TOWARD SYNERGISM IN MACRO ACTION ENSEMBLES

Anonymous authors
Paper under double-blind review

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 as a whole. The formulation of a sequential decision problem enables coherency in the macro actions to be considered. Also, such an evaluation procedure takes *synergism* into account since the *synergism* among the macro action ensemble exhibits when jointly used by an agent. Our experimental results show that our framework is able to discover synergistic macro action ensembles. We further perform a series of experiments to validate the *synergism* property among the macro action ensemble.

1 INTRODUCTION

Through the history of artificial intelligence, "*macro action*" (or simply "*macro*") has long been a prevalent tool (McGovern et al., 1997; McGovern & Sutton, 1998; Xu et al., 2018; Onda & Ozawa, 2009; Braylan et al., 2015) to reduce the efforts on searching the optimal policy in sequential decision problems. A macro action is executed by an agent atomically, where it is typically defined as a sequence of primitive actions. Once a macro is decided, the agent then sequentially takes the primitive actions within it without remaking decisions. Macro actions can be used to reduce the number of decisions required to be made. Recent works have shown that macro actions are beneficial for the learning processes in a range of environments. Unfortunately, discovering efficacious macro actions from an intractable number of possible combinations of primitive actions is far from straightforward.

A number of previous works have been devoted to developing macro actions construction procedures (Yoshikawa & 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 to solving complex tasks. Incompatible macros may fail to synthesize required behaviors for solving such tasks. In addition, simply adding many macro actions degrades search performance, because not all macros are always useful for every situations. Some are too specialized and might confuse the agent in other occasions. Hence, *synergism* among the macro actions emerges to be crucial for solving complex tasks. 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 the *synergism* property among the macros. Unfortunately, an appropriate approach for constructing such synergistic 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 & 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 layer is decided by an RL-based controller at each step and 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 focus on searching 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

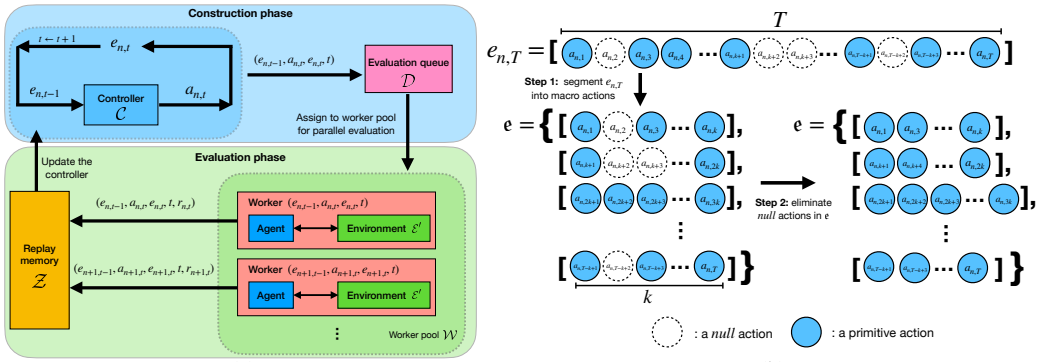

Figure 1: (a) Overview of the proposed ensemble searching framework. (b) **An illustration of the transformation procedure from a decided action sequence $e_{n,T}$ to a macro ensemble $\mathfrak{e}$.** The entire transformation procedure consists of two steps. The first step segments $e_{n,T}$ based on a maximum length k of each macro and forms $\mathfrak{e}$. The second step eliminates actions whose values are $null$ in $\mathfrak{e}$.

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 whole generated macro ensemble is evaluated by an agent in a target task, and then the feedback is returned to the controller. The resultant evaluation performances of the generated macro ensembles, which reflect effectiveness as well as *synergism*, can then be utlized to guide the controller to refine its macro action 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 methodology grounded on RL and a parallel asynchronous framework to accelerate the construction procedure. We then validate the effectiveness of our methodology by conducting experiments on a range of *Atari 2600* (Bellemare et al., 2013) environments. The qualitative and quantitative 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 macros.

The remainder of this paper is organized as the following. Section 2 briefly reviews the related works. Section 3 describes the proposed methodology. Section 4 presents our experimental results. Section 5 concludes this paper. The provided appendix is organized as follows. Section A1 introduces the background material. Section A2 provides additional experimental results.

2 Related Work

In this section, we briefly review the previous methods for generating macros, which can be broadly categorized as the following: repeated actions, heuristic macros, and frequently used action sequences.

The first category embraces a similar concept as macro actions. Early works of this category (McGovern et al., 1997; Randlov, 1999; Braylan et al., 2015) repeat the same primitive actions for multiple timesteps, where the length of repetition is required to be pre-defined. The more recent 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. However, policies that rely only on macros consisting of repeated actions are inherently not appropriate for generating delicate behaviors.

For the second category, previous works attempt to combine 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., 2005a; Coles & Smith, 2007). Evolutionary strategies (Newton et al., 2005; 2007) have also been used to search for useful macro actions. Unfortunately, these strategies still require utility functions entailing domain knowledge, while the primary focus and the scope of discussion considered in this paper are not relying on such knowledge.

For the third category, 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 & Kurihara, 2006; Onda & Ozawa, 2009). Nonetheless, such approaches are sensitive to the quality of experience. Please note that there are two approaches introduced in (Durugkar et al., 2016): the frequency-based and the repeated-action based, where

the former is more general and includes the search space of the latter. Therefore, we select the frequency-based approach (Durugkar et al., 2016) as our primary baseline.
Please note that our work focus on macro construction in the RL domain, which is different from the planning domain (Korf, 1985; Botea et al., 2005a; Asai & Fukunaga, 2015; Chrpa & Vallati, 2019). The environment dynamics of the latter is accessible to a planner and allow the planner to extract, filter, and analyze the macros (Botea et al., 2005a;b; 2007). The achiever/requirer relationship (i.e., inner/outer entanglement) (Chrpa et al., 2014; 2019) among macros could also be clearly defined in the planning domain. In contrast, an RL agent solves a problem without accessing environment dynamics, where methods of constructing and analyzing macros based on planning are not directly applicable. In this paper, we investigate an effective method for constructing a synergistic macro ensemble in the absence of accessible environment dynamics.

3 METHODOLOGY

Our proposed methodology constructs macro ensembles using an asynchronous parallel 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. In the construction phase, $\mathcal{C}$ constructs macro ensembles and sends them to $\mathcal{W}$. During the evaluation phase, each worker in $\mathcal{W}$ is implemented as an agent for evaluating 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. Fig. 1 (a) illustrates the complete framework of the proposed method. The pseudo-code of our method is provided in Algorithm 1.
To elaborate on our methodology, in the following subsections, we first formulate the proposed macro ensemble construction process as an MDP in Section 3.1. We then walk through the construction and evaluation phases of the proposed methodology in detail in Section 3.2 and Section 3.3, respectively.

3.1 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, $\mathcal{C}$ 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 expressed 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 $r_{n,t}$ 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 A1 of the supplymentary material. 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 macros. For example, if the maximum length of each macro 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. 1 (b). Please note that each macro in $\mathfrak{e}$ must contain at least two primitive actions, otherwise it is either an empty macro or a duplicate of primitive action. Therefore, such macros are removed from $\mathfrak{e}$.

3.2 CONSTRUCTION PHASE

Based on the formulation in Section 3.1, 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). The algorithm of our method is offered in Algorithm 1.
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 decided action sequence $e_{n,t}$ as formulated in Section 3.1. The partial transition record $(e_{n,t-1}, a_{n,t}, e_{n,t}, t)$ is then buffered in the 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 & Barto, 1998), where the value of ϵ linearly decays from 1.0 to 0.0.

Algorithm 1 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
i = $\arg\max_n \{r_{n,T} \in (e_{n,T-1}, a_{n,T}, e_{n,T}, T, r_{n,T}) \mid \forall\, (e_{n,T-1}, a_{n,T}, e_{n,T}, r_{n,T}) \in \mathcal{Z}\}$
Transform $e_{i,T}$ into macro ensemble $\mathfrak{e}$

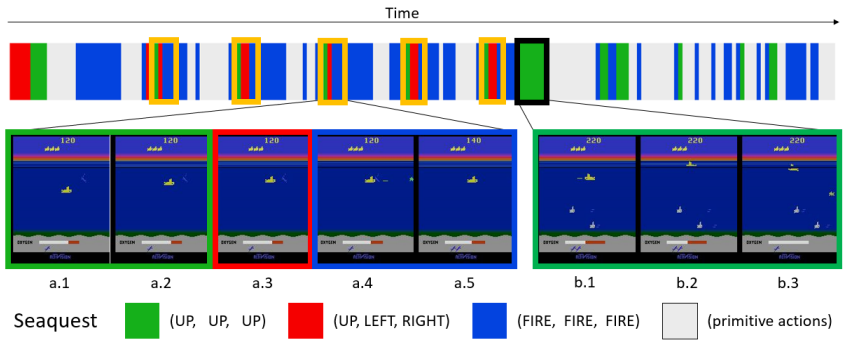

Figure 2: **A motivational example for demonstrating the *synergism* property in *Seaquest*.** The top bar shows a clip of the trajectory captured in *Seaquest*. The green color represents three consecutive `UP` macro, the red color represents (`UP`, `LEFT`, `RIGHT`) macro, and the blue one represents three consecutive `FIRE` macro. The white segments correspond to the uses of primitive actions. The macro actions and the primitive actions can be jointly and interlevely used by an RL agent, demonstrating the *synergism* property described in Section 4.2.

3.3 EVALUATION PHASE

During the evaluation phase, $r_{n,t}$ is evaluated by a worker based on Eq. equation 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. 1 (b). It then trains an agent using 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 A1 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 A1.3 of the appendix using the data samples from $\mathcal{Z}$. Since the controller $\mathcal{C}$ 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 macro actions in $\mathfrak{e}$, can guide the controller $\mathcal{C}$ to refine its macro action ensemble construction policy toward *synergism*.

4 EXPERIMENTAL RESULTS

In this section, we present the experimental results and discuss their implications. We start by a brief introduction to our experimental setup in Section 4.1, and provide a motivational example in

Algorithm 2 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 $w \in \mathcal{W}$
        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 by w
    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($\mathcal{E}', e_{n,t}$) // Algorithm A1 of the supplementary material
    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

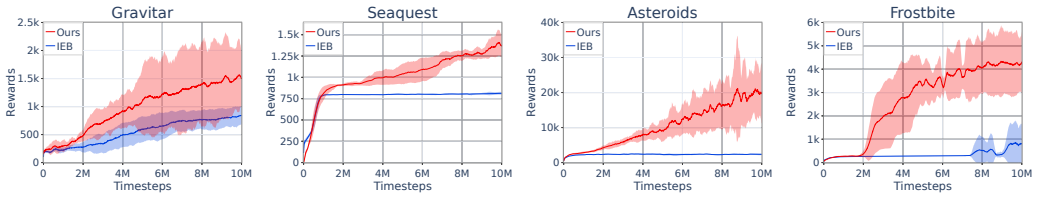

Figure 3: **Comparison of the macro action ensembles constructed by our method and IEB.** The red and the blue learning curves correspond to the agents trained with the macro ensembles constructed by our methodology and IEB, respectively. The results of the remaining environments are summarized in the appendix.

Section 4.2. We next 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 4.3. Finally, we further validate the *synergism* property by examining if it exists among the macros in the constructed macro ensembles in Section 4.4. Additional experimental results for comparing the performances of our method against the other baselines can be referred in Section A2.

4.1 EXPERIMENTAL SETUP

We first present the environments used in our experiments in Section 4.1.1. Next, we describe our controller architecture as well as the hyperparameters in Section 4.1.2. We then explain the setup of the RL agents for the evaluation phase in Section 4.1.3. Lastly, we provide a brief description of the baseline for comparison purpose in Section 4.1.4. The curves presented in the experiments are generated based on five random seeds and drawn with 68% confidence interval as the shaded areas.

4.1.1 ENVIRONMENTS

We verify our methodolody in nineteen *Atari 2600* environments: *Alien*, *Assault*, *Asteroids*, *Beamrider*, *Breakout*, *Chopper Command*, *Crazy Climber*, *Enduro*, *Frostbite*, *Gravitar*, *Krull*, *Kung-Fu Master*, *Q*bert*, *Seaquest*, *Space Invaders*, *Solaris*, *Up'n Down*, *Venture*, and *Zaxxon*, which are distributed among the four categories with different difficulty levels (Bellemare et al., 2016) (i.e., *human optimal*, *score exploit*, *dense reward*, and *sparse reward*). We plot the learning curves of the following environments in Figs. 3, 4, and 5: *Asteroids*, *Seaquest*, *Frostbite*, and *Gravitar*. The results of the rest environments are offered in Table 1 and detailed in Table A2 of the appendix.

4.1.2 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

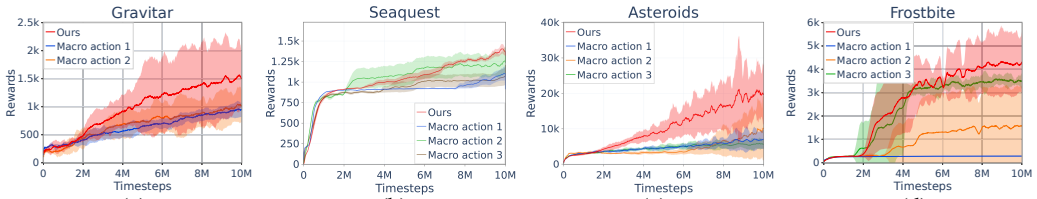

Figure 4: **Comparison of the macro ensembles constructed by our methodology and the decoupled macro actions.** For each environment, the red curve represents the performance of the agent trained with the macro ensemble constructed by our methodology, while the curves in other colors represent the performance corresponding to the individual macros decoupled from this macro ensemble. For each case, the red curve grows faster and ends up with a higher performance than the other curves. These results validate the existence of the *synergism* property of our macro ensembles. Additional results can be referred in the appendix.

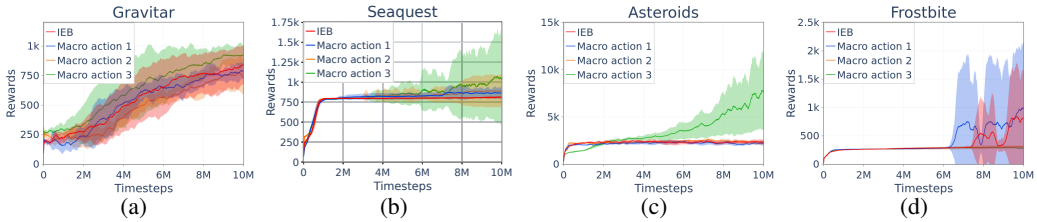

Figure 5: **Comparison of the macro ensembles constructed by IEB and the decoupled macro actions.** For each environment, the red curve represents the performance of the agent trained with the macro ensemble constructed by IEB, while the curves in other colors represent the performance corresponding to the individual macro actions decoupled from this macro ensemble. For each case, the red curve ends up with a lower performance than the best curve. The results reveal that the macro ensembles constructed by IEB may not possess the *synergism* property. Additional experimental results can be referred in the appendix.

fully connected (FC) layer as well as the output layer. The maximal length and the maximal number of the macro actions in $\mathfrak{e}$ are configured to $|m| = 3$ and $|\mathfrak{e}| = 3$, respectively. The configuration is based on the optimal setting of the macro action length in (Durugkar et al., 2016) as well as for the purpose of presenting the *synergism* property in a clear manner.

4.1.3 Setup of the RL Agents for Evaluation

We select a state of the art algorithm, proximal policy optimization (PPO) (Schulman et al., 2017), and follow its default hyperparameters for training our default RL agents. The RL agents used by our methodology are trained by the parallel workers illustrated in Fig. 1 (a). Each worker receives a macro ensemble $\mathfrak{e}$ from $\mathcal{C}$, and trains an RL agent with the augmented action space $\mathcal{M} = \mathcal{A} \cup \mathfrak{e}$ for 5M timesteps. The macro ensemble is then evaluated for 100 episodes according to Algorithm A1. To evaluate the constructed $\mathfrak{e}$, we train the agents on the provided augmented action space $\mathcal{M} = \mathcal{A} \cup \mathfrak{e}$ for additional 10M timesteps, and compare our method with the baseline described in Section 4.1.4.

4.1.4 Iterative Experience Based Baseline

Our baseline is an iterative experience based method (IEB) (Durugkar et al., 2016), which adopts a framework that iteratively constructs macro ensembles from the past trajectories of an agent. IEB selects the most frequently used fixed-length action sequences from these trajectories to form $\mathfrak{e}$ to be used by the RL agent in next evaluation phase. The RL agent is set to default, as described above.

4.2 Motivational Example of the Synergism Property

To provide a motivational example of the *synergism* property, we first illustrate a clip of the trajectory captured from *Seaquest* for the initial 200 timesteps in Fig. 2. The objective of this environment is to control a submarine to rescue the victims under the sea and eliminate the enemies. The agent in the first two screenshots `a.1` and `a.2` executes the first macro (highlighted by the green rectangle) composed of three consecutive `UP`, which brings the submarine toward the same level as the victim in the sea. The next screenshot `a.3` corresponds to the second macro (highlighted by the red rectangle) composed of `UP`, `LEFT`, and `RIGHT` moves, illustrating the agent's intention to rescue the victim. Then, the agent executes the third macro (highlighted by the blue rectangle) composed of three consecutive `FIRE` actions to deal with the new enemy appeared in the screenshot `a.4` of

Table 1: The table summarizes the evaluation results of the agents trained with the macro ensembles as well as the best decoupled macro actions constructed by the proposed methodology and the IEB baseline. The columns '*Best Decoupled Macro Perf.*', '*Ensemble Perf.*', and '*Improvement*' correspond to the scores achieved by the agents trained with the best-performing decoupled macros, the scores achieved by the agents trained with the constructed macro ensembles, and the percentage of improvements in scores provided by the macro ensembles. The standard deviations calculated based on different random seeds are reported in the parentheses. The *synergism* property exists among the macro actions if the percentage reported in the '*Improvement*' column is positive. Please note that the detailed version of this table is offered in Table A2 of the appendix.

Atari 2600	**Method**	**Best Decoupled Macro Perf.**	**Ensemble Perf.**	**Improvement**
Alien	Ours	1498.38 (512.12)	1951.32 (396.42)	30.23%
	IEB	1790.48 (151.17)	1522.42 (93.17)	-14.97%
Assault	Ours	5030.98 (973.60)	5263.31 (169.15)	4.62%
	IEB	3612.56 (246.94)	2593.04 (241.15)	-28.22%
Asteroids	Ours	10180.60 (7857.42)	19685.48 (7421.24)	93.36%
	IEB	7620.44 (3946.70)	2349.21 (189.27)	-69.17%
Beamrider	Ours	3996.31 (476.38)	4165.74 (372.66)	4.24%
	IEB	3739.30 (306.27)	1722.42 (584.84)	-53.94%
Breakout	Ours	N/A (N/A)†	319.74 (42.36)	N/A†
	IEB	200.82 (44.37)	50.62 (3.92)	-74.79%
Chopper Command	Ours	8502.19 (932.13)	11568.96 (1248.61)	36.07%
	IEB	9653.16 (1637.70)	4396.36 (868.64)	-54.46%
Crazy Climber	Ours	113472.61 (4190.81)	111466.74 (3371.71)	-1.77%
	IEB	112424.02 (2533.33)	105851.57 (7493.50)	-5.85%
Enduro	Ours	N/A (N/A)†	830.93 (113.92)	N/A†
	IEB	493.11 (50.01)	322.76 (54.82)	-34.55%
Frostbite	Ours	4010.67 (739.10)	4298.52 (1151.88)	7.18%
	IEB	998.33 (1011.80)	812.71 (747.94)	-18.59%
Gravitar	Ours	1011.37 (155.23)	1496.22 (254.70)	47.94%
	IEB	925.10 (68.26)	842.57 (138.18)	-8.92%
Krull	Ours	8166.66 (355.97)	8663.77 (1164.13)	6.09%
	IEB	6751.43 (174.00)	7427.26 (763.25)	10.01%
Kung-Fu Master	Ours	42485.00 (3921.55)	44453.20 (4955.30)	4.63%
	IEB	46255.84 (5966.10)	38236.28 (1803.30)	-17.34%
*Q*bert*	Ours	N/A (N/A)†	14540.53 (463.21)	N/A†
	IEB	11838.20 (1023.93)	5421.76 (1115.34)	-54.20%
Seaquest	Ours	1262.94 (399.28)	1362.75 (152.32)	7.90%
	IEB	1037.70 (461.58)	811.42 (11.01)	-21.81%
Solaris	Ours	2880.61 (493.98)	3236.40 (238.09)	12.35%
	IEB	2131.12 (366.63)	1856.87 (213.70)	-12.87%
Space Invaders	Ours	1076.55 (131.65)	1368.52 (159.05)	27.12%
	IEB	1086.10 (127.80)	1006.45 (90.95)	-7.33%
Up N Down	Ours	349670.21 (17272.84)	941419.24 (56575.20)	169.23%
	IEB	390699.63 (171652.74)	283856.43 (107352.04)	-27.35%
Venture	Ours	0.00 (0.00)	167.25 (334.51)	N/A‡
	IEB	12.57 (28.11)	0.00 (0.00)	-100.00%
Zaxxon	Ours	8746.27 (748.50)	8856.88 (817.74)	1.26%
	IEB	7975.56 (1629.89)	6657.54 (1686.32)	-16.53%

†: The macro ensembles constructed by our methodology in *Breakout, Enduro and Q*bert* are empty sets, implying that the action space of each environment is too simple to be benefited from macro actions. Thus, there exists no decoupled macro action (i.e., N/A in the *Best Decoupled Macro Perf.* column). The values listed in the *Ensemble Perf* column for *Breakout, Enduro and Q*bert* represent the performances of the agent trained with primitive actions. Since there exists no decoupled macro action in *Breakout, Enduro and Q*bert*, the corresponding *Improvement* are unable to calculate.

‡: In *Venture*, N/A in the *Improvement* column is due to the division by zero. Note that the agent is still benefited from the macro ensemble.

Fig. 2. The screenshot `a.5` shows that the agent has successfully rescued the victim and eliminated the enemy. The above pattern of macro actions (highlighted by the orange rectangle) frequently appears in the trajectory shown in Fig. 2. 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. 2 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) and the

macro actions are interleavedly used in the trajectory, Fig. 2 further justifies that the macro actions are compatible to the primitive action space.

4.3 Comparison of Our Method and the Iterative Experience Based Baseline

To demonstrate the effectiveness of our methodology basing on the MDP formulation described in Section 3 over IEB leveraging on an agent's past experience, we compare the agents trained with the macro ensembles constructed by these two methods. For a fair comparison, both methods construct their macro ensembles for an identical amount of wall time. The resultant macro ensembles constructed by them are then separately re-evaluated by the default RL agents from scratch in the target environments $\mathcal{E}$ for 10M timesteps. We plot the evaluation curves of four environments in Fig. 3, and summarize the remainder of the results in Table A2 of the appendix. It is observed that the macro ensembles constructed by our methodology are able to provide more beneficial impacts on the learning processes of the default RL agents than those constructed by IEB.

4.4 Analysis of the 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. In order 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 the agents trained based on $\mathfrak{e}$ against the performance of the agents trained based on each decoupled macro action $\{m\}$. The *synergism* property exists when the performance corresponding to $\mathfrak{e}$ is better than that of each $\{m\}$.

4.4.1 Our Macro Ensembles versus our Decoupled Macro Actions

It is observed from Fig. 4 that the learning curves corresponding to our macro ensembles $\mathfrak{e}$ (i.e., '*Ours*') grow faster and higher than those corresponding to the decoupled macro actions $\{m\}, \forall\, m \in \mathfrak{e}$. The results show that macro ensembles $\mathfrak{e}$ constructed by our controller exhibit the *synergism* property. In addition to improving performance, our controller is able to adjust the number of macros in an ensemble. For example, the macro ensemble constructed by our controller for *Gravitar* consists of only two macros. This is due to the fact that the third macro contains only the padding $null$ actions, and is thus removed in the evaluation phase. This implies that adding another macro into this macro ensemble may not be beneficial to the construction procedure of a synergistic macro action ensemble.

4.4.2 Macro Ensembles constructed by IEB versus their Decoupled Macros

We similarly examine the *synergism* property of the macro ensembles constructed by IEB. As shown in Fig. 5, the learning curves corresponding to the macro ensembles $\mathfrak{e}$ constructed by IEB end up with lower performances than those corresponding to the decoupled macro actions $\{m\}, \forall\, m \in \mathfrak{e}$. The results reveal that macro ensembles constructed by IEB do not benefit from *synergism* property.

4.4.3 The Impacts of the Synergism Property on Performances

In this section, we examine and compare the impacts of the the *synergism* property on the performances corresponding to our methodology and IEB. In Table 1, '*Ensemble Perf.*' denotes the performances of the RL agents trained with the constructed macro ensembles, while '*Best Decoupled Macro Perf.*' represents the scores corresponding to the best-performing decoupled macro action in the macro ensembles. In *Space Invaders*, even though the '*Best Decoupled Macro 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., 1368.52) is higher than the '*Ensemble Perf.*' of IEB (i.e., 1006.45). The results reveal that the performance of $\mathfrak{e}$ generated by our method is influenced by the *synergism* property among the macros. A detailed version of Table 1 and its analysis are offered in Table A2 of the appendix.

5 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 which relies on the MDP formulation is superior to IEB which leverages on the most frequently used action sequences. 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 compatible to the primitive action space, and outperformed the baselines in terms of episode rewards presented in supplementary. The results show that the *synergism* property is crucial to the research field of macro action ensemble.

REFERENCES

Masataro Asai and Alex Fukunaga. Solving large-scale planning problems by decomposition and macro generation. In *ICAPS*, pp. 16–24, 2015.

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

P.-L. Bacon, J. Harb, and D. Precup. The option-critic architecture. In *Proc. the Thirty-First AAAI Conf. Artificial Intelligence (AAAI-17)*, pp. 1726–1734, Feb. 2017.

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. Bellemare, S. Srinivasan, G. Ostrovski, T. Schaul, D. Saxton, and R. Munos. Unifying count-based exploration and intrinsic motivation. In *Advances in neural information processing systems*, pp. 1471–1479, 2016.

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. 2005a.

Adi Botea, Martin Müller, and Jonathan Schaeffer. Learning partial-order macros from solutions. In *ICAPS*, pp. 231–240, 2005b.

Adi Botea, Martin Müller, Jonathan Schaeffer, et al. Fast planning with iterative macros. In *IJCAI*, pp. 1828–1833, 2007.

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.

Lukáš Chrpa and Mauro Vallati. Improving domain-independent planning via critical section macro-operators. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pp. 7546–7553, 2019.

Lukáš Chrpa, Mauro Vallati, and Thomas Leo McCluskey. Mum: A technique for maximising the utility of macro-operators by constrained generation and use. 2014.

Lukáš Chrpa, Mauro Vallati, and Thomas Leo McCluskey. Inner entanglements: Narrowing the search in classical planning by problem reformulation. *Computational Intelligence*, 35(2):395–429, 2019.

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)*, pp. 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)*, pp. 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.

A. Hill, A. Raffin, M. Ernestus, A. Gleave, R. Traore, et al. Stable baselines. `https://github.com/hill-a/stable-baselines`, 2018.

Richard E Korf. Macro-operators: A weak method for learning. *Artificial intelligence*, 26(1):35–77, 1985.

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 and R. S. Sutton. Macro-actions in reinforcement learning: An empirical analysis. *Computer Science Department Faculty Publication Series*, pp. 15, 1998.

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*, pp. 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)*, pp. 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)*, pp. 3088–3093, Oct. 2009.

J. Randlov. Learning macro-actions in reinforcement learning. In *Proc. Advances in Neural Information Processing Systems (NeurIPS)*, pp. 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)*, pp. 3486–3494, Dec. 2016.

S. Xu, H. Y. Kuang, Z. Q. Zhuang, R. J. Hu, Y. Liu, and H. Y. Sun. Macro action selection with deep reinforcement learning in StarCraft, 12 2018.

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

Appendices

A1 Background Material

In this section, we start by briefly reviewing the formulations of the Markov Decision Process (MDP) Sutton & Barto (1998) and Reinforcement Learning (RL) (Sutton et al., 1999). Then, we explain the Deep Q-Network (DQN) (Mnih et al., 2015) which is used by our controller $\mathcal{C}$ during the macro ensemble construction process. Next, we describe the Proximal Policy Optimization (PPO) algorithm (Schulman et al., 2017), which is used by the worker pool of our methodology for evaluating the performance of a macro action ensemble. Finally, we provide formal definitions and formulations of a macro action as well as a macro action ensemble.

A1.1 Markov Decision Process

An MDP consists of a state space $\mathcal{S}$ that contains all possible states 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 the feedback, and then transitions to a next state s_{t+1} determined by $\mathcal{E}$ at each discrete timestep t.

A1.2 Reinforcement Learning

The goal of RL is to search for an optimal policy π^* in $\mathcal{E}$ characterized by an 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 or long-term ones. The horizon T stands for the length of one episode in $\mathcal{E}$.

A1.3 Deep Q-network

DQN (Mnih et al., 2015) approximates G_t by a parameterized function Q_θ to update π, where θ denotes the weights of a neural network. The optimal $Q_{\theta^*}(s_t, a_t)$ estimates $\mathbb{E}\big[r_t + \gamma \max_{a'} Q_{\theta^*}(s_{t+1}, a')|s_t, a_t\big]$, where the optimal weights θ^* is derived 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 $U(\mathcal{Z})$ stands for an uniform distribution over a replay buffer $\mathcal{Z}$ that stores the controller's transition records, and θ^- denotes the parameters of a target network. The target network is the same as the online network, except that its parameters θ^- are updated by the online network at predefined intervals. According to the learned Q_θ, the controller takes $\pi(s_t) = \arg\max_a Q_\theta(s_t, a)$ as its policy.

A1.4 Proximal Policy Optimization

Algorithm A1 of our work employs PPO (Schulman et al., 2017) as an RL agent responsible for evaluating the constructed macro ensemble because of its accessibility and good performance. PPO computes an update at every timestep that minimizes the cost function while ensuring the deviation from the previous policy is relatively small. One of the two main variants of PPO is a clipped surrogate objective expressed as:

$$L^{CLIP}(\theta) = \mathbb{E}\left[\frac{\pi_\theta(a|s)}{\pi_{\theta_{old}}(a|s)}\hat{A}, \text{clip}\left(\frac{\pi_\theta(a|s)}{\pi_{\theta_{old}}(a|s)}, 1-\epsilon, 1+\epsilon\right)\hat{A}\right], \tag{A2}$$

where $\hat{A}$ represents the advantage estimate, and ϵ is a hyperparameter. In Eq. (A2), the clipped probability ratio is utilized to prevent large changes to the policy between two updates. The other variant of PPO employs an adaptive penalty based on KL divergence, which is expressed as follows:

$$L^{KLPEN}(\theta) = \mathbb{E}\left[\frac{\pi_\theta(a|s)}{\pi_{\theta_{old}}(a|s)}\hat{A} - \beta KL[\pi_{\theta_{old}}(\cdot|s), \pi_\theta(\cdot|s)]\right], \tag{A3}$$

where β is an adaptive coefficient adjusted according to the observed change in the KL divergence. In this work, we follow the implementation of the Stable Baseline (Hill et al., 2018), in which the loss function is implemented as the former objective (i.e., Eq. (A2)) because of its superior empirical performances.

Algorithm A1 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

A1.5 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 transitions. Once a macro action m is selected for execution in s_t, each primitive action $a \in m$ is then executed sequentially from timesteps t to $t + k$, rather than re-deciding primitive actions based on the subsequent states. At the end of m, the agent transitions to s_{t+k} and receives a lump sum of rewards equal to $\sum_{i=1}^{k} r_{t+i-1}$. 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 $\mathfrak{e}$ for that MDP, where the augmented action space $\mathcal{M}$ is expressed as $\mathcal{M} = \mathcal{A} \cup \mathfrak{e}$.

A2 ADDITIONAL EXPERIMENTAL RESULTS

In this section, we first described the detailed experimental setup for our DQN controller and the RL agent used by the worker in Section A2.1. Next, we provide an analysis related to each decoupled macro action in an macro ensemble based on the complete version of Table 1 in Section A2.2. Finally, an additional comparison of our method against option-critic is provided in Table A3 of Section A2.3.

A2.1 ADDITIONAL DETAILS OF THE EXPERIMENTAL SETUP

In this subsection, we illustrate the detail experimental setup of our DQN controller $\mathcal{C}$ and PPO agents used by the workers. The network structure of our DQN controller is modified from (Mnih et al., 2015) by removing the convolutional layers and adjusting the dimension of the output layer to fit the required output. The detailed hyper-parameters of our DQN controller $\mathcal{C}$ could be referred in Table A1. The PPO implementation of our work is based on the default network structure as well as the hyper-parameters described in (Dhariwal et al., 2017; Hill et al., 2018). The hyper-parameters adopted in this work are specified in Table A1 as well. All the results presented are based on five different random seeds, as described in the main manuscript.

A2.2 A COMPLETE TABLE OF THE EVALUATION RESULTS

In this section, we further analyze the *synergism* property with regard to the detailed performances of the agents trained with each decoupled macro action within a macro action ensemble. The results are reported in Table A2.

In most of the environments, the macro ensemble generated by our methodology is able to deliver superior performances to the those corresponding to each individual macro actions, indicating that the *synergism* property does exist. For example, the performances of the decoupled macros corresponding to IEB are as good as *Ours* in *Beamrider*. However, when the agent is trained with the macro ensemble consisting of the the macro actions generated by IEB, its performance drops considerably. The results therefore indicate that the macro ensemble constructed by IEB is not synergistic in *Beamrider*, while the macro ensemble constructed by our method possess the *synergism* property. For *Breakout* and *Venture*, the macro ensemble generated by *Ours* is able to lead to positive scores, while the individual macros and the macro ensemble generated by IEB are unable to. On the other hand, even though the ensemble performance of *Ours* is slightly less than $\max\{m_1, m_2, m_3\}$ in *Crazy Climber*, *Ours* is still comparable to them and superior to the performance corresponding to the macro ensemble generated by IEB. The above observations and the results presented in Table A2 validate that our controller is able to construct macro ensembles possessing the *synergism* property.

Table A1: The detailed settings of the hyper-parameters usd in our experiments.

Hyperparameter	Value
Deep Q-Network (DQN) for our controller $\mathcal{C}$	
Learning rate	$5e{-6}$
Discount factor (γ)	0.99
Batch size	128
Epsilon End	0.1
Replay buffer size	10000
Initialization of the replay buffer	200 transactions
Target update frequency	per episode
Proximal Policy Optimization (PPO) for our worker	
Learning rate	$2.5e{-4}$
Learning rate annealing	constant
Discount factor (γ)	0.99
Batch Size	2560
Rollout length	128
Number of the parallel environments	20
Entropy coefficient	0.01
Value function coefficient	0.5
Weight for the advantage function (λ)	0.95
Epoches per update	4
Number of mini-batch	4
Maximum gradient clip	0.5
Clipping parameter	0.2
Number of frame skip	4

A2.3 Additional Comparison of Our Methodology versus the Option-Critic Architecture

Options (Hauskrecht et al., 1998; Kulkarni et al., 2016; Bacon et al., 2016; Heess et al., 2016; Vezhnevets et al., 2016) are closed-loop temporal abstraction based approaches, which are irrelevant to the open-loop macro actions described in our work. Options are usually defined as subroutines required for accomplishing a specific task, which is executed until the termination condition is met. Despite the difference between the two methods, we provide an additional comparison of our methodology and option-critic (Bacon et al., 2017) based on the reviewers' interest from the previous peer-review. Please note that our work focuses on the *synergism* property among macro actions in a macro action ensemble, as the title suggests. The comparison results are reported in Table A3, in which the results corresponding to the option-critic method are generated completely based on the officially released implementation[1].

A3 Computational Infrastructure

In this section, we provide the configuration of our computing infrastructure in Table A4 for reference.

A4 Reproducibility

We implemented the proposed framework using Tensorflow. The source codes are well verified and fully reproducible. All of the presented experiments in our paper are re-producible with easy-following instructions. For more details about the provided source codes, please refer to the anonymous GitHub repository at the following address: `https://github.com/MacroEnsemble/ICLR_2021_Code`.

[1]The Option-Critic Architecture: `https://github.com/jeanharb/option_critic`

Table A2: This table is the detail version of Table 1. The '$\max\{m_1, m_2, m_3\}$.' 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 '$\max\{m_1, m_2, m_3\}$.'. The *synergism* property exists among the macro actions if the percentage reported in the '*Improvement*' column is positive.

Atari 2600	**Method**	**Macro 1** m_1	**Macro 2** m_2	**Macro 3** m_3	$\max\{m_1, m_2, m_3\}$	**Ensemble Perf.**	**Improvement**
Alien	Ours	1469.33 (136.56)	1496.32 (319.39)	1498.38 (512.12)	1498.38 (512.12)	1951.32 (396.42)	30.23%
	IEB	1621.28 (381.42)	1790.48 (151.17)	1723.32 (121.45)	1790.48 (151.17)	1522.42 (93.17)	-14.97%
Assault	Ours	4698.87 (416.29)	5030.98 (973.60)	4546.01 (467.07)	5030.98 (973.60)	5263.31 (169.15)	4.62%
	IEB	3305.75 (392.32)	3612.56 (246.94)	2917.08 (97.38)	3612.56 (246.94)	2593.04 (241.15)	-28.22%
Asteroids	Ours	7105.89 (2431.10)	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%
Beamrider	Ours	3758.06 (405.62)	3996.31 (476.38)	N/A (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%
Breakout	Ours	N/A (N/A)†	N/A (N/A)†	N/A (N/A)†	N/A (N/A)†	319.74 (42.36)	N/A†
	IEB	193.47 (36.36)	60.40 (15.38)	200.82 (44.37)	200.82 (44.37)	50.62 (3.92)	-74.79%
Chopper Command	Ours	8259.09 (1161.92)	6012.87 (777.56)	8502.19 (932.13)	8502.19 (932.13)	11568.96 (1248.61)	36.07%
	IEB	4426.52 (1037.94)	5882.15 (1141.21)	9653.16 (1637.70)	9653.16 (1637.70)	4396.36 (868.64)	-54.46%
Crazy Climber	Ours	113472.61 (4190.81)	100104.19 (5012.19)	N/A (N/A)*	113472.61 (4190.81)	111466.74 (3371.71)	-1.77%
	IEB	105351.48 (4350.37)	109049.90 (3399.09)	112424.02 (2533.33)	112424.02 (2533.33)	105851.57 (7493.50)	-5.85%
Enduro	Ours	N/A (N/A)†	N/A (N/A)†	N/A (N/A)†	N/A (N/A)†	830.93 (113.92)	N/A†
	IEB	481.67 (40.68)	455.60 (24.68)	493.11 (50.01)	493.11 (50.01)	322.76 (54.82)	-34.55%
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%
Gravitar	Ours	937.76 (103.74)	1011.37 (155.23)	N/A (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%
Krull	Ours	8166.66 (355.97)	7105.39 (195.66)	6572.87 (149.59)	8166.66 (355.97)	8663.77 (1164.13)	6.09%
	IEB	6460.62 (220.01)	6751.43 (174.00)	6586.52 (116.01)	6751.43 (174.00)	7427.26 (763.25)	10.01%
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%
*Q*bert*	Ours	N/A (N/A)†	N/A (N/A)†	N/A (N/A)†	N/A (N/A)†	14540.53 (463.21)	N/A†
	IEB	11838.20 (1023.93)	10389.99 (1742.65)	5254.93 (761.42)	11838.20 (1023.93)	5421.76 (1115.34)	-54.20%
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%
Solaris	Ours	2880.61 (493.98)	2116.17 (274.59)	2366.28 (285.26)	2880.61 (493.98)	3236.40 (238.09)	12.35%
	IEB	2131.12 (366.63)	1873.44 (201.11)	2110.11 (251.30)	2131.12 (366.63)	1856.87 (213.70)	-12.87%
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%
Up N Down	Ours	187376.52 (89212.41)	119687.68 (42275.32)	349670.21 (17272.84)	349670.21 (17272.84)	941419.24 (56575.20)	169.23%
	IEB	189588.47 (101523.43)	390699.63 (171652.74)	182381.58 (113845.08)	390699.63 (171652.74)	283856.43 (107352.04)	-27.35%
Venture	Ours	0.00 (0.00)	0.00 (0.00)	N/A (N/A)*	0.00 (0.00)	167.25 (334.51)	N/A‡
	IEB	0.00 (0.00)	0.00 (0.00)	12.57 (28.11)	12.57 (28.11)	0.00 (0.00)	-100.00%
Zaxxon	Ours	6688.42 (1173.89)	8746.27 (748.50)	8317.77 (551.52)	8746.27 (748.50)	8856.88 (817.74)	1.26%
	IEB	7249.23 (1037.50)	7975.56 (1629.89)	6840.86 (1574.92)	7975.56 (1629.89)	6657.54 (1686.32)	-16.53%

*: The macro ensemble constructed by our method composed of only two macros in *Beamrider*, *Crazy Climber*, *Gravitar* and *Venture*, which implies that for each of these environments, adding another macro into the macro ensemble may not be beneficial to the construction procedure of a synergistic macro action ensemble.

†: The macro ensembles constructed by our methodology in *Breakout*, *Enduro* and *Q*bert* are an empty sets, implying that the action space of each environment is too simple to be benefited from macro actions. Thus, there exists no decoupled macro action (i.e., N/A in the *Best Decoupled Macro Perf* column). The values listed in the *Ensemble Perf* column for *Breakout*, *Enduro* and *Q*bert* represent the performances of the agent trained with primitive actions. Since there exists no decoupled macro action in *Breakout*, *Enduro* and *Q*bert*, the corresponding *Improvement* are unable to calculate.

‡: In *Venture*, N/A in the *Improvement* column is due to the division by zero. Note that the agent is still benefited from the macro ensemble.

Table A3: A summary of the final performances evaluated on the *Atari 2600* environments corresponding to our proposed methodology as well as the other two temporal abstraction baselines, including IEB and option-critic. The methods listed in this table are trained with the same amount of wall time using five different random seeds.

Atari 2600	**IEB**	**Option-Critic**	**Ours**	**Primitive**
Alien	1522.42 (93.17)	1257.15 (64.06)	1951.32 (396.42)	1723.46 (308.04)
Assault	2593.04 (241.15)	2593.81 (163.95)	5263.31 (169.15)	4671.59 (362.47)
Asteroids	2349.21 (189.27)	1007.05 (82.19)	19685.48 (7421.24)	4274.31 (3260.21)
Beamrider	1722.42 (584.84)	2274.16 (132.55)	4165.74 (372.66)	3018.64 (841.96)
Breakout	50.62 (3.92)	138.77 (81.59)	319.74 (42.36)	319.74 (42.36)
Chopper Command	4396.36 (868.64)	460.00 (531.41)	11568.96 (1248.61)	5832.08 (1165.42)
Crazy Climber	105851.57 (7493.50)	66696.55 (3569.36)	111466.74 (3371.71)	102529.56 (6485.23)
Enduro	322.76 (54.82)	0.00 (0.00)	830.93 (113.92)	830.93 (113.92)
Frostbite	812.71 (747.94)	257.80 (4.31)	4298.52 (1151.88)	291.96 (4.97)
Gravitar	842.57 (138.18)	219.82 (26.32)	1496.22 (254.70)	913.66 (193.43)
Krull	7427.26 (763.25)	2676.32 (1972.71)	8663.77 (1164.13)	8466.92 (217.69)
Kung-Fu Master	38236.28 (1803.30)	19328.57 (3488.84)	44453.20 (4955.30)	18466.67 (5517.75)
*Q*bert*	5421.76 (1115.34)	3850.67 (1548.42)	14540.53 (463.21)	14540.53 (463.21)
Seaquest	811.42 (11.01)	4914.21 (559.87)	1362.75 (152.32)	907.20 (17.89)
Solaris	1856.87 (213.70)	2055.29 (282.90)	3236.40 (238.09)	2258.39 (333.50)
Space Invaders	1006.45 (90.95)	580.62 (23.83)	1368.52 (159.05)	1313.08 (232.02)
Up N Down	283856.43 (107352.04)	2105.67 (301.02)	941419.24 (56575.20)	192992.66 (77930.57)
Venture	0.00 (0.00)	0.00 (0.00)	167.25 (334.51)	0.00 (0.00)
Zaxxon	6657.54 (1686.32)	4033.42 (2556.35)	8856.88 (817.74)	8306.71 (772.05)

Table A4: Specification of our computing infrastructure.

Component	Customized Machine
Processor	32 cores / 64 threads (3.0GHz, up to 4.2GHz)
Hard Disk Drive	6TB SATA3 7200rpm
Solid-State Disk	1TB PCIe Gen 3 NVMe
Graphics Card	NVIDIA GeForce® RTX 2080Ti (two per instance)
Memory	16GB DDR4 2400MHz (128GB in total)