CONTINUAL REINFORCEMENT LEARNING USING FEATURE MAP ATTENTION LOSS

Anonymous authors
Paper under double-blind review

ABSTRACT

Despite the growing interest in continual learning (CL) in recent years, continual reinforcement learning (CRL) has still been a challenging task because deep neural networks must infer appropriate action from each never-seen observation from new tasks sustaining old tasks' performance. To solve this problem, some CRL algorithms used both regularization-based methods constraining the weights and replay-based method used in the conventional CL. However, it costs a lot of time to learn because it required a considerable amount of memory in terms of replay-based and have complex regularization terms. In this paper, we propose a simple framework for preserving knowledge FMAL among related sequential tasks, namely Feature Map Attention Loss. Our method leverages general CNNs of a model that can perform all of the sequential tasks well, and the attention mechanism is used to extract essential features to transfer. Also, FMAL uses both the regularization method and the replay-based method, like the existing CRL method. However, the amount of memory required for learning is much smaller, and the term of regularization is relatively simple. We evaluate FMAL with the state of the art algorithms. The experiment results show that our method outperforms these baselines with higher rewards.

1 INTRODUCTION

In recent years, deep neural networks have made significant progress in optimizing a model that is fitted to a single task in various fields. However, there is still a challenge called catastrophic forgetting (McCloskey Cohen, 1989; French, 1999), which is to lose previously learned information when learning a new task given sequential tasks. Solving the catastrophic forgetting problem has seen a rapid growth of interest in recent years because it means designing a general model in neural networks that performs multiple tasks. Moreover, it is seen as one of the gateways to artificial general intelligence, such as humans (Goodfellow et al., 2013; Kemker et al., 2018).

However, preventing catastrophic forgetting is difficult due to the dilemma of stability-plasticity (Carpenter Grossberg, 1987). The dilemma refers to a trade-off phenomenon in which it is difficult for the artificial neural network to simultaneously satisfy both the learning of new tasks while maintaining the previously learned information from old tasks. In other words, deep neural networks that focus on stability try to keep the previously learned task more, while networks that emphasize plasticity focus more on adapting to new tasks.

To solve this problem, many continual learning methods (Kirkpatrick et al., 2017; Nguyen et al., 2017; Rusu et al., 2016; Li Hoiem, 2017; Farquhar Gal, 2018) have been recently proposed. While continual learning methods are mainly dealt with in supervised learning tasks, it has recently begun to be applied to reinforcement learning called continual reinforcement learning (CRL). In CRL, regularization-based methods that limit important parameters of previously trained models are often used. Recent studies that apply replay-based and regularization-methods at the same time have begun (ER for CL).

However, most of the existing CRL algorithm has several problems, such as learning time, the amount of memory required, and the complicated regularization terms for learning. The reason there are many difficulties for learning in CRL is that the CRL algorithm overlooks the use of the similarities of observations and actions. However, if the model of the neural network does not take advantage of the similarity of the actions and the observations between sequential tasks, then a much

larger model size of the deep neural networks is required, and it takes much more time to train. For this reason, we focus on selecting similar sequential tasks for games with different states but the same dynamics called Procgen.

We propose a simple framework FAML for CRL in similar environments. Our proposed framework applies both the regularization-based method and the replay-based method in the existing continual learning. In terms of the replay-based method, replayed experiences that include subset of observations, feature maps with attention, and logits are stored in memory after learning a model. When learning the new task, an observation of the stored memory is input into the model, and a regularization term is additionally used so that the feature map with the attention of the new model and the stored feature map with attention are similar. Finally, the adaptation process is performed. This process matches logit with observations of the memory of all sequential tasks as inputs. When matching logit, parameters of the generally generated CNNs layer are frozen.

Our method help maintains the previous core knowledge without disrupting the learning of new tasks as much as possible. Moreover, our experiments need only two experience-memory with subsets of state and feature maps), and our method is complementary to existing continual reinforcement learning.

In summary, the main contributions of our work are three-fold:

- The proposed framework creates a general CNN architectures that can generate features for each observation of all sequential tasks.
- The deep neural network model of the proposed framework is basically a task-free continual learning method.
- Our method is complementary, not competing with other continual reinforcement learning algorithms. By adding only one loss function, better performance can be achieved.

2 Related Work

2.1 Continual Learning

Most continual learning approaches are focused on preventing catastrophic forgetting, and CL approaches can be divided into the three categories: replay-based methods, regularization-based methods, and dynamic architecture methods. The replay-based methods (Rebuffi et al., 2017; Lopez-Paz Ranzato, 2017) keep some of the data used from the previously learned model. And when learning the model for a new task, the stored subset of the sample and the input of the new task are input into the model together to maintain the performance of the previous task and to learn new knowledge. Gradient episodic memory (GEM) (Lopez-Paz et al., 2017) uses this idea to store the data at the end of previous tasks. And when learning the next task, the gradient is adjusted so that the loss function of the previous task does not increase by loading data from the previous task each episode. In regularization-based methods, significant changes for representation learned for previous tasks are prevented. These approaches are to identify the importance of each parameter and to constraint further changes to those parameters in previously learned models (Kirkpatrick et al., 2017; Aljundi et al., 2018; Zenke et al., 2017). Inspired by Bayesian learning, elastic weight consolidation (EWC) (**?**) uses the Fisher information matrix to measure priorities for important parameters. The Synaptic Intelligence (SI) (**?**) measures the importance of the parameter through calculating the loss function. In other words, parameters that have a greater impact on the loss function are important. Finally, the approaches for dynamic architecture is to change the architecture of the deep neural network model when adapting to the new task. Dynamically Expandable Network (DEN) (?) also expands its network by selecting drifting units and retraining them on new tasks. These approaches require the architecture grow with each new task.

2.2 Continual Reinforcement Learning

Recently, there has been interest in overcoming catastrophic forgetting in RL. most of the existing CRL algorithms focus on preventing the changes of the parameters learned in old task while training on new task such as regularization-based method. Progressive Networks freezes sub-networks trained on each tasks, and Progress Compress uses EWC to consolidate the network after each

task has been learned. However, the majority of current research in the field focus on using replay-based method with regularization-based method. For example, CLEAR method stores the replayed experiences that includes states, value function, policy, and entropy. In order to sustain the old performance, the replayed experiences is used as behavior cloning. Also, Continual and Multi–task reinforcement learning with shard episodic memory uses episodic memory. This methods explicitly split the encoding of episodic memory and task-specific memory into separate sub-networks.

2.3 TRANSFER LEARNING WITH REPRESENTATION

Transfer Learning is designed to transfer knowledge learned from the source network to a target with limited data samples. In terms of our method, our work is highly related to transfer learning with knowledge distillation. Knowledge distillation focus on teacher-student method based on the similar task. This method is also used to transfer learned knowledge from teacher network to student network through aligning their outputs of specific layers. We follow the conceptual ideas of knowledge distillation to regularize the layer's outputs of the network. Compared to above work, our work extracts features, which are more meaningful information, considering not only the feature map but also the selected action. And since these features even consider actions, it is easy to make a general CNNs Network. After creating a general CNN where all tasks are acceptable (after the last task is finished), our work is terminated by going through an adaptation process that creates an FC layer that all tasks can be satisfied with.

3 LEARNING FRAMEWORK

3.1 OVERALL FRAMEWORK

In this section, we present the overall design of the proposed framework and detailed key algorithms. we first train an agent for the first task using reinforcement learning based on Section 3.2.

$$\mathcal{L}_{\texttt{policy}} \tag{1}$$

After training an agent, we intend to regulate parameters to preserve knowledge based on replayed experiences generated from previous models. Replay experiences are added after training each task and includes a subset of states, logits, feature map with attention

When learning a new task, the old states of the replayed experiences are entered into the model at the same time as the states of the new task. The difference between the feature map obtained from the old states and the stored feature map with attention of the replayed experiences is additionally defined as a loss function.

$$\mathcal{L} = \mathcal{L}_{\texttt{policy}} + \alpha\mathcal{L}_{\texttt{FAM}}, \tag{2}$$

This process is designed to create a general CNNs architecture that can be used in all the tasks learned so far.

However, as the CNNs changed, the FCs were not adjusted for previous tasks, so after all tasks were completed, the FCs have to be adjusted. In other words, The current FCs layer has not been adapted to past tasks, and can only be used in the current task. We load states and logit pairs from replayed experiences for all tasks and perform logit matching to satisfy the general FC layers.

$$\mathcal{L} = \mathcal{L}_{\texttt{LM}}, \tag{3}$$

Through this framework, we designed general CNNs and FCs.

3.2 TRAINING AN AGENT USING RL ALGORITHMS

In this section, we introduce the reinforcement learning algorithm applied to all current tasks. For example, when the second task is learned in the CL setting, this RL algorithm for learning the

second task is used. We take the CNN architecture used in IMPALA as the policy network, and the Proximal Policy Optimization (PPO) methods to train the agents. We train an agent using on-policy actor-critic architectures Sutton & Barto (1998) for training the each task. It is trained both policy $\pi(a_t|s_t;\theta)$ and value function $V(s_t;\phi)$, with π, V. we update its parameter θ by estimating an estimator of the policy gradient (e.g., the REINFORCE family of algorithms (Williams, 1992)):

$$\mathcal{L}_{\texttt{policy}} = \mathbb{E}_{(s,a)}\left[-\log\pi\left(a|f\left(s;\phi\right);\theta\right)R\right] \tag{4}$$

3.3 Getting replayed experiences with Feature-Attention map

At the end of each task, we test the model of the currently trained task through interaction with the environment, and collect replayed experiences data **D**. In this **D**, a subset of states, feature maps with attention, and action-logits are stored. We use the Grad-CAM Selvaraju et al. (2017) technique to generate attention maps to preserve knowledge from the previous tasks. For using the Grad-CAM, firstly, the states are forwarded to the trained model, obtaining feature maps A_k. Following feature maps, each gradient value for a selected action y^c is computed concerning each feature maps A_k in the specific convolution layer.

In the later part, Global average pooling (GAP) and Rectified Linear Unit (ReLU) are used in the original Grad-CAM technique. However, GAP and ReLU are not used in our study because our study used the Grad-CAM technique to extract characteristic feature maps that consider output actions as well as input states.

For each A_k, we obtain the feature importance weight α_k of A_k. All the A_k weighted by α_k are passed to obtain a final attention map for action c. To be precise, let $\alpha_k = (\frac{\partial y^c}{\partial A_k})$. Let $\alpha = [\alpha_1, \alpha_2, \ldots, \alpha_K]$ and $\mathbf{A} = [A_1, A_2, \ldots, A_K]$, where K is the number of feature maps in the layer using which attention map is to be generated. The attention map Q can be defined as

$$Q = \alpha^T \mathbf{A} \tag{5}$$

After training each task, we got $\mathcal{D} = \{(s_t, a_t, FAM_t)\}$ and save them in the replayed experiences.

3.4 Continual Learning with replayed experiences for General CNNs

In this process, Feature Attention Maps (FAMs) loss are calculated to preserve the knowledge of past task using Q in the previous section 3.3. The saved FAM becomes the target output, and the FAM extracted by inputting the saved states into the model of the current task becomes the main output. We retrieve state, FAM results in the replayed experiences, and normalize the FAM by dividing it by the L_2 norm of the map. We perform this step while computing L_{FAM}. During the training of T_{t+1}, For this input, L_{FAM} is defined as the sum of element-wise L_1 difference of the normalized, and vectorized attention map:

$$L_{FAM} = \sum_{j=1}^{l} \|current\frac{Q^c_{i,j}}{\|Q^c_{i,j}\|_2} - previous\frac{Q^c_{i,j}}{\|Q^c_{i,j}\|_2}\|_p \tag{6}$$

FAM loss is a process for maintaining the previous model, and since a policy loss is added to learn the current task, the recent total loss is as follows:

$$\mathcal{L} = \mathcal{L}_{\texttt{policy}} + \alpha\mathcal{L}_{\texttt{FAM}}, \tag{7}$$

We create a general CNNs architecture by iterating through this section until all tasks are finished.

3.5 Adaptation Process for General FCs

3.4 This process is an adaptation to make the FCs layer usable in all tasks. We load states and logit pairs from replayed experiences for all tasks and perform logit matching to satisfy the general FC

layers. This adaptation process of a fully connected layer that satisfies all tasks goes through. In this case, all parameters of the CNN layer are frozen. The logit's loss was used with the mean squared error (MSE), and the loss function was as follows.

$$\mathcal{L}_{\texttt{LM}} = \frac{1}{n} \sum_{task=1}^{n} (logits_{current} - logits_{memory})^2, \tag{8}$$

Through this framework, we designed general CNNs and FCs.

4 EXPERIMENTS

In this section, we demonstrate the effectiveness of the proposed method on environments called (StarPilot, BossFight, Coinrun, BigFish and plunder) in Procgen (Cobbe et al., 2019a). We evaluate performance of agent on previous environments that agent completed, while training on unseen environments which consist of different backgrounds, objects, and floors.

4.1 BASELINES AND IMPLEMENTATION DETAILS

Because the Procgen tasks similar to Cobbe et al. (2019b), we take the CNN architecture used in IMPALA (Espeholt et al., 2018) as the policy network, and the Proximal Policy Optimization (PPO) (Schulman et al., 2017) method to train the agents.[1] Agent is sequentially trained in total ?? different environments and trained for ?? steps in each environments. Also PPO agent collect trajectory of length ?? for training. At each timestep, 64 x 64 size frame input into agent. We used ?? optimizer to update weights. We applyied Grad-CAM to generate feature attention map. [we used a reference implementation in url]. Feature attention maps are generated in condition with action selected by agent, and the feature map of last convolutional layer.

Our method is compared with several regularization and memory-based continual learning method. We used as baselines. distillation loss, deepmind method and others.

4.2 RESULT

Task description. In this part, we describe the common parts of the four environments StarPilot, BossFight, BigFish and Plunder in Procgen. In these environment, agent observes its surroundings in the third-person point of view. Each environments have many levels, we continued learning pass through different levels. Style of backgrounds, grounds, obstacles and other features are randomly determined in each level. But all levels have same dynamics.

In StarPilot, agent is located in left side of map, and have to avoid obstacles and enemies to survive. They come from right side of the map continuously. Also agent can kill enemies through shot them. Agent get bigger rewards when it survives long. Episode ends when agent dead by enemies. Bossfight is similar with StarPilot, agent is located down side of map but there are only one boss enemy. Goal of this environment is to avoid attack of boss and damage it. And there are only fixed obstacles which not moves. In Bigfish, the goal is to become bigger than all other fishes. Agent can make bigger itself through eating other fishes that smaller than agent. If agent eat bigger one, agent dies and episodes ends. Goal of Plunder is to destroy enemy ships. Agent have to destroy enemy ships as much as possible in limited time, while not destroying friendly ships.

Ablation study. To demonstrate effectiveness of our method, we except some elements of our method and compared to original method. Our method consist of big two sub-methods, FAM loss and logits loss. More specifically FAM loss is upgraded version of Feature Matching(FM) loss, because FAM is consist of features and gradients. Thus, we devise four downgraded version of our method, only FAM loss, only logits loss, only FM loss and FM + logits loss. These methods compared to our method.

[additional description due to result]

[1] We used a reference implementation in `https://github.com/openai/coinrun`.

Embedding analysis. We analyzed state embedding of agent during continual learning process, to verify effect of FAM loss. Embeded vectors, which is output of last convolutional layer, visualized through t-SNE. Fig? shows changes of embeded vectors during continual learning process. Agent maintains state embedding of previous task during learning new task, and finds state embedding that satisfies both previous and current task.

[other embedding quantification methods]

Visual interpretation. We visualized FAMs which generated through Grad-CAM methods, because it directly shows whether agent maintains FAM in previous task, and able to learn net task. In seen env, agent In unseen env, agent entropy of normalized FAMs, [In network randomization]

REFERENCES

Karl Cobbe, Christopher Hesse, Jacob Hilton, and John Schulman. Leveraging procedural generation to benchmark reinforcement learning. *arXiv preprint arXiv:1912.01588*, 2019a.

Karl Cobbe, Oleg Klimov, Chris Hesse, Taehoon Kim, and John Schulman. Quantifying generalization in reinforcement learning. In *ICML*, 2019b.

Lasse Espeholt, Hubert Soyer, Remi Munos, Karen Simonyan, Volodymir Mnih, Tom Ward, Yotam Doron, Vlad Firoiu, Tim Harley, Iain Dunning, et al. Impala: Scalable distributed deep-rl with importance weighted actor-learner architectures. In *ICML*, 2018.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

Ramprasaath R Selvaraju, Michael Cogswell, Abhishek Das, Ramakrishna Vedantam, Devi Parikh, and Dhruv Batra. Grad-cam: Visual explanations from deep networks via gradient-based localization. In *ICCV*, 2017.

Richard S Sutton and Andrew G Barto. Reinforcement learning: an introduction cambridge. *MA: MIT Press.[Google Scholar]*, 1998.

Ronald J Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, 1992.