Chapter 1

Introduction

Humanity has consistently made progress in science and technology throughout its history. As time passes, people have increasingly integrated these advances into their daily lives to make their work more efficient. And this increase in efficiency is mainly due to significant and visible developments in the field of semiconductor technology. The growth of silicon-based chip technology has led to numerous discoveries and innovations in a wide range of fields. For instance, silicon-based chips allow computers to solve complex mathematical problems quickly, and they enable mobile devices to capture high-resolution images.

Amongst all these technological advancements, the computer has had a profound impact on society and is considered one of the most significant inventions in human history. It has greatly improved our ability to manage tasks and make our work more efficient. Over time, computers have become more compact and efficient, and the way we interact with them has also evolved. In the past, computers required a certain level of technical knowledge to operate, and people had to use keyboard commands to enter data and execute tasks. However, with the advent of graphical user interfaces (GUI) and the widespread use of the mouse, computers became much easier to use for people of all skill levels. This has made it possible for people to use computers for

a wide variety of tasks, from communication and entertainment to data analysis and problem-solving. To see this change, we can take the example of a telephone device. There has been continuous development in telephonic devices and the mode to operate it has also changed—Figure 1.1 illustrates the same. The leftmost figure is of a rotary dial telephone for which the way to input a telephone number is to rotate the dial sequentially. The following change in telephone devices came in the form of a keypad (or push-button) which replaced rotary dials. The keypad-based phone made dialing a number easier and faster. Next, the current technological change has replaced the keypad with a touch-based hand gesture. Just by making a relevant hand gesture on a screen, we can connect to other people. Maybe in the future is where touch-based gestures will be replaced by touch-less gestures or any other form of Human-Computer Interaction (HCI) technology. One such change can be explained with an example illustrated in Figure 1.2. A gesture-based feature present in smartphones enables a user to click pictures just by showing their palm to the camera instead of pressing a button for the same.

The example presented above is part of a system known as natural user interface (NUI), a branch of HCI. As the name suggests, the objective is to make interaction with computers more natural and intuitive. Bill Gates, co-founder of Microsoft believes that with NUI, computing devices will adopt and conform to the needs and preferences of humans in which they believe it is comfortable and natural for them to use computers. An example of a user interface that people often refer to as natural and easy to use

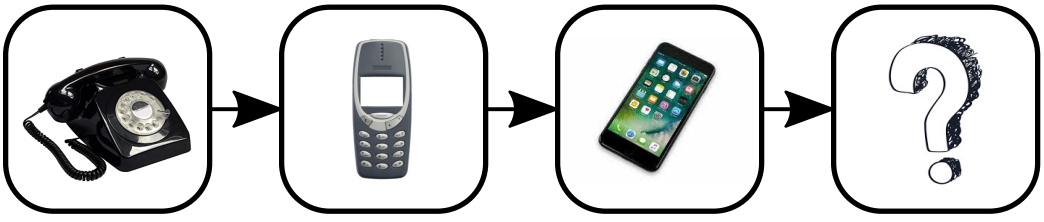


Figure 1.1: The images left to right illustrate the development in the interface through which we interact with the telephone device.

(a) Clicking photo using a physical button

(b) Making a hand gesture to click photo

Figure 1.2: Illustration of the evolution of technology that enables the way we take pictures using our smartphone.

is a multi-touch screen on smartphones. The control of the device is easier with a touch-based gesture than with a cursor-based interface. In a cursor-based interface, the pointer has to be moved from one point to another to perform an application.

There are many forms of the natural user interface. Some of them are briefly described below

- *Speech recognition system* – Allows users to interact with a system by passing commands through speech. The spoken words are captured and translated into machine-understandable form. The machine then takes action based on interpreted instructions. One possible application is the speech-to-text conversion for automatic transcription.
- *Gaze-tracking interface* – Using eye movement information as an input to control the system.
- *Brain-machine interface* – Reading the neural signals and translating those signals to actions. This type of system is particularly important for persons with paralysis. It will enable them to operate the computer through thoughts.
- *Gesture recognition system* – Detection and tracking of the positions of certain keypoints in a human body such as shoulder and positions to identify the body pose. Then translate the information conveyed through these poses into an action.

The hand gesture is another example where different hand poses can be used to pass different signals which can be used for many applications.

The products or services designed for the user interface hold significant market share. According to Market Research Future (MRFR) report [5] published in globenewswire.com, the global user interface services market size is predicted to be USD 50 billion at a 16% CAGR between 2017 to 2027. The voice-based user interface service has a market size of USD 29.02 billion from 2021 to 2026. It is also forecasted that the gesture-based user interface will have a market size of USD 43.25 billion by 2031 with a CAGR of 20.10% between 2022-2030.

The market is seeing growth in technologies based on gesture recognition. Factors of hygiene problems, the demand for contactless technologies, and technologies with advanced precision have pushed the growth. Given the digitalization globally combined with the growth in smart devices is driving the market. Additionally, the gesture-based user interface has lower technical complexity, leading to its wider adaptability. With all these positive benefits, both for users and commercially, it is vital to (1) build a robust gesture recognition system. (2) fills the gaps between current gesture recognition systems, (3) design a system that will be commercially viable for the majority of people across the globe, and (4) it should have a modular design that should easily integrate with other technologies.

This dissertation presents a few methods for hand keypoints detection which is essential for hand gesture recognition. The objective is to design algorithms that will overcome some of the limitations present in the current algorithm. The methods presented in this dissertation use machine-learning techniques to create models that facilitate hand keypoints detection. The basic details of hand keypoint detection and machine learning are provided in the subsequent sections.

1.1 Hand Pose Estimation

Pose estimation is a CV task that involves the detection, association, and tracking of predefined keypoints from an image or a sequence of video frames [6,7]. These keypoints are chosen in such a way that they define the pose of the subject in consideration. The subject may be a human or some specific body part of it or it may be an animal. If the subject is human, then the task is known as human pose estimation and the keypoints are right and left shoulders, right and left knees, head, etc.

Hand pose estimation is a process of locating the positions of hand joints and fingertips. Commonly hand pose is defined by 21 keypoints in human hands; wrist point (the point at the base of the palm), five fingertips, and three finger joints for each of the five fingers. These 21 hand keypoints used in hand pose estimation are shown in Figure 1.4b. Each unique hand pose has a unique position of these keypoints in the three-dimensional space. By determining these positions, the information conveyed through the hand pose can be predicted.

1.1.1 Importance of Hand Pose Estimation

Traditionally, object detection processes only provide a bounding box around the object of interest and it does not give a lot of details that can be utilized efficiently in an application. In order for a machine to understand human action, it requires collecting fine-grained information about the action-taking objects. This gives the machine a more natural understanding of human behaviors. For example, in order to extract information from sign language it is required to understand the position and motion of a finger and its joints, and hence hand pose estimation becomes important. A human can convey information through gestures made through eye gaze, head motions, facial expressions, hand or arms, or motions involving the whole body. Nevertheless, the information conveyed through the hands has been the most effective. The reason is its dexterous functionality in manipulation and communication. In day-to-day communications, hand gesticulation has always accompanied speech. As making a hand gesture

comes naturally and is intuitive to humans, using it over traditional control devices like a keyboard, mouse, or joystick can play a crucial role in the advancement of HCI. A few of the examples where hand pose estimation technology can be employed are shown in Fig 1.3.

Considering comfort and safety, human hands can act as touchless control or interaction tools. Object manipulation tasks like navigation, selection, and manipulation of objects in virtual or augmented environments can be performed intuitively and efficiently with hands [6]. In the health-care sector like during surgical operations, minimum touch is preferential to avoid contamination through known or unknown sources in the surroundings. Operating tools or devices with touchless gestures become very important in such circumstances. Using hand gestures can be one possible way that can be employed in such scenarios. For a person with hearing impairments, the hands are the primary tools to communicate. They form different signs through their hands to convey the information around them. A gesture recognition interface that can translate these sign languages to speech or other easily understood will alleviate the communication challenges faced by persons with these impairments [8, 9]. The gaming [10], entertainment, and robotic industries are some of the avenues where hand pose estimation can have numerous applications.

1.1.2 Types and Approaches

The hand pose estimation has been generally carried out in two dimensions or three dimensions and therefore based on the type of dimensional used it is referred to as $2D$ hand pose estimation or $3D$ hand pose estimation, respectively. The $2D$ hand pose estimation is the process of estimating the $2D$ position or spatial location of hand keypoints from the input image or video frames. The $2D$ position of a keypoint is represented as (x, y), where x and y values are in terms of image coordinates having units in pixels. Their values range from $[0, w]$ where w is the image edge's length or [0,1] if normalized by dividing it by w. $3D$ hand pose estimation is used to determine

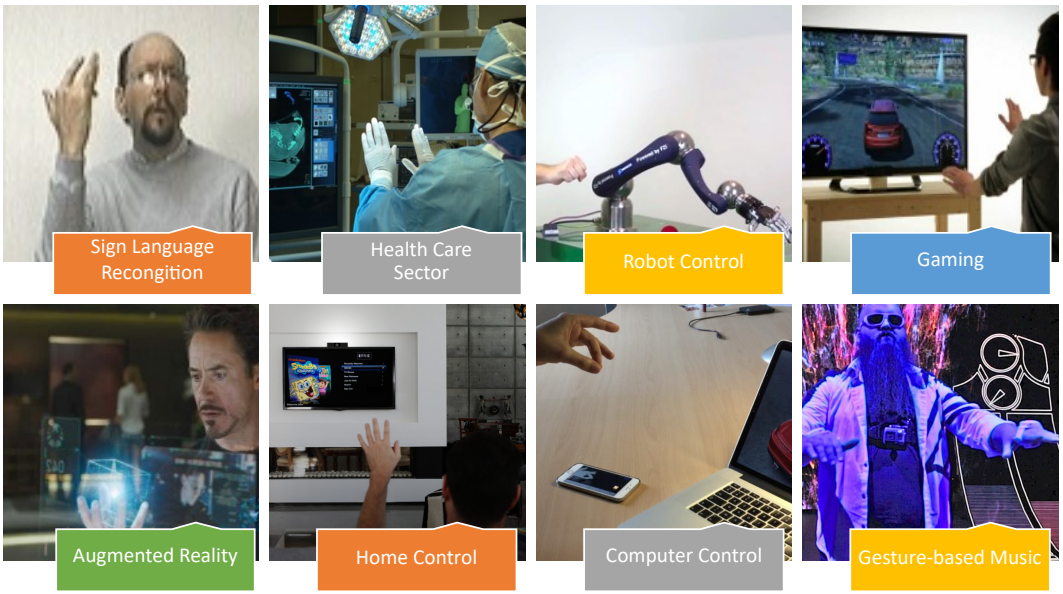


Figure 1.3: Some examples for hand pose applications area.

the locations of hand joints' position in 3D space. Here, along with the spatial location of hand keypoints their depth is also measured. Each predicted hand keypoint position can be either expressed in the camera coordinate or the world coordinate system. In the $3D$ pose of a hand, we get the joint angles and the orientation of the hand. A hand can look very different depending on the point of view in the $3D$ space. Both $2D$ and $3D$ hand pose estimation are important from the point of specific applications. The estimation process of these two can either have some common steps or all the steps involved vary drastically from each other. These estimation processes mainly depend on the input image form, availability of computational resources, application requirements, etc.

All major approaches for hand pose estimation can be grouped into two categories. They are

- *Top-down approach* - It is a two-step process where in the first step a detector first locates the region of interest (ROI) (hand in cases hand pose estimation) from the input source. In the subsequent step, the keypoints are estimated using the detected ROI.

- *Bottom-up approach* - The joint detection is performed first and then they are grouped to form a unique pose.

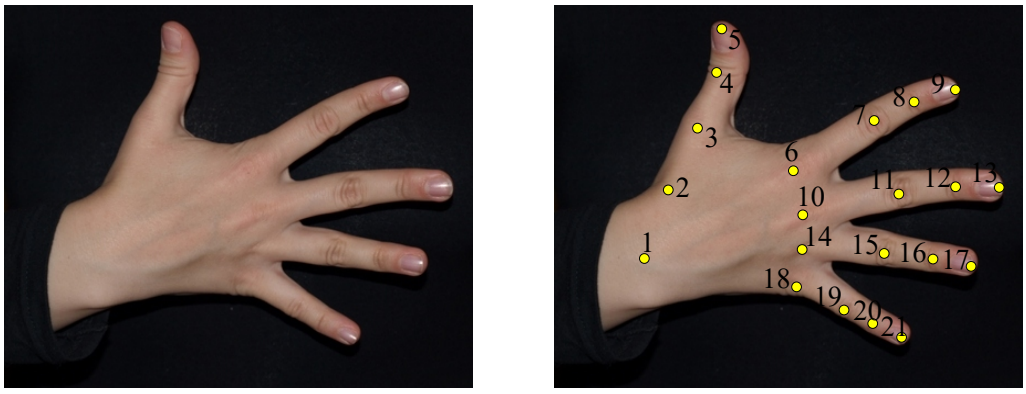
The input source for a hand pose estimation (or any other pose estimation process) is an image or video. The most common input sources are

- RGB sensor
- RGBD sensor (D represents the depth value)
- Depth sensor
- Stereo sensor.

RGB represents Red, Green, and Blue channel images which are the most common form of the imaging system and are extensively used in our day-to-day life. The RGBD image format includes a separate depth channel along with R, G, and B channels. The D channel gives the measure of the distance of the subject from the imaging sensor. The depth values can also be independently obtained (i.e., without RGB channels) from specific sensors like Microsoft Kinect. If an image is obtained only by using a single sensor then it is referred to as a monocular image. If a system of two sensors placed at some distance apart from each other is used for imaging then the system is referred to as a stereo-sensing system. Multi-sensor systems can also be used for pose estimation. One of the important benefits of the stereo and multi-sensor system is that obtaining 3D information about a subject is easier in comparison to the monocular system. However, these types of systems are costly and very complex to handle. They require specific software and a controlled environment to operate and therefore they cannot be incorporated into many of the applications.

1.1.3 Challenges

Hand pose estimation has numerous beneficial applications; still, a general solution is not available that can work for all given scenarios. Several challenging factors have to be tackled to produce a working solution for hand pose estimation. Some of them have been briefly discussed here. Many of the challenges are with hand keypoints motions



(a) (b)

Figure 1.4: A human hand and its 21 keypoint (or joints).

and tracking systems. The most accurate way to locate hand keypoints and track their motions is through wearable devices like data gloves [6,11]. However, they are not very convenient for users. They can restrict hand motions and hence limit the number of possible gestures. Also, they need proper calibration before every use leading to more inconvenience to users. Vision-based solutions can overcome some of these disadvantages that are with data gloves. But, they a solution based on vision-based technologies are faced with some other forms of difficulties. For example, one of the best possible ways to detect and track hand joints' motion is to put some machine recognizable marker on them. Since a machine is already familiar with the features of these markers, it can easily locate them in the image. However, putting these markers requires some training and cannot be readily used in all use cases. It will also be inconvenient to put them for any of the users. There are many research works available that intend to perform marker-less hand pose estimation or gesture recognition. Research work that uses an image for this purpose should be able to locate the ROI (human hands) in the frame. In many cases, as the human hand occupies only a fraction of the image area, this task becomes challenging. Additionally, the hand pose estimation system accuracy becomes dependent on the hand detection system. If either the ROI detection system failed or the output contains only a partial hand region then the location of all joints

involved in making a hand pose will not be correctly detected. The output resolution of the image-sensing device also plays an important role here. Detecting or tracking hand joints in low-resolution images is a difficult task. The operating environment of an image sensor also needs to be considered before using it in any specific application. For example, the depth sensor (some of which are also known as Time-of-Flight (ToF) light sensors) operates in a limited range [12–14]. Some of the commercially available depth sensors have an operating range limited to 3 – 5 meters only. And not many of them can work in the outside environment. The depth map (output image) resolutions are low and they suffer from noise. Alternatively, the RGB image sensor can be used to capture images and currently, high-resolution RGB cameras are very available in the market and at very low costs. However, RGB images do not preserve the depth information of the object captured, and therefore to use them in 3D pose estimation tasks requires some additional setup or processing.

Some of the challenges arise due to hand anatomy. A human hand has a high degree of freedom (DOF) [11] leading to a large number of hand motions. This high DOF of the hand creates the problem of self-occlusion of joints in the monocular view [15]. Objects in the image can also cause occlusion of joints. Apart from all these challenges, the background contents of the image and the illumination conditions can interfere with the keypoints detection process.

Another critical factor to keep in mind while working to create a solution for pose estimation is the available datasets required for features engineering or model training. Nowadays, a lot of computer vision-based solutions for pose estimation use neural networks (NN). Furthermore, training a model based on neural network architecture requires a lot of training and testing data (mainly in the case of the supervised form of model training). The process of collecting images and labeling the joints for pose estimation is not an easy task. Outsourcing this task is also costly. However, the field of hand pose estimation has recently received much attention from people in the research

community and it has led to the creation of many datasets that are publicly available. A description of some of these datasets is provided in the following subsection.

1.1.4 Datasets

Most of the research work or computer vision applications are based on machine learning (ML) technology. Data mainly drives the ML models or algorithms. They require data to create a model with high fidelity. Having or finding a quality dataset in sufficient quantity is the primary requirement of any application that utilizes AI as its backbone. For example, for the task of hand pose estimation, it is required to have an image with all 21 hand joints properly labeled. However, creating such a quality with a large number of image samples is a daunting task. In such scenarios, finding and selecting datasets available publicly becomes essential. A list of a few of the datasets available publicly or on request has been presented in Table 1.1.

The datasets for hand pose estimation can be categorized into many categories based on certain criteria. Based on the image type, a dataset may contain samples for real-world (R), computer-generated, or synthetic (S). The datasets can be classified on the basis of sensors used to collect samples, i.e., they may contain samples only from monocular RGB sensors stereo RGB sensors, or RGBD sensors, or can be a combination of all. The point of view (POV) from which image samples are captured decides the application for which it can be used. Generally, samples in POV of the third person and egocentric are available. Hence, a proper dataset is to be chosen based on the objective of the application.

1.2 Artificial Intelligence and Deep Learning

The terminology Artificial Intelligence (AI) refers to the simulation of human-like intelligence in machines. They are programmed to mimic the behavior associated with the human mind such as learning and problem-solving. This concept is based on the principle that a machine can have intelligence capabilities similar to humans where they can be made to learn, reason, and perform actions. Each specific action taken by these

Table 1.1: **A list of publicly available datasets for hand keypoints detection**

Dataset	Year	Type	Joints	POV	Samples
HGR [16]	2013	R	25	3rd	1.5K
MHP [17]	2017	R	21	3rd	80K
CMU Panoptic Hand [18]	2017	S/R	21	3rd	14,817
FreiHand [19]	2019	R	21	3rd	130K
RHD [20]	2017	S	21	3rd	41K
STB [21]	2017	R	21	3rd	18K
SCUT-EgoFinger-Dataset [12]	2018	R	5	Ego	35K
FPHA [22]	2018	R	21	Ego	100K
OneHand 10K [13]	2019	R	21	3rd	10K
InterHand2.6M [23]	2020	R	21	3rd	2.6M
Ego3DHands [24]	2020	S	21	Ego	50K
ContactPose [25]	2020	R	21	3rd	2.9M
HIU-DMTL-Data [26]	2021	R	21	3rd/Ego	40K
MVHand [27]	2021	R	21	3rd	83K

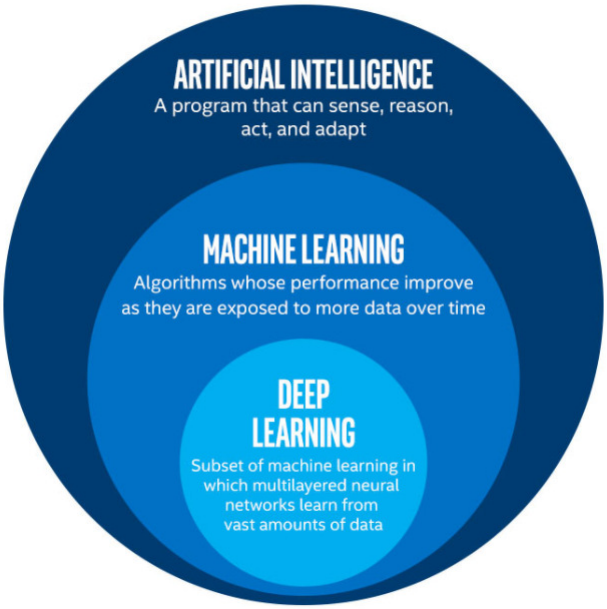


Figure 1.5: Sub-fields of Artificial Intelligence [1].

machines need not be explicitly defined or programmed and the tasks executed by these machines can vary from the simplest to more complex.

AI is continuously evolving to benefit many different applications and industries. The

self-driving cars can be an example of machines with artificial intelligence. A self-driving car has multiple sensors connected to it that scan and monitor its surroundings. The brain of the car receives these signals and weighs them for the consequences of any action if taken, as each action will impact the end result. Here, the best intended result is to drive in a way that prevents a collision.
AI is a broader term to define machine intelligence. It has a sub-field called Machine Learning (ML) and Deep Learning (DL) is the subset of ML. The Figure 1.5 illustrates the connection between AI, ML, and Deep Learning.

Stanford University defines the term ML as the science of getting computers to perform an action without being explicitly programmed. The machine learning algorithm is data-driven and it uses statistical methods on historical data to predict new output values. Deep learning is a sub-category of machine learning where algorithms are designed inspired by the structure and functions of human brain cells (neurons) and are referred to as artificial neural networks (ANN). A brief detail about deep learning is presented in the following subsection.

1.2.1 Deep Learning

Deep learning is a type of artificial intelligence where the learning system of a machine is analogous to a human way of gaining a certain type of knowledge. The learning of the machine here is very much data-driven where its prediction or action-taking behavior is modeled on data statistics and stacked levels of feature abstraction from the input.

In order to make an analogy of deep learning form of AI with human learning ways can be through an example of a toddler who is trying to differentiate between dogs and cats. First, the child starts by pointing to an object and making a guess about whether it is a dog or a cat, or some other entity. The guess made by the child is confirmed by the parents if it is right or corrected in case the made guess is wrong. The child becomes more aware of the features of cats and dogs through a repeated cycle of

guessing and correction. An illustration of this learning process is shown in Figure 1.6a. From left to right, the figure depicts the growth of knowledge and experience from a novice child to an expert adult. The concept depicted in the figure is a process of building a hierarchy in which at each age, the level of abstraction is created with the knowledge that was gained from previous experience. The same analogous process is being followed in training a machine to be designed to perform a specific task. It is trained with data collected over time and the process is repeated for a number of cycles till the learning of the machine reaches an optimum level. Then it is put to perform the said task for which it is designed and this time its inputs are real-world unseen pieces of information. The process is depicted in Figure 1.6b with the help of block diagrams.

The same process of hierarchical learning as used by a child applies to computer programs that use deep learning. The algorithm in hierarchy applies a non-linear transformation to its input and uses what it has learned so far to create a statistical model as output. The term deep in deep learning is derived from the large number of processing layers through which data passes before reaching the output.

The processing layers in a deep learning algorithm are traditionally a group of artificial neurons. A network formed by hierarchical connections of these neurons is referred to as an artificial neural network (ANN) [28]. Hence, sometimes deep neural networks (DNN) interchangeably to refer DL. The majority of currently available ANN architectures have been developed from Multi-Layered Perceptron (MLP) design. The basic building block of an MLP is a neuron (also called a perceptron [29, 30]). It is a mathematical function that receives information from one or more interconnected nodes and classifies it according to its architecture. An example of neuron architecture is shown in Figure 1.7. It receives information from three different nodes. It then weighs each of these inputs and takes a sum of it. A bias is also added to this neuron which acts as a threshold and helps in the decision-making process of the neuron. The summed value is then passed through an activation function (generally a non-linear

function). The non-linear output either can be passed to]hlthe subsequently connected neurons in the network or it may be the final output of ANN.

In an MLP architecture, the perceptrons are arranged in interconnected layers and are sometimes referred to as a fully connected network (FCN). Each perceptron in a given layer receives input from all the perceptrons in the previous layer (except the perceptrons in the input layer) and the generated output is passed to all the perceptrons in the next layer (except the perceptrons in the output layer). The perceptrons or neurons in input layers collect input features or patterns. The hidden layers transform the received input more usable with the objective that the neural network's output error will be minimal. The hidden layer basically extracts the most relevant features from its input that best describe the features of input data. Finally, the output layer in MLP produces some classification results or signals that map to the given inputs. A simple feed-forward neural network architecture is shown in Figure 1.8. It has an input layer, an output layer, and three hidden layers. There are three perceptrons in the input layer shown in orange while only one neuron in the output layer shown in red. The hidden layers are shown in green. These layers are hidden because their inputs or outputs cannot be directly determined or accessed.

There are many different feed-forward neural network architectures each with their own benefits and applications. However, MLP architectures are not preferred for image-based applications due to their certain disadvantages. Some important ones are given below.

- The FFNN takes a vector of inputs, so a 2D array of image pixels cannot be directly used for its input. The 2D array first has to be flattened into a vector. For example, we are creating a number classifier system with FFNN and we are using inputs as shown in Figure 1.9a. The Figure 1.9a is 4×4 pixel image array of digit two. In order to pass this image to a classifier architecture like the one shown in Figure 1.8 we need to transform it into a 1-D array as shown in Figure 1.9b.

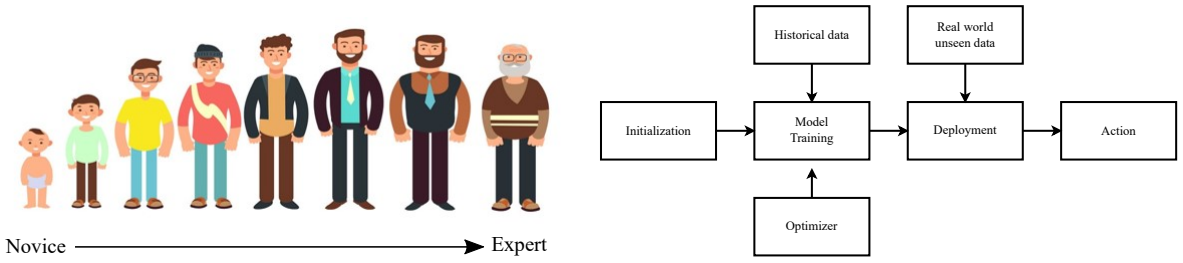
However, by doing so we lose the information on the spatial relationship between the pixels. For example, the relationship that may exist between the pixels in an image, such as pixel arrangements into the image corners, the presence of edge segments, and other features that can help differentiate one image from another.

- The computational complexity of a neural network is dependent on the input feature vector dimension. The higher the dimension higher the computational cost. Therefore, if we use a very high-resolution image as input to a neural network the computational cost will be too high, and hence the processing time will increase correspondingly making it suitable for many applications.

Many of the disadvantages associated with an FFNN with an image as input are handled well by Convolutional Neural Networks (CNNs) [31]. The architecture of CNNs (also called ConvNets) makes it suitable for image processing and CV applications. The basics of CNN are discussed next.

1.2.2 Convolution Neural Network

A convolutional neural network is a form of feed-forward neural network that accepts images as input and is suited well for image-based applications such as object classification. Operationally, there is some similarity between a fully connected neural network (FCN) and CNN. The computations performed in both forms of NN are very similar; (1) a sum of products, (2) adding bias to it, (3) passing resulting value to an



(a) Growth of child from a novice to an expert

(b) A machine initialization to deployment process

Figure 1.6: Making an analogy between human and machine ways of gaining knowledge and performing intended tasks.

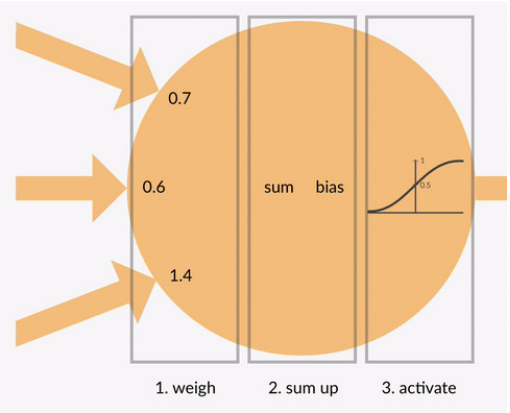


Figure 1.7: The basic working principle of an artificial neuron.

activation function, (4) The output of the activation is the input for the following layer. Despite, the computational similarity between a CNN and an FCN, there are many differences between these two. One difference is that the 2-D arras from layer to the next are subsampled to reduce sensitivity to translation variations in the input.

The working principle of CNN is based on neighborhood processing. The operation performed in the neighborhood is known as spatial convolution and the neighborhood is itself called the receptive field. A receptive field is a group of adjacent pixels in the input image. A filter slides across the image and a set of values weighs the pixels underneath. These sets of weights are called kernels. The convolution operation performs the sum of products between pixels and kernel weights. The result is a scalar value to which a

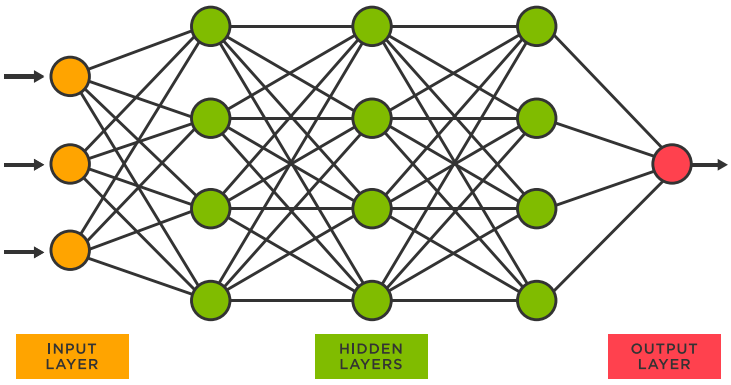
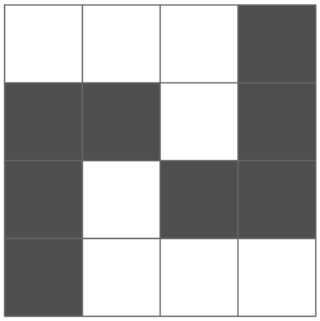


Figure 1.8: A simple neural network architecture.



(a) Digit two image

(b) Digit two image in flattened form

Figure 1.9: Representation of digital image in two different forms.

bias is then added and passed through an activation function. The convolution when repeated at every pixel location of the input image results in another 2-D array, called a feature map. The terminology feature map is derived from filtering in image processing where the use of appropriate kernel results in extracted features like edges, points, or blobs. A well-known CNN architecture is shown in Figure 1.10. The shown architecture known as *LeNet* [2] was designed for character recognition tasks like reading zip codes and digits. The LeNet architecture surpassed the accuracy of image classification by a considerable margin compared to other classification methods available at that time. In Figure 1.10 three sets of feature maps are shown. Each feature map in a forward direction of the network has a lower spatial resolution in comparison to the previous one. This is achieved through a sub-sampling operation known as pooling. At the end of the network, the last feature map is used as input to a fully connected layer. The final output is a vector that contains the classification result for the given input.

1.2.3 Applications of CNN

CNN has emerged as an excellent tool for solving vision-related problems. Image recognition and classification which were once believed to be complex problems to solve,

are now easier become easier with CNN models. The accuracy level of CNN models has surpassed humans in many object classification challenges. Given the many advantages of CNN, it has been employed in many applications. A few of them have been shown in Figure 1.11. Some notable applications of CNN are

- *Health care risk assessment* - Locating tumorous regions in digitized tissue and identifying whether they are malignant or benign.
- *Surveillance* - Recognizing single or multiple faces for surveillance purposes or monitoring human behavior under certain circumstances.
- *Visual Search* - Identify the shown object and then search it over a database. Beneficial in product searching.
- *Self-driving car* - Visually monitoring and identifying objects in the surroundings to facilitate safe driving conditions.
- *Defects detection* - Analyzing products to detect microscopic defects will be beneficial in many industries such as semiconductors where avoiding even a minor defect in a product is important.
- *Disable friendly interface* - To enable visually impaired persons easier access to computing devices, control interfaces that can work based on visual cues such as sign language become important.
- *Image restoration* - Refining the quality of old degraded images, even increasing the resolution of digital images is important in preserving historical data and

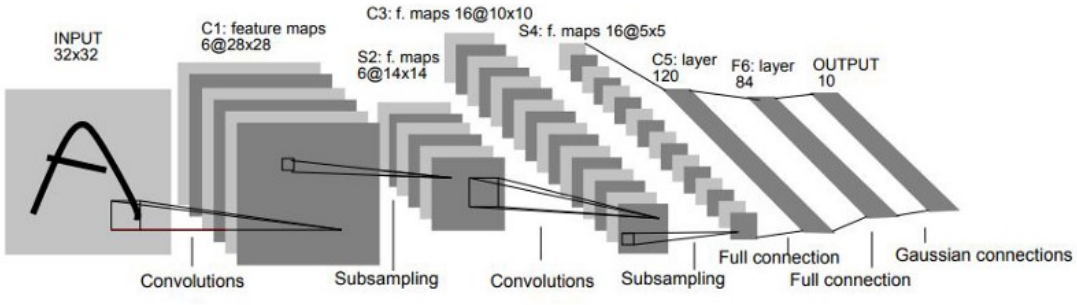


Figure 1.10: A classic CNN architecture called LeNet [2] .

knowledge.

- *Image synthesis* - A task of generating a targeted image using an existing image or producing entirely new images.
- *Image Captioning* - Generating a textual description of an image by analyzing its object and context

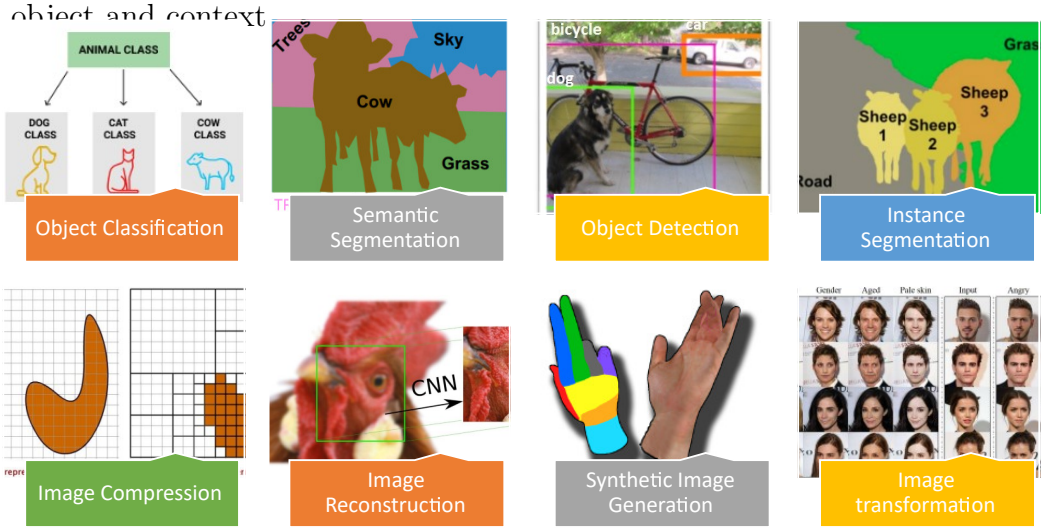


Figure 1.11: A few examples for CNN application areas.

1.2.4 Advantages and Disadvantages

Although a CNN-based model has many applications given its many architectural advantages, there are certain drawbacks associated with it too, which one should be aware of. A few of the benefits and drawbacks of CNN are presented in Table 1.2.

1.2.5 Hyper-parameter Tuning

The term 'parameter' in DL refers to the weights of the kernel of a layer in the network, which is fine-tuned during the model training process to attain an optimum value. The term hyper-parameter is used to refer to the variables that define a DNN architecture. Some examples of hyper-parameters would be the number of layers, dropout probability, batch size, number of epochs, momentum, learning rate, etc. Among all these hyperparameters, the learning rate is a significant variable whose value controls the speed of model training.

During the training process, that is performing the model's weight optimization, an

Table 1.2: **Advantages and disadvantages of a CNN model**

Advantages	Disadvantages
Accepts images as inputs and thus the spatial relationships between pixels are maintained	Requires a lot of labeled data to train
Capable of learning features automatically from raw data without any human supervision	A CNN model requires machines with high computational powers. Very few models can work on CPUs with adequate accuracy.
High accuracy at image recognition	The position and orientation of an object are not encoded by CNN
Weight Sharing resulting in lesser trainable parameters	It lacks the ability to be spatially invariant to the input data
The same learned knowledge is used across all image locations	A CNN performance can be affected by some noise. The phenomenon is called the adversarial effect.
Some CNN architecture works independently of input image size.	It does not keep track of coordinate frames.
Transfer learning capability- A model designed for one application can be used for another with fine-tuning.	The training process takes a long time.

optimization algorithm like Stochastic Gradient Descent estimates the error gradient from an error function. It then updates these weights during the error propagation process known as backpropagation. Through the backpropagation process, the objective is to minimize the model's error value to such a point where it is the lowest point in the loss curve. During the model's weights optimization, the loss moves through a loss function, an N-dimensional surface consisting of many hills, cliffs, local optima, saddle points, etc. An example of the loss function limited only to 3 dimensions is shown in Figure 1.12. As aforementioned, the target during model training is to reach the point of global minima of the loss curve. During each iteration of the training, the model weights are updated as governed by the equation (1.1).

$$w^{t+1} = w^t - \eta \frac{\partial L}{\partial w}, \tag{1.1}$$

where L is the loss function and η is the learning rate (LR). The LR is a positive scale

factor that controls the speeds of the weight update process. If the value of η is too low, the model will converge very slowly. There is also a possibility that the loss gradient gets stuck at the local minima, and optimum model weights cannot be obtained. If the value of η is set to a very high value, the rate at which the weight w update in (1.1) will be high which can lead to the model diverging instead of converging. It is always desirable to keep the learning rate value at such a rate that will cause the model to converge at a higher rate. Another challenge comes from the presence of saddle points in the loss curve. A simple example of a saddle point is shown in Figure 1.13. Saddle points in a multivariable function are such critical points where the function attains neither a local maximum value nor a local minimum value but the gradient of the loss function is zero. This results in the weight not being updated and the training not progressing.

There are several strategies to find the optimum learning rate value and let the model converge without being stuck at local minima or saddle points. L.N. Smith [32] proposed using a cyclic learning rate (CLR) policy where the learning rate is kept varying between a minimum and maximum value during complete training iteration. Another similar

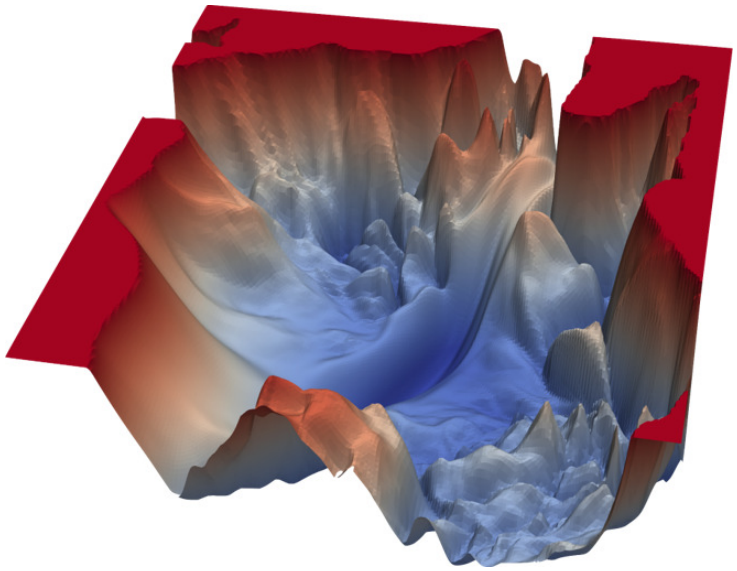


Figure 1.12: A representation of loss function in three dimensions.

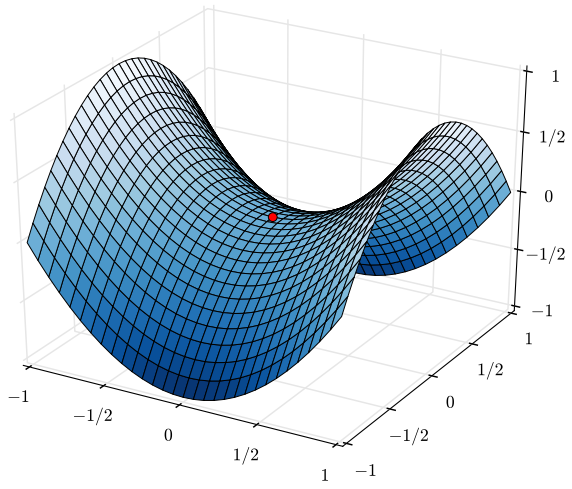


Figure 1.13: An example of a saddle point.

strategy called Stochastic Gradient Descent with Warm Restart (SGDR) [33] uses a cosine function for learning rate annealing. The learning rate is decayed at each iteration of training, and the increased again to the initial learning rate value is after completing a fixed number of iterations. Through both of these two aforementioned approaches, the model can be trained at a higher speed and model achieve higher performance values.

Polynomial Learning Rate with warm Restart. A different learning rate scheduling approach is presented in this dissertation. The learning rate scheduling approach uses a polynomial function as defined in (1.2).

$$lr = lr_0 * (1 - \frac{i}{T_i})^{power}. \tag{1.2}$$

Here, lr_0 is the initial or base learning rate, i denotes the number of iterations and T_i is the total number of iterations. In the case of polynomial learning rate policy, T_i is equal to the total number of epochs times the number of iterations in an epoch. Power term controls the shape of learning rate decay, as shown in Figure 1.14. The typical value of used power in (1.2) is 0.9 [34, 35].

The learning rate scheduling strategy using the polynomial function defined in eq consists of decaying the learning rate value from its initial point(lr0) to a minimum

point for a fraction of the total number of training iterations. The training is then warmly restarted and the learning rate is again started decaying for the remaining number of training iterations. The term warm restart here means that the training is not restarted from scratch instead it continues from the last learned weights. Only the value of the learning rate is increased at the point of the restart. There can be different variations to the strategy of the warm restart, four of which are shown in Figure 1.15. The warm restart can single as shown in Figure 1.15a or multiple Figure 1.15b - 1.15d. The advantage of these restarts is that the model achieves high accuracy faster in comparison to the polynomial learning rate policy. Moreover, the final accuracy is higher at the end of training.

The aforementioned statement is validated through testing this learning rate scheduling strategy on different datasets namely CIFAR-10 [36], CIFAR-100 [36] and tiny ImageNet [37]. Additionally, different CNN architectures are used to test the theory of the suggested polynomial learning rate warm restart strategy. The used CNN architectures are Wide Residual Network (WRN) [38], two different CNN architectures based on Residual Neural Network [39] form namely ResNet20 [40] and ResNet56 [40] where

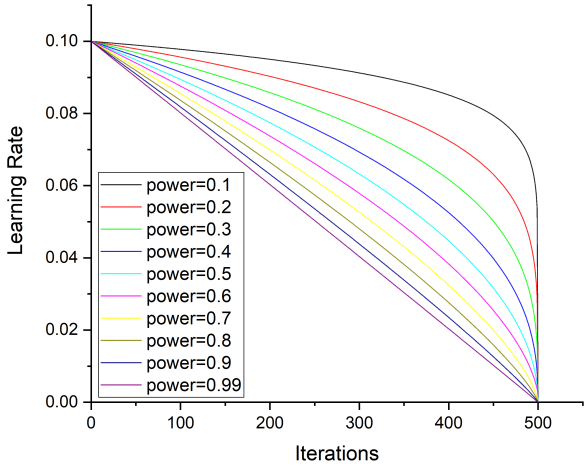
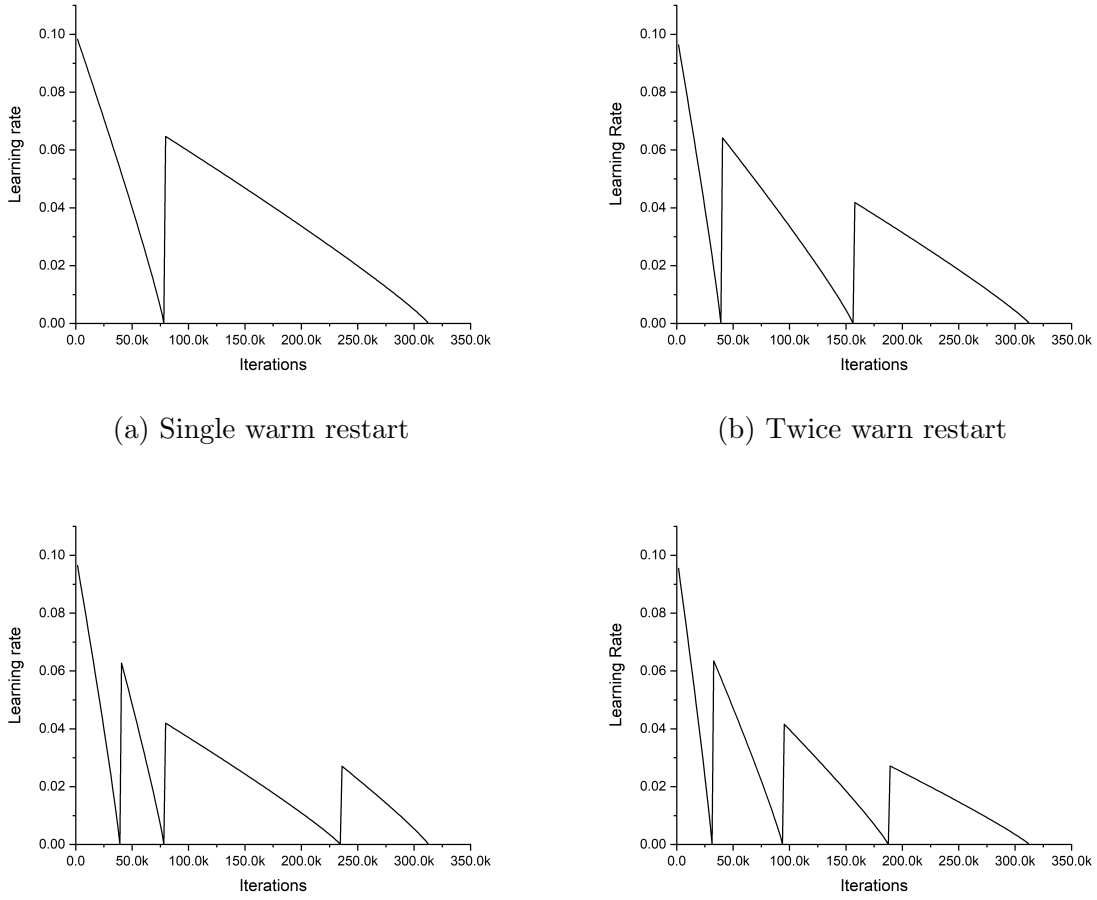


Figure 1.14: The effect of the value of the power on the form of the polynomial learning rate scheduling curve.



(a) Single warm restart

(b) Twice warn restart

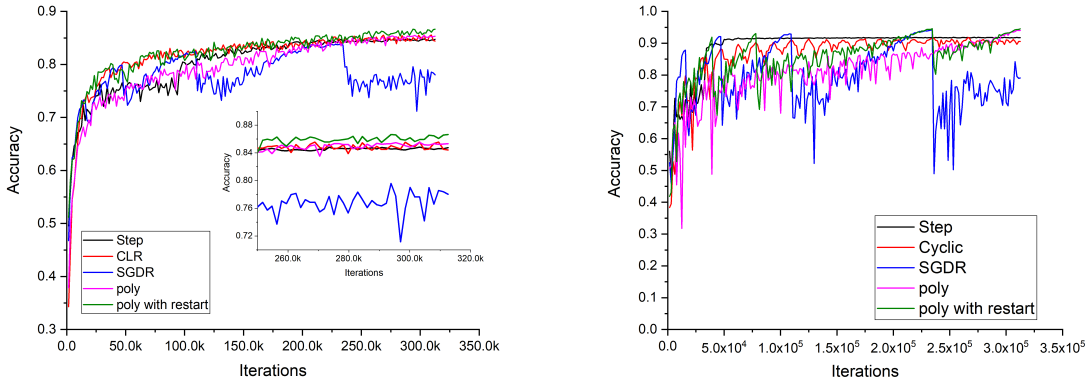
(c) Thrice warn restart

(d) Thrice warn restart at different fractions

Figure 1.15: Different warm restart strategies.

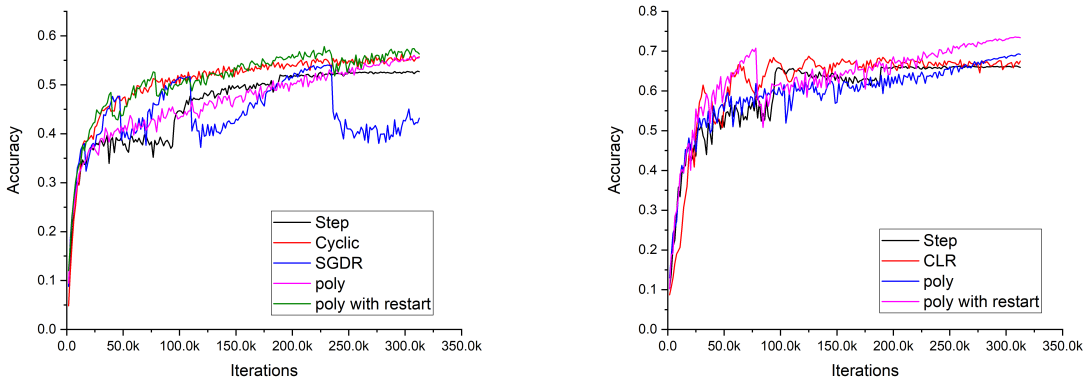
the number 20 and 56 in ResNet name represents the number of layers in the architecture. It was observed that the proposed learning rate scheduling strategy is providing higher accuracy and the model can also be trained at higher speed. The graphs are shown in Figure 1.16 and Figure 1.17 present the experimental results which affirm the above-stated statement.

The proposed polynomial learning rate scheduling policy with warm restart is compared to other available learning scheduling strategies like step, CLR, SGDR, and polynomial function-based method. The same CNN model architecture and datasets were used for comparison. The results of the comparison have been furnished in Table 1.3.



(a) CNN (b) ResNet56

Figure 1.16: Classification accuracy for different learning rate schedules on CIFAR-10 dataset.



(a) CNN (b) ResNet56

Figure 1.17: Classification accuracy for different learning rate schedules on CIFAR-100 dataset.

The polynomial LR scheduling strategy comes up as one of the best-performing strategies amongst the compared LR scheduling strategies.

The single warm restart strategy is well suited for LR scheduling where warm restart is to be at 1/4th of the total of iterations. Moreover, unlike the adaptive LR methods like Adam, there is no additional computational cost associated with the training process. However, it was also observed that the training of a neural network is dependent

Table 1.3: **Test accuracy of different learning rate policies on CIFAR-10, CIFAR-100, and tiny ImageNet datasets with moderate data augmentation**

		depth	params	LR	CIFAR-10	CIFAR-100	Tiny ImageNet
Step Decay [38]	CNN	6	1.25M	0.01	0.8482	0.5281	0.2578
	ResNet20	20	0.27M	0.01	0.9127	0.6586	0.4494
		56	0.86M	0.01	0.9239	0.6594	0.4893
	ResNet56	20	0.57M	0.1	0.9199	**0.7173**	0.4673
		56	1.67M	0.1	0.9443	0.7510	0.5199
	WRN	28-10	36.4M	0.1	0.9384	0.7589	0.5608
CLR [32] mode = triangular2	CNN	6	1.25M	0.01-0.001	0.8593	0.5727	**0.3304**
	ResNet20	20	0.27M	0.01-0.001	0.8948	0.6542	0.4389
		56	0.86M	0.01-0.001	0.9147	0.6872	0.4836
	ResNet56	20	0.57M	0.1-0.001	0.8957	0.6715	**0.4824**
		56	1.67M	0.1-0.001	0.9177	0.6802	**0.5356**
	WRN	28-10	36.4M	0.01-0.001	0.9398	0.7651	0.5721
SGDR [33] $T_0 = 10$, $T_{mult} = 2$	CNN	6	1.25M	0.01-0.001	0.8428	0.5403	0.2645
	ResNet20	20	0.27M	0.01-0.001	0.9174	0.6640	0.4386
		56	0.86M	0.01-0.001	0.9281	0.6945	0.4756
	ResNet56	20	0.57M	0.1-0.001	0.9171	0.6960	0.4582
		56	1.67M	0.1-0.001	0.9418	0.7476	0.5143
	WRN	28-10	36.4M	0.01-0.001	0.9359	0.7546	0.5619
poly power = 0.9	CNN	6	1.25M	0.01	0.8580	0.5605	0.2951
	ResNet20	20	0.27M	0.01	**0.9239**	0.6712	0.4384
		56	0.86M	0.01	**0.9296**	0.6931	0.4892
	ResNet56	20	0.57M	0.1	0.9219	0.7099	0.4725
		56	1.67M	0.1	0.9407	0.7477	0.5127
	WRN	28-10	36.4M	0.1	0.9442	0.7604	0.5662
poly with restart power = 0.9 fraction = 25% drop = 65% (proposed)	CNN	6	1.25M	0.01	**0.8661**	**0.5763**	0.3166
	ResNet20	20	0.27M	0.01	0.9137	**0.6947**	**0.4421**
		56	0.86M	0.01	0.9251	**0.7359**	**0.4967**
	ResNet56	20	0.57M	0.1	**0.9236**	0.7155	0.4803
		56	1.67M	0.1	**0.9457**	**0.7524**	0.5245
	WRN	28-10	36.4M	0.1	**0.9454**	**0.7639**	**0.5812**

on the model's architecture and the data distribution. A single LR scheduling policy may not be universally applicable to all DNN models or datasets.

The proper choice of learning rate scheduling strategy is important for hand key-points detection when the process is based on DL methods. Generally, the models used for the purpose mentioned above have comparatively more layers, and hence more parameters to be trained. Therefore, shortening the training time is important from a development and applications perspective.

1.3 Motivation

The aim of this dissertation is to propose and design algorithms for hand keypoints detection from RGB images using deep learning methods. The motivation for this work has been derived from several factors. The first and foremost influencing factor is the research work undertaken by one of the researchers in the research group of Dr. Kishor Sarawadekar in the Department of Electronics Engineering at the Indian Institute of Technology (BHU), Varanasi, India. The research aimed to develop an HCI system for a person with a vision disability [41]. The designed HCI interface uses hand gestures to form a unique dactylology developed mainly for visually impaired persons. The work presented in this dissertation is aimed to extend the previous work and overcome some of its limitations that it has. For example, the number of hand gestures to be used for communication was limited and designed for a table-top environment. The image-capture and detection interface required a fixed setup which is not portable. The hand gesture detection system worked only in a controlled environment.

The other motivating factor to undertake this work is the number of potential applications and commercial demand for hand pose estimation technologies in the market. A few of the applications for hand pose estimation have been described in Section 1.1.1.

The dissertation is aimed to answer a few questions affecting the hand keypoints detection process. Some of the questions are – How to perform hand keypoints detection from a monocular RGB image? Can hand keypoints detection be performed in a single-step method without hand location? If yes, how to design an algorithm for it that will work for real-time applications? Is the available hand pose estimation method work for single-hand or multi-hands? How to incorporate multi-hand pose estimation in a single-stage method. The subsequent chapters in this dissertation try to answer some of these questions and present some algorithms for hand keypoint detections.

1.4 Contribution of the Dissertation

The original contributions in the presented dissertation are listed below:

- Three separate algorithms were proposed for fingertip detection from the monocular image. The first two algorithms were dependent on hand localization and worked effectively in estimating the fingertips' position in a certain hand gesture. Another algorithm proposed that robustly estimates fingertips from a full-size image. The experimental results show that the algorithm works effectively in varying hand gesture scenarios. The accuracy of the model is comparatively better than the state-of-the-art method available for fingertips detection.
- Two algorithms for complete hand keypoints detection were proposed. The algorithms were designed to work in a single stage and are effective for multiple hand poses. An approach to improve the hand keypoints detection for small hands was also proposed.
- A new method for simultaneous estimation of hand keypoints from multiple hands is proposed. The methods are designed to work directly on full-size RGB images. The additional computation resources and time required during the hand localization (if a separate process is used) are saved. The experimental results prove that the algorithm effectively detects keypoints directly on multiple hands.

1.5 Organization of the Dissertation

The dissertation is organized as follows. Chapter 2 covers the processes of partial hand keypoints detection. It briefly provides the details of the work available for it and then highlights some of the limitations of those works. The chapter then provides the details and results of three different algorithms proposed to perform partial hand keypoints detection. Starting from a method based on a two-step process, the chapter provides details of the method independent of hand location for partial hand keypoints detection.

Chapter 3 provides brief details of the methods available for complete hand keypoints detection. It then covers the details of the algorithm dealing with first - single hand keypoints detection and then second – double hand keypoints detection. Both of these

algorithms were designed to work on monocular RGB images.

In Chapter 4, we present a method for hand keypoints detection for multiple hands in a single step. The basic working principle behind the selection of feature maps to perform multiple ROI selections is covered briefly. The chapter covers the cases of hand keypoints detection in the case of interacting hands.

The summary and conclusion of this dissertation are provided in Chapter 5.