movements. Gestures perform many intrapersonal and interpersonal functions beneficial to both - the person doing the gesturing and other(s) who receive it [36].

There are two types of gestures:

1. **Speech-independent gestures:** Gestures occurring independently of the speech that has a direct verbal translation, with a word or phrase, are categorized as Speech-independent gestures [38]. It is important to remember that these gestures are highly dependent on a specific region, surrounding culture, and others [36]. For example - The 'V' sign (palms facing outside) shown in Table 2.2 is usually a peace sign in the United States, but in the United Kingdom, it may be considered to be an obscene gesture [39].

**Table 2.2:** Common hand-signs and their meanings

Sign	Meaning
 'Thumbs-Up' Sign	Approval/Acceptance to something
 'V' Sign	In US, the outward-facing-palm indicates peace. In UK, the inward-facing-palm is treated obscene.
 'OK' Sign	Symbolizes everything is okay
 'Vulkan-V' Sign	Modified V-Sign signifies to live long and prosper

2. **Speech-dependent gestures:** Gestures that occur along with the speech that emphasizes the speaker's words in communication. These gestures mostly happen subconsciously [36]. For example - a manager pointing towards a specific person when introducing his team to others.

Some of the commonly-used hand-signs are shown in the Table 2.2. Among the type of hand-signs described above, speech-independent gestures will have higher interpretability in establishing human-robot interaction in humanoid robots. The next Chapter 3 explains the fundamental concepts of deep learning useful in developing a hand-signs recognition system.

# / 3

## Deep Learning Neural Networks

Hand Signs Recognition is a process of identifying a few hand-signs shown to the system via an image or video format. In mathematical terms it is simply considered to be a classification task. The resulting system must classify the input data (an image or video) into one of the defined classes (the hand signs). This research uses state-of-the-art approaches of Deep Learning to recognize hand-signs for an input image. The following sections provides a brief overview of what is deep learning, the fundamental concept of deep learning and various deep learning techniques (like Artificial Neural Network (ANN) and Convolutional Neural Network (CNN)) used in this research.

### 3.1 Deep Learning

Deep Learning is a sub-field of Machine Learning [40] inspired by the structure and function of the brain to improve the efficiency of learning algorithms. Before diving into the concept of deep learning, one must understand the working of learning algorithms - also described as **Machine Learning**.

Arthur Samuel explains Machine learning as the science of giving "*computers the ability to learn without being explicitly programmed*" [41]. Machine Learning

algorithms take input data samples and find a statistical relationship that eventually results in the automation of the original task. Such algorithms are different from traditional programming, in the sense that, they do not need static program instructions but make data-driven decisions through building models from sample inputs [42]. Suppose a system has to recognize if the input image given to it contains a dog or a cat, the machine learning model then acts as an image classifier. It uses the features in the image (shape of eyes, nose, ears, whiskers, body colour and others) to classify it into a dog image or cat image. An image has many features, but only a few are relevant to the classification task (the shape of eyes, nose and ears are significant, but the body colour might not be very suitable when classifying a dog from a cat). This process of extracting relevant features is called **Feature Extraction**. Machine Learning algorithms need feature extraction before beginning the classification task. Whereas Deep Learning algorithms implicitly handle both these processes via their hidden architecture called neural networks [40].

Another difference between deep learning and machine learning is how the model learns about these features for the given task. For the image classification task, machine learning models learn features explicitly in succession and therefore, cannot recognize complicated features (like the distance between the eyes or length of the face). On the other hand, deep learning models can determine complicated features from low-level features incrementally in their deeper layers. And these intermediate and incremental representations are learnt collectively [43].

The idea behind these representative layers in deep learning is analogous to neural layers in the human brain. The human brain consists of billions of neuron cells connected to form a network called the **Neural Network** [44]. Figure 3.1 illustrates the structure of a neuron in the human brain. Each neuron consists of a cell body, dendrites, and axons. Dendrites accept input signals from sensory organs or other neurons. The cell body processes the information and axons transmit the information to other neurons. Dendrites and axons do not physically touch each other, but there exists a tiny space between them called synapses. A neuron can forward the message further to another neuron or choose not to do so. Deep Learning strives to simulate the biological neural network in a machine by creating an **Artificial Neural Network (ANN)**.

## 3.2 Artificial Neural Network (ANN)

Artificial Neural Network is an approximation of biological nervous system of living organisms that consists of a collection of connected units called **artificial neurons**. Figure 3.2 shows the structure of an artificial neuron. The

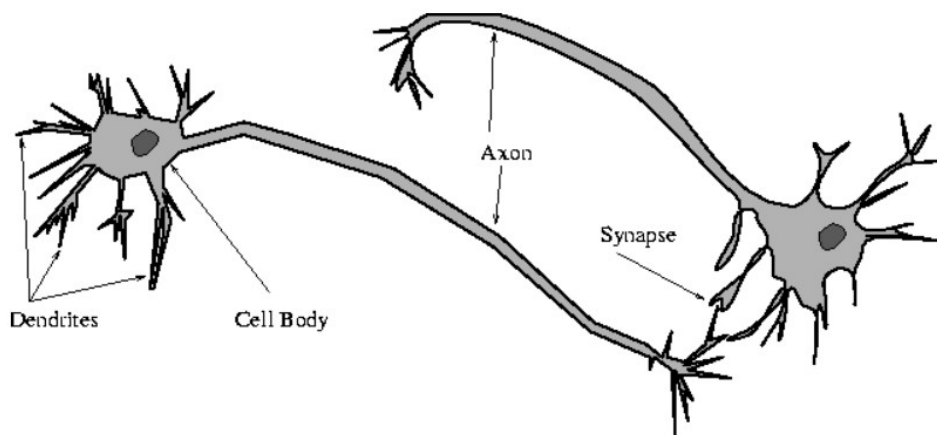


Figure 3.1: Structure of Biological Neuron, Source: Taken from [44]

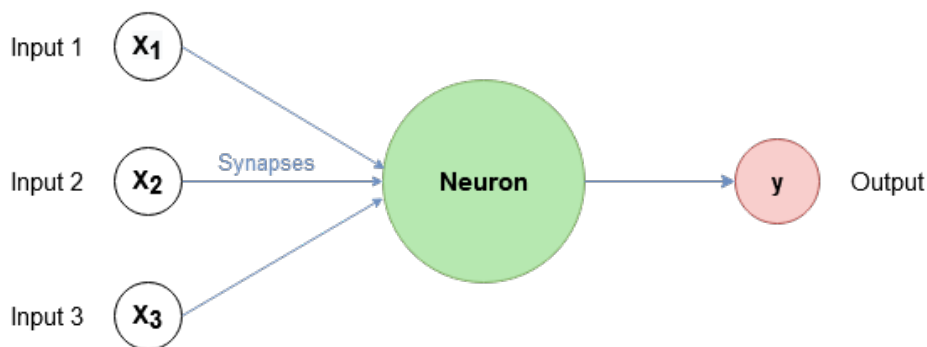


Figure 3.2: Structure of Artificial Neuron, Source: Adapted from [45]

synapse resembles a connection of one neuron transmitting information to another neuron. A neuron receives multiple input values similar to dendrites in the brain, usually denoted by  $X_i$  where  $X$  is the  $i^{th}$  input unit. These inputs are assigned weights  $w$ , real numbers expressing the importance of the corresponding input unit to predict the output  $\hat{y}$ .

The neurons that initially accept the input values form the **Input Layer** and that which produce the final output form the **Output Layer** of the ANN. The layers between the input and output layers are called **Hidden Layers**. There could be several hidden layers in the ANN architecture. Figure 3.2 shows an architecture with a single hidden layer, which is also called as a **Single Layer Perceptron** model. ANNs with two or more hidden layers are called **Multilayer Perceptron** model [46].

Deep learning strives to find a relation between the input and output variables

during the training process. The input layer receives independent variables  $X_i \{i = 1, 2, \dots, m\}$  of a single observation in a dataset. For example, if an ANN model predicts the selling price of an apartment flat, independent variables could be number of bedrooms, size of the house, nearest airport, etc. The input layers are generally standardized or normalized to speed up the training or prediction process. In simpler terms, standardizing or normalizing the input data supports the computations inside a neuron for faster results. The output layer neurons produce either a continuous value (predicting the selling price of a flat), binary value (determining a fraudulent customer for a bank) or categorical values (detecting different objects in an image).

Each neuron receives the input from the previous layer, either from an input layer or previous hidden layer. Each input value is assigned with some weight value  $w$  that gets adjusted during the training phase. These weights ultimately form the deep learning model. Inside a neuron, the weighted inputs are summed and passed through a non-linear function to produce the output. This non-linear function is commonly referred to as the activation function given by Equation 3.1.

$$\phi \left( \sum_{i=1}^m w_i x_i + b_i \right) \quad (3.1)$$

The term  $b_i$  is a bias value added to tune the weight  $w_i$  in a better way to improve the fit of the model. Note that the bias  $b$  is independent of the output of previous layers and do not interact with input data  $x$ .

## Activation functions

Activation functions introduce non-linear transformation to the input  $x$  that helps to learn and solve complex problems in deep learning. The deep learning model is a set of approximate values of the weights that produce accurate results for the use case. Activation functions essentially help to achieve this universal approximation of weights. There are various types of activation functions but most commonly used in deep learning are described below:

1. **Binary Step/Threshold function:** As the term **threshold** implies, the threshold function activates the output only when the input reaches a particular threshold value. It follows a strict nature producing either one output or none. It has a zero derivative. Therefore, it is not useful in

hidden layers but preferred in output layers [47].

$$\phi(X) = \begin{cases} 1 & \text{if } X \geq 0 \\ 0 & \text{if } X < 0 \end{cases} \quad (3.2)$$

2. **Linear Functions:** It generates a series of linear values and not just binary values. It has a fixed derivative, therefore linear functions also cannot be used to observe the learning rate in the network.
3. **Sigmoid function:** Sigmoid function is one of the most frequently used in machine learning problems. The output values are a smooth curve approximated between (0, 1). It is used in the output layer to predict probabilities for the outcome. For example, the output layer in image classification predicts the different classes (dog, cat, humans, etc.).

$$\phi(X) = \frac{1}{1 + e^{-x}} = \frac{e^x}{e^x + 1} \quad (3.3)$$

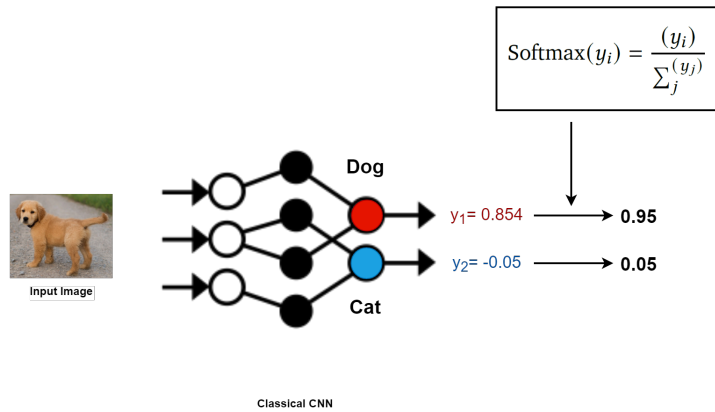
4. **Hyperbolic Tangent function:** The structure is very similar to the Sigmoid function. However, it produces output between (-1, +1). The advantage is a steeper derivative than the sigmoid function and broader range of input values which are useful for some use cases.

$$\phi(X) = \frac{1 - e^{-2x}}{1 + e^{-2x}} \quad (3.4)$$

5. **Rectifier function:** Activation by the sigmoid and hyperbolic tangent functions are very intensive, increasing the computational load of the network. Rectifier functions reduce the computations by dropping the neurons producing negative values and accelerates the training process [48]. There are various variants of the rectifier function but most widely used is the Rectifier Linear Unit given by:

$$\phi(X) = \max(X, 0) \quad (3.5)$$

6. **Softmax function:** Softmax functions are useful in multi-class neural networks where the neural network has more than two outputs. Each of the output classes predicts the chances that an input image belongs to a class with a real number. For example, a binary image-classifier predicts that the chances of an input image to be a 'Dog' image is 0.854 and it to be a 'Cat' image is -0.05. To interpret these output values in simpler terms, softmax function normalizes these output values into a



**Figure 3.3:** Softmax function for binary image-classification

probability distribution - each output class represents a probability value and all output probabilities sum to one [49, 50]. The Softmax function is implemented just before the output layer and helps in faster convergence when training the network. If  $y_i$  represents the individual probability of an output class and  $j$  represents total number of classes the Softmax function is given by:

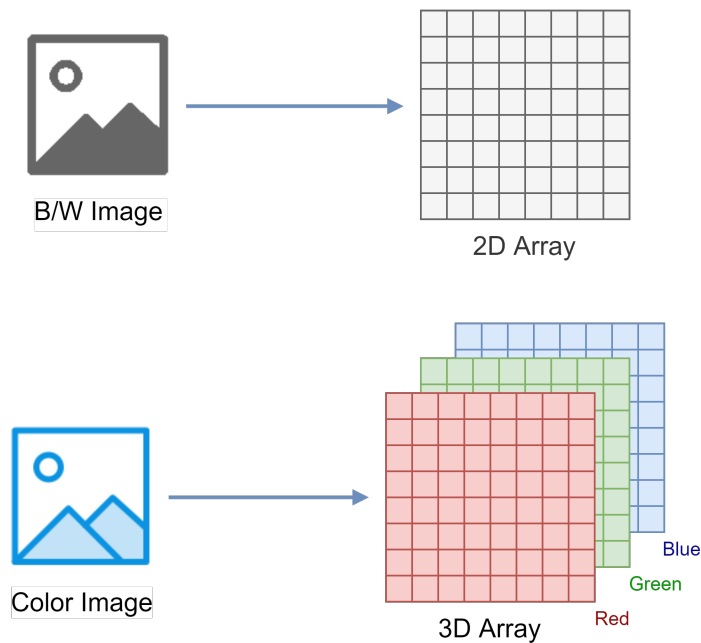
$$\text{Softmax}(y_i) = \frac{\exp(y_i)}{\sum_j \exp(y_j)} \quad (3.6)$$

Figure 3.3 shows the effect of using Softmax function for the previous example for binary image-classifier. The individual probabilities  $y_1 = 0.854$  and  $y_2 = -0.05$  are transformed to 0.95 and 0.05 respectively, giving a simpler interpretation that the network classifies the input image as a 'Dog' image.

### 3.3 Convolutional Neural Network (CNN)

Convolutional Neural Network is a deep learning algorithm that in simple terms - takes an input image, assigns some importance to the features in the image and classifies an image into the pre-defined classes (for example, a dog, cat, tiger, human, and so on). The architecture of CNN is similar to the connectivity of neurons in the human brain. The neurons in the Visual Cortex (part of the human brain responsible for handling visual inputs from the eyes) respond only in a restricted region of the visual field known as Receptive field. Similarly, the key to image recognition in CNN is to find such receptive areas, otherwise





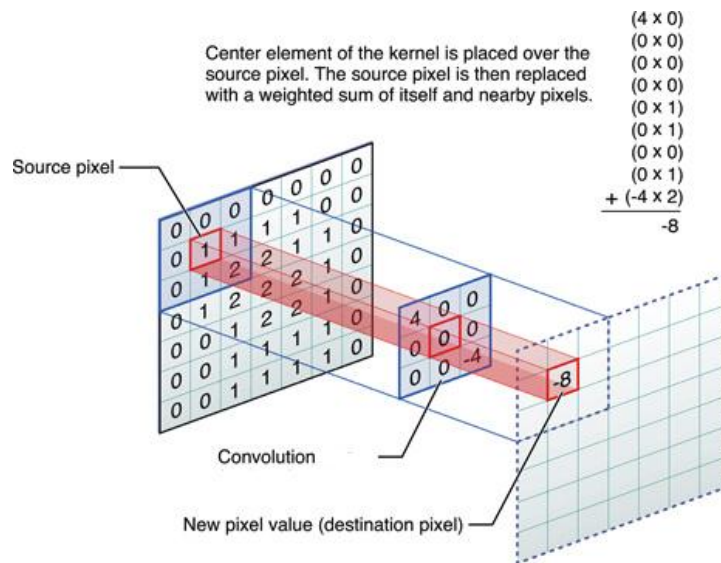
**Figure 3.4:** Matrix representations of a digital image, Source: Adapted from [51]

called **features** from the input image. An image is a matrix of pixel values based on image resolution. A black and white image converts to a 2D array whereas a RGB (Red-Green-Blue) colour image converts to a 3D array as shown in the Figure 3.4. An image recognition CNN follows four steps: Convolution, Max Pooling, Flattening and Fully-Connected Artificial Neural Network.

## Convolution

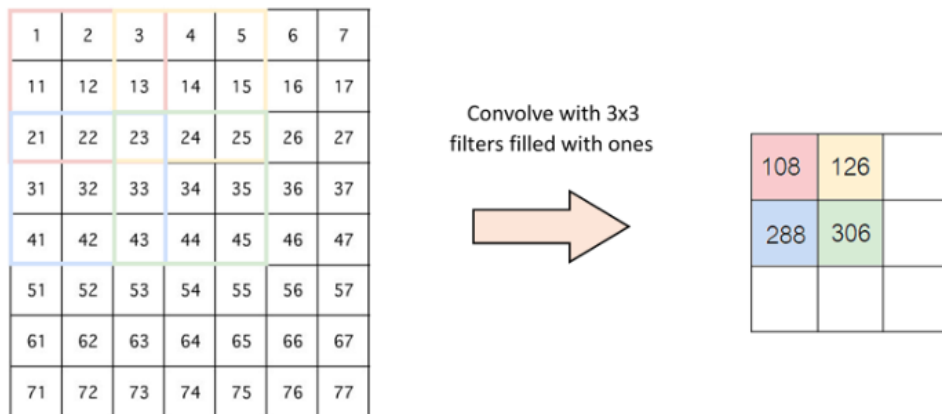
In image processing, **convolution** is a process of extracting features from the input image using a feature detector. A **feature detector** is a matrix whose size and values determine the extraction of features. A feature detector is also called **convolutional kernel** or **filter**. The convolution operation is a three-step process [52]:

1. Placing the feature detector matrix over the input image matrix.
2. Performing element-wise multiplication of these two matrices - computing the product between each value in the feature detector and corresponding input image and summing all such products into a single value. Consider a 7x7 input image and a 3x3 feature detector as shown in the Figure 3.5. The feature detector is placed over the input image on the source pixel to carry the convolution operation.



**Figure 3.5:** Convolution in CNN, Source: Taken from [52]

3. Move the feature detector by one pixel to the right and repeat above steps until the entire image is covered. The number of steps by which the feature detector moves is called the **stride length**. Figure 3.6 provides a visual representation with stride equals two for a 7x7 input image and 3x3 feature detector.



**Figure 3.6:** Convolution in CNN (Stride=2), Source: Taken from [53]

The output of the convolution is called a **feature map**, also referred as **convolved feature** or **activation map**. The importance of the convolution operation is to two folds - reduce the size of the input image to speed up processing and also extract certain essential features of the image for accurate predictions. One feature map can only extract one feature from the input image. Hence,

several feature maps are obtained using various feature detectors. Figure 3.7 shows 3 set of low level feature maps representing different facial features from a human face - eyebrows, eyes, nostrils and mouth [54].

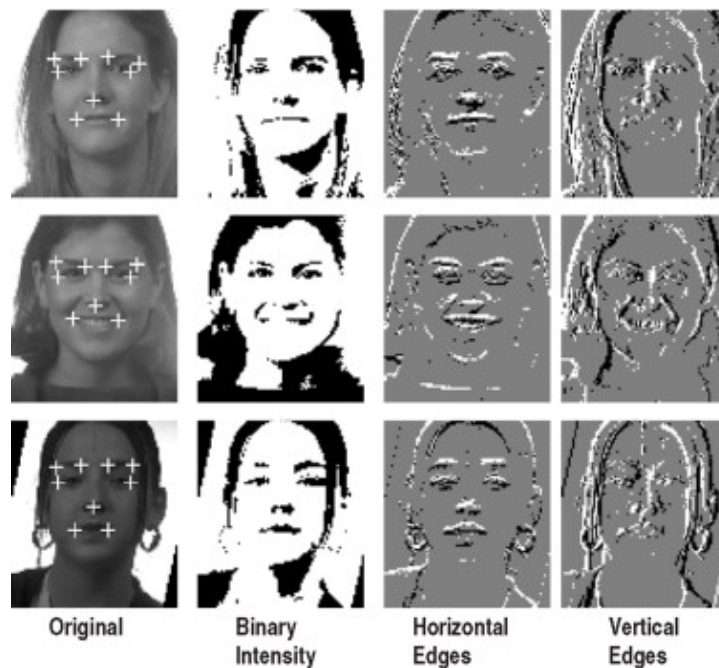


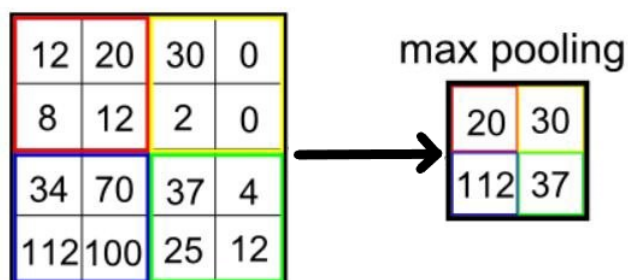
Figure 3.7: Examples of Feature Maps, Source: Taken from [54]

### Rectified Linear Unit (ReLU) Layer

A **Rectified Linear Unit (ReLU)** layer is applied to the feature maps to improve the non-linearity in the image. An image has various non-linear elements like - different objects, colours, borders, shapes, position and others. With convolution, there is a possibility to add some linearity in the resulting feature maps. Hence, a ReLU layer helps to break this possible linearity and improve the training process.

### Max Pooling

**Max Pooling** is a process to extract the dominant features which are rotational or positional invariant, thus preserving spatial variance in an image [55]. Max Pooling returns the maximum value from the portion of the image covered by the kernel. It discards the noisy features, reduces the image size and parameters preventing over-fitting and further enhancing the computational benefits. Figure 3.8 shows the max pooling operation on a 4x4 feature map.



**Figure 3.8:** Max pooling in CNN, Source: Taken from [56]

### Flattening

Pooled feature maps have two or more dimensions and cannot be directly used for computations to classify the input data. **Flattening** is a simple process of flattening the multi-dimensional pooled feature map matrix into a 1-Dimensional vector. This single long feature vector is then fed to a fully-connected Artificial Neural Network for further processing.

### Fully-Connected Artificial Neural Network

The final step in building the Convolutional Neural Network is adding a fully-connected ANN to the output vector from the previous step. The values in the output vector represent the probabilities of a certain feature belonging to the input image. For example, if the input image is of a cat, features representing whiskers would have a higher probability for it [57].

The **Fully-Connected Artificial Neural Network (ANN)** is simply an Artificial Neural Network (ANN) with densely connected hidden layers. It undergoes its entire backpropagation process to determine weights that prioritize the most appropriate output label. The second-to-last layer gets to 'vote' for each of the class labels (cat or dog), and the last layer outputs probabilities for these class labels (cat=0.79 and dog=0.21).

## Summary of CNN

To summarize the process of Convolutional Neural Network (CNN), it initially applies several feature detectors on the input image to create feature maps in the convolutional layer. A ReLU layer is used to remove any linearity in the process. Then a pooling layer is applied on the feature maps to ensure spatial invariance, reduce the size of the images and prevent overfitting of the model. Lastly, the pooled images are flattened and given to a fully-connected ANN that performs voting to predict the probabilities of the class labels. The training process involves forward and backward propagation that adjusts the weights of neurons in the full-connected layer and also the feature detectors to get best feature maps. Figure 3.9 shows the steps described in the section.

## Summary

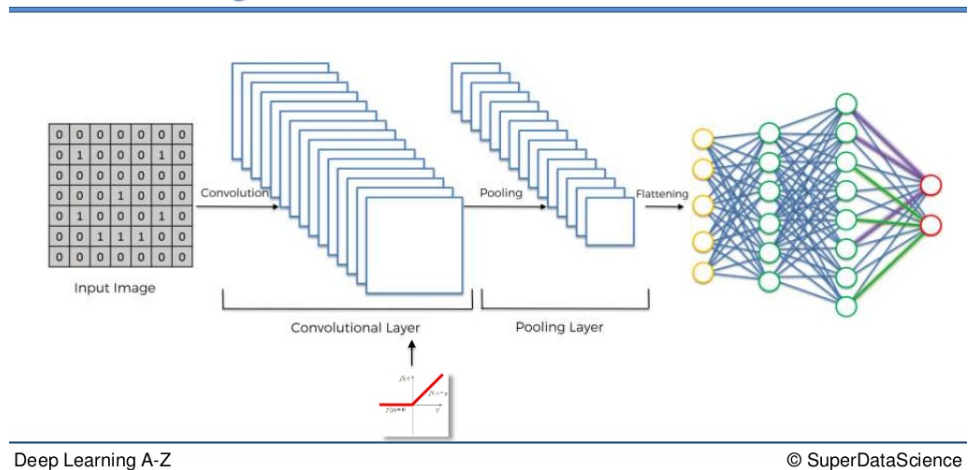


Figure 3.9: Steps in CNN, Source: Adapted from [51]

## 3.4 Training Deep Learning Neural Networks

In simpler terms, the goal of any deep learning model is the ability to generalize - that is to predict the results correctly for an unknown data (data that is never shown to the model). For image classification, the goal is to classify the input image correctly into a class used while training the deep learning model. Thus, it is crucial to evaluate the generalization for a deep learning model. This section briefly describes practices to achieve the generalization followed in every deep learning project like - data preprocessing, model evaluation and problems faced in developing a deep learning model.

## Data Preprocessing for neural networks

Data Preprocessing refers to all the transformations made to the raw data before feeding it to the neural network. Usually, the raw data is not clean - the data might have missing field values, different formats, outliers and features. The quality of training data determines the quality of the developed model. Hence, data preprocessing is a necessary step before training neural networks. Most general data preprocessing techniques are vectorization, value normalization, and handling missing values.

1. **Vectorization:** Vectorization is a process of converting the data (audio, image, text) to Tensors. A tensor is similar to an array representation of these data, almost always in a numeric format. Tensors are of different dimensions - single-digit scalars are 0-dimensional, vectors are 1-dimensional, matrices are 2-dimensional, audio signals, images, text are higher dimensional tensors.
2. **Value Normalization:** Data fed to the neural network must have values close to each other or in a similar range. Higher differences between the input and output variables may lead to lower learning rates and poor results. Normalization is a process of transforming the raw data variables to homogeneous format (all features take value in the same range, and each value lies between 0-1).
3. **Handling Missing Values:** Sometimes, values for some features are missing like the *last year's balance* for a new customer. It is necessary to handle these missing values when we develop the neural network to predict the individual's credit score. The neural network learns treats the value 0 to be 'missing data' and neglects in the prediction.

## Evaluating Deep Learning Models

Evaluating deep learning models is a process of estimating the generalization of the developed model on unseen data. It is necessary to know if the predictions are accurate and consequently, trustworthy before the actual deployment of a deep learning model. Below are the general guidelines for evaluating a deep learning model:

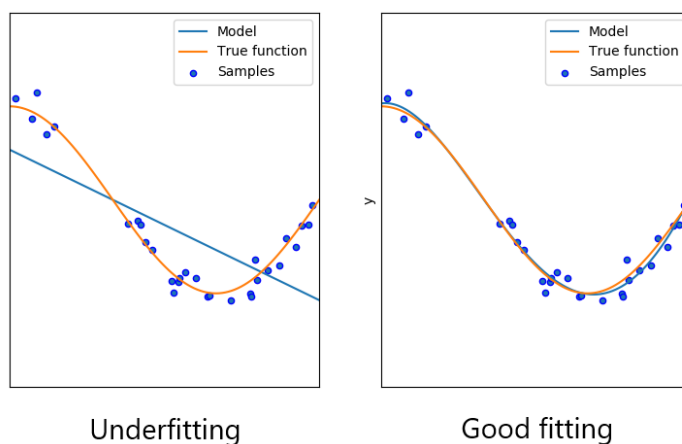
1. **Splitting data into a training set, validation set and test set:** Separating the collected data into three distinct groups (training, validation and test dataset) is useful in dealing with problems like underfitting and overfitting in the evaluation phase. The model is trained over the training set, evaluated over the validation set, and finally tested over the

test dataset, once before deployment. Common ratios are

- 70% train, 15% validation, 15% test dataset
- 80% train, 10% validation, 10% test dataset
- 60% train, 20% validation, 20% test dataset [58][59]

Developing a deep learning model involves tuning hyper-parameters of the model (like the number of epochs, batch size, steps in each epoch, etc.) which are different from the model parameters (weights of neurons, activation function, etc.) [60]. This tuning process is carried by evaluating the performance on the validation dataset. Also, the more you repeat this hyperparameter tuning, the model is indirectly learning the validation dataset, leading to a phenomenon called **Information Leakage**. Hence, a test dataset separated at the beginning helps to avoid both of these problems, and achieve a more robust model for deployment.

2. **Bias and Variance Trade-off:** The issue with deep learning models is to achieve generalization over unseen data (test data) by using optimization techniques entirely over the limited amount of existing training data. Handling this enigma establishes the performance and robustness of a deep learning model, but suffers problems related to a couple of factors in the process:
  - **Bias:** Bias refers to the simplifying assumptions a model makes to learn features from the available training data [61]. Based on how strong these assumptions are, models have one of the following:
    - (a) **Low Bias:** Models have very few assumptions about the training data decreasing the learning rate but improving the predictive performance of the model [61].
    - (b) **High Bias:** Models have more assumptions about the training data increasing the learning rate suffering from a lower predictive performance [61].
  - **Variance:** Variance refers to the change in predictive performance on using a different training dataset. Ideally, a model's predictive performance must not change too much from one training dataset to another, indicating the model is good at picking underlying features from the available dataset [61]. Based on the degree of these changes, models have one of the following:



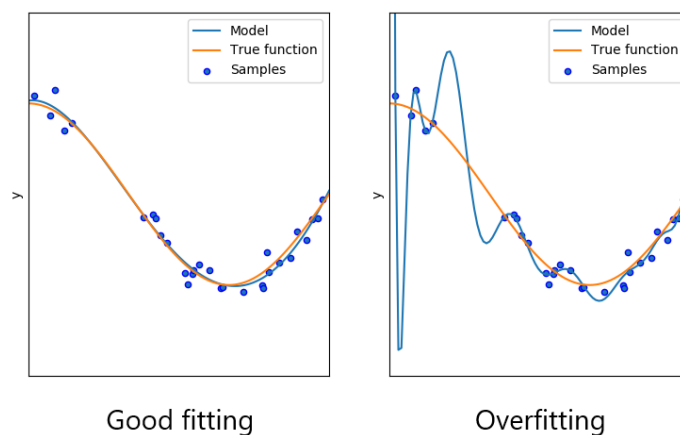
**Figure 3.10:** Underfitting vs Good fitting models, Source:[62]

- (a) **Low Variance:** It suggests small changes in the model's predictive performance on changing the training dataset [61].
- (b) **High Variance:** It suggests large changes in the model's predictive performance on changing the training dataset [61].

Ideally, a deep learning model aims to achieve low bias and low variance. In reality, bias and variance have a complex relation:

- Increasing the bias decreases the variance that leads to Underfitting [61].
  - Increasing the variance decreases the bias that leads to Overfitting [61].
- **Underfitting:** The phenomenon where the deep learning model has not learned enough relevant patterns/correlations in the training data, thus lacking the ability to generalize over unseen data. The rate of loss over the validation/test data is directly proportional to the rate of loss over the training data. Underfitting refers to a model that neither models training data nor the validation/test data. The figure shows a graph with two distributions of fit over data samples. In the left graph, the distance between the fit of the model and actual data points is very high as compared to the one in the right graph. The model in the right graph is close to the true function that represents the given data samples very well. Underfitting occurs due to inadequate training data or prematurely training the model before it achieves generalization. An underfitting model has high





**Figure 3.11:** Overfitting vs Good fitting models, Source: Taken from [62]

bias and low variance [62]. Increasing the size of datasets, data augmentation, training the model for longer periods are some of the common techniques to handle underfitting.

- Overfitting:** The phenomenon where the deep learning model has rigorously learned the patterns/correlations in the training data, thus missing the ability to generalize over unseen data. The rate of loss over the validation/test data increases in the later phases. Overfitting refers to a model that has learnt specific patterns in the training data that are irrelevant to unseen data. The right graph in the figure represents how accurately the model represents the data samples, but fails to match the **true function** describing the ideal distribution of those data samples. Overfitting occurs due to complicated models that learn from the noise and fluctuations in the training data which are unique to training data. Thus, preventing the neural network's ability to generalize. An overfitting model has low bias and high variance [62]. Reducing the network's complexity by dropping some layers, weight regularization (setting constraints on the model weights to have smaller values), and adding dropouts (randomly dropping some neurons in the layer while training) are some techniques to prevent overfitting.

## Optimizing Deep Learning Models

The goal of achieving generalization in deep learning is to find a perfectly fitting model in deep learning. The most central problem in deep learning is to reduce overfitting (where the training set accuracy is very high than

the validation/test set accuracy). Overcoming overfitting and enhancing the degree of generalization is done using various strategies or techniques, that are collectively known as **Model Regularization/Optimization** techniques [63]. Some of the common regularization techniques are discussed below:

- **Data Augmentation:** Overfitting usually occurs due to an insufficient amount of data samples to learn from that ultimately fails to achieve the desired generalization. Given unlimited data, the model would learn from every aspect of data distribution and never overfit [43]. Data augmentation generates more training data from the available training samples by augmenting the samples via several data transformations and yielding into similar-looking training data [43]. Data augmentation is very popular for computer vision problems because images or videos are high dimensional having many factors of variation that are easy to simulate. It includes operations like rotating, scaling, flipping, or translating a few pixels in each direction [63]. With the newly generated augmented images, the neural network does not see the same input image twice, but these images are highly correlated [43].
- **Dropout Regularization:** Dropout regularization for the neural network was proposed by Srivastava et al. in 2014. Dropout is a technique where randomly selected neurons are ignored or deactivated during training. Dropping some neurons limits their contribution to activation in subsequent layers in the forward pass and restricts weight updates in the backward pass. With dropout regularizations, neurons learn better representations without co-adapting with other neurons. Thus, it results in improving generalization and reducing overfitting [65].
- **L2 Regularization:** Another most commonly used and intuitive approach to reduce overfitting is to penalize the model and prevent the network from accurately modelling the training data. The optimization algorithm is now a function of two terms: **Loss term** that represents how well the model fits the data, and a **Regularization term** that describes the complexity of the model [66]. The Equation 3.7 shows the *L2* regularization where  $\theta$  represents a vector containing all the parameters of the neural network:

$$Err(x, y) = Loss(x, y) + \sum_i \theta_i^2 \quad (3.7)$$

- **Early Stopping of Training:** For sufficiently big datasets, training neural networks for a longer time reduces the generalization and results in overfitting. Early stopping helps to stop the training process as soon as

the validation error starts to increase and freezes the parameters, thus avoiding overfitting [63].

Optimization techniques help to avoid underfitting as well as overfitting problems. An ideal deep learning model balances the bias and variance just right as shown in the Figure 3.12. The validation loss and training loss are at minimum resulting in a higher degree of generalization for the model. When the validation loss gets higher than the training loss in

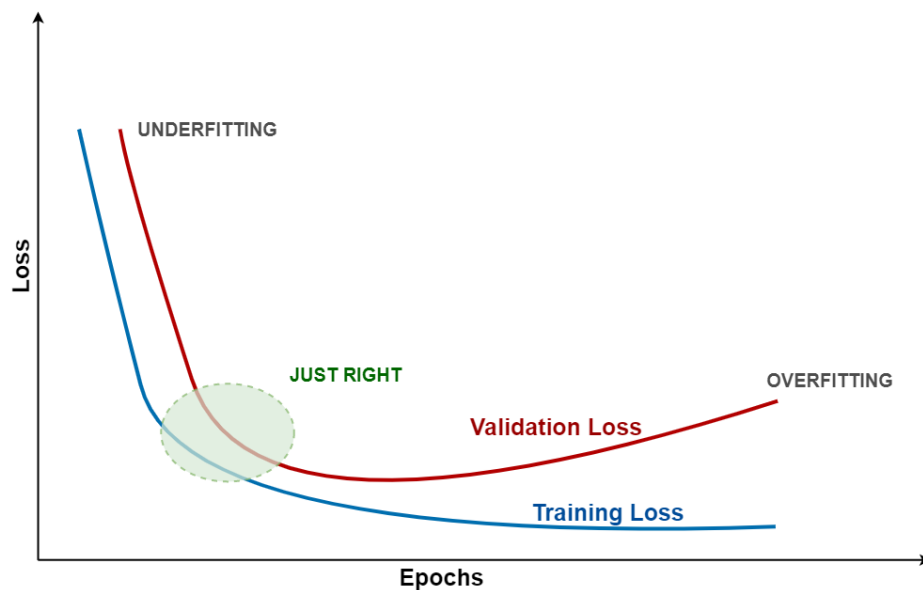


Figure 3.12: Model Complexity Graph, Source: Taken from [67]

### Selecting the final model

You create many models before arriving at the final model. Explain procedure on how intermediate models are chosen. How many models are chosen at last - single or many - combined.

## 3.5 Related Work

Deep Learning approaches hand signs recognition as an image classification problem. In simpler terms, image classification means - given a set of input images labelled with a particular hand sign (training data), the algorithm predicts the hand sign in a novel set of test images (images never shown to the algorithm) by learning certain features from the image.

Luo and Wu [68] proposed an innovative technique for Hand Signs Recognition called Combinatorial Approach Recognizer (CAR) that uses two recognizers aimed to complement the ability of discrimination. They implemented it on an autonomous mobile robot, *Ren-Q.Jr* developed in their NTU-IRA Lab. One recognizer recognizes hand gesture by hand skeleton recognizer (HSR), and the other using support vector machines (SVM). The hand skeleton recognizer includes several processes like skin colour detection and segmentation, distance mapping, and polygonal approximation. Support Vector Machines [69] are a dominant technique for pattern recognition applied after the Local Binary Pattern process on the input images. The rules of combination in the CAR are summarized that for every frame  $K$ :

1. If SVM is unreliable, combine SVM and HSR as CAR.
2. If SVM is reliable, use HSR to double-check it.

Experiments by Luo and Wu [68] showed that the combined method CAR has higher performance than the individual HSR and SVM. SVM needs lots of training data and is not very significant for the sign of two, three and four. However, CAR has a well average rate but lower for the signs of two, three and four. Their work implemented six hand signs on a service robot, as compared to the implementation in this master's thesis that uses three signs on a humanoid robot using Convolutional Neural Networks that works comprehensively well with less number of classes.

Tellaache, Kildal, and Maurtua [70] describe a gesture recognition module in a collaborative human-robot application, especially in industrial environments. The system estimate gestures relying on the processing of depth images to identify the operator skeleton and tracking its joints movements. The sensors used were Microsoft Kinect [71] incorporated in the Robot Operating System (ROS) [72]. They implemented 36 gestures using Adaptive Naive Bayesian classifiers from Machine learning. The experimental results are impressive, but it is applicable for static gestures from an RGB-D sensor input data gathered from Microsoft Kinect or similar device. In this thesis, the hand-signs recognition system uses color images recorded from a simple camera and does not need any depth information.

Bheda and Radpour [73] use deep learning techniques for gesture recognition. They present Deep Convolutional Neural Networks (Deep-CNN) that fundamentally classify (or recognize) letters and digits in American Sign Language. Deep CNNs are CNNs with denser architectures that automatically extract various features like edges, colour, discontinuities, lines, and textures from a simple input image. The architecture of Bheda and Radpour's model includes three groups - each group having 2 convolutional layers, a max pool and dropout

layer; and followed by two groups of fully connected layers, that achieves 82.5% accuracy on alphabet gestures and 92.7% on digits in American Sign Language [73]. Their implementation showed the potential to work with a simple camera without compromising performance to a greater extent. It served as a plan to overcome the hardware limitations with the NAO robot's camera and processing speeds by employing Deep Convolutional Neural Networks in this thesis.



# /4

## Methodology

### 4.1 Development Research

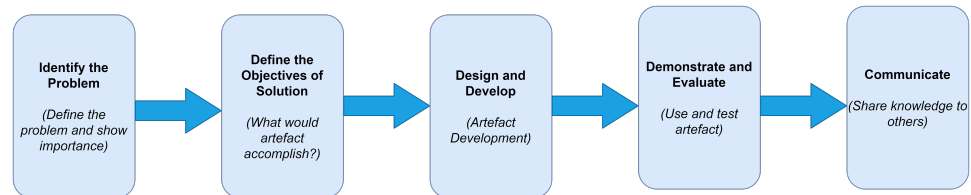
The research in this master's thesis adopts Qualitative research methodologies and implements Development research methods to conduct the research. Development research is problem-oriented, looking for new and innovative solutions, and also seeking findings that are transferable, practical and socially acceptable [19]. It has a dual focus:

1. **Development goal:** Develop an innovative way to solve a problem providing empirical evidence to achieve the research goal.
2. **Research goal:** Explore ways to use non-verbal communication in humanoid robots to improve Human-Robot Interaction and access people's responses towards adopting it.

The research work presented in this thesis presumes that non-verbal communication will improve human-robot interaction, as it does to inter-human interaction [14]. Thus, the immediate goal is to design and develop a system to interpret sign language for a humanoid robot. The researcher is actively involved in the developmental process to observe and reflect on his design choices which would also offer broad guidelines for similar use cases in the future.

This research uses **Design Science Research (DSR)** [74] strategies that provide

knowledge in the form of constructs, techniques, methods, models or theories to create an artefact that satisfies given set of requirements. In this research, it is the entire system that enables a humanoid robot to recognise hand signs using deep learning techniques. The most defining feature of DSR is to learn through building artefacts, and Peffers et al. [75] presents the following steps to learn via the DSR process model, as shown in Figure 4.1:



**Figure 4.1:** Design Science Research (DSR) process model, Source: Taken from [75]

1. **Problem Identification:** Identifying a research problem helps to develop an artefact and gives a conceptual understanding of the necessary steps required for its development. This research focuses to utilise non-verbal communication to improve human-robot interaction and also addresses a direct problem faced by people with speech impairments or hearing disabilities that are unable to make use of humanoid robots due to the verbal means of interaction. In technical terms, the developed system must recognize the hand-sign shown by the user and also perform a task mapped onto that particular hand-sign.
2. **Define the objectives of the solution:** After problem identification, goals are defined based on the knowledge gathered from problem specifications. For this research, the developed system must enable a humanoid robot to identify human hand signs and thus, hand signs recognition is an essential task in system development. The humanoid robot receives input in a form of an image, performs a prediction using the pre-trained deep learning model, and finally performs a specific task corresponding to the predicted hand-sign.
3. **Design and Development:** Once the objectives are well defined, one can start building artefacts in the form of models, methods, principles, constructs or any object in which a research contribution is conceptually rooted in the design. It requires theoretical and practical expertise to bear the solution. This research uses knowledge about state-of-the-art implementations of deep learning techniques to develop hand signs recognition and integrate it into the humanoid robot.
4. **Demonstration and Evaluation:** This step includes demonstrating the use of artefact (via functional testing, experiments, simulations, case



studies, or other appropriate activity) and evaluating the performance of the developed artefact. Evaluations must provide suitable empirical evidence or logical proof of the implemented solution. The developed system in this research involves a scenario where the user interacts with the humanoid robot to recognise a hand sign.

5. **Communication:** The final step is to present the problem along with its importance, and the developed solution with its utility, novelty, effectiveness to relevant audiences like scholars and professionals related to the field. The thesis presented here is itself a part of this step.

These steps for the artefact development are analogous to a **Software Development Life Cycle (SDLC)** model that involves problem understanding, planning, implementation, testing, and finally deploying the software system for actual use. SDLC falls under the paradigm of **Software Engineering** that systematically applies engineering principles to develop software systems [76]. This research uses Software Engineering principles as a means of the research method/strategy to carry out the design science research, intending the humanoid robot to interpret few hand signs. The development goal is to build an artefact, more specifically - a prototype simulating the hand sign recognition via a humanoid robot. And the research goal is to evaluate people's opinion on using hand-signs to operate and interact with the humanoid robot.

The following sections describe the concept of Software Engineering and Prototyping - that fulfill the development goal in this research.

## 4.2 Software Engineering

Software engineering is an engineering discipline that selectively applies different theories and practices to design, implement, test, and maintain the software in a systematic approach. The systematic approach is called a software process [76]. Software processes are a set of activities concerned with different aspects of software production:

1. **Specification:** activities that define the system goals.
2. **Development:** activities that describe the design and implementation of the system.
3. **Validation:** activities that ensure the delivery of expected results.
4. **Evolution:** activities that involve tuning the system in response to chang-

ing demands in the use-case.

A **software process model** is a simplified representation of a software process. Different types of problems require different software process models. For example, a security-critical problem is solved using the waterfall model by rigorous analysis before starting the implementation [76]. Business use-cases are solved using existing knowledge and integrating into a new system with added functionalities. Some of the popular software process models are Waterfall model, Incremental development, Boehm's Spiral Model, and many others [76]. Choosing an appropriate model depends on various factors such as - understanding the requirements, cost control, risk involvement, resource availability, the need of expertise, flexibility, overlapping phases, and numerous other factors [77].

In broader terms, this research has three phases:

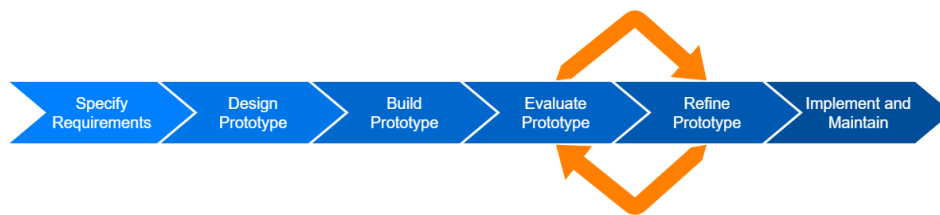
1. Development of Hand-Signs Recognition Component.
2. Configuring the NAO humanoid robot for the scenario.
3. Integrating the Hand Signs Recognition Component (HSRC) to the NAO humanoid robot

The first phase is the most critical since the Hand Signs Recognition Component (HSRC) defines the output of the overall system. Following that, configuring and integrating the NAO robot is an incremental process. Considering the risk involved with incompatibility issues of NAO robot and state-of-the-art deep learning implementations, limited documentation on using the NAO robot, troubleshooting required during the integration of NAO robot to HSRC, this research projects utilises Prototyping models for the system implementation.

### 4.3 Prototyping

**Prototyping** is a software process model where the developer(s) first create a prototype - an early version of the software system used to demonstrate concepts, analyse design options, to get a deeper understanding of the problem and possible solutions [78]. Figure 4.2 shows general steps in the Prototype Development:

1. **Specify Requirements:** A prototype model starts with gathering requirement analysis where the objectives of the prototype are defined. The goal of requirement gathering analysis is to identify the functional and non-



**Figure 4.2:** Steps in Prototype Development

functional requirements of the system [79, 80], to adequately understand the user's expectations and needs of the prototype.

2. **Design Prototype:** The second phase is making a preliminary design of the prototype. Often, it is not a complete design but specifies plans or frameworks to give the idea of the system internals to the developer or end-user. A quick design provides a blueprint to start the initial prototype development.
3. **Build Prototype:** In this phase, a prototype is built based on the information gathered in previous steps. It is a small working model that might involve parts of the system being developed incrementally instead of the entire system at once.
4. **Evaluate Prototype:** In business use-cases, this phase involves presenting the built prototype to clients for an initial evaluation to find the strengths and weakness of the model.
5. **Refine Prototype:** Refining includes tuning the prototype until the client is satisfied. This step continues until all the requirements for the end-user are satisfied.
6. **Implement and Maintain:** In large-scale projects, this phase involves testing, deploying the developed system in production, and maintaining it to prevent critical failures. This research implementation skips the maintenance phase since prototyping serves more, as a vehicle for an inquiry to generate understanding and explore the use of non-verbal communication in humanoid robot [78].

## 4.4 Research Evaluation

Evaluating this master's thesis involves evaluating functional testing of the prototype system that fulfils the development goal and understanding the

user's impression of the prototype developed for improving Human-Robot Interaction that satisfies the research goal. Functional Testing ensures that the components of a system are working correctly as per their specifications [81].

The functional tests aim to improve the performance of the prototype - particularly in terms of **Accuracy** and **Response time**. Accuracy is the ratio of correct predictions to total input samples [82]. Response time is the time taken by the robot to predict the hand sign shown by the user.

After developing the prototype, it is used in a task-based scenario to evaluate how users perceive the prototype and evaluate the research goal of using non-verbal communication to improve Human-Robot Interaction. For evaluation, one requires to collect data relevant to the research goal and later analyze it to derive meaningful results. Data collection is the process of gathering and measuring information on variables of interest in a systematic manner that enables one to answer the research questions, test hypotheses and evaluate outcomes [83]. It typically helps to maintain the integrity of the research. Due to the exploratory nature of qualitative research, all the data collected is significant. The researcher neither restricts the scope of data collection in advance nor applies the formal rules to decide what data is inadmissible or irrelevant. Different data collections methods used in qualitative research are:

1. **Interviewing:** An interview requires a systematic approach that is useful to explore experiences, views, opinions, or beliefs on specific topics. Depending upon the approach, there are several types of interviews -
  - (a) **Structured interviews** where the interviewer prepares the questions before the interview with a limited set of answers. The interviewer plays a neutral role and does not influence the participant's choices [84].
  - (b) **Semi-structured interviews** where the interviewer develops an 'interview guiding document' - a list of questions and topics to be covered in the conversation with the participant. Here the interviewer may engage more freely with the participant and guide them when felt appropriate [84].
  - (c) **Unstructured interviews** where the interviewer has developed enough understanding of the research topic, and a clear plan in mind regarding the focus of the interview. There is no structure to the interview and contain open-ended questions to explore the depth of the researched topic [84].

- (d) **Informal interviews** are typically useful in observing social phenomena and works best in the early stages of development where there is less literature about the researched topic [84].
- (e) **Observations:** Observations is a systematic data collection technique that requires prolong involvement in a research topic, methodical improvisation to understand it, and critically analyzing it to derive meaningful results [84].
- (f) **Collecting texts and artifacts:** One of the most common ways to begin the data collection to study a culture or a social setting is to look out for different types of documents related to the research. These documents could be files, statistical records, meeting notes, emails, memos, public postings, wall posters, etc. that provide some amount of information useful for research [84].

The initial choice of data collection technique was to have a semi-structured interview that presents a live demonstration of the NAO robot handling the hand-signs recognition and further record a user's feedback via a questionnaire. But due to the outbreak of COVID-19 global pandemic leading to social distancing - meeting the participants and conducting questionnaires was not permissible. Hence, the entire data collection was managed online by showing a recorded video footage of the hand-signs recognition to the participants, followed by an online questionnaire form. The video recording presents how a user interacts with the NAO robot simulating the hand-signs recognition scenario. The online questionnaire, hereafter referred as the **User-Feedback Questionnaire** - contains a set of close-ended and open-ended questions to assess the research goal. Various platforms like Microsoft, Google, Hubsoft, and many others provide services with the necessary tools to conduct and monitor online surveys [85, 86]. This research uses **Google Forms** to conduct the questionnaire. After completing the form generation, these platforms provide a public URL through which one can invite people to participate in the survey. Their responses are stored on a cloud storage usually in an online spreadsheet file.

The responses from the online questionnaire are analyzed using descriptive statistics to compare different observations of the participants. **Descriptive statistics** helps to describe the basic features of the data giving a simpler summary for the sample population [87]. Later, distribution graphs present the results summarizing the frequency of individual values or ranges of values for a variable.

One of the significant things associated in collecting and analyzing the data is to understand what the researcher brings to the evaluation task - the bias,

interests, perceptions, knowledge, communication with people, all might influence the end-results of the research. Validity in qualitative research addresses this subjective nature of data collection and analysis. Since the researcher is the tool for data collection and analysis, researchers can approach the same research in different ways. Qualitative research always results in interpretations, rather than a purely objective goal. It is often valuable for other researchers to reproduce the tests, analyze the same data, and compare results giving a strong validity to every research. It is often valuable in qualitative research that one can repeat the tests, analyze the same data, and compare results giving a strong validity to every research. To ensure the validity and transferability, all components of the prototype developed in this research are publicly available at *Github*[88] working as a source for other researchers to continue this work. HSRC uses the state-of-the-art techniques of deep learning ensuring content validity. All the components of the prototype system work consistently with each other, guaranteeing construct validity. The research is based on an underlying assumption that non-verbal communication has the potential to improve Human-Robot Interaction. Though in-person questionnaires was an ideal choice for data collection, the research had to opt for different evaluation strategies due to the COVID-19 pandemic lock down [20] beginning from 13 March 2020 until 15 August 2020 (the last phase of this thesis project). Thus, any further research based on this thesis must take into account these factors of dependability.

# /5

## Prototype development of the Humanoid robot handling Hand-Signs

This chapter presents the system requirements, the design process for each of the components of the developed system. The final version of the prototype enabling a humanoid robot to understand hand signs mainly needs three components: the humanoid robot, a layer that integrates the humanoid and the HSRC (hereafter, referred to as Integration layer), and the Hand-Signs Recognition Component itself. Further sections describe different modules required in each of these components.

### 5.1 System Requirements

Clearly defined requirements are essential for every successful project journey [79]. For a software engineering project, requirements refer to the specifications that the project must accomplish. It is broadly of two types:

1. **Functional Requirements:** Functional requirements specify what the product/service do. It captures the intended behaviour of the system in the form of features or functions [80]. Functional requirements help

to check whether the product provides all the services it intends to. Below are the functional specifications for each of the components of the prototype developed in this work:

- The NAO robot recognizes only three hand-signs - **Palms-Open**, **Thumbs-Up**, and **OK**.
  - NAO robot must be able to access the Integration Layer.
  - User must initiate the conversation with the robot and start the configured scenario using '*Start Sign Recognition*' audio command.
  - NAO uses the following eye colour to represent its state:
    - **Red**: NAO has stopped the sign recognition scenario.
    - **Green eyes**: NAO robot is ready to do a job.
    - **Blinking Blue eyes**: NAO robot is processing the hand-sign.
    - **Blinking Orange eyes**: NAO robot is processing the specified action mapped to the detected hand-sign.
  - Integration Layer architecture must support the deep learning frameworks and necessary libraries required for the model to perform predictions.
  - Integration Layer must store the deep learning model to perform hand-signs recognition process.
  - The HSRC receives an input image of size 224 x 224 (width x height).
  - User must place the hand correctly in front of the top cameras of the NAO robot during the sign recognition.
2. **Non-Functional Requirements:** Non-functional requirements specify how the system should behave. It helps to ensure the usability and effectiveness of the system. Non-functional requirements guarantee reliability, availability, and ease of use [79, 80]. Below are the non-functional requirements of the entire prototype:
- All the data collected during the process (voice commands and input image for the NAO robot) should be kept private and not be available to other users or third-parties.



- NAO must guide the user and ease the use of prototype by giving important instructions.

## 5.2 Design process of the prototype system

Following the steps in a prototype development, each of these components along with their design specifications are described below:

- **Specify design requirements:** In broader terms, designing the prototype system included designing following three components:
  1. **NAO Humanoid robot:** NAO interacts with the user and records the hand-sign shown by an user via its cameras in image format (.jpg). NAO robot interprets only three hand-signs, as shown in the Figure 5.1. These three signs are as follows:
    - (a) **Palms-Open:** Hand sign achieved by showing an open palm
    - (b) **Thumbs-Up:** Hand sign achieved by closing the fist held with the thumb extended outwards.
    - (c) **OK:** Hand sign achieved connecting the thumb and forefinger in a circle and holding the other fingers straight.

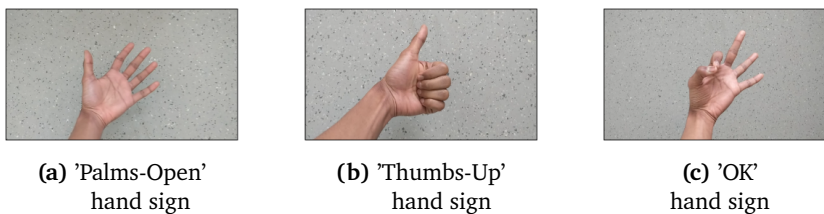


Figure 5.1: Hand-Signs recognized by the prototype

2. **Integration Layer:** Integration layer simulates a middleware primarily responsible for two things -
  - (a) Store the input image from NAO and apply any pre-processing operations as required for the deep learning model.
  - (b) On receiving the response from the model generated by the HSRC, apply post-processing operations (if required) to the results and finally, send it to the NAO robot.

3. **Hand-Signs Recognition Component (HSRC):** HSRC is the most influential component that identifies a hand sign from the fed image. Initial prototypes in this project solely focus on developing this component since the overall performance of the system would directly depend on how HSRC performs hand signs recognition. The HSRC generates a model that is used in the integration layer to predict the hand sign in the captured image using deep learning. Its performance depends on the available dataset. In this case, it refers to the number of images for each of the three hand-signs. More the number of classes, more is the data required. Hence, only three hand-signs were used to train the model that simplifies the design of HSRC by speeding data-processing steps. Although this does not change the procedure one uses to build a deep learning model.
- **Design Prototype:** Once the specifications for the system components are defined, the next step is to understand the frameworks or techniques to develop these individual components, as given below:
    1. **NAO Humanoid Robot:** NAO robot runs on the NAOqi programming framework [29] under OpenNAO distribution (based on Gentoo GNU/Linux distribution) [89]. It is formerly developed in C++ and Python providing various APIs for programming the NAO robot. Choregraphe [27] is another software toolkit that provides an interactive GUI to control multiple components of the robot without intricate programming, and hence Choregraphe is used to configure the NAO robot in this thesis.
    2. **Integration Layer:** Integration acts as a middleware between the NAO robot and the HSRC. The primary task of the integration layer is to receive the image captured by the NAO robot via APIs and run predictions over this image (test image) using the deep learning model. Additionally, it is responsible for pre-processing the received input and post-processing the generated results from the deep learning model. Pre-processing involves applying image compositions (resizing, gray scaling, and other methods) such that the input image is compatible with the deep learning model for predictions. The Post-processing might involve some data conversions (data type, data format and other parameters) required by the NAO robot for its next task execution.
    3. **Hand-Signs Recognition Component (HSRC):** Designing the HSRC deals with creating a new deep learning architecture or using existing architectures and modify them to solve the current problem via Transfer Learning. [90]. **Transfer learning** is the idea to reuse

the knowledge acquired in one domain and apply to other related domains [90]. For example, in image classification - if the goal of a deep learning model is to identify horses in an input image, but there aren't any publicly available models to do it. One can begin with an existing implementation that identifies other animals (say cats) and modify some parameters of the model to achieve similar results for horses [91].

After designing an architecture, one decides a software framework to implement the model. As of 2020, many deep learning frameworks exist - TensorFlow, Keras, Apache Spark, PyTorch, Apache MXNet, Caffe, and others [92, 93, 94, 95, 96, 97]. Choosing a software framework depends on the areas of application, support of low-level APIs, integration to other systems, user-friendliness, and other factors [98].

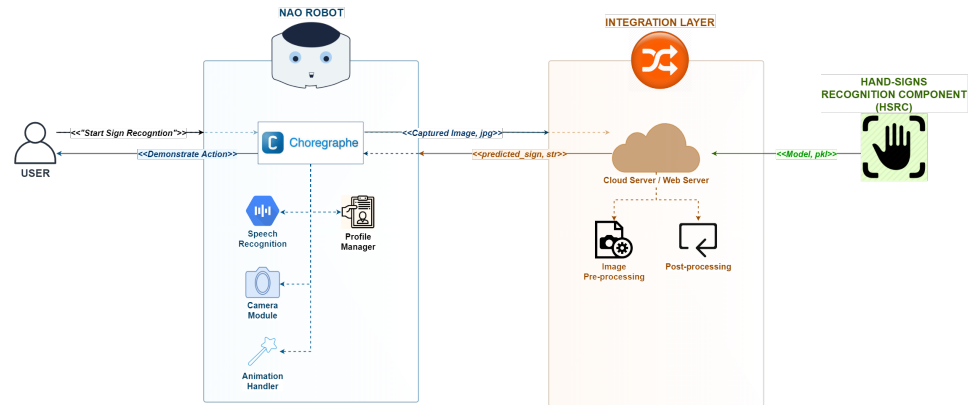
One has to make many decisions throughout the design phase like - deciding the data split, deciding frameworks, number of layers, number of neurons, values of the hyperparameters, arrangements to log the experiments, observing the results and modifying the model's architecture accordingly.

- **Build Prototype:** The building process is an incremental procedure. Initial builds of the prototype only focused on developing the deep learning models for the HSRC, and succeeding builds included its integration to the Integration Layer with NAO robot.
- **Evaluate Prototype:** In this research project, this phase evaluates the performance of the system two ways - the first is the performance of the HSRC part and second is conducting the questionnaire to understand people's opinion about the prototype's performance in establishing Human-Robot Interaction.
- **Refine Prototype:** This research does not involve refining the prototype system until the user is satisfied, but refers to functional development of the prototype system until the NAO robot, Integration Layer and HSRC respond correctly to the human subject for the presented hand-sign.

### 5.3 Development of the NAO humanoid robot

The humanoid robot acts as a front end layer that interacts with the user. One can use multiple modules and sensors available on the robot to establish the

interaction. The list below provides a few of the essential modules used in this prototype development as shown in the Figure 5.2:



**Figure 5.2:** System design of the prototype

1. **Speech Recognition** - Speech recognition is useful for speech-to-text and text-to-speech conversion. Former to process the audio commands given by the user and latter to generate an audio response to interact with the user.
2. **Camera Module** - Camera module is useful to capture the input image of the hand-sign and to implement other features like facial recognition, emotion recognition, and others. The camera module in this prototype captures the image of X x Y dimensions in (.jpeg) format [99], initially to recognize the hand-sign and later to simulate facial emotion recognition.
3. **Animation Handler** - Animation Handler animates the robot by controlling different parts of the robot such as - eye colour, voice tone, voice pitch and other parameters. It helps to add subtle nuances that improve the robot's expressiveness in the interaction.
4. **Profile Manager** - Profile Manager creates a profile of a user and stores different hand-signs along with their corresponding action/features to be performed by the humanoid robot. For example - User A wants to use 'Thumbs Up' sign to 'Play a song', but User B would use the same sign to 'Know Weather Information'. For the developed prototype, it only works for a single user, but one can easily customize these actions/features as required.

## 5.4 Development of the Integration Layer

The integration layer can be a complex middleware setup on a cloud container or a simple piece of code that is responsible for communication between the humanoid robot and the HSRC. It is mainly responsible to receive the captured image from the NAO robot, pre-process the image to be compatible for the model, load and run the predictions using the trained model, perform post-processing operations over the predicted results, and finally send it to the NAO robot. The modules commonly required in the Integration layer shown in the Figure 5.2 are described below:

1. **Pre-processing** - The preprocessing module is responsible for transforming the captured image, such that it can be acceptable by the trained model formed by the HSRC. Commonly used image transformation operations are resizing, gray scaling, removing the noise, and segmentation [100].
2. **Post-processing** - Often the predicted results have a numeric format. The postprocessing module converts the results into the desired format as required for the NAO robot. It may involve decoding, transforming into a string format or other operations as per the specifications. If the predicted results could be directly used by the NAO robot, one can skip this post-processing module.

## 5.5 Development of Hand Signs Recognition Component (HSRC)

Developing the Hand Signs Recognition Component (HSRC) essentially means to generate deep learning models using different deep learning architectures. These generated models are used by the Integration Layer to recognize the hand-sign shown by the user to the NAO robot. The development process follows the steps similar to any project in deep learning as described below:

1. **Problem Definition:** Every deep learning project starts by analyzing a problem and defining the objectives by setting specific goals to solve it. In this research, the goal is to develop a deep learning model that accepts an input image (of a person showing hand sign to the robot) and output the class label (1 out of 3 hand signs). Before designing the prototype model, an additional step in deep learning is data collection. Since deep learning models feed on data, HSRC requires a dataset that represents each of these three hand signs. The choice of three hand-signs in the prototype sufficiently demonstrates the scenario of using hand-signs to

establish human-robot interaction in humanoid robots. If one has an adequate amount of data for every hand-sign, it is easy to include more signs in the model. The dataset generated for this project has 2975 images of each sign, having 8925 images in the complete dataset.

2. **Designing the model:** The actual process of prototyping a deep learning model starts after data preprocessing. Designing involves creating a plan of action for the deep learning project depending upon the available data and goals to achieve within the project timeline. It deals with creating a new model architecture or using existing architectures and modify them to solve the current problem via Transfer Learning [90].

Transfer learning is the idea to reuse the knowledge acquired in one domain and apply to other related domains [90]. For example, in image classification - if the goal of a deep learning model is to identify horses in an input image, but there aren't any publicly available models to do it. One can begin with an existing implementation that identifies other animals (say cats) and modify some parameters to achieve similar results for horses [91].

The HSRC generates two models - one that contains various groups of convolution-pooling layers, and another that uses the transfer learning technique. The former gives greater control to design the model specifically for the use-case, but the latter provides better generalizations and re-usability. Implementation details of both these models are presented in next Chapter 6.

After designing an architecture, one has to decide the software framework to implement the model. As of 2020, many frameworks in deep learning exist - TensorFlow, Keras, Apache Spark, PyTorch, Apache MXNet, Caffe, and many more [92, 93, 94, 96, 95, 97]. Choosing a software framework depends on the areas of application, support of low-level APIs, integration to other systems, user-friendliness, and other factors [98].

The HSRC uses Keras framework [93] running over TensorFlow 2.0 backend [92]. Keras provides simple APIs for various deep learning processes like data augmentation, saving the best model in an epoch obtained during the entire training, allowing to interrupt the model training in case of poor performance and many others. Determining the optimal CNN architecture for HSRC is an incremental procedure. The initial design consisted of a shallow model with only a couple of layers yielding the 'Baseline Performance' for this problem. Further iterations of the model include minor changes to beat the Baseline Performance, like - adding an extra convolutional layer, changing the filter sizes, or

tuning one of the hyperparameters.

One has to make many decisions throughout the design phase like - deciding the data split, deciding frameworks, number of layers, number of neurons, values of the hyperparameters, arrangements to log the experiments, observing the results and modifying the architecture accordingly. All of these are discussed in the next Chapter 6.

3. **Evaluating the model:** Evaluation includes training the model and assessing its performance using various metrics like Accuracy, Precision, Recall, F1 score or AUC score [82]. The mode's accuracy gives the ratio of the correct hand -sign predictions to the total number of predictions. It works well for a balanced dataset such as the final dataset in this research having 2975 image samples for each sign.
4. **Deployment and Maintenance:** Since this research does not involve any production deployment, this phase only includes integrating the HSRC to other parts of the system.

## 5.6 Establishing the Human-Robot Interaction

Once the NAO robot, the integration layer, and Hand-Sign Recognition Component are developed, the next step is to make these components work together coherently to establish a human-robot interaction. A task-based scenario simulates it - where a user starts hand-signs recognition using a voice command to the NAO robot, and NAO performs an action based on the predicted hand-sign. The following section explains the design choices made for the different components to simulate the scenario represented in the Figure 5.3.

### Interacting with the Robot

It is a functional requirement that the user must initiate the scenario by interacting with the prototype by saying specific words - like 'Hi', 'Hello', 'Start', or others to the NAO robot. NAO robot always listens to the environment via its microphones and detects those words from its input audio using keyword spotting [101]. On successful detection, it greets the user back by saying 'Hi, I am ready' or says 'Sorry, I could not hear you clearly, could you please repeat?' on a failure. Another possible way to initiate the scenario is to use facial recognition in the NAO robot and actively begin the interaction with the user. Although NAO robot comes with pre-installed facial recognition program, it performs poorly and hence, was not skipped in the final version of the prototype.

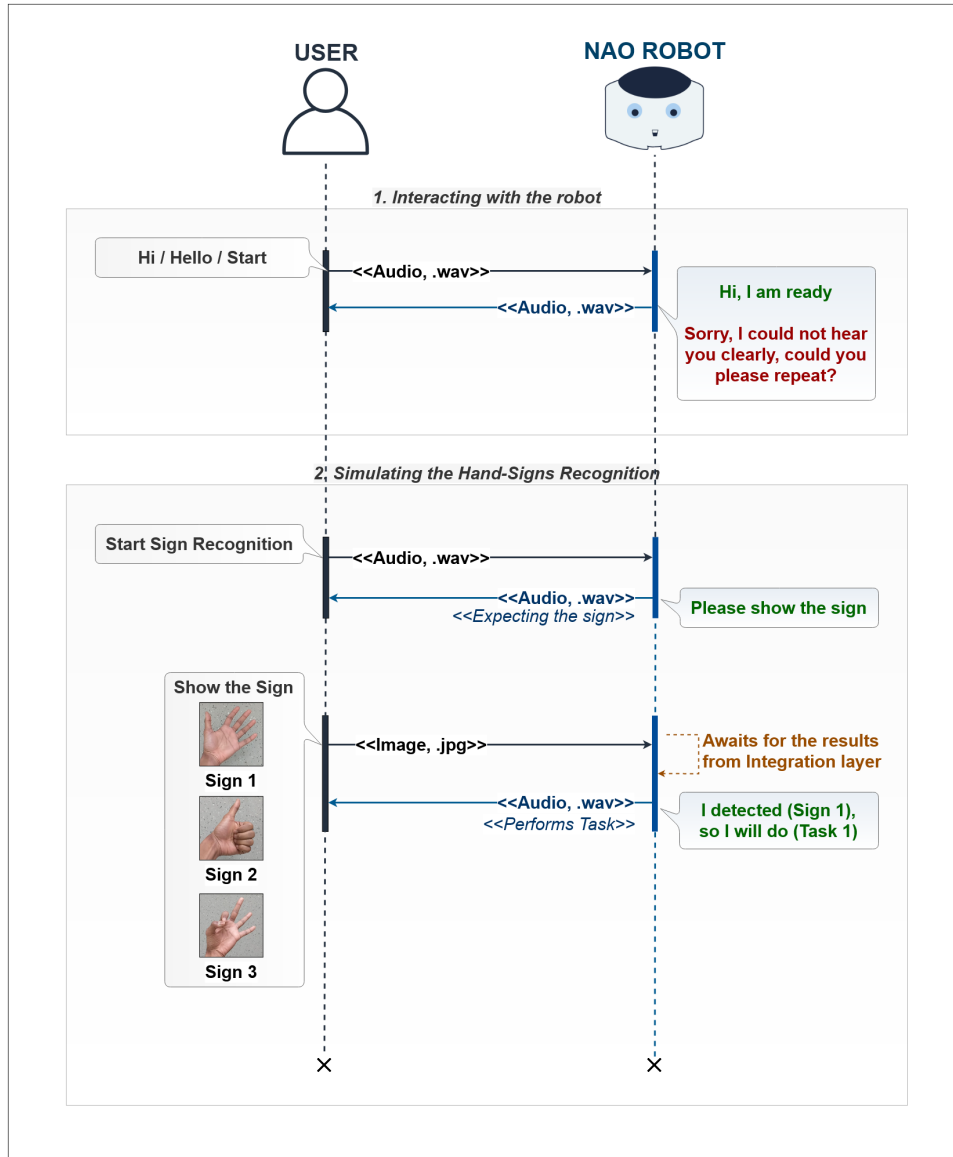


Figure 5.3: Sequence diagram of establishing the human-robot interaction in prototype



## Simulating the Hand-Signs Recognition

User can simulate the hand-signs recognition using the *'Start Sign Recognition'* audio command. NAO robot acknowledges the user and directs the user saying *'Please show the hand-sign'*. The user is supposed to position his hand in front of the NAO robot at this stage. The NAO robot records the image via its camera (.jpg file), saves the image in its local storage, and sends it to the Integration Layer using the available APIs. It waits for the results from the Integration layer. At this stage, the Integration layer stores the trained model (.pkl file) generated by the Hand Signs Recognition Component. It preprocesses the received image from the NAO and runs predictions over it. After receiving the results from the integration layer, NAO robot uses the profile manager to retrieve the task mapped to the hand-sign, and responds to the user saying - *'I detected 'Thumbs-Up', so I will play some music'*. After completing the task, NAO asks the user if he wants to stop the hand-signs recognition scenario. If no, it repeats the scenario asking the user to show the next hand-sign, else it stops the scenario saying *'Goodbye'*. Figures represent a user interacting with the NAO robot for the above steps.

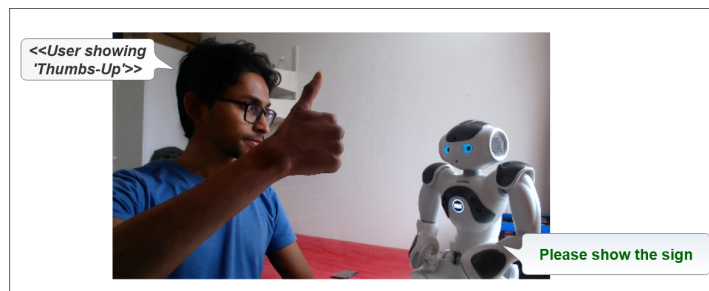


Figure 5.4: User showing the hand-sign to the NAO robot

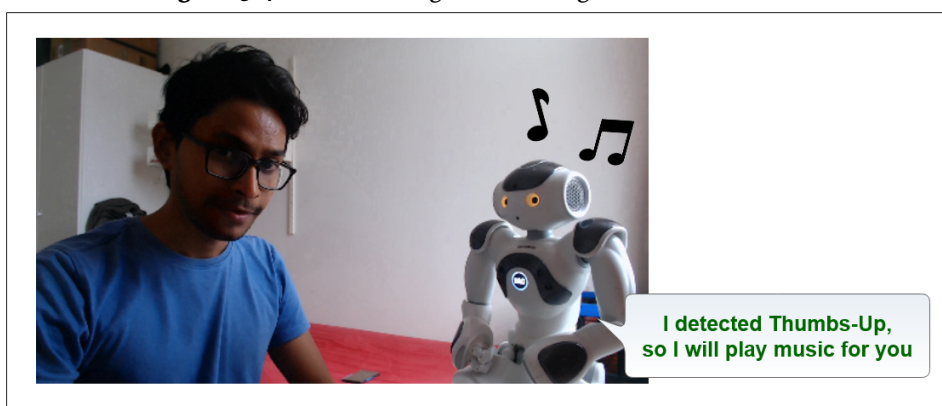


Figure 5.5: NAO robot performing a task on detecting the hand-sign



# /6

## Implementation

This chapter describes all the steps taken to form the prototype system explained in the previous Chapter 5. The prototype system comprises of three major components - the NAO robot, the Integration Layer, and the Hand Signs Recognition Component (HSRC). Since, HSRC is the most critical part of the prototype, initial build/versions of the prototype focused on developing only the HSRC. Other two components were incorporated to the subsequent builds forming the final version. These components are described independently in different sections in this chapter.

### 6.1 The NAO robot

The NAO robot is the only component that the user interacts with the prototype system. NAO acts as a front end interface of the prototype system for the user. This research employs a task-based scenario for the user where the NAO robot carries the following activities to simulate the hand signs recognition process:

1. Communicate with the user using its speech recognition modules and begin the hand signs recognition process after detecting the 'Start Sign Recognition' command.
2. Ask the user to prepare themselves for the hand signs recognition process.

3. Capture the image using its camera module, and send it to the Integration layer for predictions.
4. With the help of Profile Manager, retrieve and perform the actions for the detected hand-sign, as configured by the user.

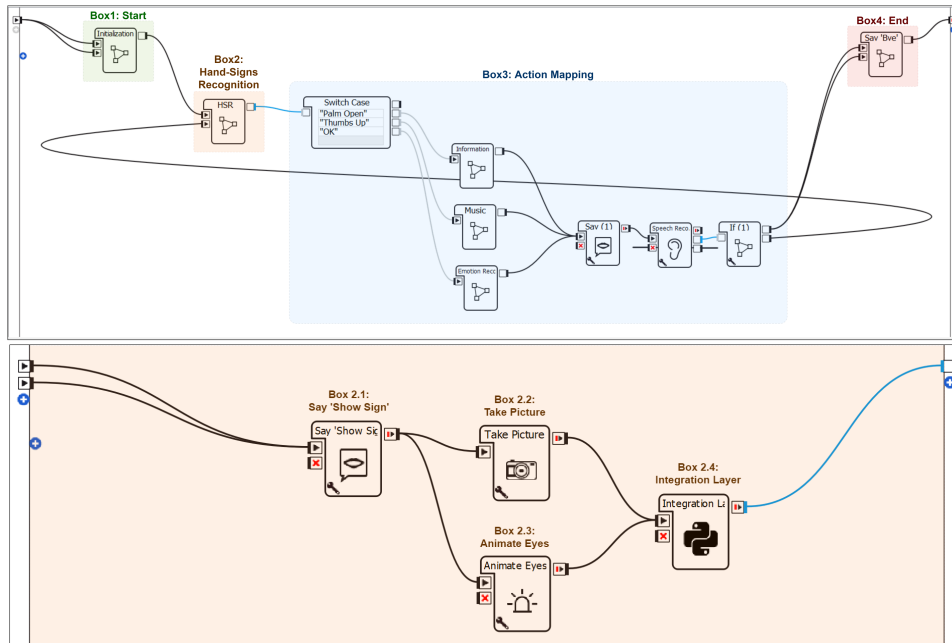
The NAO robot runs on the NAOqi software framework. NAOqi [29] is a cross-platform programming framework developed in C++ and Python. It allows homogeneous use of different hardware modules (sensors, robot parts) with software processes (information sharing, parallelism, synchronization, event handling). Programming and debugging for the above activities would be a tedious process. A better alternative is to use the **Choregraphe** programming tool, as recommended by the robot manufacturers [27]. Choregraphe is a multi-platform desktop application that allows creating complex scenarios (interact with users, customized response animations, speech recognition, and many more) without writing low-level programming code. The next section provides the implementation details on how NAO is programmed using the Choregraphe for this prototype.

### 6.1.1 Programming NAO using Choregraphe

Choregraphe is a graphical language for the NAOqi framework that allows using all the NAOqi APIs with additional visual tools that ease the programming process of the NAO robot. It employs scenario-based implementation via **Projects**. **Projects** are a collection of files and **Behaviors** to simulate a specific scenario, a use-case or an experiment with the NAO robot. **Behaviors** are a set of instructions sent or installed in the NAO robot to make it interact with its environment, process the input, and generate suitable results. **Behaviors** consist of various **Boxes** that simulate primary actions (like walking, saying a word) and complex implementations (like exploring a room, recovering from a fall).

The Figure 6.1 illustrates the hand signs recognition scenario implemented in NAO Robot using Choregraphe for this research prototype. **Box 1: Start** initializes the NAO robot with all the necessary libraries, including a speech-recognition module to understand and recognize predefined words or phrases to begin the hand-signs recognition scenario. The user must initiate this task-based scenario using 'Start Sign Recognition' voice command. NAO uses keyword spotting in its speech recognition module - a technique where it continuously searches for specific keywords ('Start Sign Recognition' in this case) in its input data.

On a successful search, NAO begins the hand-signs recognition process repre-

**Figure 6.1:** Choregraphe Behavior: 'Hand Signs Recognition'

mented by **Box 2: Hand-Signs Recognition**. The second diagram in Figure 6.1 reveals different modules used in **Box 2: Hand-Signs Recognition**:

- **Box 2.1: Say 'Show Sign'** - NAO asks the user to get ready by saying *'Please show the Sign'* and expects the user to show the hand sign positioned correctly in front of its eyes.
- **Box 2.2: Take Picture and Box 2.3: Animate Eyes**- NAO captures the image using its top camera and acknowledges the user by saying *'Sign Captured Successfully'* and animates its eyes (colour of both the Eye-LEDs are changed green) using the animation handler. NAO saves the captured image having a 1280 x 960 resolution in its internal storage at `"/home/nao/recordings/cameras/"` as `"image.jpg"`.
- **Box 2.4: Integration Layer** - The saved image is passed to the Integration layer using a custom python script module. It uses `'requests'` library to create a POST request containing the `"image.jpg"`. NAO awaits the response from the Integration layer, and the subsequent tasks are carried by **Box3: Action Mapping**.

On receiving the response from the Integration layer, **Box 3: Action Mapping** executes different actions/features of the robot to the corresponding hand-sign. This mapping of a hand-sign to a particular action/feature is predefined in this

scenario as follows:

- **Palms-Open:** NAO provides some information about the present day (date, time, weather, etc.)
- **Thumbs-Up:** NAO plays some music to the user using 'Play Music' module available in Choregraphe.
- **OK:** NAO recognizes the user's emotions from facial expressions employing Choregraphe's Emotion Recognition module.

After executing the mapped action/feature, NAO asks if the user wishes to end the scenario or perform the hand-signs recognition once more. If yes, it repeats the execution from **Box 2: Hand-Signs Recognition** else terminates the hand-signs recognition scenario saying 'Good Bye' in **Box 4: End**.

## 6.2 Integration Layer

The Integration Layer fundamentally acts as a back-end system of the developed prototype. The integration layer is primarily responsible for following tasks in the prototype:

1. Receive an input image from the NAO robot
2. Load the trained deep learning model obtained from the Hand-Signs Recognition Component (HSRC).
3. Preprocess the received input image as required for the trained model and perform the predictions
4. Send the predicted hand sign to the NAO robot

Integration layer can be a complex cloud architecture that is more suitable for production environments demanding high scalability, throughput and error handling. But it can also be implemented more simply, with a web server as used in this research prototype. The integration layer fundamentally creates an API for deep learning models. It must support the necessary deep learning frameworks on which the model got trained. Here, it requires TensorFlow 2.0, Keras, Flask and Pillow libraries:

1. **TensorFlow 2.0 and Keras:** both are the deep learning frameworks used in generating the model by the HSRC.

2. **Flask**: a python web framework to create API endpoints and serve requests from NAO robot.
3. **Pillow**: a python image processing library for preprocessing the image before predictions.

The integration layer is setup on a machine present in the same network as the NAO robot. Assuming the above dependencies are satisfied, a single file installs, and deploys the integration layer for use. Three functions of the integration layer are:

1. **main()**: On startup, the `main()` loads one of the trained models into the memory using the `load_model()` from Keras libraries.
2. **predict()**: It does the following things:
  - accepts any incoming requests
  - checks for a POST request and retrieves an image from the `'files'` attribute of the POST request
  - reads the image and preprocesses the image by calling `'prepare_image()'`
  - makes predictions using `predict_classes()` from Keras library and stores the results in a dictionary
3. **preprocess\_image(image, target\_size)**: `preprocess_image()` is responsible for all the preprocessing operations discussed previously. Here, it converts the original image to RGB and resizes it to the spatial dimensions (224 x 224 in this case) that is compatible for the trained model.

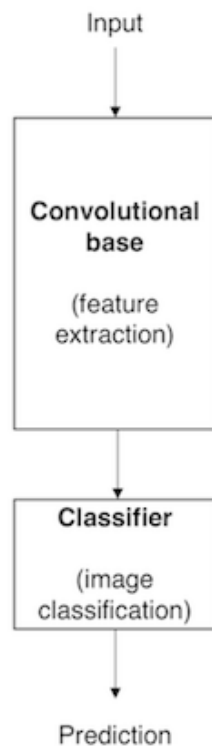
## 6.3 Hand Signs Recognition Component (HSRC)

Hand Signs Recognition Component (HSRC) is the most significant part of the prototype system developed in this research. It produces different models using state-of-the-art Convolutional Neural Networks from Deep learning. These models are stored and used by the Integration layer to identify the hand sign in an input image received from the NAO robot. Below are the most crucial steps used in developing the final model:

### 6.3.1 Data Collection/Exploration

As discussed earlier, deep learning models need a large amount of good quality training data for higher performance. The quality of training data has profound implications for the subsequent development of the model. It mainly involves gathering the data and preparing it for the model training process [102].

A custom dataset containing a total of 8925 images for 3 static hand signs (2975 images for each sign) is prepared to train the deep learning model. These images are nothing but still frames extracted from video footage recorded for each hand sign, enacted by a person using a smartphone camera with a 720p resolution in 30 frames per second. Images are derived every 0.15 seconds of the entire video. The resulting images have dimensions of width 1280 x height 720 pixels. A python script takes an input video file, extracts the image frames every  $x$  seconds ( $x = 0.15$  in this case) and stores these images in a separate directory.



**Figure 6.2:** Parts of training a CNN model [103]



### 6.3.2 Designing the model

The objective of the HSRC is to identify a hand sign in the input image. From the deep learning perspective, it is a multi-class image classification problem wherein it assumes that the input image belongs to one of the 3 available classes (each class representing a different hand sign). There are three models developed, each having a different architectural style. Since deep learning architectures ultimately operate like black boxes [104], it is best to design several architectures for the given problem.

Initially, a **Baseline Model** is developed having very few layers, as shown in the Figure 6.2 and its performance, hereafter, referred to as the '**Baseline Performance Score (BPS)**', guides the architectural design of subsequent models. The Baseline model, as the term suggests, gives a baseline performance of the model having the most simplistic CNN architecture. It serves as a reference to building subsequent models. **Accuracy** is used to estimate the performance of the HSRC. It is the ratio of correct predictions to the total predictions made by the model. The goal for the succeeding models is simply to beat the accuracy score of its preceding model.

#### Baseline Model

The Table 6.1 illustrates the CNN architecture to measure and set up the Baseline Performance Score (BPS) for the HSRC. A CNN architecture typically has two parts:

1. **Convolutional Base:** Convolutional base is a stack of convolutional and pooling layers. The way to stack these layers differ. Some CNNs follow Convolution-Pooling-Convolution-Pooling style or Convolution-Convolution-Pooling-Convolution-Convolution-Pooling style. The CNN used to measure the BPS has two blocks of Convolution-Pooling layers. The first block has 32 filters of size  $3 \times 3$ , followed by a MaxPooling layer of size  $2 \times 2$ . Each layer uses ReLU activation function, which is generally a best practice. The second block has the same layers, but with 64 filters. Filters in simple terms mean the number of neural units in a particular layer of the CNN architecture. The goal of the convolution base is to generate feature detectors [103]. Feature detectors extract valuable information from the image that helps in the classification.
2. **Classifier:** As the term suggests, the main goal of the classifier is to classify the image based on detected features from Convolutional Base. It is usually composed of full-connected layers. The feature detectors from the convolutional base are flattened and passed to the fully-connected layer

for predictions. The CNN used to measure BPS has one fully-connected layer with 128 units, followed by an output layer with 3 units (each unit representing one out of 3 hand signs) using Softmax activation.

**Table 6.1:** Training Parameters and Accuracy of Baseline Model

<b>Parameters</b>	<b>CNN Model</b>	<b>Baseline</b>
<b>Convolution Base</b>		Conv2D(16, 3x3) Maxpool2D(2x2)
<b>Classifier</b>		Dense(16) Dense(3)
<b>Epochs</b>		20
<b>Training Accuracy (%)</b>		63.97
<b>Validation Accuracy (%)</b>		59.80

### **CNN 1**

CNN 1 uses the traditional format of Convolution-Pooling style of architecture as shown in the figure. The convolutional base has four groups - each with a convolutional layer (3x3 filters) followed immediately with a pooling layer (2x2 filters). The classifier has a single fully connected layer with 128 units, followed by an output layer with 3 classes using Softmax activation.

### **CNN 2**

CNN 2 uses online pre-trained models on the generated training dataset using transfer learning. The purpose here is to leverage the previous learnings done on a large dataset, expecting the hierarchical feature representations learned by a model will help solve the problem at hand. Here, the pretrained model is VGG-19 as a convolutional base, with a custom classifier similar to the previous one, as shown in the figure. VGG-19 uses most straightforward approach to improve the performance of the model - increasing the size of model by adding more layers [105]. CNN 2 uses the convolutional base part of the VGG-19 and adds a custom classifier having a fully connected layer with 128 units, followed by a dropout layer and an output layer with 3 classes using Softmax activation.

### **6.3.3 Evaluating and Refining the model**

The accuracy of the Baseline model gives training accuracy of 63.97% and validation accuracy of 59.80%. Subsequent models were developed by adding a group of Convolution-Pooling layer or applying regularization techniques such as Dropout or L2 regularization as explained in Chapter 3. Next Chapter 7 discusses the performances of each of the intermediate builds of CNN 1 and CNN 2 in the HSRC, and the entire prototype system.





## Evaluation and Results

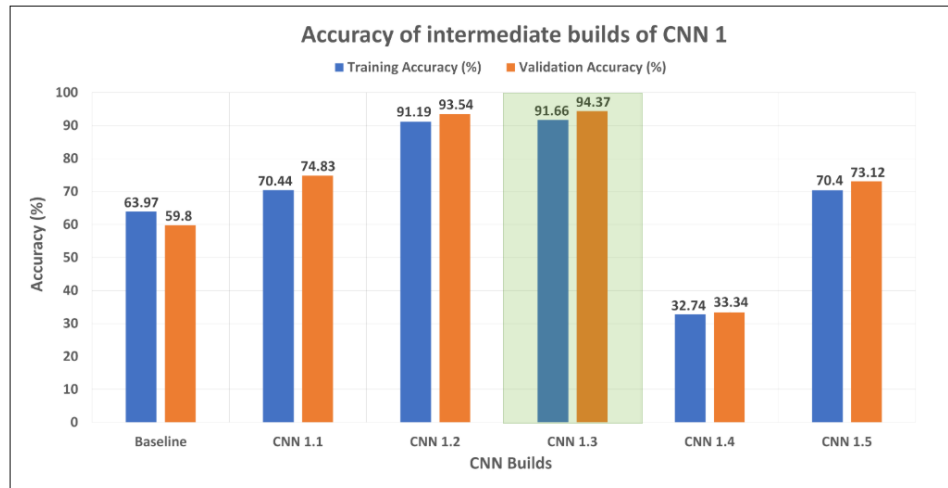
The most significant element of the prototype is the Hand-Signs Recognition component, and therefore, its performance is of primary importance before putting it to real-world usage. Below are the significant factors focused during the prototype evaluation:

- **Accuracy:** Accuracy is the ratio of correct predictions to incorrect predictions made by the robot in the hand-signs recognition task. It is a quantitative measurement expressed in percentage.
- **Responsiveness:** Responsiveness is a qualitative measure of the time taken by the robot in completing the hand-signs recognition task. In this thesis, it acts to generalize the user's opinion on how fast was the robot in the hand-signs recognition task.
- **User Acceptance:** Understanding the user's perception of using non-verbal communication as a means to improve HRI in humanoid robot.

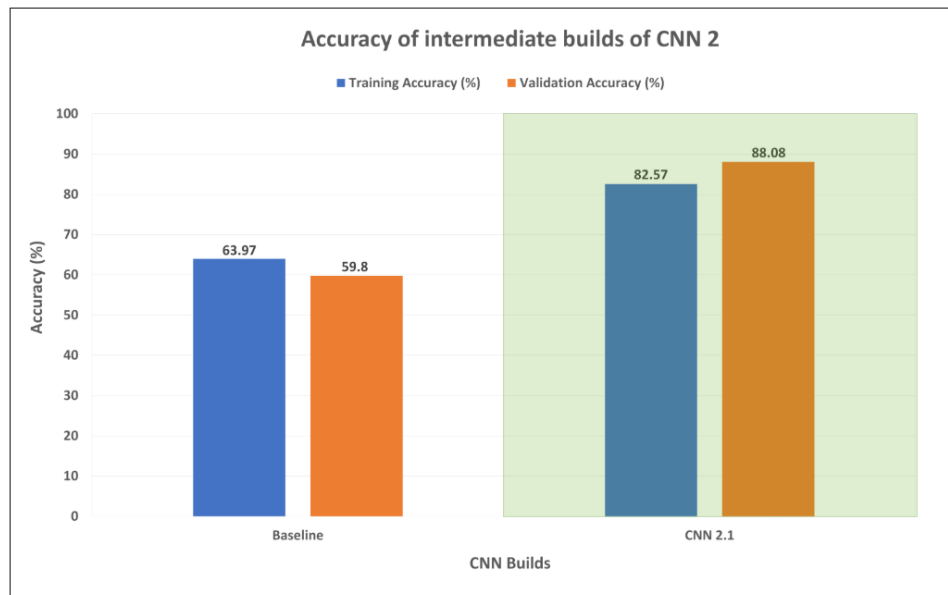
### 7.1 Accuracy of the prototype

Accuracy of the prototype is indeed the accuracy of the models generated by the HSRC. The Figure 7.1 and Figure 7.2 shows the training and validation accuracy obtained for the intermediate builds of both the models - CNN 1 and

CNN 2 respectively. The x in CNN 1.x and CNN 2.x express these intermediate builds. Every subsequent build aims to be more accurate than its previous one. Most common modifications made to a model to improve its accuracy are - adding a group of Convolution-Pooling layer with more filters, changing the filter size, or adding a dropout layer to regularize the results.



**Figure 7.1:** Accuracy of intermediate builds of CNN 1



**Figure 7.2:** Accuracy of intermediate builds of CNN 2

Table 7.1 presents the accuracy and losses of each of the builds for CNN 1. The Table 7.1 shows the detailed architecture of every intermediate build of CNN 1. The highlighted layers are the changes made in a specific build. For

example, consider CNN 1.1 - it adds a two-dimensional Convolution-Pooling layer (Conv2D(32 units, 3x3 filter size) and MaxPooling(2x2 filter size)) to the Baseline model. CNN 1.2 contains another such layer (Conv2D(64, 3x3) and MaxPooling(2x2)) and likewise. The Figure 7.1 presents the performance of each of the builds of CNN 1. It shows that adding more layers improved the accuracy of the HSRC until CNN 1.3, where it achieves maximum training accuracy of 91.66% and validation accuracy of 94.37%. These numbers are assuring for such use-case but may indicate overfitting (a scenario where the model scores high accuracy on the training set but loses generalization over unknown data). To examine it, one must observe the accuracy and loss values for every epoch in the training phase. The graph Figure 7.3 presents the accuracy and loss of the two best builds - CNN 1.2 and CNN 1.2. Both these graphs do not show characteristics of overfitting (where the validation loss is higher than the training loss) as explained in Figure 3.12 in Chapter 3. But here, the validation loss is lower than training loss suggesting limitations with the dataset, especially that the validation images are very similar to the training images.

**Table 7.1: Training Parameters and Accuracy of CNN 1**

CNN Build Parameters	Baseline	CNN 1.1	CNN 1.2	CNN 1.3	CNN 1.4	CNN 1.5
<b>Convolution Base</b>	Conv2D(16, 3x3) Maxpool2D(2x2)	Conv2D(16, 3x3) Maxpool2D(2x2) Conv2D(32, 3x3) Maxpool2D(2x2)	Conv2D(16, 3x3) Maxpool2D(2x2) Conv2D(32, 3x3) Maxpool2D(2x2) Conv2D(64, 3x3) Maxpool2D(2x2)	Conv2D(16, 3x3) Maxpool2D(2x2) Conv2D(32, 3x3) Maxpool2D(2x2) Conv2D(64, 3x3) Maxpool2D(2x2) Conv2D(128, 3x3) Maxpool2D(2x2)	Conv2D(16, 3x3) Maxpool2D(2x2) Dropout(0.25) Conv2D(32, 3x3) Maxpool2D(2x2) Dropout(0.25) Conv2D(64, 3x3) Maxpool2D(2x2) Dropout(0.25) Conv2D(128, 3x3) Maxpool2D(2x2) Dropout(0.25)	Conv2D(16, 3x3) Maxpool2D(2x2) Dropout(0.25) Conv2D(32, 3x3) Maxpool2D(2x2) Dropout(0.25) Conv2D(64, 3x3) Maxpool2D(2x2) Dropout(0.25) Conv2D(128, 3x3) Maxpool2D(2x2) Dropout(0.25)
<b>Classifier</b>	Dense(16) Dense(3)	Dense(32) Dense(3)	Dense(64) Dense(3)	Dense(128) Dense(3)	Dense(128) Dropout(0.25) Dense(3)	Dense(64) Dropout(0.25) Dense(3)
<b>Epochs</b>	20	20	20	20	20	20
<b>Training Accuracy (%)</b>	63.97	70.44	91.19	91.66	32.74	70.40
<b>Validation Accuracy (%)</b>	59.80	74.83	93.54	94.37	33.34	73.12

**Table 7.2: Training Parameters and Accuracy of CNN 2**

CNN Build Parameters	Baseline	CNN 2.1
<b>Convolution Base</b>	Conv2D(16, 3x3) Maxpool2D(2x2)	<b>ConvBase(VGG-19)</b> <b>Dropout(0.25)</b>
<b>Classifier</b>	Dense(16) Dense(3)	<b>Dense(128)</b> <b>Dropuout(0.25)</b> <b>Dense(3)</b>
<b>Epochs</b>	20	28
<b>Training Accuracy (%)</b>	63.97	82.57
<b>Validation Accuracy (%)</b>	59.80	88.08

CNN 2 uses the transfer learning technique over VGG-19 model. The detailed

architecture for CNN 2 is given in the Table 7.2. CNN 2.1 retains the convolutional base of VGG-19 and a custom classifier giving a training accuracy of 91.86% and validation accuracy of 98.61% as shown in the Figure 7.2. The accuracy and loss graphs of CNN 2.1 in Figure 7.4 are similar to Figure 7.3 implying no overfitting but interestingly, reflects that the size of the training dataset is small and too simple for a complex model like VGG-19.

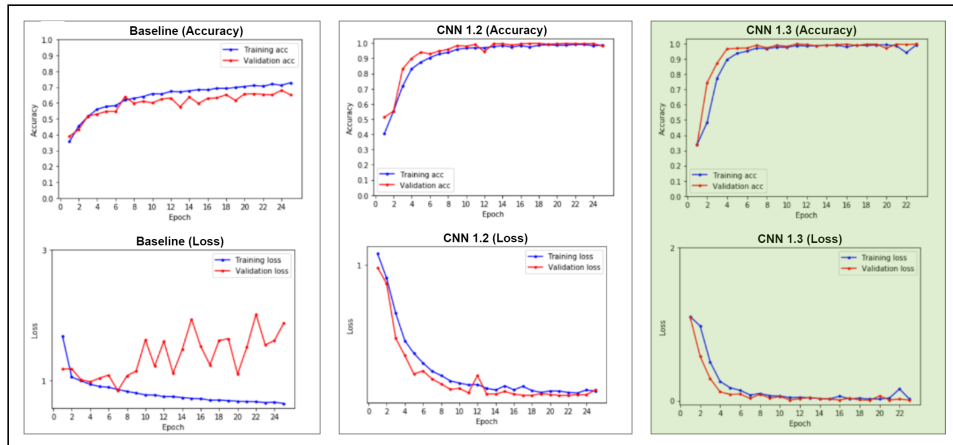


Figure 7.3: Accuracy and Loss of the best builds of CNN 1

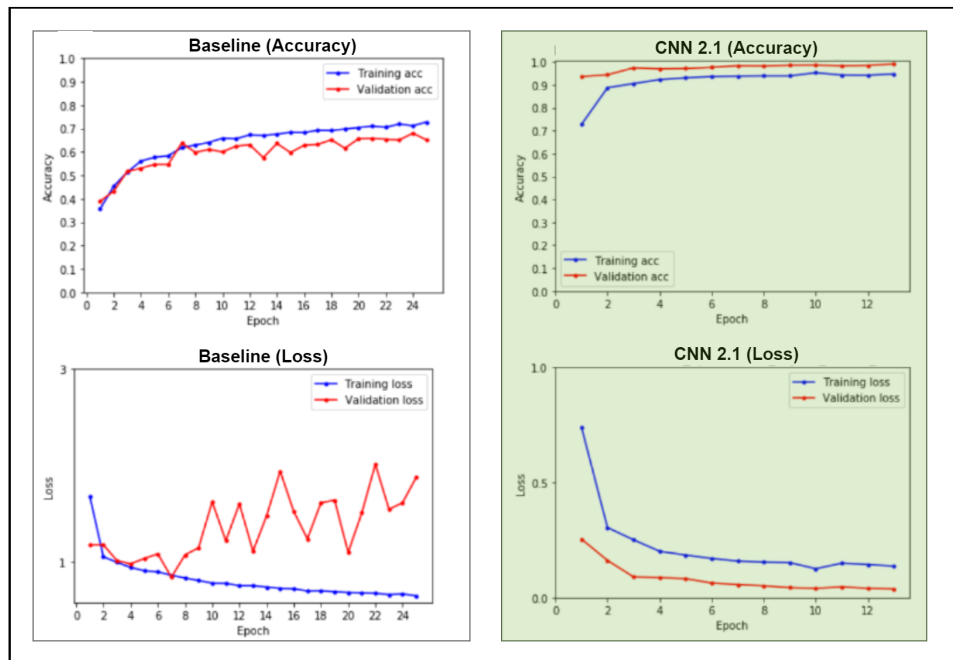


Figure 7.4: Accuracy of intermediate builds of CNN 2

Generally, the final model is also tested once with the test dataset (dataset unseen or unknown to the model) to assess its performance in real-world.



Test datasets are extremely useful if one performs hyperparameter tuning or model optimizations, as described in Chapter 3. The prototype does not use any optimization techniques, and thus preparation of test dataset is skipped from the evaluation.

## 7.2 Responsiveness and User Acceptance of the prototype

Other two factors - Responsiveness and User acceptance are measured using an online questionnaire showing a video demonstration of the NAO robot performing the hand-signs recognition task [106]. The questionnaire consists of the following questions:

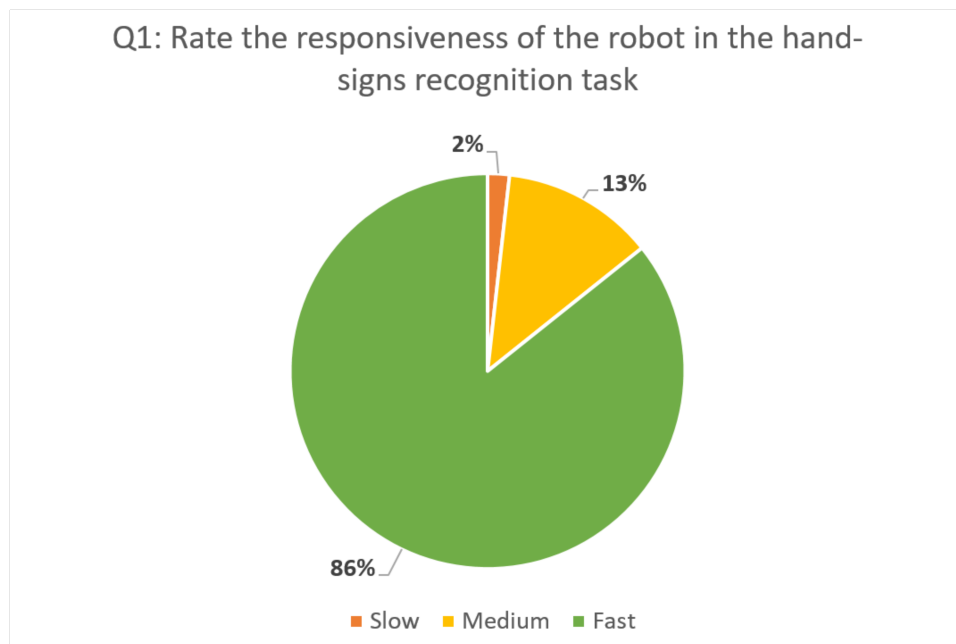
1. Rate the responsiveness of the robot in the hand-signs recognition task:
  - Slow
  - Moderate
  - Fast
2. Would you like to use non-verbal communication (hand-signs, facial expression, eye contact, haptics, etc.) to access different features of the robot?
  - Yes
  - No
3. If you had a humanoid robot, for what will you use hand-sign recognition feature?
  - Access a robot service (like reading news, playing music, etc.)
  - Personalise/Customize the robot service (a user maps his desired sign to a robot service)
  - Sign-Language Translation/Communication
  - Not use the hand-sign recognition feature
  - Other Applications
4. What are your thoughts on using non-verbal communication for a humanoid robot?

The questionnaire received replies from 56 people belonging to different age groups, gender, and work domains. The results for each question is analyzed

and presented below:

1. **Question:** Rate the responsiveness of the robot in the hand-signs recognition task

**Result:** Figure 7.5 illustrates how users thought about the responsiveness of the robot in the hand-signs recognition. 86% claimed it to be fast, 13% believed to be moderate, and 2% felt it to be slow.



**Figure 7.5:** Responsiveness of the robot analyzed from questionnaire

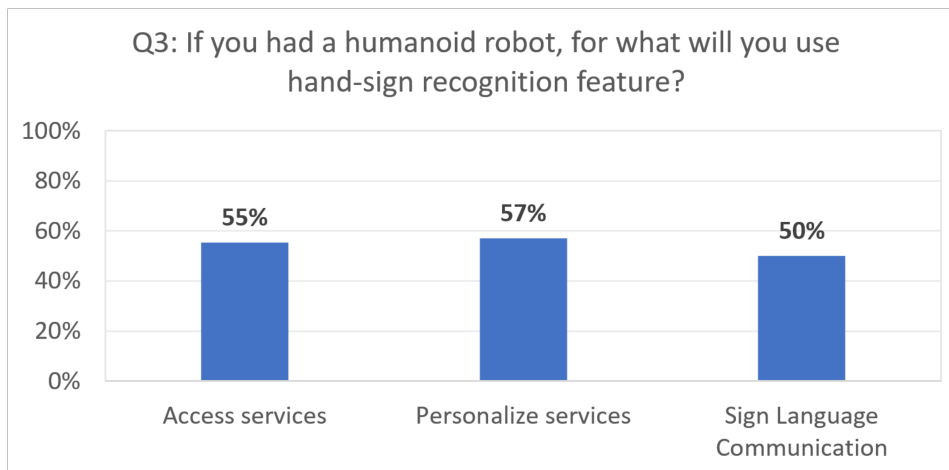
2. **Question:** Would you like to use non-verbal communication (hand-signs, facial expression, eye contact, haptics, etc.) to access different features of the robot?

**Result:** All the participants said 'Yes' for using non-verbal communication to access different features of the robot.

3. **Question:** If you had a humanoid robot, for what will you use hand-sign recognition feature?

**Result:** The table illustrated in the Figure 7.6 describes how users perceived the use of non-verbal communication to improve Human-Robot Interaction and its possible areas of application. 55% of people would like to use non-verbal communication to access a robot's service and for

sign-language translation, while 57% would also personalize the robot's services, and 50% preferred it to interpret sign language. Other than the given application areas, some suggested using non-verbal communication to control smart-home devices for people with speech impairments, assigning delegatory work to a robot when one is busy and other applications.



**Figure 7.6:** Possible application areas for using non-verbal communication

4. **Question:** What are your thoughts on using non-verbal communication for a humanoid robot?

**Result:** Some of the replies received are:

- "I recommend to shorten the processing dialogue by robot"
- "It would be a faster method and would not be restricted to language spoken region"
- "In the times of corona (COVID-19 pandemic), this could be a great contribution to non physical handling of basic services we use through the day, for example TV remote controls etc."

### 7.2.1 Summary of the results

Accuracy, Responsiveness, and User Acceptance are three parameters used to evaluate both the prototype and the research in this thesis. Accuracy of the prototype is determined by the accuracy of the Hand-Sign Recognition Component. The HSRC of the prototype produced several deep learning models to carry out the sign recognition of three different signs. CNN 1.3 and CNN 1.2 are two of the best prototype builds, both having accuracy over 90% on the

produced datasets. Referring to the table comparing architectural differences of these two builds, the only difference is the additional Conv2D(128) and Maxpool2D layers in CNN 1.3. CNN 1.4 contains Dropout(0.25) layers after every Convolution-Pooling groups. Dropout layers usually improve the performance by reducing the overfitting, but here it made the architecture more complex and resulted in one of lowest accuracy (of about 32%) among all the builds. Among CNN 1.2 and CNN 1.3, any of these builds can be chosen for the final deployment to make predictions in real-time. This project further used CNN 1.3 to evaluate the research goal of evaluating people's opinion towards using hand-signs to interact with the humanoid robot. From Figure 7.5 that represents the responsiveness of the prototype, it is evident that the majority of participants rated the prototype to have a fast or quick response. All the positive responses for the Question 2 (*Would you like to use non-verbal communication (hand-signs, facial expression, eye contact, haptics, etc.) to access different features of the robot?*) indicate people are willing to adopt non-verbal communication to interact with humanoid robot. This satisfies the research goal and further observing the Figure 7.6 shows people's interests in possible areas to implement nonverbal communication. Nearly half the participants would even like to have personalized services when accessing the robot. Though this study involved few participants, but the majority of positive responses towards non-verbal communication suggest a strong potential in adopting non-verbal features to improve human robot interaction in humanoid robots.

# / 8

## Discussion

This dissertation essentially explores the use of non-verbal communication to establish human-robot interaction in humanoid robots. The prototype described in the Chapter 6 and evaluation results presented in the Chapter 7 strive to highlight a way on how to achieve it and the people's willingness in adopting non-verbal communication for achieving HRI. There are three major components forming the prototype - the NAO humanoid robot, the Integration Layer, and the Hand-Signs Recognition Component. This chapter further discusses about the choices taken throughout the research for developing these components and to evaluate the research along with the limiting factors.

The prototype system enables an NAO humanoid robot to interact with its user using hand-signs and perform various tasks based on the recognized sign. NAO robot acts as a tool for interaction that recognizes three hand-signs - 'Palms-Open', 'Thumbs-Up', and 'OK' Sign and performs different actions on a successful recognition. NAO captures an image via its camera and sends it to the Integration layer for further processing. On receiving the output, it performs different actions based on the recognized sign. Though this research focusses on using non-verbal communication, some of the commands like starting or stopping the recognition scenario still require voice inputs from the user. Different colour codes for eyes were defined to interpret the NAO's current state as described in Chapter 5. NAO is entirely developed using the built-in modules provided by the Choregraphe software suite. The integration layer acting as a middleware for the NAO robot and the HSRC - is a webserver hosted using the Flask web framework carrying out different preprocessing

operations and loading the environment to run the deep learning model for real-time predictions.

The most significant part of the prototype is the Hand-Signs Recognition Component (HSRC) - that is fundamentally a Deep Learning model trained using Convolutional Neural Networks to detect static hand-signs (Palms-Open, Thumbs-Up, and OK). Though there are different methods in computer vision to handle hand-signs recognition, deep learning proves to be the best due to its data-driven approach as discussed in Chapter 3. Deep learning techniques thrive on large amounts of 'good' quality data, but it is harder to define what qualifies as a 'good' quality data before the training process. Thus, one may have to revisit the data generation and data preprocessing phases to generate more qualitative data for the model training process. During the development, the data generation phase was repeated a few times, adding more 'good quality' images in every iteration ('good quality' refers to images having different lighting conditions, backgrounds, etc.). Image preprocessing techniques such as background removal, Gaussian filters were carried out but did not result in better quality images. Hence, the images are directly fed to the Convolutional Neural Networks in the model training phase. This truly highlights the robustness of CNN in such image classification problems in computer vision. Another way to handle the limitations with smaller datasets is to use data-augmentation. Data augmentation is a technique that generates more training samples from the existing dataset by applying various types of transformations. It indeed played a significant role in this thesis to improve the performance of the deep learning models.

As described in the Chapter 3, deep learning models are computationally intensive and to ensure faster model training, they require more computation power to run on. Hence, a decent GPU performs exceptionally well in such problems. The deep learning models developed by the HSRC uses the Keras frameworks running over the TensorFlow 2.1 backend with GPU support. The resulting models by the Hand-Signs Recognition Component can be seamlessly integrated with the NAO robot using the Flask web framework. But the configuration of Keras and TensorFlow was a complicated process. This conventional form of configuring deep learning environments can be replaced with cloud platforms offering machine learning and deep learning support such as Amazon Web Services, Microsoft Azure, and others. These platforms offer paid services that provide easy configuration of the hardware, required libraries, version control, and automated deployment for various deep learning projects.

Testing the deep learning models yielded good results depending upon the way the user interacts to NAO's questions in the interaction. In the beginning, the user may have to repeat the voice commands to familiarize with the robot to understand - when the robot is waiting for the input, or capturing an image, or

processing the results. Though the models with the custom CNN architecture performed better than the ones using transfer learning, transfer learning is a powerful technique that implicitly uses other models trained over large datasets to one's specific use-case.

To evaluate the research, the initial plan to collect data was to conduct a semi-structured interview. The interview will begin by showing a live demonstration of the prototype to the participants and later conduct a questionnaire to record their feedback. But due to the COVID-19 pandemic occurring from 13 March 2020 until 15 August 2020, this research adopted a modernised technique of data collection using online questionnaires. An online questionnaire in simplest terms is a questionnaire conducted online, generally by hosting it on a server or using cloud platforms providing such services. Though the researcher needs to be aware of limiting his bias when forming such questionnaires, the online version provides a wider and easier reach to collect data from the target subjects. But online questionnaires may not be feasible choice if the research needs to target a specific type of people, since filtering such participants could be a tedious job and lead to noisy data.

The results from the online questionnaire show that people are willing to use non-verbal communication as a means to interact with humanoid robots, which was evident from the all the "Yes" responses to '*Question 2: Would you like to use non-verbal communication (hand-signs, facial expression, eye contact, haptics, etc.) to access different features of the robot?*' of the online questionnaire. Majority of the participants looked forward to using non-verbal communication to personalize features of a humanoid robot, which was observed from the results to '*Question 3: If you had a humanoid robot, for what will you use hand-sign recognition feature?*' of the questionnaire. Though the questionnaire involved only 56 participants, their responses shows a positive acceptance of the use of non-verbal communication in interacting with humanoid robots and could encourage robot designers to use it for enhancing human-robot interaction.





# /9

## Conclusion and Future Scope

This thesis presents a humanoid robot that establishes human-robot interaction (HRI) with its users using non-verbal communication, especially using hand-signs. The motivation behind it was to improve the acceptance of humanoid robots in daily usage by enhancing the quality of communication with its users. Humanoid robots can establish a communication either verbally or non-verbally. Verbal communication is the primary form of communication and is well-developed in the field of human-robot interaction. But non-verbal communication even being an integral part of human interactions is not yet widely used for humanoid robots. This research strives to use that impact of non-verbal communication in human-human interactions, and provides a way to use it for human-robot interactions. Both the research goals are satisfied where first a working prototype of NAO humanoid robot handles hand-signs recognition, and second where a study reflects people's opinion on adopting non-verbal features to access an robot.

The prototype development began with the development of the NAO robot, acting as a means to conduct this research. It establishes the HRI with the user by capturing an image of the user's hand and performing a specific action on a successful recognition along with some animations. The integration layer acts as a middleware between the NAO and the Hand-Signs Recognition Component (HSRC), which is the core element of the prototype. The HSRC generates

various deep learning models using convolution neural networks to recognize hand-signs from the input image. Out of the several models developed, custom CNN architectures resulted in a training accuracy over 90% Chapter 7. HSRC can be easily decoupled from other components of the prototype to run independently as a standalone application to aid people with hearing disabilities or perform hand-signs recognition for any external application or service. The implementation details are available on the *Github*[88, 106] link for those interested in it.

The methodologies from software engineering and prototyping served very well in conducting this research. This work adopts an alternative technique of online questionnaires to collect data in this thesis. As there was no particular target audience to conduct this research, online questionnaires made it easier to reach more people. Creating the questionnaire certainly needed planning, but online questionnaires do not provide the opportunity of observing participants in the live interaction. The results from the online questionnaires shows unanimous response to adopting non-verbal features to access the robot. Majority of the participants were also looking forward to have personalized features from the robot.

Considering the methodologies and results achieved in the current research, a multitude of design improvements are possible in the current prototype: Few of them are listed below:

- **Personalizing the user experience:** Personalized user experience is the key to enhancing the human-robot interaction where an user can customize his hand signs to interact with the robot and further assign desired actions to these hand-signs. For example - User 1 assigns a new sign like 'Thumbs Down' and assigns it to 'reduce the volume of the NAO robot'.
- **Developing the Profile Manager:** As proposed in the prototype design of this research, developing a profile manager includes creating a new user profile and storing the user's sign-to-action mapping. This will allow multiple people in the home or organization to use the same robot but for their own desired task.
- **Enhancing the security and privacy:** Security and privacy of users is utmost important if above features are included in the robot. Possible solution is to use facial recognition features to authenticate a user and further load the user-profile in the NAO's memory for use.
- **Automating the model development using cloud platforms:** Installation and configuration setup for a deep learning project is a tedious

process that often faces problems with hardware and software incompatibilities. In such cases, cloud services provide managed services that provide computing resources with all the required libraries, scheduled updates, and greater flexibility to train large deep learning models.

Since the HSRC is loosely-coupled, it is easy to use it with an external application requiring just the hand-signs recognition feature. In situations such as a global pandemic of COVID-19 where touching items or surfaces is restricted, the HSRC can be integrated into a digital media device to control its different features. For example - The 'Palms-Open' sign shown to a TV can switch on the TV, the 'Thumbs-Up' sign can increase the volume of the TV, and so on. The entire system including the NAO robot is also useful as a sign language interpreter. People with hearing impairments use sign language for communication. Many do not know sign language and find it difficult to interact with these people. In such cases, NAO can record the sequence of signs shown to it and later would translate it into speech for others. NAO has already been successful in elderly care. Using hand-signs recognition, elders who have movement restrictions may use hand-signs to perform tasks like opening/closing of the door or adjusting the thermostat.



# Bibliography

- [1] Avishek Choudhury, Huiyang Li, and Christopher M Greene. “Humanoid Robot: Application and Influence.” In: *International Journal of Applied Science - Research and Review* 05.04 (2018). ISSN: 23949988. DOI: 10.21767/2394-9988.100082. URL: <http://www.imedpub.com/articles/humanoid-robot--application-and-influence.php?aid=23790>.
- [2] Henny Admoni. “Nonverbal Communication in Socially Assistive Human-Robot Interaction.” In: *AI Matters* 2.4 (Dec. 8, 2016), pp. 9–10. ISSN: 23723483. DOI: 10.1145/3008665.3008669. URL: <http://dl.acm.org/citation.cfm?doid=3008665.3008669>.
- [3] Susan Goldin-Meadow and Martha Wagner Alibali. “Gesture’s Role in Speaking, Learning, and Creating Language.” In: *Annual review of psychology* 64 (2013), pp. 257–283. ISSN: 0066-4308. DOI: 10.1146/annurev-psych-113011-143802. pmid: 22830562. URL: <https://www.ncbi.nlm.nih.gov/pmc/articles/PMC3642279/>.
- [4] Social Sci. 4.2: *Types of Nonverbal Communication - Social Sci LibreTexts*. URL: [https://socialsci.libretexts.org/Bookshelves/Communication/Book%3A\\_Communication\\_in\\_the\\_Real\\_World/04%3A\\_Nonverbal\\_Communication/4.02%3A\\_Types\\_of\\_Nonverbal\\_Communication](https://socialsci.libretexts.org/Bookshelves/Communication/Book%3A_Communication_in_the_Real_World/04%3A_Nonverbal_Communication/4.02%3A_Types_of_Nonverbal_Communication).
- [5] SoftBank Robotics. *NAO the Humanoid and Programmable Robot | SoftBank Robotics*. URL: <https://www.softbankrobotics.com/emea/en/nao>.
- [6] gminsights. *Humanoid Robot Market Size to Exceed 5.5bn by 2024*. URL: <https://www.gminsights.com/pressrelease/humanoid-robot-market>.
- [7] Erhan Oztop, JST-ICORP Computational Brain Project, and Keihanna Science City. “HUMANHUMANOID INTERACTION: IS A HUMANOID ROBOT PERCEIVED AS A HUMAN?” In: (2006), p. 23.
- [8] I. A. Hameed. “Using Natural Language Processing (NLP) for Designing Socially Intelligent Robots.” In: *2016 Joint IEEE International Conference on Development and Learning and Epigenetic Robotics (ICDL-EpiRob)*. 2016 Joint IEEE International Conference on Development and Learning and Epigenetic Robotics (ICDL-EpiRob). Sept. 2016, pp. 268–269. DOI: 10.1109/DEVLRN.2016.7846830.

- [9] elsevier. *Chapter 15 - Affective HumanRobot Interaction | Elsevier Enhanced Reader*. DOI: 10.1016/B978-0-12-801851-4.00015-X. URL: <https://reader.elsevier.com/reader/sd/pii/B978012801851400015X>.
- [10] Marilyn Ong, Ling Xi Ying, and Regina Wong. *Chapter 15 Evolution of Nonverbal Communication in Hominids | Language Evolution*. URL: <https://blogs.ntu.edu.sg/hss-language-evolution/wiki/chapter-15/>.
- [11] Melinda. *Nonverbal Communication - HelpGuide.Org*. Nov. 2, 2018. URL: <https://www.helpguide.org/articles/relationships-communication/nonverbal-communication.htm>.
- [12] Anne Hakansson and Ronald Lee Hartung. *The Artificial Intelligence Book: Concepts, Areas, Techniques and Applications*. Studentlitteratur. ISBN: 978-91-44-12599-2.
- [13] Kajonpong, Punsak. "Recognizing American Sign Language Using Deep Learning." M.S. Ann Arbor, United States, 2019. 62 pp. ISBN: 9781392180211. URL: <https://search.proquest.com/pqdtglobal/docview/2235416803/abstract/3F838718CF8C441CPQ/3>.
- [14] Ahmed Kadem Hamed Al-Saedi and Abbas H Hassin Al-Asadi. "Survey of Hand Gesture Recognition Systems." In: *Journal of Physics: Conference Series* 1294 (Sept. 2019), p. 042003. ISSN: 1742-6588, 1742-6596. DOI: 10.1088/1742-6596/1294/4/042003. URL: <https://iopscience.iop.org/article/10.1088/1742-6596/1294/4/042003>.
- [15] SkillsYouNeed. *Non-Verbal Communication | SkillsYouNeed*. URL: <https://www.skillsyouneed.com/ips/nonverbal-communication.html>.
- [16] Mikael Berndtsson, ed. *Thesis Projects: A Guide for Students in Computer Science and Information Systems*. 2nd ed. London: Springer, 2008. 158 pp. ISBN: 978-1-84800-008-7.
- [17] Anne Haakansson. "Portal of Research Methods and Methodologies for Research Projects and Degree Projects." In: *Computer Engineering* (2013), p. 8.
- [18] Eileen M. Trauth, ed. *Qualitative Research in IS: Issues and Trends*. USA: IGI Global, 2001. 287 pp. ISBN: 978-1-930708-06-8.
- [19] M R De Villiers. "Three Approaches as Pillars for Interpretive Information Systems Research: Development Research, Action Research and Grounded Theory." In: (), p. 10.
- [20] Miro Jakovljevic et al. "COVID-19 Pandemia and Public and Global Mental Health from the Perspective of Global Health Securit." In: *Psychiatr Danub* (2020), pp. 6–14. URL: <https://dx.doi.org/10.24869/psyd.2020.6>.
- [21] Jan van den Akker. "DEFINITIONS AND AIMS OF DEVELOPMENT RESEARCH 3.1 Conceptual Confusion." In: 2000. URL: <https://www.semanticscholar.org/paper/DEFINITIONS-AND-AIMS-OF-DEVELOPMENT-RESEARCH-3-.-1-Akker/e1a57b0704bd9dd2417fb579ce1bcb5b09fde26a?p2df>.

- [22] Sanjit Singh Dang. *Artificial Intelligence In Humanoid Robots*. URL: <https://www.forbes.com/sites/cognitiveworld/2019/02/25/artificial-intelligence-in-humanoid-robots/>.
- [23] P. Joglekar and V. Kulkarni. "Humanoid Robot as a Companion for the Senior Citizens." In: *2018 IEEE Punecon*. 2018 IEEE Punecon. Nov. 2018, pp. 1–4. DOI: 10.1109/PUNECON.2018.8745399.
- [24] Wikipedia. *Nao (Robot)*. In: *Wikipedia*. July 22, 2020. URL: [https://en.wikipedia.org/w/index.php?title=Nao\\_\(robot\)&oldid=968928470](https://en.wikipedia.org/w/index.php?title=Nao_(robot)&oldid=968928470).
- [25] Aldebaran. *NAO H25 Aldebaran 2.1.4.13 Documentation*. URL: [http://doc.aldebaran.com/2-1/family/nao\\_h25/index\\_h25.html#nao-h25](http://doc.aldebaran.com/2-1/family/nao_h25/index_h25.html#nao-h25).
- [26] Aldebaran. *NAO - Video Camera Aldebaran 2.1.4.13 Documentation*. URL: [http://doc.aldebaran.com/2-1/family/robots/video\\_robot.html#robot-video](http://doc.aldebaran.com/2-1/family/robots/video_robot.html#robot-video).
- [27] Aldebaran. *What Is Choregraphe Aldebaran 2.4.3.28-R2 Documentation*. URL: [http://doc.aldebaran.com/2-4/software/choregraphe/choregraphe\\_overview.html](http://doc.aldebaran.com/2-4/software/choregraphe/choregraphe_overview.html).
- [28] Aldebaran. *Microphones Aldebaran 2.1.4.13 Documentation*. URL: [http://doc.aldebaran.com/2-1/family/robots/microphone\\_robot.html#robot-microphone](http://doc.aldebaran.com/2-1/family/robots/microphone_robot.html#robot-microphone).
- [29] Aldebaran. *NAOqi Framework Aldebaran Software 2.1.0.18 Documentation*. URL: <http://fileadmin.cs.lth.se/robot/nao/doc/ref/index.html>.
- [30] isocpp. *Standard C++*. URL: <https://isocpp.org/>.
- [31] python. *Welcome to Python.Org*. URL: <https://www.python.org/>.
- [32] Aldebaran. *NAOqi APIs Aldebaran Software 2.1.0.18 Documentation*. URL: <http://fileadmin.cs.lth.se/robot/nao/doc/naoqi/index.html>.
- [33] Aldebaran. *Menus, Panels and Toolbar in a Glance Aldebaran 2.1.4.13 Documentation*. URL: <http://doc.aldebaran.com/2-1/software/choregraphe/interface.html>.
- [34] Michael A. Goodrich and Alan C. Schultz. "Human-Robot Interaction: A Survey." In: *Foundations and Trends registered in Human-Computer Interaction* 1.3 (2007), pp. 203–275. ISSN: 1551-3955, 1551-3963. DOI: 10.1561/1100000005. URL: <http://www.nowpublishers.com/article/Details/HCI-005>.
- [35] KERSTIN DAUTENHAHN. *Human-Robot Interaction*. URL: <https://www.interaction-design.org/literature/book/the-encyclopedia-of-human-computer-interaction-2nd-ed/human-robot-interaction>.
- [36] Mark L. Knapp, Judith A. Hall, and Terrence G. Horgan. *Nonverbal Communication in Human Interaction*. Cengage Learning, Jan. 1, 2013. 530 pp. ISBN: 978-1-133-31159-1.
- [37] Wikipedia. *Gesture*. In: *Wikipedia*. July 5, 2020. URL: <https://en.wikipedia.org/w/index.php?title=Gesture&oldid=966228997>.

- [38] gestures334k. *Gestures*. Apr. 9, 2013. URL: <https://gestures334k.wordpress.com/2013/04/09/gestures/>.
- [39] Alex Case. *British and American Body Language and Gestures*. Sept. 14, 2018. URL: <https://www.usingenglish.com/articles/british-american-body-language-gestures.html>.
- [40] guru99. *AI vs Machine Learning vs Deep Learning: What's the Difference?* URL: <https://www.guru99.com/machine-learning-vs-deep-learning.html>.
- [41] Andres Munoz. "Machine Learning and Optimization." In: (), p. 14.
- [42] Pariwat Ongsulee. "Artificial Intelligence, Machine Learning and Deep Learning." In: *2017 15th International Conference on ICT and Knowledge Engineering (ICT KE)*. 2017 15th International Conference on ICT and Knowledge Engineering (ICT KE). Nov. 2017, pp. 1–6. DOI: 10.1109/ICTKE.2017.8259629.
- [43] Francois Chollet. *Deep Learning with Python*. 1st. USA: Manning Publications Co., 2017. 384 pp. ISBN: 978-1-61729-443-3.
- [44] teco. *Biological Neural Networks*. URL: <https://www.teco.edu/~albrecht/neuro/html/node7.html>.
- [45] Kirill Eremenko. "Deep Learning A-Z: Artificial Neural Networks (ANN) - Module 1." Education. URL: [https://www.slideshare.net/KirillEremenko/deep-learning-az-artificial-neural-networks-ann-module-1?qid=040da44b-e009-4149-ad8f-b16f3fcb5e18&v=&b=&from\\_search=2](https://www.slideshare.net/KirillEremenko/deep-learning-az-artificial-neural-networks-ann-module-1?qid=040da44b-e009-4149-ad8f-b16f3fcb5e18&v=&b=&from_search=2).
- [46] Mohan. *What Is Perceptron | Simplilearn*. URL: <https://www.simplilearn.com/what-is-perceptron-tutorial>.
- [47] Ayyuce Kizrak. *Comparison of Activation Functions for Deep Neural Networks*. June 8, 2019. URL: <https://towardsdatascience.com/comparison-of-activation-functions-for-deep-neural-networks-706ac4284c8a>.
- [48] Xavier Glorot, Antoine Bordes, and Yoshua Bengio. "Deep Sparse Rectifier Neural Networks." In: *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*. Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics. June 14, 2011, pp. 315–323. URL: <http://proceedings.mlr.press/v15/glorot11a.html>.
- [49] chm. *Softmax Function Beyond the Basics*. May 11, 2019. URL: <https://mc.ai/softmax-function-beyond-the-basics/>.
- [50] Google. *Multi-Class Neural Networks: Softmax | Machine Learning Crash Course*. URL: <https://developers.google.com/machine-learning/crash-course/multi-class-neural-networks/softmax>.
- [51] Kirill Eremenko. "Deep Learning A-Z: Convolutional Neural Networks (CNN) - Module 2." Education. URL: [https://www.slideshare.net/KirillEremenko/deep-learning-az-convolutional-neural-networks-cnn-module-2?qid=040da44b-e009-4149-ad8f-b16f3fcb5e18&v=&b=&from\\_search=1](https://www.slideshare.net/KirillEremenko/deep-learning-az-convolutional-neural-networks-cnn-module-2?qid=040da44b-e009-4149-ad8f-b16f3fcb5e18&v=&b=&from_search=1).



- [52] Madhushree Basavarajaiah. *6 Basic Things to Know about Convolution*. Apr. 2, 2019. URL: <https://medium.com/@bdhuma/6-basic-things-to-know-about-convolution-daef5e1bc411>.
- [53] Prabhu. *Understanding of Convolutional Neural Network (CNN) Deep Learning*. Mar. 4, 2018. URL: <https://medium.com/@RaghavPrabhu/understanding-of-convolutional-neural-network-cnn-deep-learning-99760835f148>.
- [54] "CHAPTER 19 - NEAR REAL-TIME ROBUST FACE AND FACIAL-FEATURE DETECTION WITH INFORMATION-BASED MAXIMUM DISCRIMINATION." In: *Face Processing*. Ed. by Wenyi Zhao and Rama Chellappa. Burlington: Academic Press, 2006, pp. 630–633. ISBN: 978-0-12-088452-0. DOI: 10.1016/B978-012088452-0/50020-0. URL: <http://www.sciencedirect.com/science/article/pii/B9780120884520500200>.
- [55] David Hutchison et al. "Evaluation of Pooling Operations in Convolutional Architectures for Object Recognition." In: *Artificial Neural Networks ICANN 2010*. Ed. by Konstantinos Diamantaras, Wlodek Duch, and Lazaros S. Iliadis. Vol. 6354. Berlin, Heidelberg: Springer Berlin Heidelberg, 2010, pp. 92–101. ISBN: 978-3-642-15824-7. DOI: 10.1007/978-3-642-15825-4\_10. URL: [http://link.springer.com/10.1007/978-3-642-15825-4\\_10](http://link.springer.com/10.1007/978-3-642-15825-4_10).
- [56] Sumit Saha. *A Comprehensive Guide to Convolutional Neural Networks the ELI5 Way*. Dec. 17, 2018. URL: <https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53>.
- [57] geva. *Fully Connected Layers in Convolutional Neural Networks: The Complete Guide*. URL: <https://missinglink.ai/guides/convolutional-neural-networks/fully-connected-layers-convolutional-neural-networks-complete-guide/>.
- [58] Tarang Shah. *About Train, Validation and Test Sets in Machine Learning*. Dec. 10, 2017. URL: <https://towardsdatascience.com/train-validation-and-test-sets-72cb40cba9e7>.
- [59] racheldraelos. *Best Use of Train/Val/Test Splits, with Tips for Medical Data*. Sept. 15, 2019. URL: <https://glassboxmedicine.com/2019/09/15/best-use-of-train-val-test-splits-with-tips-for-medical-data/>.
- [60] Jeremy Jordan. *Hyperparameter Tuning for Machine Learning Models*. Nov. 2, 2017. URL: <https://www.jeremyjordan.me/hyperparameter-tuning/>.
- [61] Jason Brownlee. *Gentle Introduction to the Bias-Variance Trade-Off in Machine Learning*. Mar. 17, 2016. URL: <https://machinelearningmastery.com/gentle-introduction-to-the-bias-variance-trade-off-in-machine-learning/>.

- [62] Artem Oppermann. *Overfitting and Underfitting in Deep Learning*. URL: <https://www.deeplearning-academy.com/p/ai-wiki-overfitting-underfitting>.
- [63] Yash Upadhyay. *Regularization Techniques for Neural Networks*. Mar. 15, 2019. URL: <https://towardsdatascience.com/regularization-techniques-for-neural-networks-e55f295f2866>.
- [64] Nitish Srivastava et al. "Dropout: A Simple Way to Prevent Neural Networks from Overfitting." In: *Journal of Machine Learning Research* 15.56 (2014), pp. 1929–1958. ISSN: 1533-7928. URL: <http://jmlr.org/papers/v15/srivastava14a.html>.
- [65] Jason Brownlee and Machine Learning Mastery. *Deep Learning with Python: Develop Deep Learning Models on Theano and TensorFlow Using Keras*. Machine Learning Mastery, 2017. URL: <https://books.google.no/books?id=eJw2nQAACAAJ>.
- [66] Google. *Regularization for Simplicity: L<sub>2</sub> Regularization*. URL: <https://developers.google.com/machine-learning/crash-course/regularization-for-simplicity/l2-regularization>.
- [67] Khush Patel. *Overfitting vs Underfitting*. Sept. 14, 2019. URL: <https://towardsdatascience.com/overfitting-vs-underfitting-ddc80c2fc00d>.
- [68] Ren C. Luo and Yen-Chang Wu. "Hand Gesture Recognition for Human-Robot Interaction for Service Robot." In: *2012 IEEE International Conference on Multisensor Fusion and Integration for Intelligent Systems (MFI)*. 2012 IEEE International Conference on Multisensor Fusion and Integration for Intelligent Systems (MFI). Sept. 2012, pp. 318–323. DOI: 10.1109/MFI.2012.6343059.
- [69] Rohith Gandhi. *Support Vector Machine Introduction to Machine Learning Algorithms*. July 5, 2018. URL: <https://towardsdatascience.com/support-vector-machine-introduction-to-machine-learning-algorithms-934a444fca47>.
- [70] Alberto Tellaeche, Johan Kildal, and Inaki Maurtua. *A Flexible System for Gesture Based Human-Robot Interaction*. DOI: 10.1016/j.procir.2018.03.017. URL: <https://reader.elsevier.com/reader/sd/pii/S221282711830115X>.
- [71] Wikipedia. *Kinect*. In: *Wikipedia*. May 26, 2020. URL: <https://en.wikipedia.org/w/index.php?title=Kinect&oldid=958970348>.
- [72] ROS. *ROS.Org | Powering the World's Robots*. URL: <https://www.ros.org/>.
- [73] Vivek Bheda and Dianna Radpour. "Using Deep Convolutional Networks for Gesture Recognition in American Sign Language." In: *ArXiv abs/1710.06836* (2017).
- [74] Vijay Vaishnavi and B Kuechler. "Design Science Research in Information Systems." In: *Association for Information Systems* (Jan. 1, 2004).

- [75] Ken Peffers et al. "A Design Science Research Methodology for Information Systems Research." In: *Journal of Management Information Systems* (2008), pp. 45–77.
- [76] Ian Sommerville. *Software Engineering*. 9th ed. USA: Addison-Wesley Publishing Company, 2010. ISBN: 0-13-703515-2.
- [77] Rajendra Ganpatrao Sabale. "Comparative Study of Prototype Model For Software Engineering With System Development Life Cycle." In: *IOSR Journal of Engineering* 02.07 (July 2012), pp. 21–24. ISSN: 22788719, 22503021. DOI: 10.9790/3021-02722124. URL: [http://www.iosrjen.org/Papers/vol12\\_issue7%20\(part-2\)/D0272124.pdf](http://www.iosrjen.org/Papers/vol12_issue7%20(part-2)/D0272124.pdf).
- [78] Stephan Wensveen and Ben Matthews. "Prototypes and Prototyping in Design Research." In: *The Routledge Companion to Design Research*. Ed. by Paul A. Rodgers and Joyce Yee. 1st ed. Routledge, Oct. 17, 2014, pp. 262–276. ISBN: 978-1-315-75846-6. DOI: 10.4324/9781315758466-25. URL: <https://www.taylorfrancis.com/books/9781317636250/chapters/10.4324/9781315758466-25>.
- [79] AltexSoft. *Functional and Nonfunctional Requirements: Specification and Types*. URL: <https://www.altexsoft.com/blog/business/functional-and-non-functional-requirements-specification-and-types/>.
- [80] guru99. *Functional Requirements vs Non Functional Requirements: Key Differences*. URL: <https://www.guru99.com/functional-vs-non-functional-requirements.html>.
- [81] Software Testing Fundamentals. *Functional Testing*. Dec. 9, 2012. URL: <http://softwaretestingfundamentals.com/functional-testing/>.
- [82] Aditya Mishra. *Metrics to Evaluate Your Machine Learning Algorithm*. Nov. 1, 2018. URL: <https://towardsdatascience.com/metrics-to-evaluate-your-machine-learning-algorithm-f10ba6e38234>.
- [83] Northern Illinois University. *Data Collection*. URL: [https://ori.hhs.gov/education/products/n\\_illinois\\_u/datamanagement/dctopic.html](https://ori.hhs.gov/education/products/n_illinois_u/datamanagement/dctopic.html).
- [84] qualres. *RWJF - Qualitative Research Guidelines Project | Interviewing | Interviewing*. URL: <http://www.qualres.org/HomeInte-3595.html>.
- [85] Microsoft. *Microsoft Forms*. URL: <https://forms.office.com/Pages/DesignPage.aspx>.
- [86] Google. *Google Forms: Free Online Surveys for Personal Use*. URL: <https://www.google.com/forms/about/>.
- [87] William M. K. Trochim. *Descriptive Statistics*. URL: <https://conjointly.com/kb/descriptive-statistics/>.
- [88] Mayuresh Amberkar. *Mayureshsa/Masters\_Thesis\_hsr*. Aug. 12, 2020. URL: [https://github.com/mayureshsa/masters\\_thesis\\_hsr](https://github.com/mayureshsa/masters_thesis_hsr).
- [89] Aldebaran. *OpenNAO - NAO OS NAO Software 1.14.5 Documentation*. URL: <http://doc.aldebaran.com/1-14/dev/tools/openna.html>.
- [90] Dipanjan (DJ) Sarkar. *A Comprehensive Hands-on Guide to Transfer Learning with Real-World Applications in Deep Learning*. Nov. 17, 2018.

- URL: <https://towardsdatascience.com/a-comprehensive-hands-on-guide-to-transfer-learning-with-real-world-applications-in-deep-learning-212bf3b2f27a>.
- [91] SCOTT MARTIN. *What Is Transfer Learning?* | *NVIDIA Blog*. Feb. 7, 2019. URL: <https://blogs.nvidia.com/blog/2019/02/07/what-is-transfer-learning/>.
- [92] tensorflow. *TensorFlow*. URL: <https://www.tensorflow.org/>.
- [93] keras. *Home - Keras Documentation*. URL: <https://keras.io/>.
- [94] Apache. *Apache Spark - Unified Analytics Engine for Big Data*. URL: <https://spark.apache.org/>.
- [95] Apache. *Apache MXNet*. URL: <https://mxnet.apache.org/>.
- [96] pytorch. *PyTorch*. URL: <https://www.pytorch.org>.
- [97] caffe. *Caffe | Deep Learning Framework*. URL: <https://caffe.berkeleyvision.org/>.
- [98] Jonathan Hui. *Deep Learning Designs (Part 3)*. Feb. 11, 2020. URL: [https://medium.com/@jonathan\\_hui/deep-learning-designs-part-3-e0b15ef09ccc](https://medium.com/@jonathan_hui/deep-learning-designs-part-3-e0b15ef09ccc).
- [99] Wikipedia. *JPEG*. In: *Wikipedia*. July 29, 2020. URL: <https://en.wikipedia.org/w/index.php?title=JPEG&oldid=970140308>.
- [100] towardsdatascience. *Image Pre-Processing - Towards Data Science*. URL: <https://towardsdatascience.com/image-pre-processing-c1aec0be3edf>.
- [101] Tom Backstrom. *Wake-Word and Keyword Spotting - Introduction to Speech Processing - Aalto University Wiki*. In: *Aalto University Wiki*. Sept. 3, 2019. URL: <https://wiki.aalto.fi/display/ITSP/Wake-word+and+keyword+spotting>.
- [102] CloudFactory. *The Essential Guide to Quality Training Data for Machine Learning*. URL: <https://www.cloudfactory.com/training-data-guide>.
- [103] Pedro Marcelino. *Transfer Learning from Pre-Trained Models*. Oct. 23, 2018. URL: <https://towardsdatascience.com/transfer-learning-from-pre-trained-models-f2393f124751>.
- [104] Dallas Card. *The ‘Black Box’ Metaphor in Machine Learning*. July 5, 2017. URL: <https://towardsdatascience.com/the-black-box-metaphor-in-machine-learning-4e57a3a1d2b0>.
- [105] Raimi Karim. *Illustrated: 10 CNN Architectures*. Oct. 17, 2019. URL: <https://towardsdatascience.com/illustrated-10-cnn-architectures-95d78ace614d>.
- [106] Mayuresh Amberkar, director. *Humanoid Robot Handling Hand Signs Recognition*. May 6, 2020. URL: [https://www.youtube.com/watch?v=tmxf\\_UQG9PM](https://www.youtube.com/watch?v=tmxf_UQG9PM).