Primitives Generation Policy Learning without Catastrophic Forgetting for Robotic Manipulation

Fangzhou Xiong[1,2], Zhiyong Liu[1,2,3], Kaizhu Huang[4], Xu Yang[1,2], Amir Hussain[5]
[1]*State Key Lab of Management and Control for Complex Systems, Institute of Automation, Chinese Academy of Science, China*
[2]*School of Computer and Control, University of Chinese Academy of Sciences (UCAS), China*
[3]*CAS Centre for Excellence in Brain Science and Intelligence Technology, Chinese Academy of Sciences, China.*
Email: {xiongfangzhou2015, xu.yang}@ia.ac.cn, zhiyong.liu@ia.ac.cn(corresponding author)
[4]*Department of EEE, Xi'an Jiaotong-Liverpool University, China*
Email: Kaizhu.Huang@xjtlu.edu.cn
[5]*School of Computing, Edinburgh Napier University, U.K.*
Email: A.Hussain@napier.ac.uk

***Abstract*—Catastrophic forgetting is a tough challenge when agent attempts to address different tasks sequentially without storing previous information, which gradually hinders the development of continual learning. Except for image classification tasks in continual learning, however, there are little reviews related to robotic manipulation. In this paper, we present a novel hierarchical architecture called Primitives Generation Policy Learning to enable continual learning. More specifically, a generative method by Variational Autoencoder is employed to generate state primitives from task space, then separate policy learning component is designed to learn torque control commands for different tasks sequentially. Furthermore, different task policies could be identified automatically by comparing reconstruction loss in the autoencoder. Experiment on robotic manipulation task shows that the proposed method exhibits substantially improved performance over some other continual learning methods.**

***Index Terms*—Variational Autoencoder, Continual learning, Catastrophic forgetting, Robotic manipulation**

I. Introduction

As a core component in artificial intelligence, learning is an essential approach to acquire new knowledge by interacting with surrounding world [1], [2]. It requires agents are capable of memorizing different knowledge in an abstract, structured and systematic way, which benefits for performing a variety of tasks. Traditional learning methods in machine learning usually require all task data collected in advance to learn different tasks together [3], [4], while that is particularly difficult in reality: different task information may not be presented simultaneously. Therefore, agents must learn to handle these tasks in a sequential fashion.

Faced with this learning pattern, however, agents are prone to forget how to perform previously learned tasks after learning new ones since previous information is inaccessible to current learning. This phenomenon is called catastrophic forgetting [5], [6] during the process of learning sequential multi-tasks (SMT). In terms of the neural network, it tends to deviate the weights learned in previously tasks after training on subsequent tasks, which leads to a loss of performance on the previously learned ones.

Consequently, it is great of importance to endow agents with the ability of addressing SMT without catastrophic forgetting so as to continuously adapt to a changing environment in their life cycle. This is known as continual learning or lifelong learning [7], [8]. Even though numerous recent advances have witnessed impressive gains across several domains by deep neural networks [9], [10], these approaches enjoy success with the precondition of all simultaneously available data on different tasks. If directly applied to SMT regime, catastrophic forgetting will inevitably occur since the weights move away from original optimal values after learning new tasks [11]. Hence, it is necessary to solve catastrophic forgetting problems with continual learning which to some extent promotes the development of artificial general intelligence.

Traditional solutions for catastrophic forgetting problems may suffer from some certain drawbacks. For instance, Naive fine-tune behaves oblivious in spite of a positive effect on new task learning by initializing new task with the optimal settings of old ones [12]. Whereas recent advances have provided multiple ways to overcome catastrophic forgetting across a variety of domains [13], [14]. Li et al. [13] employ fine-tune and the Distillation Networks to enable sequential learning by introducing a shared model with separate classification layers. Based on this basic framework, [15] designs an extra undercomplete autoencoder to capture important features on previous tasks to overcome catastrophic forgetting. Nevertheless, these methods both need to provide the knowledge of which task is being performed during the test phase, which is to say additional task identifier has to be supplied for indicating corresponding task parameters. To avoid this problem, Elastic weight consolidation (EWC) focuses on addressing SMT within one neural network by reducing the plasticity of weights that are vital to previously learned tasks [11].

These methods, however, usually concentrate on continuously learning knowledge related to image classification tasks, which requires to remember different images sequentially. In comparison, there is little research in the field of robotics.

SMT-GPS [14] tries to alleviate catastrophic forgetting between a series of robotic manipulation tasks through employing EWC algorithm. Ennen et al. [16] introduce Generative Motor Reflexes (GMR) to drive robust robotic policy representation, but it can only handle one task.

Rather than handling one task with GMR, we aims to solve multiple robotic tasks sequentially without catastrophic forgetting. In this work, we propose a novel hierarchical architecture called Primitives Generation Policy Learning (PGPL), which consists of two components: primitives generation and policy learning. Primitives generation component employs a Variational Autoencoder (VAE) [17] to generate state primitives, and policy learning component aims to learn a task specific operator to perform corresponding task. When facing a new task, previously trained VAE is utilized to generate state primitives for subsequent policy learning. In this manner, we are able to generate similar state primitives to describe tasks elementary features, while giving the model more flexibility to adapt itself to different tasks with corresponding operators. Meanwhile, task identifier could be obtained by comparing reconstruction loss in VAE among different tasks.

To sum up, the main contributions to this paper can be listed as the followings.

- We propose a novel hierarchical policy learning architecture Primitives Generation Policy Learning (PGPL) to achieve continual learning with an end-to-end manner in the field of robotics.
- By generating state primitives with VAE, different tasks could be addressed sequentially without additional task identification.
- Experiment on several robotic manipulation task shows that the proposed method exhibits substantially improved performance over some other continual learning methods.

The rest of the paper is organized as follows: Section II gives a brief review of related work. Section III presents formulation of the proposed method. Comparative experimental results are presented and discussed in section IV. Finally, concluding remarks are outlined in section V.

II. Related Work

Standard multi-task learning attempts to address multiply tasks in a single model by integrating knowledge from different domains [3]. Nevertheless, it requires the presence of all task data to train this model. In the continual learning regime, tasks are presented in a sequential fashion. Actually, continual learning [7], [8] has been proposed for several decades, which requires to learn SMT without catastrophic forgetting [5], [6]. Recently, several advances in machine learning have been made to offer different solutions to overcome catastrophic forgetting. A classical method is to fine-tune a pretrained task on the target one. Generally speaking, it uses the optimal settings of old tasks to help initialize for the next ones, which has been shown to be a successful method. However, this method gradually forget how to perform old tasks as training new tasks progresses [12].

By introducing synaptic consolidation for artificial neural networks, Elastic weight consolidation (EWC) [11] proposes to remember old tasks through employing Fisher regularisers to keep some previous weights unchanged. Besides, Lee [18] presents an incremental moment matching (IMM) algorithm from a neural network weight level to estimate the mixture of Gaussian posterior distributions of different tasks so that catastrophic forgetting problem could be resolved. However, these approaches only achieve good performances on similar tasks, since the regularization constraint is obtained in a neighbourhood of one possible minimizer of previous tasks.

Learning without Forgetting (LwF) [13] takes advantage of the output for new task to approximate the recorded output from the original network, which benefits the preservation of responses on the old tasks. However, [19] shows that LwF could reduce the performance when faced with a sequence of tasks that are drawn from different distributions. Build upon this framework, [15] introduce an additional under-complete autoencoder to construct a submanifold of important features for different tasks, which tries to restrict task distributions to enable more reasonable knowledge distillation loss [13]. Even though this method achieve good performance in image classification tasks, it has to train extra autoencoders for each new task.

Generative Motor Reflexes (GMR) [16] is the method with some similarities to our method. However, it is not designed for continual learning, but rather for learning robust robotic manipulation policy in a single task. This Generative Motor Reflex policy is implemented under the framework of Guided Policy Search [20], which assumes access to all tasks at any time. Besides, instead of optimizing motor reflex, we directly consider to learn policy for torque control commands.

III. Proposed Method

A. Preliminaries in Task Dataset

As the traditional continual learning problems mostly focus on how to perform image classification tasks, it is not difficult to handle tasks sequentially with an supervised learning manner. Generally speaking, researchers basically are free and easy to access the image classification datasets, and most of them are well labelled. For robotics tasks, however, to provide relevant datasets for continual learning is not simple and facile. For example, in a robotic manipulation problem, it is hard to find a ground truth at each time step in a complicated environment to guide the action. Moreover, robotic trajectory samples commonly appear in a temporal relationship which is intractable to find a mapping like an image classification task between images and labels. Even if loss function could be employed to label task data, it still may deviate the true data distribution since the accumulated error will increase when solving control torques step by step.

To address robotic tasks in a supervised manner, we have to inevitably collect relevant labelled task samples. Fortunately, in reinforcement learning, state-action pairs could be generated to construct task dataset for robotic manipulation problem. Here, we employ guided policy search (GPS) [21] to provide

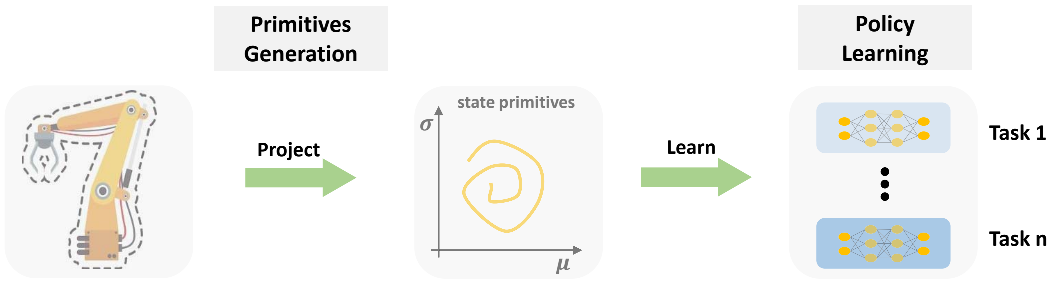


Fig. 1. The architecture of the proposed method. Best viewed in color.

state-action pairs. To this end, we could run GPS to resolve corresponding action when given a state so that task dataset (i.e. state-action pairs) could be constructed.

B. Problem Formulation

The proposed Primitives Generation Policy Learning (PG-PL) architecture consists of two components, a primitives generation and policy learning. Two components are learned together with an end-to-end manner for each task with the constructed dataset. Figure 1 illustrates the architecture when PGPL is employed to address SMT in robotics.

There are two components in PGPL. The primitives generation focuses on generating state primitives from task space. Build upon these state primitives, we could train separate policy for each task with policy learning component. After generating state primitives for current task, primitives generation component is kept unchanged to subsequent tasks. There are two main reasons behind this. One of them is that robotic trajectories in some tasks present similarity to some degree. For instance, inserting peg and opening door, both of them require agent to learn a straight line to approximate target and a certain action along a direction to reach target, which makes visited trajectories show some similarity in state space. And the other reason is that these generated state primitives are capable of describing basic features of the task, which possess a certain degree of versatility to other similar tasks. Hence, we propose to keep the primitives generation component unchanged after learning the first task, so that it can project the original state of different task to similar state primitives which benefit for subsequent policy learning.

C. Primitives Generation

In order to generate appropriate state primitives, the primitives generation component must satisfy two properties. Firstly, generated state primitives are capable of describing the basic features of the task, which can reflect the original state in task space to some degree [22]. Secondly, primitives generation component is of value to learning other tasks. To meet these two requirements, we employ the Variational Autoencoder (VAE) [17] to construct our primitives generation component (shown in Figure 2). Generally, A VAE is composed of two parts: an encoder network and a decoder network. Therein, the encoder network is responsible for mapping an input to a latent representation, while the decoder network tries to recover the input from the latent representation.

Specifically speaking, under the weights θ_{enc} an encoder network W_{enc} translates an input (i.e. state x) to a mean μ and a standard deviation σ by

$$[\mu, \sigma] = W_{enc}(x; \theta_{enc}), \tag{1}$$

then a latent representation could be obtained by $z = \mu + \sigma$. That is to say, we can achieve a state primitives z following:

$$z = W_{enc}(x), x \sim p(z|x; \theta_{enc}). \tag{2}$$

Accordingly, the reconstructed input $\hat{x}$ could be solved by the decoder network W_{dec} with the weights θ_{dec}:

$$\hat{x} = W_{dec}(z), \hat{x} \sim q(\hat{x}|z; \theta_{dec}). \tag{3}$$

The reconstruction loss L_{data} in VAE is the negative expected log-likelihood of the input x.

$$L_{data} = -\mathbb{E}_{p(z|x)}[logq(x|z)]. \tag{4}$$

Thus, we can utilize reconstruction loss L_{data} to generate the state primitives z which are the latent representation of the original state in task space.

For the second property in the primitives generation component, we employ the distribution of state primitives z to address this problem. To be specific, VAE has an ability to control the distribution of latent representation. If $p(z|x)$ follows an independent unit Gaussian distribution, then the distribution $p(z)$ will approximate an independent unit Gaussian distribution by:

$$p(z) = \sum_x p(z|x)p(x) = \sum_x N(0,1)p(x) = N(0,1), \tag{5}$$

which makes it more reasonable to generate latent variables from $N(0, I)$. To meet this precondition, the Kullback-Leibler (KL) divergence L_{kl} between the latent representation $p(z|x)$ and $N(0, I)$) has to be considered:

$$L_{kl} = D_{kl}(p(z|x)||N(0, I)). \tag{6}$$

Therefore, different tasks can establish a relationship with $N(0, I)$ accordingly. To some extent, state primitives of current task build a connection with subsequent task learning.

D. Policy Learning

On the one hand, primitives generation component is capable of generating state primitives to solve basic features of different tasks, which is benefit for learning a task. But on the other hand, these shared state primitives are not enough to describe each task perfectly since they are only the state submanifold of original task space. To perform every task sequentially, we have to compensate these inaccurate state primitives representation, which requires that every basic feature of each task should be taken into consideration. For this end, we need to handle each task separately to some extent.

Here, build upon state primitives, we propose to train each task with separate policy neural network, then different control torques could be obtained to perform corresponding task. Given the generated state primitives z, policy neural network f_i for task i and corresponding output target u^i, we can training the policy learning component through optimizing the following loss function:

$$L_{f_i} = \min_{\theta_{f_i}} l(f_i(z), u^i) \tag{7}$$

where the notation L_{f_i} denotes the loss in the policy learning component f_i for task i with the corresponding parameters θ_{f_i} to be optimized.

If the number of task to learn is M, we have to train M sets of neural network parameters for policy learning component. Even though generated state primitives are shared between different tasks, we have to decide which set of parameters to load for policy learning component when executing a task. That is to say, the corresponding task identifier I must be given to indicate parameters.

Fortunately, we do not need extra step to obtain task identifier. When studying primitives generation component, reconstruction loss L_{data_i} of each task i on the training dataset X_i could be resolved in VAE accordingly:

$$L_{data_i} = -\mathbb{E}_{p(z|x_i)}[logq(x_i|z)], x_i \in X_i, \tag{8}$$

where reconstruction loss term L_{data_i} corresponds to task i, whereas the VAE is established for all tasks.

When given a task j for testing, we firstly employ the VAE to compute the reconstruction loss L_{data_j} by sampling a trajectory, and then make a comparison with the terms $L_{data_i}, i = 1, 2, \ldots, M$ which are already solved during the training phase. Since reconstruction loss on the training dataset is related to the loss on the testing dataset within the same task, we could choose the most similar loss term to determine the task identifier I by:

$$I = argmin_i \|L_{data_j} - L_{data_i}\|^2, i = 1, 2, \ldots, M \tag{9}$$

More specifically, every task follows a different distribution in which the data belonging to task j are adopted to solve the corresponding loss term L_{data_j}, so other tasks $i (i \neq j)$ will produce greater diversity on the loss term. Therefore, the correct task identifier I could be achieved by comparing these reconstruction loss terms.

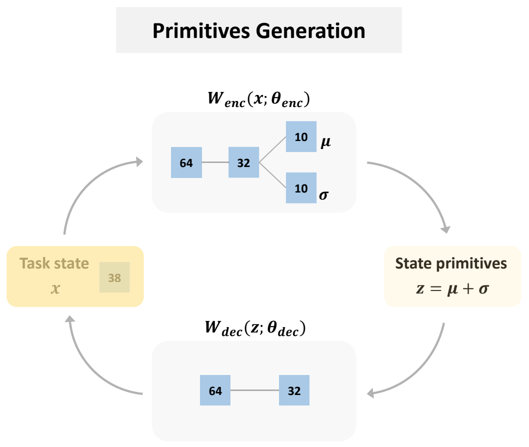


Fig. 2. The primitives generation component. A variational autoencoder is employed to generate state primitives by encoding the state x with W_{enc} and reconstructing it with W_{dec}. The number in the box stands for the corresponding number of neurons. Best viewed in color.

In addition, when these tasks behave more differently from each other, it will be easier to obtain different reconstruction error terms with greater diversity which helps to recognize the correct identifier I.

E. Primitives Generation Policy Learning

In PGPL architecture, we train two components in an end-to-end manner, which involves to learn the basic state primitives as well as individual policies. Therefore, the total loss L_{PGPL} could be presented as the following:

$$L_{PGPL} = \alpha L_{data} + \beta L_{kl} + L_{f_i} \tag{10}$$

where L_{data} denotes the reconstruction loss in VAE, L_{kl} describes the KL divergence between state primitives and a unit Gaussian distribution, and L_{f_i} presents the policy learning loss in primitives generation component for each task i. Besides, α, β are the corresponding hyperparameters which weight the importance of state primitives for all tasks.

When training on a new task, previously trained VAE is utilized to construct state primitives. In this manner, we can project different task data with the same mapping function to generate similar state primitives so that tasks elementary features could be described. For each task, we introduce separate policy learning component, which gives the architecture more flexibility to adapt itself to different tasks. Hence, control torques can be solved to perform each task accordingly. Meanwhile, task identifiers can be obtained by comparing VAE loss among different tasks.

IV. Experiment

The proposed method is compared against the several baselines on robotic tasks. We consider two tasks learned sequen-

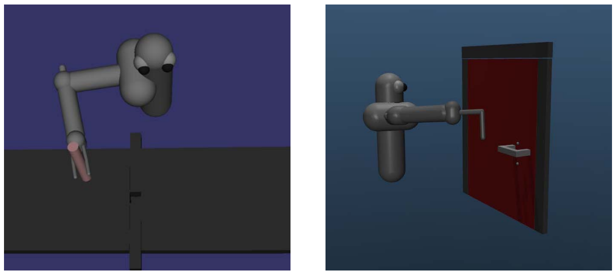

(a) Peg insertion (b) Door opening

Fig. 3. Illustrative screenshots of environments.

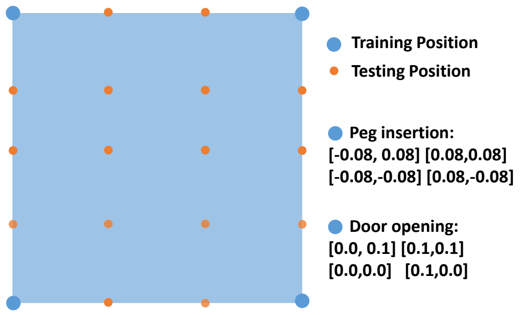


Fig. 4. Illustration of positions in testing dataset. For each task, there are only 4 initial positions for training. As for testing, initial positions for each task are picked within the area of training positions.

tially to evaluate the effectiveness of overcoming catastrophic forgetting.

A. Experimental Settings

1) Environment: Two robotic manipulation tasks are constructed in the MuJoCo simulation environment [23] to evaluate the proposed method. Figure 3 shows the relevant experimental environments for simulation. Peg insertion requires to control a 7 degree of freedom (DoF) 3D robot arm to insert a tight-fitting peg into a specified hole, while door opening requires to grasp the handle and pulling the door to a target angle with a 6 DoF 3D arm. The input state consists of joint angles, end-effector angles and their velocities, and the action includes motor torque command of each joint. More precisely, there are 26 dimensional state information and 7 torque commands for peg insertion task, while door opening task requires to learn a 6 dimensional action based on a 38 dimensional state.

2) Datasets: For peg insertion task, the Mirror Descent GPS (MDGPS) [21], a variant of GPS, is adopted to produce corresponding task dataset from 4 different training positions (shown in Figure 4). After optimizing this MDGPS agent, we could obtain 40 trajectories with 100 steps (i.e. 4000 state-action pairs) for this task to perform subsequent learning. Considering inherently discontinuous problems in door opening task, we use PILQR algorithm [24] to generate corresponding task dataset. Similarly, 4000 state-action pairs will be collected for this task.

In addition, to evaluate the effectiveness of the proposed PGPL method, different initial positions are utilitzed to testify whether the agent can finish the task sequentially. To be specific, for peg insertion task, we choose 81 points uniformly in the square constructed by 4 training positions, while 121 testing point are obtained in the same way for door opening task.

3) Architecture: Instead of taking image as a visual observation, we employ raw data directly collected from sensors in the simulation environment to train the PGPL architecture. The proposed architecture consists of two components: primitives generation and policy learning. For primitives generation component, we design a multi-layer perceptrons (MLP) with the structure of $[38-64-32-10]$ to generate the mean and standard deviation of state primitives in VAE with encoder network W_{enc} and corresponding decoder network W_{dec}. As for policy learning component, we employ separate MLP with $[10-32-7]$ for each task. The activation function ReLU is applied to the whole architecture except for the last fully connected linear layer. Note that even though the dimension of state in two tasks are different, we could conduct a zero padding to render the same input dimension for the neural network. In our experiment, peg insertion task will increase its state dimension from 26 to 38 by zero padding. Moreover, since door opening task only involves 6 dimensional action, we will extract corresponding torque commands from the last layer to perform the task.

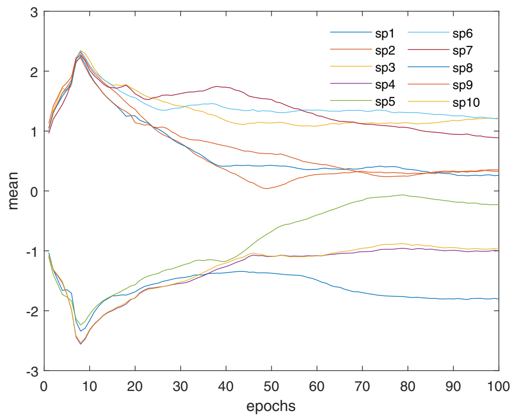
Here, the neural network is trained using an ADAM optimizer with the momentum term 0.9 and initial learning rate 0.001. Besides, we set hyper-parameter α for reconstruction loss in VAE to 0.7, a KL divergence coefficient $\beta = 0.001$, and batch size 25. Epochs are trained to 100, 150 for peg insertion and door opening task, respectively.

4) Compared Methods: The proposed PGPL architecture is compared with other 3 methods. Finetuning optimizes each new task using the previous task model as initialization. Elastic weight consolidation (EWC) introduces the Fisher information obtained from the previous task to regularize the new task learning, and incremental moment matching (IMM) reestimates the posterior distributions of different tasks to solve catastrophic forgetting problem. To make a fair comparison, the same neural network settings are adopted except for network structure designed as $[38-64-100-64-32-7]$ which tries to imitate the structure in PGPL.

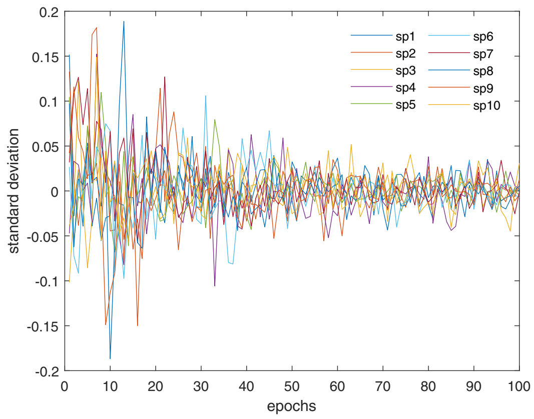
B. State Primitives Generation

The primitives generation component in the context of continual learning is typically important since it is responsible for generating state primitives so that the basic features of the task could be solved for subsequent learning. At the same time, it has some ability to generalize to other similar tasks.

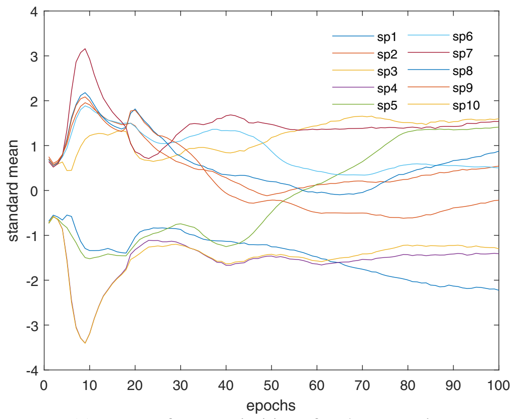
Figure 5 shows the learned information about state primitives for different tasks during the training. Although the primitives generation component is trained with the first task, it still presents some basic features among different tasks.



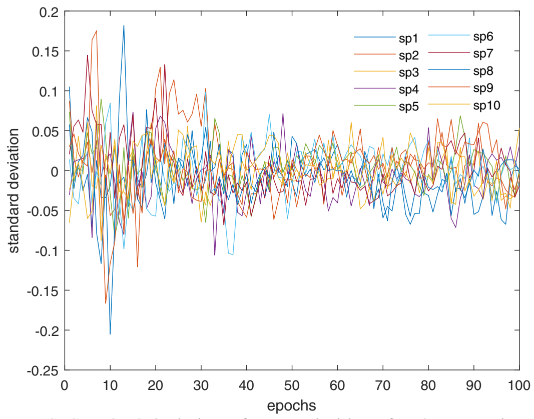
(a) Mean of state primitives for peg insertion



(b) Standard deviation of state primitives for peg insertion



(c) Mean of state primitives for door opening



(d) Standard deviation of state primitives for door opening

Fig. 5. Results of mean and standard deviation of state primitives (sp for short) in VAE for two tasks. We can divide state primitives coarsely into two groups. Group 1 contains sp2, sp3, sp6, sp7, sp8 and sp9, while the rest of state primitives (i.e. sp1, sp4, sp5 and sp10) constitute the group 2. Note that standard deviation term located in longitudinal coordinates actually describes the log function of the square of standard deviation $log\sigma^2$. Best viewed in color.

State primitives are mainly divided into two groups and state primitives in each group show some similarity. For instance, state primitives in group 1 always increase firstly and then behave an oscillatory descent to achieve a stable value at the end of training. This observation indicates that primitives generation component is able to capture the basic features among different tasks in spite of only trained on the first one. One possible explanation is that robotic tasks for peg insertion and door opening, to some extent, present some similar patterns during the learning phase.

As Equation 6 presents, since each task builds a KL constrain with an independent unit Gaussian distribution $N(0,1)$, they establish some inherent relationships correspondingly with each other. This can been verified from Figure 5 as well.

C. Task Identifying

Since each task has the corresponding policy learning component, we have to solve a task identifier to indicate which policy to execute. In fact, there is a discrepancy between different tasks even for the same primitives generation component. To explain this problem, we pick a state primitive and plot its learning process for two tasks in Figure 6. Though the state primitive 5 (sp5 for short) belongs to group 2 among these 10 primitives both in two tasks, it presents differently during the learning process. After 50 epochs, sp5 will increase sharply in door opening task and achieve the final mean at 1.41, while in peg insertion task sp5 varies slowly to get a mean of -0.23 at the end. Therefore, VAE presents some difference in state primitives among different tasks.

Actually, the loss of VAE could be employed to indicate the corresponding task identifier. Given a task, we sample some trajectories ahead to calculate the average loss under different task policies, and compare the loss term to choose the task identifier. Table I illustrates the relevant result. If we sample 5 trajectories to solve VAE loss terms, we can obtain the average reconstruction loss of 2.34 to match corresponding task parameters when testing door opening task. It is worth noting that only Data loss (i.e. reconstruction loss) is employed to choose the task identifier since KL loss term in VAE is constrained with an independent unit Gaussian distribution

TABLE I
Results of recognizing the task identifier.

	Policy for peg insertion		Policy for door opening	
	Data Loss	KL Loss	Data Loss	KL Loss
Peg insertion	**0.09**	4.9	9.21×10^7	4.58×10^7
Door opening	4.67	8.64	**2.34**	11.64

$N(0, 1)$, which makes it present a certain similarity among different tasks.

D. Overcoming Catastrophic Forgetting

In this section, the proposed PGPL method is applied to learn tasks sequentially. Here, we make a comparison with Finetuning, EWC and IMM to verify the effectiveness of PGPL. Under the same experiment settings, peg insertion and door opening task are learned sequentially among different methods.

As can be seen from Table II, the proposed method can perform the task well regardless of peg insertion or door opening. After sequentially learning, we can achieve 96.29% and 95.87% for these two tasks respectively. In fact, this result is guaranteed by recognizing task identifier correctly. The last section already verifies the effectiveness of VAE to select task identifier.

For other methods, they all forget how to perform the previous task after learning on new tasks, since it is much difficult for them to solve outputs within one set of parameters. To be specific, in robotic manipulation problem, to solve the torque commands for the final output is a regression problem, which is different from the classification problem in image classification. More importantly, in order to complete a task, robotic manipulation usually involves to execute a specific trajectory. Whereas these robotic trajectory appears in a temporal form which means every executed action will have an influence on subsequent decision since new state of trajectory is up to the last state, last action and dynamics in the environment. Apparently, it will cause accumulated error if we can not learn a relatively precise policy. But for image classification task, each classification operation is independent from each other. To this extent, there actually is some difference between image classification and robotic problem in the field of continual learning.

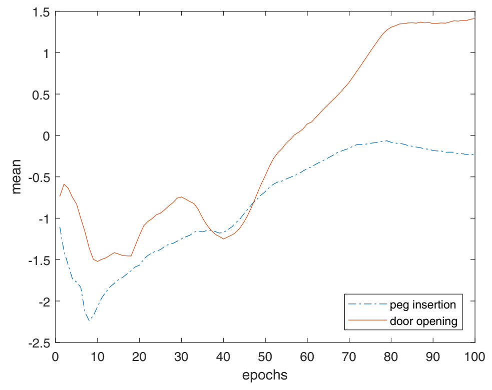


Fig. 6. Result of the mean of state primitive 5 in learning VAE.

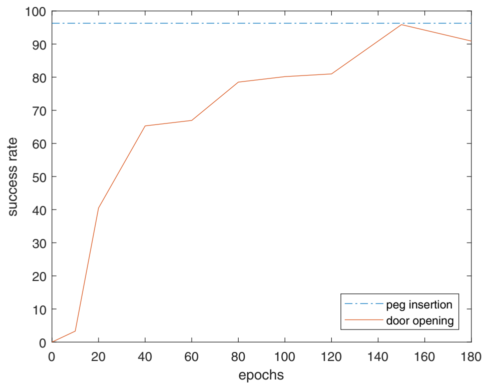


Fig. 7. Result of success rate during the process of learning door opening.

Even though EWC has been applied to reinforcement learning problem (e.g. Atari task) with Deep Q Networks (DQN) [25], it actually employs task specific information to handle different tasks. As paper [11] stated, "task-specific bias and gains at each layer" are used, which indicates that it is difficult to handle complex task within only one set of policy.

An interesting result is that EWC or IMM loss their advantage than traditional finetuning technique. One possible explanation is EWC proposes to solve continual learning problem with a Bayesian manner which requires to compute the posterior probability of previous tasks in a Laplace approximate way. Hence, the final solution only works in the neighbourhood of previously learned tasks. If directly applying this method to complex situation where tasks behave differently from each other, EWC can not find an effective solution to remember previous tasks without catastrophic forgetting. As for IMM, it considers to recompute the neural network parameters by introducing the mixture of Gaussian posterior with averaging operation (with or without Fisher information weighted), which actually is too simple to represent sophisti-

TABLE II
Results of success rate on two tasks learned sequentially. For peg insertion, if the Euclidean distance between the peg linked to the robotic end-effector and the bottom of the specified hole is less than 0.06, we think it is a successful experiment. For door opening, an experiment is considered as a successful case where the angle between the final plane of door and the starting plane is more than 1 radian.

	Peg insertion	Door opening	Average
Fintuning	0%	71.90%	35.95%
EWC	0%	58.68%	29.34%
IMM	0%	46.28%	23.14%
PGPL	**96.29%**	**95.87%**	**96.08%**

cated behaviours with only one set of policies in the robotic tasks.

Figure 7 illustrates the variation of task success rate in the learning process of how to open the door. After learning on peg insertion task, the VAE component remains unchanged, and the corresponding policy learning component is kept for this task, which makes the success rate always keep at 96.29%. With the shared VAE component and training on new policy learning component, we can achieve the optimal result (i.e. 95.87%) for door opening task after learning 150 epochs.

V. Conclusion

This work proposes Primitives Generation Policy Learning, a novel hierarchical architecture to address continual learning problem without catastrophic forgetting. Different from traditional continual learning algorithms in image classification, we focus on solving problems sequentially in the field of robotic manipulation. Specifically speaking, by generating state primitives, we are able to provide some basic state features for policy learning, and indicate which task is being performed by comparing reconstruction loss. And then agent could learn a separate policy with more flexibility to adapt itself to different tasks in the policy learning component. Relevant experiment on robotic task demonstrates that the proposed method presents substantially improved performance over some other continual learning methods.

Acknowledgment

The authors are grateful to the anonymous reviewers for their insightful comments and suggestions, which helped improve the quality of this paper. This work is supported by National Key Research and Development Plan of China grant 2017YFB1300202, NSFC grants U1613213, 61375005, 61503383, 61210009, the Strategic Priority Research Program of Chinese Academy of Science under Grant XDB32050100, Dongguan core technology research frontier project (2019622101001), and Suzhou Science and Technology Program grant SYG201712, SZS201613. The work is also supported by the Strategic Priority Research Program of the CAS (Grant XDB02080003), Key Program Special Fund in XJTLU (KSF01), and the UK Engineering and Physical Sciences Research Council (EPSRC) grant no. EP/M026981/1.

References

[1] S. Legg and M. Hutter, "Universal intelligence: A definition of machine intelligence," *Minds and Machines*, vol. 17, no. 4, pp. 391–444, 2007.

[2] R. S. Michalski, J. G. Carbonell, and T. M. Mitchell, *Machine learning: An artificial intelligence approach*. Springer Science & Business Media, 2013.

[3] R. Caruana, "Multitask learning," *Machine learning*, vol. 28, no. 1, pp. 41–75, 1997.

[4] S. Thrun and L. Pratt, *Learning to learn*. Springer Science & Business Media, 2012.

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

[6] R. M. French, "Catastrophic forgetting in connectionist networks," *Trends in cognitive sciences*, vol. 3, no. 4, pp. 128–135, 1999.

[7] S. Thrun, "Lifelong learning algorithms," *Learning to learn*, vol. 8, pp. 181–209, 1998.

[8] W. Hao, J. Fan, Z. Zhang, and G. Zhu, "End-to-end lifelong learning: a framework to achieve plasticities of both the feature and classifier constructions," *Cognitive Computation*, pp. 1–13, 2017.

[9] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *nature*, vol. 521, no. 7553, p. 436, 2015.

[10] Y. Xie, J. Xiao, K. Huang, J. Thiyagalingam, and Y. Zhao, "Correlation filter selection for visual tracking using reinforcement learning," *IEEE Transactions on Circuits and Systems for Video Technology*, 2018.

[11] J. Kirkpatrick, R. Pascanu, N. Rabinowitz, J. Veness, G. Desjardins, A. A. Rusu, K. Milan, J. Quan, T. Ramalho, A. Grabska-Barwinska *et al.*, "Overcoming catastrophic forgetting in neural networks," *Proceedings of the National Academy of Sciences*, p. 201611835, 2017.

[12] R. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2014, pp. 580–587.

[13] Z. Li and D. Hoiem, "Learning without forgetting," *IEEE transactions on pattern analysis and machine intelligence*, vol. 40, no. 12, pp. 2935–2947, 2017.

[14] F. Xiong, B. Sun, X. Yang, H. Qiao, K. Huang, A. Hussain, and Z. Liu, "Guided policy search for sequential multitask learning," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 2018.

[15] A. Rannen, R. Aljundi, M. B. Blaschko, and T. Tuytelaars, "Encoder based lifelong learning," in *Proceedings of the IEEE International Conference on Computer Vision*, 2017, pp. 1320–1328.

[16] P. Ennen, P. Bresenitz, R. Vossen, and F. Hees, "Learning robust manipulation skills with guided policy search via generative motor reflexes," in *Robotics and Automation (ICRA), 2019 IEEE International Conference on*. IEEE, May. 2019.

[17] D. P. Kingma and M. Welling, "Auto-encoding variational bayes," in *Proceedings of the International Conference on Learning Representations (ICLR)*, 2014.

[18] S.-W. Lee, J.-H. Kim, J. Jun, J.-W. Ha, and B.-T. Zhang, "Overcoming catastrophic forgetting by incremental moment matching," in *Advances in Neural Information Processing Systems*, 2017, pp. 4652–4662.

[19] R. Aljundi, P. Chakravarty, and T. Tuytelaars, "Expert gate: Lifelong learning with a network of experts," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 3366–3375.

[20] S. Levine and V. Koltun, "Guided policy search," in *Proceedings of the 30th International Conference on Machine Learning (ICML-13)*, Jun. 2013, pp. 1–9.

[21] W. H. Montgomery and S. Levine, "Guided policy search via approximate mirror descent," in *Advances in Neural Information Processing Systems*, 2016, pp. 4008–4016.

[22] X. Yang, K. Huang, R. Zhang, and A. Hussain, "Learning latent features with infinite nonnegative binary matrix trifactorization," *IEEE Transactions on Emerging Topics in Computational Intelligence*, no. 99, pp. 1–14, 2018.

[23] E. Todorov, T. Erez, and Y. Tassa, "Mujoco: A physics engine for model-based control," in *Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference on*. IEEE, 2012, pp. 5026–5033.

[24] Y. Chebotar, K. Hausman, M. Zhang, G. Sukhatme, S. Schaal, and S. Levine, "Combining model-based and model-free updates for trajectory-centric reinforcement learning," in *Proceedings of the 34th International Conference on Machine Learning (ICML-17)*, vol. 70, Aug. 2017, pp. 703–711.

[25] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski *et al.*, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, p. 529, 2015.