7th ICML Workshop on Automated Machine Learning (2020)

Toward Synergism in Macro Action Ensembles

Yu-Ming Chen ALLENCHEN0958@GAPP.NTHU.EDU.TW
National Tsing Hua University, Taiwan

Kuan-Yu Chang S102020009@GAPP.NTHU.EDU.TW
National Tsing Hua University, Taiwan

Chien Liu LIU_CHIEN@GAPP.NTHU.EDU.TW
National Tsing Hua University, Taiwan

Tsu-Ching Hsiao JOEHSIAO@GAPP.NTHU.EDU.TW
National Tsing Hua University, Taiwan

Zhang-Wei Hong WILLIAMD4112@GAPP.NTHU.EDU.TW
National Tsing Hua University, Taiwan

Chun-Yi Lee CYLEE@CS.NTHU.EDU.TW
National Tsing Hua University, Taiwan

Abstract

Macro actions have been demonstrated to be beneficial for the learning processes of an agent. A variety of techniques have been developed to construct more effective macro actions. However, they usually fail to provide an approach for combining macro actions to form a synergistic macro action ensemble. A synergistic macro action ensemble performs better than individual macro actions within it. Motivated by the recent advances of neural architecture search, we formulate the construction of a synergistic macro action ensemble as a sequential decision problem and evaluate the ensemble in a task. The formulation of sequential decision problem enables coherency in the macro actions to be considered. Also, our evaluation procedure takes *synergism* into account since the *synergism* among the macro action ensemble exhibits when jointly used by an agent. The experimental results show that our framework is able to discover synergistic macro action ensembles. We further perform experiments to validate the *synergism* property among the macro action ensemble.

1. Introduction

A number of previous works have been devoted to developing macro action (or simply "macro") construction procedures (Yoshikawa and Kurihara, 2006; Newton et al., 2007; Durugkar et al., 2016). In the hope of alleviating the combinatorial complexity of constructing macro actions, these works attempted to reduce the scope down to searching macros individually, instead of developing multiple macros jointly. However, searching macros individually overlooks the *synergism* amongst macros, which is one of the keys for solving complex tasks. Incompatible macros may fail to synthesize required behaviors for solving such tasks. Hence, *synergism* among macros emerges to be of crucial importance. An ensemble of synergistic macro actions can facilitate the synthesis of complex behaviors. Synergistic macros can be executed either interleavely or jointly with primitive actions. Therefore, an ideal macro action ensemble (or simply "macro ensemble") should exhibit

synergism among the macros in it. Unfortunately, an appropriate approach to construct such an macro ensemble still remains a challenge and unexplored.

To address this challenge, we borrow the idea from the recent advances of Neural Architecture Search (NAS). Recent works (Baker et al., 2017; Zoph and Le, 2017; Liu et al., 2018) in NAS cast architecture design of a neural network (NN) as a Markov Decision Process (MDP), and solve this MDP by reinforcement learning (RL) (Baker et al., 2017). In this MDP, the type of each layer is decided by an RL-based controller at each decision step. The layers determined at all steps are then chained as a complete NN. This process is analogous to macro ensemble construction since NAS and macro ensemble construction both aim to search for the optimal sequence of decisions.

In the light of this analogy, we formulate macro ensemble construction as an MDP and optimize the construction process via RL. An RL-based controller decides a primitive action at each construction step based on the previously decided action sequence. At the end of an episode of this MDP, all primitive actions selected in this episode are then segmented into multiple macros that together form a macro ensemble. The generated macro ensemble is evaluated by an agent in a target task, and then the feedback is returned to the controller. The performances of macro ensembles, which reflect effectiveness as well as *synergism*, guide the controller to refine its macro ensemble construction policy toward *synergism*.

Our principal contributions are (1) introducing the *synergism* property, which is crucial to a macro ensemble, and (2) formulating a macro ensemble construction process as an MDP. In this paper, we propose an effective construction method grounded on RL and a parallel asynchronous framework to accelerate the construction process. We then validate the effectiveness of our methodology through conducting experiments on a range of *Atari 2600* (Bellemare et al., 2013) environments. The experimental results show that our method is able to discover effective and synergistic macro ensembles that improve RL agents' performance. Moreover, we perform a series of analyses to verify the existence and influence of the *synergism* property among macro actions constructed by our methodology.

The rest of the paper is organized as follows. Section 2 describes our methodology. Section 3 presents our experimental results. Section 4 concludes. The appendix is organized as follows. Section A reviews the related works. Section B introduces the background material. Sections C and D provide the experimental setup and our additional results.

2. Methodology

Our proposed methodology constructs macro ensembles using an parallel asynchronous framework, which is composed of two phases: a construction phase and an evaluation phase. The framework contains two main components: a controller $\mathcal{C}$, which is an RL agent updating its policy in an off-policy manner, and a worker pool $\mathcal{W}$, which is a set of asynchronous parallel workers implemented as agents. In the construction phase, $\mathcal{C}$ constructs macro ensembles and sends them to $\mathcal{W}$. During the evaluation phase, each worker in $\mathcal{W}$ evaluates the performance of the assigned macro ensemble in its own copy of the target environment. The evaluated performances of the constructed macro ensembles are stored in a replay memory. The controller $\mathcal{C}$ then updates its policy using the data sampled from the replay memory. The construction and evaluation phases are concurrently executed until a fixed number N of macro ensembles are constructed.

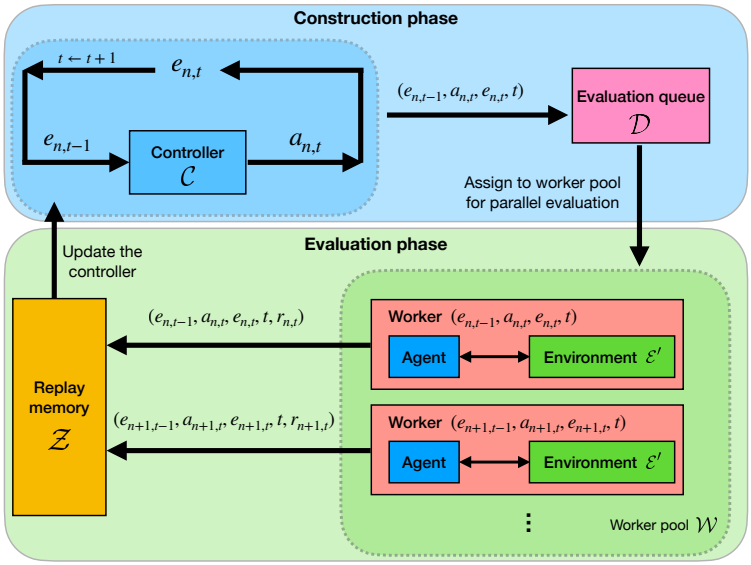

Figure 1: Overview of the proposed macro action ensemble construction framework.

In the following subsections, we first formulate the macro ensemble construction process as an MDP in Section 2.1. We then detail the construction and evaluation phases of our proposed methodology in Sections 2.2 and 2.3, respectively. Fig. 1 illustrates the complete framework of our methodology, while the pseudo code is summarized in Algorithm A1 .

2.1 Formulation of the Macro Action Ensemble Construction Process

In this section, we formulate the macro ensemble construction process as an MDP with a fixed episode length equal to T. In this MDP, the controller $\mathcal{C}$ decides a primitive action $a_{n,t} \in \mathcal{A} \cup \{null\}$ at timestep t, where n is the episode number, $\mathcal{A}$ is the primitive action space of the target environment $\mathcal{E}$, and $null$ is a padding action to be ignored by the worker. The decision of $a_{n,t}$ is based on the previously decided action sequence $e_{n,t-1}$, which can be represented as $e_{n,t-1} = [a_{n,1}, a_{n,2}, \cdots, a_{n,t-1}]$. The determined $a_{n,t}$ is then appended to $e_{n,t-1}$ to form $e_{n,t} = e_{n,t-1} \parallel a_{n,t}$, serving as the information required for the next decision. After taking $a_{n,t}$, $\mathcal{C}$ receives the corresponding reward $r_{n,t}$, which is defined as:

$$r_{n,t} \leftarrow \begin{cases} 0 & t < T \\ \textsc{Evaluate Ensemble}(\mathcal{E}, e_{n,t}) & t = T, \end{cases} \tag{1}$$

where Evaluate Ensemble is a function presented in Algorithm A3 of the appendix. Once episode n is completed, $e_{n,T}$ is then transformed to a macro ensemble $\mathfrak{e}$. The transformation first segments $e_{n,T}$ into macro actions. For example, if the maximum length of each macro action m in $\mathfrak{e}$ is set to three (i.e., $|m| \leq 3, \forall\ m \in \mathfrak{e}$), $[a_{n,1}, a_{n,2}, a_{n,3}]$ is assigned to m_1, $[a_{n,4}, a_{n,5}, a_{n,6}]$ is assigned to m_2, and so on. The transformation is then done by removing the $null$ actions from each m in $\mathfrak{e}$, allowing macros to have various lengths. The complete transformation procedure is illustrated in Fig. A1 of the appendix..

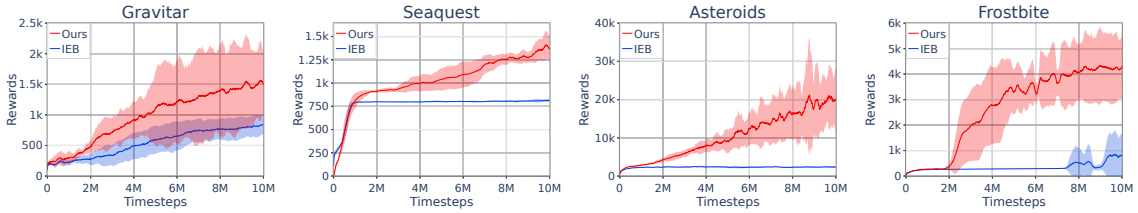

Figure 2: **Comparison of the macro action ensembles constructed by our method and IEB.** The red and the blue lines represent the performances of the agents trained with the macro ensembles constructed by our method and IEB, respectively.

2.2 Construction Phase

Based on the formulation in Section 2.1, we implement $\mathcal{C}$ as a Deep Q-Network (DQN) (Mnih et al., 2015), with an objective to maximize the return in the formulated MDP. The selection of DQN as $\mathcal{C}$ is due to the large state space considered in our work (e.g., 14^T for *Kung-Fu Master*, where 14 is the number of the available primitive actions of it). The overall algorithm is presented in Algorithm A1 of the appendix.

For each episode n, $e_{n,0}$ is first initialized as an empty sequence. For each construction step $t \leq T$ during the construction phase, $\mathcal{C}$ predicts $a_{n,t}$ based on $e_{n,t-1}$ and generate the new action sequence $e_{n,t}$ as described in Section 2.1. The partial transition record $(e_{n,t-1}, a_{n,t}, e_{n,t}, t)$ is then buffered in an evaluation queue $\mathcal{D}$, waiting for the workers to evaluate $r_{n,t}$ before storing the complete transition record $(e_{n,t-1}, a_{n,t}, e_{n,t}, t, r_{n,t})$ to a replay memory $\mathcal{Z}$. The controller $\mathcal{C}$ updates its policy using the data sampled from $\mathcal{Z}$. Please note that $\mathcal{C}$ explores the formulated MDP based on the ϵ-greedy algorithm (Sutton and Barto, 1998), where ϵ linearly decays from 1 to 0.

2.3 Evaluation Phase

During the evaluation phase, $r_{n,t}$ is evaluated by a worker based on Eq. (1) in order to generate the complete transition record from the partial one stored in $\mathcal{D}$. In the case of $t < T$, zero is assigned to $r_{n,t}$. On the other hand, when $t = T$, the worker first transforms the received $e_{n,T}$ to $\mathfrak{e}$, as described in Fig. A1. It then trains an agent using an augmented action space $\mathcal{M} = \mathcal{A} \cup \mathfrak{e}$ to interact with $\mathcal{E}$. After the worker trains the agent for a fixed number of timesteps $\mathcal{H}$, the trained agent is evaluated as described in Algorithm A3 of the appendix. Once the evaluation procedure is finished, the complete transition record $(e_{n,t-1}, a_{n,t}, e_{n,t}, t, r_{n,t})$ is stored into $\mathcal{Z}$. The controller $\mathcal{C}$ is then updated by minimizing the loss function stated in Section B.3 of the appendix using the data samples from $\mathcal{Z}$. Since $\mathcal{C}$ updates its policy by maximizing the overall performance of $\mathfrak{e}$ rather than the performance of each individual macro action within $\mathfrak{e}$, the complete transition records in $\mathcal{Z}$, which reflect effectiveness of $\mathfrak{e}$ as well as *synergism* among the macros in $\mathfrak{e}$, guide the controller to refine its macro ensemble construction policy toward *synergism*.

3. Experimental Results

In this section, we present the experimental results and discuss their implications. We provide a brief introduction to our experimental setup in Section C of the appendix. We

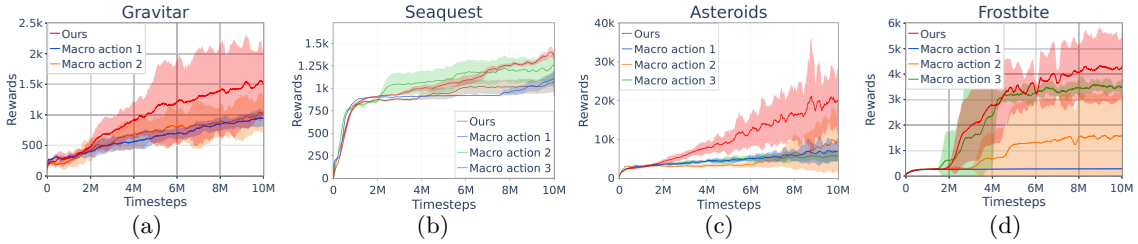

Figure 3: **Comparison of the macro ensembles constructed by our methodology and the decoupled macro actions.** For each environment, the red curve represents the performance of the agent trained with the macro ensemble constructed by our method while the curves with the other colors represent that of the individual macros decoupled from this macro ensemble. The red curve rises faster and ends up with higher performance than the other curves. These results suggest that the macro ensemble derived from our method can lead to higher performance than that corresponding to individual macros in the ensemble, showing the existence of the *synergism* property of the macro ensembles built by our method.

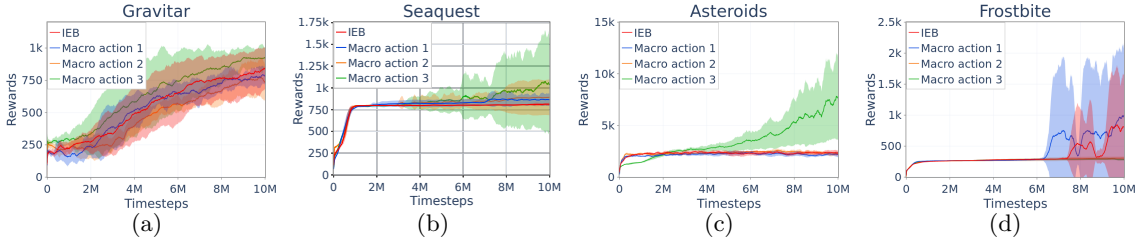

Figure 4: **Comparison of the macro ensembles constructed by IEB and the decoupled macro actions.** For each environment, the red curve represents the performance of the agents trained with the macro ensemble constructed by IEB and the curve of other colors represents that of the individual macro actions decoupled from the macro ensemble. In all environments, the red curves rise slower and end up with lower performance. The results reveal that the macro ensembles constructed by IEB may not possess the *synergism* property.

compare the performance of the macro ensembles constructed by our method against those constructed with an iterative experience based method (Durugkar et al., 2016) in Section 3.1. Then, we further validate the *synergism* property by examining if it exists among the macros in the constructed macro ensembles in Section 3.2. We additionally investigate whether or not the proposed methodology is capable of discovering good macro ensembles for improving the training performance of an RL agent in Section D.1 of the appendix.

3.1 Comparison of our Macro Action Ensemble versus Iterative Experience Based

To demonstrate the effectiveness of our method based on the MDP formulation over the baselines leveraging on agent's experience, we compare the performance of our method against that of IEB. For a fair comparison, both methods construct macro ensembles for identical amount of wall time. After the macro construction process, we evaluate the resultant macro ensembles by training the agents in the target environments with them for the same training timesteps. As shown in Fig. 2 and Table A1 of the appendix, the performances of the macro ensembles constructed our methodology outperform those constructed by IEB.

3.2 Synergism Property

An ensemble of macro actions is said to possess the *synergism* property if the performance achievable by the macro ensemble is higher than the maximum performance of the individual macros within that macro ensemble. To inspect the *synergism* property of the constructed macro ensemble $\mathfrak{e}$, we decouple $\mathfrak{e}$ into multiple $\{m\}, \forall\ m \in \mathfrak{e}$, and compare the performance of agents trained based on $\mathfrak{e}$ against those trained with each decoupled macro $\{m\}$.

3.2.1 Comparison of our Macro Ensembles versus Decoupled Macro Actions

It is observed that the learning curves corresponding to $\mathfrak{e}$ (i.e., *ours* in Fig. 3) grow faster and higher than those corresponding to each $\{m\}$ in Fig. 3. The results show that the macro ensembles constructed by our controller exhibit *synergism* property as expected. In addition to improving performance, our method is able to adjust the number of the macros in an ensemble. For example, there are only two macros within the constructed macro ensemble in *Gravitar*).

3.2.2 Comparison of Macro Action Ensembles constructed by IEB versus Decoupled Macro Actions

We similarly examine the *synergism* property of the macro ensembles constructed by IEB. As shown in Fig. 4, the learning curves corresponding to $\mathfrak{e}$ constructed by IEB grow slower and end up with lower performances than those corresponding to decoupled macro actions. Therefore, the curves reveal that the ensembles constructed by IEB do not benefit from the *synergism* property.

3.2.3 Comparison of Our Method versus IEB

In Table A1 of the appendix., '*Ensemble Perf.*' denotes the performance of the RL agent trained with the constructed macro ensemble, while '*Maximum Perf.*' represents the maximum scores of all the decoupled macro actions in that macro ensemble. In *Space Invaders*, even though the '*Maximum Perf.*' of our method (i.e., 1076.55) is lower than that of IEB (i.e., 1086.10), the '*Ensemble Perf.*' of our method (i.e., 1,368.52) is higher than the '*Ensemble Perf.*' of IEB (i.e., 1,006.45). The above results reveal that the performance of our constructed $\mathfrak{e}$ is influenced by the *synergism* property among the macro actions. In *Beamrider*, the performance of the decoupled macros corresponding to IEB are as good as ours, as shown in Table A1. However, when those macros are combined together as a macro ensemble, the performance corresponding to that macro ensemble drops considerably. The results therefore indicate that the macro ensemble constructed by IEB is not synergistic in *Beamrider*.

4. Conclusions

In this paper, we presented a methodology for constructing macro action ensembles based on the MDP formulation. We proposed a parallel framework with an RL-based controller to generate candidate macro ensembles and evaluate them asynchronously. We evaluated the proposed methodology in a number of *Atari 2600* environments against IEB. We demonstrated that our method is superior to IEB. We further provided analysis and verified

the existence of the *synergism* property among the macro actions contained in the constructed macro ensemble. Moreover, our experimental results validated that the macro ensembles discovered by our method are complementary to the primitive action space, and outperformed this baseline in terms of episode rewards presented in appendix. The results show that the *synergism* property is consequential to the research field of macro action ensemble.

Acknowledgement

This work was supported by the Ministry of Science and Technology (MOST) in Taiwan under grant nos. MOST 109-2636-E-007-018 (Young Scholar Fellowship Program) and MOST 109-2634-F-007-017. The authors acknowledge the financial support from MediaTek Inc., Taiwan. The authors would also like to acknowledge the donation of the GPUs from NVIDIA Corporation and NVIDIA AI Technology Center (NVAITC) used in this research work. Additionally, Yu-Ming Chen acknowledges the fellowship supported by Novatek Inc., Taiwan.

References

P. L. Bacon, J. Harb, and D. Precup. The option-critic architecture. In *Proc. Association for the Advancement of Artificial Intelligence (AAAI)*, Feb. 2016.

B. Baker, O. Gupta, N. Naik, and R. Raskar. Designing neural network architectures using reinforcement learning. In *Proc. Int. Conf. Learning Representations (ICLR)*, Apr. 2017.

M. G. Bellemare, Y. Naddaf, J. Veness, and M. H. Bowling. The arcade learning environment: An evaluation platform for general agents. *J. Artificial Intelligence Research (JAIR)*, 2013.

A. Botea, M. Enzenberger, M. Müller, and J. Schaeffer. Macro-FF: Improving AI planning with automatically learned macro-operators. *J. Artificial Intelligence Research (JAIR)*, 24:581–621, Oct. 2005.

A. Braylan, M. Hollenbeck, E. Meyerson, and R. Miikkulainen. Frame skip is a powerful parameter for learning to play Atari. In *Proc. Association for the Advancement of Artificial Intelligence (AAAI) Conf. Workshop*, Jan. 2015.

Y. Burda, H. Edwards, A. Storkey, and O. Klimov. Exploration by random network distillation. *arXiv preprint arXiv:1810.12894*, 2018.

A. I. Coles and A. J. Smith. Marvin: A heuristic search planner with online macro-action learning. *J. Artificial Intelligence Research (JAIR)*, 28:119–156, Jan. 2007.

P. Dhariwal, C. Hesse, O. Klimov, A. Nichol, M. Plappert, A. Radford, J. Schulman, S. Sidor, Y. Wu, and P. Zhokhov. Openai baselines. `https://github.com/openai/baselines`, 2017.

A. Dulac, D. Pellier, H. Fiorino, and D. Janiszek. Learning useful macro-actions for planning with n-grams. In *Proc. Int. Conf. Tools with Artificial Intelligence (ICTAI)*, pages 803–810, Nov. 2013. doi: 10.1109/ICTAI.2013.123.

I. P. Durugkar, C. Rosenbaum, S. Dernbach, and S. Mahadevan. Deep reinforcement learning with macro-actions. *arXiv:1606.04615*, Jun. 2016.

M. Hauskrecht, N. Meuleau, L. P. Kaelbling, and and C. Boutilier T. Dean. Hierarchical solution of markov decision processes using macro-actions. In *Proc. Conf. Uncertainty in Artificial Intelligence (UAI)*, pages 220–229. Morgan Kaufmann Publishers Inc., 1998.

K. Heecheol, M. Yamada, K. Miyoshi, and H. Yamakawa. Macro action reinforcement learning with sequence disentanglement using variational autoencoder. *arXiv:1903.09366*, May 2019.

N. Heess, G. Wayne, Y. Tassa, T. P. Lillicrap, M. A. Riedmiller, and D. Silver. Learning and transfer of modulated locomotor controllers, 2016.

T. D. Kulkarni, K. Narasimhan, A. Saeedi, and J. B. Tenenbaum. Hierarchical deep reinforcement learning: integrating temporal abstraction and intrinsic motivation. In *Proc. Advances in Neural Information Processing Systems (NeurIPS)*, Dec. 2016.

A. S. Lakshminarayanan, S. Sharma, and B. Ravindran. Dynamic action repetition for deep reinforcement learning. In *Proc. the Thirty-First AAAI Conf. Artificial Intelligence (AAAI-17)*, Feb. 2017.

Hanxiao Liu, Karen Simonyan, and Yiming Yang. Darts: Differentiable architecture search. *ArXiv*, abs/1806.09055, 2018.

A. McGovern, R.S. Sutton, and A. H. Fagg. Roles of macro-actions in accelerating reinforcement learning. In *Proc. Grace Hopper celebration of women in computing*, volume 1317, 1997.

V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, et al. Human-level control through deep reinforcement learning. *Nature*, 518:529–533, Feb. 2015.

M. Newton, J. Levine, and M. Fox. Genetically evolved macro-actions in AI planning problems. In *Proc. the UK Planning and Scheduling Special Interest Group (PlanSIG) Workshop*, pages 163–172. Citeseer, Jan. 2005.

M. A. H. Newton, J. Levine, M. Fox, and D. Long. Learning macro-actions for arbitrary planners and domains. In *Proc. Int. Conf. Automated Planning and Scheduling (ICAPS)*, pages 256–263, Sep. 2007.

H. Onda and S. Ozawa. A reinforcement learning model using macro-actions in multi-task grid-world problems. In *Proc. Int. Conf. Systems, Man and Cybernetics (SMC)*, pages 3088–3093, Oct. 2009.

J. Randlov. Learning macro-actions in reinforcement learning. In *Proc. Advances in Neural Information Processing Systems (NeurIPS)*, pages 1045–1051, Dec. 1999.

J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov. Proximal policy optimization algorithms, 2017.

S. Sharma, A. S. Lakshminarayanan, and B. Ravindran. Learning to repeat: Fine grained action repetition for deep reinforcement learning. In *Proc. Int. Conf. Learning Represenations (ICLR)*, Apr.–May 2017.

R.S. Sutton and A. G. Barto. *Reinforcement Learning: An Introduction*. MIT Press, Cambridge, MA, USA, 1st edition, 1998. ISBN 0262193981.

R.S. Sutton, D. Precup, and S. Singh. Between MDPs and semi-MDPs: A framework for temporal abstraction in reinforcement learning. *Artificial Intelligence*, 112(1-2):181–211, Aug. 1999.

A. Vezhnevets, V. Mnih, S. Osindero, A. Graves, O. Vinyals, J. Agapiou, et al. Strategic attentive writer for learning macro-actions. In *Proc. Advances in Neural Information Processing Systems (NeurIPS)*, pages 3486–3494, Dec. 2016.

T. Yoshikawa and M. Kurihara. An acquiring method of macro-actions in reinforcement learning. In *Proc. IEEE Int. Conf. Systems, Man, and Cybernetics (SMC)*, pages 4813–4817, Nov. 2006.

B. Zoph and Q. V. Le. Neural architecture search with reinforcement learning. In *Proc. Int. Conf. Learning Representations (ICLR)*, Apr. 2017.

Appendix A. Related Works

With a similar concept to macro actions, early works McGovern et al. (1997); Randlov (1999); Braylan et al. (2015) repeat the same primitive actions for multiple timesteps, where the number of repetition is required to be predefined. The later works relax human priors Vezhnevets et al. (2016); Lakshminarayanan et al. (2017); Sharma et al. (2017) by further substituting the fixed-length action repetition with adaptive ones. Nevertheless, action repetition is unlikely to result in delicate behaviors.

Previous works attempt to combine multiple different primitive actions to form a macro action, whereas these works rely on either domain knowledge (Heecheol et al., 2019) or structural assumptions about planners (Botea et al., 2005; Coles and Smith, 2007). Moreover, researchers have investigated approaches that construct macro actions grounded on frequently used sequences of primitive actions from the experience of an agent Durugkar et al. (2016); Dulac et al. (2013); Yoshikawa and Kurihara (2006); Onda and Ozawa (2009). Nonetheless, such approaches are sensitive to the quality of experience. In addition, evolutionary strategies have also been employed to search for useful macro actions Newton et al. (2005, 2007). Unfortunately, these strategies rely on utility functions entailing domain knowledge.

On the other hand, option discovery methods (Hauskrecht et al., 1998; Kulkarni et al., 2016; Bacon et al., 2016; Heess et al., 2016; Vezhnevets et al., 2016) also seek to reduce the the number of decisions required to be made by master policy. Options are usually defined as subroutines required for accomplishing a specific task, which is executed until the termination condition is met. Despite the similarity in an abstract sense, these works substantially differ from our focused topic (i.e., macro action ensemble or simply 'macro ensemble'), since options re-make decisions at every timestep. On the contrary, no additional decision will be made after a macro action is chosen to take. This difference diverges option-based approaches and macro actions into two research directions with their own niches.

Appendix B. Background

In this section, we start by briefly reviewing the formulations of Markov Decision Process (MDP) Sutton and Barto (1998) and Reinforcement Learning (RL) Sutton et al. (1999), then explain Deep Q-network (DQN) Mnih et al. (2015) which is used in our macro ensemble construction process and finally formalize macro actions.

B.1 Markov Decision Process

An MDP consists of a state space $\mathcal{S}$ that contains all possible circumstances of an environment $\mathcal{E}$, a primitive action space $\mathcal{A}$, and a reward function $\mathcal{R} : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$. In an MDP, a controller perceives a state $s_t \in \mathcal{S}$, takes an action $a_t \in \mathcal{A}$ according to its policy $\pi : \mathcal{S} \to \mathcal{A}$, receives a reward $r_t = \mathcal{R}(s_t, a_t)$ as feedback, and a transition to a next state s_{t+1} determined by $\mathcal{E}$ at each discrete timestep t.

B.2 Reinforcement Learning

The goal of RL is to search for the optimal policy π^* in $\mathcal{E}$ characterized by MDP. An RL-based controller performs episodes of a task and iteratively updates its π to search for π^*

via collections of transition record (s_t, a_t, r_t, s_{t+1}), where π^* maximizes the expected return $G_t = \mathbb{E}\big[\sum_{\tau=t}^{T} \gamma^{\tau-t} r_\tau\big]$ within an episode. The discount factor γ can be used to represent the controller's extent of preference for short-term rewards over long-term ones. The horizon T stands for the length of one episode in $\mathcal{E}$.

B.3 Deep Q-network

Deep Q-network (DQN) Mnih et al. (2015) approximates $\mathbb{E}\big[\sum_{\tau=t}^{T} \gamma^{\tau-t} r_\tau \big| s_t, a_t\big]$ by a parameterized function Q_θ to update π, where θ denotes the weights of the neural network. The optimal $Q_{\theta^*}(s_t, a_t)$ estimates $\mathbb{E}\big[r_t + \gamma \max_{a'} Q_{\theta^*}(s_{t+1}, a') | s_t, a_t\big]$. The optimal weights θ^* can be found by minimizing the loss function $L(\theta)$ with respect to the current θ:

$$L(\theta) = \mathbb{E}_{(s_t, a_t, r_t, s_{t+1}) \sim U(\mathcal{Z})} \Big[\big(r_t + \gamma \max_{a'} Q_\theta(s_{t+1}, a') - Q_\theta(s_t, a_t) \big)^2 \Big], \tag{A1}$$

where the $U(\mathcal{Z})$ stands for an uniform distribution over the replay memory $\mathcal{Z}$ that stores the controller's transition records. Based on the learned Q_θ function, the controller takes $\pi(s_t) = \arg\max_a Q_\theta(s_t, a)$ as its policy.

B.4 Macro Action and Macro Action Ensemble

A macro action m is an open-loop policy and is defined as a finite sequence of primitive actions, where $m = [a_1, \cdots, a_k]$, and k is the length of m. Similar to taking a primitive action, an agent can atomically execute a macro action, where the only difference is the resultant transition. Once a macro m is selected for s_t, each primitive action $a \in m$ is then executed sequentially for the following timesteps t to $t + k$, instead of re-deciding actions based on the subsequent states. At the end of m, the agent transitions to s_{t+k} and receives $\sum_{i=1}^{k} r_{t+i-1}$.

A macro ensemble $\mathfrak{e}$ is defined as a set of macros, expressed as $\mathfrak{e} = \{m_1, m_2, \ldots, m_\omega\}$, where ω is an arbitrarily non-negative integer. To equip an agent with $\mathfrak{e}$ in an MDP, the primitive action space $\mathcal{A}$ is augmented by $\mathfrak{e}$ in that MDP, where the augmented action space $\mathcal{M}$ is expressed as $\mathcal{M} = \mathcal{A} \cup \mathfrak{e}$.

Appendix C. Experimental Setup

We first present the environments used in our experiments in Section C.1, followed by a brief description of the baseline for comparison purpose in Section C.2. Next, we describe our controller architecture as well as the hyperparameters in Section C.3. Finally, we explain the setup of the RL agents for the evaluation phase in Section C.4. The curves presented in the experiments are generated based on five random seeds and drawn with 95% confidence interval as the shaded areas.

C.1 Environments

In order to verify our work with various types of environments, we employ eight *Atari 2600* environments: *Asteroids*, *Seaquest*, *Frostbite*, *Gravitar*, *Beamrider*, *Kung-Fu Master*, *Solaris*, and *Space Invaders*, which are distributed among the four categories (Burda et al., 2018) (i.e., *human optimal*, *score exploit*, *dense reward*, and *sparse reward*). We present the learning

$$
\begin{aligned}
e_{n,T} = \quad & [a_{n,1}, \ldots, \boldsymbol{a_{n,\alpha} = null}, \ldots, \boldsymbol{a_{n,\beta} = null}, \ldots, \boldsymbol{a_{n,\gamma} = null}, \ldots, a_{n,T}] \\
\textbf{Step one:} \quad \mathfrak{e} = \quad & \{[a_{n,1}, \ldots, a_{n,\alpha-1}, \boldsymbol{a_{n,\alpha} = null}, a_{n,\alpha+1}, \ldots, a_{n,k}], \\
& [a_{n,k+1}, \ldots, a_{n,\beta-1}, \boldsymbol{a_{n,\beta} = null}, a_{n,\beta+1}, \ldots, a_{n,2k}], \\
& \qquad\qquad \vdots \\
& [a_{n,T-k+1}, \ldots, a_{n,\gamma-1}, \boldsymbol{a_{n,\gamma} = null}, a_{n,\gamma+1}, \ldots, a_{n,T}]\} \\
\textbf{Step two:} \quad \mathfrak{e} = \quad & \{[a_{n,1}, \ldots, a_{n,\alpha-1}, a_{n,\alpha+1}, \ldots, a_{n,k}], \\
& [a_{n,k+1}, \ldots, a_{n,\beta-1}, a_{n,\beta+1}, \ldots, a_{n,2k}], \\
& \qquad\qquad \vdots \\
& [a_{n,T-k+1}, \ldots, a_{n,\gamma-1}, a_{n,\gamma+1}, \ldots, a_{n,T}]\}
\end{aligned}
$$

Figure A1: **Transformation of a decided action sequence $e_{n,T}$ to a macro ensemble $\mathfrak{e}$.** Step one, segment $e_{n,T}$ base on maximum length k of each macro and form $\mathfrak{e}$. Step two, eliminate the *null* actions in $\mathfrak{e}$ (i.e., $a_{n,\alpha}$, $a_{n,\beta}$, and $a_{n,\gamma}$). Please note that each macro in $\mathfrak{e}$ must contains at least two primitive actions, otherwise it is either an empty macro or a duplicate of a primitive action. These types of macros are removed from $\mathfrak{e}$.

Algorithm A1 Macro action ensemble construction algorithm

input: The total number of ensemble searching episodes N
input: The fix length of an episodes T
input: The target Environment $\mathcal{E}$
input: A Worker Pool $\mathcal{W}$
output: Best constructed macro ensemble $\mathfrak{e}$
Initialize a Controller $\mathcal{C}$ with random weights
Initialize an empty Replay Memory $\mathcal{Z}$
Initialize an empty Evaluation Queue $\mathcal{D}$
Launch Parallel Evaluation($\mathcal{E}$, $\mathcal{W}$, $\mathcal{Z}$, $\mathcal{D}$)
for $n = 1, 2, ..., N$ **do** // the n^{th} ensemble
    Initialize empty sequence $e_{n,0}$
    for $t = 1, 2, ..., \mathrm{T}$ **do**
        $a_{n,t} \leftarrow \begin{cases} \text{Select a random action } a, & \text{with probability } \epsilon \\ \text{Action predicted by } \mathcal{C} \text{ using } e_{n,t-1}, & \text{with probability } 1-\epsilon \end{cases}$
        $e_{n,t} \leftarrow e_{n,t-1} \parallel a_{n,t}$
        // $r_{n,t}$ is evaluated in Parallel Evaluation
        Push partial transition record $(e_{n,t-1}, a_{n,t}, e_{n,t}, t)$ into $\mathcal{D}$
    end for
    // $\mathcal{Z}$ filled by Parallel Evaluation
    Optimize $\mathcal{C}$ by sampling mini-batches from $\mathcal{Z}$ using DQN method
end for
$\imath = \arg\max_n \{r_{n,T} \in (e_{n,T-1}, a_{n,T}, e_{n,T}, T, r_{n,T}) \mid \forall\ (e_{n,T-1}, a_{n,T}, e_{n,T}, r_{n,T}) \in \mathcal{Z}\}$
Transform $e_{i,T}$ into macro ensemble $\mathfrak{e}$

curves of the following four environments *Asteroids*, *Seaquest*, *Frostbite*, and *Gravitar* in Figs. 2, 3 and 4. The results of the remaining environments are reported in Table A1.

Algorithm A2 Parallel Evaluation

input: Environment $\mathcal{E}$;
input: Worker Pool $\mathcal{W}$;
input: Replay Memory $\mathcal{Z}$;
input: Evaluation Queue $\mathcal{D}$;
function Parallel Evaluation($\mathcal{E}$, $\mathcal{W}$, $\mathcal{Z}$, $\mathcal{D}$)
    while True **do**
        Wait for an available worker in $\mathcal{W}$
        Pop $(e_{n,t-1}, a_{n,t}, e_{n,t}, t)$ from $\mathcal{D}$
        Run Worker Routine($\mathcal{E}$, $e_{n,t-1}, a_{n,t}, e_{n,t}, t$, $\mathcal{Z}$) asynchronously with the available worker
    end while
end function

function Worker Routine($\mathcal{E}$, $e_{n,t-1}, a_{n,t}, e_{n,t}, t$, $\mathcal{Z}$)
    Duplicate $\mathcal{E}$ as $\mathcal{E}'$
    if $t == T$ **then**
        $r_{n,t}$ = Ensemble Evaluation($\mathcal{E}', e_{n,t}$)
    else
        $r_{n,t} = 0$
    end if
    Store transition record $(e_{n,t-1}, a_{n,t}, e_{n,t}, t, r_{n,t})$ into $\mathcal{Z}$
end function

Algorithm A3 Ensemble Evaluation

input: Environment $\mathcal{E}$
input: Decided Action Sequence e;
output: Reward $\hat{r}$
function Ensemble Evaluation($\mathcal{E}$, e)
    Reset Environment $\mathcal{E}$
    Transform e into macro ensemble $\mathfrak{e}$
    $\mathcal{M} \leftarrow \mathcal{A} \cup \mathfrak{e}$
    Learn a policy ν over $\mathcal{M}$ in $\mathcal{E}$ for $\mathcal{H}$ timesteps
    Evaluate 100 episodes with the policy ν
    Eliminate the highest 10 episodes rewards and lowest 10 episodes
    return Average reward $\hat{r}$ of the rest 80 episodes
end function

C.2 Baselines

To evaluate the constructed $\mathfrak{e}$, we select a state of the art algorithm, proximal policy optimization (PPO) (Schulman et al., 2017), and follow the default hyperparamters described in (Dhariwal et al., 2017) for training the agents on the provided augmented action space $\mathcal{M} = \mathcal{A} \cup \mathfrak{e}$, and compare our proposed method with the following baselines.

Iterative Experience Based (IEB). We study the iterative experience based method (IEB) (Durugkar et al., 2016) by comparing the performance of an agent trained with the macro ensemble constructed by it. The method constructs a macro ensemble based on the most frequently used action sequences during the macro ensemble construction procedure.

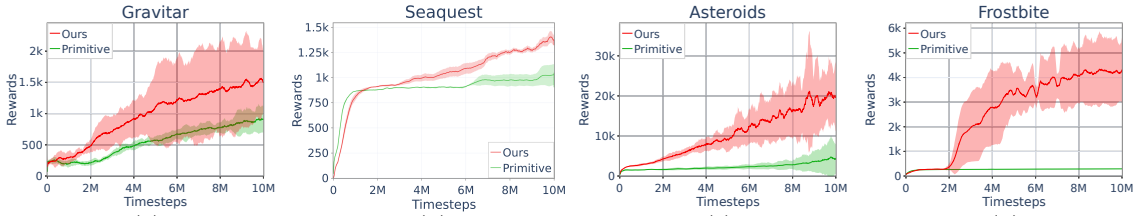

Figure A2: **Comparison of our methodology with the *primitive action* baseline.** "Ours" (red line) represents the performance of an agent trained with the macro ensemble constructed by our method. "Primitive" (green line) represents the performance of an agent trained with the primitive actions in the target environment $\mathcal{E}$. This figure shows that our method is able to construct efficacious macro ensemble.

Primitive action. We additionally compare the performance of the agents trained on $\mathcal{M}$ against those trained on $\mathcal{A}$ to demonstrate that $\mathcal{M}$ is superior to $\mathcal{A}$. The results are presented in Section D.1.

C.3 Controller Setup

Our controller $\mathcal{C}$ is a DQN agent whose objective is to generate a synergistic $\mathfrak{e}$. We modify the architecture of Nature-DQN (Mnih et al., 2015) by removing the convolutional layer and keeping the fully connected layer as well as the output layer. The maximal number and the maximal length of the macros in $\mathfrak{e}$ are configured to pre-defined values $|\mathfrak{e}| = 3$ and $|m| = 3$, respectively. The configuration is based on the optimal setting of the macro length in (Durugkar et al., 2016) as well as the reasonable time budget for analyzing the *synergism* property.

C.4 Setup of the RL Agents for Evaluation

The default RL agent employed in this work is set to PPO. The RL agents are trained by the parallel workers illustrated in Fig. 1. Each worker receives a macro ensemble $\mathfrak{e}$ from $\mathcal{C}$, and trains an RL agent with $\mathfrak{e}$ for 5M timesteps. The macro ensemble is then evaluated for 100 episodes according to Algorithm A3.

Appendix D. Additional Experimental Results

In this section, we present the additional experimental results and the complete version of table.

D.1 Comparison of our Macro Action Ensemble versus Primitive Actions

In this section, we compare the performance of the RL agents trained with the macro ensembles constructed by our proposed method for 10M timesteps against the *primitive action baseline* described in Section C.2. The results are depicted in Fig. A2. It is observed that the RL agents trained with our macro ensembles perform better than the *primitive action* baseline for all of the four environments. The learning curves of the above cases also reveal that the macro ensembles constructed by the proposed method is complementary to the primitive action space, and do lead to higher episode rewards with same timesteps.

Table A1: The '*Maximum Perf.*' column represents the maximum score of all the performances corresponding to the decoupled macros. The '*Ensemble Perf.*' column represents the performance of the constructed macro ensemble. The '*Improvement*' represents the improvements of '*Ensemble Perf.*' over '*Maximum Perf.*'. The *synergism* property exists among the macros if the score in the column of '*Ensemble Perf.*' is higher than that in the column of '*Maximum Perf.*' in the same row.

Atari 2600	Method	**Macro action 1** m_1	**Macro action 2** m_2	**Macro action 3** m_3	**Maximum** $max\{m_1, m_2, m_3\}$	**Ensemble Perf.**	**Improvement**
Beamrider	Ours	3758.06 (405.62)	3996.31 (476.38)	N/A	3996.31 (476.38)	4165.74 (372.66)	4.24%
	IEB	3739.30 (306.27)	3653.49 (405.27)	2699.69 (173.01)	3739.30 (306.27)	1722.42 (584.84)	-53.94%
Kung-Fu Master	Ours	42485.00 (3921.55)	34240.31 (6655.82)	33829.44 (5635.99)	42485.00 (3921.55)	44453.20 (4955.30)	4.63%
	IEB	46255.84 (5966.10)	38848.04 (1184.16)	38914.63 (2347.86)	46255.84 (5966.10)	38236.28 (1803.30)	-17.34%
Seaquest	Ours	1105.22 (204.40)	1262.94 (399.28)	1072.03 (263.65)	1262.94 (399.28)	1362.75 (152.32)	7.90%
	IEB	865.28 (59.28)	891.97 (179.63)	1037.70 (461.58)	1037.70 (461.58)	811.42 (11.01)	-21.81%
Space Invaders	Ours	713.95 (627.45)	1076.55 (131.65)	642.19 (521.62)	1076.55 (131.65)	1368.52 (159.05)	27.12%
	IEB	1086.10 (127.80)	952.24 (78.32)	974.43 (87.87)	1086.10 (127.80)	1006.45 (90.95)	-7.33%
Asteroids	Ours	7105.89 (2432.10)	10180.60 (7857.42)	5589.20 (1049.35)	10180.60 (7857.42)	19685.48 (7421.24)	93.36%
	IEB	2187.89 (215.14)	2274.65 (133.90)	7620.44 (3946.70)	7620.44 (3946.70)	2349.21 (189.27)	-69.17%
Gravitar	Ours	937.76 (103.74)	1011.37 (155.23)	N/A	1011.37 (155.23)	1496.22 (254.70)	47.94%
	IEB	783.26 (50.80)	730.39 (119.08)	925.10 (68.26)	925.10 (68.26)	842.57 (138.18)	-8.92%
Frostbite	Ours	282.88 (11.95)	1594.20 (1488.34)	4010.67 (739.10)	4010.67 (739.10)	4298.52 (1151.88)	7.18%
	IEB	998.33 (1011.80)	308.29 (20.56)	283.63 (14.15)	998.33 (1011.80)	812.71 (747.94)	-18.59%

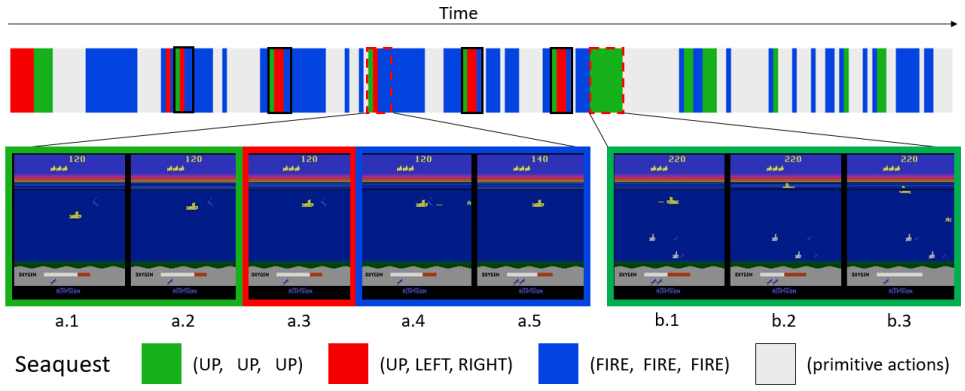

Figure A3: **Clip of history action sequence of *Seaquest*.** The top bar shows a trajectory in *Seaquest* with green representing three consecutive `UP` macro, red representing (`UP`, `LEFT`, `RIGHT`) macro, and blue representing three consecutive `FIRE` macro. The white segment represents the action seqeuences composed of primitive actions. The macro actions and primitve actions are able to be jointly and interlevely used by a RL agent, which demonstrates the *synergism* property empirically.

D.1.1 Qualitative Analysis

To further explain the *synergism* property, we scrutinize the clip of action sequence composed of the initial 200 timesteps from the history of evaluation in *Seaquest*, and illustrate it in Fig. A3. The objectives of this environment are to rescue the victims under the sea with a submarine and eliminate the enemies. The agent in the first two screenshots `a.1` and `a.2` shown in Fig. A3 executes the first macro (highlighted as the green rectangle) composed of three consecutive `UP`, which brings the agent to the same level as the victims in the sea. The next screenshot corresponds to the second macro (highlighted as the red rectangle) which is composed of `UP`, `LEFT`, and `RIGHT` moves, illustrating the agent's intention to rescue the victims. Then, the agent executes the third macro (highlighted as the blue rectangle) which is composed of three consecutive `FIRE` actions to deal with the new enemy appeared

in the screenshot `a.4` of Fig. A3. The screenshot `a.5` shows that the agent has successfully rescued the victim and eliminated the enemy. The above pattern of macro actions (depicted as the black rectangles) frequently appears in the action sequence shown in Fig. A3. This observation implies that the agent is able to utilize the constructed macro actions in a synergistic manner. The three screenshots `b.1`, `b.2` and `b.3` in Fig. A3 further show that the agent is able to bring the victim to the top of the water by the first macro and accomplish the objective. The above interpretations explain the *synergism* property of the constructed macro ensemble. Since the primitive actions (depicted as white segments) as well as the macro actions are interleavedly used in the action sequence, Fig. A3 also justifies that the macro actions are compatible to the primitive action space.