A Continuous Learning Approach for Probabilistic Human Motion Prediction

Jie Xu*, Shihong Wang*, Xingyu Chen, Jiahao Zhang, Xuguang Lan†, and Nanning Zheng

***Abstract*— Human Motion Prediction (HMP) plays a crucial role in safe Human-Robot-Interaction (HRI). Currently, the majority of HMP algorithms are trained by massive pre-collected data. As the training data only contains a few pre-defined motion patterns, these methods cannot handle the unfamiliar motion patterns. Moreover, the pre-collected data are usually non-interactive, which does not consider the real-time responses of collaborators. As a result, these methods usually perform unsatisfactorily in real HRI scenarios. To solve this problem, in this paper, we propose a novel Continual Learning (CL) approach for probabilistic HMP which makes the robot continually learns during its interaction with collaborators. The proposed approach consists of two steps. First, we leverage a Bayesian Neural Network to model diverse uncertainties of observed human motions for collecting online interactive data safely. Then we take Experience Replay and Knowledge Distillation to elevate the model with new experiences while maintaining the knowledge learned before. We first evaluate our approach on a large-scale benchmark dataset Human3.6m. The experimental results show that our approach achieves a lower prediction error compared with the baselines methods. Moreover, our approach could continually learn new motion patterns without forgetting the learned knowledge. We further conduct real-scene experiments using Kinect DK. The results show that our approach can learn the human kinematic model from scratch, which effectively secures the interaction.**

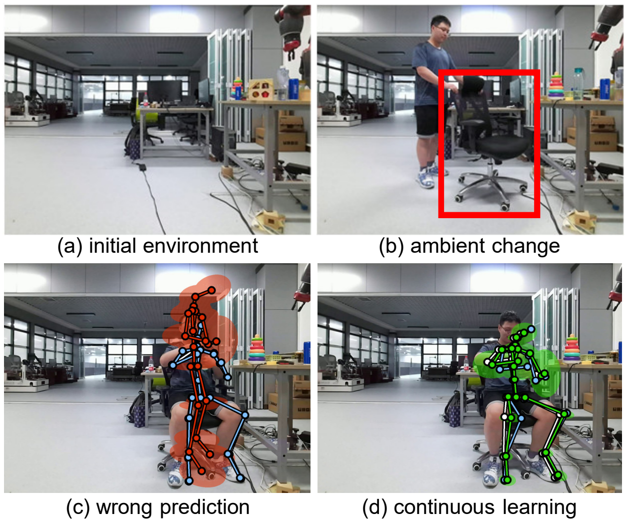

Fig. 1. In HRI scenarios, it is impossible to learn all human motion patterns in advance. Ambient changes (upper right) may derive novel patterns. The robot should avoid giving wrong predictions when it first meets an unseen motion (bottom left), and the robot should be capable of continuous learning from these new motions (bottom right).

I. INTRODUCTION

Human motion prediction (HMP) is a significant task in safe Human-Robot-Interaction (HRI) since it helps the robot plan its trajectories for avoiding collision with the collaborators.

Currently, previous HMP methods could be grouped into two lines: (1) Deterministic methods such as [1], [2], [3], [4] usually rely on Recurrent Neural Network (RNN) which ignores the randomness of human motions. (2) Probabilistic methods such as [5], [6], [7], [8] are compatible to offer predictions with uncertainty using generative model such as GAN [9], VAE [10]. However, most of these methods are trained by pre-collected data which only contains limited motion patterns. These methods assume that all possible patterns of human motion are known. This assumption may lead to catastrophic consequences in HRI scenarios as it ignores the diversity of human behaviors. For instance, a robot probably makes risky decisions when it observes unfamiliar motion patterns. Moreover, the collected training data are usually non-interactive, which does not take the real-time responses of collaborators into consideration. Therefore, these methods could not satisfy the requirements of online HRI scenarios.

In a safe HRI scenario, as shown in Fig. 1, we think a reliable HMP should have two important characteristics: **First**, the HMP algorithm should capture the uncertainties of the observed human motions which help the robot recognize the unfamiliar motion patterns. **Second**, the HMP should be capable of adapting to the new circumstance and new motion patterns. In other words, it needs to continually learn from the observed human motions without forgetting to enhance its predictions and uncertainty capturing ability.

For this purpose, in this paper, we propose a continual learning method for probabilistic human motion prediction (PHMP). (1) Our approach is uncertainty-aware, helping the robot's decision make safer, while at the same time it allows our model to be safely deployed into online scenarios using random initialization parameters. (2) Our approach owns continuously learning capability. It hardly forgets learned knowledge and performs even better on previous tasks. In addition, compared to joint training where the training cost increases over the amount of data, our approach handles

* Equal contribution

† Corresponding author

J. Xu, S. Wang, X. Chen, J. Zhang, X. Lan and N. Zheng are with the Institute of Artificial Intelligence and Robotics, Xi'an Jiaotong University, Xi'an 710049, China (e-mail: jie.xu@stu.xjtu.edu.cn; jack3shihong@gmail.com; xingyuchen1990@gmail.com; sakfzr@stu.xjtu.edu.cn; xglan@mail.xjtu.edu.cn; nnzheng@mail.xjtu.edu.cn;)

human motion data stream of infinite length with a fixed training cost.

In summary, our contributions are as follows

- We propose a continual-learning approach for probabilistic human motion prediction. The proposed approach not only makes predictions with the corresponding uncertainties for the observed motion sequences but also keeps adapting to the new human motion patterns.
- We evaluate our approach on a popular human motion prediction benchmark dataset Human3.6m [11] by designing a series of experiments. The experimental results show that our approach performs better than other baseline methods.
- We conduct real-scene experiments using Kinect DK. The results show that our approach can learn the human kinematic model from scratch, which effectively secures the interaction.

II. RELATED WORKS

A. Human Motion Prediction

A number of deep learning based approaches [1], [2], [3], [5], [6], [12] achieve remarkable performance on HMP. In detail, [1], [2], [3] take advantages of Recurrent Neural Networks (RNNs) and RNN variants (e.g. Seq2Seq [4]) while [5], [6], [12] employ generative models (e.g. GAN [9]) to generate future motions. However, lacking the ability to offer uncertainty of their predictions, the above method may cause problem in real HRI scenarios. For the Probabilistic Human Motion Prediction (PHMP) task, traditional methods [13], [14], [15] leverage Gaussian Process, Hidden Markov Model and Dynamic Forest Model approaches. However, it is tremendously hard to apply them on large-scale datasets. Recently, some deep-learning based probabilistic methods [16], [17], [18] cast light on learning multiple motion patterns by utilizing Variational Auto-Encoder (VAE) [10] and GAN [9]. For example, Butepage et al. [6] use a conditional VAE to give multiple predictions by sampling variables in a latent space. Barsoum et al. [16] design a model which offer future human poses with possibility. In contrast, we utilize Bayesian Analysis for uncertainty estimation which is more reliable and safer in HRI scenario.

B. Bayesian Neural Network

To turn a conventional neural network into a Bayesian one [19], [20], [21], the crucial step is to replace deterministic parameters with distributions over parameters. Thus, each time of forward propagation, the network samples a set of parameters from the distribution and offers predictions. By sampling over weight distribution, the network outputs distributions of future motions. These distributions further help to determine uncertainties. In [22], Loquercio et al. propose a general framework without retraining. By utilizing Monte Carlo Dropout, Kendall et al. [23] capture two kinds of uncertainty in BNN, Epistemic Uncertainty (EU) and the Heteroscedastic Aleatoric Uncertaintys (HAU). Also, Kendall et al. [23] propose a general framework to combine EU and HAU.

C. Continual Learning

To solve Catastrophic Forgetting [24], a number of methods have been developed: Orthogonal gradient based methods [25], [26], [27] take orthogonal gradient steps over previous tasks to minimize loss increasing. Even though they can continually learn with a little forgetting [27], they are generally designed for classification tasks instead of regression tasks. Regularization based methods[28], [29], [30] use various kinds of regularization, forcing the network not to forget obtained knowledge by not updating parameters important to previous tasks. HAT [29] learns a hard attention mask during the training phase to identify important neurons, whereas PackNet [30] achieves by iteratively pruning on weights via a binary mask. Although, regularization based methods shows a little forgetting, however, limited network parameters declare that performance on previous tasks inevitably deteriorates after several tasks. Memory based methods [31], [32], [33] reduce forgetting by memory subsets of previous tasks datasets or representations. Gradient Episodic Memory (GEM) [31] and its variance (A-GEM [32]) memory previous task's gradients to guide current gradient update. Differently, iCaRL [33] preserves examplar sets of each task. Then it combines all examplar sets with online data to train the network. However, the memory based methods usually assign special layers to each task, resulting in prolonged training time. Our approach does not devote any layer to each task, which eases the network training.

III. METHOD

The goal of our approach is to enable robot learning continually so as to offer safer probabilistic human motion predictions with uncertainty in HRI scenarios. As shown in Fig. 2, we proposed a replay based CL algorithm for BNN which learns continually by saving examples of previous tasks. Our approach consists of three components: Sampling Policy, Update Parameters, and Buffer Management. Sampling Policy assigns sampling weights to each example in buffer $\mathcal{P}$ and the dataset $\mathcal{X}^s$ from online dataflow. In Update Parameters, the network preserves previous knowledge by optimizing distillation loss $\mathcal{L}_{dis}$ and learns new motions by optimizing regression loss $\mathcal{L}_{reg}$. Buffer Management maintains a buffer $\mathcal{P}$ which is an approximate distribution of known human motion. Note that **TASK** in this paper refers to a new segment of motion sequence collected online.

A. Bayesian Seq2seq

Notation. Human motion data continuously crowd into our model in the form of data streams. We divide the data stream into tasks ($\mathcal{X}^s|_{s=1,2,3,...}$) by a fixed time interval T_{split}. Since we use a Bayesian Neural Network to model two kinds of uncertainty, our neural network parameter w becomes a random variable so that EU can be captured, and the output of our neural network is defined as the parameters of Gaussian distribution so that HAU can be captured. In the CL setting, we have $w \sim p(w|\mathcal{X}^s, \mathcal{P})$, where $\mathcal{P}$ is a buffer set updated after a task training. A

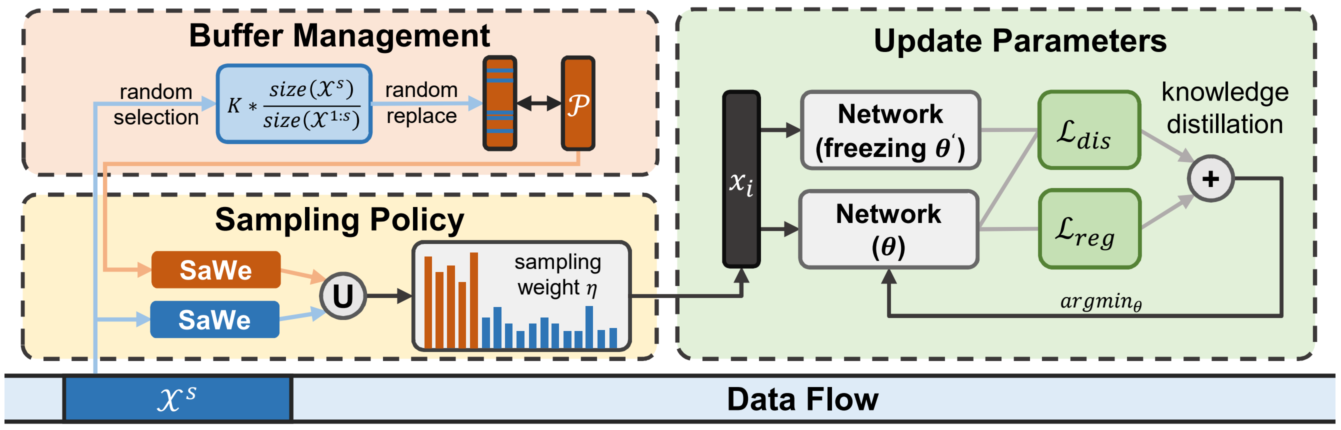

Fig. 2. Pipeline of our method. When new data $\mathcal{X}^s$ observed, buffer $\mathcal{P}$ and $\mathcal{X}^s$ are first assigned sampling weights then sampled into training set $\mathcal{X}_i$. During training, the network parameters are updated with distillation loss and regression loss. After training, the buffer $\mathcal{P}$ is updated by $\mathcal{X}^s$.

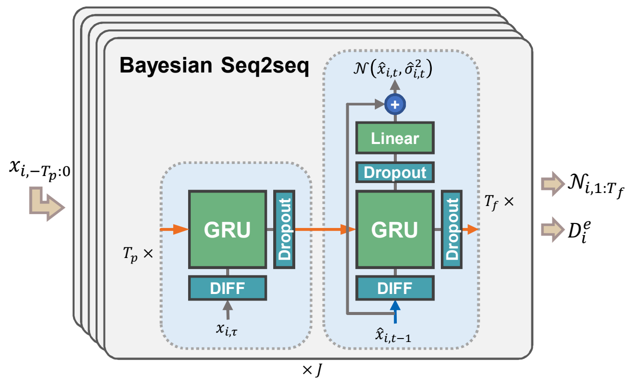

Fig. 3. Capture two kinds of uncertainties: EU and HAU. Architecture of our network basically follows Seq2Seq [4]. Monte-Carlo Dropout is abbreviated as Dropout. Outputs are obtained by a linear layer together with residual path. Inputs are first fed into difference layers to generated velocity and acceleration. By evaluating J times sampling, network predicts future motion $\mathcal{N}_{i,1:T_f}$ and EU D_i^e.

motion sequence data is denoted as $x_i = x_{i,-T_p:T_f}$ for each key point i, and the future part $x_{i,1:T_f}$ is not available for the test phase. Receiving an observed motion sequence $x_{i,-T_p:0}$, our network f outputs the Gaussian distributions $\mathcal{N}_{i,t} := \mathcal{N}(\hat{x}_{i,t}, \hat{\sigma}^2_{i,t})$ of future motion at each time step $t = 1, ..., T_f$, where

$$(\hat{x}_{i,1:T_f}, \hat{\sigma}^2_{i,1:T_f}) = \mathbb{E}_{w\sim p(w|\mathcal{X}^s,\mathcal{P})}[f^w(x_{i,-T_p:0})] \qquad (1)$$

Bayesian Implementation. We simply add Monte-Carlo Dropout both in training and testing procedures to convert a common neural network into a Bayesian one. The reason for Monte Carlo-Dropout is that, by invoking Bayesian Estimation [34], we use variational parameters $q_\theta(w)$ to approximate posterior distribution of weights $p(w|\mathcal{X}^s, \mathcal{P})$. By minimizing $KL(q_\theta(w)\|p(w|\mathcal{X}^s, \mathcal{P}))$, our regression loss function can be written as

$$\mathcal{L}_{reg} = \frac{1}{T_f}\sum_{t=1}^{T_f}\frac{\| x_{i,t} - \hat{x}_{i,t} \|^2}{2\hat{\sigma}^2_{i,t}} + \frac{1}{2}\log\hat{\sigma}^2_{i,t} \qquad (2)$$

Further, the sampling of $w \sim q_\theta(w)$ is equivalent to the sampling of MC-Dropout in our method, in another word, the row vectors of each parameter matrix follow a Bernoulli distribution. Technically, the parameters of the network $w = I^{mask}\theta$ is sampled by a MC-Dropout binary mask I^{mask}.

Uncertainties. To capture two kinds of uncertainties: HAU $\hat{\sigma}^2_{i,1:T_f}$ and EU D_i^e, as shown in Fig. 3, our network samples J sets of different parameters from weight distribution $q_\theta(w)$. By making forward propagation, we obtain J prediction sequences $(\hat{x}_{i,1:J,1:T_f}, \hat{\sigma}^2_{i,1:J,1:T_f})$. Finally, in order to estimate HAU, we use Monte Carlo integration to approximate Equation (1). Similarly, EU D_i^e can be approximated as follow

$$D_i^e \approx \frac{1}{J}\sum_j \hat{x}^2_{i,j,1:T_f} - (\frac{1}{J}\sum_j \hat{x}_{i,j,1:T_f})^2 \qquad (3)$$

Unseen Motion Detector. With D_i^e of each input obtained, Unseen Motion Detector (UMD) can identify unseen motions by a threshold D^e_{thd}. We chose 95th percentile of EU D_i^e in buffer $\mathcal{P}$ as D^e_{thd}. At testing times, the model refuses to offer prediction when D_i^e exceeds D^e_{thd}.

B. Sampling Policy

Previous CL algorithms [27], [29], [31], [33] define the task as a sequence of new classes of examples, and they need task identifiers to start the network learning new knowledge from the new class. However, examples in an online-collected sequence are a mixture of different motion patterns. On the other hand, as the length of $\mathcal{X}^s$ is agnostic, simply concatenating $\mathcal{P}$ with $\mathcal{X}^s$ together to train the model [31], [33] may cause the network only learns from $\mathcal{X}^s$, ignoring previous knowledge preserved in $\mathcal{P}$ since tremendous online data crowd in at an instance. So, learning from mixed online data and preserving previous knowledge when learning from $\mathcal{X}^s$ of arbitrary length are two crucial problems that need to be addressed.

The reason why previous methods cannot learn from the mixed sequence is that they need task identifiers to

Algorithm 1 Sampling Weights (SaWe)

```
Input: $\mathcal{D}$ // dataset
Input: $\theta$ // network parameters
Output: $\eta$ // sampling weights of $\mathcal{D}$
 1: for all $x_i \in \mathcal{D}$ do
 2:     for $j = 1, ..., J$ do
 3:         $w_j \leftarrow I_j^{mask}\theta$ // MC-Dropout sampling
 4:         $(\hat{x}_{i,j,1:T_f}, \cdot) \leftarrow f^{w_j}(x_{i,-T_p:0})$
 5:     end for
 6:     Calculate $D_i^e$ using Equation (3)
 7: end for
 8: for all $x_i \in \mathcal{D}$ do
 9:     $\widetilde{\eta}_i \leftarrow \frac{D_i^e - \min_i D_i^e}{\max_i D_i^e - \min_i D_i^e} + 1$
10:     $\eta_i \leftarrow \frac{\widetilde{\eta}_i}{\sum_i \widetilde{\eta}_i}$
11: end for
12: $\eta = \{\eta_1, \eta_2, \eta_3, ...\}$
13: return $\eta$
```

Algorithm 2 CL Algorithm for PHMP

```
Input: $\theta$ // network parameters
Input: $\mathcal{P}$ // buffer
Input: $\mathcal{X}^s$ // data set for task $s$
Output: $\theta$
 1: $\mathcal{D} \leftarrow P \bigcup \mathcal{X}^s$
 2: $\theta^{'} \leftarrow \theta$ // freezing parameters
 3: for $i_{step} = 1, ..., N$ do // $N$ iterations for training
 4:     // weighted sampling
 5:     if $i_{step}$ $mod$ $M == 1$ then // update $\eta$ every $M$-it
 6:         $\eta \leftarrow \frac{1}{2}\text{SaWe}(\mathcal{P}, \theta) \bigcup \frac{1}{2}\text{SaWe}(\mathcal{X}^s, \theta)$
 7:     end if
 8:     $x_i \leftarrow$ sample from $\mathcal{D}$ with weight $\eta$
 9:     // sample $w, w^{'}$ using same dropout mask $I^{mask}$
10:     $w, w^{'} \leftarrow I^{mask}\theta, I^{mask}\theta^{'}$
11:     $\mathcal{N}_{i,1:T_f} \leftarrow f^{w}(x_{i,-T_p:0})$
12:     $\mathcal{N}^{'}_{i,1:T_f} \leftarrow f^{w^{'}}(x_{i,-T_p:0})$
13:     train the network by minimizing Equation (5)
14: end for
15: update buffer $\mathcal{P}$ using a subset of $\mathcal{X}^s$
16: return $\theta$
```

imply the network learning new knowledge. EU describes network's epistemic level on input. Thus, by evaluating EU, examples in $\mathcal{X}^s \bigcup \mathcal{P}$ can be given different sampling weights. Unlearned motions of high EU are then fully learned.

Details of the proposed sampling algorithm are shown in Algorithm 1. To learn $\mathcal{X}^s$ of arbitrary length, we let the network mainly learns high EU examples. Specifically, we first assign sampling weights for all examples according to EU. Examples with high EU are given higher possibilities to be chosen as training data. Next, instead of directly sampling according to EU, we normalize possibilities of all examples to the size of $\mathcal{P}$ and $\mathcal{X}^s$. Note that we add 1 after min-max normalization. This is because we not only make the network learns high EU examples, but also expect the network learns the distribution of examples, in case that near-zero EU examples that play important role in maintaining distribution are neglected.

On one hand, sampling with balanced possibilities helps the model not to only choose training examples from $\mathcal{X}^s$ when massive online data crowd in. As a result, the model is free from only learning $\mathcal{X}^s$ and ignoring knowledge in $\mathcal{P}$. On the other hand, it helps the network learns new knowledge in $\mathcal{X}^s$ quickly in a limited time, since high EU examples are frequently chosen and well learned.

C. Update Parameters

We develop our Update Parameters part, shown in Algorithm 2. Knowledge Distillation [35] helps the network preserving previous knowledge when training on current task. The key idea is forcing the network with trained parameter θ to keep the output distribution $\mathcal{N}(\hat{x}_{i,t}, \hat{\sigma}^2_{i,t})$ the same as the frozen parameter $\theta^{'}$ output $\mathcal{N}(\hat{x}^{'}_{i,t}, \hat{\sigma^{'}}^2_{i,t})$ for all $x_i \in \mathcal{P}$. To achieve that, we minimize $KL(\mathcal{N}^{\theta^{'}}_{i,t} || \mathcal{N}^{\theta}_{i,t})$. Note that, for a same input, the dropout masks I^{mask} of different sub-network are same. In practice, since output for each timestamp is a Gaussian distribution, we sum up KL divergences over all time steps, and the distillation term is

$$\mathcal{L}_{dis} = \sum_{t=1}^{T_f} \log \frac{\hat{\sigma^{'}}_{i,t}}{\hat{\sigma}_{i,t}} + \frac{\hat{\sigma}^2_{i,t} + (\hat{x}_{i,t} - \hat{x}^{'}_{i,t})^2}{2\hat{\sigma^{'}}^2_{i,t}} \tag{4}$$

Training with distillation loss item enables our model to retain performance on the previous tasks.

Overall, our loss function combines distillation loss for CL and regression loss for PHMP together. With a hyper-parameter λ, we balance forgetfulness and plasticity:

$$\mathcal{L} = \mathcal{L}_{reg} + \delta_{x_i \in \mathcal{P}} \lambda \mathcal{L}_{dis} \tag{5}$$

D. Buffer Management

After the training phase, we update our buffer $\mathcal{P}$. As neural network learns knowledge preserved in buffer, we update $\mathcal{P}$ with current $\mathcal{X}^s$ to absorb the knowledge of $\mathcal{X}^s$ into $\mathcal{P}$. To achieve that, we random select $round(K * \frac{size(\mathcal{X}^s)}{size(\mathcal{X}^{1:s})})$ examples from $\mathcal{X}^s$, where K is the buffer size. These examples of $\mathcal{X}^s$ preserve distribution of $\mathcal{X}^s$ as we randomly choose them. Then we replace randomly chosen examples in $\mathcal{P}$ with these examples since the buffer is at a fixed size of K. Finally, the new buffer contains examples from both $\mathcal{X}^s$ and the previous buffer.

IV. Experiments

In this section, we analyze the advantages of our algorithm compared with other CL algorithms on the large public dataset HUMAN3.6M. Our experiments focus on the following points, (1) The prediction error and forgetting of the model learned by our method, compared with related methods; (2) Detailed analysis of our approach, including the forgetting during the training phase, the effect of memory size, safety, etc.; (3) Application of our method in a real HRI scenario.

TABLE I

QUANTITATIVE RESULTS OF DIFFERENT CL ALGORITHM ON HUMAN3.6M [11]. ALL CL METHODS ARE TRAINED UNDER 5 TASK ORDERS RANDOMLY GENERATED (THE SAME SEEDS ARE USED FOR EACH METHOD). THE AVERAGE IS SHOWN IN THE TABLE. BEST RESULTS ARE IN BOLD, AND SECOND-BEST RESULTS ARE MARKED WITH A UNDERLINE.

Method	Rej. (%)	MPJPE (↓)						
		Average	BWT	200ms	400ms	600ms	800ms	1000ms
Zerovel	-	0.080	-	0.041	0.072	0.095	0.112	0.126
Finetune	-	0.083	0.015	0.033	0.072	0.101	0.123	0.140
iCaRL [33]	-	0.071	0.000	0.031	0.060	0.083	0.102	0.117
A-GEM [32]	-	0.089	0.011	0.033	0.075	0.109	0.134	0.154
GPM [27]	-	0.073	0.001	0.032	0.062	0.087	0.107	0.124
Ours	-	<u>0.061</u>	**-0.006**	<u>0.024</u>	<u>0.051</u>	<u>0.074</u>	<u>0.093</u>	<u>0.110</u>
Ours+UMD	7.02%	**0.057**	**-0.006**	**0.022**	**0.047**	**0.068**	**0.086**	**0.102**
Joint	-	0.057	-	0.022	0.047	0.069	0.086	0.102
Joint+UMD	6.68%	0.053	-	0.020	0.044	0.064	0.081	0.096

A. Experimental Settings

Baselines. The CL baselines we selected are: (1) **Zerovel** takes the last input of past human motion as the prediction. (2) **Finetune** simply finetunes parameters on the new task. (3) **Joint** represents joint training all tasks. For CL algorithms, we select (4) **iCaRL** and (5) **A-GEM** on behalf of replay-based methods and (6) **GPM** for orthogonal-based methods. The same network architecture is applied to all of the above methods.

Dataset. Human3.6m[11] is one of the largest 3D human motion datasets which contains 15 kinds of actions (walking, eating, etc.), and we treat them as 15 tasks. We choose subsets 1, 6, 7, 8, 9 for the training set, subset 11 for the validation set, and subset 5 for the testing set, which contain 172K, 26K, and 46K samples respectively.

Performance Metrics. We use Mean Per Joint Prediction Error (MPJPE) [36], Negative Log-Likelihood (NLL), and Backward Transfer (BWT) [31] to evaluate our method. MPJPE measures the mean Euclidean distance between prediction pose and ground truth. After learning task i, MPJPE on task j is denoted as $R_{i,j}$. NLL measures accuracy of predictive distribution which can be obtained by Equation 2. BWT measures model's forgetting on previous tasks, and after learning S tasks it can be calculated as BWT $= \frac{1}{S-1}\sum_{s=0}^{S-1} R_{S,s} - R_{s,s}$. For these metrics, the lower value represents the better performance.

Implementation Details. We set the memory size to 500 for all replay-based methods. For our method, λ is set to 2. The learning rate is set to $3e-4$ and decreases by 0.5% after each epoch. AdamW [37] optimizer is adopted, and weight decay is set to 0.01. We carry out all experiments on Pytorch 1.9 with a single NVIDIA RTX3090 GPU.

B. Accuracy and Backward Transfer

As shown in Table I, we report MPJPEs (prediction error) within 1000ms and their average, then BWT after learning 15 tasks. "Rej. (%)" are the percentage of the predictions rejected by UMD, and "+" stands for combination. Our method without UMD reports the smallest prediction error (0.061) compared to other methods (iCaRL, A-GEM, and GPM). What's more, our method achieves the smallest BWT(-0.006) among all CL algorithms. The negative BWT indicates learning subsequent tasks helps previous tasks. After the UMD rejecting 5% (7.02% in table) predictions, our method with UMD reports a better prediction accuracy. Finetune shows the greatest BWT(0.015), suggesting that Catastrophic Forgetting happens, which offers poor prediction. There are gaps between all CL algorithms and joint training, the upper bound of CL setting. However, our method has the smallest performance gap (0.057 v.s. 0.061). In summary, our results have the highest prediction accuracy and hardly forget previous learned knowledge.

C. Detailed Analysis

Forgetting. To evaluate model forgetting, we conduct experiments shown in Fig. 5. We monitor model performance on the first task during the training of different algorithms. From the left, GPM and iCaRL performances degenerate, representing the model forgetting previous knowledge. However, our performance is increasing steadily. From the right, the predictive distribution becomes more precise. This suggests the knowledge learned from the subsequent tasks can help the model better understand the previous task in our method.

Memory size. Also, Fig. 5 shows the effect of memory size in our method. It reveals that 1) Our method's prediction error is stable to different memory size (left). 2) The larger memory size helps the model gain lower and more stable NLL (right), especially on tasks 9 and 11. In conclusion, the memory size has a gentle effect on prediction accuracy but the larger memory size has a prominent ability on anti-forgetting for NLL.

Ablation study. We also investigate the effectiveness of our components, i.e., Weighted Sampling(WS), Distillation Term (DT), and UMD, as shown in Table II. Results with UMD are shown in parentheses. It can be seen that: Both WS and DT help the model in keeping previous knowledge and obtaining more accurate predictions. All of the components are effective and their combination achieves the best.

Safety. We further investigate rejection rate and prediction error trends during the network learning process, as shown

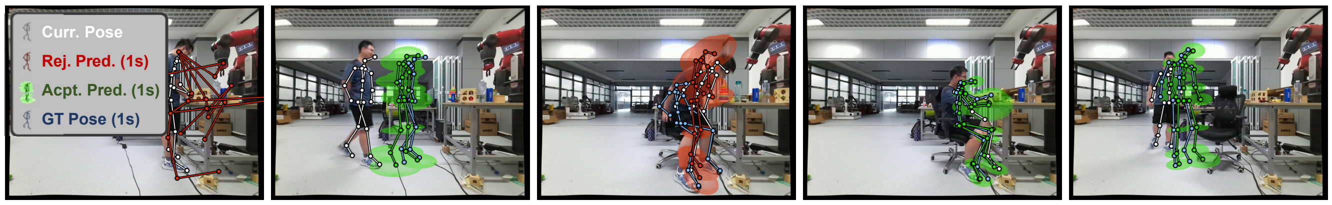

Fig. 4. Real-scene Experiments. The robot learns continually from online collected data at 10 minutes interval.

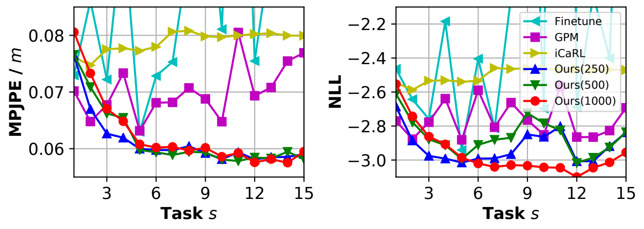

Fig. 5. Trends of MPJPE and NLL on the first task as learning continues of different CL algorithms and different memory sizes for our method.

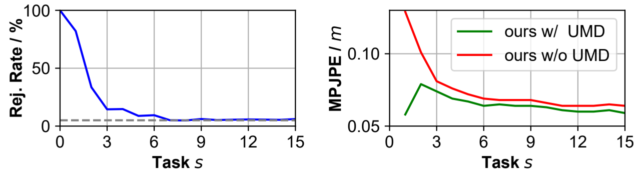

Fig. 6. Trends of rejection rate and MPJPE as learning continues. Dotted line (left) represents 5% rejection rate.

in Fig. 6. It can be seen that: 1) With the instruction of EU, UMD significantly decreases prediction error on accepting part (right), contributing to safer HRI. It helps most at the beginning and keeps effective in the whole learning process. 2) Rejection rate drops as the network continually learns (left), indicating the network is getting familiar with more motion patterns. UMD helps the model not offer highly EU predictions, enabling us to deploy the model without pre-train in a real scenario.

D. Real Scene Experiments

A real scene experiment is shown in Fig. 4. In the scenario, a man is trying to approach the desk and interact with a robot with gadgets on the desk. The robot captures human poses continually and predicts future motion based on captured motions, acquired by the camera, to help itself cooperate with human.

TABLE II
ABLATION STUDY ON DIFFERENT COMPONENTS: WEIGHTED SAMPLING (WS), DISTILLATION TERM (DT) AND UNSEEN MOTION DETECTOR (UMD). RESULTS WITH UMD ARE IN PARENTHESES.

Method	Rej. (%)	MPJPE (↓)	BWT (↓)
ours	5.28	0.067 (0.064)	0.003
ours + WS	7.44	0.062 (0.058)	-0.002
ours + DT	6.53	0.063 (0.059)	-0.003
ours + WS + DT	7.02	**0.061 (0.057)**	**-0.006**

Visualization Details. In Fig. 4, receiving several frames of captured human poses (white), accepted prediction for the next 1 second is shown in green and the rejected in red. Ground truths are shown in blue. Uncertainty area with 95% confidence interval of the predictive distribution $\mathcal{N}_{i,t=1000ms}$ is shown in ellipse for each key point i.

Experimental Results. At the beginning, the robot predicts human motion with randomly initialized parameters before training. Thanks to our UMD, the robot rejects horrible predictions (left). In the first ten minutes, a man approaches the desk and interacts with gadgets on the desk. After learning the data collected in the first ten minutes, the robot successfully predicts a man walking (second from left). However, when ambient change happens where a chair is placed in front of the desk, the robot fails to predict new action, e.g. sitting (middle). Since the robot never learns motions like sitting before, the robot is confused about motion whose knees are slightly bent and mistakenly predicts the man keeps walking. Consequently, the robot rejects the prediction. After learning data collected in the next ten minutes, the robot successfully predicts the man sitting and cooperating (second from the right). Finally, the robot keeps previous knowledge and still succeeds in predicting the man walking toward the desk (right). In summary, in the real scenario, 1) the awareness of uncertainty helps the robot interact with human safely even with a scratch model. 2) The CL ability enables the robot continuously to learn new human motion patterns without forgetting.

V. CONCLUSIONS

In this paper, we propose a novel replay-based CL method for PHMP. The proposed approach captures multiple uncertainties to ensure safety when interacting with human collaborators, while continuously learning various motion patterns. The experimental results on the benchmark dataset show that our approach outperforms other CL methods with both higher prediction accuracy and lower forgetting. In a real-scene HRI scenario, our approach demonstrates its ability to avoid giving the wrong prediction and learn new motion patterns continuously without forgetting.

ACKNOWLEDGEMENT

This work was supported in part by National Key R&D Program of China under grant No. 2021ZD0112700, NSFC under grant No.62125305, No.62088102, No.61973246, and the program of the Ministry of Education.

References

[1] K. Fragkiadaki, S. Levine, P. Felsen, and J. Malik, "Recurrent network models for human dynamics," 2015.

[2] A. Jain, A. R. Zamir, S. Savarese, and A. Saxena, "Structural-rnn: Deep learning on spatio-temporal graphs," 2016.

[3] J. Martinez, M. J. Black, and J. Romero, "On human motion prediction using recurrent neural networks," 2017.

[4] I. Sutskever, O. Vinyals, and Q. V. Le, "Sequence to sequence learning with neural networks," 2014.

[5] G. W. Taylor, G. E. Hinton, and S. Roweis, "Modeling human motion using binary latent variables," in *Advances in Neural Information Processing Systems*, B. Schölkopf, J. Platt, and T. Hoffman, Eds., vol. 19. MIT Press, 2007.

[6] J. Bütepage, M. Black, D. Kragic, and H. Kjellström, "Deep representation learning for human motion prediction and classification," 2017.

[7] H. Sidenbladh, M. J. Black, and L. Sigal, "Implicit probabilistic models of human motion for synthesis and tracking," in *European Conf. on Computer Vision*, vol. 1, 2002, pp. 784–800.

[8] J. Xu, X. Chen, X. Lan, and N. Zheng, "Probabilistic human motion prediction via a bayesian neural network," 2021.

[9] I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial networks," 2014.

[10] D. P. Kingma and M. Welling, "Auto-encoding variational bayes," 2014.

[11] C. Ionescu, D. Papava, V. Olaru, and C. Sminchisescu, "Human3.6m: Large scale datasets and predictive methods for 3d human sensing in natural environments," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 36, no. 7, pp. 1325–1339, 2014.

[12] Y. Wang, L.-Y. Gui, X. Liang, and J. M. F. Moura, "Adversarial geometry-aware human motion prediction," in *Proceedings of (ECCV) European Conference on Computer Vision*. Springer, September 2018.

[13] J. Wang, A. Hertzmann, and D. J. Fleet, "Gaussian process dynamical models," in *Advances in Neural Information Processing Systems*, Y. Weiss, B. Schölkopf, and J. Platt, Eds., vol. 18. MIT Press, 2006.

[14] D. Kulic, D. Lee, C. Ott, and Y. Nakamura, "Incremental learning of full body motion primitives for humanoid robots," in *Humanoids 2008 - 8th IEEE-RAS International Conference on Humanoid Robots*, 2008, pp. 326–332.

[15] A. M. Lehrmann, P. V. Gehler, and S. Nowozin, "Efficient nonlinear markov models for human motion," in *2014 IEEE Conference on Computer Vision and Pattern Recognition*, 2014, pp. 1314–1321.

[16] E. Barsoum, J. Kender, and Z. Liu, "Hp-gan: Probabilistic 3d human motion prediction via gan," 2017.

[17] H. Sidenbladh, M. J. Black, and L. Sigal, "Implicit probabilistic models of human motion for synthesis and tracking," in *Proceedings of the 7th European Conference on Computer Vision-Part I*, ser. ECCV '02. Berlin, Heidelberg: Springer-Verlag, 2002.

[18] J. Bütepage, H. Kjellström, and D. Kragic, "Anticipating many futures: Online human motion prediction and synthesis for human-robot collaboration," 2017.

[19] C. Blundell, J. Cornebise, K. Kavukcuoglu, and D. Wierstra, "Weight uncertainty in neural networks," 2015.

[20] B. J. Frey and G. E. Hinton, "Variational learning in nonlinear gaussian belief networks," *Neural Comput.*, vol. 11, no. 1, p. 193–213, Jan. 1999.

[21] T. D. Bui, J. M. Hernández-Lobato, Y. Li, D. Hernández-Lobato, and R. E. Turner, "Training deep gaussian processes using stochastic expectation propagation and probabilistic backpropagation," 2015.

[22] A. Loquercio, M. Segu, and D. Scaramuzza, "A general framework for uncertainty estimation in deep learning," *IEEE Robotics and Automation Letters*, vol. 5, no. 2, p. 3153–3160, Apr 2020.

[23] A. Kendall and Y. Gal, "What uncertainties do we need in bayesian deep learning for computer vision?" 2017.

[24] M. McCloskey and N. Cohen, "Catastrophic interference in connectionist networks: The sequential learning problem," *Psychology of Learning and Motivation*, vol. 24, pp. 109–165, 1989.

[25] G. Zeng, Y. Chen, B. Cui, and S. Yu, "Continual learning of context-dependent processing in neural networks," *Nature Machine Intelligence*, vol. 1, no. 8, p. 364–372, Aug 2019.

[26] M. Farajtabar, N. Azizan, A. Mott, and A. Li, "Orthogonal gradient descent for continual learning," 2019.

[27] G. Saha, I. Garg, and K. Roy, "Gradient projection memory for continual learning," 2021.

[28] J. Kirkpatrick, R. Pascanu, N. Rabinowitz, J. Veness, G. Desjardins, A. A. Rusu, K. Milan, J. Quan, T. Ramalho, A. Grabska-Barwinska, D. Hassabis, C. Clopath, D. Kumaran, and R. Hadsell, "Overcoming catastrophic forgetting in neural networks," 2017.

[29] J. Serrà, D. Surís, M. Miron, and A. Karatzoglou, "Overcoming catastrophic forgetting with hard attention to the task," 2018.

[30] A. Mallya and S. Lazebnik, "Packnet: Adding multiple tasks to a single network by iterative pruning," 2018.

[31] D. Lopez-Paz and M. Ranzato, "Gradient episodic memory for continual learning," 2017.

[32] A. Chaudhry, M. Ranzato, M. Rohrbach, and M. Elhoseiny, "Efficient lifelong learning with a-gem," 2019.

[33] S.-A. Rebuffi, A. Kolesnikov, G. Sperl, and C. H. Lampert, "icarl: Incremental classifier and representation learning," 2017.

[34] Y. Gal, "Uncertainty in deep learning," Ph.D. dissertation, University of Cambridge, 2016.

[35] G. Hinton, O. Vinyals, and J. Dean, "Distilling the knowledge in a neural network," 2015.

[36] A. Rasouli, "Deep learning for vision-based prediction: A survey," 2020.

[37] I. Loshchilov and F. Hutter, "Fixing weight decay regularization in adam," 2018.