University of Tartu

Faculty of Science and Technology

Institute of Technology

Zafarullah

Gaze Assisted Neural Network based Prediction of End-Point of Human Reaching Trajectories

Master's thesis (30 EAP)
Robotics and Computer Engineering

Supervisor:

Arun Kumar Singh

Tartu 2020

Abstract

One of the key problems in human-robot collaboration is the prediction of the end-point of the human-reaching trajectory. Such predictions are essential for example to predict the human-robot hand-over locations in a collaborative manufacturing task. This thesis proposes a data-driven approach for end-point prediction. The key intuition behind the thesis is that human gaze encodes a significant amount of information regarding the end-point the human is trying to reach. On this intuition, we built three neural network models that leverage gaze information and trajectory of the hand motion for high fidelity prediction of the end-point of the reaching trajectory. Furthermore, the thesis also contributes in developing a robust experimental set-up for data collection that uses concepts from robotics and computer vision for creating annotated examples of hand trajectory, gaze and trajectory end-point.

Keywords: Neural Network, Multi-Layer Perceptron, Human Gaze, Reaching Trajectory.

CERCS: T125 Automation, Robotics, Machine learning

Abstract in Estonian

Inimese ja roboti vahelise koostöö üks peamisi probleeme on inimese liigutuste trajektoori lõpp-punkti ennustamine. Sellised ennustused on hädavajalikud näiteks selleks, et ette arvata inimese ja roboti vahelist üleandmiskohta ühise tootmisülesande täitmisel. Antud lõputöö pakub välja andmepõhise lähenemisviisi lõpp-punkti ennustamiseks. Lõputöö alustalaks on see, et inimese pilk annab märkimisväärses koguses teavet selle kohta, mis liigutusi ta tegema hakkab ning kus on nende liigutuste lõpp-punkt. Selle põhjal lõime kolm närvivõrgu mudelit, mis kasutavad infot inimese pilgu ning käe liikumise trajektoori kohta, et ennustada haaramistrajektoori lõpp-punkt. Lisaks annab antud lõputöö oma panuse eksperimentaalse andmekogumise meetodi välja töötamisel, mis kasutab robootikas ja reaalnägemises varem kasutatud kontseptsioone, et luua näiteid käetrajektoori, pilgu ning trajektoori lõpp-punkti kohta.

Võtmesõnad: Närvivõrk, Mitmekihiline perceptroon, Inimese pilk, Trajektoori jõudmine

CERCS: T125 Automaatika, Robootika, Masinõpe

Contents

List of Figures

List of Tables

1 Introduction

Robots are gradually transitioning from isolated factory floors to settings where it needs to work alongside human and collaborate with it to perform a given task. Human-robot collaboration is one of the hotbeds of robotics research with potential application in manufacturing, healthcare, elderly care etc. One of the fundamental challenges in human-robot collaboration is predicting the possible behavior or intent of the human. A specific sub-problem in this challenge is to predict the human reaching trajectory and particularly its end-point. This forms an essential requirement in applications like collaborative manufacturing [1] where the human hands over tools to the robot or vice versa. Herein, the robot needs to predict the preferred hand-over location of the human. Similar prediction capabilities are also required when robots are used as assistants at homes, for elderly care etc.

1.1 Objectives and Contributions

The objective of this thesis is to develop neural network based models for predicting the end-point of the 3D human reaching trajectories. The thesis puts forward the following contribution towards this end.

- The underlying idea behind this thesis is that the human gaze encodes a significant amount of information about the end point that the human is aiming to reach. We present three different neural-network models to leverage this insight. Our first model is a multi-layer perceptron (MLP) which regresses between the pitch, roll and yaw angle of the gaze and the end-point of human reaching trajectory. Our third model is a neural network (MLP) that fuses the approximate end-point prediction resulting from the human-hand motion history with the gaze information. Thus, as an intermediate step towards the third model, we also built a second neural network (MLP) that predicts end-point by taking as input the time stamped sequence of hand position following the work of [2]. But in contrast to [2], we use neural network with MLP architecture rather than Long Short-Term Memory.

- We present an analysis of the gaze based models with the baseline model that predicts end-point based on just the sequence of hand-positions.

- Since the neural network models are trained in a supervised fashion, they need examples of human reaching trajectories and an accurate information of their corresponding end-points. However, accurately measuring the 3D position of human hand is a challenging

task in the absence of specialized sensors like imu-fitted data gloves. Thus, the second contribution of this thesis lies in development and validation of a novel data collection set-up that biases the human motion towards the end-effector of a robotic arm and subsequently uses off-the-shelf cameras and common computer vision techniques to accurately measure human hand trajectories.

1.2 Organization of Thesis

The thesis is organized as follows:

- **The current chapter:** This chapter introduces the thesis and motivation behind this work, and also discusses objectives and contribution for this work.
- **Background:** This chapter explains the concepts that are involved in this work.
- **Design:** This chapter discusses the experimental setup, data collection and implementation of machine learning models for the work.
- **Results:** The results of the models are compared and explained in this chapter.
- **Future work:** The future directions and possible improvements are explained in this chapter.
- **Conclusion**: This chapter concludes and discusses the outcomes of the work.

2 Background

This chapter explains the basic related concepts used in machine learning, gaze and trajectory detection and optimization. First, neural network and its various techniques are discussed to explain how a neural network works. Second, the gaze detection process is discussed. Third, AR tracking is introduced and then ROS is explained briefly.

2.1 Neural Networks

Neural networks, also called artificial neural networks, are biologically-inspired programming algorithms [3] that enable a machine to learn from the data under observation. A neural network works the way human brain operates to recognize the relationships in a data set [4]. A neural network (Figure 2.1) [5] is a combination of highly interconnected layers of neurons; also called perceptrons or nodes; [6] that serially transforms the data and generates an output or prediction. A neuron is a mathematical function with an input and a corresponding output.

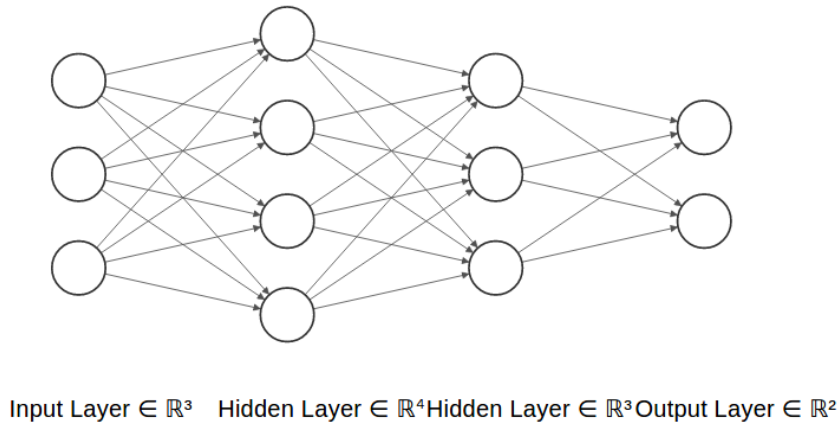

Figure 2.1: *A simple three layer Neural Network model*

A simple neural network model (Figure 2.1) consists of three types of layers:

- Input layer – It is used to pass input data to neural network. Input data can be textual, pictorial or audio.

- Hidden layers – The hidden layers are used for nonlinear transformation of the inputs [7].
- Output layer – The output layer provides the predicted output.

The neurons of a layer are all interconnected with neighbouring layer neurons and data transfers from input all the way to output layer through hidden layers. The predicted output of a neuron is mathematically expressed by the following equation:

$$Output = AF \times (W \times Input + B) \tag{2.1}$$

Where AF is activation function, W is the weight and B is the bias value. Activation functions are mathematical equations attached to each neuron to determine its output and hence the output of the network [8]. The weights play a vital role in learning a process as they contain the learned information and are therefore optimized through a back-propagation process. Each weight contributes to neural network's net output error. Back-propagation is the process of going back through layers in each iteration and adjust the weights to minimize that error.

2.1.1 Neural Network Parameters

The design and performance of a neural network varies with the type of input data and certain parameters. Some of the important parameters that play a key role in a network's performance are as follow:

- The number of hidden layers – The number of hidden layers highly depends on the type of data. If the data is linear or easily separable, very few or no hidden layers are needed at all. The more the data is complex in terms of features, the more hidden layers are required.
- Loss function – Loss or Cost function is the difference between predicted and actual output. It can also be seen as a measure of how well a neural network did in predicting an output. The most commonly used cost functions are Mean Squared Error (MSE), Cross Entropy, Hinge Loss etc[9].
- Optimizer – Optimizers help to minimize the cost function by updating weight parameters[10]. Some of the most popular optimizers are Adaptive Moment Estimation (ADAM), Stochastic Gradient Descent (SGD), Adaptive Gradient Algorithm (Adagrad) etc.
- Learning rate – It is the amount by which the weights are updated in each iteration during the network training [11].
- Activation function – The mathematical functions attached to each neuron to determine and normalize the neuron output. Activation functions can be linear and nonlinear but the most commonly used activation functions are nonlinear. Non-linearity generalizes the modeling to adapt with the variety of data[12]. The common nonlinear activation functions are ReLU, tanh, and sigmoid (Figure 2.2) [13].

- Number of Epochs or iterations – The number of times a learning algorithm is repeated during the training process. One iteration means the flow of data throughout the network model and then coming back all the way to adjust the weights to minimize the error.

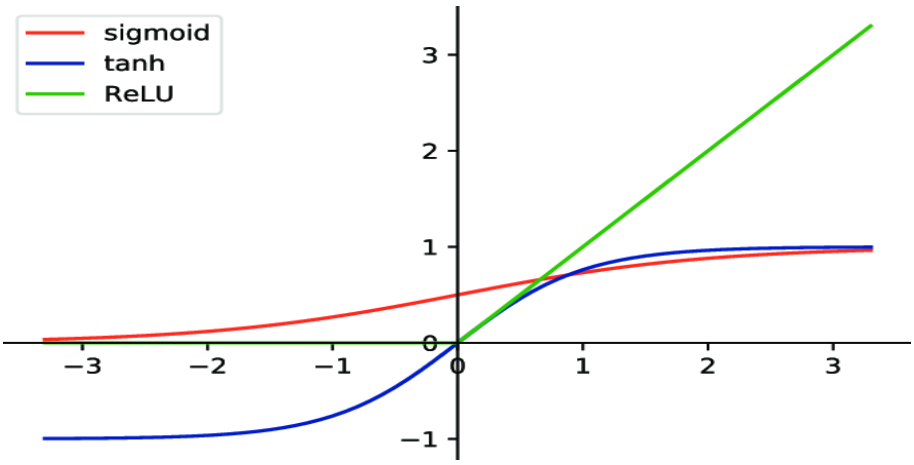

Figure 2.2: *Commonly used nonlinear activation functions*

2.1.2 Training and Validation Losses

A learning curve shows how a learning model performed with respect to time. The evaluation of a model on training and validation dataset is expressed in training and validation loss curves (Figure 2.3) [14].

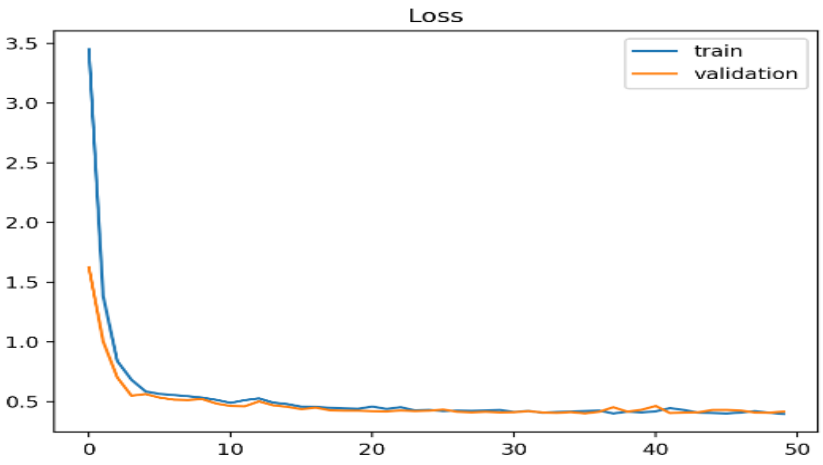

Figure 2.3: *Training and Validation loss curves showing a goodfit*

The curves show if a model is underfit, overfit or a goodfit. A network is underfit if validation loss is very higher than training loss whereas an overfit model is the one which has learned too well from dataset including noise. In a goodfit model, the training and validation losses come to a stable point and hence have a minimum difference between them.

2.1.3 Neural Network Types

There are different types of neural networks having their specifications and suitability for different applications. The most common or famous neural network examples are Multilayer Perceptron (MLP), Convolutional Neural Network (CNN), Recurrent Neural Network (RNN) and Long Short Term Memory (LSTM). MLP and CNN are briefly introduced here.

2.1.3.1 Multilayer Perceptron (MLP)

A multilayer perceptron model comprises of three or more fully connected layers (see Figure 2.1) which means every neuron in a layer is connected to every neuron of the neighbouring layer. MLP has input, output and one or more hidden layers. MLP has bi-directional propagation; forward and back propagation. MLP uses nonlinear activation functions to classify nonlinearly separable data.

2.1.3.2 Convolutional Neural Network (CNN)

Convolutional neural networks (Figure 2.4) [15], are widely used in the domains of speech recognition and computer vision. Unlike standard neural network models, CNN has three-dimensional arrangement of neurons and each neuron has a tiny part of visual information [16]. Convolution is a mathematical operation in which two functions are convoluted to produce a resultant function and it shows the impact of one function over other. Each convolution layer in CNN applies this operation on input and passes the result to next one [17]. CNN uses uni-directional propagation. In CNN, the visual data also goes through image processing techniques to get the final results.

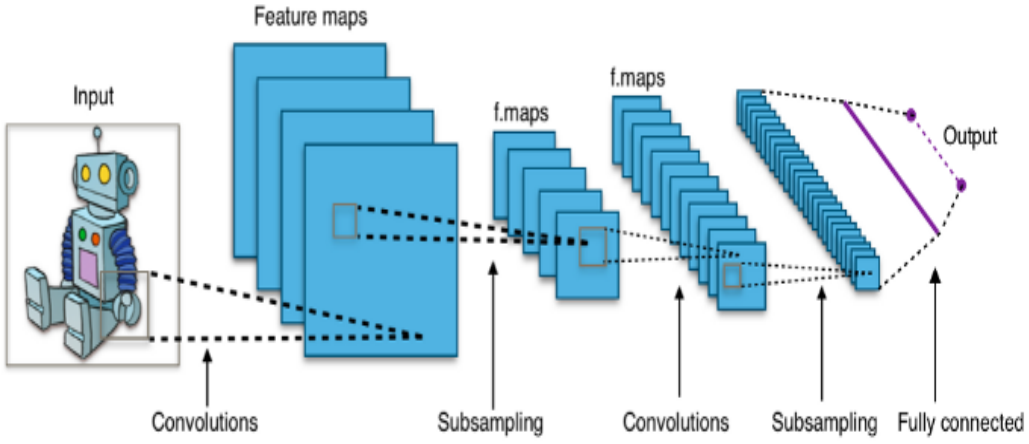

Figure 2.4: *Convolutional neural network architecture*

2.2 Robot Operating System (ROS)

Robot Operating System (ROS) is an open source Linux based framework that provides the services of an operating system for a robot [18]. The services of ROS include device control, message transfer between different processes, function implementation, and managing packages. ROS provides different communication styles to make different processes communicate with one another at run time. These processes, called nodes, are executable codes written for different tasks in a robot. The nodes communicate and share data with one another through a ROS master. The nodes either subscribe or publish to message containing topics depending on if they need data or they generate data. A node that needs some data subscribes to a related topic to fetch data from, whereas a node that generates data, it publishes messages to a topic. The ROS nodes run independently and do not affect the performance of other nodes at run time. This makes it possible to individually code different parts of a robot to solve a complete task and then combine them in the form of a package.

2.2.1 MoveIt Package

When it comes to move the end-effector, or gripper, of a robotic arm from one position to another position for any desired task such as gripping and picking an object, one needs to change all joint values properly and accurately to bring the end-effector from current position to the desired position. This task of motion planning has not been an easy one. MoveIt is a ROS framework that solves the problems of motion planning, grasping and obstacle avoidance for a robotic arm [19]. MoveIt provides a sequence of joint values to move the arm from current to the desired location. MoveIt can be used for both planning and executing a motion in a robot. MoveIt has different planners which are programming codes to generate a motion plan. MoveIt configuration process needs a URDF file that describes the robot. Obstacles can also be mentioned in URDF file so MoveIt can plan to avoid obstacle collision and with itself.

2.2.2 RViz

The plan and execution of robotic arm motion can be visualised using a ROS graphical interface, called RViz, which a powerful tool for 3D visualization. RViz uses plugins to visualize information for different available topics [20]. With the help of RViz plugins one can visualize the robot with respect to its position and orientation on floor plane or grid. RViz uses URDF file which describes a robot structure.

2.3 Head Pose Estimation and Deepgaze

Computer vision has been a very interesting topic among computer engineers and roboticists struggling to find out the ways to make machines recognize objects in a human environment. Image processing has played a key important role in solving many real-world scenarios such as face detection and recognition, and moving object detection and tracking etc. Head pose estimation is one of the similar problems which has applications in human robot interaction

and driving assistance [21]. Head pose estimation helps to find roll, pitch and yaw of head, or in simple words pose of the head.

Deepgaze [22] is an open source python library and a project for human-computer interaction, motion detection and motion tracking. Deepgaze is based on computer vision and machine learning libraries named OpenCV and Tensorflow. Deepgaze uses convolutional neural networks (CNN) and makes the life easier for applications like face detection and head pose estimation with few lines of code. Deepgaze also has some very useful packages for different applications such as face detection, head pose estimation, skin color detection, histogram-based classification, motion detection and motion tracking.

2.4 ARTag Tracking

As discussed in previous section, computer vision helps to detect and recognize different objects. ARTag (Figure 2.5) [23], is an augmented reality marker that can help to facilitate visualization of virtual objects. ROS has a very cool package named *ar_track_alvar* which allows to detect and recognize these AR tags with the use of a camera. The square shape of these tags is considered to be their main power in approximating their position and orientation in relation to camera frame based on image size. The library provides both position and orientation of AR tags which can be used for further applications. These tags can be put on a moving object to determine the motion of the moving object as long as its in the range of the camera.

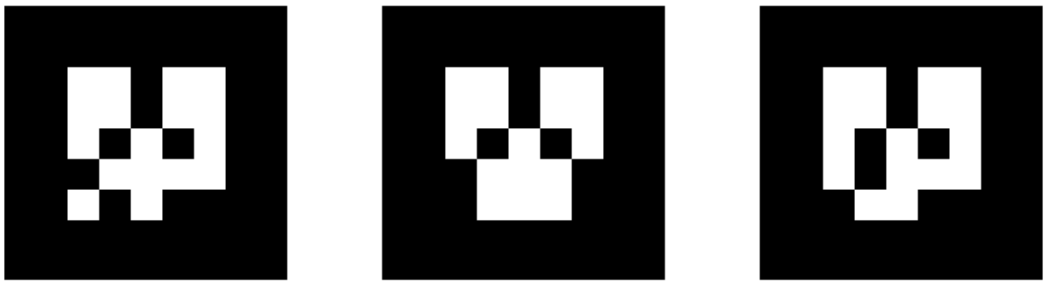

Figure 2.5: *Sample ARTags*

2.5 Existing Works on Reaching Trajectory and Hand-Over Point Prediction

The thesis aligns with two class of existing work pertaining to prediction of human-reaching trajectories and prediction of hand-over point in human-robot collaboration tasks. In the following, we review each of these class of works and contrast them with the proposed research.

2.5.1 Human-Reaching Trajectory Prediction

Analysis of human-reaching trajectories has been an important problem in both neuroscience and robotics community. Optimal control models presented in the former show that human reaching trajectories are characterized by straight line motions [24], [25]. Thus, the prediction of end-point of the trajectory becomes a simple problem. However, these models are based on simplifying assumptions about human motion and tasks and thus are not suitable for prediction in complex human-robot collaboration tasks. Thus, several machine learning based models are presented in literature. For example, [26] uses time-series classification methods for prediction of reaching motion. [27] solves a similar problem using a Gaussian Mixture Model based approach. [28] used an inverse optimal control/inverse reinforcement learning for reaching trajectory prediction. A Recurrent Neural Network based approach was presented in [29].

2.5.2 Hand-Over Point Prediction

The problem considered in this thesis is also closely related to problem of predicting the point where the robot is going to hand over an object to a human in a human-robot collaboration task. A recent work in this regard [30] presented a rigorous user study to elucidate the factors that affect human preferred choice of hand-over points. Factors such as distance between the robots and humans and the height of the human were considered important factors in deciding the hand-over point. They then presented an algorithm for predicting the hand-over point based on the hand motion trajectory and previously analyzed factors. Along similar lines, [31] built their hand-over predictor based on user-study but in this case, the study focused on human-human interaction and aimed to transfer the knowledge to human-robot case. [32] adopted a dynamical systems approach for predicting the hand-over point. The parameters of the dynamical system were again tuned based on user studies conducted for human to human hand-over. [33] presented an approach based on so-called probabilistic motion primitives that were learned from demonstration.

The work proposed in this thesis is different from the above cited works in the sense that we do not rely on any feature engineering from human-human or human-robot user studies. Instead, we adopt a purely end-to-end approach. Second, we are not aware of any work which fuses gaze and hand motion trajectory in the manner proposed in this thesis.

3 Requirements

3.1 Functional Requirements

- Building a neural network model that uses gaze parameters and predicts end-effector position.
- Building a neural network model that uses a complete trajectory and then predicts end-effector position.

3.2 Software Requirements

- Linux Ubuntu 16.04 to run ROS Kinetic or higher for later distributions.
- Python libraries, including PyTorch, Deepgaze, Tensorflow, Numpy, and matplot for plotting.
- Python and C++ languages for coding nodes for this work.

3.3 Hardware Requirements

- UR5 robotic arm.
- Two RGB cameras.

4 Methodology

This chapter explains the experimental setup, data collection and the development of different neural-network models for end-point prediction of human reaching trajectories.

4.1 Experimental Setup and Data Collection

The experimental setup is shown in Fig. 4.1 and comprises of a UR5 robotic arm and two RGB cameras named "camera-a" and "camera-b". To develop our neural network models, we need annotated data set consisting examples of human gaze and reaching trajectory for a given 3D point in space. There are two key challenges here. Firstly, how to infer accurately the spatial position of the hand in space. Second, how to bias the human gaze and motion to different spatial positions to ensure diversity in the recorded data. The key idea of the thesis to handle both these challenge involves the use of the UR5 robotic arm. The robotic arm is moved around in space and we can infer the 3D position of its end-effector with respect to the UR5 with high accuracy based on joint angles and forward kinematics relationship. The human is always asked to perform reaching motion towards the end-effector. Throughout the experiment, we observed that the human participant was able to perform the reaching motion towards the end-effector with reasonable accuracy and thus allowing us to indirectly infer the 3D hand-position at the end of the reaching trajectory.

In the experimental set-up, Camera-a is used for getting images of human face and then extract gaze information from those images whereas camera-b is focused on human hand and is used by AR tracker to get AR tag positions. Neural Network models then take the gaze parameters and trajectories as their inputs and end-effector positions as their respective labels for supervised learning.

4.1.1 Gaze Data

One of the primary goals of the research is to get the human gaze parameters, in other words where the human is looking. Camera-a, shown in Fig.4.1, is fixed at an appropriate height to get images of human face. As described in earlier chapter, Deepgaze library [22] is used to estimate the gaze. The said library has many functions used for solving different problems. One of the most important functions in Deepgaze is Head Pose Estimator which is the main function used in this experiment. The CNN based Head Pose Estimator gets images from camera-a, processes the image and returns calculated head pose parameters roll, pitch and yaw.

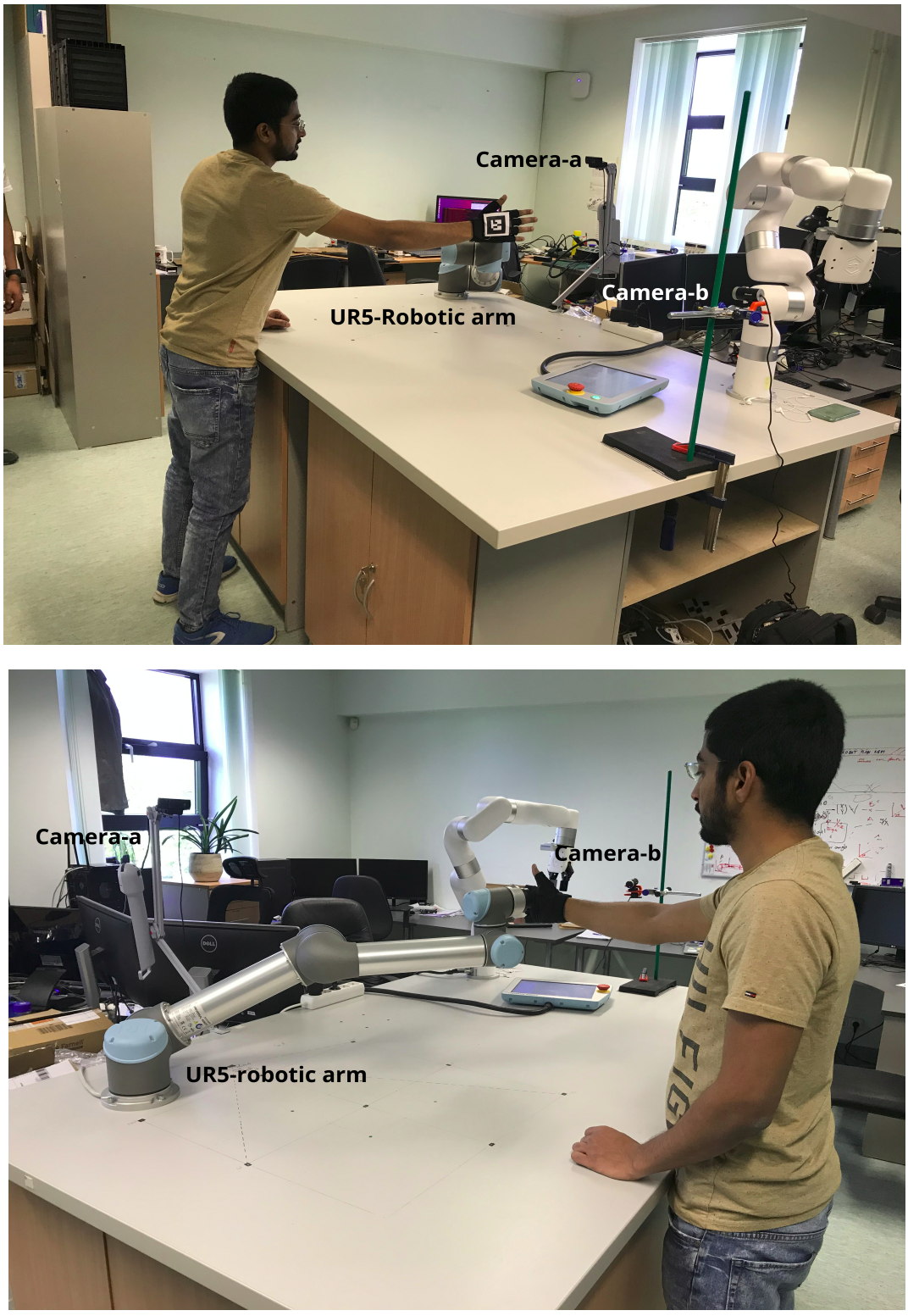

Figure 4.1: *Experimental setup*

In the experiment, the human is aimed to look at the end- effector of the robotic arm, shown in Fig. 4.2. Camera-a captures the image which is then used to calculate the head pose, i.e roll, pitch and yaw. The calculated roll, pitch and yaw of the image in Fig. 4.2 are 8.85896, -10.5195 and 39.5309 respectively. Each gaze record is with respect to its corresponding end-effector position.

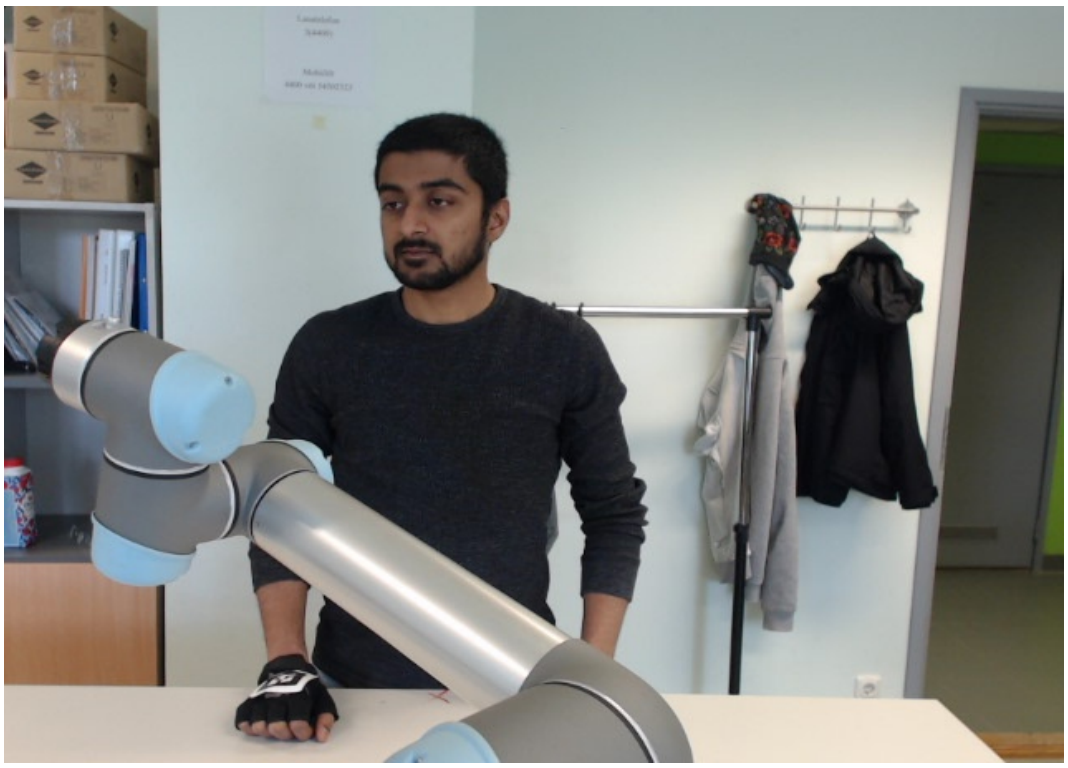

Figure 4.2: Deepgaze experiment for calculating human gaze (Roll, Pitch and Yaw) with respect to End-effector position

A total of three hundred and thirty (330) images and gaze records have been collected for this experiment (see the scatter plot in Fig. 4.3) that serves as the primary data for our first neural network model.

4.1.2 Trajectory Data

Besides gaze detection, the second goal is to detect and record a moving hand. For this purpose, an AR tag (see Figure 2.5) is placed on human hand glove. Camera-b is used by AT tracker algorithm that aims to get position of AR tag while human is moving his hand towards robotic arm end-effector.

The sequence of points from start to end-effector position constitutes a trajectory. In practice, the number of points in a trajectory can not be fixed and vary with trajectory length. To counter this problem, the data needs to be re-sampled to make sure all trajectories have equal number of points before passing it as input to neural network model. In the experiment,

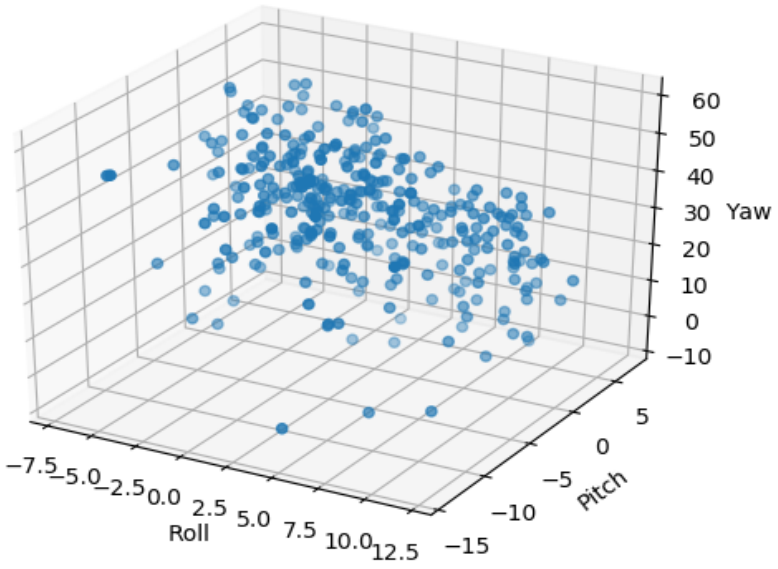

Figure 4.3: Scatter plot shows Gaze Data (Roll, Pitch and Yaw)

each trajectory is supposed to have twenty number of points. The trajectory is up-sampled if it has less than twenty number of points, meaning creating new points between the first and the last point. If the trajectory has more than twenty points, extra points are randomly removed from the trajectory.

After re-sampling, each trajectory is smoothed by fitting a spline to it. A total of three hundred and thirty (330) trajectories shown in Fig. 4.4 have been collected for the experiment.

4.1.3 End-effector Data

The end-effector position of UR5 robotic arm plays a vital role in all this process. The end-effector position is read with the help of a ROS node and recorded for different instances. Both gaze and trajectory are basically according to end-effector position. For every end-effector position, gaze and trajectory are recorded hence the number of end-effector positions are also three hundred and thirty, shown in scatter plot in Fig. 4.5.

The robotic arm end-effector position is used as the label for training machine learning models, the machine learning models will be discussed in detail in the next section.

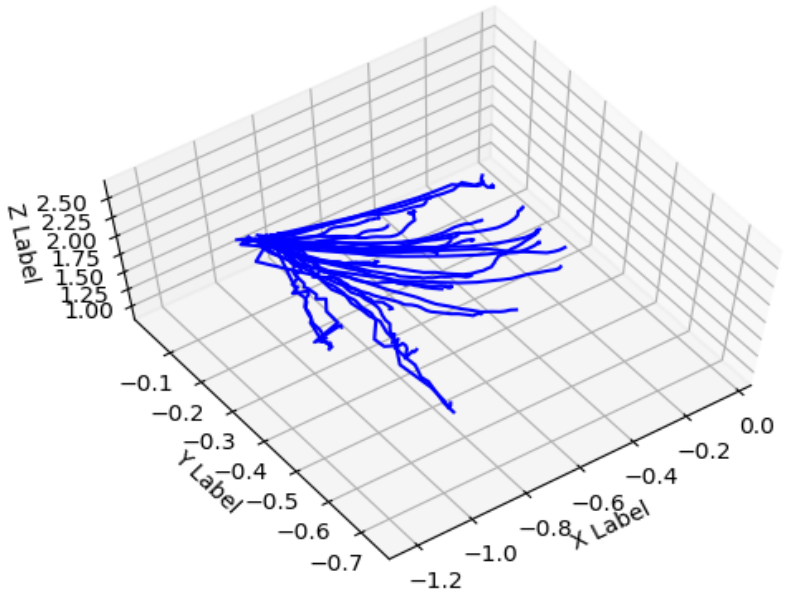

Figure 4.4: Trajectories with respect to their End-effector positions

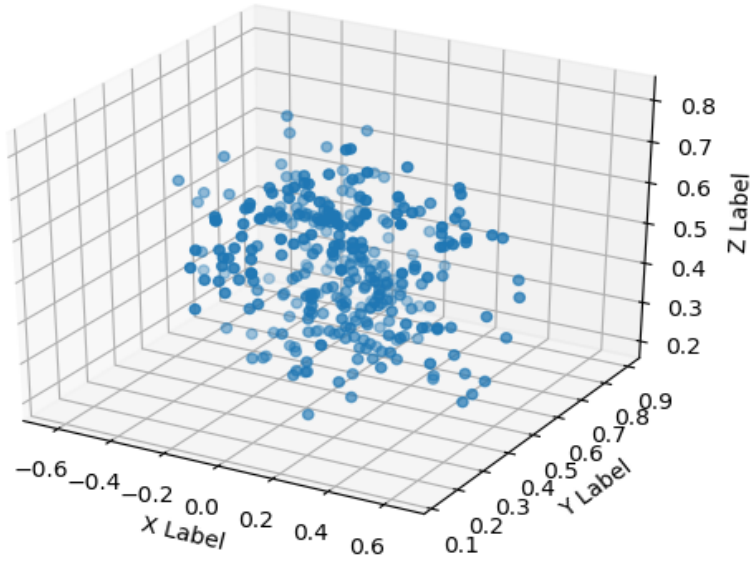

Figure 4.5: Scatter plot shows End-effector positions

4.2 Neural-Network Models

As explained in the previous chapter, supervised learning is a method in machine learning that uses input data and output data or labels to train the model. Using the supervised method, neural network models are designed for the end-point prediction of human reaching trajectories. Three different models are built. The first model relies purely on human-gaze information, the second model relies purely on the human hand trajectory, i.e. time stamped sequence of x, y, z position of hand and the final model uses a combination of gaze information with the prediction obtained from purely the trajectory data as the input feature of the neural network.

The neural network models are built and implemented using PyTorch framework and have been discussed in earlier chapter in detail. As PyTorch uses data in the form of tensors, the data is pre-processed and is converted to tensor form before using it. The models are sequential meaning each layer has one input tensor and one output tensor. The number of neurons at input and output layers in each model depends on the number of features of the corresponding data.

Neurons in each layer get the input and produce the output based on activation function they use. Two of the most commonly used activation functions, ReLU and Tanh, were used for this experiment and their corresponding results will be discussed in detail in next chapter. In order to learn from the data, the weights of the network are required to update during the training process. The models in this experiment use Adam optimizer which is computationally efficient and straightforward to implement in ML models. In every epoch, the result of output layer is compared with the desired output and error is calculated using the loss function. MSE is used as the loss function in these models.

Once the training is done, the model is tested on data to predict the output. The possible errors, maximum error 4.2 and average or mean error 4.2, between desired and predicted outputs are calculated using the following mathematical equations.

$$Error_{Max} = \max \sqrt{(x_d - x_p)^2 + (y_d - y_p)^2 + (z_d - z_p)^2} \tag{4.1}$$

$$Error_{Avg} = (\frac{1}{N}) \sqrt{(x_d - x_p)^2 + (y_d - y_p)^2 + (z_d - z_p)^2} \tag{4.2}$$

where $(.)_d$ stands for the ground-truth and $(.)_p$ stands for the prediction from the neural network model and N is the number of points.

4.2.1 Machine learning model for gaze and end-effector

This neural network model, block diagram shown in Fig. 4.6, takes gaze parameters roll, pitch and yaw as input and end-effector position coordinates x, y and z as their corresponding

label or output. The network model block diagram shown in Fig. 4.6 has same number of neurons at input and output layers because they have same number of features. The model has a total of seven layers, six hidden layers and an output layer. The number of neurons in each layer are 10, 20, 25, 15, 10, 5 and 3 respectively.

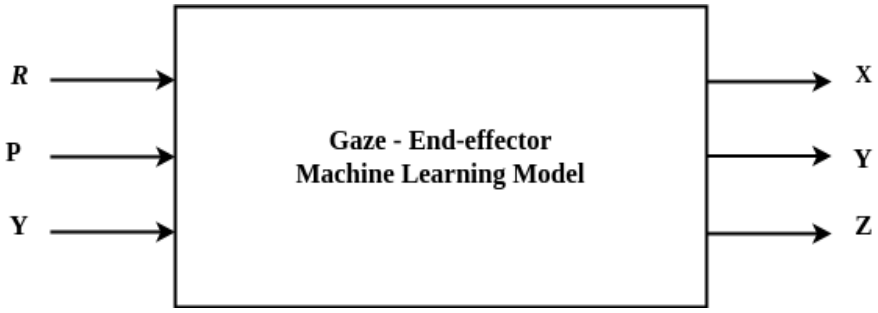

Figure 4.6: Machine learning model block diagram for gaze and end-effector

4.2.2 Machine learning model for trajectory and end-effector

This neural network model takes all the points in a trajectory as input and end-effector position as its corresponding output label. As mentioned earlier in this chapter, each trajectory is comprised of 190 points and each point is comprised of x, y and z coordinates, making a sum of 570 features at input. Similar to previous model, the output is end-effector position and has only three features. This model has six layers in total, five hidden layers and an output layer. The number of neurons in each layer of this network are 350, 200, 75, 25, 5 and 3 respectively.

4.2.3 Machine learning model for gaze, trajectory and end-effector

This model is basically a combination of previous two models. The predicted output of trajectory, which is the predicted end-effector position with respect to trajectory, along with gaze coordinates is used as input to this model and end-effector position is its corresponding output. The input has six features at its input and three output features. This model is also comprised of seven layers, six hidden layers and an output layer. The number of neurons in each layer are 15, 30, 50, 25, 10, 5 and 3 respectively.

5 Results

This chapter presents the prediction results of the three neural network models described in the previous chapter.

5.1 Experimental Results

As described in chapter two, the performance of a neural network model depends on the number of layers and neurons in each layer, optimizer, activation function, loss function, number of epochs and the learning rate. Besides all these parameters, data plays the main role in terms of performance. The amount and accuracy of data are the key factors to affect the training of a model. The more and accurate the data is, the better the performance is from the model. The data for this experiment comprised of gaze, trajectories and end-effector positions, three hundred and thirty (330) each in total. Neural Network models were trained using three hundred of each data set and tested on rest thirty.

5.1.1 Gaze to End-effector model results

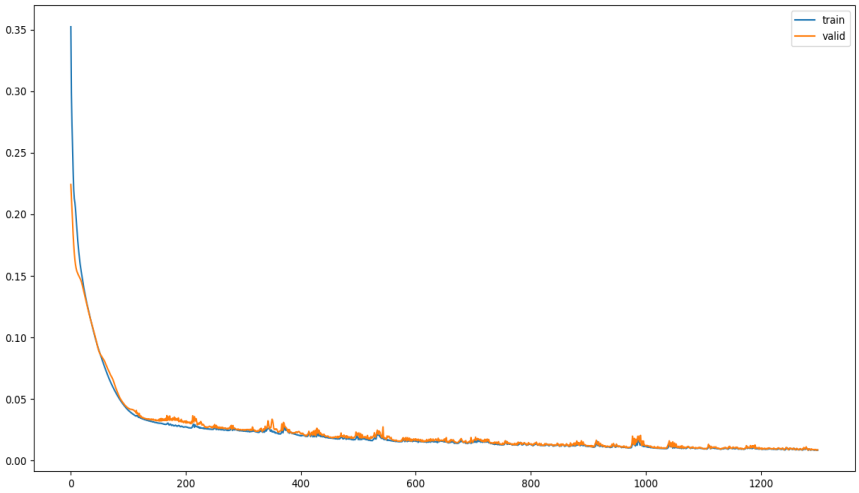

Figure 5.1: Training and validation loss curves for Gaze to End-effector model

This neural network model, uses gaze data at input and end-effector position as their respective labels. The model was trained with different number of hidden layers, epochs, learning rates, optimizer and activation functions to get better possible results. The model was trained at 1500 epochs and at a learning rate of 0.0025 with Adam optimizer, Tanh activation function and MSE loss function to get the final results. The training and validation loss are summarized in Fig. 5.1.

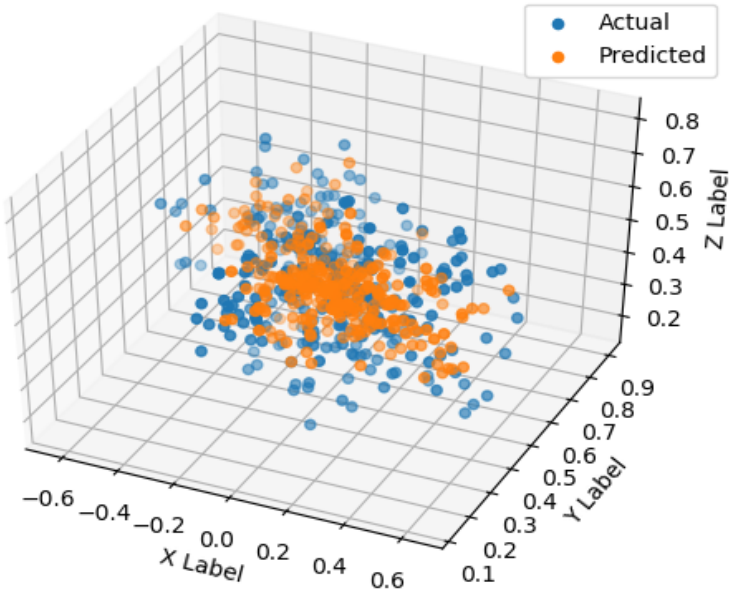

Figure 5.2: Actual and predicted End-effector positions using Gaze training data

As can be seen from Fig. 5.1 the training error is decreasing with the number of epochs. Furthermore, the validation loss is near to training loss which shows the model has fitted well to the data and over-fitting has been avoided. The model was then used to predict the end point position. Two separate predictions were performed in the training and testing data respectively. The former was primarily done to infer the best possible accuracy of the learned model. Fig. 5.2 and Fig. 5.3 show the actual and predicted end-effector positions for training and testing data inputs respectively.

For training data, the maximum and average errors are 41.2cm and 14.3cm. Whereas for testing data the errors are much higher which are 109.8cm as maximum and 43.6cm as average error.

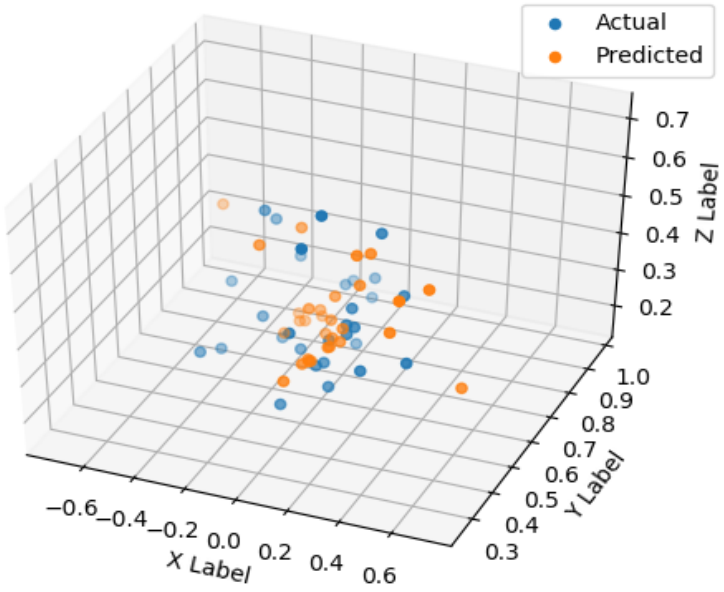

Figure 5.3: Actual and predicted End-effector positions using Gaze testing data

5.1.2 Trajectory to End-effector model results

Trajectory to End-effector neural network model takes human hand trajectory as inputs and predicts the end point position. Like previous model, this model was also trained with varying parameters to get the best possible results that were obtained with similar parameters as of previous model, except for the number of layers and neurons in each layer. Training and validation loss curves shown in Fig. 5.4 validate the learning performance of the model.

Training and validation loss curves shown in Fig. 5.4 show the learning process of the model. From curves one can observe that the model learns efficiently and is a good-fit as train and validation curves are very close to each other and the error decreases with increase in number of epochs. Similar to previous model, this model was trained using three hundred (300) trajectories and their corresponding end-effector positions and was tested on remaining thirty (30) trajectories. Fig. 5.5 and 5.6 show the predicted end-effector positions with training and testing data.

For training data, the maximum error is 7.98cm and the average error is 1.85cm whereas the errors for testing data are 48.3cm and 9.1cm respectively.

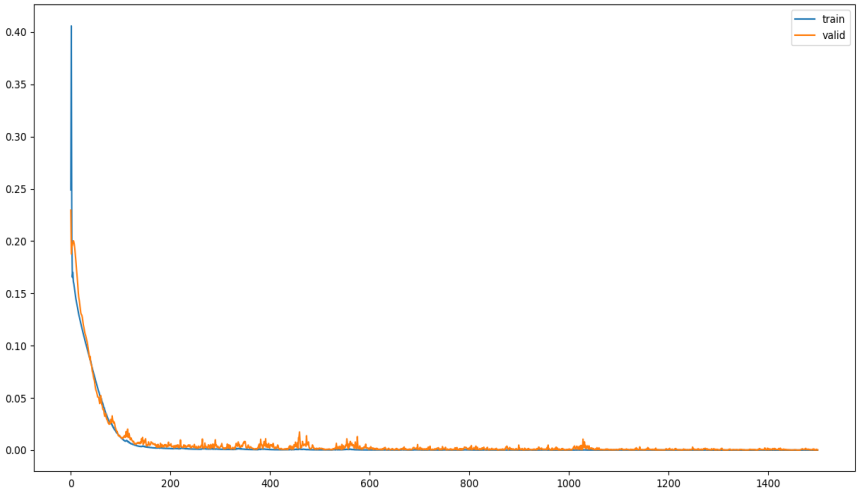

Figure 5.4: Training and validation loss curves for Trajectory to End-effector model

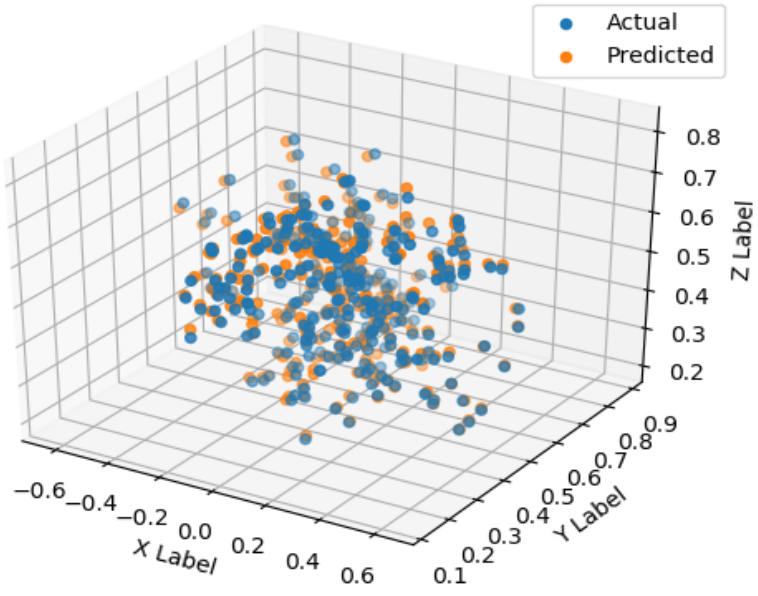

Figure 5.5: Actual and predicted End-effector positions using Trajectory training data

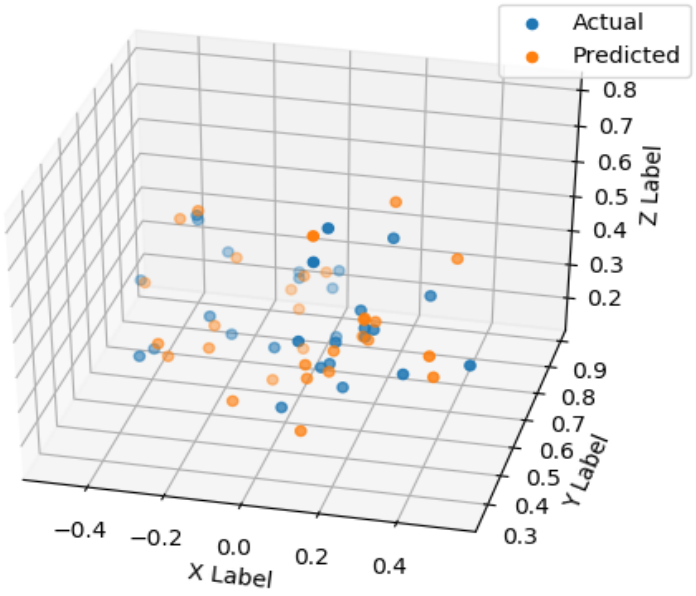

Figure 5.6: Actual and predicted End-effector positions using Trajectory testing data

5.1.3 Gaze, Trajectory and End-effector model results

This model is a combination of previous two models. The predicted end point positions from the second model, which was based on human hand trajectory, are used as input along with gaze to predict the final end point positions. All parameters are similar to previous two models except number of layers and neurons in them and the learning rate. This model gave its best predicted output at a learning rate of 0.003.

Training and validation loss curve in Fig. 5.7 show that the model is learning and is a good-fit. The error decreases with the number of epochs and both curves are almost at same level at the end. The predicted end-effector positions for training and testing data are shown in Fig. 5.8 and Fig. 5.9.

In this model, the maximum and average errors for training data are 8.4cm and 1.98cm where as for testing data the errors are 37cm and 14.8cm respectively.

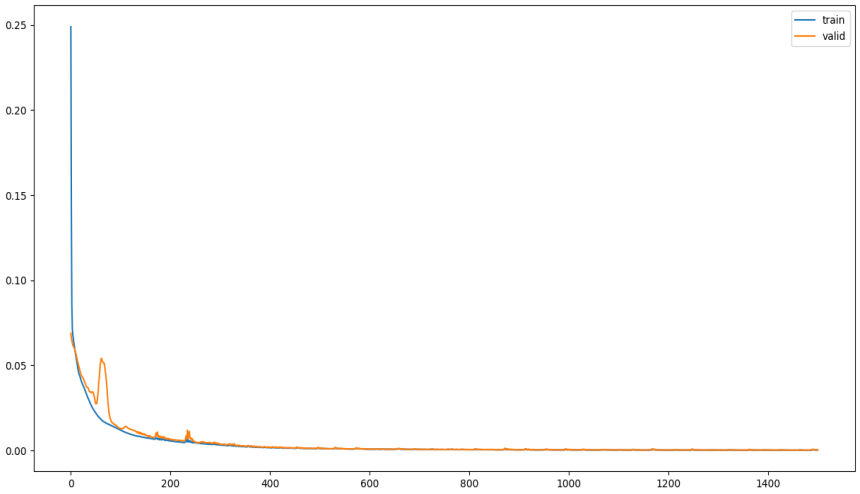

Figure 5.7: Training and validation loss curve for Gaze, Trajectory and End-effector model

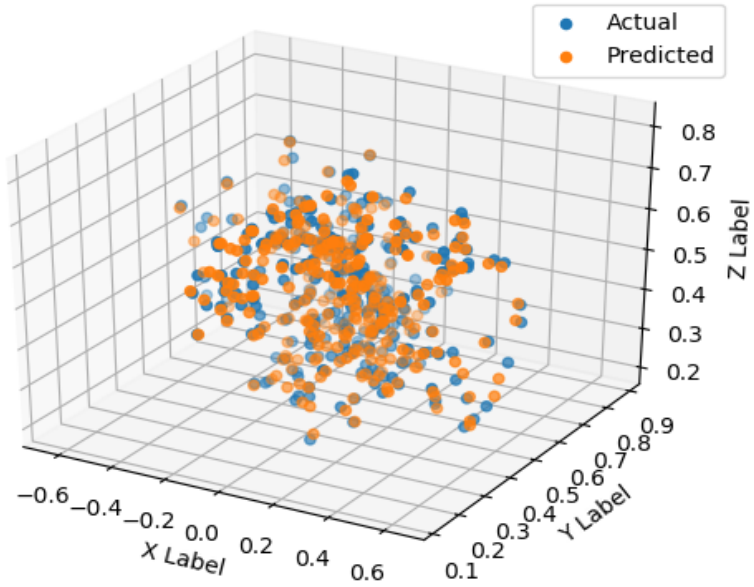

Figure 5.8: Actual and predicted End-effector positions using Trajectory-Gaze training data

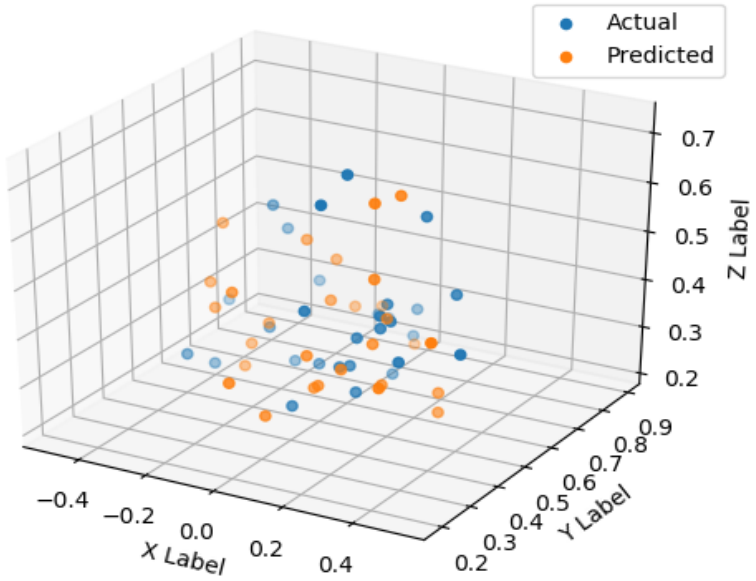

Figure 5.9: Actual and predicted End-effector positions using Trajectory-Gaze testing data

Table 5.1: Comparison Table shows the mean and average errors for each model

Errors	Gaze-EE	Traj: - EE	Gaze, Traj: - EE
Train (Max) (cm)	41.2	7.98	8.4
Train (Avg) (cm)	14.3	1.85	1.98
Test (Max) (cm)	109.8	48.3	37
Test (Avg) (cm)	43.6	9.1	14.8

5.2 Comparison

The experimental results in Table 5.1 show that the Gaze to End-effector model is the least accurate in predicting end point positions of the reaching trajectory. The errors are too big to implement them in practical applications. Trajectory to End-effector model is the most efficient model and the predictions are close to the actual point positions, especially in the case of training data set. The fusion of Gaze and Trajectory to End-effector model gives the least maximum error whereas prediction based on the hand trajectory (Trajectory to End-effector model) gives the least average error.

The performance of the models to give better predictions can surely be improved by increasing the volume of data. Gaze alone is not sufficient to build effective models for practical applications however it can used along with trajectory to get better results as in the last model.

6 Conclusions and Future work

In this thesis, we presented three neural network models to predict the end-point of human reaching trajectory. An extensive experimental set-up was built during the thesis for accurate data collection and involved integrating concepts from robotics and computer vision. Three neural network models with MLP architecture were proposed that leveraged gaze information as obtained from an off-the-shelf library and hand motion trajectory to obtain prediction of end-point of the reaching trajectory with best case maximum error of 37 cm and best case average error of 9.1 cm.

There are several directions to expand the current thesis. Firstly, the current result is preliminary in the sense that it was tested on a limited amount of data. Training on a larger data set can dramatically improve the prediction performance. Second, the prediction model can be integrated in a real-world human-robot collaboration task such as collaborative manufacturing. Finally, the thesis only explored neural networks with MLP architecture. Thus, the performance of other architectures such as LSTM needs to be explored.

Acknowledgements

I would like to acknowledge and thank

my teacher and supervisor Arun Kumar Singh for his guidance and support in this work,

my parents for their motivation and support during my struggle

and other fellows including Morteza Daneshmand, Robert Valner and Houman Masnavi who were always there to help me to get this work done.

I also thank Shobit Jain for his time and help in data collection process.

Bibliography

[1] Weitian Wang et al. "Controlling object hand-over in human–robot collaboration via natural wearable sensing". In: *IEEE Transactions on Human-Machine Systems* 49.1 (2018), pp. 59–71.

[2] Phongtharin Vinayavekhin et al. "Human-like hand reaching by motion prediction using long short-term memory". In: *International Conference on Social Robotics*. Springer. 2017, pp. 156–166.

[3] Michael A. Nielsen. *Neural Networks and Deep Learning*. Determination Press, 2015.

[4] *Neural Network*. https://www.investopedia.com/terms/n/neuralnetwork.asp. [Online; 20-July-2020].

[5] *Publication-ready NN-architecture schematics*. http://alexlenail.me/NN-SVG/index.html. [Online; 20-July-2020].

[6] *Understanding Neural Networks*. https://towardsdatascience.com/understanding-neural-networks-22b29755abd9. [Online; 20-July-2020].

[7] *Hidden layer*. https://deepai.org/machine-learning-glossary-and-terms/hidden-layer-machine-learning. [Online; 20-July-2020].

[8] *7 Types of Neural Network Activation Functions: How to Choose?* https://missinglink.ai/guides/neural-network-concepts/7-types-neural-network-activation-functions-right/. [Online; 20-July-2020].

[9] *A Walk-through of Cost Functions*. https://medium.com/machine-learning-for-li/a-walk-through-of-cost-functions-4767dff78f7. [Online; 21-July-2020].

[10] *Overview of different Optimizers for neural networks*. https://medium.com/datadriveninvestor/overview-of-different-optimizers-for-neural-networks-e0ed119440c3. [Online; 21-July-2020].

[11] *Understand the Impact of Learning Rate on Neural Network Performance*. https://machinelearningmastery.com/understand-the-dynamics-of-learning-rate-on-deep-learning-neural-networks/. [Online; 21-July-2020].

[12] *Activation Functions*. https://www.researchgate.net/figure/Function-curves-of-sigmoid-Tanh-and-ReLU_fig1_330467251. [Online; 21-July-2020].

[13] *Activation Functions in Neural Networks*. https://towardsdatascience.com/activation-functions-neural-networks-1cbd9f8d91d6. [Online; 22-July-2020].

[14] *How to use Learning Curves to Diagnose Machine Learning Model Performance*. https://machinelearningmastery.com/learning-curves-for-diagnosing-machine-learning-model-performance/. [Online; 22-July-2020].

[15] *Typical CNN architecture*. https://en.wikipedia.org/wiki/Convolutional_neural_network/media/File:Typical_cnn.png. [Online; 22-July-2020].

[16] *Types of Neural Networks and Definition of Neural Network*. https://www.mygreatlearning.com/blog/types-of-neural-networks/multilayeredperceptron. [Online; 22-July-2020].

[17] *7 types of Artificial Neural Networks for Natural Language Processing*. https://medium.com/@datamonsters/artificial-neural-networks-for-natural-language-processing-part-1-64ca9ebfa3b2. [Online; 22-July-2020].

[18] *What is ROS*. http://wiki.ros.org/ROS/Introduction. [Online; 05-August-2020].

[19] *What is moveit_ros? All about MoveIt! ROS*. https://www.theconstructsim.com/ros-movelt/. [Online; 06-August-2020].

[20] *ROS - Data display with Rviz*. https://www.stereolabs.com/docs/ros/rviz/. [Online; 06-August-2020].

[21] Massimiliano Patacchiola and Angelo Cangelosi. "Head Pose Estimation in the Wild using Convolutional Neural Networks and Adaptive Gradient Methods". In: *Science Direct* (2017), p. 38.

[22] *What is Deepgaze*. https://github.com/mpatacchiola/deepgaze. [Online; 22-July-2020].

[23] *ar_track_alvar*. http://wiki.ros.org/ar_track_alvar. [Online; 06-August-2020].

[24] Emanuel Todorov and Michael I Jordan. "Optimal feedback control as a theory of motor coordination". In: *Nature neuroscience* 5.11 (2002), pp. 1226–1235.

[25] Tamar Flash and Neville Hogan. "The coordination of arm movements: an experimentally confirmed mathematical model". In: *Journal of neuroscience* 5.7 (1985), pp. 1688–1703.

[26] Claudia Pérez-D'Arpino and Julie A Shah. "Fast target prediction of human reaching motion for cooperative human-robot manipulation tasks using time series classification". In: *2015 IEEE international conference on robotics and automation (ICRA)*. IEEE. 2015, pp. 6175–6182.

[27] Ruikun Luo, Rafi Hayne, and Dmitry Berenson. "Unsupervised early prediction of human reaching for human–robot collaboration in shared workspaces". In: *Autonomous Robots* 42.3 (2018), pp. 631–648.

[28] Jim Mainprice, Rafi Hayne, and Dmitry Berenson. "Goal set inverse optimal control and iterative replanning for predicting human reaching motions in shared workspaces". In: *IEEE Transactions on Robotics* 32.4 (2016), pp. 897–908.

[29] Philipp Kratzer, Marc Toussaint, and Jim Mainprice. "Motion prediction with recurrent neural network dynamical models and trajectory optimization". In: *arXiv preprint arXiv:1906.12279* (2019).

[30] Heramb Nemlekar, Dharini Dutia, and Zhi Li. "Object transfer point estimation for fluent human-robot handovers". In: *2019 International Conference on Robotics and Automation (ICRA)*. IEEE. 2019, pp. 2627–2633.

[31] Justin W Hart et al. "Predictions of human task performance and handover trajectories for human-robot interaction". In: *Proceedings of the HRI Workshop on Human-Robot Teaming (HRI'15)*. 2015.

[32] José R Medina et al. "A human-inspired controller for fluid human-robot handovers". In: *2016 IEEE-RAS 16th International Conference on Humanoid Robots (Humanoids)*. IEEE. 2016, pp. 324–331.

[33] Guilherme J Maeda et al. "Probabilistic movement primitives for coordination of multiple human–robot collaborative tasks". In: *Autonomous Robots* 41.3 (2017), pp. 593–612.

Non-exclusive licence to reproduce thesis and make thesis public

I, Zafarullah

1. herewith grant the University of Tartu a free permit (non-exclusive licence) to reproduce, for the purpose of preservation, including for adding to the DSpace digital archives until the expiry of the term of copyright,

   "Gaze Assisted Neural Network based Prediction of End-Point of Human Reaching Trajectories"

   supervised by Arun Kumar Singh

2. I grant the University of Tartu a permit to make the work specified in p. 1 available to the public via the web environment of the University of Tartu, including via the DSpace digital archives, under the Creative Commons licence CC BY NC ND 3.0, which allows, by giving appropriate credit to the author, to reproduce, distribute the work and communicate it to the public, and prohibits the creation of derivative works and any commercial use of the work until the expiry of the term of copyright.

3. I am aware of the fact that the author retains the rights specified in p. 1 and 2.

4. I certify that granting the non-exclusive licence does not infringe other persons' intellectual property rights or rights arising from the personal data protection legislation.

Zafarullah
15.08.2020