GPFS: A Graph-based Human Pose Forecasting System for Smart Home with Online Learning

XIN LI, Department of Computer Science and Engineering, Lehigh University
DAWEI LI, Samsung Research America

Forecasting human poses given a sequence of historical pose frames has several important applications, especially in the domain of smart home safety. Recently, computer vision-based human pose forecasting has made a breakthrough using deep learning technology. However, to implement a practical system deployed on an IoT edge environment, there are still two issues to be addressed. First, existing methods on pose forecasting fail to model the coherent structural information of connected human joints and thus cannot achieve satisfactory prediction accuracy, especially for long-term predictions. Second, a general and static pre-trained prediction model may not perform well in the deployment environment due to the visual domain shift problem. In this article, we propose a hybrid cloud-edge system called GPFS to solve those issues. Specifically, we first introduce a novel graph convolutional neural network (GCN)-based sequence-to-sequence learning method, which enhances the sequence encoder by using a graph to represent both the spatial and temporal connections of the human joints in the input frames. The GCN improves the forecasting accuracy by capturing the motion pattern of each joint as well as the correlations among different human joints over time. Second, to address the domain shift issue and protect data privacy, we extend the system to perform online learning on the IoT edge to adapt the cloud trained general model with online collected on-site domain data. Extensive evaluation on Human 3.6M and Penn Action datasets demonstrates the superiority of our proposed system.

CCS Concepts: • **Computer systems organization** → **Embedded systems**; • **Computing methodologies** → *Computer vision*;

Additional Key Words and Phrases: Deep learning, IoT, pose forecasting, graph neural networks, online learning

ACM Reference format:
Xin Li and Dawei Li. 2021. GPFS: A Graph-based Human Pose Forecasting System for Smart Home with Online Learning. *ACM Trans. Sen. Netw.* 17, 3, Article 34 (June 2021), 19 pages.
https://doi.org/10.1145/3460199

1 INTRODUCTION

Human pose forecasting is the computer vision task that, given a sequence of historically observed human poses (in forms of human joints [4]), predicts the human pose changes in the near future (e.g., the next second). Human pose forecasting has several important applications in a smart home

Xin Li and Dawei Li contributed equally to this research.

Authors' addresses: X. Li, Department of Computer Science and Engineering, Lehigh University (Building C 113 Research Drive, Lehigh Mountaintop Campus, Bethlehem, PA 18015, United States); email: xil915@lehigh.edu; D. Li (corresponding author), Samsung Research America (665 Clyde Ave, Mountain View, CA 94043, United States); email: lidaweitz@gmail.com.

1550-4859/2021/06-ART34 $15.00
https://doi.org/10.1145/3460199

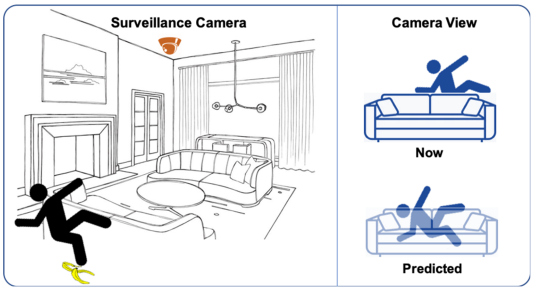

Fig. 1. An example of pose forecasting a home safety monitoring system: The person is falling down out of the camera's field of view. A semantic action prediction model can predict the falling action, but the severity of the falling could only be inferred with pose forecasting (e.g., whether the head touched the ground).

environment. For example, it can monitor the safety of a person even when the monitored person moves out of the camera's field of view (see Figure 1). As another example, as **virtual reality (VR)** is getting more and more popular, it can help the VR application to automatically adjust the content based on the predicted human poses to avoid certain dangers. Pose forecasting can also be used in the process of rehabilitation to help doctors monitor the recovery level by comparing the predicted and the actual captured poses.

To solve the human pose forecasting problem, researchers have recently formulated it as a **sequence-to-sequence (seq2seq)** learning problem and use deep neural networks to learn a prediction model [5, 23, 30]. However, those methods have focused on improving either the general seq2seq learning performance [5, 23] or the loss functions [30], but failed to incorporate and model the inherent spatial structure among human joints. For example, the joint of the elbow is always connected with the shoulder joint and the wrist joint. Simply ignoring this critical and inherent constraint makes the forecasting task unnecessarily hard, and the solutions are relatively unstable, especially when making long-term predictions. However, it is not trivial to embed such information into the deep neural network learning as it requires proper input data representation and the matched neural network operations.

Deploying a deep learning model that is trained using a generic dataset may not have the optimal performance at the deployment environment [9, 16]. The reasons are twofold in the human pose forecasting application: visual domain shift and person bias. First, the input pose sequence relies on the human pose extraction algorithm running real time on a sequence of input video frames (e.g., using OpenPose ([2])), which could be affected by the visual domain at the deployment environment. Second, many safety-related smart-home applications (e.g., monitoring an elderly movement), once deployed to the edge, only make predictions on a limited number of persons. Thus, a model specialized for this limited set of persons would be useful for improving the model's accuracy. On the other hand, it is relatively easy to obtain an online training dataset for pose forecasting that is specific to domain and person in the deployed environment since no ground truth needs to be manually labeled.

In this article, we propose GPFS, a graph-based human pose forecasting system with online learning capability. GPFS addresses the above-discussed issues in designing a practical system with two major contributions. First, we introduce a graph neural network-based solution to address issues of existing human pose forecasting methods. In GPFS, we represent the input sequence with an undirected graph and use the edge to indicate whether two joints are directly connected in the human skeleton. Such graph representation clearly incorporates the pre-knowledge of the inherent spatial structure among different joints. To process this graph input, we propose to enhance

the sequence encoder with a **graph convolutional (GC)** component. The basic building block in the GC component is a stack of convolutional layers that captures the temporal context for an individual human joint and a graph operation layer that captures the correlation among different joints. This GC component can be concatenated with any existing seq2seq architecture (e.g., an RNN-based encoder-decoder) to make predictions. Second, to enable performance optimization at deployed domains, GPFS collects onsite domain-specific samples and automatically builds an online training dataset that is used to fine-tune and adapt the model with domain and person awareness. In addition, considering that uploading on-site personal data to the cloud to enhance the model may raise privacy issues, we perform this online model adaption directly on the IoT edge, making GPFS a hybrid cloud-edge system.

We run extensive experiments to evaluate the proposed pose forecasting algorithm and the GPFS system. To show the effectiveness of how our algorithm improves over the existing solutions, experiments have been conducted on two benchmark datasets, Human 3.6M [12] and Penn Action [42]. The experiment results show that our method has a fairly stable performance for both short-term (pose changes in the future 400 ms) and long-term prediction (pose changes in the future 1,000 ms) and significantly improves over the state of the art. We further demonstrate that the online learning module can significantly boost the pose forecasting accuracy. Lastly, we build a prototype system and show that the system is suitable for smart home deployment considering the edge computing resource constraints.

To summarize, the contributions of this article are as follows:

- We propose a novel graph neural network-based model to address issues of existing human pose forecasting methods.
- An online adaption framework is proposed to enable performance optimization at deployed domains.
- Extensive experiments on two well-known benchmark datasets prove that our proposed solution performs much better than the state-of-the-art schemes, and the online learning module can significantly boost the pose forecasting accuracy in new environments.
- A prototype system is implemented, and it shows that the system is suitable for smart home deployment.

The rest of this article is organized as follows. In Section 2, we briefly discuss related work, which is followed by the system overview in Section 3. In Section 4, we describe our proposed human pose forecasting algorithm and the online model adaptation method in Section 4. We describe the implementation details and report our experimental results in Section 5. Finally, we conclude this article in Section 6.

2 RELATED WORK

Seq2Seq human pose forecasting. Martinez et al. [23] introduced a seq2seq **recurrent neural network (RNN)** model with a residual connection between the input and the output of each RNN cell. Such a model can predict future motion for multiple actions, while previous works only focused on building action-specific models. Then, in [5], Chiu et al. propose a new action-agnostic method for short- and long-term human pose forecasting by modeling the hierarchical and multi-scale characteristics of human dynamics. The major contribution of this work [5] is that they model human dynamics in visual scenes by encoding the temporal dependencies of different time-scales in a hierarchical interconnected sequence of RNN cells. Pavllo et al. [30] propose the QuaterNet to represents rotations of each joint with quaternions, and its loss function performs forward kinematics on a skeleton to penalize absolute position errors instead of angle errors. With such an approach, they achieved better experimental results than existing works.

Although all of these seq2seq schemes perform better than previous solutions, they ignore the impacts of connected joints, which results in them suffering from missing structural information of human bodies. We argue that considering structural information can achieve a better prediction result. The reason is that the motion of a joint is never independent, and it is impacted by other related joints. Thus, a model will make a wrong prediction, especially when making a long-term prediction, if it only considers one particular joint while ignoring the motion of other associated joints.

Coarse-grained action prediction. Recent years have witnessed extensive research of human action prediction [1, 19, 26, 33], which predicts a semantic action class label given a sequence of incomplete initial video frames. However, such coarse-grained prediction is not adequate for many application scenarios. For example, in an edge home safety monitoring system (as shown in Figure 1), a coarse-grained action prediction model can predict a human is falling (i.e., a semantic label) but cannot differentiate whether it is severe (e.g., the head touches the ground first) or not. In those scenarios, pose forecasting comes in handy as it can give a precise description of the spatial position relations among different human body parts in the near future. In addition, coarse-grained semantic prediction cannot be used in applications such as human-robot interaction [15] or entertainment video generation [41] while they purely rely on accurate pose forecasting.

Future frame prediction. Quite a few works [20–22, 24, 27, 29, 34, 36, 39] have been done to predict future frames. These solutions take the previous one or multiple frames as input and generate frames in the future using some generative models, e.g., **generative adversarial network (GAN)**. Their purposes are merely focusing on predicting realistic pixel values in future frames. If we want to apply such solutions in human pose forecasting, additional object detection and tracking models are required to understand the generated future frames. Therefore, this pipeline cannot meet the efficiency requirement in some application scenarios.

GCNs. Graph Convolutional Networks (GCNs) have been used to analyze paper citation data [14] and to predict traffic congestion [6]. Recently, researchers have tried to employ GCNs in the task of skeleton-based human action recognition [35, 40] and achieved better performance than the existing solutions. Both human action recognition models consist of many graph convolution layers (nine layers) to learn a deep representation of the data. In this article, considering that the input coordinates are useful for the forecasting task, we use fewer layers and add some Batch Normalization layers to make the model easy to train. Compared to those previous works, we also add more BN layers to help the training phase be more stable (especially the one after the input).

Deep online learning for computer vision. Deep learning models for computer vision tasks may suffer from the domain variance problem where the data distributions in the training set and the deployment environment are different. The problem is particularly severe for computer vision applications at IoT or edge computing since each deployment environment may have unique data distributions. Deep online learning solutions have been proposed to address this problem. [16] introduced a system that engages edge users to collaboratively collect domain-specific images and trains a domain-specific model for object recognition. [17] proposed a near-real-time incremental learning method for object detection in the absence of training data from previously learning object classes. An unsupervised online method for face identity learning from video streams was introduced in [31], which takes advantage of the temporal coherence of visual data. Our work has been the first that tackles the online learning problem for human pose-related applications.

3 SYSTEM OVERVIEW

In this section, we present the overview and the working pipeline of the proposed GPFS, a hybrid cloud-edge system. On the cloud, a general model is trained with a large amount of data collected with numerous different persons. While on the edge, the model is adapted with

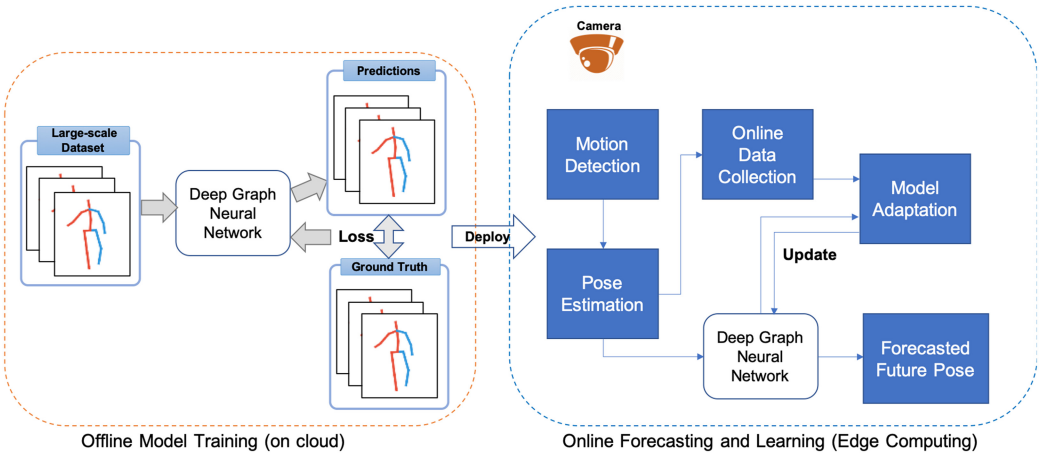

Fig. 2. GPFS system overview. The left part is the offline model training pipeline that trains our proposed deep graph neural network using pre-collected domain agnostic data. The right part is the online pose forecasting and model adaptation pipeline after the pre-trained deep learning model is deployed to the edge.

limited domain-specific data for optimized accuracy and privacy protection. The overall system is illustrated in Figure 2, and it includes the following three major modules:

- **Offline Training:** Initially, a dataset is collected to capture (1) as many different person identities, (2) as many different visual variances, and (3) as many different activities as possible. The dataset is split into training and validation datasets to develop the deep learning models. We propose a graph neural network-based modeling method, and the model is trained in an end-to-end manner so that no hand-crafted feature is used and no post-processing is required. The offline training module is generally deployed on the cloud with advanced training processors like GPUs and TPUs.
- **Online Forecasting:** With the pre-trained deep learning model, the system is deployed at a smart home environment with a camera (or possibly multiple cameras) connected to an edge server where the streamed videos are processed. The system can be triggered by actively monitoring humans with person/face detection or passively triggered by certain sensors like motion sensors. The pose estimation sub-module extracts the human poses (e.g., using OpenPose [2]) in real time and once enough frames have been buffered, the pose forecasting sub-module gets started to work. The system will alert if it predicts dangerous future poses or identifies obvious pose differences between predicted poses and the actually captured human poses.
- **Online Data Collection and Model Adaptation:** In the deployment environment, the system actively collects online data, which could reflect the data distribution at the particular visual domain and users. With the collected online data, the system adapts the pre-trained model by model fine-tuning. We elaborate on the design of this module in Section 4.3.

4 ALGORITHM DESIGN

In this section, we present the core algorithms of GPFS. We start by presenting the problem formulation of human pose forecasting. Following that, the proposed deep learning method is introduced that is a graph neural network-based sequence-to-sequence learning solution. The online data collection and learning are detailed at the end.

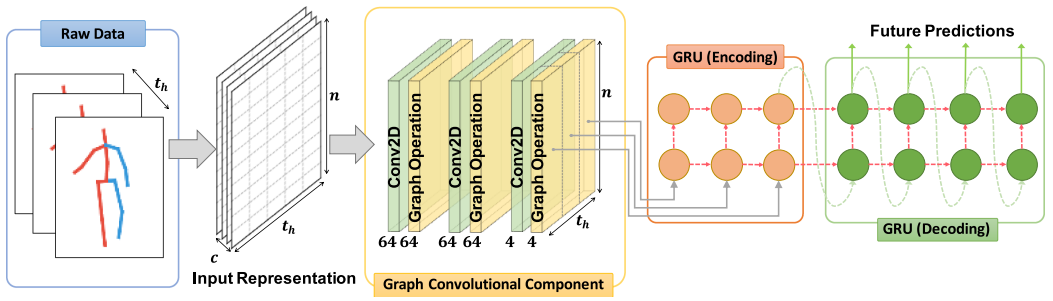

Fig. 3. The architecture of the proposed deep graph neural network architecture for pose forecasting. The raw pose data is converted to a graph input representation and is then processed by the graph convolutional component. The output is fed into a GRU-based encoder-decoder network.

4.1 Problem Formulation of Pose Forecasting

Before introducing our proposed scheme, we first present the problem formulation for the pose forecasting problem. Specifically, given the inputs X, which are position histories over t_h time steps of all human joints from a human skeleton, the system is trying to predict the future positions Y over t_f time steps for each joint:

$$X = [p^{(1)}, p^{(2)} \ldots, p^{(t_h)}] \tag{1}$$

and

$$Y = [p^{(t_h+1)}, p^{(t_h+2)}, \ldots, p^{(t_h+t_f)}], \tag{2}$$

where

$$p^{(t)} = [crd_0^{(t)}, crd_1^{(t)}, \ldots, crd_n^{(t)}] \tag{3}$$

are the coordinates of all joints at time t, and n is the number of joints of a human skeleton. Depending on the pose skeleton format and the dataset, the coordinates can be 3D (e.g., in Human 3.6M Dataset),

$$crd_i^{(t)} = \left[x_i^{(t)}, y_i^{(t)}, z_i^{(t)}\right], \tag{4}$$

or 2D (e.g., in Penn Action Dataset),

$$crd_i^{(t)} = \left[x_i^{(t)}, y_i^{(t)}\right]. \tag{5}$$

The pose forecasting task is a sequence-to-sequence learning problem with a strong inherent structure constraint. Therefore, it is critical to design a data representation format to capture this inherent structure and proper neural network modules to process this data representation. In the following section, we will introduce our solution, especially how the structure constraint is encoded with our encoder.

4.2 Graph Neural Network-based Human Pose Forecasting

Figure 3 illustrates the framework of our proposed scheme. It consists of three components: (1) Input Representation, (2) Graph Convolutional Component, and (3) GRU-based Encoder-Decoder module. The details for each component are described below.

4.2.1 Input Representation. The input representation component tries to (1) reformat the input pose coordinates sequence into a format suitable to be processed by a convolutional neural network and (2) generate a graph to represent the connections among joints.

Spatial-Temporal Coordinate Image. Convolutional Neural Networks (CNNs) have been extensively studied and proved to work pretty well on spatial structural information, e.g., images. Therefore, if the raw input data can be represented in a format that is similar to an image (with a size of $w * h * c$), we can take advantage of CNNs' strength. We construct the spatial-temporal coordinate image as follows: assuming that coordinates of n joints of a human body in the past t_h time steps are given, such information can be represented in a 3D tensor F_{input} with a size of $(n \times t_h \times c)$ ("Input Representation" shown in Figure 3). Each row (n, the first dimension) has t_h values that correspond to t_h historical coordinates of one joint, each column (t_h, the second dimension) consists of the coordinates of all joints at that time step, and c is the dimension of the coordinates ($c = 2$ if 2D coordinates are used, $c = 3$ for 3D coordinates, or other values for other representations).

Input Graph Construction. In the human pose forecasting application scenario, the motion of a joint is impacted by its connected joints. This is highly similar to people's behaviors on a social network (one person can be directly impacted by his/her friends). This inspires us to represent the inter-joint interaction using an undirected graph $G = \{V, E\}$ (where V is a node set and E is an edge set) as what researchers have done for a social network [7].

Each node in the node set V corresponds to a joint of a human body. The node set V is defined as $V = \{v_{it} | i = 1, \ldots, n, t = 1, \ldots, t_h\}$ because the state of a joint may be different at different time steps, where n is the number of considered joints of a human body, and t_h is the observed time steps. The feature vector v_{it} on a node is the coordinate of the ith object at time t. Such a definition is the same as what we described in the previous subsection. Thus, the input representation is shared with the node set.

The edge set indicates if there is a connection between the two joints. As the "Raw Data" in Figure 3 shows, joints have interactions that are connected with edges at each time step t. Such an interaction is constructed according to the human skeleton; e.g., the left shoulder is connected to the left elbow, the left elbow is connected to the left wrist, and so forth. We call such a connection "spatial edge" and denote it as $E_S = \{v_{it}v_{jt} | (i, j \in D)\}$, where D is a set in which joints i and j are connected to each other at time t. In addition, because a sequence of poses are being processed, only considering the spatial edges at each independent time step is not enough. The historical connection information must also be incorporated frame by frame in temporal space. In this work, each joint in one time-step is connected to itself in another time-step via the temporal edge and denoted as $E_F = \{v_{it}v_{i(t+1)}\}$. All edges in E_F of one particular joint represent its trajectory over time steps. Thus, the complete edge set $E = \{E_S, E_F\}$.

To make the computation more efficient, we represent this graph using an adjacency matrix $A = \{A_0, A_1\}$, where A_0 is an identity matrix I representing self-connections in temporal space (corresponding to E_F), and A_1 is a spatial connection adjacency matrix (corresponding to E_S). Thus, at any time t,

$$A_0[i][j](or A_1[i][j]) = \begin{cases} 1, \text{if edge } \langle v_{it}, v_{jt} \rangle \in E \\ 0, \text{otherwise.} \end{cases} \tag{6}$$

Both A_0 and A_1 have a size of $(n \times n)$, where n equals the number of considered joints of a human body.

4.2.2 Graph Convolutional Component. The GC component is composed of several basic building blocks and each includes a convolutional layer and a graph operation layer. To be specific, useful temporal features are extracted, e.g., motion pattern of one joint, using convolutional layers, and they handle the inter-joint interaction in spatial space using graph operations. This means, flowing through each GC block, the input data is processed to refine the features temporally and spatially alternatively. In Figure 3, a GC component with three building blocks is illustrated.

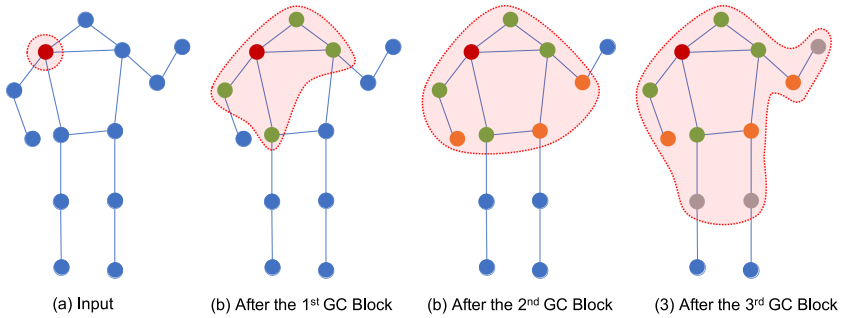

Fig. 4. The effect of the graph convolutional component for propagating joint information with stacked GC blocks.

Convolutional layer. Given a preprocessed input data $F_{input} := \mathbb{R}^{n \times t_h \times c}$ (where n is the number of joints, t_h is the observed time steps, and c is the dimension of the coordinates), the model passes it through a convolutional layer to compute convolutional feature maps f_{conv}. To ensure convolutional layers are extracting temporal features, their kernel sizes are set to (1×3) to force them to process the data along the temporal dimension (the second dimension). This can also be regarded as a 1D convolution over the time dimension for each human joint. Appropriate paddings and strides are added to make sure that each layer has an output feature map with the expected size.

Graph operation layer. Then, the generated convolutional feature maps f_{conv} are fed to a graph operation layer to take the interactions of connected joints into account. To make sure the value range of feature maps remains unchanged after performing graph operations, Graph A is normalized using the following equation (as done in [14]):

$$A_{norm} = \Lambda_j^{-\frac{1}{2}} A_j \Lambda_j^{-\frac{1}{2}}, \tag{7}$$

where Λ_j is computed as

$$\Lambda_j^{ii} = \sum_k \left(A_j^{ik}\right) + \alpha, \tag{8}$$

and we set $\alpha = 0.001$ to avoid empty rows in A_j.

Considering that the graph A is generated based on a manually designed rule (human skeleton), it may not be able to represent the interactions of joints properly. Thus, a trainable weight, denoted as A_{weight}, is added that has the same shape as graph A for each graph operation layer. Thus, the model will be trained to tune the A_{weight} to assign an appropriate weight to each edge. Then, the graph operation is calculated as

$$f_{graph} = \sum_{j=0}^{1} \left(A_{weight}^j \circ A_{norm}^j\right) f_{conv}, \tag{9}$$

where $\circ$ is the element-wise product.

In Figure 4, the effect of our graph convolutional component in the spatial domain is illustrated with an example input of 13 human joints. Before feeding the input data (Figure 4(a)) into the model, one joint (marked using a red circle) only contains its own information (i.e., its coordinates). After the data is passed through one GC block (Figure 4(b)), the information at the red joint is updated by capturing information on all highlighted joints (circled with a red dashed line). These highlighted joints have a direct connection to the red joint, so the GC block incorporates the impact of all of them. If the data is passed through more GC blocks (Figures 4(c) and 4(d)), the information

from more joints will be correlated. In other words, the proposed model can extract features by gradually expanding its receptive field based on human structural information.

4.2.3 GRU Encoder-Decoder Module. Following the GC component is the GRU-based (Gated Recurrent Units) encoder-decoder network that predicts the future poses for all considered joints simultaneously. The output of the graph convolutional component is fed into the encoder GRU at each time step. Then, the hidden feature of the encoder GRU, as well as coordinates of joints at the previous time step, are fed into a decoder GRU to predict the position coordinates at the current time step. Such a decoding process is repeated several times until the model predicts positions for all expected time steps (t_f) in the future. It is worth noting that the GC component can be concatenated with any more advanced sequence-to-sequence learning architecture such as the transformer [38] to get improved prediction performance.

4.2.4 Loss Function. The proposed model is trained as a regression task at each time. The overall loss can be computed as

$$Loss = \frac{1}{t_f} \sum_{t=1}^{t_f} loss^t \tag{10}$$

$$= \frac{1}{t_f} \sum_{t=1}^{t_f} \left\| Y^t_{pred} - Y^t_{GT} \right\|, \tag{11}$$

where t_f is the time step in the future (in Figure 3, $t_f = 3$), $loss^t$ is the loss at time t, and Y_{pred} and Y_{GT} are predicted positions and ground truth, respectively. The model is trained to minimize the *Loss*.

4.3 Online Learning for Model Adaptation

Online learning is the process of adapting a pre-trained deep neural network with the collected on-site data that reflects the real data distribution for optimized inference/testing. In GPFS, this adaptation is achieved by fine-tuning the pre-trained model with online collected person- and domain-specific pose data. While the model is being adapted online, we keep using the same model structure and same training loss function as we described in Section 4.2. However, the online learning uses a smaller learning rate 1e-4 (instead of 1e-3) and only the new data collected in the new environment to fine-tune the model (i.e., only weights of the model are updated) so that it would perform better in the new domain (new environment). The online adaption pipeline is illustrated in Figure 5.

Specifically, at the particular edge environment where the system is deployed, it consistently monitors the presence of persons, and a face detection and recognition system module is embedded to obtain the person's identity. Whenever a new person/face is identified, the system registers him/her as a new user with the associated features for face recognition. Before the adapted model is available for a user, the pre-trained model should be used to forecast future poses. At the same time, the real-time captured pose sequences for the particular user are dumped at the edge storage (e.g., a hard drive connected to the edge processing device).

Different from many other computer vision problems that require manual annotations to obtain the ground-truth learning target, the pose forecasting can automatically obtain the ground truth results by splitting a captured sequence into multiple (potentially overlapped) sub-sequences of length l_{raw}. Each sub-sequence forms a training sample of two parts: (1) the input part with length l_{input} (50 frames in our experiment) and (2) the output part with length l_{output} (10 frames for short-term forecast and 25 frames for long-term forecast; $l_{raw} = l_{input} + l_{output}$).

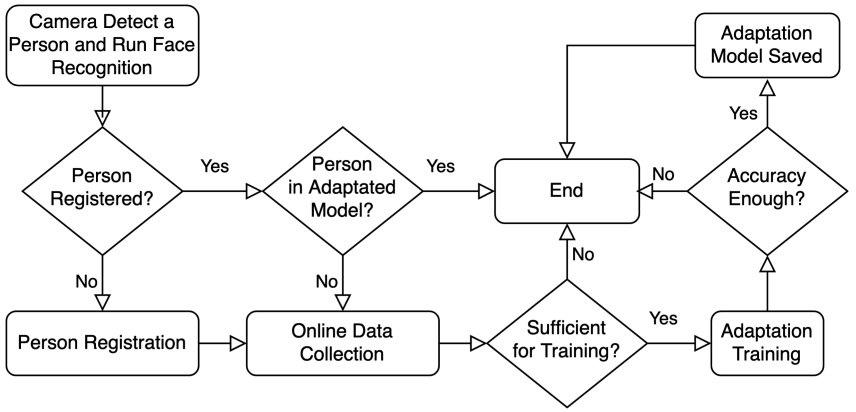

Fig. 5. The detailed flowchart of the online learning subsystem.

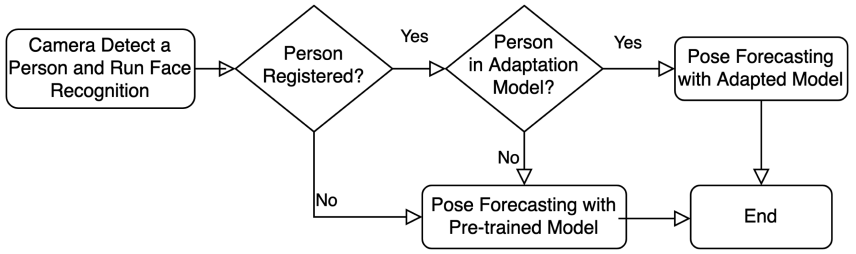

Fig. 6. The detailed flowchart of the pose forecasting subsystem.

With onsite data samples collected, the model adaptation dataset with both training and validation splits is built and a reasonably good dataset should cover data collected spanning multiple days at different times. Prior to performing model adaptation, the baseline accuracy A_{base} for a user is calculated using the pre-trained model on the constructed validation split. The new adapted model will only be deployed when its accuracy achieves $A_{base} + A_{thres}$, where A_{thres} is a configurable system hyper-parameter. When multiple users present, the adaptation would be performed using the training dataset for all the users and the adapted model would only be accepted if the validation accuracy on each user's validation split satisfies the requirement.

In Figure 6, we present the detailed flowchart of the pose forecasting subsystem in action. Note that the pre-trained model would always be available for users not included in the adaptation model.

5 EXPERIMENTS

We first describe the implementation details. Then we introduce the datasets and evaluation metrics. After that, we present the experimental results that include (1) the forecasting performance on the benchmark dataset (both quantitively and qualitatively) by comparing it with the state-of-the-art methods, (2) the online learning performance, and (3) the system overhead.

5.1 Implementation Details

We describe the implementation details and the settings of important parameters of the deep neural networks as follows.

5.1.1 Edge Processing Server. We deploy the system on an edge server running Ubuntu 16.04 with 4.0 GHz Intel Core i7 CPU, 32GB memory, and an NVIDIA Titan Xp Graphics Card. We use PyTorch 1.3.1 [28] as our deep learning library.

5.1.2 Hyper-Parameters for Deep Neural Network.

Input preprocessing. For the human pose forecasting task, we consider n joints (n is variant for different datasets) of a human body. We follow Pavllo et al. [30] to convert the input into quaternion representation, because quaternions are free of discontinuities and singularities, are more numerically stable, and are more computationally efficient than other representations [32]. Thus, the number of input channels $c = 4$ after being represented with quaternions. In addition, n joints are connected according to the human skeleton (similar as shown in Figure 4). Thus, the graph of current input (adjacency matrix) is fixed and will be passed into the following graph convolutional blocks.

Graph convolutional component. The graph convolutional component consists of three basic building blocks (Figure 3). Therefore, it includes three convolutional layers, denoted as $\{conv2d\_i | i = 1, 2, 3\}$. All Conv2D layers have a convolutional kernel with a size of (1×3). For $conv2d\_1$ and $conv2d\_2$, the number of their output channels is increased to 64 to learn more complicated features. For the last convolutional layer ($conv2d\_3$), we try to extract semantically more meaningful features (in our case the joint coordinates) [37] and reduce the number of channels from 64 back to 4, which is the dimension of the quaternions used in the input. The reduced number of channels is also critical for efficient computation. Each convolutional layer is followed by a graph operation layer, which does not change the size of feature maps, and they share the same adjacency matrix (derived from the Input Preprocessing component) but with different trainable weights (A_{weight}). Therefore, the output of the graph convolutional component has the same size of $(n \times t_h \times 4)$ as the input. We add a **Batch Normalization (BN)** layer after the Input Preprocessing and each graph operation layer. To avoid overfitting, we randomly drop out features (0.5 probability) after each graph operation.

GRU Encoder-Decoder Module. Both the encoder and decoder of this prediction model are two-layer **Gated Recurrent Unit(GRU)** networks. We set the number of hidden units of these two GRUs equal to the output dimension (e.g., $4 \times n$, where n is the number of objects and 4 is the quaternion representation). The input of the encoder has four channels that are the same as the output of the Graph Convolutional Model.

Training process. We train the model using the **Stochastic Gradient Descent (SGD)** optimizer with a 0.001 starting learning rate. The learning rate is reduced by multiplying 0.1 once per 5 epochs until the loss becomes converged. We set $batch\_size = 128$ during training. The learning rate is set to 1e-4 for online learning.

5.2 Datasets

Human 3.6M dataset. The Human 3.6M dataset [12] is one of the largest datasets of human motion capture. This dataset consists of motion capture data from seven subjects (person IDs are 1, 5, 6, 7, 8, 9, and 11) performing 15 different actions. In this dataset, 32 joint locations of each person are provided. Following previous work [5, 13, 23], we use Subject 5 as the test data and Subjects 1, 6, 7, 8, 9, and 11 as training. Consistent with the previous work [5, 13, 23], we train our model using the past 50 frames (2,000 ms) and predict the future 10 frames (400 ms) for short-term prediction. For the long-term forecast, the model predicts the next 25 frames (1,000 ms). Following the previous work [5, 13, 23], during training (offline training and online training), we randomly select one clip from each sample in one epoch.

Penn Action dataset. The Penn Action dataset [42] provides 13 human joint coordinates over 15 different actions. In total, this dataset consists of 2,326 video sequences. Following [3, 5, 42],

Table 1. MAE for Short-Term Prediction over Four Actions from Human 3.6M Dataset

	Walking				Eating				Smoking				Discussion			
Milliseconds	**80**	**160**	**320**	**400**	**80**	**160**	**320**	**400**	**80**	**160**	**320**	**400**	**80**	**160**	**320**	**400**
ERD [8]	0.93	1.18	1.59	1.78	1.27	1.45	1.66	1.80	1.66	1.95	2.35	2.42	2.27	2.47	2.68	2.76
LSTM-3LR [8]	0.77	1.00	1.29	1.47	0.89	1.09	1.35	1.46	1.34	1.65	2.04	2.16	1.88	2.12	2.25	2.23
SRNN [13]	0.81	0.94	1.16	1.30	0.97	1.14	1.35	1.46	1.45	1.68	1.94	2.08	1.22	1.49	1.83	1.93
Residual [23]	0.28	0.49	0.72	0.81	0.23	0.39	0.62	0.76	0.33	0.61	1.05	1.15	0.31	0.68	1.01	1.09
TP-RNN [5]	0.25	0.41	0.58	0.65	0.20	0.33	**0.53**	0.67	0.26	0.47	0.88	0.90	0.30	0.66	0.96	1.04
QuaterNet [30]	0.21	**0.34**	0.56	**0.62**	0.20	0.35	0.58	0.70	0.25	0.47	0.93	0.90	0.26	0.60	0.85	0.93
GPFS (Ours)	**0.20**	**0.34**	**0.55**	**0.62**	**0.17**	**0.30**	0.54	**0.66**	**0.23**	**0.43**	**0.85**	**0.82**	**0.23**	**0.57**	**0.82**	**0.90**

In each column, the best results are typeset in boldface and the second best are underlined (the lower the better).

we split the dataset into 1,258 training video sequences and 1,068 testing video sequences. For this dataset, the model only takes the first frame as its input and predicts the next 15 frames.

5.3 Metrics of Forecasting Accuracy

Two metrics are used in evaluation:

MAE distance. Following [5], **mean average error (MAE)** is used for the Human 3.6M dataset. It measures the mean average distance between the predicted pose in the angle space and the ground-truth pose.

PCK@0.05. The PCK@0.05 metric is used for the Penn Action dataset as done in [3, 5]. The PCK metric calculates the percentage of joint locations correctly predicted by the model. With threshold 0.05, a joint location is counted as correctly predicted if the normalized distance between its predicted and ground-truth locations is less than 0.05. The distance is normalized typically based on the size of the full body or the head. Since we have the coordinates of human body joints, we normalize the distance by $max(h, w)$, where h and w are the height and width of the human body (longest distance between any two joints in height and width dimensions separately).

5.4 Benchmark Results for Pose Forecasting

5.4.1 State-of-the-Art Methods in Comparison. To demonstrate the performance improvement with the introduced graph convolutional component, we compare our result with the current best-performing solutions on our selected evaluation datasets. In particular, we have compared with the following methods: ERD [8], LSTM-3LR [8], SRNN [13], Residual [23], TP-RNN [5], QuaterNet [30], and Droupoout-AE [11]. None of those methods has included the graph module in their design.

5.4.2 Benchmark Results on Human 3.6M Dataset. We first evaluate the performance of our proposed GPFS to predict human motions using the Human 3.6M dataset. In Table 1, we report short-term prediction results over four common actions from the Human 3.6M dataset.

From this table, we can see that our proposed GPFS achieves better results than the state-of-the-art solution. GPFS only performs a little worse (0.01 worse in terms of MAE) than TP-RNN at 320 ms for the "Eating" action. Besides that, GPFS performs better than all existing solutions. Especially for "Smoking" and "Discussion," GPFS achieves much lower error than the state-of-the-art scheme (QuaterNet). These results prove that GPFS performs well in the human motion short-term prediction task.

We also report GPFS's long-term prediction results in Table 2 and Table 3. From Table 2, one can see that GPFS achieves better results at all considered future time steps on four common actions of the Human 3.6M dataset. If we compare Table 1 and Table 2, we can notice that GPFS performs better in making long-term predictions than short-term predictions. Specifically, when

Table 2. MAE for Long-Term Prediction over Four Actions from Human 3.6M Dataset

	Walking					Eating					Smoking					Discussion				
Milliseconds	**80**	**160**	**320**	**560**	**1,000**	**80**	**160**	**320**	**560**	**1,000**	**80**	**160**	**320**	**560**	**1,000**	**80**	**160**	**320**	**560**	**1,000**
ERD [8]	1.30	1.56	1.84	2.00	2.38	1.66	1.93	2.88	2.36	2.41	2.34	2.74	3.73	3.68	3.82	2.67	2.97	3.23	3.47	2.92
LSTM-3LR [8]	1.18	1.50	1.67	1.81	2.20	1.36	1.79	2.29	2.49	2.82	2.05	2.34	3.10	3.24	3.42	2.25	2.33	2.45	2.48	2.93
SRNN [13]	1.08	1.34	1.60	1.90	2.13	1.35	1.71	2.12	2.28	2.58	1.90	2.30	2.90	3.21	3.23	1.67	2.03	2.20	2.39	2.43
Droupoout-AE [11]	1.00	1.11	1.39	1.55	1.39	1.31	1.49	1.86	1.76	2.01	0.92	1.03	1.15	1.38	1.77	1.11	1.20	1.38	1.53	1.73
Residual [23]	0.32	0.54	0.72	0.86	0.96	0.25	0.42	0.64	0.94	1.30	0.33	0.60	1.01	1.23	1.83	0.34	0.74	1.04	1.43	1.75
TP-RNN [5]	0.25	0.41	0.58	0.74	0.77	0.20	0.33	**0.53**	0.84	1.14	0.26	0.48	0.88	0.98	1.66	0.30	0.66	0.98	1.39	1.74
GPFS (Ours)	**0.22**	**0.37**	**0.56**	**0.69**	**0.73**	**0.19**	**0.32**	**0.53**	**0.66**	**1.13**	**0.24**	**0.45**	**0.87**	**0.94**	**1.58**	**0.25**	**0.58**	**0.84**	**1.29**	**1.67**

In each column, the best results are typeset in boldface and the second best are underlined (the lower the better).

Table 3. MAE for Long-Term Prediction over the Remaining 11 Actions in Human 3.6M Dataset

	Directions						Greeting						Talking on the Phone					
Milliseconds	80	160	320	400	560	1,000	80	160	320	400	560	1,000	80	160	320	400	560	1,000
Residual [23]	0.44	0.69	0.83	0.94	1.03	1.49	0.53	0.88	1.29	1.45	1.72	1.89	0.61	1.12	1.57	1.74	1.59	1.92
TP-RNN [5]	0.38	0.59	0.75	0.83	0.95	1.38	0.51	0.86	1.27	1.44	1.72	1.81	0.57	1.08	1.44	1.59	**1.47**	**1.68**
GPFS (Ours)	**0.34**	**0.49**	**0.70**	**0.80**	**0.89**	**1.28**	**0.45**	**0.75**	**1.11**	**1.27**	**1.53**	**1.62**	**0.56**	**1.07**	**1.41**	**1.55**	1.52	1.72
	Posing						**Purchases**						**Sitting**					
Milliseconds	80	160	320	400	560	1,000	80	160	320	400	560	1,000	80	160	320	400	560	1,000
Residual [23]	0.47	0.87	1.49	1.76	1.96	**2.35**	0.60	0.86	1.24	1.30	1.58	**2.26**	0.44	0.74	1.19	1.40	1.57	2.03
TP-RNN [5]	0.42	0.76	1.29	1.54	1.75	2.47	0.59	**0.82**	**1.12**	**1.18**	**1.52**	2.28	0.41	0.66	1.07	1.22	1.35	1.74
GPFS (Ours)	**0.22**	**0.49**	**1.09**	**1.34**	**1.61**	**2.35**	**0.58**	0.83	1.18	1.23	1.55	2.31	**0.34**	**0.53**	**0.91**	**1.09**	**1.22**	**1.58**
	Sitting Down						**Taking Photo**						**Waiting**					
Milliseconds	80	160	320	400	560	1,000	80	160	320	400	560	1,000	80	160	320	400	560	1,000
Residual [23]	0.51	0.93	1.44	1.65	1.94	2.55	0.33	0.65	0.97	1.09	1.19	1.47	0.34	0.65	1.09	1.28	1.61	2.27
TP-RNN [5]	0.41	0.79	1.13	1.27	1.47	1.93	0.26	0.51	0.80	0.95	1.08	1.35	0.30	0.60	1.09	1.31	1.71	2.46
GPFS (Ours)	**0.35**	**0.68**	**0.99**	**1.11**	**1.28**	**1.84**	**0.22**	**0.45**	**0.72**	**0.85**	**0.97**	**1.18**	**0.28**	**0.55**	**0.99**	**1.20**	**1.54**	**2.23**
	Walking Dog						**Walking Together**						**Average of All 15**					
Milliseconds	80	160	320	400	560	1,000	80	160	320	400	560	1,000	80	160	320	400	560	1,000
Residual [23]	0.56	0.95	1.28	1.39	1.68	1.92	0.31	0.61	0.84	0.89	1.00	1.43	0.43	0.75	1.11	1.24	1.42	1.83
TP-RNN [5]	0.53	0.93	1.24	1.38	1.73	1.98	0.23	0.47	0.67	0.71	0.78	1.28	0.37	0.66	0.99	1.11	1.30	1.71
GPFS (Ours)	**0.50**	**0.84**	**1.14**	**1.27**	**1.56**	**1.87**	**0.21**	**0.43**	**0.59**	**0.63**	**0.69**	**1.24**	**0.33**	**0.59**	**0.91**	**1.02**	**1.20**	**1.62**

In each column, the best results are typeset in boldface (the lower the better).

making long-term predictions, GPFS gains more improvement from the state-of-the-art solution than making short-term predictions. It is because GPFS considers the impacts of nearby objects, which helps the model make a longer precise prediction.

In Table 3, we report the remaining 11 actions in the Human 3.6M dataset and the average prediction results over all of the actions. We have excluded other low-performing methods for more clear presentation but just keep the most competitive methods, Residual and TP-RNN. GPFS achieves better long-term predictions for most actions. Precisely, GPFS only performs worse in 7 out of 66 predictions than the existing solutions. The majority of the slightly sub-optimal predictions (compared to TP-RNN) are made on the scenario "Making Purchases," where the training poses and the testing poses are not quite consistent. On average (the "Average of All 15" columns), GPFS improves the performance of the state-of-the-art solution by almost 0.1 in terms of MAE.

Table 4. Comparison to Prior Works Using the Penn Action Dataset in Terms of PCK@0.05 (the Higher the Better)

Future Frame	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
Residual [23]	82.4	68.3	58.5	50.9	44.7	40.0	36.4	33.4	31.3	29.5	28.3	27.3	26.4	25.7	25.0	24.5
3D-PFNet [3]	79.2	60.0	49.0	43.9	41.5	40.3	39.8	39.7	40.1	40.5	41.1	41.6	42.3	42.9	43.2	43.3
TP-RNN w/o init vel. [5]	82.3	68.9	61.5	56.9	53.9	51.7	50.0	48.5	47.3	46.2	45.6	45.0	44.6	44.3	44.1	43.9
TP-RNN w/ init vel. [5]	**84.5**	**72.0**	64.8	60.3	57.2	55.0	53.4	52.1	50.9	50.0	49.3	48.7	48.3	47.9	47.6	47.3
GPFS (Ours)	84.0	71.7	**64.8**	**60.7**	**57.8**	**55.9**	**54.2**	**52.8**	**52.4**	**51.9**	**51.6**	**51.4**	**51.3**	**51.3**	**51.2**	**51.1**

In each column, the best results are typeset in boldface and the second best are underlined.

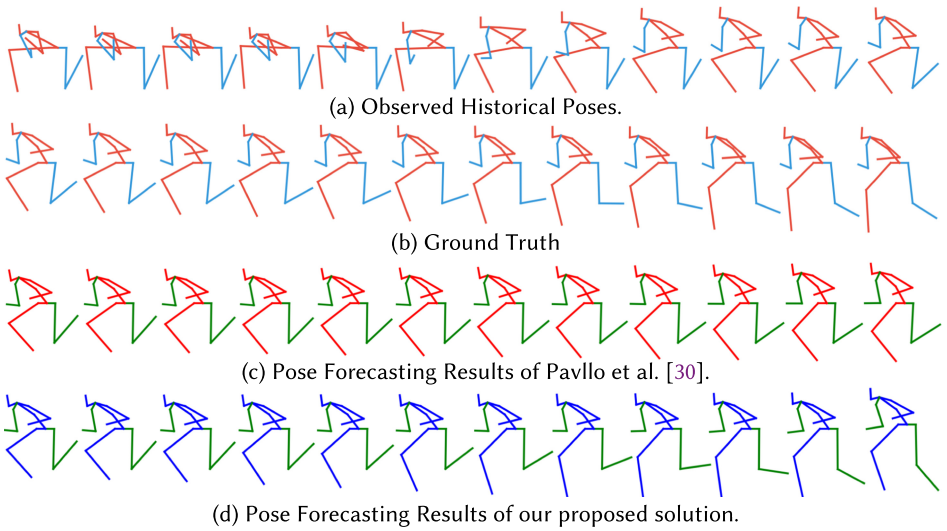

Fig. 7. Visualized forecasting results using a "takingphoto" testing sample from the Human 3.6M dataset. The person is taking a photo while leaning on one leg. After a few frames, the person stands up.

5.4.3 Benchmark Results on Penn Action Dataset. We evaluate the performance of our GPFS on another human motion prediction dataset, the Penn Action dataset. In Table 4, we compare our GPFS with prior works in terms of PCK@0.05, and a higher value means better performance. On this dataset, GPFS also achieves better results than the state-of-the-art solutions at most time steps (14 out of 16 predictions). Consistently, GPFS works much better in making long-term predictions than other schemes.

5.4.4 Visualization of Prediction Results. Figure 7 and Figure 8 show visualized results using two testing samples from the Human 3.6M dataset. One can see that our proposed scheme achieves much better results than Pavllo et al. [30]. Comparing Figures 7(c) to 7(d), Pavllo et al. does not predict the person is going to stand up, while our proposed solution correctly forecasts such an intent. Comparing Figures 8(c) to 8(d), Pavllo et al. predict the person is going to raise his (or her) hand over his (or her) head. However, our forecasted results are close to the ground truth.

5.5 Experimental Results of Online Adaptation

Here, we evaluate the performance of our Online Model Adaptation Module on the Human3.6M dataset. To simulate the real-life application scenarios, we design the following two online adaptation settings:

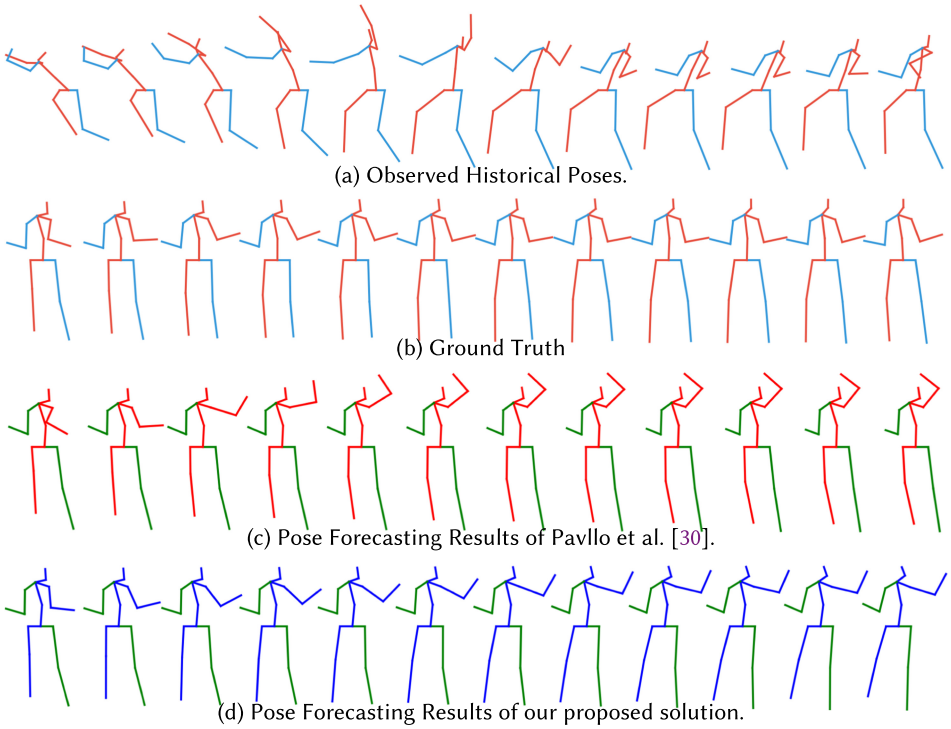

Fig. 8. Visualized results of a "posing" testing sample in the Human 3.6M dataset. The person is posing in various poses.

Table 5. MAE Results of Online Adaptation (**Setting 1**) on the Human 3.6M Dataset

Prediction Horizon (ms)	**320**	**400**	**560**	**1,000**	**4,000**	**Sum**	**Improve**	**Time (s)**
GPFS (Ours)	0.93	1.06	1.28	1.79	2.40	7.46	-	-
Fine-tune GPFS 10 Epochs	0.91	1.02	1.21	1.64	2.24	7.02	−0.44	1.16
Fine-tune GPFS 20 Epochs	0.91	1.02	1.21	1.62	2.18	6.94	−0.52	2.33
Fine-tune GPFS 30 Epochs	0.90	1.01	1.20	1.61	2.16	6.88	−0.58	3.52
Fine-tune GPFS 40 Epochs	0.90	1.01	1.19	1.60	2.12	6.82	−0.64	4.68
Fine-tune GPFS 50 Epochs	0.90	1.00	1.19	1.59	2.12	6.80	−0.66	5.82

For MAE, the lower the better, so we use "-" to represent performance improvement. We collected computation time on a single NVIDIA TitanXp GPU.

Setting 1: The online collected data covers all potential actions of the edge user. To simulate such an application scenario, we set Subject 5 in the Human3.6M dataset (S5) as the edge user, and the data belonging to him are regarded as the online collected data (in total 20 minutes long). We equally split the collected data of S5 into the training and testing subset (1:1) for each of the 15 actions. Table 5 shows the experimental results.

Setting 2: The newly collected data only covers several (not all) potential actions of the edge user to evaluate the generalization capability. This is a more challenging but also more realistic setting. To simulate such an application scenario, we also set Subject 5 in the Human3.6M dataset (S5) as the edge user, and we use the leave-one-out strategy; i.e., each time, one action is selected for testing, and the remaining action classes are used for training. The Human3.6M dataset consists

Table 6. MAE Results of Online Adaptation (**Setting 2**) on the Human 3.6M Dataset

Prediction Horizon (ms)	320	400	560	1,000	4,000	Sum	Improve	Time (s)
GPFS (Ours)	0.93	1.06	1.28	1.79	2.40	7.46	-	-
Fine-tune GPFS 10 Epochs	0.91	1.03	1.23	1.66	2.28	7.11	−0.35	1.19
Fine-tune GPFS 20 Epochs	0.91	1.02	1.22	1.64	2.20	6.99	−0.47	2.37
Fine-tune GPFS 30 Epochs	0.90	1.02	1.21	1.62	2.19	6.94	−0.52	3.56
Fine-tune GPFS 40 Epochs	0.90	1.02	1.21	1.62	2.18	6.93	−0.53	4.76
Fine-tune GPFS 50 Epochs	0.90	1.01	1.20	1.61	2.18	6.90	−0.56	5.97

We collected computation time on a single NVIDIA TitanXp GPU.

Table 7. MAE Results of Online Adaptation (**Setting 2 Half Finetuning Data**) on the Human 3.6M Dataset

Prediction Horizon (ms)	320	400	560	1,000	4,000	Sum	Improve	Time (s)
GPFS (Ours)	0.93	1.06	1.28	1.79	2.40	7.46	-	-
Fine-tune GPFS 10 Epochs	0.91	1.03	1.23	1.66	2.27	7.10	−0.37	1.16
Fine-tune GPFS 20 Epochs	0.91	1.03	1.23	1.65	2.21	7.03	−0.44	2.35
Fine-tune GPFS 30 Epochs	0.91	1.03	1.22	1.65	2.20	7.01	−0.46	3.54
Fine-tune GPFS 40 Epochs	0.91	1.02	1.21	1.63	2.18	6.95	−0.52	4.72
Fine-tune GPFS 50 Epochs	0.90	1.01	1.21	1.63	2.19	6.93	−0.54	5.86

We collected computation time on a single NVIDIA TitanXp GPU.

of 15 action classes. Thus, we repeat such a process 15 times and average the results. In addition, we run another set of experiments by reducing the size of the training dataset by half (each action has only one recorded video) to evaluate the impact of data volume. The experimental results for this setting are reported in Table 6 and Table 7.

The first row (**GPFS (Ours)**) of all three tables indicates the accuracy of the pre-trained models before online adaptation. The following rows show the accuracy after online adaptation as well as the training time cost with a varying number of training epochs. Table 5 shows consistent results as shown in Table 6 and Table 7. First, the model can be fine-tuned within a very short time (less than 6 seconds) for 50 epochs. Second, after fine-tuning, the performance of the model improved roughly 10% (fine-tuned 50 epochs), though the second setting has made the problem more challenging. In addition, such online learning would not require too much data to achieve satisfactory accuracy improvement by comparing the results of Table 6 and Table 7. Based on the experimental results shown in both tables, one can conclude that our proposed GPFS can run extremely fast on a local edge processing server and can be online enhanced by using newly collected data.

5.6 System Overhead

In Table 5 and Table 6, we show that the online model training can be finished in near real time. Here, we further demonstrate that GPFS is computationally feasible for edge computing in terms of inference time and storage cost. The evaluation is performed using the Human 3.6M dataset.

5.6.1 Inference Time. The system needs to be computationally efficient to enable real-time inference (25 fps) at the edge. The inference includes two parts: the pose estimation and the future pose forecasting. For pose estimation, we use XNect [25] for real-time processing with a Single RGB Camera. With a single NVIDIA TitanXp GPU, XNect could generate the pose skeleton from a single image in 7.4 ms (i.e., 135 fps). For future pose forecasting, we present the experiment results in Table 8, where 50 historical frames of 2 seconds are used as input and results of different

Table 8. Inference Time of Pose Forecasting on a Single NVIDIA TitanXp GPU

Prediction Horizon (ms)	80	160	320	400	560	1,000	4,000
GPFS (Ours)	5.5 ms	6.0 ms	7.4 ms	8.2 ms	9.5 ms	13.3 ms	39.1 ms

prediction horizons have been listed to demonstrate the efficiency. For example, forecasting the pose of the future 400 ms only takes 8.2 ms.

5.6.2 Storage Cost for Online Data Collection. The online data collected for edge users is storage-friendly since only skeleton coordinates are required other than the raw RGB frames. Specifically, for valid pose recording of 24 hours long (i.e., assuming the person presented at each frame), the storage cost is only 1.4 GB in the Numpy data format. In addition, since the online training scheme happens at the network edge, there would be no privacy issue. The data can be deleted after the online model adaptation is finished.

6 CONCLUSION

In this article, we propose a graph-based human pose forecasting system to forecast human pose in the Smart Home/IoT environment. The contributions of GPFS are twofold. First, GPFS introduces a graph-based method to represent the interaction among all considered joints of a human body and employs a graph-enhanced encoder-decoder framework to make predictions. This new architecture design effectively captures the coherent connectivity among human joints and enables learning in an end-to-end manner. Second, considering the domain bias at a particular deployment environment, an online learning pipeline is designed for model adaptation to further improve the performance. In addition, a hybrid cloud-edge architecture is adopted for privacy protection. The prototype implementation and experimental results on two popular benchmark datasets show that (1) the graph-based deep learning model achieves better forecasting results than existing methods, (2) the online learning significantly boosts the accuracy, and (3) the deployed system runs efficiently on an edge server.

Our future work will be focused on two directions. First, we would like to explore more on the deep neural network architecture design to further improve the accuracy. Specifically, by taking advantage of advanced seq2seq learning architecture such as conv-seq2seq [10] and transformer [38], we would expect further accuracy improvement as well as improved computational efficiency (due to their parallel processing capability). Second, we would like to study the feasibility of applying model adaptation methods other than fine-tuning to avoid maintaining multiple models at the edge. Particularly, we would like to adapt the *learning without forgetting* method [18], which prevents a model from forgetting its old knowledge while learning continually on data of new domains.

REFERENCES

[1] Yazan Abu Farha, Alexander Richard, and Juergen Gall. 2018. When will you do what? Anticipating temporal occurrences of activities. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 5343–5352.

[2] Z. Cao, G. Hidalgo, T. Simon, S.-E. Wei, and Y. Sheikh. 2019. "OpenPose: realtime multi-person 2D pose estimation using Part Affinity Fields." *IEEE transactions on pattern analysis and machine intelligence* 43, 1 (2019), 172–186.

[3] Yu-Wei Chao, Jimei Yang, Brian Price, Scott Cohen, and Jia Deng. 2017. Forecasting human dynamics from static images. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 548–556.

[4] Yucheng Chen, Yingli Tian, and Mingyi He. 2020. Monocular human pose estimation: A survey of deep learning-based methods. *Computer Vision and Image Understanding* 192 (2020), 102897. DOI: https://doi.org/10.1016/j.cviu.2019.102897

[5] Hsu-kuang Chiu, Ehsan Adeli, Borui Wang, De-An Huang, and Juan Carlos Niebles. 2019. Action-agnostic human pose forecasting. In *2019 IEEE Winter Conference on Applications of Computer Vision (WACV'19)*. IEEE, 1423–1432.

[6] Zhiyong Cui, Kristian Henrickson, Ruimin Ke, and Yinhai Wang. 2018. Traffic graph convolutional recurrent neural network: A deep learning framework for network-scale traffic learning and forecasting. *arXiv preprint arXiv:1802.07007* (2018).

[7] Wenqi Fan, Yao Ma, Qing Li, Yuan He, Eric Zhao, Jiliang Tang, and Dawei Yin. 2019. Graph neural networks for social recommendation. In *The World Wide Web Conference (WWW'19)*. Association for Computing Machinery, New York, NY, 417–426. DOI: https://doi.org/10.1145/3308558.3313488

[8] Katerina Fragkiadaki, Sergey Levine, Panna Felsen, and Jitendra Malik. 2015. Recurrent network models for human dynamics. In *Proceedings of the IEEE International Conference on Computer Vision*. 4346–4354.

[9] Yaroslav Ganin and Victor Lempitsky. 2015. Unsupervised domain adaptation by backpropagation. In *Proceedings of the 32nd International Conference on Machine Learning (Proceedings of Machine Learning Research)*, Francis Bach and David Blei (Eds.), Vol. 37. PMLR, Lille, France, 1180–1189. http://proceedings.mlr.press/v37/ganin15.html

[10] Jonas Gehring, Michael Auli, David Grangier, Denis Yarats, and Yann N. Dauphin. 2017. Convolutional sequence to sequence learning. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*. JMLR.org, 1243–1252.

[11] Partha Ghosh, Jie Song, Emre Aksan, and Otmar Hilliges. 2017. Learning human motion models for long-term predictions. In *2017 International Conference on 3D Vision (3DV'17)*. IEEE, 458–466.

[12] Catalin Ionescu, Dragos Papava, Vlad Olaru, and Cristian Sminchisescu. 2013. Human3. 6m: Large scale datasets and predictive methods for 3d human sensing in natural environments. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 36, 7 (2013), 1325–1339.

[13] Ashesh Jain, Amir R. Zamir, Silvio Savarese, and Ashutosh Saxena. 2016. Structural-RNN: Deep learning on spatio-temporal graphs. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 5308–5317.

[14] Thomas N. Kipf and Max Welling. 2016. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907* (2016).

[15] Hema S. Koppula and Ashutosh Saxena. 2015. Anticipating human activities using object affordances for reactive robotic response. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 38, 1 (2015), 14–29.

[16] D. Li, T. Salonidis, N. V. Desai, and M. C. Chuah. 2016. DeepCham: Collaborative edge-mediated adaptive deep learning for mobile object recognition. In *2016 IEEE/ACM Symposium on Edge Computing (SEC'16)*. 64–76.

[17] Dawei Li, Serafettin Tasci, Shalini Ghosh, Jingwen Zhu, Junting Zhang, and Larry Heck. 2019. RILOD: Near real-time incremental learning for object detection at the edge. In *Proceedings of the 4th ACM/IEEE Symposium on Edge Computing (SEC'19)*. Association for Computing Machinery, New York, NY, 113–126. DOI: https://doi.org/10.1145/3318216.3363317

[18] Z. Li and D. Hoiem. 2018. Learning without forgetting. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 40, 12 (2018), 2935–2947. DOI: https://doi.org/10.1109/TPAMI.2017.2773081

[19] Junwei Liang, Lu Jiang, Juan Carlos Niebles, Alexander G. Hauptmann, and Li Fei-Fei. 2019. Peeking into the future: Predicting future person activities and locations in videos. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 5725–5734.

[20] Xiaodan Liang, Lisa Lee, Wei Dai, and Eric P. Xing. 2017. Dual motion GAN for future-flow embedded video prediction. In *Proceedings of the IEEE International Conference on Computer Vision*. 1744–1752.

[21] Ziwei Liu, Raymond A. Yeh, Xiaoou Tang, Yiming Liu, and Aseem Agarwala. 2017. Video frame synthesis using deep voxel flow. In *Proceedings of the IEEE International Conference on Computer Vision*. 4463–4471.

[22] William Lotter, Gabriel Kreiman, and David Cox. 2016. Deep predictive coding networks for video prediction and unsupervised learning. *arXiv preprint arXiv:1605.08104* (2016).

[23] Julieta Martinez, Michael J. Black, and Javier Romero. 2017. On human motion prediction using recurrent neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2891–2900.

[24] Michael Mathieu, Camille Couprie, and Yann LeCun. 2015. Deep multi-scale video prediction beyond mean square error. *arXiv preprint arXiv:1511.05440* (2015).

[25] Dushyant Mehta, Oleksandr Sotnychenko, Franziska Mueller, Weipeng Xu, Mohamed Elgharib, Pascal Fua, Hans-Peter Seidel, Helge Rhodin, Gerard Pons-Moll, and Christian Theobalt. 2020. XNect: Real-time multi-person 3D motion capture with a single RGB camera. *ACM Transactions on Graphics* 39, 4 (2020), 17. DOI: https://doi.org/10.1145/3386569.3392410

[26] L. Minh Dang, Kyungbok Min, Hanxiang Wang, Md. Jalil Piran, Cheol Hee Lee, and Hyeonjoon Moon. 2020. Sensor-based and vision-based human activity recognition: A comprehensive survey. *Pattern Recognition* 108 (2020), 107561. DOI: https://doi.org/10.1016/j.patcog.2020.107561

[27] Junhyuk Oh, Xiaoxiao Guo, Honglak Lee, Richard L. Lewis, and Satinder Singh. 2015. Action-conditional video prediction using deep networks in atari games. In *Advances in Neural Information Processing Systems*. 2863–2871.

[28] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. 2019. Pytorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems*. 8026–8037.

[29] Viorica Patraucean, Ankur Handa, and Roberto Cipolla. 2015. Spatio-temporal video autoencoder with differentiable memory. *arXiv preprint arXiv:1511.06309* (2015).
[30] Dario Pavllo, David Grangier, and Michael Auli. 2018. Quaternet: A quaternion-based recurrent model for human motion. *arXiv preprint arXiv:1805.06485* (2018).
[31] Federico Pernici, Federico Bartoli, Matteo Bruni, and Alberto Del Bimbo. 2018. Memory based online learning of deep representations from video streams. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR'18)*.
[32] Edward Pervin and Jon A. Webb. 1982. *Quaternions in Computer Vision and Robotics*. Technical Report. Carnegie-Mellon UNIV Pittsburgh PA Dept of Computer Science.
[33] Siyuan Qi, Siyuan Huang, Ping Wei, and Song-Chun Zhu. 2017. Predicting human activities using stochastic grammar. In *Proceedings of the IEEE International Conference on Computer Vision*. 1164–1172.
[34] Nima Sedaghat. 2016. Next-flow: Hybrid multi-tasking with next-frame prediction to boost optical-flow estimation in the wild. *arXiv preprint arXiv:1612.03777* 1, 2 (2016), 6.
[35] Lei Shi, Yifan Zhang, Jian Cheng, and Hanqing Lu. 2019. Two-stream adaptive graph convolutional networks for skeleton-based action recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 12026–12035.
[36] Nitish Srivastava, Elman Mansimov, and Ruslan Salakhudinov. 2015. Unsupervised learning of video representations using LSTMs. In *International Conference on Machine Learning*. 843–852.
[37] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. 2015. Going deeper with convolutions. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 1–9.
[38] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems*. 5998–6008.
[39] Tianfan Xue, Jiajun Wu, Katherine Bouman, and Bill Freeman. 2016. Visual dynamics: Probabilistic future frame synthesis via cross convolutional networks. In *Advances in Neural Information Processing Systems*. 91–99.
[40] Sijie Yan, Yuanjun Xiong, and Dahua Lin. 2018. Spatial temporal graph convolutional networks for skeleton-based action recognition. In *32nd AAAI Conference on Artificial Intelligence*.
[41] Ceyuan Yang, Zhe Wang, Xinge Zhu, Chen Huang, Jianping Shi, and Dahua Lin. 2018. Pose guided human video generation. In *Proceedings of the European Conference on Computer Vision (ECCV'18)*. 201–216.
[42] Weiyu Zhang, Menglong Zhu, and Konstantinos G. Derpanis. 2013. From actemes to action: A strongly-supervised representation for detailed action understanding. In *Proceedings of the IEEE International Conference on Computer Vision*. 2248–2255.

Received July 2020; revised February 2021; accepted April 2021