Data-Efficient Behavior Prediction for Safe Human-Robot Collaboration

Ruixuan Liu

CMU-RI-TR-21-30

July 28, 2021

The Robotics Institute
School of Computer Science
Carnegie Mellon University
Pittsburgh, PA

Thesis Committee:
Changliu Liu
Oliver Kroemer
Katia Sycara
Jaskaran Grover

Submitted in partial fulfillment of the requirements for the degree of Master of Science in Robotics.

Abstract

Predicting human behavior is critical to facilitate safe and efficient human-robot collaboration (HRC) due to frequent close interactions between humans and robots. However, human behavior is difficult to predict since it is diverse and time-variant in nature. In addition, human motion data is potentially noisy due to the inevitable sensor noise. Therefore, it is expensive to collect a motion dataset that comprehensively covers all possible scenarios. The high cost of data collection leads to the scarcity of human motion data in certain scenarios, and therefore, causes difficulties in constructing robust and reliable behavior predictors. This thesis uses online adaptation (an online approach) and data augmentation (an offline approach) to deal with the data scarcity challenge in human motion prediction and intention prediction respectively. This thesis proposes a novel adaptable human motion prediction framework, RNNIK-MKF, which combines a recurrent neural network (RNN) and inverse kinematics (IK) to predict human upper limb motion. A modified Kalman filter (MKF) is applied to robustly adapt the model online to improve the prediction accuracy, where the model is learned from scarce training data. In addition, a novel training framework, iterative adversarial data augmentation (IADA), is proposed to learn safe neural network classifiers for intention prediction. The IADA uses data augmentation and expert guidance offline to augment the scarce data during the training phase and learn robust neural network models from the augmented data. The proposed RNNIK-MKF and IADA are tested on a collected human motion dataset. The experiments demonstrate that our methods can achieve more robust and accurate prediction performance comparing to existing methods.

Acknowledgments

I would like to thank my advisor Professor Changliu Liu for her advice on my work and continuous support of my Master's study. She has been extremely helpful, supportive, and patient throughout my study and I have learned a lot from her insights in the research field.

I would also like to thank Professor Oliver Kroemer, Professor Katia Sycara, and Jaskaran Grover for being my committee members. They provide a lot of insightful comments on my work, which are really helpful for my future research.

I want to also express my appreciation to my colleagues in the Intelligent Control Lab and all the amazing people I met at the Robotics Institute.

Last but definitely not least, great thanks to my family and friends. I could never achieve this without their unconditional support and understanding throughout the journey.

Funding

This work is in part supported by Ford Motor Company.

Contents

1 Introduction **1**
- 1.1 Challenges . . . 2
- 1.2 Motion Prediction . . . 3
- 1.3 Intention Prediction . . . 4

2 Related Work **7**
- 2.1 Human Motion Prediction . . . 7
- 2.2 Human Intention Prediction . . . 8
  - 2.2.1 Data Augmentation . . . 8
  - 2.2.2 Adversarial Training . . . 8

3 Problem Formulation **11**
- 3.1 Motion Prediction Formulation . . . 11
- 3.2 Intention Prediction Formulation . . . 13

4 Approach **15**
- 4.1 RNNIK-MKF Motion Prediction . . . 15
  - 4.1.1 RNN for Wrist Motion Prediction . . . 15
  - 4.1.2 IK for Arm Motion Prediction . . . 17
  - 4.1.3 Online Adaptation with MKF . . . 18
  - 4.1.4 Online Adaptation Estimation Error Propagation . . . 19
- 4.2 IADA Intention Prediction . . . 22
  - 4.2.1 Formal Verification to Find Adversaries . . . 23
  - 4.2.2 Expert Guidance for Labeling . . . 24
  - 4.2.3 Iterative Adversarial Data Augmentation . . . 25
  - 4.2.4 Prediction Confidence Estimation . . . 26
  - 4.2.5 IADA Analysis . . . 27

5 Experiments **29**
- 5.1 Data Collection . . . 29
- 5.2 RNNIK-MKF Motion Prediction . . . 30

5.2.1 Prediction Experiments 31
5.2.2 Unseen Humans Experiments 33
5.2.3 Unseen Tasks Experiments 35
5.2.4 Motion Occlusion Experiments 36
5.2.5 Online Adaptation Uncertainty 37
5.3 Intention Prediction with IADA 38
5.3.1 2D Binary Classification 39
5.3.2 2D Binary Classification with Biased Data 41
5.3.3 MNIST 43
5.3.4 Intention Prediction 44
5.3.5 Intention Prediction Confidence 44
5.3.6 Visualization of IADA for Intention Prediction 45
5.3.7 Discussion 48

6 Conclusion and Future Work **51**

Bibliography **53**

List of Figures

1.1 Visualizations of the human motion prediction using RNNIK-MKF. . . . 2
1.2 Visualizations of human intention prediction. 3
1.3 Adversarial data augmented by IADA. 5

3.1 Illustration of the arm motion prediction problem. 12
3.2 Illustration of the intention prediction problem. 14

4.1 RNNIK-MKF motion prediction framework. 16
4.2 Iterative adversarial data augmentation (IADA) training framework. . . . 23

5.1 Data collection experiment setup. 30
5.2 Examples of human motions in the collected dataset. 31
5.3 Visualizations of the predicted trajectories by different methods. 32
5.4 Prediction error for 1-60 prediction steps (Prediction Test). 33
5.5 Prediction error relative to motion range (Prediction Test). 33
5.6 Prediction error for 1-60 prediction step (Unseen Humans Test). 34
5.7 RMSE of the error for online adaptation (Unseen Humans Test). 35
5.8 Prediction error for 1-60 prediction step (Unseen Tasks Test). 36
5.9 Prediction uncertainty by RNNIK-MKF. 38
5.10 Visualizations of the FCNN decision boundaries on 2D classification. . . 40
5.11 Biased 2D training data. 41
5.12 Visualizations of the FCNN decision boundaries on biased 2D classification. 42
5.13 Example of the learning process by IADA on 2D biased classification. . . 43
5.14 Intention prediction confidence comparison. 46
5.15 Visualizations of the adversarial trajectories generated online by IADA. . 47
5.16 Model training time comparison. 48
5.17 Time cost decomposition for IADA. 48
5.18 Percentage of true adversarial data found by formal verification online. . 49

List of Tables

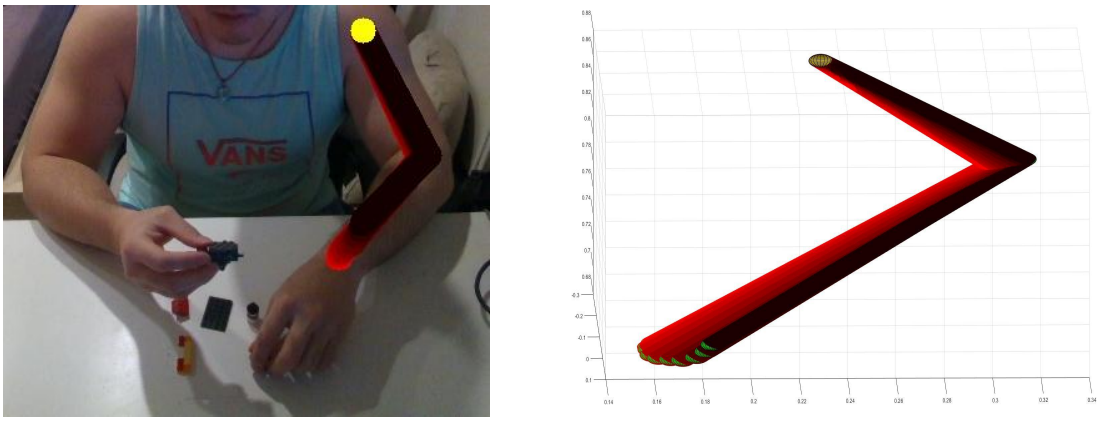
5.1 Motion prediction variation with occlusion. 37
5.2 Comparison of the model accuracy on 2D binary classification. 39
5.3 Comparison of the model accuracy on biased 2D binary classification. . . 41
5.4 Comparison of the model accuracy on MNIST digits classification. 44
5.5 Comparison of the model accuracy on human intention prediction. 45

Chapter 1

Introduction

The rapid development of **human-robot collaboration (HRC)** addresses contemporary needs by enabling more efficient and flexible production lines [37, 49]. Working in such environments, human workers are required to collaborate with robot arms in confined workspaces. Due to the frequent physical interactions, any collision could lead to severe harm to the human workers. Therefore, it is essential to ensure **safety** while maximizing **efficiency** when facilitating HRC. One key technology to enable safe and efficient HRC is to have **accurate**, **robust**, and **adaptable human behavior prediction** [27]. In this work, we mainly focus on two aspects of human behavior, human skeleton motion as well as semantic intention. By understanding human intentions and predicting human motion trajectories, the robot can plan ahead to better work with the human while avoiding potential collisions and safety violations.

This thesis considers the typical situation in an assembly line where the human worker sits in front of a desk and uses the components on the table to assemble a target object as shown in Fig. 1.1 and Fig. 1.2. Therefore, we focus on studying the human upper limb motion since the major body parts involved in the collaboration on production lines are arms. In this thesis, human motion prediction predicts the motion trajectory of the human arm as shown in Fig. 1.1, which is essentially a regression problem. On the other hand, human intention prediction predicts the intention label that describes the human behavior as shown in Fig. 1.2 (*i.e.*, assembling, reaching, retrieving, etc), which is essentially a classification problem. A detailed discussion of the problems is in chapter 3.



(a) Projected visualization of predicted arm motion. Yellow: shoulder joint.

(b) 3D visualization of predicted arm motion. Yellow: shoulder. Green: wrist.

Figure 1.1: Visualizations of the predicted human arm motion trajectory (10 steps) using RNNIK-MKF. Red: current arm configuration. Black: 10-step future arm configuration.

1.1 Challenges

Human behavior is difficult to predict. One major challenge is that it is expensive to collect a motion dataset that comprehensively covers all possible scenarios. One reason that leads to the high cost is the **diversity** of the data. For example, in a human motion dataset where the humans are doing assembly tasks, it is expensive (if not impossible) to collect data from all human subjects in all possible task situations. However, human subjects with different habits, body structures, moods, and task proficiencies may exhibit different motion patterns. Failure to include sufficient data to reflect these differences will lead to poor performance of the learned prediction model in real situations.

Another reason for the high cost of data collection is the existence of exogenous **input disturbances**. For example in Fig. 1.3(b), the images captured by a camera are likely to have black pixels not be completely black due to the inevitable sensor noise. Although these pixel-level differences do not confuse humans, they can easily fool the image classification models [46, 10]. Another example is shown in Fig. 1.3(c) for human motion data. Due to the inevitable sensor noise and algorithm uncertainty, the captured motion trajectory (red) may slightly deviate from the ground truth motion (white). These disturbances ideally should not change the output of the prediction model, but can actually easily fool the prediction model to make a wrong prediction if the model has not seen sufficiently many

(a) Assembling. (b) Reaching. (c) Retrieving.

Figure 1.2: Visualizations of intention prediction. Red: predicted operation type. Green: ground-truth operation type. White dots: observed human motion (10 steps). Red bounding boxes: usable components. Numbers on the red bounding boxes: likelihood to be the next piece for use.

disturbed data during training. However, it is expensive (if not impossible) to generate a full distribution of these disturbances on each data point.

Due to these high costs of data collection, a well-distributed and sufficient human behavior dataset is usually not available. Early works [2, 22, 18, 25] have shown that an insufficient amount of training data would make it difficult to train the neural network (NN) model, as the performance of the learned model will deteriorate in testing. Therefore, the learned model is not deployable in real applications. To deal with the data deficiency, we explore methods, both online and offline, that are suitable for constructing robust NN motion and intention prediction models.

1.2 Motion Prediction

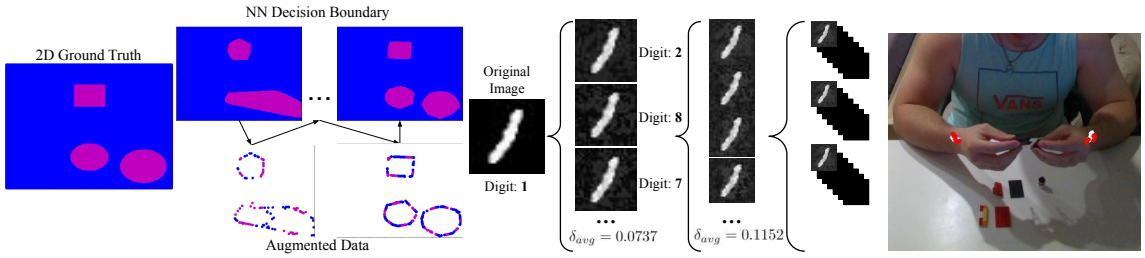
Online adaptation is an online approach to deal with data scarcity. We propose an online adaptable motion prediction framework (RNNIK-MKF) to predict human arm motion. In particular, a modified Kalman filter (MKF) is used for adapting the prediction model online. The MKF incrementally updates the prediction model using the incoming data and makes the initially poorly trained model perform better during the runtime.

The proposed method has several advantages. First, the method is adaptable, hence can easily and robustly generalize to unseen situations that the training data does not cover. Many existing approaches [35, 15, 36] have been proven to work well in trained environments, but have limited generalizability. However, it is impossible to obtain motion data from all workers with all possible demonstrations, and comprehensively validate the model for all situations. The proposed RNNIK-MKF enables online model adaptation

using MKF to achieve better generalizability with few training data. Second, the proposed structure can explicitly encode the kinematic constraints into the prediction model, which is explainable. Current approaches, such as [21], used complex graph neural network structures to encode the structural information, which is complicated and implicit. The proposed method uses the physical model derived from the human arm, which is explainable and intuitive. Thus, the predicted arm motion, as shown in Fig. 1.1, preserves the physical human arm structure. Third, the proposed method is robust when occlusion of the arm happens. Few existing methods [41] considered the situation when a partial human body is not observable. Yet the situation is common during collaboration as the robot could go between the sensor and the human arm, and thus, occluding the body part. Our method using the physical arm model can robustly predict the arm positions even when a portion of the arm, e.g., the elbow, is blocked from view. The proposed framework is tested on collected human motion data [33] with up to 2 s prediction horizon. The experiments demonstrate that the proposed method improves the prediction accuracy by approximately 14% comparing to the existing methods on seen situations. It stably adapts to unseen situations by keeping the maximum prediction error under 4 cm, which is 70% lower than other methods. Moreover, it is robust when the arm is partially occluded. The wrist prediction remains the same, while the elbow prediction has 20% less variation.

1.3 Intention Prediction

We propose an iterative adversarial data augmentation (IADA) framework to train robust NN classifiers **offline** with limited data to tackle the data scarcity challenge in intention prediction. This framework leverages formal verification and expert guidance for iterative data augmentation during training. The key idea of IADA is that we need to find the most "confusing" samples to the NN, e.g., the samples on the decision boundary of the current NN, and add them back to the dataset. We use formal verification to find these samples by computing the closest adversaries to the existing data (called roots) in L_∞ norm. These samples are called "adversaries" since the current NN predicts that they have different labels from the labels of roots (hence they are on the decision boundary of the current NN). The IADA framework will seek expert guidance to label these samples in order to ensure the correctness of adversaries. The sample is a true adversary if its ground truth label is the same as the label of its root; otherwise, this sample is a false adversary. The proposed



(a) (b) (c)

Figure 1.3: Adversarial data augmented by IADA. (a) Artificial 2D dataset. Left: Ground truth decision boundary. Right first row: NN decision boundary. Right second row: adversarial data augmented by formal verification and expert guidance given the current NN model. (b) MNIST dataset. Left: original training image. Second column: the first level adversaries found around the original image. Third column: the second level adversaries found around the first level adversarial images. Right: further expansions. (c) Human motion dataset. White: original wrist trajectory. Red: adversarial wrist trajectories being augmented to the training data.

IADA framework incorporates human-in-the-loop verification as the expert guidance. The labeled samples will be added back to the training dataset no matter what labels they are. The true adversarial samples can improve the robustness of the network, while the false adversarial samples can help recover the ground truth decision boundary. The IADA framework iteratively expands the dataset and trains the NN model.

The IADA framework has several advantages. First, it is composable since it allows easy switch of individual modules (*i.e.*, the training algorithm, the formal verification method, and the method for expert guidance). Second, it is safe and explainable due to the inclusion of expert guidance, unlike other adversarial training methods. Lastly, it is generic and appliable to general NNs, unlike existing data augmentation methods that require strong domain knowledge. To verify the effectiveness of the IADA training, we applied it to the human intention prediction task on a collected human motion dataset [33]. To validate the generalizability of IADA, we applied it to two additional applications, including a 2D artificial binary classification task as shown in Fig. 1.3(a), and the MNIST digits classification task [28]. We compared our training framework against several existing training methods. The results demonstrate that our training method can improve the robustness and accuracy of the learned model from scarce training data. The IADA training framework is suitable for constructing reliable and robust intention prediction models given the scarce human motion dataset.

The contributions of this thesis work can be summarized as follows:

1. Constructs a human motion dataset that studies the human upper limb motion when doing LEGO assembly tasks.
2. Proposes a novel RNNIK-MKF adaptable human arm motion prediction framework to predict high-fidelity arm motion. We demonstrate that the RNNIK-MKF outperforms the existing methods by showing that it can predict high-fidelity arm motion; it is generic to unseen humans or tasks; it is more robust when partial arm motion is occluded.
3. Proposes a novel IADA training framework that addresses the data scarcity challenge offline during the model training phase. We demonstrate that the IADA training can learn robust prediction models from scarce data for intention prediction.

Chapter 2

Related Work

2.1 Human Motion Prediction

Human motion prediction has been widely studied [43]. Early approaches [14] addressed the problem in a probabilistic way using Hidden Markov Models to estimate the possible areas that human arms are likely to occupy. Assuming human motions are optimal, [35] intended to learn a cost function of human behaviors by inverse optimal control and made predictions according to the learned cost function. Recent works [26, 4] addressed the prediction as a reaching problem by specifically learning the motion of human hands using neural networks. The recent development of recurrent neural network (RNN) had an outstanding performance in motion prediction [44]. The Encoder-Recurrent-Decoder (ERD) structure [15], which transformed the joint angles to higher-dimensional features, was shown to be effective in motion prediction. [36] added the sequence-to-sequence architecture to address the prediction as a machine translation problem. Recently, [21, 8] devoted to embedding the structural information of the human body into the neural networks. However, existing methods suffer from several problems. First, neural networks are pre-trained and fixed, which may have limited generalizability or adaptability to unseen situations. [12] incorporates online adaptation to improve the model performance online, but it only focuses on the single-joint motion. Second, the physical constraints of the human body are encoded using complex neural network structures, which is unintuitive and difficult to verify.

2.2 Human Intention Prediction

Human intention prediction has been widely studied recently. Many works, such as [53], focus on predicting the operation types. However, to the best of our knowledge, no existing works have addressed the data scarcity challenge specifically in the human intention prediction context. Nonetheless, there are approaches in other applications that address the data scarcity challenge.

2.2.1 Data Augmentation

Learning from insufficient training data has been widely studied [2, 22, 18, 25]. Data augmentation (DA) is a widely used approach. People use different approaches to generate additional data given the existing dataset [45] to improve the generalizability of the classifiers. [55, 13] propose either to manually design a policy or search for an optimal policy among the pre-defined policies to generate new data. On the other hand, instead of explicitly designing the policy, [40, 29, 3, 7] propose to learn generative NN models to create new data. However, manually designing the augmentation policy requires strong domain knowledge. In addition, the designed policy might only be suitable for a small range of related tasks. On the other hand, using NNs (GANs) for DA is knowledge-free. However, it has poor explainability, which might be a potential concern for safety-critical tasks.

2.2.2 Adversarial Training

Adversarial training [5] is a widely used approach for improving the NN model robustness. It has been widely studied since [46] first introduced the adversarial instability of NNs. Given the existing training data D_0, the adversarial training objective is formulated as a minimax problem,

$$\min_{\theta} \mathbf{E}_{(x_i,y_i)\in D_0} \max_{||\delta_i||_\infty \leq \varepsilon} L(f_\theta(x_i+\delta_i), y_i), \tag{2.1}$$

where δ_i is the adversarial perturbation and ε is the maximum allowable perturbation. Early works [46, 16] efficiently estimate the adversarial perturbations based on the gradient direction. However, [38] showed that the estimation in the gradient direction might be

inaccurate, and thus, makes the trained model sub-optimal. Recent work [11] proposes to adaptively adjust the adversarial step size during the learning process. Based on the idea of curriculum learning, [54] propose to replace the traditional inner maximization with a minimization, which finds the adversarial data that minimizes the loss but has different labels. The adversarial data generated via minimization is also known as the *friendly adversarial data*. On the other hand, neural network verification provides a provably safe way to find the adversarial samples with minimum perturbations, which are the most friendly adversaries. In fact, many works on neural network verification [38, 6, 47] have shown that using the adversarial samples from verification is effective for adversarial training. However, these methods might incorrectly take false adversaries as true adversaries, which might adversely harm the model training. Moreover, they only improve the local robustness of the trained model around the training data, and thus, have limited capacity to improve the generalizability of the network in real situations.

Chapter 3

Problem Formulation

3.1 Motion Prediction Formulation

The motion prediction in this thesis tackles the skeleton motion prediction problem on production lines, where humans collaborate with robot arms in confined workspaces. In these environments, the trunk of the human body tends to stay still and the major body parts interacting with the robot are the human arms. Therefore, we focus on predicting human arm motion, including the elbow and the wrist, as shown in Fig. 3.1. Note that the motion prediction is a regression problem, in which the prediction model maps the N-step historical arm motion observation to the future M-step arm motion.

In the motion prediction context, this thesis uses regular symbols and $\wedge$ to denote the observation and prediction respectively. The wrist and elbow positions at time k are denoted as $w_k,\ e_k \in \mathbb{R}^3$. The position vector $p \in \mathbb{R}^6$ is constructed by stacking the positions of the wrist and elbow. We define the observation and prediction at time k as

$$x_k = \begin{bmatrix} p_{k-N+1} \\ \cdots \\ p_{k-1} \\ p_k \end{bmatrix} \in \mathbb{R}^{6N}, \quad \hat{y}_k = \begin{bmatrix} \hat{p}_1^k \\ \hat{p}_2^k \\ \cdots \\ \hat{p}_M^k \end{bmatrix} \in \mathbb{R}^{6M}. \tag{3.1}$$

The constants $M,\ N \in \mathbb{N}$ are the prediction and observation horizons. $\hat{p}_j^k$ is the jth step prediction at time k. We need to build a prediction model, $\hat{y}_k = f(x_k)$, that predicts the future skeleton motion.

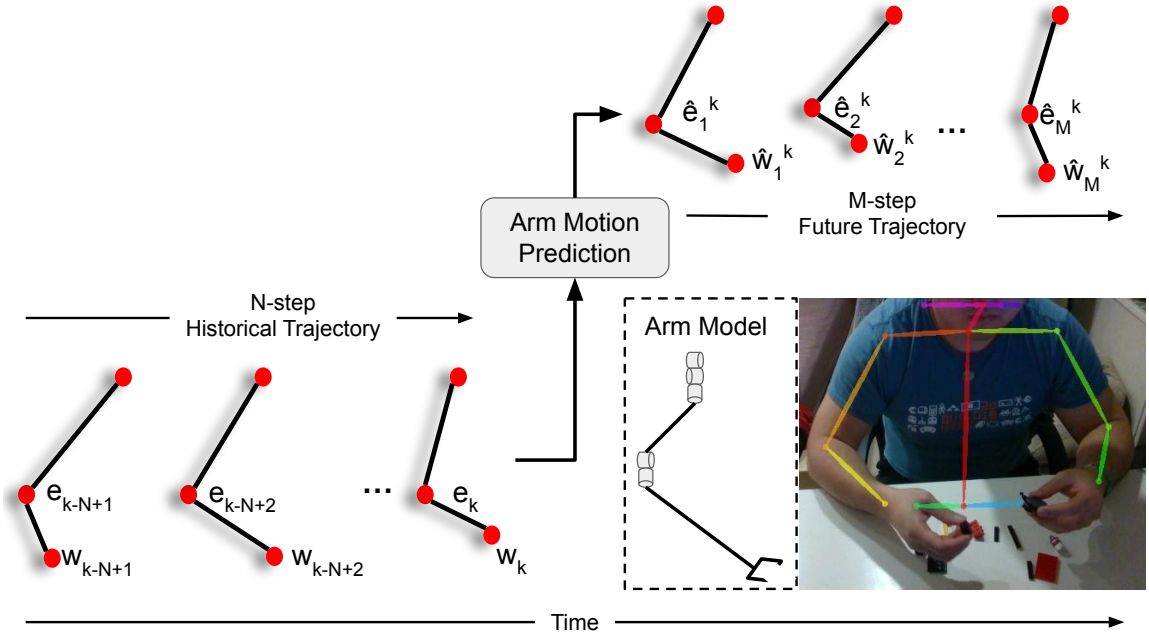


Figure 3.1: Illustration of the arm motion prediction problem. The problem takes in the N-step historical arm trajectory and outputs the M-step future arm trajectory. The human arm is modeled as a 5-DOF manipulator.

However, due to the data scarcity in the human motion dataset, the learned model f, which is validated in the training environment, would be less accurate and has poor generalizability in real situations. Therefore, we use online adaptation to address the data scarcity challenge in motion prediction. The goal is to construct an adaptable model, $\hat{y}_k = f(x_k, \hat{y}_{k-1})$, which is a time-varying function. The model f takes in the observed N-step historical trajectory and the previous M-step prediction and outputs the M-step future trajectory. The adaptation algorithm adjusts the model online and improves the generalizability.

3.2 Intention Prediction Formulation

Human intention prediction predicts the intention labels that describe the human behavior as shown in Fig. 3.2. Note that the intention prediction is a classification problem, in which the model maps the input, including the N-step historical arm motion observation and the environment information, to the corresponding pre-defined intention labels as shown in Fig. 3.2.

At timestep k, the prediction model is given with the N-step observation of the human motion and the environment information as

$$x_k^w = \begin{bmatrix} w_{k-N+1} \\ \cdots \\ w_{k-1} \\ w_k \end{bmatrix} \in \mathbb{R}^{6N}, \quad env = \begin{bmatrix} p_1^c \\ \cdots \\ p_{K-1}^c \\ p_K^c \end{bmatrix} \in \mathbb{R}^{3K}, \tag{3.2}$$

where K denotes the number of possible components and $p_i^c \in \mathbb{R}^3$ denotes the position of the workpiece i. The prediction model outputs the intention label y, including 1) the operation type and 2) the next assembly step. In our intention prediction case, the operation type has four different labels as shown in Fig. 3.2, including assembling, retrieving, reaching, and abnormal. The next assembly step has K different labels, corresponding to all possible components. Therefore, the intention prediction model can be written as $y_k = f(x_k^w, env_i)$. However, due to data scarcity, it is difficult to learn a robust and accurate f by merely designing sophisticated NN structures. Therefore, we use a standard fully-connected neural network (FCNN) to construct the prediction model f since it is decently powerful to model arbitrary nonlinear functions. Then, the major problem becomes how to learn a robust and accurate f given the insufficient human motion dataset.

Given a training dataset $D_0 = \{(x_i, y_i)\}_{i=1}^n$ where x is the input and y is the output label, the regular supervised training learns the NN model by solving the following optimization

$$\min_\theta \frac{1}{n} \Sigma_{i=1}^n L(f_\theta(x_i), y_i) \left(=: \mathbf{E}_{(x_i,y_i)\in D_0} L(f_\theta(x_i), y_i)\right), \tag{3.3}$$

where L is the loss function, θ is the NN model parameter, and f_θ is the NN transfer function. Our goal is to learn a model that minimizes the expected loss when deployed in

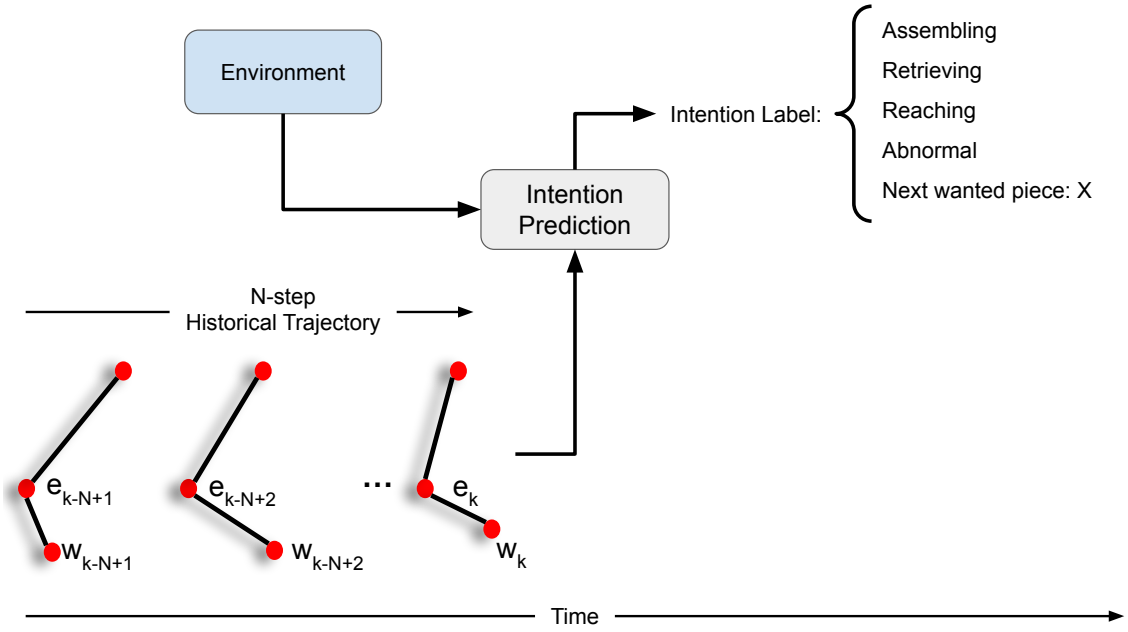


Figure 3.2: Illustration of the intention prediction problem. The problem takes in the N-step motion observation and outputs the corresponding pre-defined intention labels.

real applications:

$$\min_{\theta} \mathbf{E}_{(x,y)\in D}(L(f_\theta(x), y)), \tag{3.4}$$

where D represents the real input-output data distribution, which is unavailable during training. When D_0 is similar to D, we can obtain a f_θ that behaves similarly as the ground truth f. However, in our case (*i.e.*, behavior/intention prediction), D_0 is insufficient, and thus, leading to a poorly trained f_θ that behaves differently as f. The goal is to learn a robust and accurate f offline given the insufficient human motion dataset.

Chapter 4

Approach

4.1 RNNIK-MKF Motion Prediction

In general assembly tasks, humans move their hands with clear intention while the elbows are mainly to support the hand motion. Hence, we separate the motion prediction problem into wrist motion prediction and arm motion prediction. We use a recurrent neural network (RNN) for modeling the complex wrist motion and inverse kinematics (IK) to encode the physical body constraints and extend the wrist prediction to full-arm prediction. The modified Kalman filter (MKF) is used for online adapting the model to different humans and tasks and addressing the data scarcity challenge in human motion prediction. The proposed RNNIK-MKF framework is shown in Fig. 4.1.

4.1.1 RNN for Wrist Motion Prediction

Human wrist motion is complex and highly nonlinear. In addition, human motion has strong temporal connections. Therefore, we choose RNN since it is powerful to model arbitrary nonlinear functions and has hidden states to memorize past information. In particular, the Long Short-term Memory (LSTM) cell [19] is used since it has adequate gates to control the memory either to remember or to forget. We use an N-to-1 structure [15], where the RNN takes in the N-step wrist history and outputs the next single step wrist prediction as shown in Fig. 4.1. For an M-step prediction, the network iteratively appends the new prediction to the input and then predicts for the next step, until the M-step prediction is obtained as shown in Eq. (4.3). The structure has the advantage to enable more flexibility

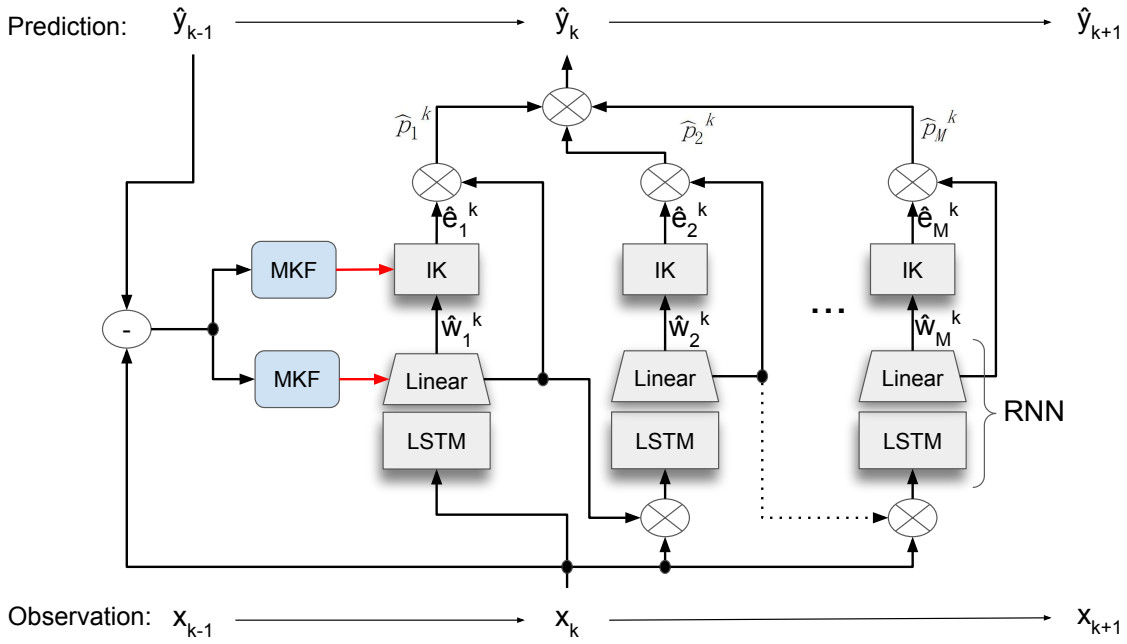


Figure 4.1: RNNIK-MKF motion prediction framework.

on the online adaptation since we can adapt the model as soon as a new observation is available.

We use $LSTM$ and RNN to denote the transition of a single LSTM cell and the N-to-1 prediction respectively. Given the N-step historical wrist trajectory at time k, the hidden states of the LSTM cells are propagated as

$$\begin{aligned}
[h_1,\, c_1] &= LSTM(w_{k-N+1},\, 0,\, 0),\\
[h_2,\, c_2] &= LSTM(w_{k-N+2},\, h_1,\, c_1),\\
&\dots\\
[h_N,\, c_N] &= LSTM(w_k,\, h_{N-1},\, c_{N-1}).
\end{aligned} \tag{4.1}$$

The first step wrist prediction is obtained using a linear layer

$$\hat{w}_1^k = \phi_k h_N + b_l, \tag{4.2}$$

where ϕ_k is the adaptable weight matrix (discussed in section 4.1.3) of the linear layer at time k and b_l is a constant bias. The LSTM cells Eq. (4.1) and the linear layer Eq. (4.2)

construct the N-to-1 RNN. We solve the prediction problem as

$$\begin{aligned} \hat{w}_1^k &= RNN(\phi_k,\ [w_{k-N+1},\ ...,\ w_k]^T), \\ \hat{w}_2^k &= RNN(\phi_k,\ [w_{k-N+2},\ ...,\ w_k,\ \hat{w}_1^k]^T), \\ &\dots \end{aligned} \tag{4.3}$$

until M-step wrist prediction is obtained.

4.1.2 IK for Arm Motion Prediction

The arm motion is mainly supporting the wrist motion, and thus, has a relatively simple motion pattern. Therefore, we use the model-based IK to model the arm motion. A general human arm can be decomposed into a 5-DOF manipulator as shown in Fig. 3.1. The shoulder is decomposed into 3 revolute joints and the elbow is decomposed into 2 revolute joints. The wrist is considered as the end-effector. The state of the arm in the joint space is $\theta \in \mathbb{R}^5$. The end-effector state is $w \in \mathbb{R}^3$. The IK problem solves for $\hat{\theta}$ under the predicted wrist position $\hat{w}$ where $\hat{\theta} = IK(\hat{w} \mid \theta, w)$, given the current states θ and w. A popular approach to solve the IK problem is using the Jacobian [50]. The Jacobian $J(\theta) \in \mathbb{R}^{3\times n}$ for an n-DOF robot manipulator is

$$J(\theta) = \frac{\partial FK(\theta)}{\partial \theta} \approx \left[\frac{\partial IK(w)}{\partial w}\right]^{-1}, \tag{4.4}$$

where $FK(\theta) : \mathbb{R}^n \to \mathbb{R}^3$ is the forward kinematics. Using matrix transpose to replace matrix inverse, we can solve the IK problem as

$$\hat{\theta} = \theta + AJ(\theta)^{\mathsf{T}}(\hat{w} - w), \tag{4.5}$$

where $A \in \mathbb{R}^{n\times n}$ is an adaptable parameter matrix (discussed in section 4.1.3) that encodes the individual differences on the joint velocities, *e.g.*, some workers tend to place the elbow on the table, while others prefer to move with the wrist. We can solve the arm prediction by solving the IK for each predicted wrist position.

Algorithm 1 Modified Kalman Filter

Input: MKF parameters: $\lambda > 0$, $\sigma_w \geq 0$, $\sigma_v \geq 0$.
Input: EMA parameters: $0 \leq \mu_v < 1$, $0 \leq \mu_p < 1$.
Input: Φ_{k-1}, $E_k = (y_{k-1} - \hat{y}_{k-1})$, X_{k-1}.
Output: Φ_k.
Internal State: Z, V.
$K = Z_{k-1}X_{k-1}^T(X_{k-1}Z_{k-1}X_{k-1}^T + \sigma_w I)^{-1}$
$V_k = \mu_v V_{k-1} + (1-\mu_v)KE_k$
$\Phi_k = \Phi_{k-1} + V_k$
$Z^* = \frac{1}{\lambda}(Z_{k-1} - KX_{k-1}Z_{k-1} + \sigma_u I)$
$Z_k = \mu_p Z_{k-1} + (1-\mu_p)Z^*$

4.1.3 Online Adaptation with MKF

Human motion is diverse and time-varying. For instance, a worker might initially move the entire arm. But after a while, the worker would probably rest the elbow on the table and only move the wrist. Moreover, different individuals can perform the same task very differently due to different body structures or task proficiency. It is expensive (if not impossible) to collect a dataset that comprehensively includes all possible situations. Therefore, the data we use for training is highly likely to be scarce and insufficient. Due to the data scarcity, the learned prediction model might deviate from the true prediction model, and thus, perform poorly during testing or deployment. Therefore, online adaptation is important to make the method robust and generic to unseen situations. In the proposed framework, the wrist predictor has a linear output layer Eq. (4.2) in ϕ, and the arm predictor Eq. (4.5) is linear in A. Therefore, we apply linear adaptation to the system.

Many existing online adaptation algorithms are based on stochastic gradients [23]. However, these methods have no guarantee of optimality. Thus, we use a second-order method in order to achieve better convergence and optimality. Recursive least-squares parameter adaptation algorithm (RLS-PAA) [17] is an optimal method. However, since the regular RLS-PAA does not consider a noise model of the system, it is inefficient to tune. In addition, we need to apply smoothing techniques to the internal adaptation parameters to ensure stable adaptation and prediction under noisy measurements. Moreover, we want the more recent information to have more impact on the current estimation. Hence, we propose to use MKF for online adaptation. We add a forgetting factor, λ, to the conventional Kalman filter to prevent the estimation from saturation. In addition, we apply

the Exponential Moving Average (EMA) filtering method discussed in [1] to smooth the adaptation process.

The linear prediction system can be written as

$$\begin{aligned} \Phi_k &= \Phi_{k-1} + \omega_k, \\ \hat{y}_k &= \Phi_k X_k + v_k. \end{aligned} \tag{4.6}$$

For wrist prediction, $\hat{y}$ is the wrist prediction. Φ and X are the parameter matrix ϕ and hidden feature h_N in Eq. (4.2). For arm prediction, $\hat{y}$ is the arm prediction, Φ and X are equivalent to A and $J(\theta)^{\mathsf{T}}(\hat{w}-w)$ in Eq. (4.5). $\omega_k \sim N(0,\sigma_w)$ and $v_k \sim N(0,\sigma_v)$ are virtual Gaussian white noises. The MKF adaptation algorithm is summarized in algorithm 1. There are two internal variables: the covariance matrix Z and the parameter update step V. The learning gain, K, is calculated on line 6 using the Kalman filter's formula. V is calculated using the EMA filtering on line 7. The parameter matrix Φ is updated on line 8. Then Z is updated using the forgetting factor on line 9 and then smoothed using EMA on line 10.

Following the approach in [12] that applies RLS-PAA to the linear output layer of a fully connected neural network (FCNN), we apply the MKF to Eq. (4.2) with respect to ϕ as well as to Eq. (4.5) with respect to A. The virtual noises ω and v are tunable in the adaptation.

The proposed RNNIK-MKF framework is then summarized in algorithm 2. MKF denotes the adaptation in algorithm 1. The superscripts w and a distinguish the variables for wrist and arm predictions. $FK_{ew} = [FK; FK_e]$ where FK and FK_e are the forward kinematics to wrist and elbow. IK_{ew} solves for the arm state θ_{pre} when both wrist and elbow positions are known. For each time step, the algorithm obtains the current configuration in line 6. The parameters are updated using MKF on lines 11, 13. If occlusion happens, the adaptation is turned off on line 16. Then the algorithm iteratively predicts the future wrist and elbow trajectories for M steps on lines 20 and 22.

4.1.4 Online Adaptation Estimation Error Propagation

As the MKF consistently adjusts the model parameters online, the performance of the prediction model varies as time changes. It is important to quantify the error of the motion prediction as the adaptation proceeds. Given the prediction system in Eq. (4.6), the uncertainty of the prediction by RNNIK-MKF can be quantified by 1) the statistical

Algorithm 2 RNNIK-MKF Motion Prediction

Input: Pre-trained RNN, Arm Model, $\phi_0 = \phi_{trained}$, $A = I$.
Input: MKF parameters: $0 < \lambda^w \leq 1$, $\sigma_w^w \geq 0$, $\sigma_v^w \geq 0$, $0 < \lambda^a \leq 1$, $\sigma_w^a > 0$, $\sigma_v^a \geq 0$.
Input: EMA parameters: $\mu_v^w,\ \mu_p^w,\ \mu_v^a,\ \mu_p^a \in [0,1)$.
Output: Full arm trajectory prediction $\hat{y}$
for $i = 1, 2, \ldots, k$ **do**
    Obtain current configuration w_i, e_i.
    $x_i^w = [w_{i-N+1}; \ldots; w_{i-1}; w_i]$, $w_{pre} = w_i$.
    for $j = 1, 2, \ldots, M$ **do**
        if $j = 1$ **then**
            if Observation Available **then**
                $\phi_i = MKF(\phi_{i-1},\ w_i - \hat{w}_1^{i-1}, h_{i-1})$
                $X_{i-1}^a = J^\mathsf{T}(\theta_{i-1})(w_i - w_{i-1})$
                $A_i = MKF(A_{i-1},\ \theta_i - \hat{\theta}_1^{i-1},\ X_{i-1}^a)$
                $\theta_{pre} = IK_{ew}([w_i;\ e_i])$
            else
                $\phi_i = \phi_{i-1}$, $A_i = A_{i-1}$
                $\theta_{pre} = IK_{ew}([w_i;\ \hat{e}_1^{i-1}])$
            end if
        end if
        $\hat{w}_j^i = RNN(\phi_i, x_i^w)$
        $x_i^w = [w_{i-N+j}; \ldots; w_i; \ldots; \hat{w}_j^i]$
        $\hat{\theta}_j^i = \theta_{pre} + A_i J(\theta_{pre})^\mathsf{T}(\hat{w}_j^i - w_{pre})$
        $[w_{pre}; \hat{e}_j^i] = FK_{ew}(\hat{\theta}_j^i)$, $\theta_{pre} = \hat{\theta}_j^i$
    end for
    $\hat{y}_i = [\hat{w}_1^i; \hat{e}_1^i; \ldots; \hat{w}_M^i; \hat{e}_M^i]$
end for

standard deviation of the prediction error and 2) the propagation of the mean squared estimation error (MSEE) [30].

Statistical standard deviation At timestep k, the prediction error is calculated as $e_k = y_k - \hat{y}_k$. The statistical standard deviation of the prediction error is calculated as

$$\sigma_y = \sqrt{\frac{\sum_1^T e_k^2}{T}} = \sqrt{\frac{\sum_1^T (y_k - \hat{y}_k)^2}{T}}, \tag{4.7}$$

where T is the total prediction timesteps.

Propagation of a priori MSEE As illustrated in algorithm 1, the covariance matrix, Z_k, propagates and is the MSEE of the model parameters during online adaptation. Given the covariance matrix Z_k at time step k, the covariance matrix of the motion prediction $\hat{y}_k$ can by calculated as

$$Z_k^{\hat{y}} = \Phi_k Z_k \Phi_k^T + v_k, \tag{4.8}$$

where v_k is the Gaussian noise and Φ_k is the parameter matrix being adapted as illustrated in Eq. (4.6). And $Z_k^{\hat{y}}$ is the a priori MSEE of the prediction $\hat{y}_k$ and quantifies the prediction uncertainty error.

4.2 IADA Intention Prediction

Due to the data scarcity, a standard fully-connected neural network (FCNN) classifier can achieve outstanding performance on the training data but fails to perform well on the testing data or real deployment. To address the challenge, we aim to augment the insufficient dataset D_0 with adversarial data D_{adv} to robustly learn the true decision boundaries of the real but unknown data D as discussed in section 3.2. The goal is to ensure the learned model on $D_0 \cup D_{adv}$ is as close as possible to Eq. (3.4). To achieve the goal, we formulate the training problem as a two-layer optimization

$$\begin{aligned} &\min_{\theta} \mathbf{E}_{(x,y)\in D_0\cup D_{adv}}(L(f_\theta(x),y)), \\ &\quad D_{adv} = \left\{ \left(x', \text{label}(x')\right) \mid \exists (x_0,y_0) \in D_0, x' = \arg\min_{x' \text{ s.t. } ||x_0-x'||_\infty \le \varepsilon \wedge f_\theta(x') \neq y_0} L'(f_\theta(x'),y_0) \right\}. \end{aligned} \tag{4.9}$$

The outer objective optimizes the NN model parameters θ to minimize the loss on the data from $D_0 \cup D_{adv}$. The inner objective constructs D_{adv} given θ, which essentially finds the most friendly adversarial data for all training data. The most friendly adversarial data for $(x_0, y_0) \in D_0$ is an input sample x' that is at most ε distance away from x_0 in the L_∞ norm but changes the network output from y_0 with the smallest loss L'. The loss L' for the inner objective may or may not be the same as the original loss L. Ideally, we should choose a loss that guides us to the most "confusing" part of the input space. The label of these friendly adversarial data is decided by an additional labeling function, which ideally should match the ground truth. If $\text{label}(x') = y_0$, we call x' a true adversary; otherwise a false adversary.

We propose an iterative adversarial data augmentation (IADA) training framework to solve Eq. (4.9). Figure 4.2 illustrates the proposed training framework that aims to solve the problem defined in Eq. (4.9). In particular, the inner objective will be solved using formal verification to be discussed in section 4.2.1, while the labeling will be performed under expert guidance to be discussed in section 4.2.2. The iterative approach to solve the two-layer optimization will be discussed in section 4.2.3.

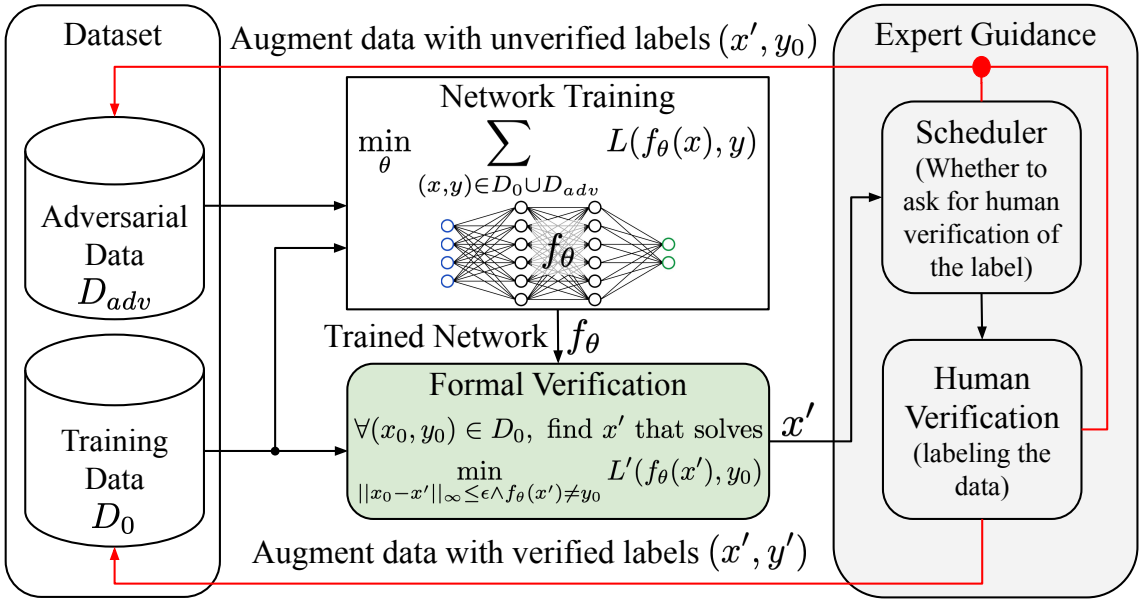


Figure 4.2: Iterative Adversarial Data Augmentation (IADA) training framework. For $(x,y) \in D_0 \cup D_{adv}$, the framework updates the model. If $(x_0,y_0) \in D_0$, the formal verification finds the most friendly adversarial sample x' around x_0 if exists. The scheduler determines if human verification is required for x'. If not, (x',y_0) is augmented to D_{adv}. Otherwise, if human is able to assign y', then (x',y') is added to D_0. If not, then (x',y_0) is appended to D_{adv}.

4.2.1 Formal Verification to Find Adversaries

The two-layer optimization in Eq. (4.9) is similar to the formulation of adversarial training Eq. (2.1). The major distinction lies in the inner objective. The minimax formulation in Eq. (2.1) might reduce the model accuracy and generalizability when maximizing the robustness [48] since the inner maximization could be too aggressive when generating adversaries. Similar to the approach in [54], instead of finding the adversaries via the inner maximization, we take the adversarial label $f_\theta(x_0+\delta_0)$ into consideration when generating the adversarial sample $x' = x_0 + \delta_0$. In particular, we design the inner loss to penalize the magnitude of δ_0. Then the inner optimization in Eq. (4.9) can be written as

$$\min_{x'} \|x_0 - x'\|_\infty, \text{ s.t. } \|x_0 - x'\|_\infty \leq \varepsilon, f_\theta(x_0) \neq f_\theta(x'). \tag{4.10}$$

The reason why we use the distance metric in the input space as the loss L' instead of using the original loss or any other loss that penalizes the output is that this metric reflects the "confusing" level of samples. We generally expect that the learned model is regular

at the training data for generalizability. The easier it is to change the label by perturbing the input data, the less regular the model is, and hence more "confusing". In addition, this formulation provides a quantitative metric δ_0 for evaluating the robustness online. Therefore, we can prioritize enhancing the weaker boundaries and obtain full control of the training process. Nevertheless, the optimization in Eq. (4.10) is nontrivial to solve due to the nonlinear and nonconvex constraints introduced by f_θ. To obtain a feasible and optimal solution, we use formal verification [32] to find the appropriate x'. The optimization in Eq. (4.10) essentially finds the minimum input adversarial bound, which can be solved by various neural verification algorithms. In particular, primal optimization-based methods such as MIPVerify [47] and NSVerify [34] solve the problem exactly by encoding the neural network into a mixed-integer linear program; dual optimization-based methods [52] can compute an upper bound of the problem by relaxing the nonlinear ReLU activation functions; reachability-based methods [51] can also compute an upper bound by over-approximating the reachable set and binary search for the optimal loss.

4.2.2 Expert Guidance for Labeling

With the adversaries obtained from formal verification, we need to label them before augmentation. The proposed IADA framework uses expert guidance to guarantee the safety and accuracy of the training. The framework requires the human expert to verify the newly added adversaries. It uses a scheduler to balance the required human effort and the training accuracy. We use an ensemble NN and the L_∞ distance check to construct the scheduler. The ensemble NN indicates whether the adversary is meaningful. We say data is meaningful if humans can properly interpret the data label. For example, the disturbance to a human trajectory that violates human kinematic constraints or to MNIST images that have two digits shown in one image is not meaningful. However, note that it is difficult to obtain an accurate ensemble model that can filter out all unmeaningful adversaries. However, it is tolerable since the framework keeps two datasets, D_0 and D_{adv} discussed in section 4.2.3, although the human expert might have extra work to do, which is more than needed. The scheduler requires human verification if $\delta_0 > d$ or the ensemble model agrees that the adversary is meaningful, where d is a pre-defined threshold. Based on the application, a smaller d can be chosen to require more frequent human verification to improve the training accuracy, while a larger d alleviates the amount of human effort.

4.2.3 Iterative Adversarial Data Augmentation

We use an iterative approach to solve the two-layer optimization in Eq. (4.9) by incrementally augmenting the data with the outer NN training loop. In one iteration, we find the adversaries generated in section 4.2.1, which are the most "confusing" points for the current NN. The adversaries will be labeled by experts and augmented to the dataset, either D_0 or D_{adv}, as shown in Fig. 4.2. In the next iteration, the framework will further verify, label, and expand the dataset based on the augmented dataset. The framework maintains two datasets D_0 and D_{adv}, where D_{adv} is initially empty and D_0 starts with the original training data. The adversaries are only augmented to D_0 if they are verified by the human expert, otherwise, they are pushed into D_{adv}. Note that the adversaries generated in section 4.2.1 can either be true or false adversaries, but both are informative and useful for improving the NN learning. However, incorrectly mixing these two types of adversaries can greatly harm NN learning. Therefore, we maintain D_0 and D_{adv} to distinguish the safe data and potentially incorrect data. The framework iteratively expands only from the data $(x_0, y_0) \in D_0$. Therefore, the framework can further expand the available training data safely. D_{adv} will be refreshed before verification as shown in algorithm 3, since the framework only wants temporary effect from D_{adv} but permanent effect from D_0 for safety and correctness. Also note that the IADA framework is reduced to standard adversarial training, except that we have a different inner minimization loss objective if the scheduler requires no human verification. On the other hand, it is equivalent to standard online data augmentation when the scheduler always requires human verification (similar to data aggregation in imitation learning [42]). Based on the applications, the scheduler can be tuned to either require more or less expert knowledge to balance the training accuracy and the human effort.

The IADA framework is summarized in algorithm 3. The verification rate r_v indicates the frequency of online formal verification and data augmentation. The verification number C indicates a maximum number of points to verify at each iteration. Q_v is a priority queue based on the level of expansion and the perturbation bound δ. The level is defined as 0 for the original training data. The adversaries generated from the original data have level 1. Q_v determines the weakest points and prioritizes those points during verification and expansion. On line 12, the system gets the weakest point and verifies it on line 14. If the robustness is not satisfied, the system either asks for expert knowledge to assign a label or assumes it to have the same label as the root. If the label is assigned by humans, the

Algorithm 3 Iterative Adversarial Data Augmentation (IADA)

1: **Input:** Original dataset $(x_0, y_0) \in D_0$.
2: **Input:** Robustness bound ε, verify rate r_v, verification number C, learning rate α.
3: **Output:** NN parameter θ.
4: **Initialize:** $\theta = \theta_0$, $Q_v = \{D_0\}$.
5: **repeat**
6: **if** VerificationRound(r_v) **then**
7: // Refresh D_{adv} since it might contains data with incorrect labels.
8: Clear D_{adv}.
9: **for** $i = 1, \ldots, C$ **do**
10: Breaks if Q_v is empty.
11: // Q_v prioritizes the point with 1) the lowest level of expansion and 2) the smallest perturbation bound.
12: $(x,\ y) = Q_v.pop$.
13: // Find an adversary by solving Eq. (4.10).
14: $x' = Verify(\theta, x, y, \varepsilon)$.
15: Skip if no adversary x' is found.
16: **if** Scheduler && Human Verified **then**
17: Obtain the verified label y'.
18: Push (x,y) and (x',y') to Q_v and append (x',y') to D_0.
19: **else**
20: Append (x',y) to D_{adv}.
21: **end if**
22: **end for**
23: **end if**
24: **for** minibatch $\{X,Y\}$ in $D_0 \cup D_{adv}$ **do**
25: $\theta \longleftarrow \theta - \alpha L(f_\theta(X), Y)$.
26: **end for**
27: **until** *max_epoch* reached.
28: **return** θ.

adversarial sample is added to D_0 for further expansion. The NN model is updated using the training data and the adversarial data at lines 24-26.

4.2.4 Prediction Confidence Estimation

The prediction model outputs the corresponding classification labels. However, it is possible that the output label is incorrect. Mistakenly trusting the incorrect prediction would harm the downstream modules and potentially lead to safety hazards. Therefore, it is important to estimate the reliability of the prediction. We use a confidence value to quantitatively

estimate the reliability of the prediction. Ideally, it is desired that the correct predictions have higher confidence and incorrect predictions have lower confidence. And we can use the confidence value to distinguish the correct and incorrect predictions.

Assuming uncertainty exists in the input x ($e.g.$, the noise when capturing the human motion trajectories), the uncertainty of x can be modeled as a Gaussian white noise as $X \sim N(0, \sigma_X)$. We use the perturbation bound for estimating the prediction confidence. Given a learned NN classifier, it is expected that the prediction closer to the decision boundary is more likely to be uncertain. Therefore, we can estimate the prediction confidence C as

$$\begin{aligned} C = P(|t| \leq \delta) &= P(t \leq \delta) - P(t \leq -\delta) \\ &= \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{\delta} e^{-\frac{t^2}{2}} dt - \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{-\delta} e^{-\frac{t^2}{2}} dt \\ &= \frac{1}{\sqrt{2\pi}} \int_{-\delta}^{\delta} e^{-\frac{t^2}{2}} dt, \end{aligned} \tag{4.11}$$

where P is the cumulative distribution function of $X \sim N(0, \sigma_X)$ and $\delta = ||x - x'||_\infty$ is the perturbation bound by solving Eq. (4.10).

Note that the confidence estimation in Eq. (4.11) is generic and can be applied in different applications and NN models. However, the quality of the learned NN determines if the metric is valid. This metric would be more informative and reliable for NN models with decision boundaries closer to the ground-truth decision boundaries. This metric is valid and informative here since the prediction model learned by IADA is close to the true prediction model.

4.2.5 IADA Analysis

The IADA framework is unique in several ways. First, it is well-known that there exists a trade-off between accuracy and robustness in general adversarial training [48]. This is mainly due to incorrectly mixing the true and false adversaries. However, by introducing expert guidance, the true adversaries can improve the NN robustness and enlarges the decision area, while the false adversaries can enhance the ground truth decision boundary for better accuracy. Second, although the augmented data may not recover the ground truth data distribution, the augmented data will converge to and fully cover the ground truth decision boundary in the limit (proof left for future work). As a result, the learned

NN will converge to the ground truth with the correct decision boundary. Hence, this data augmentation is most effective.

Based on these features, we argue that IADA is most suitable for tasks that have non-trivial distribution of the data around the decision boundary, such as human intention prediction in section 5.3.4. Such tasks generally have false adversaries close to the training data and can easily have "confused" decision boundaries. On the other hand, image-related tasks, such as MNIST, are suitable for using IADA, but not necessarily required. It is mainly due to that image data is generally easier to collect, thus, unlikely to have insufficient data. In addition, it is rare to have false adversaries close to the original images.

Chapter 5

Experiments

5.1 Data Collection

This work considers the situation on production lines, where the human worker sits in front of a desk; and we mainly predict the human behavior based on the upper body motion. To the best of our knowledge, such a human motion dataset that focuses on human arm motion does not exist publicly. Therefore, we collected our human motion dataset in order to test and validate the proposed human motion prediction and intention prediction methods.

Figure 5.1 demonstrates the experiment setup for collecting the motion data. In order to simulate the situation in typical HRC production lines, we have humans sitting in front of a desk doing assembly tasks using LEGO pieces. A motion capture sensor, an Intel RealSense D415 camera, faces downward to record the human upper limb motion at approximately 30Hz. A visualizer displayed on the screen in front of the human guides the human subjects to walk through the data collection experiment. The experiments have 3 humans doing 5 assembly tasks for 3 trials each. There are 8 available LEGO pieces provided on the desk. Each assembly task requires 4 LEGO pieces. For each trial, the human is given the pictures of the desired assembled object by the visualizer and then uses the provided LECO pieces to assemble the target object based on their interpretation. The recorded data durations of the tasks vary from 30 s to 90 s based on the task proficiency of the human subjects. The OpenPose [9] is used to extract the human pose as shown in Fig. 3.1. Figure 5.2 shows the example image frames of the collected motion data. The collected human motion dataset includes the following typical operations, including

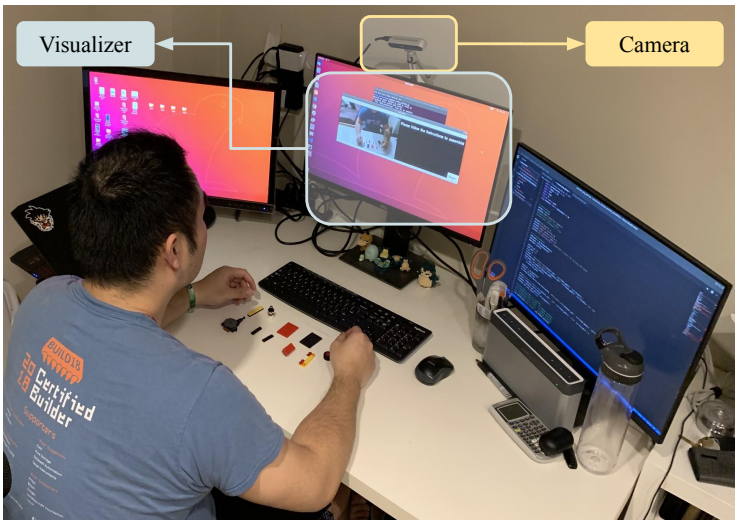


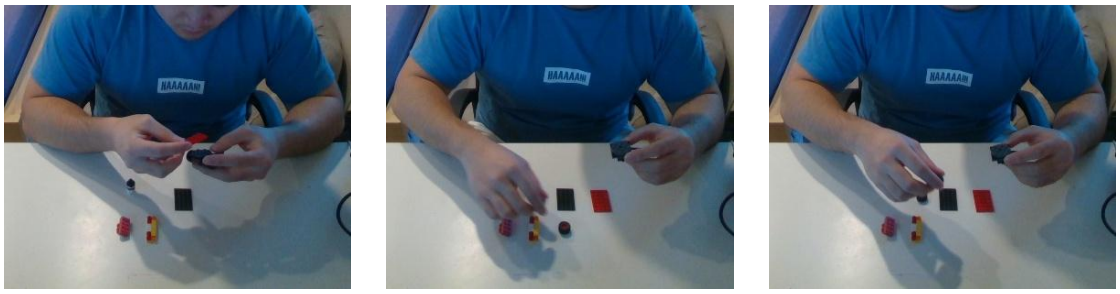
Figure 5.1: Data collection experiment setup.

1. Assembling: the human assembles two LEGO pieces together in place Fig. 5.2(a).
2. Reaching: the human moves the wrist toward the intended LEGO piece on the desk Fig. 5.2(b).
3. Retrieving: the human grabs the LEGO piece and moves the wrist back to the assembling area Fig. 5.2(c).

However, due to individual differences (*i.e.*, task proficiency, arm length, etc), the behaviors are very different in terms of displacement and duration. Note that even though the collected dataset includes different humans doing different assembly tasks for several trials, the demonstrations by human subjects are far from sufficient. It is highly possible that the test set includes motion patterns that never exist in the training set. Therefore, the adaptability of RNNIK-MKF and the robust learning by IADA are critical to building robust motion and intention prediction models.

5.2 RNNIK-MKF Motion Prediction

We use the motion data in section 5.1 to evaluate the proposed RNNIK-MKF motion prediction framework. We compare our method to several existing methods, including FCNN, FCNN with RLS-PAA (NN-RLS) [12], ERD [15], and LSTM-3LR [15]. The proposed RNNIK-MKF has one recurrent layer with 128 hidden size. To establish a fair



(a) Assembling. (b) Reaching. (c) Retrieving.

Figure 5.2: Examples of human motions in the collected dataset.

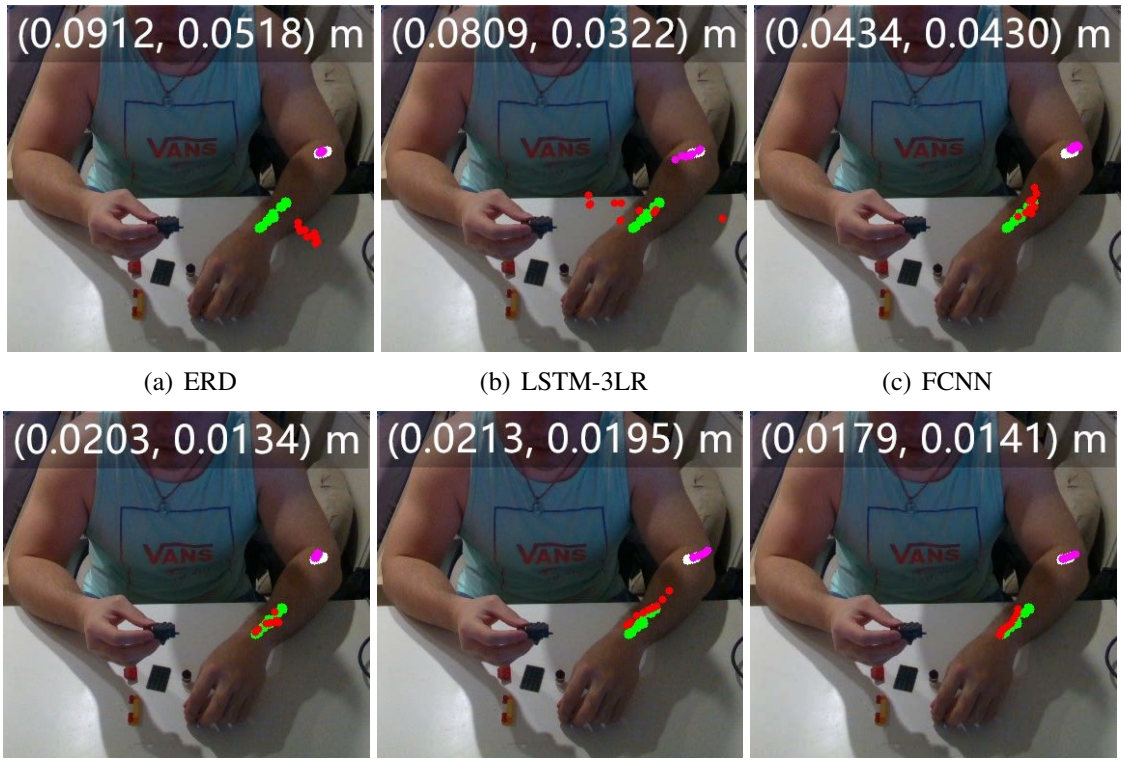
comparison, we design the FCNN with two ReLU layers and a linear output layer [1]. We decrease the hidden size of the ERD and LSTM-3LR to match the size of ours. All models have the loss function set to be the Huber loss [20] and the post-training losses for all models are around 5×10^{-3}. All models are trained using the trajectories randomly selected from humans 1 and 3 doing tasks 1, 2, 3. Since the regular human reaction time to visual stimulus is around 0.25 s, we set the input horizon to be 10, which is equivalent to approximately 0.3 s. The experiments test the prediction horizon from 1 to 60, which is equivalent to 0.033 s to 2 s. We define the average prediction error for each prediction step as,

$$E_j = \frac{1}{T}\sum_{i=0}^{T}(||p_{i+j} - \hat{p}^i_j||_2), \tag{5.1}$$

to quantitatively evaluate the prediction accuracy. T denotes the total time steps. In the following discussion, RNNIK represents the proposed method with wrist MKF turned off.

5.2.1 Prediction Experiments

We use trajectories from humans 1 and 3 doing tasks 1, 2, 3 but different trials to evaluate the prediction quality. Figure 5.3 shows an example of the predicted trajectories from different methods. The prediction errors from different methods are labeled on the images. We can see RNNIK-MKF outperforms others in terms of prediction accuracy since it achieves the lowest prediction error. The predicted trajectories of the wrist and elbow are plotted in red and purple on the images. From the plotted predictions, RNNIK-MKF also outperforms others in terms of prediction quality. Figure 1.1(b) shows an example of the predicted arm



(a) ERD (b) LSTM-3LR (c) FCNN

(d) NN-RLS (e) RNNIK (f) RNNIK-MKF

Figure 5.3: Visualizations of the predicted trajectories (10 steps) using different methods. Green: wrist ground-truth trajectories. White: elbow ground-truth trajectories. Red: wrist predictions. Purple: elbow predictions. Due to projection, the errors might appear to be larger than the true errors. True errors are labeled as (Wrist Error, Elbow Error) m.

trajectory in 3D and Fig. 1.1(a) demonstrates the projected visualization of the arm motion on the 2D image. The visualizations demonstrate that the RNNIK-MKF generates accurate prediction, but more importantly, the prediction preserves the arm's physical structure. As shown in Fig. 5.3, the predictions by others might reach certain configurations that are infeasible by the human arm. However, as shown in Fig. 1.1, the predicted motion of the human arm by our method follows the kinematics constraint and makes the prediction more authentic and reliable.

Figure 5.4 demonstrates the overall prediction accuracy quantitatively. The proposed RNNIK-MKF and RNNIK have very close performance, which is shown in Fig. 5.3 as well. In general, the RNNIK-MKF has the lowest prediction error. It has 14% lower prediction errors on average comparing to LSTM-3LR, which has the lowest errors among

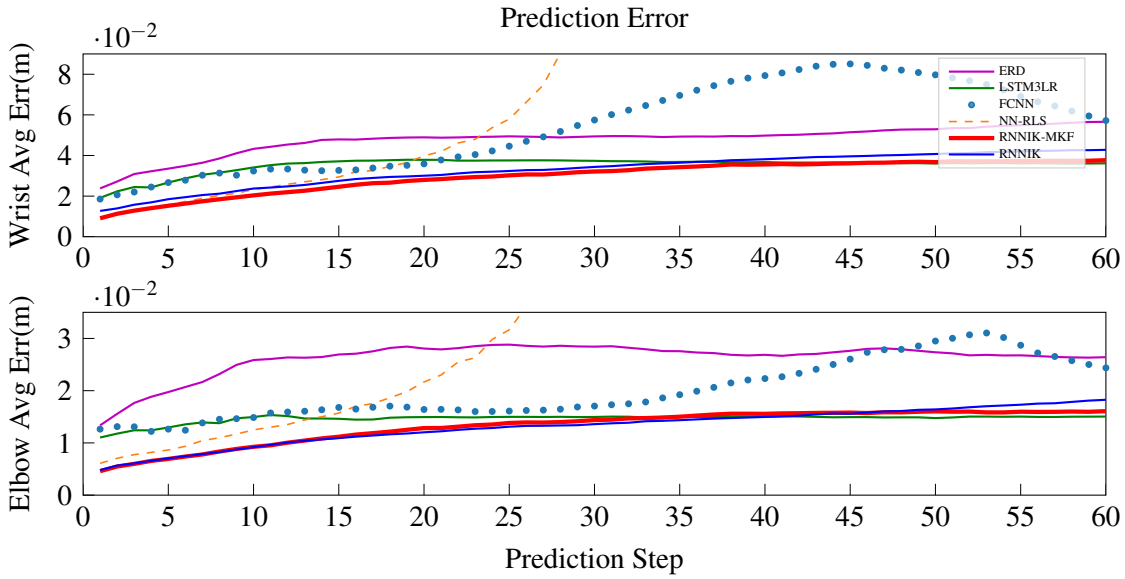


Figure 5.4: Prediction error for 1-60 prediction steps. Training and test data are from the same person doing the same tasks but different trials.

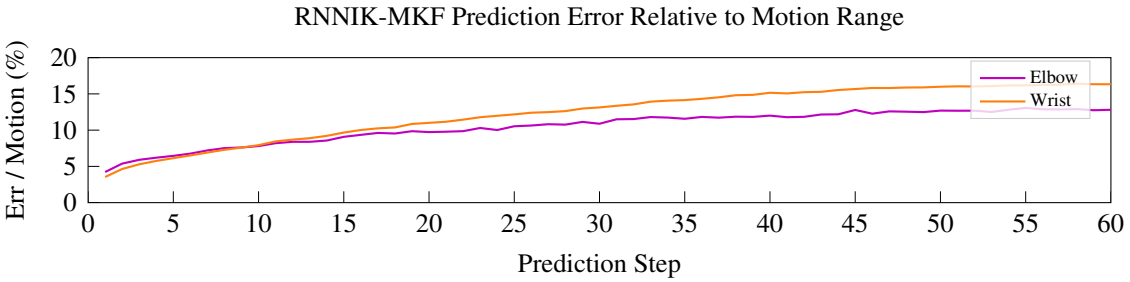


Figure 5.5: Prediction error relative to motion range.

the comparing methods. Fig. 5.5 shows the ratio of prediction error over the motion range. When predicting within 1s, RNNIK-MKF can maintain the error at around 10% relative to the motion range. For longer prediction step, the percentage can be maintained at around 15%. From the experiments, we can see that the proposed RNNIK-MKF can generate high-fidelity human arm motion predictions that are competitive to state-of-art methods.

5.2.2 Unseen Humans Experiments

We test the methods with trajectories from human 2 doing tasks 1, 2, 3, to verify the effect of MKF online adaptation. Figure 5.6 indicates that NN-RLS and RNNIK-MKF have significantly smaller prediction errors. Methods without adaptation have large errors and

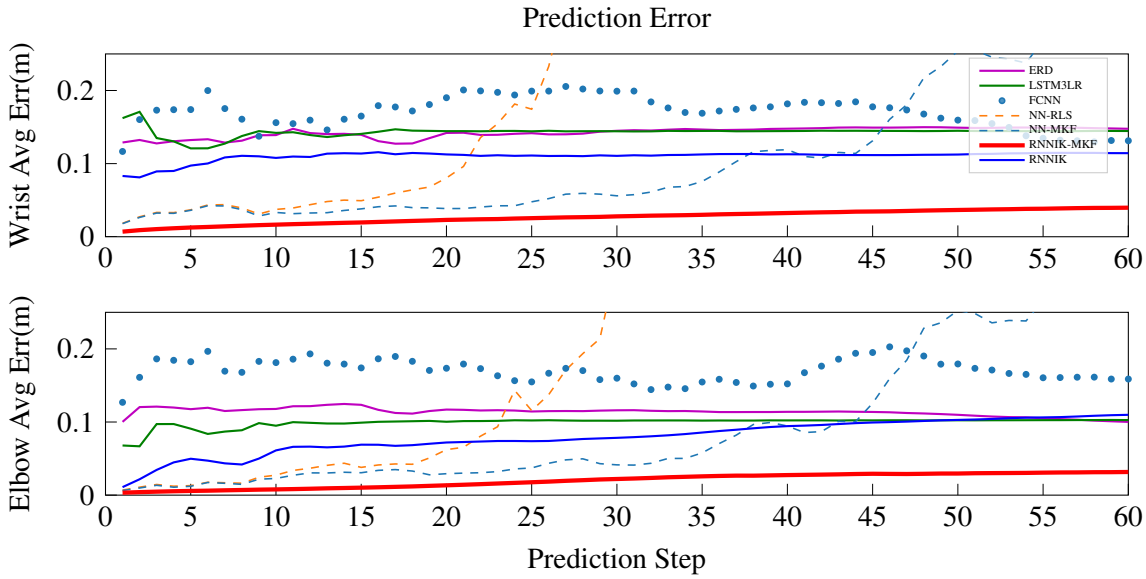


Figure 5.6: Prediction error for 1-60 prediction step. Training and test data are from different people doing the same tasks.

do not achieve similar performance as in the trained environment. In terms of adaptation, the RLS-PAA effectively reduces the error generated in FCNN when the prediction step is less than 20. But it explodes for longer prediction steps. On the other hand, RNNIK-MKF stably reduces the error comparing to RNNIK. It keeps the maximum prediction error under 4 cm, which is at least 70% lower than other methods without adaptation. Since the data contains humans with different heights and arm lengths, the results demonstrate that the method is robust across different scales of motion.

To directly compare the performance of MKF and RLS-PAA, we apply the MKF to FCNN. Figure 5.6 demonstrates that NN-RLS and NN-MKF perform similarly for a short prediction horizon. NN-MKF is able to maintain the error stable for a longer prediction. However, it still explodes for predicting more than 35 steps. Thus, we can conclude that the proposed MKF is more robust than RLS-PAA for online adaptation in this context since it has a virtual noise model and internal smoothing techniques. In addition, RNNIK outperforms FCNN since NN-MKF is still unstable for longer prediction compared to RNNIK-MKF.

In addition, we investigate the adaptation error of MKF. Figure 5.7 shows the root mean square error (RMSE) for wrist and arm predictor with and without MKF. There exists a peak

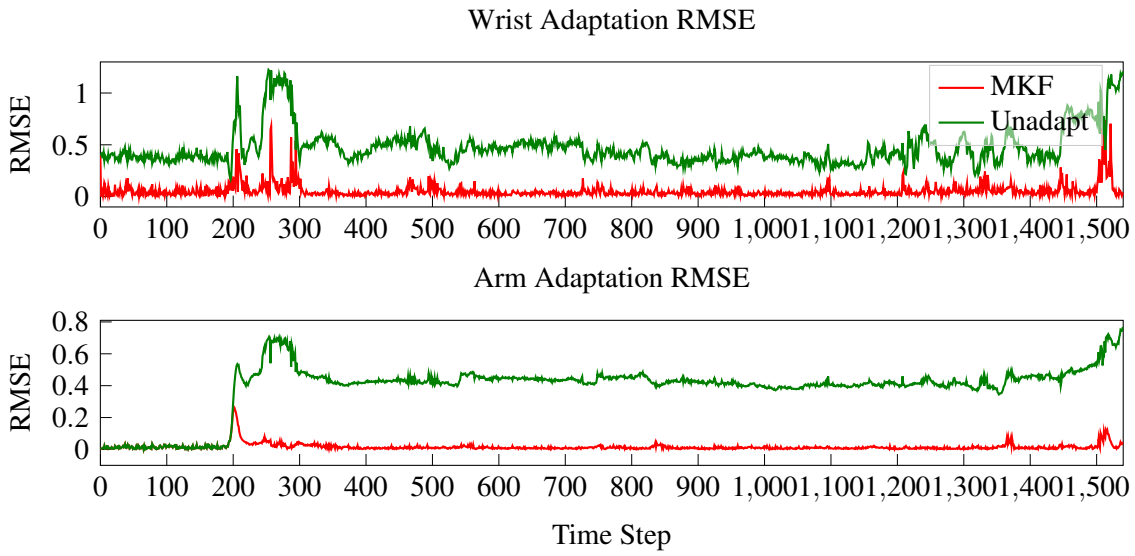


Figure 5.7: RMSE of the error for online adaptation.

at time 200 for both, which is caused by sudden direction change by the human. But the peak is much smaller when MKF is turned on. There exist multiple smaller spikes for the wrist. This is because we do not encode the context (*e.g.*, task state), while the wrist motion heavily depends on the context. On the other hand, there are fewer spikes for the elbow, since the behavior remains relatively stable throughout the task. The plot demonstrates that MKF effectively reduces the error caused by the model mismatch. Figure 5.7 also indicates a fast convergence by MKF. The RMSE for arm settles within 100 steps, which is equal to 3.3 s. The RMSE for wrist settles faster within 20 steps, which is around 0.67 s. The settling times are different because the wrist adaptation starts with a parameter matrix $\phi = \phi_{trained}$, which has prior knowledge encoded, while the arm adaptation starts with $A = I$.

5.2.3 Unseen Tasks Experiments

We also test the online adaptation using trajectories from humans 1 and 3 doing tasks 4 and 5. Figure 5.8 shows that the proposed RNNIK-MKF achieves the lowest prediction errors among all. The tasks in training and testing have different high-level features but are likely to share similar low-level features since they are done by the same humans. From the results in Fig. 5.8, the adaptation successfully adapts the network to the new tasks,

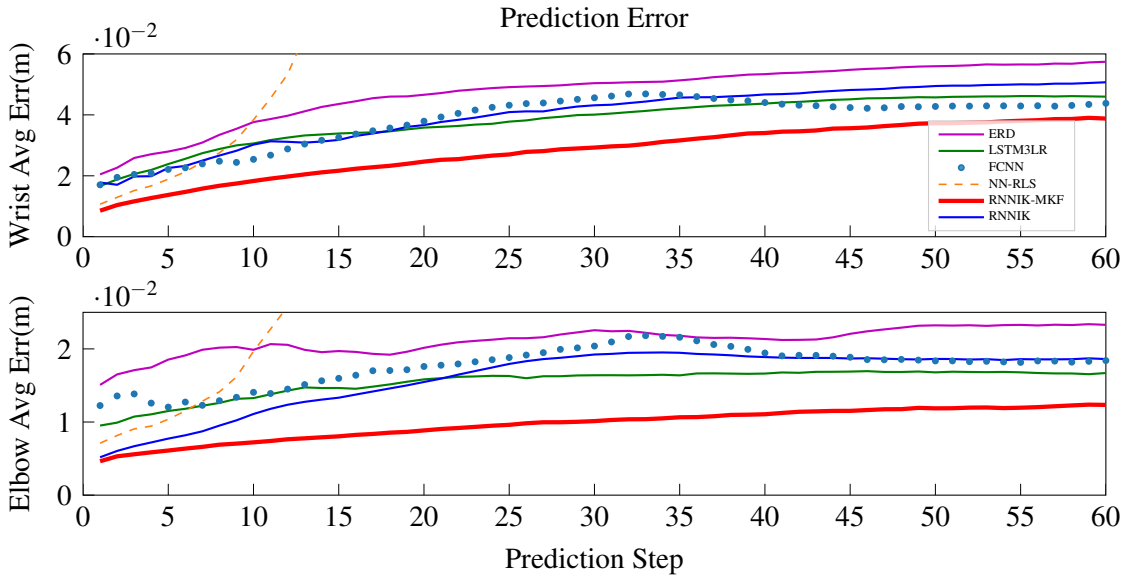


Figure 5.8: Prediction error for 1-60 prediction step. Training and test data are from the same person doing different tasks.

hence, makes the model generalize to new tasks.

5.2.4 Motion Occlusion Experiments

We use the same data from section 5.2.1 to test the situation with occlusion. Several random segments are blocked to the pipeline. We mainly consider the occlusion on the elbow joint. When occlusion happens, the framework assumes that the observation aligns with the prediction and turns off the adaptation as shown in algorithm 2. Hence, the occlusion problem is essentially equivalent to a longer-term prediction problem under the N-to-1 structure. For instance, when at time k, the second step predictions, $\hat{w}_2^k$ and $\hat{e}_2^k$, are predicted with input being $[w_{k-N+2}; e_{k-N+2}; ...; w_k; e_k; \hat{w}_1^k; \hat{e}_1^k]$. This is equivalent to when occlusion happens at time $k+1$. The first step predictions at time $k+1$, $\hat{w}_1^{k+1}$ and $\hat{e}_1^{k+1}$, are predicted with input being $[w_{k-N+2}; e_{k-N+2}; ...; w_k; e_k; \hat{w}_1^k; \hat{e}_1^k]$.

Therefore, we only consider the error for the first step prediction and we use the error metric $E = \frac{1}{T}\sum_i^T (||\hat{o}_1^i - \hat{p}_1^i||_2)$ to describe the variation between the predictions. T denotes the total steps of occlusion and $\hat{o}_1^i$ denotes the prediction with occlusion. The algorithm is considered robust if the variation is small between the predictions with and without

Algorithm	Wrist Variation (m)	Elbow Variation (m)
ERD	0.05537	0.02591
LSTM-3LR	0.03059	0.01468
FCNN	0.01795	0.01619
RNNIK	**0.0**	0.03042
RNNIK-MKF	**0.0**	**0.01164**

Table 5.1: Motion prediction variation with occlusion.

occlusion.

Table 5.1 shows the variation for the wrist and elbow predictions of each algorithm. ERD and LSTM-3LR have significantly larger variations for wrist predictions than elbow predictions although the occluded joint is elbow. This is because the models couple the wrist and elbow in prediction. The elbow is determined by the upper three joints while the wrist is affected by all five joints. Thus, the wrist predictions have larger accumulated variation. On the other hand, since RNNIK and RNNIK-MKF decouple the predictions with arm model, the occlusion of the intermediate arm does not influence the prediction of the wrist. However, RNNIK has the largest elbow variation while RNNIK-MKF has the lowest elbow variation, which is 20% lower than others. This demonstrates that MKF successfully reduces the error by mismatched models. From table 5.1, we can infer that the proposed RNNIK-MKF is more robust compared to the existing methods when the arm motion is partially blocked from view.

5.2.5 Online Adaptation Uncertainty

As MKF adjusts the model parameters online, it is important to quantify the uncertainty of the prediction output. As discussed in section 4.1.4, the prediction uncertainty can be quantified by 1) the statistical standard deviation of the prediction error and 2) the propagation of the MSEE. Figure 5.9 shows an example of the prediction uncertainty of RNNIK-MKF. The purple line indicates the absolute prediction error, which is calculated as $e_k = ||\hat{y}_k - y_k||$. The blue dashed line indicates the statistical standard deviation of the prediction error e_k. The green line illustrates the MSEE propagation, which is calculated as $\sigma = \sqrt{Z_k^{\hat{y}}}$.

As shown in Fig. 5.9, the σ value offers an upper bound of the prediction uncertainties as the MKF adjusts the model parameters online. The prediction error (purple) lies beneath

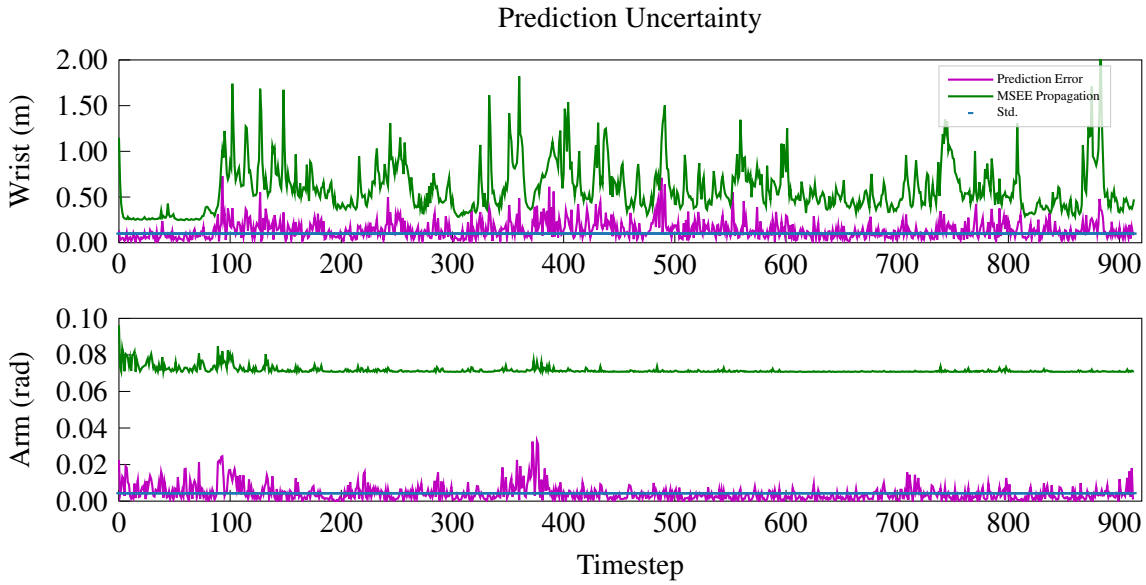


Figure 5.9: Prediction uncertainty during MKF online adaptation. Green: MSEE Propagation. Blue: Standard deviation of prediction error. Purple: Prediction error.

the σ bound (green). In addition, the MSEE increases as the prediction error changes (model parameters change). This behavior indicates that MSEE is able to capture the time-varying property of the prediction system. Moreover, we observe that the prediction error and the σ bound are more stable for the arm than the wrist. This corresponds to our assumption that the wrist motion is more complex and prone to change, while the arm motion is more stable since it mainly supports the wrist. However, unlike the MSEE, the statistical standard deviation does not properly bound the prediction error in real-time as the error (purple) easily goes beyond the standard deviation bound (blue).

5.3 Intention Prediction with IADA

We use a single-layer fully-connected neural network (FCNN) with the hidden neuron size being 32 and the ReLU activation function to construct the classifier for intention prediction. We compare three training methods, including the regular supervised training (REG), the robust training via the convex outer adversarial polytope (COAP) [52], and our IADA training. We evaluate the effectiveness of the proposed IADA training framework by comparing the classification performance of the trained FCNNs. In addition to the intention

Table 5.2: Comparison of the FCNN classification accuracy on 2D binary classification. REG+DA and COAP+DA are the training methods with uniform random data augmentation with the same amount of data augmented by IADA. D: data size. E: epochs.

	REG	REG + DA	COAP[52]	COAP + DA	IADA
D: 1k, E: 1k	97.16%	95.62%	97.22%	**98.01%**	97.62%
D: 500, E: 1k	94.97%	94.97%	**95.89%**	95.69%	95.53%
D: 500, E: 10k	97.93%	98.46%	97.61%	98.55%	**99.01%**
D: 500, E: 50k	97.78%	98.99%	97.31%	99.39%	**99.51%**

prediction problem, we test the IADA training in two additional applications, including a 2D artificial binary classification problem shown in Fig. 1.3(a), and the MNIST digits classification problem [28]. The formal verification is implemented using the MIPVerify in NeuralVerification.jl toolbox [31]. The regular and IADA training use the Cross-Entropy loss with the Adam optimizer [24] implemented in PyTorch [39]. The learning rate is set to 0.01 for all learning methods. The ε values are set to 0.1, 0.1, and 0.05 for the 2D, MNIST, and intention problems respectively. For IADA, the verify rate is set to be $r_v = 500$, the verification number C is set to be 5000, 2000, and 5000 for the 2D, MNIST, and intention problems respectively. All experiments are run in Windows 10 with AMD Ryzen 3700X 8-Core processor, 16GB RAM, and an RTX 2070 Super GPU.

5.3.1 2D Binary Classification

The 2D toy example mainly provides better visualization and an intuitive understanding of the IADA training process. Figure 1.3(a) shows the ground truth of the problem, where we have two classes shown in blue and purple. Given an available training dataset, we want the FCNN classifier to recover the true decision boundary. The scheduler is implemented only using the distance check since the 2D data point does not have a semantic meaning. During the human verification, ideally, the human assigns the ground truth label. But in this experiment, we assume the human has the ground truth knowledge and directly use the ground truth to automatically assign the labels. We evaluate the training quality by visualizing the decision boundaries of the classifier (shown in Fig. 5.10) and testing the classification accuracy (shown in table 5.2). Note that the result in this example for COAP has $\varepsilon = 0.01$. Since the adversarial bound computed in COAP is over-approximated, enlarging the robustness to $\varepsilon = 0.1$ will result in too many false adversaries incorrectly

(a) REG: D:1k, E:1k. (b) REG: D:500, E:1k. (c) REG: D:500, E:10k. (d) REG: D:500, E:50k.

(e) COAP: D:1k, E:1k. (f) COAP: D:500, E:1k. (g) COAP: D:500, E:10k. (h) COAP: D:500, E:50k.

(i) IADA: D:1k, E:1k. (j) IADA: D:500, E:1k. (k) IADA: D:500, E:10k. (l) IADA: D:500, E:50k.

Figure 5.10: Visualizations of the decision boundaries of the trained FCNN. D denotes the training data size. E indicates the number of training epochs.

taken as true adversaries, which will fail the training.

We start with 1000 training samples. From table 5.2, we can see that all trained classifiers have decent classification accuracy, which is above 95%, since we have sufficient training data available. We then decrease the training data size to 500 to simulate the situation with insufficient training data. As shown in table 5.2, with limited training epochs, all trained models have worse accuracy. When we increase the training epochs, we observe that the trained models have higher accuracy, where IADA has the highest among all. However, as shown in Fig. 5.10(c) and Fig. 5.10(g), the recovered regions skew and overfit the training data, whereas IADA recovers similar regions as the ground truth as shown in Fig. 5.10(k). As we further increase the number of training epochs to 50000, we can see that there is more overfitting by regular training and COAP as the accuracy starts to drop. On the other hand, IADA further refines the decision boundary and improves classification accuracy. Figure 1.3(a) shows the data augmentation process by IADA. The augmented data converges to the true decision boundary as mentioned in section 4.2.5.

To demonstrate the effectiveness of using adversaries for data augmentation, we add

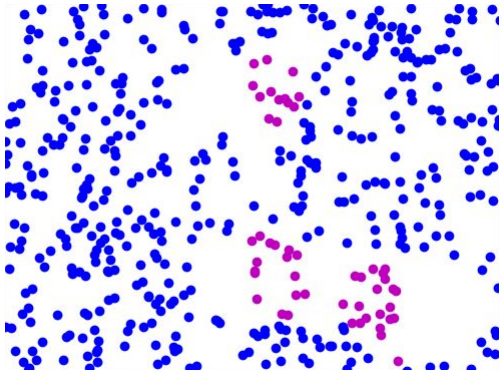


Figure 5.11: Biased 2D training data. Data size: 500. The blue class is uniformly sampled, while the purple class is only available in half of the rectangle and the two circles. No blue points are sampled between the circles. The number of blue vs purple is 9:1.

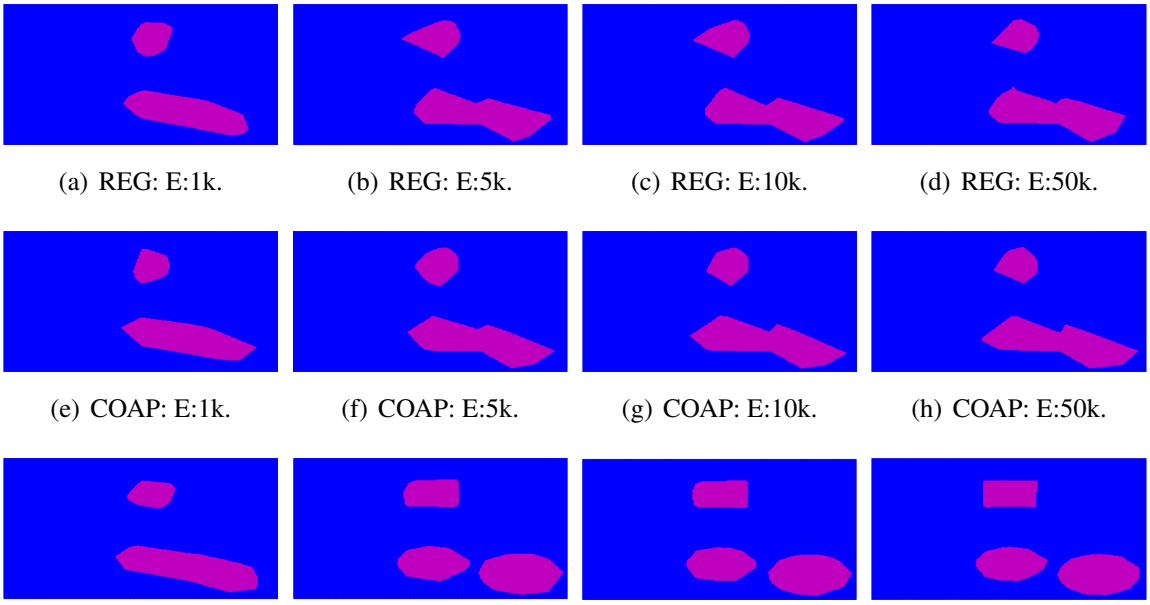
Table 5.3: Comparison of the FCNN classification accuracy on 2D binary classification with biased training data.

	REG	COAP	IADA
Epochs: 1k	90.65%	90.70%	**92.75%**
Epochs: 5k	93.07%	92.71%	**99.07%**
Epochs: 10k	93.13%	93.29%	**99.35%**
Epochs: 20k	93.62%	93.32%	**99.41%**
Epochs: 30k	93.81%	93.08%	**99.21%**
Epochs: 40k	93.67%	93.20%	**99.54%**
Epochs: 50k	92.59%	93.48%	**99.33%**

the same amount of data by random sampling to the original training dataset, which is uniformly sampled, for REG and COAP. The model accuracy is shown in the second and fourth columns of table 5.2. We can see that although the performance improves for both methods, IADA still achieves the highest accuracy. This demonstrates that the adversaries are more informative than uniformly sampled data. IADA is more effective and requires fewer additional data for training.

5.3.2 2D Binary Classification with Biased Data

We further study the IADA training performance with biased training data on the 2D classification problem. The biased training data is shown in Fig. 5.11. The data size is 500, which is the same as the experiments presented in section 5.3.1. However, 450 points belong to the blue class, whereas only 50 points belong to the purple class. In addition, the blue points are uniformly distributed, whereas the purple points are only available in half of the clusters (*i.e.*, the right half of the rectangle, the right half of the left circle, and the left half of the right circle). Therefore, the purple class is locally biased. Also, there are no blue data points sampled between the two purple circles, making the local distribution between the circles different from the ground truth. The training parameters are identical to the setting in section 5.3.1.



(a) REG: E:1k. (b) REG: E:5k. (c) REG: E:10k. (d) REG: E:50k.

(e) COAP: E:1k. (f) COAP: E:5k. (g) COAP: E:10k. (h) COAP: E:50k.

(i) IADA: E:1k. (j) IADA: E:5k. (k) IADA: E:10k. (l) IADA: E:50k.

Figure 5.12: Visualizations of the decision boundaries of the trained FCNN learned from the biased training data. E indicates the number of training epochs.

Table 5.3 shows the classification accuracy of the models learned by different training algorithms. Using biased training data, it is obvious that the NNs learned by REG and COAP have significantly lower accuracy. REG initially improves the accuracy but soon starts to decrease the model accuracy due to overfitting. Similarly, COAP struggles to further recover the NN decision boundary. On the other hand, IADA continuously improves the model and achieves better classification performance. Figure 5.12 shows the visualizations of the learned decision boundaries. After the first 1000 epochs, all methods perform similarly. As shown in Fig. 5.12(a), 5.12(e) and 5.12(i), all models are not able to correctly distinguish the three purple clusters. However, as the training runs for more epochs, REG and COAP continue to fail to recover the three clusters, while IADA is able to correctly distinguish the three clusters as shown in Fig. 5.12(j) to 5.12(l). The shapes recovered by IADA are similar to the ground truth. Figure 5.13 demonstrates the development of the learned NN model and the augmented adversarial data given the learned model. By iterative data augmentation and expert guidance, IADA can correctly fill the missing halves of the rectangle and circles. It is also able to expand the blue points to fill the empty space between the circles to recover the ground truth.

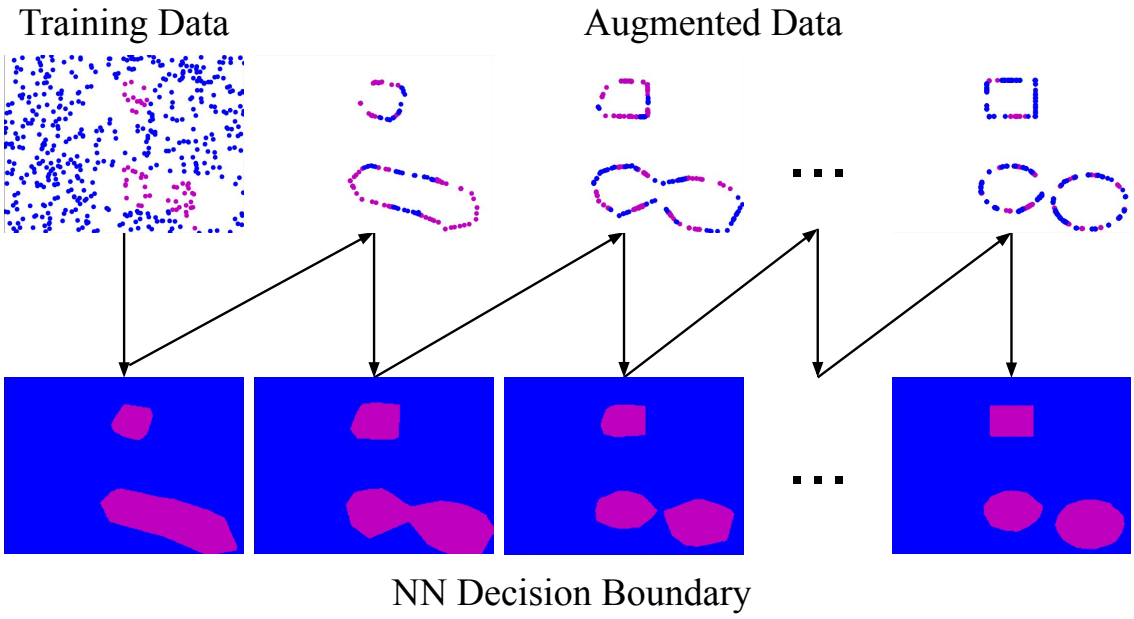


Figure 5.13: Example of the learning process by IADA on the 2D biased task. Training Data: original biased training data. First row: adversarial data augmented by formal verification and expert guidance given the current NN model. Second row: NN decision boundary learned by augmenting the data.

5.3.3 MNIST

MNIST is a more realistic classification problem in a higher dimension. It is rare to have adversarial images that humans cannot interpret the ground truth label. Thus, the scheduler is implemented only using the distance check. During the human verification, the human is given the adversarial image and asked to assign the ground truth label. Table 5.4 shows the performance of different training methods on the MNIST dataset. Each entry includes the classification accuracy using the MNIST testing dataset and the average perturbation bound of the trained model. Given the testing data $D_t = \{x_t, y_t\}_{t=1}^{n}$, the average perturbation bound is calculated as

$$p_b = \frac{1}{n}\Sigma_{t=1}^{n}\delta_t, \tag{5.2}$$

where $\delta_t = 0$ if $f_\theta(x_t) \neq y_t$. If $f_\theta(x_t) = y_t$, δ_t is the solution to (4.10) on the point x_t. Figure 1.3(b) shows a visualization of the adversarial images found by IADA and the iterative expansion. As we expand for more iterations, the adversarial images have more perturbations added relative to the original image. From table 5.4, we can see that IADA, in general, achieves the highest accuracy, which indicates that the model learned a better

Table 5.4: Comparison of the FCNN classification accuracy on MNIST digits classification. Each entry is in the format of accuracy (average perturbation bound). D: data size. E: epochs

	REG	COAP [52]	IADA
D: 3k, E: 1k	87.60% (0.00719)	88.30% (0.01393)	**89.60% (0.01491)**
D: 2k, E: 1k	84.85% (0.00689)	85.30% **(0.01346)**	**87.05%** (0.01309)
D: 1k, E: 1k	82.45% (0.00821)	83.95% **(0.01495)**	**85.55%** (0.01361)
D: 500, E: 1k	80.80% (0.00975)	80.90% **(0.01599)**	**82.55%** (0.01301)
D: 500, E: 2k	80.80% (0.01006)	80.75% **(0.01521)**	**82.85%** (0.01094)
D: 500, E: 3k	80.70% (0.00972)	80.65% (0.01492)	**83.10% (0.01641)**

decision boundary. However, COAP achieves a larger average perturbation bound in general. This is mainly due to the different objectives for the two training methods. The objective for COAP is to enlarge the perturbation bound, whereas IADA focuses on recovering the true decision boundary. IADA expands the input space by querying new data, which might lead to a slower expansion comparing to COAP. However, the expansion is safe due to the expert guidance, and thus, leads to better classification accuracy than COAP.

5.3.4 Intention Prediction

We apply IADA to learn the intention prediction model discussed in section 3.2. The FCNN is trained using the first two trials of human subject 1 doing task 1 and tested using the third trial. The input to the FCNN is the previous 10-step (0.3s) historical trajectories for both right and left wrists and the output is the intention label. Figures 1.3(c) and 5.15 show the adversarial wrist trajectories being added to the dataset by IADA. The model validation accuracy is shown in table 5.5. We can see that the accuracy for REG and COAP increase initially. But soon the accuracy starts to drop in regular training due to the overfitting while COAP stays constant. On the other hand, by using expert knowledge and iterative expansion, IADA is able to continuously improve the model accuracy. As we further increase the training epochs, we expect the prediction accuracy by the learned FCNN will further increase.

5.3.5 Intention Prediction Confidence

In the intention prediction context, we model the input noise as $X \sim N(0, \frac{0.05}{3}) = N(0, 0.0167)$, which is a normal distribution with the 3σ value corresponding to the IADA epsilon value

Table 5.5: Comparison of the FCNN classification accuracy on human intention prediction.

	REG	COAP[52]	IADA
Epochs: 500	81.99%	77.94%	**82.53%**
Epochs: 1k	**85.92%**	77.95%	85.48%
Epochs: 2k	85.26%	81.66%	**88.75%**
Epochs: 3k	82.97%	83.95%	**89.52%**
Epochs: 4k	82.86%	83.95%	**90.83%**
Epochs: 5k	81.55%	83.95%	**92.03%**

$\varepsilon = 0.05$. The prediction confidence is calculated as shown in Eq. (4.11) using the perturbation bound. Figure 5.14 shows a comparison of the prediction confidence using the FCNN prediction model trained by REG, COAP, and IADA. We can see that, first, the prediction model trained by IADA has significantly fewer incorrect predictions, which is also shown in table 5.5. Secondly, IADA can effectively enlarge the true perturbation bound, making the incorrect predictions have lower confidence while correct predictions have higher confidence. On the other hand, the model trained using REG has many incorrect predictions with high confidence. The predictions by the model trained using COAP have significantly higher overall confidence, which indicates larger perturbation bounds. This is due to the COAP objective, which is to enlarge the perturbation bound. However, both correct and incorrect predictions have high confidence values since COAP mixes the true and false adversaries during training. Therefore, the confidence value becomes uninformative. The confidence value is more informative on the prediction model learned by IADA. Therefore, it indicates that IADA recovers a better prediction model closer to the ground-truth model.

5.3.6 Visualization of IADA for Intention Prediction

In this section, we show more visualizations of the adversaries generated during the training of the intention prediction classifier. Human motion has non-trivial distribution around the decision boundary. It is likely to have false adversaries close to the training data and can easily have "confused" decision boundaries. This confusion also applies to human experts. In order to minimize the confusion, instead of solely visualizing the 3D trajectory points, we project the trajectory on the image along with the background for the human expert to better understand the context.

Figure 5.15 shows the visualizations of the adversaries generated by IADA. The first

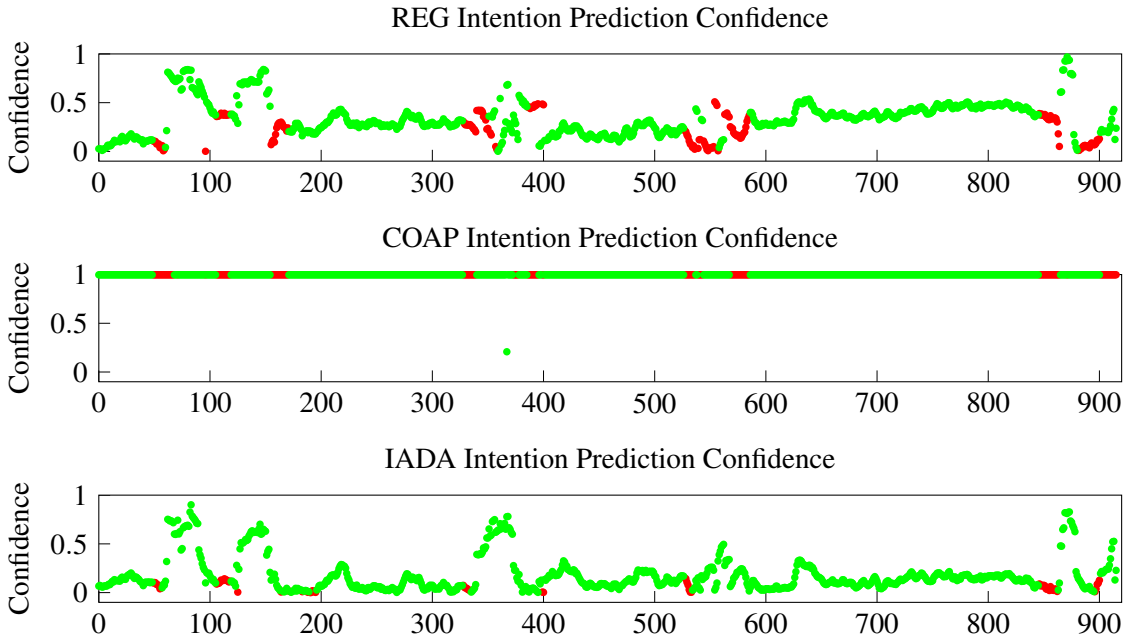


Figure 5.14: Intention prediction confidence comparison of the prediction model learned by different training methods. Green: correct prediction. Red: incorrect prediction.

row shows the true adversaries, where the generated adversaries share the same intention label as the root. We can see from the images that the red trajectories are close to the white trajectories, and share the same motion pattern. The second row shows the false adversaries. These adversaries got different labels assigned by the expert guidance. For instance, the left adversarial trajectory in Fig. 5.15(e) has significantly increased vertical translation comparing to the original motion, which makes it assigned as *Reach*. In Fig. 5.15(f), the original intention is *Assemble*. However, the adversarial attack makes the right trajectory have a large translation. The translation motion makes the expert assigned it to be *Reach*. The third row shows the ambiguous adversaries. Due to the naive design of the scheduler, there exist adversaries assigned to the human expert that do not have a clear meaning.

Also, note that the labeling process is subjective to the human expert. Since trajectories can be ambiguous to humans, it is important to design an intuitive user interface to enable better labeling. For the current visualization, we project the adversaries onto images with the background. However, this will eliminate the information in the normal direction of the image. In the future, we will investigate methods to better visualizing and labeling the adversarial trajectories (*i.e.*, simultaneously visualize both 2D and 3D trajectories or

(a) GT: *Retrieve*. EG: *Retrieve*. (b) GT: *Retrieve*. EG: *Retrieve*. (c) GT: *Assemble*. EG: *Assemble*.

(d) GT: *Assemble*. EG: *Retrieve*. (e) GT: *Assemble*. EG: *Reach*. (f) GT: *Assemble*. EG: *Retrieve*.

(g) GT: *Reach*. EG: N/A. (h) GT: *Retrieve*. EG: N/A. (i) GT: *Retrieve*. EG: N/A.

Figure 5.15: Visualizations of the adversarial trajectories generated online by IADA. White: original trajectory. Red: adversarial trajectory. GT: the intention label of the original trajectory. EG: the intention label assigned by expert guidance to the adversary.

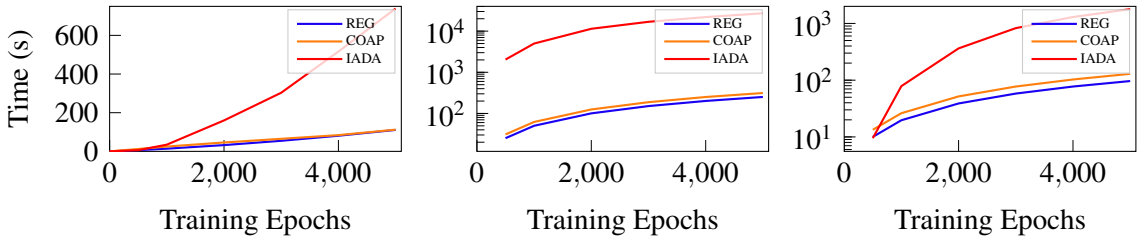


Figure 5.16: Training time comparison of different training methods on different tasks. Left: 2D binary. Middle: MNIST. Right: Intention Prediction.

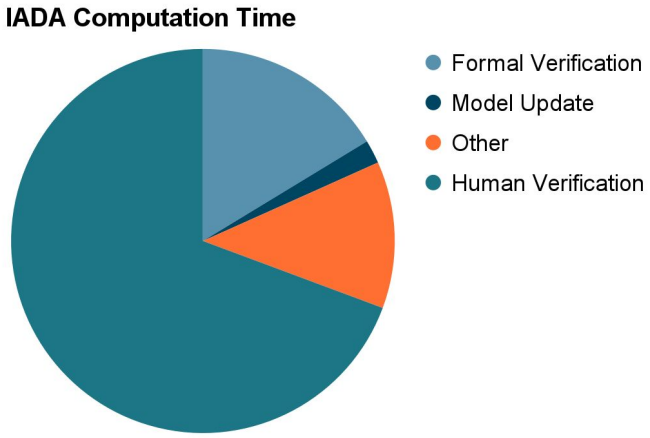


Figure 5.17: Time cost decomposition for IADA training on 2D classification. Model update refers to training using $D_0 \cup D_{adv}$. Other refers to data operation (*i.e.*, priority queue operation).

directly synthesize a video corresponding to each adversarial trajectory).

5.3.7 Discussion

Time Efficiency Figure 5.16 shows the training time for each method in each problem. We can see that COAP and REG have very comparable time costs. IADA is significantly more expansive in terms of computation time. On the 2D problem, it is around 5 to 7$\times$ the time of REG and COAP, but around 20$\times$ and 100$\times$ on larger problems, *i.e.*, MNIST and intention prediction. We can expect the time cost to be higher when scaling to more complex and larger NN structures. Figure 5.17 shows the time decomposition of the IADA training on the 2D example. We can see the most expensive part is the human verification since the

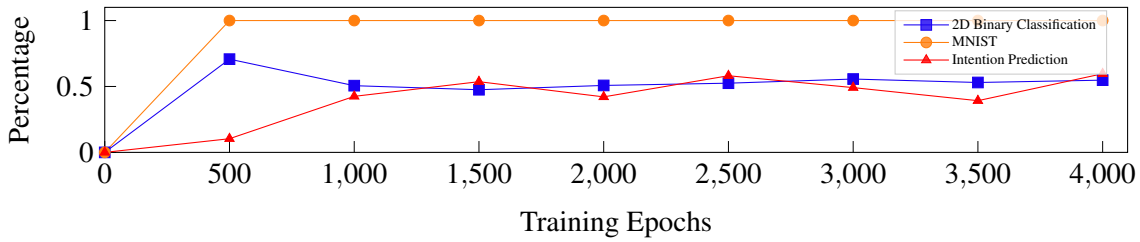


Figure 5.18: Percentage of true adversarial data found by formal verification online.

problem requires more expert guidance. Due to the low dimensionality of the problem, the formal verification takes a smaller ratio in the total computation time. Similarly for MNIST and intention prediction, human verification and formal verification takes more than 85% of the total training time, but formal verification has a significantly larger ratio. This is mainly due to the problem requires less human verification for MNIST, and the formal verification is much more computationally expensive on higher-dimensional problems. However, note that the comparison established here is slightly unfair since both REG and COAP are implemented solely in Python, whereas IADA has the formal verification implemented in Julia. The formal verification time accounts for the time exchanging between platforms.

Human Effort It is interesting to observe that COAP fails on the 2D problem when $\varepsilon = 0.1$. But it works properly when $\varepsilon = 0.01$. For the 2D task and the intention prediction task, the data is more likely to be on the decision boundary, thus, more likely to find false adversaries online. Therefore, expert guidance is critical to make the training safe and robust. Figure 5.18 shows the percentage of the true adversaries generated online during training. It is obvious that MNIST has significantly more true adversaries generated online. Therefore, the required human effort is significantly lower for image-related tasks, *i.e.*, MNIST, compared to other tasks, *i.e.*, intention prediction. By introducing expert guidance, IADA is most suitable for tasks with "confusing" decision boundaries, in which case it is easy to mix true and false adversaries, as discussed in section 4.2.5.

Chapter 6

Conclusion and Future Work

This work studies human behavior prediction (motion prediction and intention prediction) with data scarcity for safe HRC. To overcome data scarcity, this thesis proposes a novel RNNIK-MKF adaptable motion prediction framework and IADA training framework to address the challenge respectively.

The RNNIK-MKF uses the RNN to predict the wrist motion and IK to extend the wrist prediction to full-arm prediction based on the physical arm model. The proposed MKF adapts the model in real-time to the current user or task. By comparing to existing methods, our method outperforms by showing that it can predict the arm motion with 14% lower prediction errors; it is generic to unseen humans or tasks since it has 70% lower prediction errors; it is more robust when partial arm motion is blocked as the occlusion has no impact on wrist prediction and has 20% less influence on elbow prediction.

The IADA framework learns robust NN classifiers from scarce data by acquiring more data offline during the model training. It uses formal verification online to find the most vulnerable part of the network, such as samples on the decision boundary of the network (adversaries). By acquiring human expert knowledge, the framework augments the training data using the adversaries verified by humans and iteratively expands the available data. The experiments demonstrate that our training method can improve the robustness and accuracy of the learned model from initially scarce data. The IADA training is suitable for training robust intention prediction models.

There are many future directions that we would like to pursue.

1. Exploring the potential usage of the end-to-center (wrist to arm) method in other

applications (*i.e.*, full human body motion, robot team exploration).

2. Analyzing the optimality of MKF; developing a strategy for finding optimal hyperparameters (*i.e.*, noise model, system initialization).
3. Formally proving that the IADA framework will ensure convergence of the learned NN model to the ground truth.
4. Exploring efficient online verification algorithms (both implementation-wise and algorithm-wise) to improve the scalability of the IADA framework.
5. Designing a better scheduler in order to balance the required human effort and the learning accuracy.
6. Extending the IADA framework to general regression NN where we will augment new data at places with larger gradients.
7. Exploring the possibility of offering human experts the freedom to create new labels during human verification.
8. Exploring the interaction between motion prediction and intention prediction. Currently, the motion prediction and the intention prediction only consider the past motion observation. However, the motion prediction could be extended to embed contextual information (*i.e.*, intention), whereas the intention prediction could also be extended to consider the motion prediction.

Bibliography

[1] Abulikemu Abuduweili and Changliu Liu. Robust nonlinear adaptation algorithms for multitask prediction networks. *International Journal of Adaptive Control and Signal Processing*, 35(3):314–341, 2021.

[2] R. Anand, K.G. Mehrotra, C.K. Mohan, and S. Ranka. An improved algorithm for neural network classification of imbalanced training sets. *IEEE Transactions on Neural Networks*, 4(6):962–969, 1993.

[3] Antreas Antoniou, Amos Storkey, and Harrison Edwards. Data augmentation generative adversarial networks. *arXiv preprint arXiv:1711.04340*, 2018.

[4] S. Arai, A. L. Pettersson, and K. Hashimoto. Fast prediction of a worker's reaching motion without a skeleton model (f-premo). *IEEE Access*, 8:90340–90350, 2020.

[5] Tao Bai, Jinqi Luo, Jun Zhao, Bihan Wen, and Qian Wang. Recent advances in adversarial training for adversarial robustness. *arXiv preprint arXiv:2102.01356*, 2021.

[6] Osbert Bastani, Yani Ioannou, Leonidas Lampropoulos, Dimitrios Vytiniotis, Aditya Nori, and Antonio Criminisi. Measuring neural net robustness with constraints. In *Advances in Neural Information Processing Systems*, volume 29, 2016.

[7] Christopher Bowles, Liang Chen, Ricardo Guerrero, Paul Bentley, Roger Gunn, Alexander Hammers, David Alexander Dickie, Maria Valdés Hernández, Joanna Wardlaw, and Daniel Rueckert. Gan augmentation: Augmenting training data using generative adversarial networks. *arXiv preprint arXiv:1810.10863*, 2018.

[8] Yujun Cai, Lin Huang, Yiwei Wang, Tat-Jen Cham, Jianfei Cai, Junsong Yuan, Jun Liu, Xu Yang, Yiheng Zhu, Xiaohui Shen, et al. Learning progressive joint propagation for human motion prediction. In *European Conference on Computer Vision*, pages 226–242. Springer, 2020.

[9] Z. Cao, G. Hidalgo Martinez, T. Simon, S. Wei, and Y. A. Sheikh. Openpose: Realtime multi-person 2d pose estimation using part affinity fields. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2019.

[10] Nicholas Carlini and David Wagner. Towards evaluating the robustness of neural networks. *arXiv preprint arXiv:1608.04644*, 2017.

[11] Minhao Cheng, Qi Lei, Pin-Yu Chen, Inderjit Dhillon, and Cho-Jui Hsieh. Cat: Customized adversarial training for improved robustness. *arXiv preprint arXiv:2002.06789*, 2020.

[12] Y. Cheng, W. Zhao, C. Liu, and M. Tomizuka. Human motion prediction using semi-adaptable neural networks. In *2019 American Control Conference (ACC)*, pages 4884–4890, 2019.

[13] Ekin D. Cubuk, Barret Zoph, Dandelion Mane, Vijay Vasudevan, and Quoc V. Le. Autoaugment: Learning augmentation policies from data. *arXiv preprint arXiv:1805.09501*, 2019.

[14] H. Ding, G. Reißig, K. Wijaya, D. Bortot, K. Bengler, and O. Stursberg. Human arm motion modeling and long-term prediction for safe and efficient human-robot-interaction. In *2011 IEEE International Conference on Robotics and Automation*, pages 5875–5880, 2011.

[15] K. Fragkiadaki, S. Levine, P. Felsen, and J. Malik. Recurrent network models for human dynamics. In *2015 IEEE International Conference on Computer Vision (ICCV)*, pages 4346–4354, 2015.

[16] Ian Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. In *International Conference on Learning Representations*, 2015.

[17] G.C. Goodwin and K.S. Sin. *Adaptive Filtering Prediction and Control*. Dover Books on Electrical Engineering. Dover Publications, 2014. ISBN 9780486137728.

[18] Haibo He and Edwardo A. Garcia. Learning from imbalanced data. *IEEE Transactions on Knowledge and Data Engineering*, 21(9):1263–1284, 2009.

[19] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. 9(8):1735–1780, nov 1997.

[20] Peter J. Huber. Robust estimation of a location parameter. *Annals of Mathematical Statistics*, 35(1):73–101, March 1964.

[21] A. Jain, A. R. Zamir, S. Savarese, and A. Saxena. Structural-rnn: Deep learning on spatio-temporal graphs. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5308–5317, 2016.

[22] Nathalie Japkowicz and Shaju Stephen. The class imbalance problem: A systematic study. *Intelligent Data Analysis*, pages 429–449, 2002.

[23] J. Kiefer and J. Wolfowitz. Stochastic estimation of the maximum of a regression function. *Ann. Math. Statist.*, 23(3):462–466, 09 1952.

[24] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2017.

[25] Bartosz Krawczyk. Learning from imbalanced data: open challenges and future

directions. *Progress in Artificial Intelligence*, 5(4):221–232, 2016.

[26] C. T. Landi, Y. Cheng, F. Ferraguti, M. Bonfè, C. Secchi, and M. Tomizuka. Prediction of human arm target for robot reaching movements. In *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 5950–5957, 2019.

[27] P.A. Lasota, T. Song, and J.A. Shah. *A Survey of Methods for Safe Human-Robot Interaction*. Foundations and Trends(r) in Robotics Series. Now Publishers, 2017. ISBN 9781680832785.

[28] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[29] Joseph Lemley, Shabab Bazrafkan, and Peter Corcoran. Smart augmentation learning an optimal data augmentation strategy. *IEEE Access*, 5:5858–5869, 2017.

[30] Changliu Liu and Masayoshi Tomizuka. Safe exploration: Addressing various uncertainty levels in human robot interactions. In *2015 American Control Conference (ACC)*, pages 465–470, 2015.

[31] Changliu Liu, Tomer Arnon, Christopher Lazarus, and Mykel J. Kochenderfer. NeuralVerification.jl: Algorithms for verifying deep neural networks. In *Workshop on Debugging Machine Learning,*, 2019.

[32] Changliu Liu, Tomer Arnon, Christopher Lazarus, Christopher Strong, Clark Barrett, and Mykel J. Kochenderfer. Algorithms for verifying deep neural networks. *arXiv preprint arXiv:1903.06758*, 2020.

[33] Ruixuan Liu and Changliu Liu. Human motion prediction using adaptable recurrent neural networks and inverse kinematics. *IEEE Control Systems Letters*, 5(5):1651–1656, 2021.

[34] Alessio Lomuscio and Lalit Maganti. An approach to reachability analysis for feed-forward relu neural networks. *arXiv preprint arXiv:1706.07351*, 2017.

[35] J. Mainprice, R. Hayne, and D. Berenson. Predicting human reaching motion in collaborative tasks using inverse optimal control and iterative re-planning. In *2015 IEEE International Conference on Robotics and Automation (ICRA)*, pages 885–892, 2015.

[36] J. Martinez, M. J. Black, and J. Romero. On human motion prediction using recurrent neural networks. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4674–4683, 2017.

[37] Eloise Matheson, Riccardo Minto, Emanuele G. G. Zampieri, Maurizio Faccio, and Giulio Rosati. Human–robot collaboration in manufacturing applications: A review. *Robotics*, 8(4):100, Dec 2019.

[38] Seyed-Mohsen Moosavi-Dezfooli, Alhussein Fawzi, and Pascal Frossard. Deepfool: a simple and accurate method to fool deep neural networks. *arXiv preprint*

arXiv:1511.04599, 2016.

[39] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems 32*, pages 8024–8035. 2019.

[40] Luis Perez and Jason Wang. The effectiveness of data augmentation in image classification using deep learning. *arXiv preprint arXiv:1712.04621*, 2017.

[41] Lingteng Qiu, Xuanye Zhang, Yanran Li, Guanbin Li, Xiaojun Wu, Zixiang Xiong, Xiaoguang Han, and Shuguang Cui. Peeking into occluded joints: A novel framework for crowd pose estimation. *arXiv preprint arXiv:2003.10506*, 2020.

[42] Stephane Ross, Geoffrey Gordon, and Drew Bagnell. A reduction of imitation learning and structured prediction to no-regret online learning. In *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*, volume 15 of *Proceedings of Machine Learning Research*, pages 627–635. PMLR, 11–13 Apr 2011.

[43] Andrey Rudenko, Luigi Palmieri, Michael Herman, Kris M Kitani, Dariu M Gavrila, and Kai O Arras. Human motion trajectory prediction: a survey. *The International Journal of Robotics Research*, 39(8):895–935, Jun 2020.

[44] Hojjat Salehinejad, Sharan Sankar, Joseph Barfett, Errol Colak, and Shahrokh Valaee. Recent advances in recurrent neural networks. *arXiv preprint arXiv:1801.01078*, 2018.

[45] Connor Shorten and T. Khoshgoftaar. A survey on image data augmentation for deep learning. *Journal of Big Data*, 6:1–48, 2019.

[46] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. Intriguing properties of neural networks. In *International Conference on Learning Representations*, 2014.

[47] Vincent Tjeng, Kai Xiao, and Russ Tedrake. Evaluating robustness of neural networks with mixed integer programming. *arXiv preprint arXiv:1711.07356*, 2019.

[48] Dimitris Tsipras, Shibani Santurkar, Logan Engstrom, Alexander Turner, and Aleksander Madry. Robustness may be at odds with accuracy. *arXiv preprint arXiv:1805.12152*, 2019.

[49] Valeria Villani, Fabio Pini, Francesco Leali, and Cristian Secchi. Survey on human–robot collaboration in industrial settings: Safety, intuitive interfaces and applications. *Mechatronics*, 55:248 – 266, 2018.

[50] C. Welman. *Inverse Kinematics and Geometric Constraints for Articulated Figure*

Manipulation [microform]. Canadian theses on microfiche. Thesis (M.Sc.)–Simon Fraser University, 1993. ISBN 9780315912564.

[51] Tsui-Wei Weng, Huan Zhang, Hongge Chen, Zhao Song, Cho-Jui Hsieh, Duane Boning, Inderjit S. Dhillon, and Luca Daniel. Towards fast computation of certified robustness for relu networks. *arXiv preprint arXiv:1804.09699*, 2018.

[52] Eric Wong and Zico Kolter. Provable defenses against adversarial examples via the convex outer adversarial polytope. In *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 5286–5295. PMLR, 10–15 Jul 2018.

[53] A. M. Zanchettin, A. Casalino, L. Piroddi, and P. Rocco. Prediction of human activity patterns for human–robot collaborative assembly tasks. *IEEE Transactions on Industrial Informatics*, 15(7):3934–3942, 2019.

[54] Jingfeng Zhang, Xilie Xu, Bo Han, Gang Niu, Lizhen Cui, Masashi Sugiyama, and Mohan Kankanhalli. Attacks which do not kill training make adversarial learning stronger. In Hal Daumé III and Aarti Singh, editors, *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pages 11278–11287. PMLR, 13–18 Jul 2020.

[55] Zhun Zhong, Liang Zheng, Guoliang Kang, Shaozi Li, and Yi Yang. Random erasing data augmentation. *Proceedings of the AAAI Conference on Artificial Intelligence*, 34 (07):13001–13008, Apr. 2020.