IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received June 3, 2019, accepted July 1, 2019, date of publication July 5, 2019, date of current version August 16, 2019.

Digital Object Identifier 10.1109/ACCESS.2019.2927028

Transferable Environment Model With Disentangled Dynamics

QI YAN[1,2], SHANGQI GUO[1,2], DAGUI CHEN[1,2], ZHILE YANG[1,2], AND FENG CHEN[1,2,3], (Member, IEEE)

[1]Department of Automation, Tsinghua University, Beijing 100086, China
[2]Beijing Innovation Center for Future Chip, Tsinghua University, Beijing 100086, China
[3]LSBDPA Beijing Key Laboratory, Tsinghua University, Beijing 100084, China

Corresponding author: Feng Chen (chenfeng@mail.tsinghua.edu.cn)

This work was supported in part by the National Natural Science Foundation of China under Grant 61671266 and Grant 61836004, and in part by the Tsinghua University Initiative Scientific Research Program under Grant 20161080084.

ABSTRACT Foresight is a manifestation of human-level intelligence. In model-based reinforcement learning, obtaining a perfect environment model is essential and significant. Although recent research has achieved impressive advances in improving prediction accuracy, existing environment models are unable to generalize well across environments, mainly because appearance-independent dynamics learning completely relies on appearance-specific representation learning. In this paper, we propose a novel predictive model named transferable environment model (TEM) which disentangles state representation and dynamics. The disentanglement allows the model to deal with different observation distributions and meanwhile share a cross-environment latent dynamics. In a 3D visual platform, we show that our model has good generalization performance in target environments with very few data. Furthermore, the TEM is able to continually adapt to a sequence of target environments without forgetting the knowledge for previous environments. To the best of our knowledge, this paper is the first to endow a predictive model with the ability to work across multiple environments.

INDEX TERMS Environment model, model-based reinforcement learning, domain adaptation, disentangled representation.

I. INTRODUCTION

Foresight is a manifestation of human-level intelligence, the idea of which has been widely applied in various fields [1]–[3]. In *reinforcement learning* (RL) [4], the ability to imagine and evaluate the future evolution of an environment is significant for artificial intelligence agents to act in complex environments. Equipped with an environment model, also called predictive model or dynamics model, an agent is endowed with the predictive capability to simulate how the environment changes in response to their actions. Therefore, the agent with an environment model can act or plan more effectively as allowed to consider consequences of different actions without interacting in the real environments [5]. In the field of RL, the algorithms using dynamics models are called *model-based* RL approaches [6]. With the development of RL, research on predictive models has already been an important topic for agent control. Recently, significant progress [7]–[9] has been made in constructing accurate models, which can perform high prediction accuracy in complex environments such as Arcade Learning Environment [10] and Deepmind Lab [11].

Despite these advances in obtaining high prediction accuracy, existing environment models still suffer from limited *cross-environment adaptability* meaning that a well-trained model in an environment (the source) can predict in other unseen environments (the target). Due to over-optimization in a single environment, previous models are very sensitive to the changes of the environment, even minor ones like appearance or components. An ideal model could generalize to different environments rather than overfitting to a single environment; and could learn a common dynamics shared by similar environments (see Fig. 1). Moreover, re-training a well-performing environment model always requires numerous interactions between the agent and environment, which is often prohibitively difficult an expensive, especially in a real-world application. Thus, reusing a trained model is particularly crucial when training data are short for supply due

The associate editor coordinating the review of this manuscript and approving it for publication was Tossapon Boongoen.

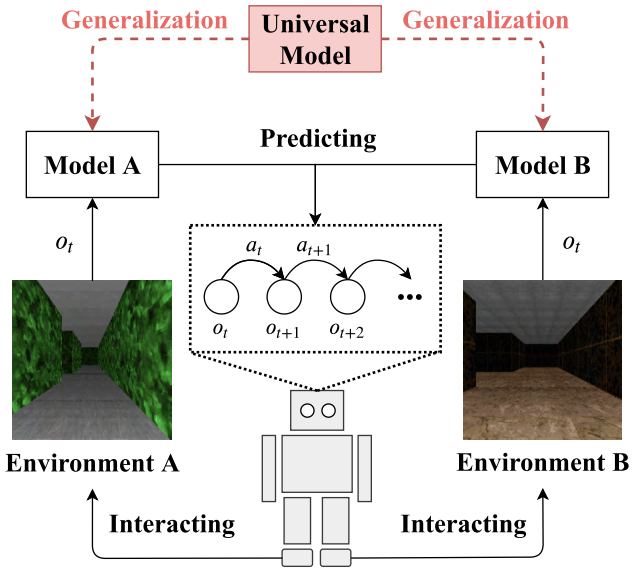


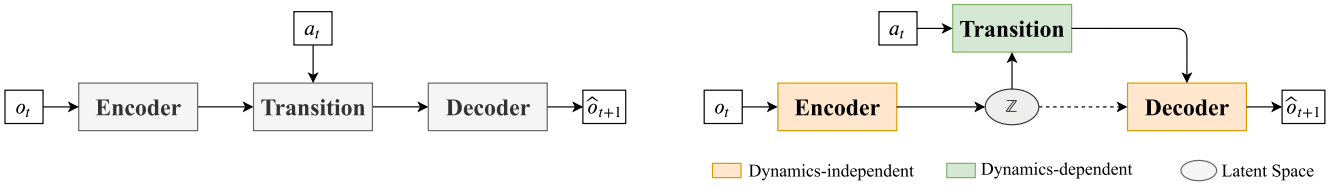
FIGURE 1. A model-based agent acts with dynamics models. Past models can only simulates a single environment. An ideal model (the universal model) should has the ability to simultaneously work in different environments.

to the interaction difficulty. In this paper, we focus on developing a more universal predictive model that can generalize across different environments. Hereinafter, the environment for first training is called the *source environment* and novel environments are called the *target environments*.

A typical model contains two basic parts (see Fig. 2(a)): (i) static *state representation* for encoding observations and generating future predictions, (ii) dynamic *state transition* for abstract state transition from the current time-step to the next by integrating action, which represents intrinsic dynamics. In the classic framework, the appearance-independent dynamics learning completely relies on the appearance-specific representation learning, resulting in the coupling of the transition module and representation module. In consequence, a past model always fails to dealing with the changes of the environment appearance. We propose that the key limitation to generalization of environment models is the entanglement of representation and transition. As is well known, humans can easily handle variety of environments because the advanced human behavior is based on environment-independent semantic concepts rather than raw perceptual inputs. Inspired by this, we argue that an environme‘nt model should disentangle representations with dynamics and high-level predictions are made upon the abstract representations (see Fig. 2(b)). Considering that there exists common intrinsic dynamics between the source and target environments, a disentangled framework also allows a model to share a common transition function and only adapt the representation module to inputs from the target environment. Hence the learned knowledge of internal dynamics can be reused across environments ignoring the variance of the appearance.

In this paper, we propose a novel environment model named *Transferable Environment Model* (TEM) with disentangled state representation and transition modules, named *abstractor* and *predictor* respectively. The abstractor is an autoencoder, which projects the raw (pixel) observations to a high-level feature space. The predictor is formed with the action-conditional architecture [8], which estimates the future features based on the latent space. In a single environment (the source environment), we have access to a model via learning the abstractor and predictor in two phases. Specially, in view of that the disentangled learning can lead to the convergence to a local optimal solution, we propose *joint prediction error* as a loss function to overcome this unfavorable situation. For generalizing the source model to a target environment, we freeze the predictor and only adapt the abstractor to the distribution of target observations. By introducing the cycle consistence loss [12], we map the target observations to the latent space learned in the source environment. Utilizing the source latent space as a common space, we can ensure the consistence of the inputting space for sharing predictor and we can also ensure the adapted model work for the source environment.. A versatile abstractor and a shared predictor makes up our model which is endowed with the cross-environment generalizability.

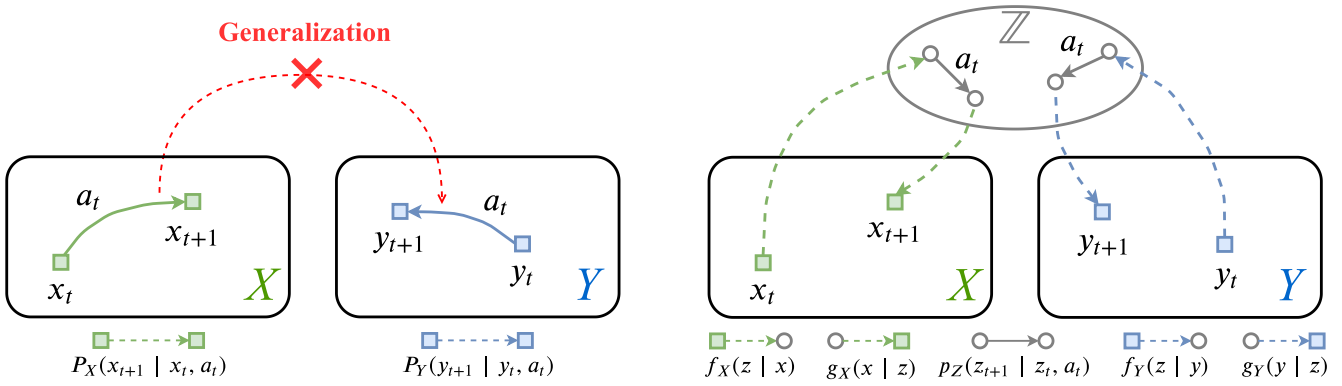
To our best knowledge, this is the first for environment models to generalize across environments. In order to demonstrate the capacity of TEM, we conducted our experiments on the *Vizdoom Domain* [13] that provides sufficient maze-like 3D environments. Predicting in a single environment (the source environment), TEM achieved the close performance to the state-of-the-art model. Next, the source TEM can adapt to the target environments by only one observation without any interaction data, called *zero-interaction adaptation*.



(a) Classic framework

(b) Disentangled framework

FIGURE 2. The frameworks comparison. (a) The classic architecture of the state-of-the-art model (Oh Model) couples latent dynamics (transition module) with state representation (encoder-decoder). (b) We propose the disentangled-dynamics architecture of TEM. The representation module is independent of the transition module.



(a) Classic framework (b) Disentangled framework

FIGURE 3. Cross-environment transfer comparison. (a) Oh Model is overfitting in the source environment X and unable to straightly transferred to a target environment Y. (b) The disentangled model learns a common feature space $\mathcal{Z}$ to connect the observation spaces of X and Y.

Meanwhile, the performance of the adapted TEM rivals that of Oh-model trained with full transition data. Furthermore, TES can continually adapt across a sequence of environments without forgetting, called *continual prediction*. Here, zero-interaction adaptation and continual prediction are the abilities that previous models do not possess.

II. DISENTANGLED PREDICTION FRAMEWORK

The goal of an environment model is to estimate a state transition function $P(o_{t+1}|o_t, a_t)$ predicting the next state $\hat{o}_{t+1}$ based on the current state o_t and selected action a_t. The prediction mapping can be expressed as $P : \mathcal{O} \times A \rightarrow \mathcal{O}$, where $\mathcal{O}$ denotes the observation space and A is the action space. As illustrated in Fig. 3(a), we learn $P_X : \mathcal{O}_X \times A \rightarrow \mathcal{O}_X$ in the source environment X. Since coupling of $\mathcal{O}$ and A increases the complexity of input space, it exists inestimable difficulties to reuse P_X into the learning of $P_Y : \mathcal{O}_Y \times A \rightarrow \mathcal{O}_Y$ for the target environment Y. We must retrain a target model as we did in X.

The theory of disentangled representation [14]–[16] focuses on learning underlying representations of the world, which are not task or domain specific [17]. In this paper, we introduce the idea of disentangled representation to our model. Inspired by this, we consider to disentangle the state representation with the state transition, which is necessary for finding common state representations to connect the observation spaces of source and target environments. Mathematically, we divide the original prediction function P: $\mathcal{O} \times \mathcal{A} \longrightarrow \mathcal{O}$ into (i) state representation function $\mathcal{F} : \mathcal{O} \rightarrow \mathcal{Z} \rightarrow \mathcal{O}$ and (ii) latent transition function $P_{\mathcal{Z}}$: $\mathcal{Z} \times \mathcal{A} \longrightarrow \mathcal{Z}$, where $\mathcal{Z}$ is a latent feature space. In this section, we will first construct and learn TEM in a single environment.

A. DISENTANGLED PREDICTION

Under the disentangled framework, the full prediction function is composed of separate state representation and latent transition. Then TEM is correspondingly divided into two modules respectively named *abstractor* and *predictor*.

1) ABSTRACTOR

The front module for state representation is formed with a variational autoencoder [18] that has the ability to extract features and reconstruct inputs via unsupervised learning. Inputting a raw observation o, abstractor encodes o into a d-dimensional feature $z \in \mathbb{R}^d$ and then translates z to a reconstructed observation $\hat{o}$. This module works for mapping the raw (pixel) observations of the observation space $\mathcal{O}$ to the latent feature space $\mathcal{Z}$ and then returning to $\mathcal{O}$. Here the encoding mapping $f : \mathcal{O} \rightarrow \mathcal{Z}$ and decoding $g : \mathcal{Z} \rightarrow \mathcal{O}$ as described above are respectively denoted by $f_\phi(z|o)$ and $g_\theta(o|z)$.

2) PREDICTOR

This internal module $p_\omega(z_{t+1}|z_t, a_t)$ estimates the next observation relying on the encoded feature z_t of current observation by the abstractor. When selecting an action represented using a one-hot vector a_t, the predictor aggregates the input feature z_t and a_t to obtain a predictive representation $\hat{z}_{t+1}$ of the next state:

$$\hat{z}_{t+1} \sim p_\omega(z_{t+1}|z_t, a_t). \tag{1}$$

where the aggregation of z_t and a_t is the element-wise vector multiplication.

3) COMPLETE PREDICTION

At time step t, given the input observation o_t and action vector a_t, TEM predicts the next observation o_{t+1} in three steps. First, abstractor compressed the current o_t into low-dimensional $z_t \sim f_\phi(z|o_t)$. Then predictor integrates the abstracted representation z_t with one-hot action vector a_t to predict the next latent representation $\hat{z}_{t+1}$ with (1). Finally, abstractor decodes the predictive coding $\hat{z}_{t+1}$ to generate the estimating observation $\hat{o}_{t+1} \sim g_\theta(o|\hat{z}_{t+1})$ at the time

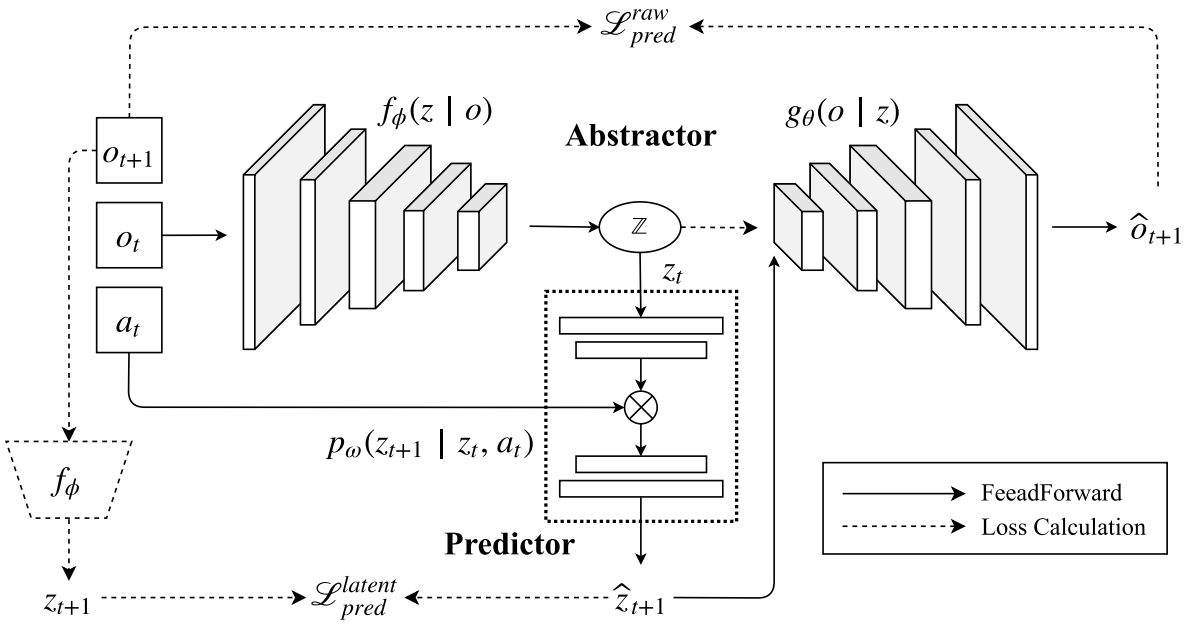


FIGURE 4. The disentangled architecture of TEM. Under the framework, the abstractor maps o_t to a feature space $\mathcal{Z}$ and then the predictor reasons about the future feature $\hat{z}_{t+1}$ based on (z_t, a_t). The latent prediction error $\mathcal{L}_{pred}^{latent}$ and raw prediction error $\mathcal{L}_{pred}^{raw}$ compose the joint prediction loss.

step $t+1$. The full prediction function from o_t, a_t to o_{t+1} can be expressed as:

$$P(o_{t+1}|o_t, a_t) = f_\phi(z_t|o_t) p_\omega(z_{t+1}|z_t, a_t) g_\theta(o_{t+1}|z_{t+1}), \quad (2)$$

where the abstractor $f_\phi(\cdot)g_\theta(\cdot)$ is independent of the predictor $p_\omega(\cdot)$.

B. PREDICTION LEARNING

We train TEM in two phases. To train the model, we sample transition data from experiences of an agent interacting with the environment. A transition (o_t, a_t, o_{t+1}) at time step t is composed of the current observation o_t, action a_t and next observation o_{t+1}. The transition dataset is denoted by $\mathcal{D}_\mathcal{T} = \{(o_0, a_0, o_1), (o_1, a_1, o_2), \cdots, (o_{n-1}, a_{n-1}, o_n)\}_{n=1}^N$.

Phase I (State Representation Learning): In the first phase, we train the abstractor independent of the predictor. We employ the observation dataset $\mathcal{D}_\mathcal{O}$ as training set. $\mathcal{D}_\mathcal{O} = \{o_1, o_2, \cdots, o_M\}$, only containing the observations, is a subset separated from $\mathcal{D}_\mathcal{T}$. Using the autoencoder, we construct a variational bound [18] on the data log-likelihood for the abstractor. Given an observation point $o \in \mathcal{D}_\mathcal{O}$, the variational bound is expressed as:

$$\begin{aligned}
\log p(o) &= \log \int p_z(z) p_\theta(o|z) dz \\
&= \log \int f_\phi(z|o) \frac{p_z(z) g_\theta(o|z)}{f_\phi(z|o)} dz \\
&\geq \int f_\phi(z|o) \left[\log g_\theta(o|z) - \log \frac{q_\phi(z|s)}{p_z(z)} \right] dz \\
&= \mathbb{E}_{f_\phi(z|o)} \left[\log g_\theta(o|z) \right] - D_{KL} \left(f_\phi(z|o) || p_z(z) \right), \quad (3)
\end{aligned}$$

where $p_z(z)$ is the prior assumed as $\mathcal{N}(0, I)$. The first term is the expected reconstruction error, which is formed with the mean square error. The second KL-divergence term is a regularizer to avoid the overfitting in the finite dataset by encouraging the encoded features close to the prior $p_z(z)$. Upon the training set $\mathcal{D}_\mathcal{O}$, we yield the loss function expressed as:

$$\begin{aligned}
&\mathcal{L}_{abs}(\phi, \theta; \mathcal{D}_\mathcal{O}) \\
&\quad = \frac{1}{|\mathcal{D}_\mathcal{O}|} \sum_{o \in \mathcal{D}_\mathcal{O}} \|\hat{o} - o\|_2 + D_{KL} \left(f_\phi(z|o) || p_z(z) \right). \quad (4)
\end{aligned}$$

The abstractor is optimized by minimizing (4) with gradient ascent methods.

Phase II (Latent Dynamics Learning): Given the transition dataset $\mathcal{D}_\mathcal{T}$ and a trained abstractor $f_{\phi^*}(z|o)g_{\theta^*}(o|z)$, we aim to optimize the internal module predictor in the second phase. With the observation o_t and a_t at time step t, the goal of learning the predictor is to obtain the predicted feature $\hat{z}_{t+1}$ close to the groundtruth z_{t+1} representing the future state o_{t+1}. An intuitive approach is to directly minimize the distance between the features $\hat{z}_{t+1}$ and z_{t+1}. However, only focusing on the feature distance will lead the parameters of predictor to converging at a local optimum. We are more interested in obtaining accurate observation-level outputs. Thus training the predictor module, we introduce a *joint prediction error* as the loss function, composed of (i) the feature prediction error $\mathcal{L}_{pred}^{latent}$ and (ii) the observation prediction error $\mathcal{L}_{pred}^{raw}$. The joint prediction error focuses on reducing differences between the predicted outputs and the groundtruth in both $\mathcal{Z}$ and $\mathcal{O}$.

We first calculate the feature prediction error. For each data point $(o_t, o_t, o_{t+1}) \in \mathcal{D}_\mathcal{T}$, we obtain the predicted feature

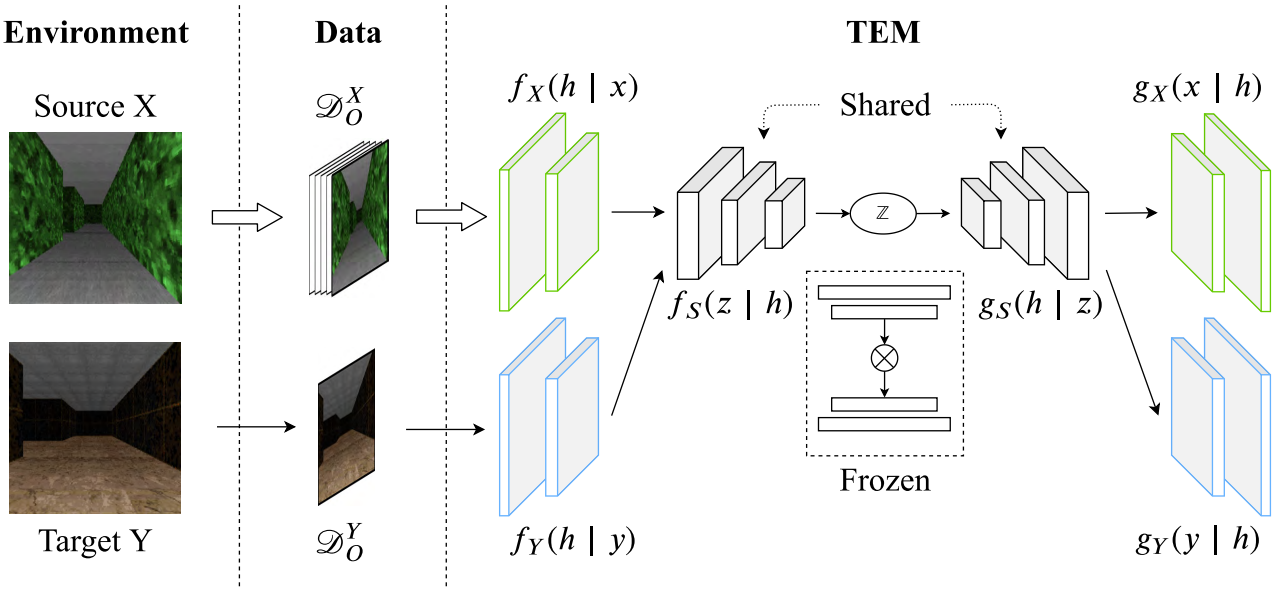


FIGURE 5. Cross-environment adaptation of TEM. In the source model, we extend the abstractor to a paired-stream architecture and frozen the predictor. The target observations are mapped to the feature space $\mathcal{Z}$ by sharing the hidden encoder and decoder. Note that the adaptation to Y can only use one observation data.

$\hat{z}_{t+1}$ and compare it with the groundtruth z_{t+1}. For the whole dataset, the feature predictive error can be expressed as:

$$\mathcal{L}_{pred}^{latent}(\omega, \overline{\phi^*}; \mathcal{D}_\mathcal{T}) = \frac{1}{|\mathcal{D}_\mathcal{T}|} \sum_{(o_t, a_t, o_{t+1}) \in \mathcal{D}_\mathcal{T}} \|\hat{z}_{t+1} - z_{t+1}\|_2, \tag{5}$$

where $\hat{z}_{t+1} \sim f_{\phi^*}(z_t|o_t)p_\omega(z_{t+1}|z_t, a_t)$ and $z_{t+1} \sim f_{\phi^*}(z_{t+1}|o_{t+1})$. The bar (such as $\overline{\phi^*}$) represents that the optimal parameters will not be updated during the module learning. In addition, we introduce the pixel-pixel error to improve the pixel-level accuracy of the final outputs. Then the observation prediction error is formed by the mean distance between real and decoded observations, expressed as:

$$\mathcal{L}_{pred}^{raw}(\omega, \overline{\phi^*}, \overline{\theta^*}; \mathcal{D}_\mathcal{T}) = \frac{1}{|\mathcal{D}_\mathcal{T}|} \sum_{(o_t, a_t, o_{t+1}) \in \mathcal{D}_\mathcal{T}} \|\hat{o}_{t+1} - o_{t+1}\|_2, \tag{6}$$

where the predicted observation $\hat{o}_{t+1} \sim g_{\theta^*}(o_{t+1}|\hat{z}_{t+1})$ is decoded with the predicted feature $\hat{z}_{t+1}$ by the abstractor module.

In summary, the joint prediction error is expressed as:

$$\mathcal{L}_{pred}(\omega, \overline{\phi^*}, \overline{\theta^*}; \mathcal{D}_\mathcal{T}) = \mathcal{L}_{pred}^{latent}(\omega, \overline{\phi^*}) + \beta \mathcal{L}_{pred}^{raw}(\omega, \overline{\phi^*}, \overline{\theta^*}), \tag{7}$$

where β is a trade-off parameter. Through minimizing (7), we can obtain the optimal predictor.

With the abstractor and predictor, we have access to a basic TEM for predicting in a single environment. In the next section, we will introduce how to adapt the basic model to other novel environments.

III. CROSS-ENVIRONMENT PREDICTION

With a basic model, we can acquire the dynamics mapping $P_X: \mathcal{O}_X \times A \to \mathcal{O}_X$ for the source environment X. Focusing on the cross-environment transfer to a novel environment Y, we attempt to acquire a target prediction mapping $P_Y: \mathcal{O}_Y \times A \to \mathcal{O}_Y$ utilizing the learned mapping P_X. The theory of transfer learning [19] defined that the source and target domains are *related*. Here, we share a common latent space $\mathcal{Z}$ and a common latent predictor $P_\mathcal{Z}$ across environments. In many cases, such as navigation tasks in different rooms, there exist physical laws between the source and target environments which we name as *related environments*. It is natural for us to consider sharing common high-level representations and latent dynamics for cross-environment prediction.

In this section, we illustrate how to transfer the learned P_X (source model) from X to Y. Furthermore, we expound the continual adaptability of TEM during adapting to a sequence of environments.

A. ADAPTATION ACROSS TWO ENVIRONMENTS

In our cross-environment prediction task, we convert the problem from dynamics adaptation to observation adaptation. As shown in Fig. 3(b), the learning of P_X has been done in two steps: (i) we first learn the representation mapping $\mathcal{F}_X: \mathcal{O}_X \to \mathcal{Z} \to \mathcal{O}_X$ (the abstractor module) and meanwhile obtain a latent feature space $\mathcal{Z}$ corresponding to low-dimensional representations; (ii) based on the latent space $\mathcal{Z}$, we learn the latent prediction mapping $P_\mathcal{Z}: \mathcal{Z} \times A \to \mathcal{Z}$ (the predictor) representing the high-level dynamics. We respectively set $\mathcal{Z}$ and $P_\mathcal{Z}$ as common latent feature space and latent transition function. Then we can adapt the source model to the target environment Y only by learning a new

representation mapping $\mathcal{F}_Y$: $\mathcal{O}_Y \rightarrow \mathcal{Z} \rightarrow \mathcal{O}_Y$ which connects $\mathcal{O}_Y$ and $\mathcal{Z}$. Combing $\mathcal{F}_Y$ with the constant $P_{\mathcal{Z}}$, we obtain a full predictive model of Y with disentangled prediction function P_Y: $(\mathcal{O}_Y \rightarrow \mathcal{Z}) \times A \rightarrow (\mathcal{Z} \rightarrow \mathcal{O}_Y)$. The adaptation does not need the agent to interacting with the target environment, which is called zero-interaction adaptation.

The source model $f_X \cdot p_Z \cdot g_X$ is extended to a *paired-stream* architecture as shown and described in Fig. 5, where a target stream is introduced into abstraction module of the source model and the predictor p_Z is kept constant. In the source stream, the abstractor is divided into four parts: individual encoder f_X, common encoder f_S, common decoder g_S and individual decoder g_X, of which parameters are denoted by $\phi_X, \phi_S, \theta_S, \theta_X$ Here individual parts deal with the external observation space and shared parts link the common feature space. For pulling the observations of Y in, we add individual encoder f_Y and decoder g_Y cloned from the source stream. For clarity, we denote the full encoder and decoder of the source path with $F_X = f_X(h|o)f_S(z|h)$ and $G_X = g_S(h|z)g_X(o|h)$. Similarly, $F_Y = f_Y(h|o)f_S(z|h)$ and $G_Y = g_S(h|z)g_Y(o|h)$ in the target path.

Our goal is to capture common representations shared by X and Y. Feed an observation $o_y \in \mathcal{O}_Y$ into the target stream, then target encoder generates a latent representation z_y. We aim to enforce z_y satisfying the latent feature distribution of X as much as possible. Here we introduce the *Cycle Consistency Loss* (CC loss) referring to CycleGAN [12]. Given a target observation $y \in \mathcal{O}_Y$, a translation from the target to the source is denoted by $Y \rightarrow X$ which means that y is first encoded by F_Y and then decoded into the source style by G_X. We use $T_{YX} = F_Y \cdot G_X$ and $T_{XY} = F_X \cdot G_Y$ to denote the translation $Y \rightarrow X$ and $X \rightarrow Y$ respectively. A cycle $Y \rightarrow X \rightarrow Y$ means the cycle translation $y \rightarrow T_{YX}(y) \rightarrow T_{XY}(T_{YX}(y))$. Through reducing the distance between y and $T_{XY}(T_{YX}(y))$, we enforce the feature z_y having common attributes with the feature $z_x \in \mathcal{Z}$, where $\mathcal{Z}$ is the latent space earlier learned by the source model. Sampling the observation set $\mathcal{D}_{\mathcal{O}}^Y$ from Y, we construct the cycle consistency loss as:

$$\mathcal{L}_{cycle}\left(\Theta_X, \Theta_Y, \overline{\Theta_S}; \mathcal{D}_{\mathcal{O}}^Y\right) = \frac{1}{|\mathcal{D}_{\mathcal{O}}^Y|} \sum_{y \in \mathcal{D}_{\mathcal{O}}^Y} \|T_{XY}(T_{YX}(y)) - y\|_2, \quad (8)$$

where shared and individual parameters are denoted by $\Theta_S = (\phi_S, \theta_S)$, $\Theta_X = (\phi_X, \theta_X)$ and $\Theta_Y = (\phi_Y, \theta_Y)$ respectively.

The cycle consistency loss mainly focuses on enforcing the target representations z_y to obtain common attributes with the source representations $z_X \in \mathcal{Z}_X$. In addition, we also expect that z_y fits the individual distributions in Y. Thus we add the reconstruction loss as the constraint to avoid the representations deviating the original observation distribution in the target Y. The reconstruction loss $\mathcal{L}_{recons}^Y$ is formed with (4) using $\mathcal{D}_{\mathcal{O}}^Y$, expressed as:

$$\mathcal{L}_{recons}^Y\left(\Theta_Y, \overline{\Theta_S}; \mathcal{D}_{\mathcal{O}}^Y\right) = \mathcal{L}_{abs}^Y\left(\mathcal{D}_{\mathcal{O}}^Y\right) \quad (9)$$

Considering that the learned predictor is based on the latent space $\mathcal{Z}$ in the source environment, we should keep $\mathcal{Z}$ still maintaining the source attributes to make the predictor effective. In this purpose, the parameters of shared networks will not be updated by the data from Y. Noting that the bars exists on Θ_S, it indicates that f_S and g_S are detached during backpropagation of loss (8) and (9). And we introduce the reconstruction loss $\mathcal{L}_{recons}^X$ for X to maintain the source mapping $\mathcal{F}_X$: $\mathcal{O}_X \rightarrow \mathcal{Z} \rightarrow \mathcal{O}_X$:

$$\mathcal{L}_{recons}^X\left(\Theta_X, \Theta_S; \mathcal{D}_{\mathcal{O}}^X\right) = \mathcal{L}_{abs}^X\left(\mathcal{D}_{\mathcal{O}}^X\right) \quad (10)$$

Algorithm 1 Cross-Environment Prediction

Input:
Source dataset $\mathcal{D}_{\mathcal{O}}^X = \{o_1^X, o_2^X, \cdots, o_M^X\}$
Target dataset $\mathcal{D}_{\mathcal{O}}^Y = \{o_0^Y\}$
Paired-path abstractor parameters $\Theta = \Theta_S \cup \Theta_X \cup \Theta_Y$.
Frozen predictor parameters ω.
`// Initialization`
Initialize parameters $\Theta_{\mathcal{Y}}$ and batch size N_b;
`// Transfer learning`
while *not converged* **do**
  for *iteration step* $i = 1, \cdots, \left[\frac{M}{N_b}\right]$ **do**
    Sample a batch $\mathcal{B}_i \in \mathcal{D}_{\mathcal{O}}^X$;
    Feed $\mathcal{B}_i$ and $\mathcal{D}_{\mathcal{O}}^Y$ into two-path abstractor;
    Calculate $\mathcal{L}_{transfer}\left(\Theta; \mathcal{B}_i, \mathcal{D}_{\mathcal{O}}^Y\right)$ from (11) ;
    Update $\Theta \leftarrow \Theta - \alpha_t \nabla \mathcal{L}_{transfer}(\Theta)$;
  end
end
`// Module Aggregation`
Aggregate Θ of extended abstractor and ω of predictor
Obtain a cross-environment predictive model with (Θ, ω)

Above all, the full loss function of adaptation from X to Y is:

$$\mathcal{L}_{transfer}\left(\Theta_X, \Theta_Y, \Theta_S; \mathcal{D}_{\mathcal{X}}, \mathcal{D}_{\mathcal{Y}}\right) = \mathcal{L}_{cycle} + \lambda_Y \mathcal{L}_{recons}^X + \lambda_X \mathcal{L}_{recons}^Y \quad (11)$$

where λ_X, λ_Y are a trade-off parameter. By minimizing (11), we can obtain the abstractor adaptive to both source X and target Y. Combined the abstractor and predictor, a full prediction function for the target environment can be available:

$$P_Y(o_{t+1}|o_t, o_t) = f_Y(h_t|o_t)f_S(z_t|h_t) \cdot p_{\mathcal{Z}}(z_{t+1}|z_t, a_t)g_S(h_{t+1}|z_{t+1})g_Y(o_{t+1}|h_{t+1}). \quad (12)$$

The process of cross-environment adaptation for TEM is described in Algorithm. 1. Referring to [20], we can use only one observation to achieve adaptation of the abstractor as shown in Algorithm. 1.

B. CONTINUAL PREDICTION ACROSS MULTIPLE ENVIRONMENTS

Continual (lifelong) learning is the ability to learn consecutive tasks without forgetting how to perform previously trained tasks, considered as an important capacity for artificial general intelligence [21]. In this paper, we define *continual adaptation* for predictive models as the ability to continually adapt to a novel environment without forgetting the dynamics learned in the previous environments.

Algorithm 2 Continual Prediction

Input:
Source observation dataset $\mathcal{D}_{\mathcal{O}}^{X} = \{o_1, o_2, \cdots, od_M\}$;
Target environments set $\mathbb{Y} = \{Y_1, Y_2, \cdots, Y_n\}$;
Parameters set $\Theta = \{\Theta_S, \Theta_X\}$;
// Initialization
Initialize learning rate α_t ;
Clone (Θ_X, Θ_S) into temporal parameters (Θ'_X, Θ'_S) ;
Clone Θ_X into Θ_Y and randomly initialize Θ_Y;
// Continual adaption process
for $Y = Y_1, Y_2, \cdots, Y_n \in \mathbb{Y}$ **do**
  Randomly initialize Θ_Y ;
  Aggregate Θ_Y, Θ_X and Θ_S into paired-path model ;
  Adapt the source model to Y with Algorithm. 1 ;
  // Parameters storage
  Put individual Θ_Y into $\Theta \leftarrow \Theta \cup \{\Theta_Y\}$;
  // Reused in the previous
  if *used in* $Y_i \leq Y$ **then**
    Combine Θ_{Y_i} and Θ'_S to form model $P_{\Theta_{Y_i}\Theta'_S}$;
    Make predictions in $\mathcal{Y}_t$ using $P_{\Theta_{\mathcal{Y}_t}\Theta'_S}$.
  end
end

We assume such a scenario that a model covers a sequence of different environments $Y_0, Y_1, Y_2, \ldots Y_n\}$. The environment sequence can be in any order, and the first Y_0 is set as the source environment X. At the first task step, we can learn a predictive model in X (the source environment). We denote the shared modules as $P_h^S = f_S(z|h)p_z(z|z,a)g_f(h|z)$ and storage all model parameters Θ_X, Θ_S in a parameter set $\Theta = \{\Theta_X, \Theta_S\}$. When encountering the first target environment Y_1, the source model can adapt to Y_1 with Algorithm. 1. Then we get individual encoder f_{Y_1} and decoder g_{Y_1} to form a full predictive model $f_{Y_1} \cdot P_h^S \cdot g_{Y_1}$ working for Y_1. We storage individual parameters $\Theta_{Y_1} = (\phi_{Y_1}, \theta_{Y_1})$ in $\Theta = \Theta \cup \{\Theta_{Y_1}\}$ where the size of Θ_{Y_1} is small. Since shared P_h^S only depends on the observation distribution of X, transferring the source model to others will not affect the parameters of shared P_h^S. Even if the source model is transferred to Y_2, it can still be effective to Y_1 only by aggregating Θ_{Y_1} with Θ_S. Therefore, the model can sequentially adapt to novel environments and meanwhile retain the learned dynamics function in a small memory cost. The continual prediction learning is illustrated in Fig. 6 and described in Algorithm. 2. So the prediction across two environments can be generalize to multiple environments only using one model.

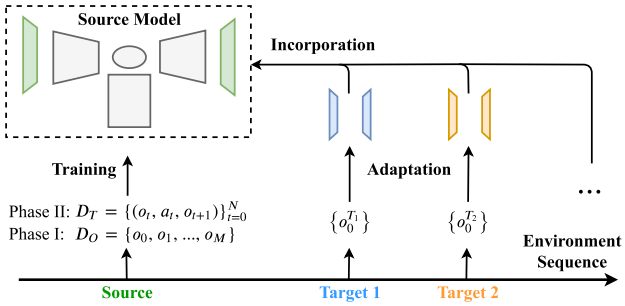


FIGURE 6. Continual prediction of TEM.

IV. EXPERIMENTS

Our experiments are designed to (i) evaluate the performance on prediction for an environment, (ii) verify the ability of cross-environment prediction and (iii) demonstrate the ability of continual prediction without forgetting. We compare our model with a baseline, the state-of-the-art model of Oh. In the following, we test TEM and Oh Model in the source environment to predict on testing dataset. Then we focus on cross-environment prediction. We succeed in transferring the source TEM to multiple target environments with only one observation while Oh Model failed. And the prediction performance of transferred TEM is close to the Oh Model trained with full transition data. Furthermore, we use only one model to adapt to a sequence of environments, in which TEM can continually adapt to them and remember the previous environments but Oh Model will lose its earlier memory.

A. ENVIRONMENT SETTINGS

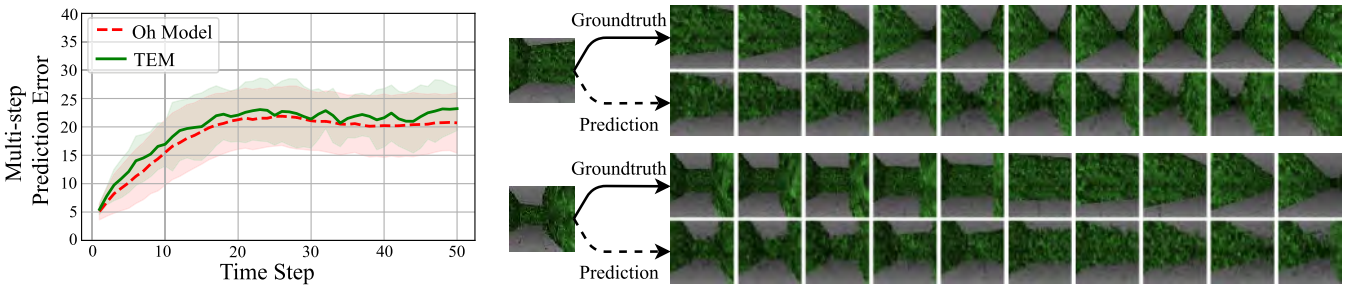
In our experiment setting, multiple environments are required. We conduct our experiments on the ViZDoom, an AI research platform which employs the first-person perspective in a semi-realistic 3D world [22]. We generate variant environments with the *Slade* [13], a map editor dedicated to Vizdoom. In Slade, we can design the styles or structures of environments, and also add different components. In our experiments, we sample four environments composed of a source environment and three target environments, as shown in Fig. 7. We train both TEM and baseline in the source environment and then attempt to reuse the source models into the other target environments.

B. DATA AND PREPROCESSING

The dataset $\mathcal{D}_{\mathcal{T}}$ contains 5000 transition tuples $\{(o, a, o')_i\}_{i=1}^{5000}$ from the interactions between the agent and environments. The observation is resized as a 64×64 RGB image. We use an exploration-based A3C [23] agent, capable of covering the entire maze, to sample experiences as training data. In the source environment, the training of both TEM and Oh Model requires the whole transition set $\mathcal{D}_{\mathcal{T}}^{X}$. In each target environment, the dataset of TEM only contains

(a) Source environment (b) Target environment I (c) Target environment II (d) Target environment III

FIGURE 7. **The environments used in our experiment. The map structure and initial observation are respectively shown in the left and right of each sub-figure.**



(a) K-step prediction error ($K = 50$) (b) Visualization of predicted trajectories

FIGURE 8. **Prediction performance in the source environment. (a) TEM has almost the same prediction accuracy in a single environment. (b) We visualize the predicted trajectories of TEM in 10 time steps.**

one image captured from the initial observation o_0 of the agent. Yet, the target dataset $\mathcal{D}_{\mathcal{T}}^{Y_i}$ of Oh Model consists of the same number of transitions as the dataset for the source environment.

C. EVALUATION METHODS

In each of the environments built above, we collect several trajectories a testing set, expressed as $\{((o_0, a_0, o_1), \cdots, (o_{K-1}, a_{K-1}, o_K))^{(i)}\}_{i=1}^{N_t}$ which consists of N_t sampled trajectories with the length of K time steps. Given a sampled trajectory, predictive models output the corresponding predicted trajectory. The predicted trajectory is generated by *unrolling* [8] the model, which uses the predicted frame as an input for the next time-step. We evaluate the prediction performance with *quantitative* evaluation and *qualitative* evaluation. In quantitative evaluation, we use K-step prediction error, calculated with *mean square error* between the prediction and groundtruth. We plot the error-step curve for each model to compare our model with the baseline. In qualitative evaluation, we visualize the predicted trajectories into the sequences of images. In this way, one can more intuitively evaluate the prediction performance.

D. PREDICTION IN THE SOURCE ENVIRONMENT

In this part, we expect to demonstrate that our model still is able to make accurate predictions although the prediction of TEM is based on the high-level feature space. Then we evaluate the predictive capacity of TEM on prediction in a single environment (the source environment X). We train TEM and Oh Model with $\mathcal{D}_{\mathcal{T}}^{X}$ the source environment. Feeding testing set to the trained models, we obtain the predicted trajectories of both models in the source environment.

1) QUANTITATIVE EVALUATION

First, we evaluate by comparing the predicted trajectories with those of the baseline. In quantitative evaluation, the error-step curves ($K = 50$) of both models are plotted in Fig. 8(a). We can see that the predicted trajectories are almost exactly consistent with the baseline. As already mentioned above, the introduction of a disentangled structure loses pixel-level information of the predicted outputs. And our model is slightly worse than Oh Model in prediction accuracy, which can prove it. However, our goal is indeed to exchange the cross-environment predictive capability by sacrificing little model accuracy for a single environment. Even so, our model is still able to maintain predictive ability close to the state-of-the-art model.

2) QUALITATIVE EVALUATION

Second, we evaluate by visually comparing the prediction with the groundtruth in qualitative evaluation. For more intuitive illustration, we visualize the predicted trajectories and the groundtruth in two streams as shown in Fig. 8(b). As seen in the observation trajectories, we can find that the predicted outputs from TEM are realistic compared with the real trajectories. This also proves that the disentanglement has little adverse effect on prediction accuracy in a single environment.

E. ZERO-INTERACTION TRANSFER PERFORMANCE

The cross-environment transfer task from X to Y_i is defined as learning a model for the target environment Y_i by utilizing the trained source model. As entangling the observations and actions, the learning of Oh Model must depend on the interactions between the agent and target environments. However,

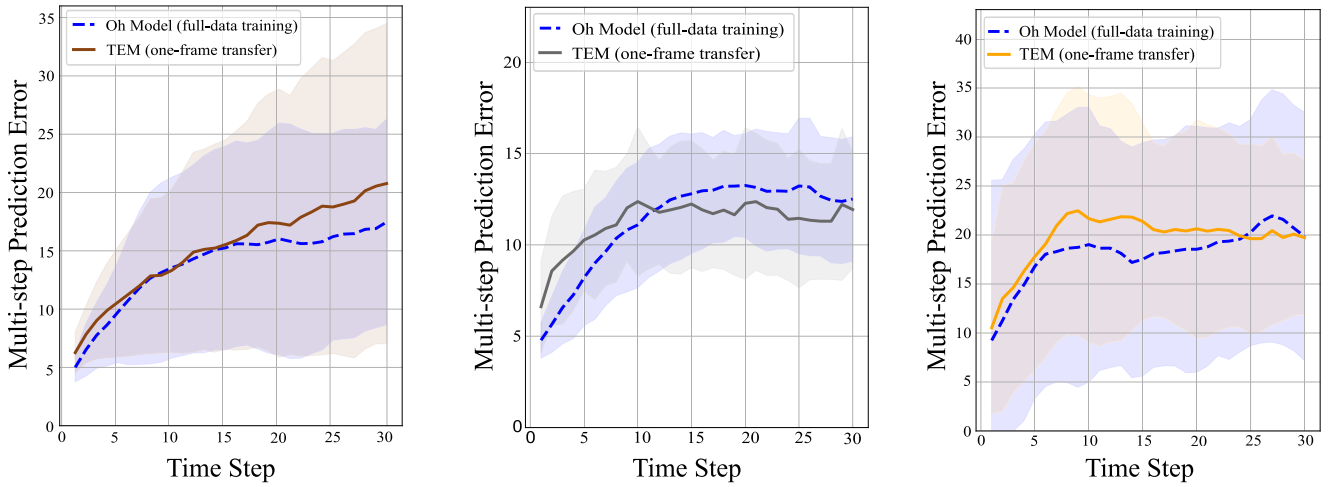


FIGURE 9. Cross-environment prediction performances. The TEM adapted by only an image has competitive performances with Oh Model trained by full transition data.

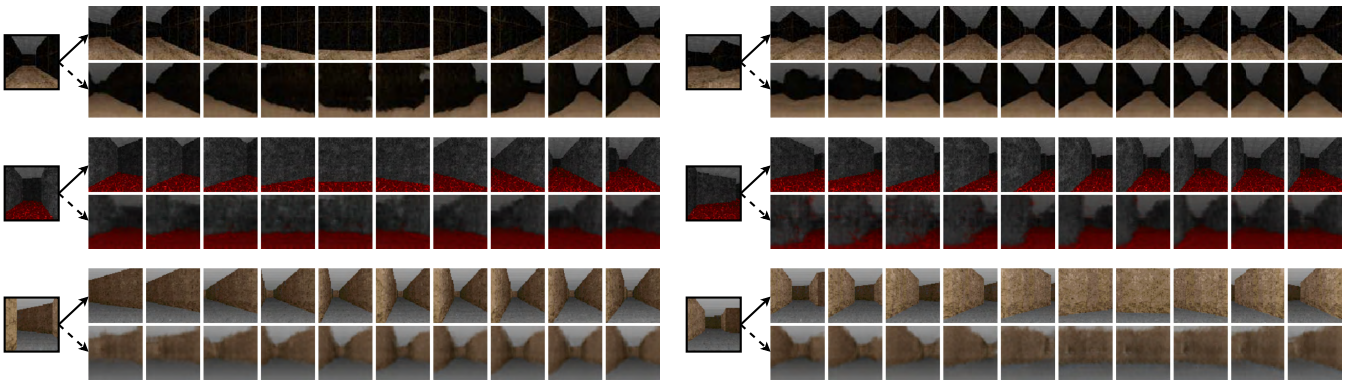


FIGURE 10. Visualization of cross-environment prediction. Given an input observation and determinate actions, a predicted trajectory can be visualized. The predicted trajectories (pointed by dashed arrows) from adapted TEM are almost consistent with the groundtruth trajectories (pointed by solid arrows) from the real platform.

in many cases, the access to interacting with the unknown environments is expensive. In the following, As mentioned in Section. III, TEM can adapt to a novel environment without trial-and-error interactions. We clone the source model to three copies and respectively transfer the three cloned models to target environments As described in Algorithm. 1, we obtain the TEM working for the given target environment with only one observation, which uses the initial inputting observation. In this part, we will test on the three target environments to demonstrate the unique ability of TEM. The results are shown in Fig. 9 and Fig. 10.

1) QUANTITATIVE EVALUATION

We compare the cross-environment prediction performance of TEM with that of Oh Model trained train with the full transition data. In Fig. 9, the models predict in 30 steps ($K = 30$). We can see that our model can achieve almost the same performance as the Oh Model. The adaption for TEM to each target environment only use one initial observation without any interactions. In the figure, TEM has close performance to Oh Model while Oh Model needs to be retained with lots of transition data sampled from the target environment. Note that the values of prediction error have large difference in different environments. That is in that the prediction error is the mean error of pixels and average pixel values are not completely close in different environments.

2) QUALITATIVE EVALUATION

We still evaluate the performance of TEM in target environments with output visualization. Similarly as we did in the source environment, we visualize the predicted trajectories of TEM and compare them with the groundtruth for all of the three target environments, as shown in Fig. 10. From the figure, the predicted images can reflect the real trends of the state changes in respond to the actions. Noise exists in the output images due to information loss during transfer process. But we can see that the basic structure and components are presented in the predicted observations. Therefore, the shared

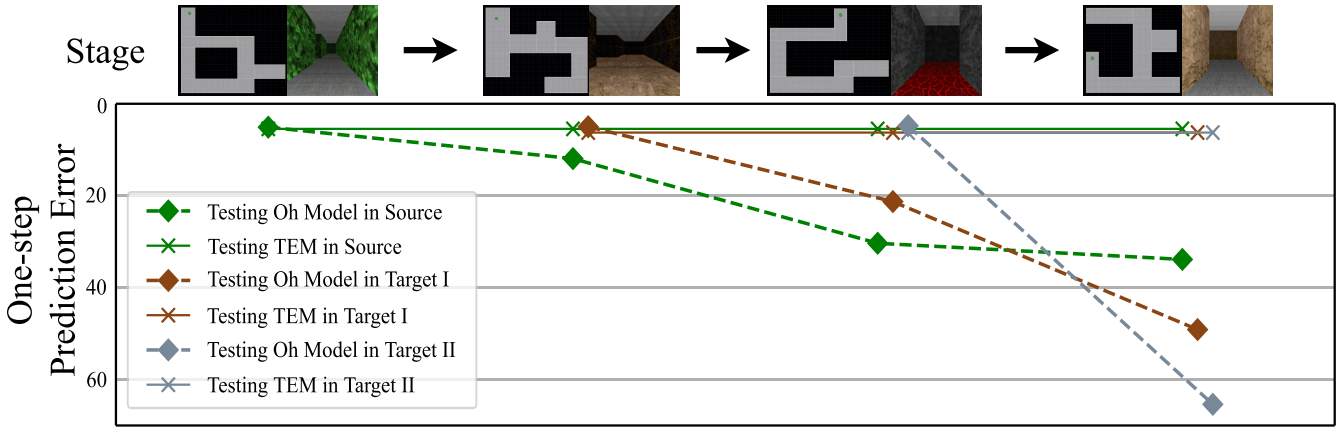


FIGURE 11. Continual prediction performance. We sequentially train Oh Model or adapt TEM in a sequence of environments. TEM in the previous environments still performs high accuracy while Oh Model gets worse.

feature space does reflect high-level representations of the observations in these related environments. And we also capture the common latent dynamics of the related environments by constructing the common predictor module.

F. CONTINUAL PREDICTION PERFORMANCE

Another advantage of TEM is that it can continually adapt to a sequence of environments without forgetting the knowledge learned in the previous. In general settings, a model-based agent is equipped with only one predictive model. When the agent needs to cope with multiple environments, it will be difficult for the agent to perform well in all of them considering the catastrophic forgetting [24]. In this experiment, we aim to verify that our model can remember the learned dynamics but Oh Model fails to.

1) TASK SETTING

We consider a task that we use only one network to simulate a sequence of environments for TEM or Oh Model. In this task, both TEM and Oh Model will cope with the environment sequence, which is set as: source $\rightarrow$ target I $\rightarrow$ target II $\rightarrow$ target III as shown in Fig. 7. First, we randomly initialize the parameters of both models. In the source environment, we train TEM and Oh Model on the full transition set $\mathcal{D}_{\mathcal{T}}^{X}$. Then we adapt TEM and fine-tune Oh Model from target environment I to target III. The initial models to simulate a novel environment are from its last environment. And having adapted to a novel environment, the model will be tested on the prediction performance in all of the previous environments.

2) EVALUATION AND ANALYSIS

We evaluate the continual prediction performance with one-step prediction error, shown in Fig. 11. The curves of TEM This proves that TEM can continually adapt to novel environments without performance degradation in the previous environments. In contrast, for Oh Model, the prediction error in the previous environments increases badly. It indicates that Oh Model will lose the old knowledge after learning new dynamics. We can find that our model is more versatile than Oh Model. Furthermore, the continual prediction performance of TEM also proves that our model learned a common feature space that fits these related environments and meanwhile the predictor dose learn the latent dynamics which is shared among them. Thus, TEM is also a general environment model which can predict across multiple environments simultaneously.

V. RELATED WORK

A. MODEL-BASED RL

Model-based RL aims to control agents by using a predictive model to reason about the future. Having a perfect environment model, model-based RL is a more straight and efficient approach compared with model-free RL [23], [25], [26]. Earlier, researchers have also demonstrated that RL can be accelerated if an environment model is learned and used for policy learning [6], [27]–[29]. In recent years, model-based RL approaches have shown impressive success. In current model-based methods, predictive models can be mainly used to (i) plan by imagining different potential trails [30]–[32]; (ii) improve data efficiency [33]–[35]; (iii) improve exploration efficiency in sparse-reward tasks [36], [37]. In this paper, we focus on another advantage of model-based RL in transfer problem. For the environment dynamics model is independent of the tasks, model-based RL methods are more data-efficient and flexible to handle multiple tasks [38], [39] in the certain environment compared with mode-free RL. However, the current model-based RL methods cannot do well on tasks across different environments since existing dynamics models have one-to-one relationships with environments.

B. ENVIRONMENT DYNAMICS MODELING

Model-based RL methods decompose the RL problems into two subproblems: how to learn a model and how to control agent with the model [30]. The research on how to construct

an accurate environment model is significant and essential for model-based Rl methods. Schmidhuber and Huber [40] first introduced the idea of building a model to simulate the environment for vision-based RL problems. More recently, Khansari-Zadeh and Billard [41] used Gaussian mixture models to learn more general dynamics model for robot control. Deep neural networks have recently enabled significant advances in simulating complex environments, allowing for models that consider high-dimensional visual inputs across a wide variety of domains [42]–[44]. In current stage, the model of Oh *et al.* [8] represents the state-of-the-art in this area, demonstrating high accuracy in deterministic and discrete-action environments. Chiappa *et al.* [9] built on the model of Oh and advanced in long-term prediction performance. The architecture proposed by Oh is still representing the state-of-the-art framework. However, the models mentioned above only fit in one environment and are unable to generalize across environments. In this paper, we focus on the generalization of predictive models. To our knowledge, our model is the first one to have the ability to generalize across multiple environments.

C. DOMAIN ADAPTION

The goal of domain adaptation (DA), a form of transfer learning, is to deal with changes to the input distribution [17]. It is widely applied in computer vision (CV) [45]–[48] and reinforcement learning (RL) [17], [49], [50], aiming at adapting a model trained in a source domain for using in a target domain [51]. Our work can be viewed as a bridge to connect both CV domain adaptation and RL domain adaptation. A major idea is to map the source and target domains into a shared feature space [52]–[54], which inspired us to construct a common latent space connecting different environments. Another line is to reduce the differences between the distributions of source and target domains by image-to-image translating techniques [12], [20], [55]. The previous works mainly deal with visual inputs while our model needs to response to numerous combinations of observations and actions containing temporal information. This is the first time to apply DA into the dynamics learning, which has more complex inputting spaces than common classification tasks.

VI. CONCLUSION

Addressing the issue of modeling transferable dynamics, we present a novel environment model named TEM with the ability to generalize across multiple environments. TEM is the first environment model to simulate in multiple environments. Different from the classic models, our model first learns a visual module for abstracting high-level state representations and then learn a latent predictor upon the feature space. The disentangled framework ensures different observation distributions to share a common feature space and latent predictor. For adapting to target environments, we demonstrate that our model only using few target observation images performs competitive results with the state-of-the-art model (Oh Model) trained by full target interaction data. In an advanced task for continual adaptation, TEM also outperforms Oh Model for looking back in the previous environments.

In this paper, we show that our model is a more general model in response to multi-environment dynamics learning. Since we mainly focus on the generalization performance, the problem of long-term prediction accuracy was not considered a lot. In future work, recurrent neural networks can be introduce into our framework for pursuing high accuracy in long-term prediction across environments.

REFERENCES

[1] J. K. O'Regan and A. Noë, "A sensorimotor account of vision and visual consciousness," *Behav. Brain Sci.*, vol. 24, no. 5, pp. 939–973, 2001.
[2] M. L. Littman and R. S. Sutton, "Predictive representations of state," in *Proc. Adv. Neural Inf. Process. Syst.*, 2002, pp. 1555–1561.
[3] Y. Niv, "Reinforcement learning in the brain," *J. Math. Psychol.*, vol. 53, no. 3, pp. 139–154, 2009.
[4] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*. Cambridge, MA, USA: MIT Press, 2018.
[5] K. Gregor, G. Papamakarios, F. Besse, L. Buesing, and T. Weber, "Temporal difference variational auto-encoder," 2018, *arXiv:1806.03107*. [Online]. Available: https://arxiv.org/abs/1806.03107
[6] L. Kuvayev and R. S. Sutton, "Model-based reinforcement learning with an approximate, learned model," in *Proc. 9th Yale Workshop Adapt. Learn. Syst.*, 1996, pp. 101–105.
[7] I. Lenz, R. A. Knepper, and A. Saxena, "DeepMPC: Learning deep latent features for model predictive control," in *Proc. Robot., Sci. Syst.*, 2015, pp. 1–9.
[8] J. Oh, X. Guo, H. Lee, R. Lewis, and S. Singh, "Action-conditional video prediction using deep networks in Atari games," in *Proc. Adv. Neural Inf. Process. Syst.*, 2015, pp. 2863–2871.
[9] S. Chiappa, S. Racaniere, D. Wierstra, and S. Mohamed, "Recurrent environment simulators," 2017, *arXiv:1704.02254*. [Online]. Available: https://arxiv.org/abs/1704.02254
[10] M. G. Bellemare, Y. Naddaf, J. Veness, and M. Bowling, "The arcade learning environment: An evaluation platform for general agents," *J. Artif. Intell. Res.*, vol. 47, pp. 253–279, Jun. 2013.
[11] C. Beattie *et al.*, "DeepMind lab," 2016, *arXiv:1612.03801*. [Online]. Available: https://arxiv.org/abs/1612.03801
[12] J.-Y. Zhu, T. Park, P. Isola, and A. A. Efros, "Unpaired image-to-image translation using cycle-consistent adversarial networks," in *Proc. IEEE Int. Conf. Comput. Vis.*, Jul. 2017, pp. 2223–2232.
[13] M. Kempka, M. Wydmuch, G. Runc, J. Toczek, and W. Jaśkowski, "ViZDoom: A doom-based AI research platform for visual reinforcement learning," in *Proc. IEEE Conf. Comput. Intell. Games (CIG)*, Sep. 2016, pp. 1–8.
[14] J. Schmidhuber, "Learning factorial codes by predictability minimization," *Neural Comput.*, vol. 4, no. 6, pp. 863–879, 1992.
[15] X. Chen, Y. Duan, R. Houthooft, J. Schulman, I. Sutskever, and P. Abbeel, "InfoGAN: Interpretable representation learning by information maximizing generative adversarial nets," in *Proc. Adv. Neural Inf. Process. Syst.*, 2016, pp. 2172–2180.
[16] M. F. Mathieu, J. J. Zhao, J. Zhao, A. Ramesh, P. Sprechmann, and Y. LeCun, "Disentangling factors of variation in deep representation using adversarial training," in *Proc. Adv. Neural Inf. Process. Syst.*, 2016, pp. 5040–5048.
[17] I. Higgins, A. Pal, A. Rusu, L. Matthey, C. Burgess, A. Pritzel, M. Botvinick, C. Blundell, and A. Lerchner, "DARLA: Improving zero-shot transfer in reinforcement learning," in *Proc. 34th Int. Conf. Mach. Learn.*, vol. 70, 2017, pp. 1480–1490.
[18] D. P. Kingma and M. Welling, "Auto-encoding variational Bayes," 2013, *arXiv:1312.6114*. [Online]. Available: https://arxiv.org/abs/1312.6114
[19] S. J. Pan and Q. Yang, "A survey on transfer learning," *IEEE Trans. Knowl. Data Eng.*, vol. 22, no. 10, pp. 1345–1359, Oct. 2010.
[20] S. Benaim and L. Wolf, "One-shot unsupervised cross domain translation," in *Proc. Adv. Neural Inf. Process. Syst.*, 2018, pp. 2108–2118.
[21] J. Xu and Z. Zhu, "Reinforced continual learning," in *Proc. Adv. Neural Inf. Process. Syst.*, 2018, pp. 907–916.

[22] M. Kempka, M. Wydmuch, G. Runc, J. Toczek, and W. Jaskowski, "ViZDoom: A doom-based AI research platform for visual reinforcement learning," 2016, *arXiv:1605.02097*. [Online]. Available: https://arxiv.org/abs/1605.02097

[23] V. Mnih, A. P. Badia, M. Mirza, A. Graves, T. Harley, T. P. Lillicrap, D. Silver, and K. Kavukcuoglu, "Asynchronous methods for deep reinforcement learning," in *Proc. Int. Conf. Mach. Learn.*, 2016, pp. 1928–1937.

[24] K. James, 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," *Proc. Nat. Acad. Sci. USA*, vol. 114, no. 13, pp. 3521–3526, Mar. 2017.

[25] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. Riedmiller, "Playing Atari with deep reinforcement learning," 2013, *arXiv:1312.5602*. [Online]. Available: https://arxiv.org/abs/1312.5602

[26] V. Mnih, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, S. Petersen, C. Beattie, A. Sadik, I. Antonoglou, H. King, D. Kumaran, D. Wierstra, S. Legg, and D. Hassabis, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, p. 529, 2015.

[27] A. W. Moore and C. G. Atkeson, "Prioritized sweeping: Reinforcement learning with less data and less time," *Mach. Learn.*, vol. 13, no. 1, pp. 103–130, 1993.

[28] J. Peng and R. J. Williams, "Efficient learning and planning within the Dyna framework," *Adapt. Behav.*, vol. 1, no. 4, pp. 437–454, 1993.

[29] S. Gu, T. Lillicrap, I. Sutskever, and S. Levine, "Continuous deep Q-learning with model-based acceleration," in *Proc. Int. Conf. Mach. Learn.*, 2016, pp. 2829–2838.

[30] D. Silver, H. van Hasselt, M. Hessel, T. Schaul, A. Guez, T. Harley, G. Dulac-Arnold, D. Reichert, N. Rabinowitz, A. Barreto, and T. Degris, "The predictron: End-to-end learning and planning," in *Proc. 34th Int. Conf. Mach. Learn.*, vol. 70, 2017, pp. 3191–3199.

[31] A. Tamar, Y. Wu, G. Thomas, S. Levine, and P. Abbeel, "Value iteration networks," in *Proc. Adv. Neural Inf. Process. Syst.*, 2016, pp. 2154–2162.

[32] J. Oh, S. Singh, and H. Lee, "Value prediction network," in *Proc. Adv. Neural Inf. Process. Syst.*, 2017, pp. 6118–6128.

[33] H. Van Seijen and R. S. Sutton, "Efficient planning in MDPs by small backups," in *Proc. 30th Int. Conf. Mach. Learn.*, vol. 28, 2013, pp. III-361–III-369.

[34] T. Weber, S. Racanière, D. P. Reichert, L. Buesing, A. Guez, D. J. Rezende, A. P. Badia, O. Vinyals, N. Heess, Y. Li, R. Pascanu, P. Battaglia, D. Hassabis, D. Silver, and D. Wierstra, "Imagination-augmented agents for deep reinforcement learning," in *Proc. Adv. Neural Inf. Process. Syst.*, 2017, pp. 5690–5701.

[35] D. Corneil, W. Gerstner, and J. Brea, "Efficient model-based deep reinforcement learning with variational state tabulation," 2018, *arXiv:1802.04325*. [Online]. Available: https://arxiv.org/abs/1802.04325

[36] B. C. Stadie, S. Levine, and P. Abbeel, "Incentivizing exploration in reinforcement learning with deep predictive models," 2015, *arXiv:1507.00814*. [Online]. Available: https://arxiv.org/abs/1507.00814

[37] D. Pathak, P. Agrawal, A. A. Efros, and T. Darrell, "Curiosity-driven exploration by self-supervised prediction," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. Workshops*, Jul. 2017, pp. 16–17.

[38] M. E. Taylor, N. K. Jong, and P. Stone, "Transferring instances for model-based reinforcement learning," in *Proc. Joint Eur. Conf. Mach. Learn. Knowl. Discovery Databases*. Springer, 2008, pp. 488–505.

[39] S. R. Sekharan, R. Natarajan, and S. Rajasekaran, "Transfer from multiple linear predictive state representations (PSR)," 2017, *arXiv:1702.02184*. [Online]. Available: https://arxiv.org/abs/1702.02184

[40] J. Schmidhuber and R. Huber, "Learning to generate artificial fovea trajectories for target detection," *Int. J. Neural Syst.*, vol. 2, nos. 1–2, pp. 125–134, 1991.

[41] S. M. Khansari-Zadeh and A. Billard, "Learning stable nonlinear dynamical systems with Gaussian mixture models," *IEEE Trans. Robot.*, vol. 27, no. 5, pp. 943–957, Oct. 2011.

[42] N. Wahlström, T. B. Schön, and M. P. Deisenroth, "From pixels to torques: Policy learning with deep dynamical models," in *Proc. Deep Learn. Workshop 32nd Int. Conf. Mach. Learn. (ICML)*, Lille, France, Jul. 2015, pp. 1–10.

[43] M. Watter, J. Springenberg, J. Boedecker, and M. Riedmiller, "Embed to control: A locally linear latent dynamics model for control from raw images," in *Proc. Adv. Neural Inf. Process. Syst.*, 2015, pp. 2746–2754.

[44] V. Patraucean, A. Handa, and R. Cipolla, "Spatio-temporal video autoencoder with differentiable memory," 2015, *arXiv:1511.06309*. [Online]. Available: https://arxiv.org/abs/1511.06309

[45] K. Saenko, B. Kulis, M. Fritz, and T. Darrell, "Adapting visual category models to new domains," in *Proc. Eur. Conf. Comput. Vis.* Springer, 2010, pp. 213–226.

[46] R. Gopalan, R. Li, and R. Chellappa, "Domain adaptation for object recognition: An unsupervised approach," in *Proc. Int. Conf. Comput. Vis.*, Nov. 2011, pp. 999–1006.

[47] B. Gong, Y. Shi, F. Sha, and K. Grauman, "Geodesic flow kernel for unsupervised domain adaptation," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2012, pp. 2066–2073.

[48] J. Hoffman, S. Guadarrama, E. Tzeng, R. Hu, J. Donahue, R. Girshick, T. Darrell, and K. Saenko, "LSDA: Large scale detection through adaptation," in *Proc. Adv. Neural Inf. Process. Syst.*, 2014, pp. 3536–3544.

[49] C. Finn, X. Y. Tan, Y. Duan, T. Darrell, S. Levine, and P. Abbeel, "Deep spatial autoencoders for visuomotor learning," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, May 2016, pp. 512–519.

[50] S. Höfer, A. Raffin, R. Jonschkowski, O. Brock, and F. Stulp, "Unsupervised learning of state representations for multiple tasks," in *Proc. Deep Learn. Workshop Conf. Neural Inf. Process. Syst. (NIPS)*, 2016, pp. 1–10.

[51] S. J. Pan, I. W. Tsang, J. T. Kwok, and Q. Yang, "Domain adaptation via transfer component analysis," *IEEE Trans. Neural Netw.*, vol. 22, no. 2, pp. 199–210, Feb. 2011.

[52] E. Tzeng, J. Hoffman, T. Darrell, and K. Saenko, "Simultaneous deep transfer across domains and tasks," in *Proc. IEEE Int. Conf. Comput. Vis.*, Jun. 2015, pp. 4068–4076.

[53] E. Tzeng, J. Hoffman, N. Zhang, K. Saenko, and T. Darrell, "Deep domain confusion: Maximizing for domain invariance," 2014, *arXiv:1412.3474*. [Online]. Available: https://arxiv.org/abs/1412.3474

[54] J. Hoffman, D. Wang, F. Yu, and T. Darrell, "FCNs in the wild: Pixel-level adversarial and constraint-based adaptation," 2016, *arXiv:1612.02649*. [Online]. Available: https://arxiv.org/abs/1612.02649

[55] M.-Y. Liu, T. Breuel, and J. Kautz, "Unsupervised image-to-image translation networks," in *Proc. Adv. Neural Inf. Process. Syst.*, 2017, pp. 700–708.

QI YAN received the B.S. degree from the Department of Automation, Tsinghua University, Beijing, China, in 2015, where he is currently pursuing the Ph.D. degree.

His current research interests include computer vision, artificial intelligence, and reinforcement learning.

SHANGQI GUO received the B.S. degree in mathematics and physics basic science from the University of Electronic Science and Technology of China, Chengdu, China, in 2015. He is currently pursuing the Ph.D. degree with the Department of Automation, Tsinghua University, Beijing, China.

His current research interests include inference in probabilistic graphical models, artificial intelligence, brain-inspired computing, and computational neuroscience.

DAGUI CHEN received the B.S. degree from the Department of Automation, Tongji University, Shanghai, China, in 2016. He is currently pursuing the master's degree with the Department of Automation, Tsinghua University, Beijing, China.

His current research interests include computer vision, artificial intelligence, and reinforcement learning.

ZHILE YANG received the B.S. degree in automation from Tsinghua University, Beijing, China, where he is currently pursuing the M.S. degree with the Department of Automation.

His current research interests include computer vision, reinforcement learning, and robot control.

FENG CHEN received the B.S. and M.S. degrees in automation from Saint-Petersburg Polytechnic University, Saint Petersburg, Russia, in 1994 and 1996, respectively, and the Ph.D. degree from the Automation Department, Tsinghua University, Beijing, China, in 2000.

He is currently a Professor with Tsinghua University. His current research interests include computer vision, brain-inspired computing, and inference in graphical models.