On the Limits of AI-based Prediction of Hand Movement Trajectories for Real-Time Remote Robotic Control

E. Ruiz-Moreno[1], K. Kousias[2], Pattiwar Shravan Kumar[2], M.E.J. Tilleman[2], B. Beferull-Lozano[1], Ö. Alay[2], C. Griwodz[2], A. Filippeschi[3], J.d.R. Millán[4], and M. Bergamasco[3]

[1]SIGIPRO Department, Simula Metropolitan Center for Digital Engineering, Oslo, Norway
[2]Department of Informatics, University of Oslo, Oslo, Norway
[3]Institute of Mechanical Intelligence, School of Advanced Studies Sant'Anna, Pisa, Italy
[4]Chandra Department of Electrical and Computing Engineering, The University of Texas at Austin, Texas, United States

12/02/2025

Abstract

Accurate human motion prediction is essential for remote control applications in robotics and human-computer interaction. Despite this, the scalability, task-specific accuracy, and performance under varying prediction horizons of existing off-the-shelf forecasting methods remain insufficiently explored. This technical report evaluates several Machine Learning (ML) forecasting models, with a particular focus on Recurrent Neural Network (RNN)-based architectures, analyzing their accuracy, strengths, and limitations across different time horizons and pose complexities. Specifically, we examine the prediction of hand movement trajectories recorded using a Rokoko Smartglove connected via WiFi-6 to a Shadow Robot Dexterous Hand. Our experiments reveal that RNN-based models exhibit a Mean Absolute Error (MAE) that increases linearly with the prediction horizon in a range from 100ms to 500ms. Additionally, as the number of poses grows, MAE also increases, highlighting potential scalability challenges towards free movement (unlimited poses). While more complex models can improve accuracy or mitigate the error growth by scaling with the number of poses, they also carry higher computational costs. We believe that these findings can provide valuable insights for developing robust and scalable forecasting models that improve human motion prediction.

The associated code and data presented in this report are available here: `https://github.com/SIGIPRO/ShadowHandMotionPrediction`

1 Introduction

In recent years, several vertical applications, including healthcare, agriculture, manufacturing, and transportation and logistics, have been seeking automated solutions to enhance work efficiency, improve safety, reduce operational costs, and enable the execution of tasks in remote sites. Autonomous robots can be used to deliver these benefits as they can perform a wide range of jobs without human intervention. Via advanced mechanisms that enable real-time data processing and adaptive decision-making based on their environment, autonomous robots are transforming how work is carried out across these sectors. However, autonomous robots are typically designed to handle more repetitive and simpler tasks and often struggle when deployed in more complex scenarios that demand high precision and adaptability. To facilitate these more complex use cases, we focus on empowering and embodying actions in real-time through remotely-controlled robots. Real-time coordination and interaction through remote-controlled robotic systems is an area of growing importance across various fields, including Public Protection and Disaster Relief (e.g., search and rescue missions in impassable areas struck by disasters), eHealth (e.g., remote physiotherapy or cybersurgery), and upcoming Industry 5.0 (e.g., manufacturing of parts in factories remotely). Even though these applications differ in purpose, they share some common requirements: high precision, low latency, and immersive interaction between the robot and its environment. These requirements are necessary to ensure that remotely-operated robots can perform tasks accurately and efficiently in real-world scenarios.

One of the primary factors towards achieving optimal conditions for real-time remote manipulation is the underlying network infrastructure for communication between the two sides (i.e., human and robot). Recent advancements in mobile networks, particularly in 5G and B5G, have introduced several technologies aimed at significantly enhancing network performance on each end. Such examples include massive Multiple-Input Multiple-Output, beamforming, network slicing, edge computing, and Machine Learning (ML)-driven network functions. Together, these innovations aim to reduce latency on the radio site to a range of 1-50 ms, depending on the environment and distance, meeting the stringent requirements of Ultra-Reliable Low Latency Communications, while also minimizing network jitter compared to previous generations. However, even when peak network performance is achieved, the overall responsiveness of the system can still be constrained by mechanical limitations on the robot's side. For example, in the majority of scenarios motor actuators are not capable of reacting to human motions with sufficient speed, thereby introducing delays that impact real-time interaction, which highlights the need for additional measures and actions to ensure seamless real-time control.

One of the most effective ways to overcome such limitations is to enforce motion prediction through the adoption of ML algorithms. Motion prediction refers to the process of forecasting the subsequent position of a given movement based on its current state and historical input data. By leveraging accurate predictions, we can proactively reposition the robot to its expected position,

effectively masking the end-to-end latency and thus enhancing immersion. Popular techniques in this field include Long Short-Term Memory (LSTM) and Gated Recurrent Unit (GRU) neural networks, both specialized architectures of Recurrent Neural Networks (RNN) designed to handle sequential data and capture long-term dependencies efficiently.

In this paper, we address the challenge of motion prediction using ML within the context of real-time remote control robotic systems, aiming to provide valuable insights on both performance and limitations aspects. Typically, this involves tracking a set of state variables (kinematics expressed in rotations), thus resulting in a multivariate prediction problem. Our primary focus is to study and evaluate whether and how accurately we can predict human hand motion by leveraging four off-the-shelf methods for time series forecasting algorithms. In particular, by adopting an empirical approach, we conduct a series of experiments to enable human-robot interaction using an end-to-end system composed of three key components; i) a Rokoko Smartglove that captures human hand motion and expressing it using quaternions [Kuipers, 1999], ii) an ML framework that processes the input data and generates multivariate models with 24 variables, each representing a distinct degree of freedom of the hand joints, to predict future positions within a specified time horizon, and iii) a Shadow Robot Dexterous Hand that executes actions based on the predictions generated by the ML models.

Figure 1: Example of a pose with the Rokoko Smartglove and the Shadow Robot Dexterous Hand

The multivariate nature of the collected dataset allows us to analyze the

joint behavior over time and exploit possible interdependencies. Our study involves multiple repetitions of five distinct hand movement poses carried out by five different participants. An example of a single pose is shown in Figure 1). For each pose, we explore the performance boundaries of four ML models in terms of prediction accuracy. Additionally, we analyze key factors such as the performance of ML models in relation to the complexity of human poses and the time horizon over which accurate predictions can be made. This provides a comprehensive evaluation of the models' effectiveness in real-world scenarios. As an additional contribution, we open-source both the dataset and the ML framework to allow for replicability of the experiments as well as further research with other ML algorithms.

The rest of the paper is structured as follows. Section 2 presents the related literature, while Section 3 provides background information on the joint structure mapping between the human hand and the Shadow Robot as well as an overview of the ML models employed in this study. Section 4 describes the experimental setup and outlines the collected dataset, while Section 5 presents the evaluation of the motion prediction analysis. Section 7 concludes the paper.

2 Related Work

2.1 Remote control and manipulation of robotic systems

Real-time remote coordination of robots has been studied in multiple occasions in the literature from a plethora of different angles. In [Groshev et al., 2022], a forecast-based recovery mechanism for real-time remote control of robotic manipulators (FoReCo) that leverages ML towards inferring lost commands caused by interference in the wireless channel was proposed. Similar approaches leveraging AI and ML techniques for teleoperation tasks are presented in [Solanes et al., 2020, Rubagotti et al., 2019]. In addition, a lot of focus has been put into hand gesture recognition in teleoperation scenarios involving robotic systems and AI [Hu et al., 2003, Qi et al., 2021, Ajili et al., 2017, Gao et al., 2022, Chamorro et al., 2021]. While these studies offer valuable insights into enhancing communication and human-robot interaction, they primarily focus on classification-based methodologies, which restrict the range of movements a human hand can perform, making it more applicable to the case of autonomous robots. In our study, we aim to go beyond these approaches and focus on predicting hand movement trajectories using AI-based methodologies.

2.2 Multivariate time series forecasting

We dissect the time series forecasting models into three categories, which are not mutually exclusive (some models can fit into several categories), as usually found in the literature. We review both univariate and multivariate approaches. Even though most of the univariate approaches can be extended to multivariate ones in various ways, we are interested in those approaches that can exploit

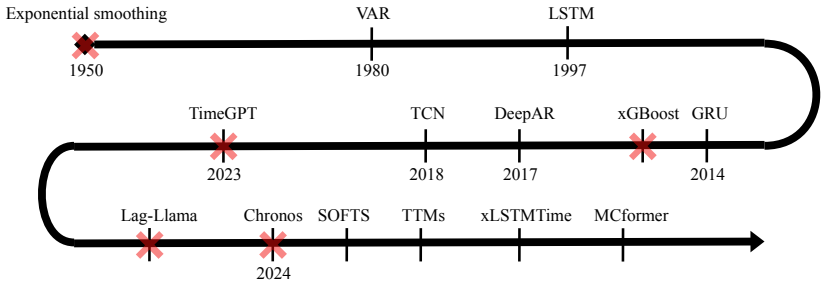


Figure 2: Chronology of main forecasting models. The red × indicates that the model is meant for univariate time series.

multivariate relations off-the-shelf. Fig. 2 illustrates the chronology of some of the most popular models discussed next to have a better perspective of the advancements in the field.

Statistical models. Statistical models make use of statistical assumptions and mathematical relations that help explain the process generating the time series data. In this category, one can find exponential smoothing [Brown and Meyer, 1961], vector autoregressive (VAR) models [Lütkepohl, 2005], and Prophet [Taylor and Letham, 2018], among others.

Deep learning models. Deep learning (DL) models usually use relatively large neural networks, i.e., a family of parametric models, that can be tuned to approximate a functional relationship. Some of the most popular DL-based models use LSTM-based architectures, which are RNNs designed to capture temporal dependencies [Hochreiter, 1997]. In this category, one can also find models using GRUs, which can be seen as a simplified version of LSTMs, which are usually faster to train while having comparable performance to LSTMs. On the other hand, Temporal Convolution Networks (TCNs) [Bai et al., 2018] provide some advantages over RNNs, i) better gradient flow: TCNs avoid issues such as vanishing or exploding gradients, and ii) faster training: unlike RNNs, TCNs can process data in parallel, making them potentially faster during training and inference. On the other hand, TCNs rely on fixed-size convolutions, requiring padding or other techniques to handle variable-length inputs, making them less flexible in some cases.

Ensembles and hybrid models. Combining simpler models (e.g., VAR models) with more sophisticated approaches (e.g., DL) in ensemble frameworks, can often yield robust predictions for multivariate series. For example, DeepAR combines RNNs with statistical modeling [Salinas et al., 2020].

2.2.1 Off-the-shelf models

From the approaches discussed before, the VAR, GRU, and LSTM-based forecasting models are arguably the most popular choices for multivariate time series prediction. This may be due to their simplicity or available libraries, e.g.,

[Paszke et al., 2019], that allow a seamless implementation off-the-shelf.

2.2.2 State-of-the-art models

In the last couple of years, there has been some research towards leveraging the capabilities of large language models (LLM) for time series prediction. Some of the first models exploring this research direction are timeGPT [Garza and Mergenthaler-Canseco, 2023] (proprietary software), lag-Llama [Rasul et al., 2023], and Chronos [Ansari et al., 2024], among others. However, all these are intended for univariate time series.

Alternative approaches trying to solve the multivariate setting limitation above include SOFTS [Han et al., 2024a], xLSTMTime [Alharthi and Mahmood, 2024], MCformer [Han et al., 2024b], and Tiny Time Mixers (TTMs) [Ekambaram et al., 2024].

Zeng et al. [2023] argue that the nature of the permutation invariant self-attention mechanism in Tranformer-based architectures (used by most LLMs) inevitably results in temporal information loss. In this regard, they propose a simple alternative model called Long Term time-Series Forecasting (LTSF)-Linear that outperforms existing sophisticated Transformer-based LTSF models in several real-world datasets, often by a large margin. Another relatively extended drawback of DL models, especially among the state-of-the-art, is data hungriness [Lee Jr, 1973], i.e., large amounts of training data needed for a model to generate a prediction within a decent accuracy level.

3 Background

This section's intent is twofold: first, to provide a high-level overview of the finger joints mapping between the human hand, the Rokoko Smartglove, and the Shadow Robot Dexterous Hand, and second, to present the main features of the ML prediction models employed in this work for multivariate time series forecasting.

3.1 Mapping of Finger Joints

Both the Rokoko Smartglove and the Shadow Robot are specifically designed to capture and reproduce the kinematics and dexterity of the human hand, respectively. While both of these components process hand motion information, they handle the collected data using different formats. In particular, the Rokoko glove captures human motion and translates it into quaternion values, which define the position and rotation of each joint. On the contrary, Shadow Robot relies on input from the state of the joint expressed in angle degrees, offering 24 degrees of freedom. Figure 3 illustrates the finger joint mapping between the human hand, the Rokoko Smartglove, and the Shadow Robot Dexterous Hand, while Table 1 provides an overview of the fingers joints, as expressed by the Shadow Robot.

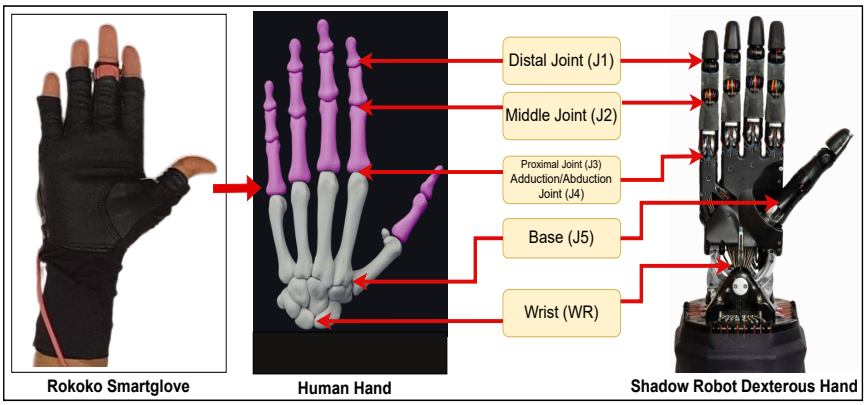


Figure 3: Joint structure mapping of the human hand from the Rokoko Smartglove to the Shadow Robot Dexterous Hand

The direction of rotation varies depending on the finger, i.e., for the FF and MF, *anti-clockwise rotation* is considered positive, while, for the RF and the LF, *clockwise rotation* is considered positive. This distinction in rotational behavior ensures precise lateral control of individual fingers, allowing the robotic hand to perform intricate tasks that require coordinated finger movements.

3.2 Prediction models

The prediction models we consider, are trained *just* to predict the *next* data point in the movement trajectory. That is, at time stamp $x_t \in \mathbb{R}$ they aim at predicting $\boldsymbol{y}_{t+1} \in \mathbb{R}^F$. We denote this prediction (the model *output*) as $\boldsymbol{o}_{t+1}$ and is also in $\mathbb{R}^F$.

To predict for further time horizons, we apply the prediction model recursively on its own predictions.

3.2.1 Linear model

We consider a linear model of the form

$$\boldsymbol{o}_t = \boldsymbol{A}_1 \boldsymbol{y}_{t-1} + \cdots + \boldsymbol{A}_P \boldsymbol{y}_{t-P}, \tag{1}$$

where the matrices $\boldsymbol{A}_i \in \mathbb{R}^{F \times F}$ for all $i \in \{1, \ldots, P\}$ contain the model learnable parameters, and P is the number of lags, i.e., the number of previous data points used in the prediction.

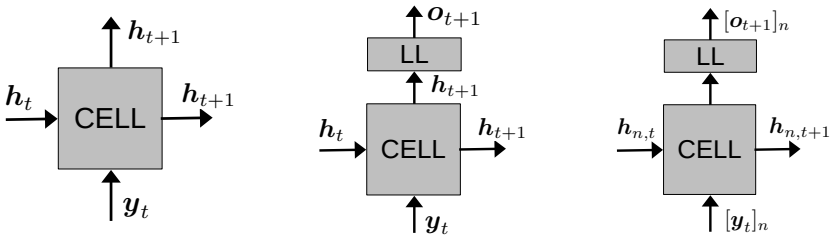
Joint	Description	FF	MF	RF	LF	TH	WR
Distal (J1)	Located near the fingertip and responsible for fine-tuned movements	✓	✓	✓	✓	✓	X
Middle (J2)	Positioned between the proximal and distal, enabling intermediate flexion and extension	✓	✓	✓	✓	✓	X
Proximal (J3)	Closest to the palm, allowing forward and backward bending	✓	✓	✓	✓	✓	X
Adduction/Abjuction (J4)	Located near the proximal, handling lateral movements, such as spreading the fingers apart (negative rotation) or bringing them together (positive rotation)	✓	✓	✓	✓	✓	X
Base/Palm (J5)	Used by the thumb to enable more precise movement and by the little finger to enable palm movement	X	X	X	✓	✓	X
Wrist	Represented by two angles, enabling hand rotation and positioning	X	X	X	X	X	✓

Table 1: Overview of the finger joints states. The following finger notation is used; First Finger (FF), Middle Finger (MF), Ring Finger (RF), Little Finger (LF), Thumb (TH), and Wrist (WR).

Note that the linear prediction model in (1) may fall into unstable dynamics if we try to predict recursively without guaranteeing stability. This is because if we recast (1) as

$$\begin{bmatrix} \boldsymbol{o}_t \\ \boldsymbol{y}_{t-1} \\ \vdots \\ \boldsymbol{y}_{t-P+1} \end{bmatrix} = \begin{bmatrix} \boldsymbol{A}_1 & \boldsymbol{A}_2 & \cdots & \boldsymbol{A}_{P-1} & \boldsymbol{A}_P \\ \boldsymbol{I} & \boldsymbol{0} & \cdots & \boldsymbol{0} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{I} & \cdots & \boldsymbol{0} & \boldsymbol{0} \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ \boldsymbol{0} & \boldsymbol{0} & \cdots & \boldsymbol{I} & \boldsymbol{0} \end{bmatrix} \begin{bmatrix} \boldsymbol{y}_{t-1} \\ \boldsymbol{y}_{t-2} \\ \vdots \\ \boldsymbol{y}_{t-P} \end{bmatrix}, \tag{2}$$

it is only stable iff the spectral radius of the multiplicative matrix, also known as companion matrix, is within the unit circle.
Let us refer to the companion matrix in (2) as $\boldsymbol{\Gamma} \in \mathbb{R}^{F\times F} \otimes \mathbb{R}^{P\times P}$. Then, one way of forcing stability is making sure that the model parameters are chosen such that $\rho(\Gamma) < 1$, where ρ refers to the spectral radius.



(a) Poorly scalable. (b) Moderately scalable. (c) Scalable.

Figure 4: Several RNN architectures. The first architecture scales poorly with the output size because the size of the hidden state needs to match it. The middle architecture detaches the output size from the size of the hidden state by using a linear layer (LL) that matches the sizes afterward. This allows for smaller cells, thus helping with scalability (hidden state size smaller than output size) or allowing for richer features (hidden state size bigger than output size). The last architecture concurrently reuses the same cell for each component of the input while maintaining separate hidden states This is the architecture that scales the best (constant with respect to the output input/size).

3.2.2 Recurrent Neural Network (RNN)-based model

We use an RNN-based architecture as a predictor. The input at time step t is the data point $\boldsymbol{y}_t$. The hidden state at time step t, i.e., $\boldsymbol{h}_t$ is a vector of H (hidden state size) elements that acts as a memory. The output of the RNN cell is transformed with a linear layer to match the dimensionality of the data points. In this way, we can choose a hidden size H independently of the size of the data point F. Finally, the RNN cell is constituted by L GRU units. The value L is usually referred to as the number of layers.

In particular, we consider two architectures: a gated recurrent unit (GRU)-based model and a long short-term memory (LSTM)-based model. The general idea of the architecture chosen is represented in Figure 4b.

3.2.3 Optimization cost

As mentioned before, the prediction models are trained to predict the next data point. Formally, the trained parameters of the models are obtained by solving

$$\arg\min_{\boldsymbol{\theta}\in\boldsymbol{\Theta}} \frac{1}{M}\sum_{m=1}^{M}\frac{1}{T_m}\sum_{t=\tau}^{T_m}\frac{1}{F}\|\boldsymbol{o}_t^{(m)} - \boldsymbol{y}_t^{(m)}\|_1, \tag{3}$$

where $\boldsymbol{\theta}$ is a vector that contains the model parameters, and $\boldsymbol{\Theta}$ is the parameter space, e.g., $\boldsymbol{\Theta} \subseteq \mathbb{R}^R$ where R is the number of learnable parameters. Note that the initial time step τ may change depending on the model. For example, a linear model with a lag $P = 3$ will be trained from the 3rd data point onwards, i.e., $\tau = 3$.

We have chosen the ℓ_1 norm as the loss mainly because it can be directly interpreted as the mean absolute error (MAE) over all joint angles and it also provides robustness against outliers [Jadon et al., 2024].

It should be noticed that, the form of the cost (objective function) in (3) plays a crucial role in the performance of the trained forecasting models. For example, in some tasks, one may require predicting certain joint angles more accurately than others. This can be done with a tailor-devised optimization cost. However, this is out of the scope of this technical report and is left for future work.

4 Experimental Setup

Figure 5 depicts the experimental setup adopted in this paper. It consists of two main parts, namely, i) the server side where motion is generated using the Rokoko Smartglove, and ii) the client side where motion prediction is carried out and fed into the Shadow Robot Dexterous Hand. A high-level description of the experimental flow is described in the following.

The Rokoko Smartglove captures the human hand motion leveraging its suite of sensors, including inertial measurement units (IMUs), gyroscopes, accelerometers, and magnetometers. These sensors measure linear acceleration, angular velocity, and orientation to track hand motion and rotational movements with high precision. The captured motion is then transmitted via a WiFi-6 connection to a Windows computer (as depicted in the figure) running the Rokoko Studio, a software application designed for visualizing and editing the data. Motion data is formatted as time-series and expressed as quaternions, where each different angle is expressed by using its position (a_1, a_2, a_3) and rotation (b_1, b_2, b_3, b_4) values.

Using its streaming service, the collected data are then forwarded via the same Wifi-6 connection to an Ubuntu laptop running Robot Operating System (ROS Noetic) that handles all different operations required by the Shadow Robot Dexterous hand. The received time-series data are filtered to isolate the hand motion data, sampled, and queued as input to the ML models which predict the next position of each angle in a discrete time horizon. The output of the models are fed into a Next Unit of Computing (NUC) which controls the Shadow Robot's actuators. Real-time kinematic algorithms, including inverse kinematics, ensure precise motion replication. Equipped with position sensors, force sensors, and tactile sensors, Shadow Robot provides feedback on joint angles, applied force, and surface interactions, allowing for highly accurate and responsive movement replication.

4.1 Data collection

Using the setup described above, we collected a dataset comprising of 125 samples, with each sample represented as a time series of 5 repetitions of a specific

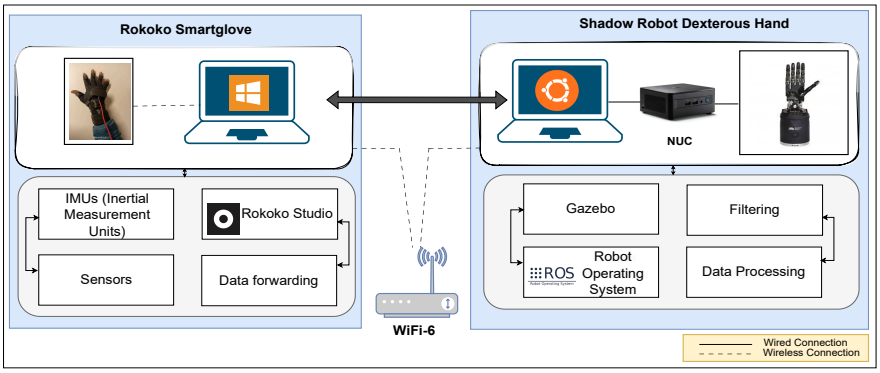


Figure 5: Robotic hand experimental testbed setup.

pose. In particular, 5 participants performed a set of 5 different poses, each repeated 5 times, so some *seasonality* is present in the data.

All experiments commence with the Shadow Robot Dexterous Hand in an open and neutral position. Then, the recorded movements are as follows:

1. **Pose 1:** Close and open the hand (without using the thumb)
2. **Pose 2:** Close the hand to form a fist, lay the thumb on the fingers, and return to the open hand position.
3. **Pose 3:** Bend the fingers one by one starting with the little finger and move towards the thumb. As soon as it is closed, the fingers are straightened one by one, going from the thumb to the little finger.
4. **Pose 4:** Close the fingers while flexing the wrist. Then, straighten the fingers while extending the wrist.
5. **Pose 5:** Close the hand, then extend the thumb, index, and middle finger.

The data was collected using the Rokoko Smartgloves (both medium and large size gloves were used). Hand motion was tracked using 7 integrated IMU and EMF sensors, with the positions of all joints streamed in real time as JSON files containing quaternion data at a frequency of 100 fps to a Windows machine running the Rokoko Studio, where it was downsampled to 10 fps. Then, the quaternions were converted to Euler angles using the relative rotation of the joints, thus resulting to 24 joint angles that were stored in a dataset for training and testing the motion prediction models described in this work.

4.2 Dataset description

Each sample of the dataset consists of a multivariate time series of 24 features. Each univariate time series describes a different joint angle of the *Shadow Robot Dexterous Hand*, in degrees.

Joint Name	**Min degree**	**Max degree**
FF1	0	77
MF1	0	82
RF1	0	79
LF1	0	71
FF2	0	114
MF2	0	121
RF2	0	116
LF2	0	105
FF3	-28	84
MF3	-33	92
RF3	-44	101
LF3	-56	106
FF4	-22	14
MF4	-8	33
RF4	-47	0
LF4	-76	0
LF5	0	0
TH1	-18	50
TH2	-27	75
TH3	-63	2
TH4	-34	337
TH5	-335	159
WR1	-179	179
WR2	-179	179

Table 2: Observed motion range of collected data. Min/Max degree indicates the minimum and maximum degree of the movement range for each joint.

The amplitude value of the time series is quantized to the units (that is, the values are integers), and their observed range is presented in Table 2. The time stamps are uniformly distributed with a separation of 100 ms, and the length of each sample (that is, the overall time duration of the hand movement poses) varies along the dataset.

We describe every mth sample as a collection of T_m tuples of time stamps and data points. Formally, the mth sample is a sequence $\{(x_t, \tilde{\boldsymbol{y}}_t^{(m)}) : t = 1, \ldots, T_m\}$, where x_t denotes the tth time stamp, and $\tilde{\boldsymbol{y}}_t^{(m)}$ the corresponding data point, which is a vector of $F = 24$ components (one for each univariate time series).

Henceforth, for the sake of simplicity in the notation, we will drop the superindex $^{(m)}$ when we are not referring to any particular mth sample or when it is clear from the context.

4.3 Data preprocessing

The range of the time series values (i.e., joint angle values in degrees) has been normalized by dividing the series values by 360. This preprocessing step has two main benefits: i) It helps the prediction models to focus on the fluctuations (relative changes) rather than on the magnitude of the values; ii) It reduces numerical issues during training related to vanishing or exploding gradients. In this way, the resulting data points, i.e., $\boldsymbol{y}_t := \tilde{\boldsymbol{y}}_t \div 360$, are vectors in $\mathbb{R}^{24}$, and the range of its components is (roughly) a unit.

On the other hand, the data range observed for the whole dataset (all hand motion trajectories) is shown in Table 2. The knowledge of this observed range can be used (being careful not to leak any test data) in the data preprocessing, or even be used to tailor the optimization cost in Section 3.2.3. However, this is out of the scope of this technical report and is left for future work.

5 Evaluation of Motion Prediction Analysis

In these experiments, we have trained four models: a linear, stable linear, GRU-based, and LSTM-based model. Each one of the models has been trained over five different data settings, thus leading to five different model versions. Specifically, one data setting for the first pose, another one for the two first poses, and so on until the data setting includes all five poses.

For each experiment, we randomly divide the dataset in a training-validation-test proportion of 60-20-20 percent. We have used a batch size of 5 regardless of the dataset size. Lastly, the linear and RNN-based models are trained for 150 and 500 epochs, respectively.

5.1 Hyperparameter optimization

For the hyperparameter optimization, we have used a random search approach [Bischl et al., 2023]. We have conducted 20 trials per model. In every trial, we train the model for 1 person and 1 pose (chosen randomly) to reduce the

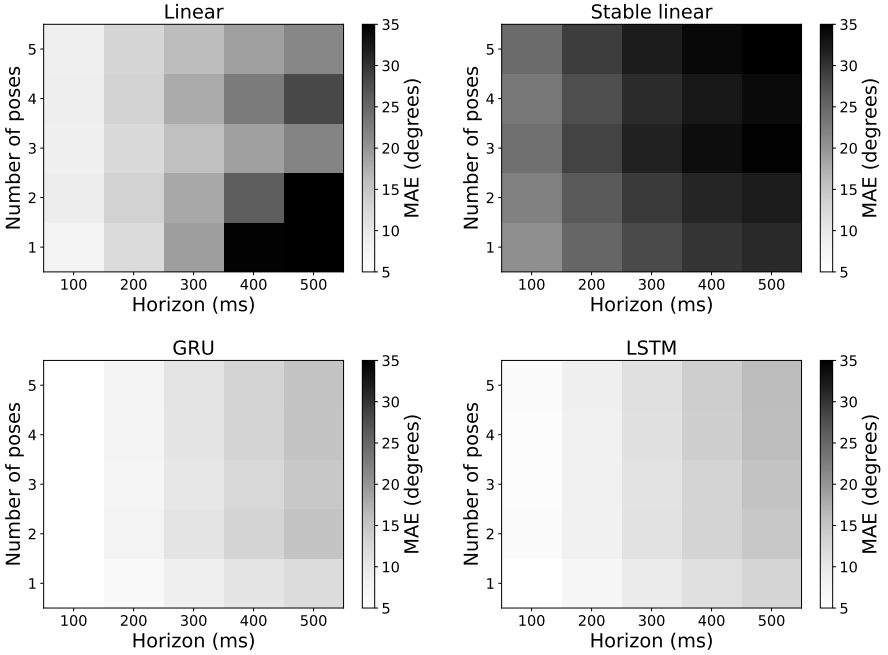


Figure 6: Heatmap plot of the MAE incurred for various poses and prediction horizons. All the models, except the linear one, increase their error with the number of poses and prediction horizon. This is probably because acute overfitting is promoted in simpler settings (less diverse poses) leading to instability.

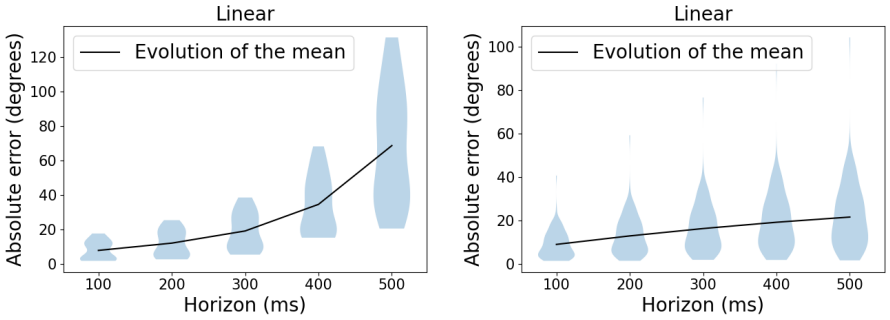
computation overhead. Then, we choose the hyperparameter configuration with the lowest validation error.

One person and one pose comes down to a training set of 3 samples and a validation set of 1 sample. The batch size chosen for all trials is 1. We use Adam, with the standard configuration [Kingma, 2014], as the optimizer for all trials. The linear models are trained for 200 epochs, while the RNN-based ones are trained for 500 epochs.

5.1.1 Search space

For the linear models, the search space consists of the model lag that can take natural numbers between 1 and 10, and the learning rate, chosen uniformly randomly between 0.1 and 0.0001.

For the RNN-based models, the search space includes: i) the size of the hidden state, a value chosen randomly between 16, 32, and 48, ii) the number of layers, randomly selected to be 1, 2, or 3, iii) the learning rate, in this case uniformly randomly chosen between 0.001 and 0.00001, and iv) the dropout rate, uniformly randomly chosen between 0 and 0.3.



(a) 1 pose.

(b) 5 poses.

Figure 7: Unstable forecasting dynamics of the linear model due to overfitting. The left graph corresponds to a Linear model trained for 1 pose, while the right one displays a Linear model trained for all 5 poses. As you can see, the model trained over a more reduced and less diverse training dataset (left) shows a more prominent unstable behavior.

5.1.2 Best hyperparameter configurations found

These are the best hyperparameter configurations we have found:

- Linear: lag of 1 and learning rate of ~ 0.032.
- Stable linear: lag of 2 and learning rate of ~ 0.079.
- GRU-based: hidden state size of 32, 2 layers, a learning rate of ~ 0.00078, and a dropout rate of ~ 0.035.
- LSTM-based: hidden state size of 48, 1 layer, a learning rate of ~ 0.00027, and no dropout.

5.2 Results and discussion

Our experiments show that the RNN-based forecasting models trained for all 5 poses can achieve an MAE of approximately 15 degrees over all joints for a prediction horizon of 500 ms. We also observe that, in general, the error noticeably increases as we add more poses, see Fig. 6. Therefore, it is reasonable to expect a similar extrapolated behavior as we further increase the number of poses, potentially up to free movement. This can be solved by enlarging the parameter size of the model (thus, its model complexity and approximation capability) but, this does not scale well computationally (i.e., training and inference computational resources grow superlinearly) with RNN-based models.

On the other hand, the Linear model performs better in terms of the MAE as the number of poses increases, see Fig. 6 again. We believe that this counter-intuitive behavior is the result of the limited number of poses in the training set;

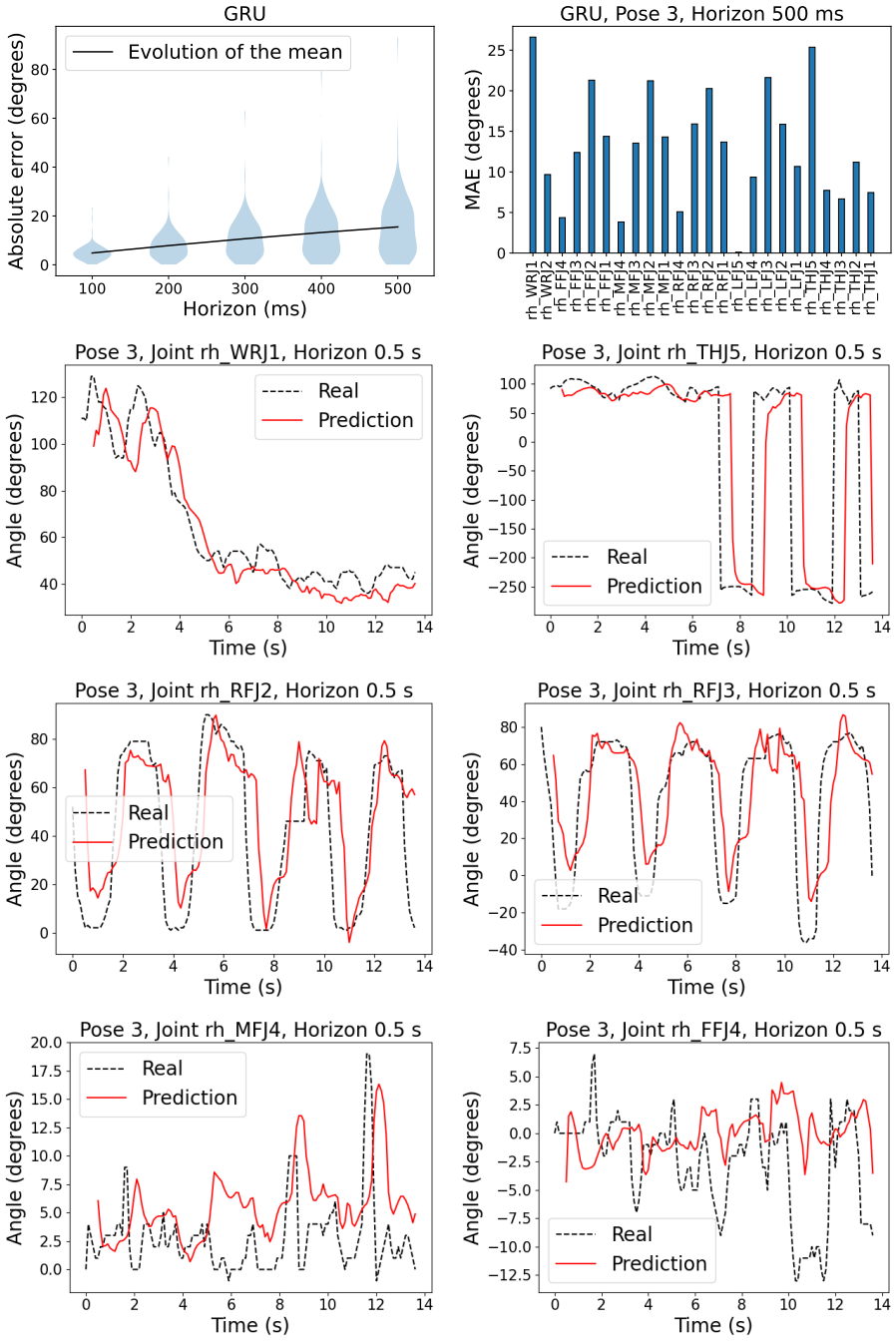


Figure 8: Example of motion prediction. Reconstruction of a test sample with the GRU-based model trained for all poses.

	Spectral radius $\rho(\mathbf{\Gamma})$	
Poses	Linear	Stable Linear
1	~ 1.73	~ 0.78
1,2	~ 1.29	~ 0.80
1,2,3	~ 1.12	~ 0.79
1,2,3,4	~ 1.15	~ 0.81
1,2,3,4,5	~ 1.03	~ 0.75

Table 3: Comparison of the spectral radius of the companion matrices of the Linear models trained over datasets with different amounts of poses. A linear model producing a stable chain of forecasts requires a spectral radius strictly less than 1.

thus, less diverse, allows for higher odds of overfitting and such overfitting may lead to unstable forecasting dynamics. This instability can be observed in the evolution of the MAE as we increase the horizon, see Fig. 7, and in the spectral radius of the companion matrix of the trained models, see Table 3. For varying prediction horizons, this unstable behavior can be addressed by constraining the feasible set of companion matrices that can be learned to only those that are stable. However, this increases the MAE incurred at shorter-sighted horizons, as shown in Fig. 6. In summary, forecasting models can benefit from regularization techniques [Kukačka et al., 2017] (to reduce the risk of overfitting) when the complexity of the model exceeds the diversity of the dataset.

Finally, it is important to take into account the effects of the optimization cost, see Section 3.2.3, in the distribution of errors. Even though the mode (the most likely MAE) of the distribution of errors is to some extent regularly below the MAE, there is still high variability in the error between some joints. This can be seen in Fig. 8.

In this case, the unevenly incurred errors are mostly due to the fact that the optimization cost does not take into account the effective range of motion of each joint. Therefore, tailored optimization costs for specific tasks, or adaptive costs for more generic activities, can help to reduce the variability of the error. Thus, reducing the risk of suffering from isolated error outliers. In this regard, a thorough design of the optimization cost may benefit future works.

6 Research directions and future work

In this section, we showcase ongoing work that complements the presented results and that will be reported and completed in future work.

6.1 Task-specific motion prediction

Another important topic is to study how the prediction error distribution (of the joint angles) influences certain different tasks. In particular, we are focusing

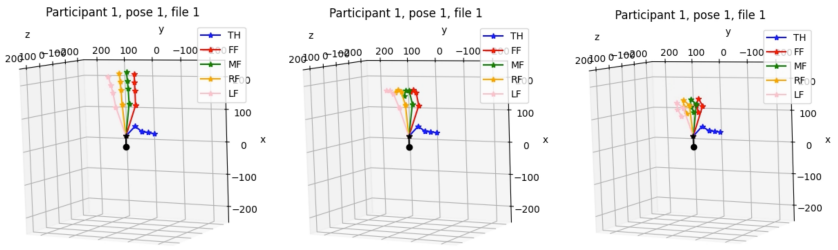


Figure 9: Forward kinematics for the Shadow Robot, from joint angles to a hand representation. The image shows three frames for a given motion hand trajectory.

on those tasks in which a high accuracy in the position of the tips is desired.

This work mainly covers the following research topics:

1. A probabilistic modeling of the prediction error distribution (of the joint angles). Accordingly, the training loss is typically chosen to minimize some metric over the prediction error, e.g., the Euclidean norm for a normally distributed prediction error.

2. A model for the forward kinematics of the Shadow Robot. This allows us to compute the position of the tips from the joint angles. Concretely, we use the Denavit and Hartenberg notation and methodology.

3. Studying how the prediction error distribution propagates through the forward kinematics transformation. This knowledge could be used to better tailor the prediction model for certain tasks. For this, we carry out first some empirical analysis, which will be followed by a theoretical analysis in our future work. Another goal is to generalize the presented architectures to allow them to train with respect to the position of the tips (rather than only with respect to the joint angles) in order to validate the results of our analyses experimentally.

6.1.1 A probabilistic modeling of the prediction error

Let us denote by $\boldsymbol{y}_1, \boldsymbol{y}_2, \ldots, \boldsymbol{y}_T$ the sequence of joint angles of a hand motion trajectory and by $\boldsymbol{o}_2, \ldots, \boldsymbol{o}_T$ the sequence of predicted joint angles.

Then, the prediction error at the tth time stamp can be modeled as $\epsilon_t = \boldsymbol{y}_t - \boldsymbol{o}_t$. Identifying how this error is distributed can help us generate better estimates (under certain metric, e.g., the lowest mean squared error (MSE)) for the joint angles and for any other magnitude obtained through transformation, e.g., the finger tip position with respect to the base of the wrist.

6.1.2 Forward kinematics of the Shadow Robot

The term forward kinematics refers to the transformation that determines the position (and derivatives, e.g., velocity) and orientation of an end-effector, e.g.,

i	$a_{i,i+1}$	$\alpha_{i,i+1}$	$\theta_{i,i+1}$	$d_{i,i+1}$
0	32	$\pi/2$	θ_{WRJ2}	0
1	0	$\pi/4$	$\pi/2 + \theta_{\text{WRJ1}}$	0
2	0	$-\pi/2$	θ_{THJ5}	44.7
3	38	$\pi/2$	$\theta_{\text{THJ4}} - \pi/2$	0
4	0	$-\pi/2$	θ_{THJ3}	0
5	32	$-\pi/2$	θ_{THJ2}	0
6	27.5	$\pi/2$	θ_{THJ1}	0

Table 4: DH parameters for the thumb (code: TH). The distances are in millimeters, and the angles are in radians.

i	$a_{i,i+1}$	$\alpha_{i,i+1}$	$\theta_{i,i+1}$	$d_{i,i+1}$
0	34	$\pi/2$	θ_{WRJ2}	0
1	FF:95 MF:99	$-\pi/2$	θ_{WRJ1}	FF:34 MF:12
2	0	$\pi/2$	θ_{FJ4}	0
3	45	0	θ_{FJ3}	0
4	25	0	θ_{FJ2}	0
5	26	$-\pi/2$	θ_{FJ1}	0

Table 5: DH parameters for the index (code: FF) and middle fingers (code: MF). The distances are in millimeters, and the angles are in radians.

i	$a_{i,i+1}$	$\alpha_{i,i+1}$	$\theta_{i,i+1}$	$d_{i,i+1}$
0	34	$\pi/2$	θ_{WRJ2}	0
1	RF:95 LF:86	$\pi/2$	θ_{WRJ1}	RF:-12 LF:-34
2	0	$-\pi/2$	θ_{FJ4}	0
3	45	0	θ_{FJ3}	0
4	25	0	θ_{FJ2}	0
5	26	$-\pi/2$	θ_{FJ1}	0

Table 6: DH parameters for the ring (code: RF) and little fingers (code: LF). The distances are in millimeters, and the angles are in radians.

the tip of a finger of the Shadow Robot, based on the known joint angles and parameters of its kinematic chain. In essence, it calculates the position of the end-effector given the joint angles.

The Denavit–Hartenberg (DH) parameters Denavit and Hartenberg [1955] are used, along with the DH convention, for attaching reference frames to the links of a spatial kinematic chain. The DH convention can be summarized in the following three rules: **i)** the joints are numbered consecutively, **ii)** every ith basis origin $(\hat{x}_i, \hat{y}_i, \hat{z}_i)$ is fixed in the link that has the joints $i-1$ and i, and **iii)** every joint axis $\hat{x}_i$ is perpendicular to $\hat{z}_{i-1}$ and $\hat{z}_i$. Regarding the DH parameters:

- $a_{i,i+1}$ is the distance, along $\hat{x}_{i+1}$, from $\hat{z}_i$ to $\hat{z}_{i+1}$
- $\alpha_{i,i+1}$ is the angle from $\hat{z}_i$ to $\hat{z}_{i+1}$ (as seen from $\hat{x}_{i+1}$)
- $\theta_{i,i+1}$ is the angle from $\hat{x}_i$ to $\hat{x}_{i+1}$ (as seen from $\hat{z}_i$)
- $d_{i,i+1}$ is the distance, along $\hat{z}_i$, from $\hat{x}_i$ to $\hat{x}_{i+1}$

We have constructed the DH parameters of the Shadow Robot from the official blueprints in `https://shadow-robot-company-dexterous-hand.readthedocs-hosted.com/en/latest/user_guide/md_kinematics.html`, and we have summarized them in Table 4, 5, and 6.

The forward kinematics transformation of a serial chain of $L \in \mathbb{N}$ links, with DH parameters $\tau_{i,i+1} = (a_{i,i+1}, \alpha_{i,i+1}, \theta_{i,i+1}, d_{i,i+1})$ for $i \in \{0, \dots, L-1\}$, is given by:

$$ {}^0T_L = \prod_{i=0}^{L-1} T(\tau_{i,i+1}), \tag{4} $$

where

$$ T(\tau_{i,i+1}) = \begin{bmatrix} cos(\theta_{i,i+1}) & -cos(\alpha_{i,i+1})sin(\theta_{i,i+1}) & sin(\alpha_{i,i+1})sin(\theta_{i,i+1}) & a_{i,i+1}cos(\theta_{i,i+1}) \\ sin(\theta_{i,i+1}) & cos(\alpha_{i,i+1})cos(\theta_{i,i+1}) & -sin(\alpha_{i,i+1})cos(\theta_{i,i+1}) & a_{i,i+1}sin(\theta_{i,i+1}) \\ 0 & sin(\alpha_{i,i+1}) & cos(\alpha_{i,i+1}) & d_{i,i+1} \\ 0 & 0 & 0 & 1 \end{bmatrix}. \tag{5} $$

In this way, we can compute the position of the end-effector $\boldsymbol{e}_L$, e.g., the tip of the finger, with respect to the position of the basis of the first link $\boldsymbol{e}_0$, e.g., the wrist, as:

$$ \boldsymbol{e}_L = {}^0T_L \, \boldsymbol{e}_0. \tag{6} $$

Figure 9 shows a representation of the forward kinematics over a motion hand trajectory of the Shadow Robot.

6.1.3 Prediction error propagation

Having a model for the forward kinematics, see Sec. 6.1.2, allows us to study (theoretically and empirically) how the prediction error distribution propagates when estimating the position of the finger tips of the Shadow Robot.

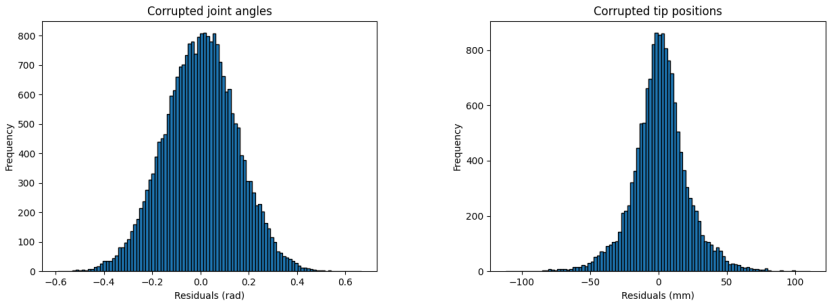


Figure 10: Example of propagation of the prediction error distribution over the joint angles using the forward kinematics of the Shadow Robot. This transformation shows our prediction error distribution over the position of the fingertips. For illustration purposes, the noise has been generated synthetically from a Gaussian distribution $\mathcal{N}(0, 0.15)$, with a dispersion of 0.15 rad $\simeq$ 8.6 degrees.

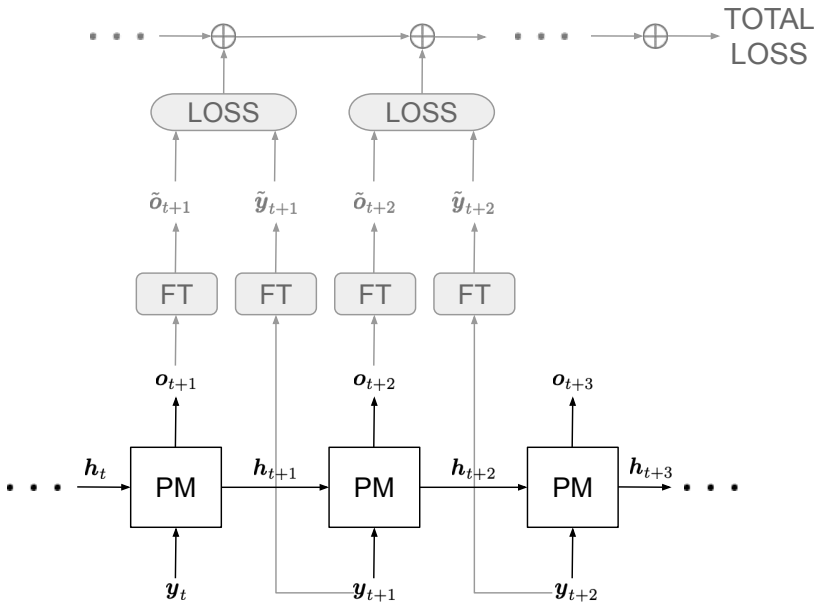


Figure 11: Modified architecture for the prediction model (PM) that includes a feature transformation (FT). The feature transformation (gray elements) is used only during training. Here, the transformed elements, e.g., from joint angles to position of the fingertips, are denoted with the tilde symbol $\sim$.

This knowledge allows us to have better control not only over the mean of the prediction error in both domains (joint angles and tip positions), but also over its dispersion.

Being able to control the distribution of the prediction error over the position of the fingertips directly from the distribution of joint angles, for example, via an adequate objective function, is convenient because we communicate with the Shadow Robot through the joint angles. This methodology can also be extended to other magnitudes (derived from the joint angles).

To further motivate the convenience of this approach, we can compare it with two alternative approaches:

Prediction in the domain of the fingertips. One could transform the motion hand trajectory from joint angles into fingertip positions and then train a model to predict directly the next fingertip position. However, this approach requires inverse kinematics to transform the predicted fingertip position into joint angles when communicating with the Shadow Robot.

A training loss that incorporates the forward kinematics. The forward kinematics can be incorporated as an intermediate step in the already presented architectures. However, our preliminary results show a noticeable increase in computation time. On the other hand, the gradient computed from the back propagation over the forward dynamics does not appear to be particularly affected by the consecutive matrix transformations, e.g., a vanishing gradient, and the predictor model can be trained as usual.

6.2 Transformer-based Motion prediction

In this report, we have dealt with predetermined hand movements, or poses, for simplicity. However, the intricate, high-dimensional nature of *free* hand movements and their (highly likely) non-linear spatio-temporal correlations across numerous joints still present significant challenges regarding modeling and prediction.

Current advances in the deep learning community promote transformer-based architectures Vaswani et al. [2017] as a compelling choice for multivariate time series prediction, and more specifically, for human motion Mao et al. [2019] or hand motion prediction. Transformer-based architectures inherently possess the capability to capture long-range dependencies and global interactions across an entire sequence, regardless of (temporal) distance. This feature has the potential to capture coordinated patterns in hand gestures, where a movement at one joint can influence many others both synchronously, i.e., at the same time, and asynchronously, i.e., at different time instants. Moreover, transformers can attend to all past information simultaneously (rather than through a usually distorted memory), offering a powerful framework to discern subtle but critical relationships within multivariate time series data, paving the way for more accurate and robust hand motion predictions.

On the other hand, transformer-based architectures still face several notable challenges for their application to multivariate time series prediction. Beyond the limitations already discussed in Sec. 2.2.2 such as multivariate setting con-

straints, temporal information losses, and usually requiring large amounts of data for training, their primary caveat is arguably their computational intensity in the training process. This is because their self-attention mechanism (at least in the standard form) scales quadratically with both sequence length and number of time-series, rendering them expensive computationally for long sequences or real-time applications.

In our future work, we will conduct experiments to evaluate transformer-based architectures (including optimization approaches to alleviate their complexity) for hand movement trajectory prediction, and will compare it with the rest of approaches considering the performance-complexity trade-offs.

7 Conclusion

We show that the available off-the-shelf multivariate time series forecasting models we have tested, may struggle to deliver accurate predictions (depending on the task) at certain time horizons. Moreover, these models have scalability issues and suffer accuracy losses as the number of poses (potentially infinite) increases.

Our future work may be inspired by these experimental observations. We believe that parallelizable architectures (in training and inference) should be used to alleviate the scalability issues. It is also important to exploit the nature and nuances of the motion prediction problem to increase the prediction accuracy further. For example, this can be done by designing new prediction models that can effectively incorporate additional information complementary to the motion trajectory such as brain signals. Furthermore, in real applications, there is a need to design flexible models that can learn online and adapt across various subjects, with streaming data and different types of non-stationarities.

References

Jack B Kuipers. *Quaternions and rotation sequences: a primer with applications to orbits, aerospace, and virtual reality*. Princeton university press, 1999.

Milan Groshev, Javier Sacido, and Jorge Martín-Pérez. Foreco: a forecast-based recovery mechanism for real-time remote control of robotic manipulators. In *Proceedings of the SIGCOMM'22 Poster and Demo Sessions*, pages 7–9. IEEE, 2022.

J Ernesto Solanes, Adolfo Muñoz, Luis Gracia, Ana Martí, Vicent Girbés-Juan, and Josep Tornero. Teleoperation of industrial robot manipulators based on augmented reality. *The International Journal of Advanced Manufacturing Technology*, 111:1077–1097, 2020.

Matteo Rubagotti, Tasbolat Taunyazov, Bukeikhan Omarali, and Almas Shintemirov. Semi-autonomous robot teleoperation with obstacle avoidance via model predictive control. *IEEE Robotics and Automation Letters*, 4(3): 2746–2753, 2019.

Chao Hu, Max Qinghu Meng, Peter Xiaoping Liu, and Xiang Wang. Visual gesture recognition for human-machine interface of robot teleoperation. In *Proceedings 2003 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2003)(Cat. No. 03CH37453)*, volume 2, pages 1560–1565. IEEE, 2003.

Wen Qi, Salih Ertug Ovur, Zhijun Li, Aldo Marzullo, and Rong Song. Multi-sensor guided hand gesture recognition for a teleoperated robot using a recurrent neural network. *IEEE Robotics and Automation Letters*, 6(3):6039–6045, 2021.

Insaf Ajili, Malik Mallem, and Jean-Yves Didier. Gesture recognition for humanoid robot teleoperation. In *2017 26Th IEEE international symposium on robot and human interactive communication (RO-MAN)*, pages 1115–1120. IEEE, 2017.

Qing Gao, Zhaojie Ju, Yongquan Chen, Qiwen Wang, and Chuliang Chi. An efficient rgb-d hand gesture detection framework for dexterous robot hand-arm teleoperation system. *IEEE Transactions on Human-Machine Systems*, 53(1):13–23, 2022.

Simon Chamorro, Jack Collier, and François Grondin. Neural network based lidar gesture recognition for realtime robot teleoperation. In *2021 IEEE international symposium on safety, security, and rescue robotics (SSRR)*, pages 98–103. IEEE, 2021.

Robert G Brown and Richard F Meyer. The fundamental theorem of exponential smoothing. *Operations Research*, 9(5):673–685, 1961.

Helmut Lütkepohl. New introduction to multiple time series analysis. *NY: Springer*, 2005.

Sean J Taylor and Benjamin Letham. Forecasting at scale. *The American Statistician*, 72(1):37–45, 2018.

S Hochreiter. Long short-term memory. *Neural Computation MIT-Press*, 1997.

Shaojie Bai, J Zico Kolter, and Vladlen Koltun. An empirical evaluation of generic convolutional and recurrent networks for sequence modeling. *arXiv preprint arXiv:1803.01271*, 2018.

David Salinas, Valentin Flunkert, Jan Gasthaus, and Tim Januschowski. Deepar: Probabilistic forecasting with autoregressive recurrent networks. *International journal of forecasting*, 36(3):1181–1191, 2020.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems*, 32, 2019.

Azul Garza and Max Mergenthaler-Canseco. Timegpt-1. *arXiv preprint arXiv:2310.03589*, 2023.

Kashif Rasul, Arjun Ashok, Andrew Robert Williams, Arian Khorasani, George Adamopoulos, Rishika Bhagwatkar, Marin Biloš, Hena Ghonia, Nadhir Hassen, Anderson Schneider, et al. Lag-llama: Towards foundation models for time series forecasting. In *R0-FoMo: Robustness of Few-shot and Zero-shot Learning in Large Foundation Models*, 2023.

Abdul Fatir Ansari, Lorenzo Stella, Caner Turkmen, Xiyuan Zhang, Pedro Mercado, Huibin Shen, Oleksandr Shchur, Syama Sundar Rangapuram, Sebastian Pineda Arango, Shubham Kapoor, et al. Chronos: Learning the language of time series. *arXiv preprint arXiv:2403.07815*, 2024.

Lu Han, Xu-Yang Chen, Han-Jia Ye, and De-Chuan Zhan. Softs: Efficient multivariate time series forecasting with series-core fusion. *arXiv preprint arXiv:2404.14197*, 2024a.

Musleh Alharthi and Ausif Mahmood. xlstmtime: Long-term time series forecasting with xlstm. *AI*, 5(3):1482–1495, 2024.

Wenyong Han, Tao Zhu, Liming Chen, Huansheng Ning, Yang Luo, and Yaping Wan. Mcformer: Multivariate time series forecasting with mixed-channels transformer. *IEEE Internet of Things Journal*, 2024b.

Vijay Ekambaram, Arindam Jati, Pankaj Dayama, Sumanta Mukherjee, Nam H Nguyen, Wesley M Gifford, Chandra Reddy, and Jayant Kalagnanam. Tiny time mixers (ttms): Fast pre-trained models for enhanced zero/few-shot forecasting of multivariate time series. *CoRR*, 2024.

Ailing Zeng, Muxi Chen, Lei Zhang, and Qiang Xu. Are transformers effective for time series forecasting? In *Proceedings of the AAAI conference on artificial intelligence*, pages 11121–11128, 2023.

Douglass B Lee Jr. Requiem for large-scale models. *Journal of the American Institute of planners*, 39(3):163–178, 1973.

Aryan Jadon, Avinash Patil, and Shruti Jadon. A comprehensive survey of regression-based loss functions for time series forecasting. In *International Conference on Data Management, Analytics & Innovation*, pages 117–147. Springer, 2024.

Bernd Bischl, Martin Binder, Michel Lang, Tobias Pielok, Jakob Richter, Stefan Coors, Janek Thomas, Theresa Ullmann, Marc Becker, Anne-Laure Boulesteix, et al. Hyperparameter optimization: Foundations, algorithms, best practices, and open challenges. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, 13(2):e1484, 2023.

Diederik P Kingma. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Jan Kukačka, Vladimir Golkov, and Daniel Cremers. Regularization for deep learning: A taxonomy. *arXiv preprint arXiv:1710.10686*, 2017.

Jacques Denavit and Richard S Hartenberg. A kinematic notation for lower-pair mechanisms based on matrices. 1955.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

Wei Mao, Miaomiao Liu, Mathieu Salzmann, and Hongdong Li. Learning trajectory dependencies for human motion prediction. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 9489–9497, 2019.