MIntNet: Rapid Motion Intention Forecasting of Coupled Human-Robot Systems With Simulation-to-Real Autoregressive Neural Networks

John Atkins and Hyunglae Lee, *Member, IEEE*

***Abstract*—This letter describes the use of a simulation-to-real training pipeline using autoregressive neural networks (MIntNet) for coupled-human robot motion intention prediction. Using only general prior knowledge about the interaction task, a large simulation dataset was generated and used to train a multi-output variation of the classic autoregressive model. The network used an encoding-decoding method to construct condensed representations of the coupled system kinematics over a sequence of time windows and generated their condensed latent representations to predict multiple sequences of the future system states. This method was then tested on 10 real human subjects' data for the interaction task and the simulation-to-real generalization performance was evaluated for the proposed network along with alternative implementations of standard multilayered perceptron, convolutional, and long-short term memory based networks. Results show the proposed network has better generalization performance compared to the alternatives, capable of closely predicting positions during fast motion along non-constant curvatures subject to low-frequency disturbances. The MIntNet was able to accurately predict future positions in a 200 ms window with errors of 3.1 ± 4.8 mm averaged over the prediction window with inference times of 0.26 ± 0.44 ms. Performance was higher for short range predictions with errors over the time window growing as 2.3 ± 3.4 mm at 50 ms, 2.4 ± 4.4 mm at 100 ms, and 5.5 ± 6.7 mm at 200 ms. Together these properties allow for agile predictions of motion intention that can be used to inform assistive control policies for enhanced collaborative control of coupled human-robot systems.**

***Index Terms*—Intention recognition, physical human-robot interaction, modeling and simulating humans.**

I. Introduction

MOTION intention prediction in the context of coupled physical human-robot interaction (pHRI) describes the problem of inferring the human's desired motion over a suitably long time horizon using only information measured through the interaction port between a robot's end effector and the human operator [1]. This problem is widely seen in the domains of assistive exoskeletons and other wearable devices where the robotic system has limited ability to observe the human operator's full body motion and must rely on sensing such as torque/force sensors, inertial measurement units, and joint encoders in order to infer the user's intended motion [2].

This problem is amplified for assistive devices designed to follow and aid a user's arm motion due to the possibility of sudden, fast motion of the hands or forearms not seen in repetitive, slow motion seen in walking or trunk movements. Additionally, interaction forces measured from a coupled interface such as a handle or arm brace are subject to intermittent, oscillations in the 0 to 15 Hz frequency range - often referred to as tremors [3] - that cannot be filtered without significant loss of system bandwidth required to quickly assist the user.

Common approaches for predicting motion intention of coupled human-robot systems often employ regression methods such as fitting polynomial or time-series forecasting models like autoregressive integrated moving average (ARIMA) models to a sequence of past system states for prediction of future states [4], [5]. Work in adaptive control often uses Radial Basis Function neural networks (RBFNNs) for intention and reference trajectory estimation in conjunction with model-based adaptive methods [6], [7]. Recent work has also extended adaptive methods with meta-learning methods for human movement prediction while quickly adapting to unseen trends in new users [8], [9].

Other data driven methods operating on electromyography (EMG) signals to detect motion intention are also used [10], [11], but the requirement of needing to place EMG sensors on a subject and then be properly calibrated limits their use in many practical applications such as in outdoor environments or industrial use.

The problem of coupled human-robot motion prediction is in essence a multi-dimensional time-series forecasting problem. Model-based prediction methods as seen in model predictive control formulations are widely used in robotics [12], [13], but they do rely on knowing sufficient information about the system of which the model depends. For human-robot interaction cases, there is often a lack of knowledge about the true state of the human's behavior that is available during high speed motion assistance cases. This motivates observation-based approaches for modeling how past system states and measurements are

Manuscript received 19 February 2023; accepted 29 July 2023. Date of publication 18 August 2023; date of current version 25 August 2023. This letter was recommended for publication by Associate Editor S. Farokh Atashzar and Editor J.-H. Ryu upon evaluation of the reviewers' comments. This work was supported by the National Science Foundation under Grant 1925110. *(Corresponding author: Hyunglae Lee.)*

This work involved human subjects or animals in its research. Approval of all ethical and experimental procedures and protocols was granted by Institutional Review Board of Arizona State University as STUDY under Application No. 00010123.

The authors are with the School for Engineering of Matter, Transport and Energy and the Neuromuscular Control and Human Robotics Laboratory, Arizona State University, Tempe, AZ 85287 USA (e-mail: john.t.atkins@asu.edu; hyunglae.lee@asu.edu).

Digital Object Identifier 10.1109/LRA.2023.3306646

predictive of future system states that would be robust to low-frequency disturbances that commonly appear in real human motion data [3].

Over the past decade, there has been a boom of research into using new data driven methods to fit time-series data such as Deep AutoRegressive Networks (DARNs) [14], temporal convolutional networks (TCNs) [15], dynamical variational autoencoders (DVAEs) [16], and temporal fusion transformers (TFTs) [17] for learning the dynamics of sequences and for generative forecasting [18]. These methods show amazing performance learning complex time-series relationships for difficult problems. However, they feature complex networks in order to capture the dynamics correctly which may be inefficient when run on limited computational hardware commonly seen on wearable robotics. These systems would greatly benefit from efficient estimation of parameters, state forecasting, and intention classification which motivates the development of lightweight, interpretative models to tackle these problems while operating on portable hardware such as a Raspberry Pi (Raspberry Pi Foundation, Cambridge, U.K.) or a NVIDIA Jetson Nano (NVIDIA Corporation, Santa Clara, California).

This letter outlines the use of a lightweight neural network extending the traditional vector autoregression model that was capable of sub-millisecond inference times. The model is tailored for short term motion intention prediction and features high generalization performance for use in agile coupled human-robot systems. It used a simulation-to-real training pipeline using limited prior knowledge about the dynamics and disturbances that was able to generalize to real human subject data with only a small drop in performance.

II. Methods

A. Problem Setup

Vector autoregression (VAR) models are a form of multi-dimensional time-series statistical models used to describe the evolution of a system's state as a function of a series of M lags, or past observations of the state, that take the form of

$$\mathbf{x}_i = \mathbf{c} + A_1\mathbf{x}_{i-1} + A_2\mathbf{x}_{i-2} + \cdots + A_M\mathbf{x}_{i-M} + \mathbf{e}_t \quad (1)$$

where $\mathbf{x}_i$ is the k-dimensional state vector at time index i, $\mathbf{x}_{i-m} \forall m \in (1, M)$ are the relative state vectors, $\mathbf{c}$ is a constant term, and $\mathbf{e}_t$ is a vector of error terms with assumptions of zero mean, A_m is a $k \times k$ covariance matrix, and no correlation with errors at previous states. This form of model is useful for the application of pHRI motion prediction as it relies on observations of a series of possibly low-dimensional observable states available to a robot over a chosen sampled time window rather than fully coupled system state information at the current time during online operation.

However, if the number of M lags is relatively low over a long time window, individual relative state vectors may be sensitive to disturbances from the human. These disturbances cannot be filtered out of the stream of incoming measurement data due to loss of useful information and filtering induced time delays that lower the possible bandwidth of the assistance. One method to mitigate the effect of a short range disturbance is to use a greater number of lags with smaller spacing between the terms, however this quickly increases the dimensionality and complexity of the model.

In our method, we made the use of a learned encoding function, $\Theta(\cdot)$, that mapped a sequence of sampled state vectors over a short time window to a single latent vector $\mathbf{z}_i$ that served as a condensed representation of the system state's behavior over the sampled window. This encoder can be passed over the stream of state vectors at a larger spacing to form a sequence of latent vectors $\mathbf{z}_{i-n}$, $n \in [0, N]$, that can then form a non-linear forecasting variation of the VAR model as

$$\mathbf{z}_{i+1} = f(\mathbf{z}_0, \mathbf{z}_1, \ldots, \mathbf{z}_N) \quad (2)$$

where $\mathbf{z}_{i+1}$ is the predicted latent variable, $f(\cdot)$ is the autoregressive forecasting model, and N is the number of relative state vectors.

$$\begin{bmatrix} \mathbf{z}_{i+1} \\ \mathbf{z}_{i+2} \\ \vdots \\ \mathbf{z}_{i+M} \end{bmatrix} = \begin{bmatrix} f_1(\mathbf{z}_i, \mathbf{z}_{i-1}, \ldots, \mathbf{z}_{i-N}) \\ f_2(\mathbf{z}_i, \mathbf{z}_{i-1}, \ldots, \mathbf{z}_{i-N}) \\ \vdots \\ f_M(\mathbf{z}_i, \mathbf{z}_{i-1}, \ldots, \mathbf{z}_{i-N}) \end{bmatrix} \quad (3)$$

where M is the number of forward latent vectors to be predicted simultaneously. Expressed symbolically, the encoding function $\Theta(\cdot)$ can be written as $\Theta(\cdot) : \{\mathbf{x}_{i-nq-fu}\} \rightarrow \{\mathbf{z}_{i-n}\}, n \in [0, N], f \in [0, F]$. The forecasting functions $f_m(\cdot), m \in [1, M]$ can be composed with a decoding function $\phi_m(\cdot)$ to form a combined forecasting decoder $\Phi(\cdot) = \phi(f(\cdot))$ which maps $\Phi_m(\cdot) : \{\mathbf{z}_{i-n}\}_{n=0}^{N} \rightarrow \{\mathbf{x}_{i+mp-gv}\}$, where $\{\mathbf{z}_{i-n}\}_{n=0}^{N}$ is the input sequence of $N+1$ latent vectors, p is the spacing of the forward clusters relative to the current index i, g is the index of the G points in the output sampled window, and v is the spacing of sampled points inside of the main window.

This multi-time scale approach can be described using a pair of stencils parameterized by terms (N, q, F, u) and (M, p, G, v) representing backwards N sampled time windows with clusters of points spaced q indices from each other relative to the base sampling rate with F points inside the cluster at a spacing of u indices from each other intra-cluster points. Note this indexing method is functionally equivalent to choosing stride and dilation parameters for a 1-D convolutional kernel and the benefit of this approach is only for ease of analysis. Similarly, for the forward prediction clusters, there were M predicted clusters spaced at p indices with G points inside the cluster with intra-cluster spacing of v indices (Fig. 1(a)). This stencil approach allowed the terms for the model's inputs and outputs to be easily generated from a stream of data sampled at a higher frequency than the underlying system's movement. This is often the case for wearable devices or assistive robotics where the sampling rate of many common sensors is in the range of 100–1000 Hz, while most large scale human movement taking place at less than 10 Hz. Uneven lengths for the input and output sequences were used for this approach to allow for the selection of a long, densely sampled input window and short, sparsely sampled output window better suited for motion intention forecasting.

For this problem, the full state of the system was modeled as the relative end effector position with its own value from 200 ms

IEEE Robotics and Automation Letters (RA-L) paper, presented at ICRA 2024, Yokohama, Japan. Cite as RA-L paper.

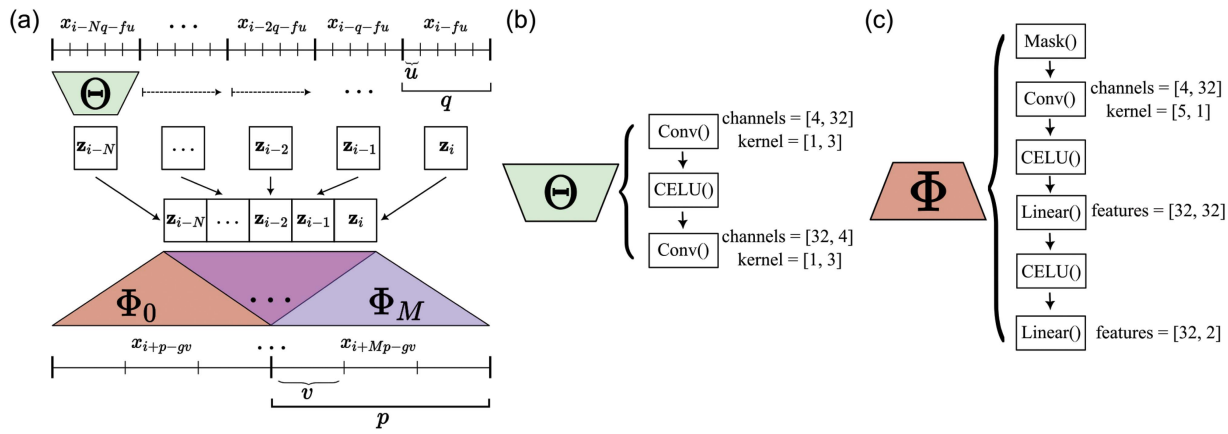

Fig. 1. (a) The proposed architecture operated as a series of convolutional kernels passing over a series of the input sequence of short time windows to form a sequence of latent vectors, $\mathbf{z}_{i-N}, \ldots, \mathbf{z}_i$. The latent sequence was then passed to M decoders in parallel which applied a learned layer mask to the sequence before decoding the masked sequence to the target section of the output sequence. (b) Internal architecture of the encoder. (c) Internal architecture of the parallel decoders.

prior, together with the end effector velocity and acceleration on a $2D$ plane. Predicting the relative positions made the network shift invariant to allow predicting positions relative to the end effector and not the arbitrary system origin. As the predicted future position was the key interest in this study, the relative positions were taken to be key variable of the output state for analysis.

The chosen stencil parameters are $(N, q, F, u) = (4, 20, 5, 5)$ and $(M, p, G, v) = (2, 20, 3, 10)$, or $4 + 1$ input clusters spaced 20 indices apart with 5 intra-cluster points spaced 5 indices apart, and 2 output clusters spaced 20 indices apart with 3 intra-cluster points spaced 10 indices apart. By satisfying the condition of $p = (G - 1)v$, this spacing causes the last point in the first output cluster and the first point in the second cluster to overlap. This allows the predicted point clusters to be translated to align with the input window or previous output cluster to compensate for undesirable biases in predictions between outputs.

B. Network Architecture

The network architecture chosen to implement the proposed MIntNet model was based on an encoding-decoding scheme (Fig. 1). The encoder was a three layer convolutional network that operates on the short-time window sampled point clusters, condensing the window into a single latent vector (Fig. 1(b)). The encoder's convolutional kernel was passed over the input sequence at a regular spacing of u indices relative to the base sampling rate to form a sequence of latent vectors. In this implementation, the input was arranged into a 3-D tensor of shape $(\dim(\mathbf{x}_{\text{input},i}), N, F)$ where $\dim(\mathbf{x}_{\text{input},i})$ is the dimension of the input state vector, N is the number of clusters, and F is the number of points per cluster, and the convolutional kernels is a 2-D kernel that operates on the last dimension of the tensor. This was done for ease of analysis and processing and an equivalent kernel could be generated from a 1-D convolutional kernel with appropriate stride and dilation parameters.

This generated the latent sequence of $(\mathbf{z}_{i-N}, \ldots, \mathbf{z}_i)$ which was then passed to M decoder functions $(\Phi_m(\cdot))$ implemented as an additive masking layer and then was passed to fully connected multi-layered perceptron (MLP) mapping the entire masked latent sequence to the points in each output cluster (Fig. 1(c)). Multiplicative masks applied in the form of self-attention layers as seen in [19] were tested as alternatives in early studies and faster training with better final losses were consistently observed when compared to a straightforward additive mask or gate for these datasets for and this problem. The mask was computed with a Hadamard product of the latent sequence and layer weights with values in the range of $[-1, 1]$ and then averaged as a straightforward method inspired by the long history of prior work of gated units such as the Gated Linear Unit [20] and the Gated Residual Network [17]. This allows weights after activations of near -1 to cancel out the input rather than scaling the entire input sequence and kept the number of learnable parameters as low as the length of the input sequence.

The motivation for decoding each target cluster in parallel was threefold. Firstly, the masking layer allowed the full latent sequence to be used for prediction while simultaneously reducing the influence of latent vectors that are less predictive of the movement for a particular time window. Secondly, it allowed the weights in each decoder to be dependent on fewer points in the output sequence to encourage faster convergence during training. Lastly, having the decoders operate independently allowed each decoder to operate in multi-threaded applications for faster inference.

C. Alternate Networks for Performance Comparison

Performance of the proposed convolutional encoder with parallel linear decoder network (MIntNet) was compared with common alternative network architectures used for similar approaches. Five alternative networks were used in this study: 1) a six-layered fully connected MLP network that reshapes layers acting on the input and output to maintain standard input and output representations (Linear); 2) a Linear encoding-decoding MLP network based on autoencoder networks that use three layers each for the encoder and decoder with similar reshaping

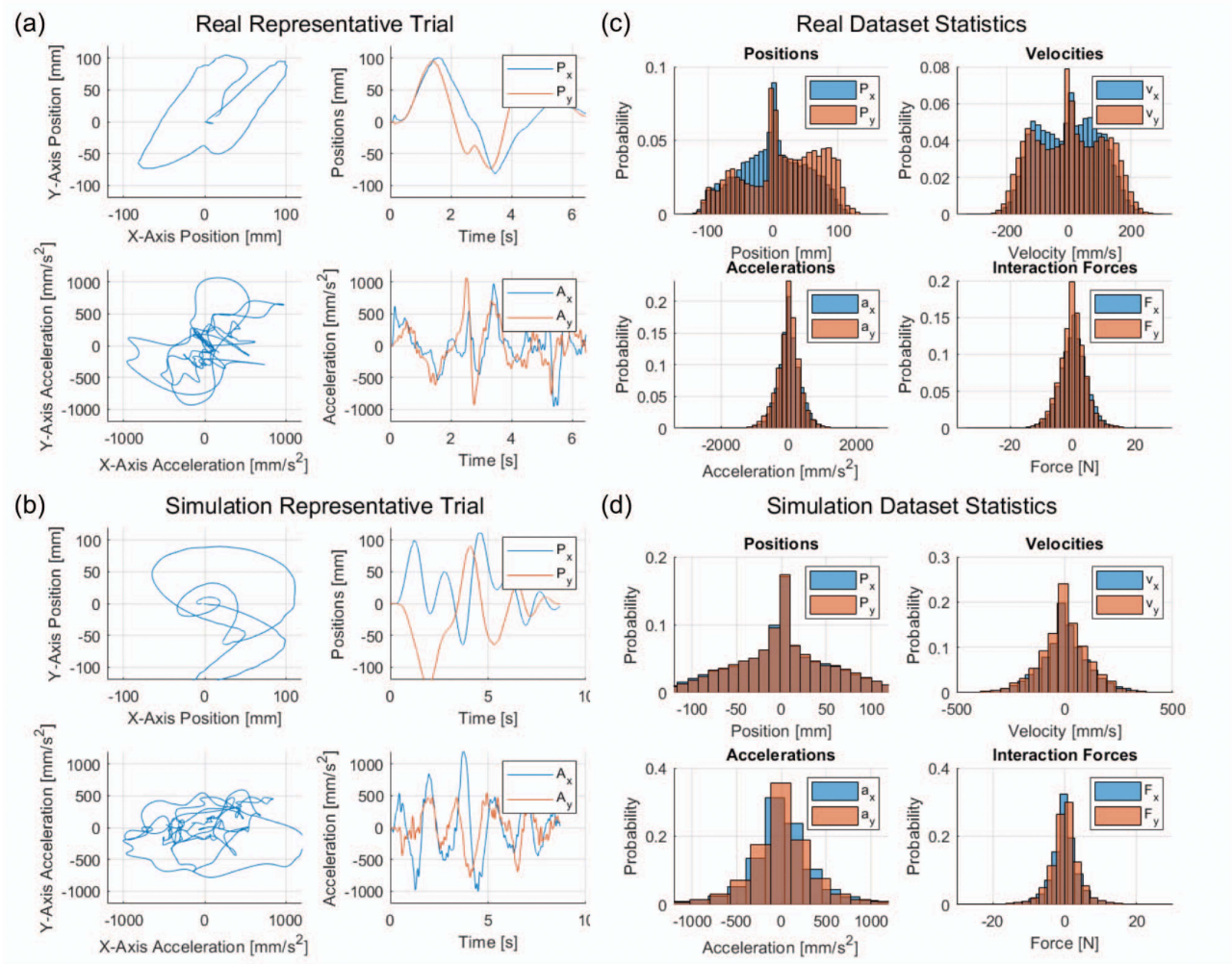

Fig. 2. Comparison of the real recorded datasets and the simulated datasets used for training. A small sample of three real human subjects' data was analysed to determine target ranges and distributions of state variables to inform creation of a simulated dataset to be used for training. From the information gathered from 6 minute of real data, roughly 45 minutes of simulated data from 3 simulated subjects and 5 trajectory and noise cases with 20 randomly generated trials was created and used for the training set. (a), (b) Representative trials of both the real datasets and simulation datasets showing both the trajectory of positions and accelerations over the workspace and over time. (c), (d) Distributions of the real and simulated datasets used to tune ranges of velocities, accelerations, and forces for simulation dataset generation.

layers (Linear-Linear); 3) a radial basis function neural network (RBFNN) [6]; 4) a convolutional encoder with parallel convolutional decoders (Conv-Conv) [21]; and 5) a convolutional encoder with long short-term memory and a final MLP layer decoder (Conv-LSTM) [22].

The base output feature or channel size was 32 features for all alternative networks. Kernel sizes of 3×1 or 1×3 were used as a base kernel size for all convolutional layers, and the LSTM featured two internal layers. Continuously Differentiable Exponential Linear Units (CELU) activation was applied between all layers, and Hardtanh activation was used for the masking layer weights. Further details for the specific parameters for all networks can be found in Table II in the Appendix. The base channel and feature size was chosen based on iterative hyperparameter tuning and 32 was found to give lower losses during training than smaller channel numbers. Larger channel numbers up to 128 were tested and the final performance was similar although training and inference times were much longer.

After the masking layers, the parallel linear decoders were implemented as a convolutional layer with kernel size equal to the sequence length to allow easy scaling of the total number of parameters with the latent sequence length. This approach had the same connections between inputs and outputs as a fully connected MLP layer with slight implementation differences followed by two MLP layers with standard settings. All networks used the same input and output tensors defined by the stencil parameters to ensure the comparison of the performances is not affected by representation of the inputs and outputs.

D. Simulation Dataset Generation

A source dataset of pHRI motion data for an arm reaching task was chosen as a reference to generate the simulated dataset (Fig. 2). In this task, a human operator guided the end effector of a 7-DOF robotic arm (LBR iiwa R820, KUKA, German) quickly through a series of five way points. The interaction was

controlled by a variable admittance controller with interaction forces measured with a 6-axis force/torque load cell (Delta IP60, ATI Industrial Automation, NC) [23], [24].

The source dataset consisted of data from 10 young, healthy individuals (age: 22–31, height: 152–188 cm, weight: 50–82 kg, sex: 9 males and 1 female, handedness: 7 right handed and 3 left handed), which was approved by the Institutional Review Board of Arizona State University (STUDY 00010123). They provided informed, written consent prior to participation.

The data was recorded with two cases, the first used constant admittance parameters of 10 kg and 10 Nm/s for inertial and damping values in all directions. The stiffness was varying between 0 N/m and 300 N/m proportional to the scalar product of the current velocity and acceleration. The second case used a stiffness of 10 N/m with equilibrium position set 3 cm on a straight line along the current velocity direction. This dataset was chosen as it represents a common scenario in pHRI where the motion of the operator's hand follows a complicated trajectory. The trajectories are over a limited workspace of a 20×20 cm^2 area in front of the user. The instantaneous velocity of the user's hand reaches speeds of over 20 cm/s, allowing the end effector to traverse the entire width of the workspace in less than one second with peak accelerations near 100 cm/s^2 in regions of high curvature.

A simulation-to-real approach was chosen to generate a large set of trajectories that contained not only motions that mimicked what was observed in the source dataset, but unseen motions such as straight line motion and curves with sudden directional changes. This was done to allow the network to observe a superset of motion trends to encourage generalization to unobserved motions that may not been seen in a limited set of only curved motion subject to disturbances (Fig. 2(c) and (d)).

Initially a set of three human subject's data was recorded for the two admittance controller cases. Ten trials were sampled at 1000 Hz lasting approximately 6 seconds each and analyzed to determine the range of velocities, accelerations, forces, and disturbances (Fig. 2(a)). The simulation uses these ranges to define simulation parameters such as way point positions and velocities, time between way points, simulated human control input, and noise or disturbance inputs for a variety of cases.

The five cases were generated to produce a diverse dataset of motions to assist with simulation-to-real generalization. The first case featured straight line motion between way point positions without noise or disturbances. The second case featured random way point velocity directions at the way point positions to create smooth arcs free of noise or disturbances. The third case was similar as the second case but included disturbance inputs. The fourth case set the velocity directions as the mean direction of the previous and next way point positions, and the fifth case was similar to the fourth but also includes disturbance inputs (Fig. 2(b)). The disturbance inputs were modeled as forces with channels independently sampled from unit normal distributions at random rates sampled from a uniform distribution between 100 ms and 200 ms. This creates disturbance forces with magnitudes and frequency responses in the range observed from the real human subject data that avoid time-dependent trends. Predictable, time-dependent trends in the simulated disturbance forces could be learned by the networks during training but would not appear in the real human subject motion and thus would harm simulation-to-real generalization performance. The simulated trials were filtered with a fifth order low-pass Butterworth filter with 10 Hz cutoff frequency before downsampling to 250 Hz for more efficient storage of the large dataset. Subjects were simulated by standard cascaded PID controllers. These gains were set first to prioritize both position and velocity reference signals equally, the second to prioritize position signals over velocity, and the third to prioritize velocity over position. The gains were selected to be stable and were tuned to mimic unrefined behavior of a human user interacting with an unfamiliar robot.

E. Training

All networks were trained on the training and validation sets pulled from the simulation dataset. The test set was formed from 10 real human subjects recorded for the two admittance controller cases for 10 trials each. Each trial lasted approximately six seconds totaling roughly 20 minutes of human subject data. The channels for the inputs and outputs were scaled to have unit variance and the input and output tensors were arranged into batches of 128.

During training, Gaussian noise, sampled independently from a zero mean with 0.05 variance, was added to the scaled inputs before applying uniform dropout with a rate of 10%. Both of these were done for regularization of the network parameters and to promote continuity of similar input sequences mapping to similar output sequences.

The Adam optimizer was used for training with an initial learning rate of $1e-4$ and was adjusted lower by a factor of $1/3$ when a plateau in the validation loss was detected. The training was ended early if the percentage improvement of the loss was less than 0.5% with a maximum epoch number of 40.

The PyTorch [25] library was used for the implementation of the networks and for training. An NVIDIA GeForce GTX 1060 GPU (NVIDIA Corporation, Santa Clara, California) was used for training. An Intel Core i7-8750H CPU (Intel Corporation, Santa Clara, California) was used for evaluation and inference times to reflect performance on most wearable robotic devices lacking GPUs. For evaluation and inference speed testing, the networks were compiled as TorchScript to reduce computational overhead from Python.

III. Results

A. Network Performance Comparison

The training and validation results showed fast convergence for all networks except the LSTM implementation (Fig. 3(a)). The evaluation on the validation set showed comparable mean errors of 1.50 ± 1.37 mm and 1.56 ± 1.53 mm for the Linear network and MIntNet, respectively, as the best two performing networks on the validation set taken from only simulated data (Fig. 3(b)).

Prediction error for both simulation and real human subjects datasets, training time, and inference time statistics were evaluated for all the tested networks. The results were averaged over

TABLE I
NETWORK COMPARISON SUMMARY

Network Model	Sim. Set Error Mean (SD) $[mm]$	Real Set Error Mean (SD) $[mm]$	Training Time per Epoch Mean (SD) $[s]$	Inference Time Mean (SD) $[ms]$
Linear	1.50 (1.37)	4.25 (4.59)	3.38 (1.43)	0.030 (0.171)
Linear-Linear	2.22 (2.19)	4.94 (4.99)	3.45 (1.13)	0.027 (0.162)
RBFNN	1.99 (2.13)	4.65 (4.93)	16.93 (4.96)	0.701 (0.455)
Conv-Conv	1.67 (1.53)	4.03 (4.70)	25.18 (16.21)	0.284 (0.450)
MIntNet	1.56 (1.53)	3.11 (4.79)	30.04 (9.24)	0.256 (0.435)
Conv-LSTM	2.94 (2.82)	4.42 (4.69)	38.47 (8.27)	1.007 (0.471)

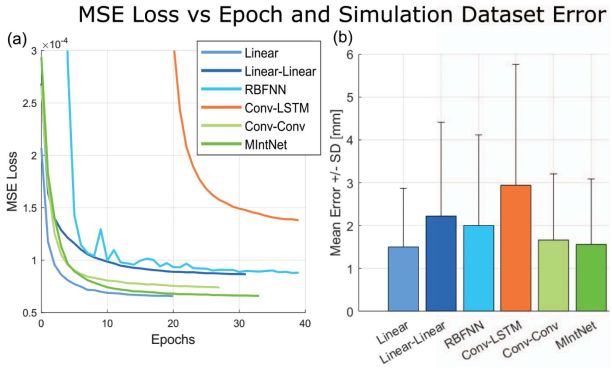

Fig. 3. (a) Validation losses for training multiple network architectures over 40 epochs. (b) Mean error averaged over all seven points in the prediction window of 200 ms for networks tested on a validation set from simulated trials.

all subjects, cases, trials, and forecasting windows (Table I). Simulation dataset errors for the Linear, Conv-Conv, and MIntNet were comparatively similar in both mean and standard deviation. When generalizing to the real dataset, MIntNet shows the best mean performance at a marginally higher standard deviation than the alternatives. While training times for all convolutional encoder networks were similar, inference times for the MIntNet network are comparable to the Linear networks without loss of generalization performance.

B. *Generalization to Real Human Data*

All the networks took in a 525 ms input window and predicted coupled-system behavior out to 200 ms in the future (Fig. 4). The network was robust to disturbances in the input and can reliably predict sharp directional changes and non-constant curvatures. Accuracy decreased as the prediction window increased due to dependence on unseen disturbances not present in the input window.

The networks were evaluated on the real subject dataset and prediction errors were averaged over subjects, cases, and trials to compare overall performance between the networks over the prediction window (Fig. 5). Of all the tested networks, the MIntNet model had consistently lower mean and standard deviations of prediction error over the entire window although the difference is more pronounced before 100 ms. For the 50 ms and 100 ms predictions, MIntNet's errors were 2.29 ± 3.40 mm and 2.42 ± 4.43 mm, respectively. For the 200 ms prediction, the error was 5.53 ± 6.74 mm and was comparable to other networks' performance.

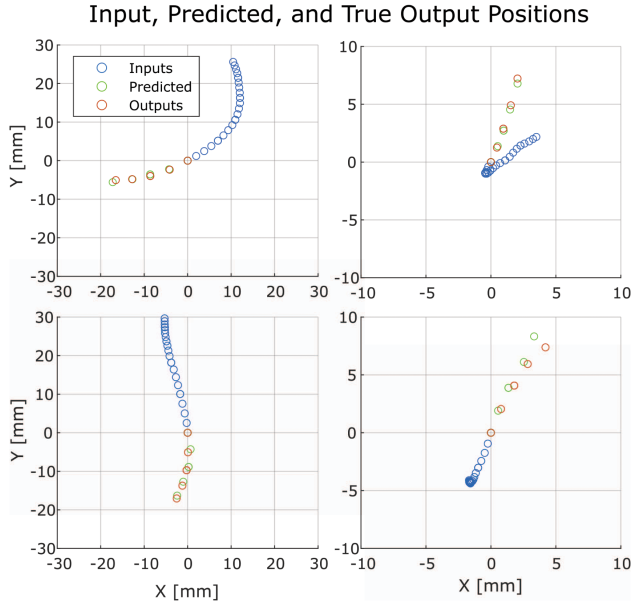

Fig. 4. Sample results of the MIntNet's performance. The network allows predicting positions out to 200 ms in the future using a 525 ms input window. These predictions worked through sudden changes in direction and for non-constant curvatures over a range of velocities using training on only simulated data. The output cluster spacing is such that the last point of the first cluster and the earliest point in the second cluster to overlap. This allows for compensation of any biases between independent outputs to ensure prediction of continuous motions.

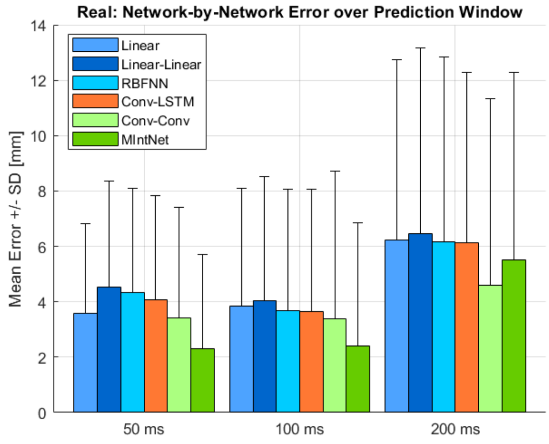

Fig. 5. Comparison of tested networks' prediction accuracy for real human subjects' data over the prediction window. All networks trained on purely simulated data show an increase in both mean and standard deviation of errors when evaluated on the 10 real human subjects dataset as expected. The proposed MIntNet model showed the smallest growth in mean error for short-range predictions with comparable performance to alternatives near the end of the prediction window.

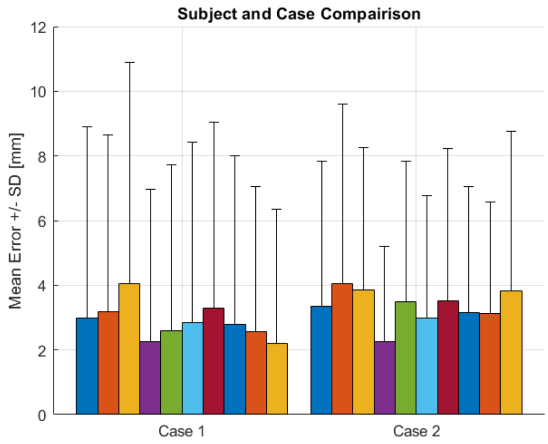

Fig. 6. MIntNet showed similar performance between both variable admittance controller schemes when tested on real subjects data while only being trained on simulated data that used a constant admittance controller parameter set.

The MIntNet model was further analysed subject-by-subject and case-by-case (Fig. 6). Grouped errors averaging over all points in the 200 ms prediction window for all 10 subjects for Cases 1 and 2 were 3.13 ± 3.13 mm and 3.70 ± 4.15 mm.

IV. Discussion

All tested networks showed mean prediction errors of less than 3 mm on the simulation set. Mean errors increased to under 5 mm for all networks showing good generalization performance for the mean prediction with far better performance for short-range position forecasts (Table I).

The high variance in the prediction errors for all networks in the real subjects datasets at first glance would suggest low performance of the generated predictions. However, the underlying issues with the variable performance originates from the datasets themselves. The trajectories the network was trained on were subjected to random disturbances that can rise and fall in the range of the prediction window. Additionally, the disturbances can start after the end of the input window. This caused a sharp increase of error for short regions of the trials that are later averaged into the reported prediction errors. This is fundamentally unavoidable in forecasting with unknown disturbance inputs, but for the purposes of predicting the user's motion intention this is only a minor concern.

Directly using forecasted relative positions in the presence of large disturbances would result in sudden, non-smooth changes in the generated assistance forces which would generate vibrations and a drop in assistance quality. The major benefit of the MIntNet's ability to predict a sequence of future states at a fast rate is that the estimates can be smoothed to extract directional intention and changes in velocity relative to the current state. This approach would not be viable when using networks with long inference times as the required computational time along would be too long to maintain high speed cycle time requirements for agile assistance.

High variance relative to the mean inference times was observed on all tested networks suggesting the presence of periodic drops in speed during inference. This was likely the result of evaluating the networks on a computer running a non-real time operating system which does not ensure consistent cycle times due to task switching to background processes. Inference times on a real-time operating system would likely yield consistent speeds close or below the reported means for a more even comparison.

This method for handling pHRI tasks allowed simulation-to-real training pipelines using minimal prior knowledge about the task itself and no prior knowledge of the human's dynamics. Arbitrary amounts of randomly generated simulated trajectories can be used to train networks as a forecasting policy with good generalization performance. This performance should be understood as the performance of a base forecasting policy that can be further trained offline on collected human subject data as needed. Together these properties show promise for the use of lightweight autoregressive neural networks for pHRI applications. Future work is planned that uses this model for predicting reference trajectories to be used with a variable admittance controller for improving assistive force generation to a human user.

Appendix

Training and network parameters were selected by iterative hyperparameter tuning during early testing and were chosen to balance convergence time and final evaluation loss (Table II).

TABLE II
Network Parameter Summary

Model	Parameter	Value
Linear	Feature Number	32
	Layer Number	6
	Activation	CELU
Linear-Linear	Feature Number	32
	Latent Features	4
	Encoder Layer Number	3
	Decoder Layer Number	3
	Activation	CELU
RBFNN	Feature Number	32
	Latent Features	12
	Gaussian RBF Number	24
	RBF Width	0.5
MIntNet	Channel Number	32
	Feature Number	32
	Latent Features	4
	Encoder Layer Number	3
	Encoder Kernel Size	$[1, 3]$
	Decoder Layer Number	3
	Decoder First Layer Kernel Size	$[5, 1]$
	Mask Features	5
	Mask Activation	Hardtanh
	Activation	CELU
Conv-Conv	Channel Number	32
	Latent Features	4
	Encoder Layer Number	3
	Encoder Kernel Size	$[1, 3]$
	Decoder Layer Number	3
	Decoder Kernel Size	$[3, 1]$
	Mask Features	5
	Mask Activation	Hardtanh
	Activation	CELU
Conv-LSTM	Channel Number	32
	Latent Features	4
	Encoder Layer Number	3
	Encoder Kernel Size	$[1, 3]$
	LSTM Features	32
	LSTM Layers	2
	Activation	CELU

ACKNOWLEDGMENT

The authors would like to thank Fatemeh Zahedi for granting us the use of the real human subjects data used in this study.

REFERENCES

[1] Y. Li et al., "A review on interaction control for contact robots through intent detection," *Prog. Biomed. Eng.*, vol. 4, no. 3, 2022, Art. no. 032004.

[2] G. Kang, H. S. Oh, J. K. Seo, U. Kim, and H. R. Choi, "Variable admittance control of robot manipulators based on human intention," *IEEE/ASME Trans. Mechatron.*, vol. 24, no. 3, pp. 1023–1032, Jun. 2019.

[3] R. J. Elble, "Central mechanisms of tremor," *J. Clin. Neuriophysiol.*, vol. 13, no. 2, pp. 133–144, 1996.

[4] T. Flash and N. Hogan, "The coordination of arm movements: An experimentally confirmed mathematical model," *J. Neurosci.*, vol. 5, no. 7, pp. 1688–1703, 1985.

[5] Y. Wang, Y. Sheng, J. Wang, and W. Zhang, "Optimal collision-free robot trajectory generation based on time series prediction of human motion," *IEEE Robot. Automat. Lett.*, vol. 3, no. 1, pp. 226–233, Jan. 2018.

[6] W. He, C. Xue, X. Yu, Z. Li, and C. Yang, "Admittance-based controller design for physical human–robot interaction in the constrained task space," *IEEE Trans. Automat. Sci. Eng.*, vol. 17, no. 4, pp. 1937–1949, Oct. 2020.

[7] Y. Huo, X. Li, X. Zhang, and D. Sun, "Intention-driven variable impedance control for physical human-robot interaction," in *Proc. IEEE/ASME Int. Conf. Adv. Intell. Mechatron.*, 2021, pp. 1220–1225.

[8] L.-Y. Gui, Y.-X. Wang, D. Ramanan, and J. M. Moura, "Few-shot human motion prediction via meta-learning," in *Proc. Eur. Conf. Comput. Vis.*, 2018, pp. 432–450.

[9] H.-s. Moon and A. H. M. Prediction, "Fast user adaptation for human motion prediction in physical human–robot interaction," *IEEE Robot. Automat. Lett.*, vol. 7, no. 1, pp. 120–127, Jan. 2022.

[10] Y. Huang et al., "Real-time intended knee joint motion prediction by deep-recurrent neural networks," *IEEE Sensors J.*, vol. 19, no. 23, pp. 11503–11509, Dec. 2019.

[11] L. Bi, G. Feleke, and C. Guan, "A review on EMG-based motor intention prediction of continuous human upper limb motion for human-robot collaboration," *Biomed. Signal Process. Control*, vol. 51, pp. 113–127, 2019.

[12] M. Faroni, M. Beschi, and N. Pedrocchi, "An MPC framework for online motion planning in human-robot collaborative tasks," in *Proc. IEEE 24th Int. Conf. Emerg. Technol. Factory Automat.*, 2019, pp. 1555–1558.

[13] C. E. García, D. M. Prett, and M. Morari, "Model predictive control: Theory and practice—A survey," *Automatica*, vol. 25, no. 3, pp. 335–348, 1989. [Online]. Available: https://www.sciencedirect.com/science/article/pii/0005109889900022

[14] K. Gregor, I. Danihelka, A. Mnih, C. Blundell, and D. Wierstra, "Deep autoregressive networks," in *Proc. 31st Int. Conf. Mach. Learn.*, 2014, pp. 1242–1250.

[15] C. Lea, R. Vidal, A. Reiter, and G. D. Hager, "Temporal convolutional networks: A unified approach to action segmentation," in *Proc. Eur. Conf. Comput. Vis.*,, 2016, pp. 47–54.

[16] L. Girin, S. Leglaive, X. Bie, J. Diard, T. Hueber, and X. Alameda-Pineda, "Dynamical variational autoencoders: A comprehensive review," *Found. Trends Mach. Learn.*, vol. 15, no. 1/2, pp. 1–175, 2021.

[17] B. Lim, S. Arık, N. Loeff, and T. Pfister, "Temporal fusion transformers for interpretable multi-horizon time series forecasting," *Int. J. Forecasting*, vol. 37, no. 4, pp. 1748–1764, 2021.

[18] P. R. Vlachas, G. Arampatzis, C. Uhler, and P. Koumoutsakos, "Multiscale simulations of complex systems by learning their effective dynamics," *Nature Mach. Intell.*, vol. 4, no. 4, pp. 359–366, 2022.

[19] A. Vaswani et al., "Attention is all you need," in *Proc. 31st Int. Conf. Neural Inf. Process. Syst.*, 2017, pp. 6000–6010.

[20] Y. N. Dauphin, A. Fan, M. Auli, and D. Grangier, "Language modeling with gated convolutional networks," in *Proc. 34th Int. Conf. Mach. Learn.*, 2017, pp. 1551–1559.

[21] H. Su, W. Qi, Z. Li, Z. Chen, G. Ferrigno, and E. D. Momi, "Deep neural network approach in EMG-Based force estimation for human-robot interaction," *IEEE Trans. Artif. Intell.*, vol. 2, no. 5, pp. 404–412, Oct. 2021.

[22] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Computation*, vol. 1780, no. 9, pp. 1735–1780, 1997.

[23] F. Zahedi, J. Arnold, C. Phillips, and H. Lee, "Variable damping control for pHRI: Considering stability, agility, and human effort in controlling human interactive robots," *IEEE Trans. Hum.-Mach. Syst.*, vol. 51, no. 5, pp. 504–513, Oct. 2021.

[24] J. Arnold and H. Lee, "Variable impedance control for pHRI: Impact on stability, agility, and human effort in controlling a wearable ankle robot," *IEEE Robot. Automat. Lett.*, vol. 6, no. 2, pp. 2429–2436, Apr. 2021.

[25] A. Paszke et al., "Automatic differentiation in PyTorch," in *Proc. 31st Conf. Neural Inf. Process. Syst.*, 2017.