Zero-Shot Action Generalization with Limited Observations

Abdullah Alchihabi[1*] **Hanping Zhang**[1*] **Yuhong Guo**[1,2]
[1]School of Computer Science, Carleton University, Ottawa, Canada
[2]Canada CIFAR AI Chair, Amii, Canada
{abdullahalchihabi@cmail., jagzhang@cmail., yuhong.guo@}carleton.ca

Abstract

Reinforcement Learning (RL) has demonstrated remarkable success in solving sequential decision-making problems. However, in real-world scenarios, RL agents often struggle to generalize when faced with unseen actions that were not encountered during training. Some previous works on zero-shot action generalization rely on large datasets of action observations to capture the behaviors of new actions, making them impractical for real-world applications. In this paper, we introduce a novel zero-shot framework, Action Generalization from Limited Observations (AGLO). Our framework has two main components: an action representation learning module and a policy learning module. The action representation learning module extracts discriminative embeddings of actions from limited observations, while the policy learning module leverages the learned action representations, along with augmented synthetic action representations, to learn a policy capable of handling tasks with unseen actions. The experimental results demonstrate that our framework significantly outperforms state-of-the-art methods for zero-shot action generalization across multiple benchmark tasks, showcasing its effectiveness in generalizing to new actions with minimal action observations.

1 INTRODUCTION

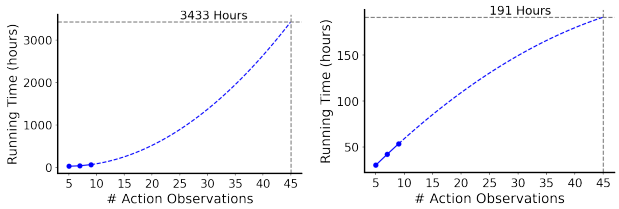
Reinforcement Learning (RL) has exhibited significant success in addressing sequential decision-making problems (Mnih et al., 2015; Silver et al., 2016) across diverse application domains, from robotics (Kober et al., 2013) to healthcare (Yu et al., 2021). However, this success is predominantly constrained to settings where agents are tested in environments identical or similar to those encountered during training, using the same set of actions. RL policies often exhibit poor generalization when faced with unseen tasks, novel environments, or unfamiliar action spaces. This limitation poses a significant challenge to deploying RL agents in dynamic, real-world settings. To mitigate this issue, recent work has sought to improve the generalization of RL agents, aiming to enhance their robustness and adaptability to new tasks and diverse environmental conditions (Cobbe et al., 2019; Pinto et al., 2017; Finn et al., 2017).

Despite these investigations, generalization to new unseen actions at test time remains a relatively underexplored area. While several studies have investigated policy generalization to unseen actions (Chandak et al., 2020; Ye et al., 2023), these approaches typically rely on fine-tuning the learned policies, which limits their applicability in real-world scenarios where retraining is impractical or costly. Recently, some work has focused on zero-shot generalization to unseen actions, where policies trained on observed actions are directly applied to new actions without any fine-tuning or additional adaptation. Jain et al. (2020) introduced a novel approach using an action embedder module, which learns action representations from task-agnostic action observations that capture the intrinsic properties of actions. These representations are then leveraged by a policy module for action selection. While this approach offers a promising step toward generalizing to unseen actions, it has a significant limitation. The action embedder requires a large number of observations for each action to generate informative and discriminative representations. In many real-world domains, obtaining such a large dataset of action observations is impractical, as these observations are typically generated through hand-crafted interactions between the action and environment by domain experts, a process that can be both time-consuming and expensive. Moreover, as the number of required observations increases, so does the

Proceedings of the 28[th] International Conference on Artificial Intelligence and Statistics (AISTATS) 2025, Mai Khao, Thailand. PMLR: Volume 258. *Equal contribution.

time needed to generate the observations and to train the action embedder, as shown in Figure 1. Figure 1a illustrates the relationship between the number of action observations and the time required for their generation in the CREATE environment, while Figure 1b highlights the time required to train the action embedder over the corresponding observations. This underscores the need for methods that can generalize to new, unseen actions without fine-tuning and with access to a very limited number of action observations. Developing such methods is crucial for enabling RL agents to be more broadly applicable in resource-constrained environments.

In response to these needs, in this work we propose a novel framework, named AGLO, for zero-shot Action Generalization from Limited Observations. This framework is made up of two modules: an action representation learning module and a policy learning module. The action representation learning module learns discriminative action embeddings (representations) from a limited number of associated observations. This is achieved by first using a coarse action observation encoder that learns a coarse representation for each action observation based solely on itself. Subsequently, a refined action observation encoder utilizes the similarity between coarse action observation representations and the corresponding action labels of each observation to learn refined action observations through graph contrastive learning and action observation representation classification. Furthermore, we employ a hierarchical variational auto-encoder to improve the learned observation representations through a reconstruction loss. Eventually, action representations are generated by pooling over the learned representations of their corresponding observations. The policy learning module leverages action representations to learn a dynamic policy. To improve the generalizability of the learned policy to previously unseen actions, we augment the action embedding space by generating synthetic action representations that encourage exploration during policy learning and minimize overfitting to actions encountered during training. We evaluate our proposed framework against state-of-the-art zero-shot action generalization methods on multiple tasks where each action is associated with a limited number of observations. The empirical results demonstrate the efficacy and superior generalization capacity of our proposed AGLO framework.



(a) Observation Generation (b) Action Rep. Learning

Figure 1: Estimated running time (in hours) for varying numbers of action observations in the CREATE environment. Previous studies utilized 45 observations per action (Jain et al., 2020). (a) Running time estimate for generating action observations. (b) Running time estimate for action representation learning.

2 RELATED WORKS

2.1 Generalization in Reinforcement Learning

Generalization in RL is a fundamental challenge aimed at reducing the generalization gap between training and testing environments. The primary focus is on how well an RL policy learned during training can perform on previously unseen states. Kirk et al. (2023) categorize existing methods to address generalization into three main types. The first type aims to increase the similarity between training and testing environments, leveraging techniques like data augmentation (Mazoure et al., 2022; Zhang and Guo, 2022) and environment generation (Jiang et al., 2021) to artificially enhance the diversity of training environments, thus improving the robustness of the learned policies across a broader range of testing conditions. The second type focuses on addressing discrepancies between the training and testing environments by modeling the variations and training policies that can adapt to such shifts (Song et al., 2020). The third category focuses on RL-specific strategies, such as modifying value or policy functions to mitigate overfitting. Methods such as decoupling value and policy functions (Raileanu and Fergus, 2021), rethinking exploration strategies (Moon et al., 2022), and fast adaptation techniques (Raileanu et al., 2020) have been proposed to improve policy generalization without requiring additional environment modifications. The aim of all these methods is to improve RL generalization and hence enable the deployment of RL in more robust, real-world applications where unseen states and environments are prevalent.

2.2 Generalization to Unseen Actions

Recently, the focus has shifted from generalization within the state space to the more challenging problem of handling unseen actions. While some prior works have investigated generalization to new actions, these approaches typically rely on fine-tuning the model for each new action (Chandak et al., 2020; Ye et al., 2023), which limits their practicality in dynamic environments. Jain et al. (2020) introduced a pioneering method aimed at improving the performance of RL agents on unseen

actions without the need for fine-tuning. Their approach leverages task-agnostic action observations to learn action embeddings through a Hierarchical Variational Autoencoder (HVAE) (Edwards and Storkey, 2017). This enables a policy learning module to select actions based on their learned embeddings, representing a significant step towards more adaptable RL agents. However, a key limitation of this method is its reliance on a large number of action observations to learn sufficiently representative action embeddings, which can be costly and time-consuming to obtain. More recently, Sinii et al. (2023) proposed a Headless Algorithm Distillation method which employs a causal transformer to infer the embeddings of unseen actions based on in-context examples. Despite these innovations, none of the existing approaches can generalize to unseen actions from a limited number of task-agnostic action observations without the need for policy fine-tuning, highlighting an open challenge in the field.

3 PROBLEM SETUP

In this work, we focus on the RL task of zero-shot generalization to new actions from a limited number of action observations. Specifically, the goal is to train a policy on a set of seen actions and evaluate its performance on a set of previously unseen actions without any re-training or fine-tuning on the new action set. Each action is associated with a small set of task-agnostic observations that capture the properties and behavior of the action, enabling the agent to generalize to unseen actions using only this limited information.

3.1 Reinforcement Learning

The RL problem is formulated as an episodic Markov Decision Process (MDP) with a discrete action space. Each episode of the MDP is characterized by a tuple $(\mathcal{S}, \mathcal{A}, \mathcal{T}, \mathcal{R}, \gamma)$, where $\mathcal{S}$ represents the set of states, $\mathcal{A}$ denotes the set of actions, $\mathcal{T}$ is the state transition function, $\mathcal{R}$ is the reward function, and γ is the discount factor. During its interaction with the environment, at each timestep t, the RL agent selects an action $a_t \in \mathcal{A}$ using a policy π based on the current state $s_t \in \mathcal{S}$. The agent then receives a reward $\mathcal{R}(s_t, a_t)$ and transitions to a new state s_{t+1} according to the environment's transition dynamics $\mathcal{T}$. The objective of the agent is to maximize the accumulated discounted reward over the course of an episode, given by:

$$J = \sum_{t=0}^{T-1} \gamma^t r_t \tag{1}$$

where T is the length of the episode and $r_t = \mathcal{R}(s_t, a_t)$ denotes the reward at timestep t.

3.2 Zero-Shot Action Generalization

In the context of zero-shot generalization to new actions, the set of all actions is divided into two non-overlapping sets: the set of seen actions $\mathbb{A}$ and the set of unseen actions $\mathbb{A}'$. This setup consists of two sequential phases: the training phase and the evaluation phase. During the training phase, the set of actions available in each training episode $\mathcal{A}$ is sampled from the set of seen actions, $\mathbb{A}$. In the evaluation phase, the trained agent is tested in each evaluation episode using a set of actions $\mathcal{A}$ sampled from the set of unseen actions, $\mathbb{A}'$. The objective is to learn a policy $\pi(a|s, \mathcal{A})$ that maximizes the expected discounted reward for any action set $\mathcal{A}$ sampled from the set of unseen actions $\mathbb{A}'$:

$$J = \mathbb{E}_{\mathcal{A} \subset \mathbb{A}', \tau \sim \pi} \Big[\sum_{t=0}^{T-1} \gamma^t r_t \Big] \tag{2}$$

where τ denotes a trajectory—a sequence of states, actions, and rewards—generated by interacting with environment using the policy. Each action $a \in \mathbb{A} \cup \mathbb{A}'$ is associated with a set of observations $\mathcal{O} = \{o_1, \cdots, o_n\}$, where n is the number of observations collected for each action a. These action observations can take the form of state trajectories, videos, or images that capture the intrinsic properties and behavior of the corresponding action. The set of observations associated with the seen actions is denoted by $\mathbb{O} = \{\mathcal{O}_1, \cdots, \mathcal{O}_N\}$, while the set of observations corresponding to the unseen actions is denoted by $\mathbb{O}' = \{\mathcal{O'}_1, \cdots, \mathcal{O'}_{N'}\}$, where N and N' represent the numbers of seen and unseen actions, respectively.

Previous works assume the availability of a large number of observations $\mathcal{O}$ for each action a, for both seen or unseen actions. However, this assumption introduces significant challenges in application domains where collecting a large number of action observations is expensive or impractical. Additionally, as the number of action observations increases, the cost and time required for generating such observations, as well as for training agents using these observations, increase considerably. Therefore, in this work, we operate under the assumption that the number of available observations for each action, n, is severely limited.

4 PROPOSED METHOD

In this section, we present our proposed AGLO framework for zero-shot Action Generalization with Limited Observations. The framework is composed of two key modules: the action representation learning module and the policy learning with action representation augmentation module. The action representation learning module is designed to generate discriminative action embeddings by encoding the limited set of observations

for each action. The policy learning module leverages the learned action embeddings to augment the representations of the set of seen actions with synthetic action representations. The policy is subsequently trained using both the original representations of the actions and their augmented synthetic counterparts, facilitating the learning of a policy that generalizes effectively to unseen actions. In the following sections, we provide detailed descriptions of each module within our proposed AGLO framework.

4.1 Action Representation Learning

The action representation learning module is designed to generate representative action embeddings by encoding the limited set of observations associated with each action. To achieve this, the module first employs a coarse observation encoder, followed by a refined observation encoder that enhances cohesion and action discrimination in the learned observation representations. Subsequently, action representations are generated by pooling the associated refined observation embeddings. The action representations are further refined using a hierarchical variational auto-encoder to minimize the observation reconstruction loss. This ensures that the generated action embeddings faithfully capture the key characteristics of the action observations.

4.1.1 Observation Representation Learning

Coarse Action Observation Encoder The coarse action observation encoder independently encodes each action observation, without considering the relationships between observations of the same action or other actions. Specifically, the coarse action observation encoder, denoted as g_{co}, takes an individual observation o of an action a as input, and outputs its coarse action observation embedding c as follows:

$$c = g_{\text{co}}(o). \tag{3}$$

Refined Action Observation Encoder The coarse action observation embeddings, generated independently, do not exploit the information available in other observations of the same action or in observations of other actions. Given the limited number of observations per action, it is crucial to leverage the available information across all observations to generate more representative action observation embeddings. To achieve this, we propose a graph-based refined action observation encoder that encourages cohesion in the observation embedding space by promoting similarity between related observations while ensuring discrimination between different actions.

The input to the refined action observation encoder consists of a batch of actions and their corresponding observations. We begin by constructing an action observation graph $G = (V, E)$, where V is the set of nodes, with each node u representing an action observation o. The total number of nodes in the graph is $|V| = n \times K$, where K denotes the number of actions in each batch. Each node u is associated with an input feature vector that corresponds to the coarse observation embedding c generated by the coarse action observation encoder g_{co}. The feature vectors of all nodes in the graph are organized into a matrix $C \in \mathbb{R}^{|V| \times d}$, where d is the size of the coarse action observation embeddings. E is the set of edges of the graph G, represented by a weighted undirected adjacency matrix A, which is constructed as follows:

$$A[u, v] = \begin{cases} \sigma(C_u, C_v) & \text{if } \sigma(C_u, C_v) > \epsilon \\ 0 & \text{otherwise.} \end{cases} \tag{4}$$

Here, $\sigma(.,.)$ represents the Pearson correlation function, and $\sigma(C_u, C_v)$ measures the similarity between the coarse action observation embeddings of nodes u and v. The threshold ϵ controls the sparsity of the graph by filtering out edges with low similarities. Additionally, each node u is associated with a label y that indicates the action to which the corresponding observation belongs. The labels of all nodes are organized into a label indicator matrix $Y \in \{0, 1\}^{|V| \times K}$.

To effectively utilize the constructed action observation graph G, we define the refined observation encoder as a graph neural network-based function, g_{re}, that learns refined observation representations through message passing along the graph. The refined observation embeddings are generated as follows:

$$\tilde{C} = g_{\text{re}}(C, A). \tag{5}$$

Here, $\tilde{C}$ represents the refined action observation embedding matrix. To promote cohesion in the learned embedding space, we train both g_{re} and g_{co} to push similar observations closer to each other. This is achieved by minimizing the following contrastive loss:

$$\mathcal{L}_{\text{cont}} = -\sum_{u \in V} \log \frac{\exp(\sigma(\tilde{C}_u, \tilde{C}_{v_u})/\kappa)}{\exp(\sigma(\tilde{C}_u, \tilde{C}_{v_u})/\kappa) + \sum_{v' \in \mathcal{N}_u^c} \exp(\sigma(\tilde{C}_u, \tilde{C}_{v'})/\kappa)} \tag{6}$$

Here, node v_u represents a positive instance randomly sampled from the set of nodes connected to node u ($\mathcal{N}_u = \{v \in V \mid A[u, v] > 0\}$), while $\mathcal{N}_u^c$ denotes the set of negative instances of size K' randomly sampled from the set of all nodes not connected to u ($\mathcal{N}_u^c \subset V \setminus \mathcal{N}_u$). The temperature parameter κ controls the scaling of the similarity scores. To further enhance action discrimination in the refined observation embeddings, we also define a classification function ρ that predicts

the corresponding action labels based on the refined embeddings as follows:

$$P = \rho(\tilde{C}) \tag{7}$$

where P is the predicted action probability distribution. g_{re}, g_{co}, and ρ are jointly trained to minimize the following action observation classification loss:

$$\mathcal{L}_{ce} = \frac{1}{|V|} \sum\nolimits_{u \in V} \ell(P_u, Y_u) \tag{8}$$

where ℓ denotes the standard cross-entropy loss function, and P_u and Y_u are the predicted and ground-truth class distribution vectors, respectively, for node u.

4.1.2 Action Representation Pooling

The goal of the action representation learning module is to generate representative action embeddings by utilizing the refined observation embeddings for each action's observation set. Specifically, for each action a_i, the action representation function g encodes the corresponding refined action observation embedding set $\tilde{\mathcal{C}}_i = \{\tilde{C}_u \mid u \in V \wedge Y_{u,i} = 1\}$ as follows:

$$(\mu_i, \sigma_i^2) = g(\tilde{\mathcal{C}}_i) \tag{9}$$

where the embedding of action a_i is encoded as a Gaussian distribution $\mathcal{N}(\mu_i, \sigma_i^2)$.

To ensure that the learned action embedding distributions effectively capture the information present in the corresponding action observations, we employ a Hierarchical Variational Auto-Encoder (HVAE) (Edwards and Storkey, 2017). The HVAE samples an action embedding $\hat{c}_i$ from the Gaussian distribution $\mathcal{N}(\mu_i, \sigma_i^2)$ and conditions both the observation encoder $q_\psi(z_i^j | o_i^j, \hat{c}_i)$ and the decoder $p(o_i^j | z_i^j, \hat{c}_i)$ on the action embedding $\hat{c}_i$. This process is applied over the entire set of observations $\mathcal{O}_i$ associated with action a_i, where o_i^j is the j-th observation of action a_i and z_i^j represents the j-th encoded observation obtained from the action observation encoder q_ψ. The action representation function g and the HVAE are trained jointly to minimize the following action observation reconstruction loss:

$$\mathcal{L}_{\text{reconst}} = \sum_{\mathcal{O}_i \in \mathbb{O}} \Big[\mathbb{E}_{g(\hat{c}_i | \tilde{\mathcal{C}}_i)} \Big[\sum_{o_i^j \in \mathcal{O}_i} \mathbb{E}_{q_\psi(z_i^j | o_i^j, \hat{c}_i)} \log p(o_i^j | z_i^j, \hat{c}_i) - D_{\text{KL}}(q_\psi || p(z_i^j | \hat{c}_i)) \Big] - D_{\text{KL}}(g || p(\hat{c}_i)) \Big] \tag{10}$$

where D_{KL} represents the Kullback–Leibler (KL) divergence function. The two D_{KL} terms denote the KL-divergences between the output distributions of the observation encoder q_ψ and its prior, and between the output distributions of the action representation function g and its prior, respectively.

All the components of the action representation learning module are trained jointly in an end-to-end fashion to minimize the following overall loss:

$$\mathcal{L}_{\text{total}} = \mathcal{L}_{\text{reconst}} + \lambda_{\text{ce}} \mathcal{L}_{ce} + \lambda_{\text{cont}} \mathcal{L}_{\text{cont}} \tag{11}$$

Here, λ_{ce} and λ_{cont} are hyper-parameters controlling the contributions of the observation classification loss $\mathcal{L}_{ce}$ and the contrastive learning loss $\mathcal{L}_{\text{cont}}$, respectively, to the overall loss.

4.2 Policy Learning with Action Representation Augmentation

4.2.1 Action Representation Augmentation

The policy learning module aims to learn a policy $\pi(a|s, \mathcal{A})$ over a sampled subset of seen actions $\mathcal{A} \subset \mathbb{A}$ that can generalize effectively to unseen actions $\mathbb{A}'$. The policy leverages the representations of these seen actions, which are produced by the action representation learning module, where each action representation $\hat{c}_i$ is sampled from a Gaussian distribution $\mathcal{N}(\mu_i, \sigma_i^2)$. To prevent the learned policy from overfitting to seen actions, we employ action representation augmentation to generate synthetic representations of the seen actions in the learned action embedding space. These synthetic action representations promote exploration during policy learning, thereby reducing overfitting and enhancing the policy's generalization to unseen actions.

In particular, for each action a_i with representation $\hat{c}_i$ sampled from a Gaussian distribution $\mathcal{N}(\mu_i, \sigma_i^2)$, we randomly select another action $a_j \in \mathcal{A}$ $(a_i \neq a_j)$ with representation $\hat{c}_j$ sampled from $\mathcal{N}(\mu_j, \sigma_j^2)$. A new synthetic representation for a_i is then generated by applying mixup augmentation in the action embedding space between $\hat{c}_i$ and $\hat{c}_j$, resulting in $\hat{c}_{i,\text{syn}}$ which follows a Gaussian distribution $\mathcal{N}(\mu_{i,\text{syn}}, \sigma_{i,\text{syn}}^2)$, where the parameters $\mu_{i,\text{syn}}$ and $\sigma_{i,\text{syn}}^2$ are defined as follows:

$$\begin{aligned} \mu_{i,\text{syn}} &= \lambda \mu_i + (1-\lambda)\mu_j \\ \sigma_{i,\text{syn}}^2 &= \lambda \sigma_i^2 + (1-\lambda)\sigma_j^2 + \lambda(1-\lambda)(\mu_i - \mu_j)^2 \end{aligned} \tag{12}$$

Here, λ is the mixing coefficient sampled from a Beta distribution, $\text{Beta}(\alpha, \alpha)$, which controls the mixing rate between the two action embeddings. By applying this synthetic action representation generation process to each action $a \in \mathcal{A}$, we double the size of the action representation set in each episode where each action a is represented by both its original representation generated based on its observations and a synthetic representation generated through augmentation.

Algorithm 1 Policy Training Procedure

Input: Seen actions $\mathbb{A}$, action observations $\mathbb{O}$, Learned Action Embedder
Output: Learned Policy parameters
for iter = 1 **to** maxiters **do**
  while episode not finished **do**
    Sample a subset $\mathcal{A}$ of K actions from $\mathbb{A}$
    $\forall a \in \mathcal{A}, \hat{c} \sim \mathcal{N}(\mu, \sigma^2)$
    Augment action representations via Eq.(12)
    Select action $a_t \sim \pi(\cdot|s_t, \mathcal{A})$ based on Eq.(13)
    Execute action a_t: $s_{t+1}, r_t = \text{ENV}(s_t, a_t)$
    Store (s_t, a_t, r_t, s_{t+1}) in replay buffer
  end while
  Update policy π using Eq.(14)
end for

4.2.2 Policy Learning

The policy learning module consists of two key components: a state encoding function and a utility function. The state encoding function, f_ω, takes the current state s_t as input and outputs a corresponding state encoding at time t. The utility function, f_ν, then takes a representation $\hat{c}_i$ of an action a_i and the state encoding $f_\omega(s_t)$ as input and outputs the predicted utility of the action a_i in state s_t based on the representation $\hat{c}_i$. Since each action a_i is represented by both its original representation $\hat{c}_i$ and a synthetic representation $\hat{c}_{i,\text{syn}}$ generated through augmentation, the predicted score for a_i is the sum of the utility scores predicted based on both $\hat{c}_i$ and $\hat{c}_{i,\text{syn}}$. The predicted scores for all actions in $\mathcal{A}$ are then normalized into a probability distribution using a softmax function as follows:

$$\pi(a_i|s_t, \mathcal{A}) = \frac{\exp(f_\nu(\hat{c}_i, f_\omega(s_t))) + \exp(f_\nu(\hat{c}_{i,\text{syn}}, f_\omega(s_t)))}{\sum_{j=1}^{K}(\exp(f_\nu(\hat{c}_j, f_\omega(s_t))) + \exp(f_\nu(\hat{c}_{j,\text{syn}}, f_\omega(s_t))))} \quad (13)$$

This allows the policy to select actions based on both original and synthetic action representations.

In some environments, the discrete actions are associated with additional parametrization intended to encode the location/coordinates of applying the selected action in the environment. In such cases, we define an auxiliary function f_χ, which takes the state encoding as input and outputs the location/coordinates to apply the selected action, similar to (Jain et al., 2020). The overall action information in the environment is determined by sampling the action location/coordinates from the auxiliary function and the discrete action from Eq.(13). The state encoding function, the utility function and the auxiliary function (if needed) are trained jointly in an end-to-end fashion using policy gradients (Sutton et al., 1999). The objective of the policy training is to maximize the expected reward while promoting exploration. This is achieved by incorporating an entropy regularization term into the training objective, encouraging diversity in the actions selected by the policy:

$$\max_{\nu,\omega,\chi} \mathbb{E}_{\mathcal{A}\subset\mathbb{A},\tau\sim\pi}\left[\sum_{t=0}^{T-1} \gamma^t r_t + \beta\mathcal{H}(\pi(\cdot|s_t, \mathcal{A}))\right] \quad (14)$$

where $\mathcal{H}(.)$ denotes the entropy function, and β is a hyper-parameter controlling the contribution of the entropy regularization term. The policy training procedure is presented in Algorithm 1.

5 EXPERIMENTS

5.1 Experimental Setup

5.1.1 Environment

We evaluate our AGLO framework in a sequential decision-making environment, Chain REAction Tool Environment (CREATE). CREATE (Jain et al., 2020) is a physics-based environment where the agent's objective is to manipulate the trajectory of a ball in real-time by strategically placing tools in its path to guide the ball towards a goal position. In this work, we focus on three tasks within this environment: Push, Navigate and Obstacle. The agent's action selection involves choosing which tool to place and specifying the 2D coordinates for placing the selected tool. The action observations for each tool consist of state observations that describe the trajectories of a ball interacting with the tool at various speeds and from different directions. Further details about the CREATE environment can be found in (Jain et al., 2020).

5.1.2 Evaluation Setup

We follow the evaluation procedure outlined in (Jain et al., 2020), where actions are divided into three non-overlapping sets: train, validation, and test actions. We use the same train/validation/test action split provided by Jain et al. (2020), where 50% of the actions are allocated to the training set, 25% to the validation set, and 25% to the test set. For each action, a limited number of observations is generated. The action representation learning module is trained on the training actions set and their corresponding action observations. Once trained, the action representation learning module is used to generate embeddings for the actions in the training set, and a policy is trained to solve the environment tasks using these learned action representations.

During the evaluation phase, the learned policy is assessed on the test action set over 200 episodes, where

Table 1: The overall performance (standard deviation within brackets) on the Push, Navigate and Obstacle tasks in the CREATE environment with 3 different numbers of observations per action (5, 7 and 9).

		Push Task			**Navigate Task**			**Obstacle Task**		
		Target Hit	Goal Hit	Reward	Target Hit	Goal Hit	Reward	Target Hit	Goal Hit	Reward
5 Obs.	VAE	$25\%_{(0.4)}$	$5\%_{(0.1)}$	$0.80_{(0.4)}$	$1\%_{(0.2)}$	$0\%_{(0.0)}$	$0.10_{(0.2)}$	$2\%_{(0.1)}$	$0\%_{(0.0)}$	$0.83_{(0.8)}$
	HVAE	$15\%_{(0.8)}$	$3\%_{(0.2)}$	$0.49_{(0.2)}$	$17\%_{(0.1)}$	$2\%_{(0.2)}$	$0.48_{(0.3)}$	$1\%_{(0.1)}$	$0\%_{(0.0)}$	$0.73_{(0.8)}$
	AGLO	$\mathbf{48}\%_{(0.1)}$	$\mathbf{23}\%_{(0.1)}$	$\mathbf{2.79}_{(0.1)}$	$\mathbf{49}\%_{(0.5)}$	$\mathbf{13}\%_{(0.5)}$	$\mathbf{1.82}_{(0.5)}$	$\mathbf{37}\%_{(0.1)}$	$\mathbf{0}\%_{(0.0)}$	$\mathbf{2.49}_{(0.2)}$
7 Obs.	VAE	$15\%_{(0.1)}$	$3\%_{(0.3)}$	$0.55_{(0.4)}$	$2\%_{(0.2)}$	$1\%_{(0.1)}$	$0.18_{(0.1)}$	$1\%_{(0.1)}$	$0\%_{(0.0)}$	$0.82_{(0.9)}$
	HVAE	$21\%_{(0.3)}$	$3\%_{(0.1)}$	$0.58_{(0.5)}$	$2\%_{(0.2)}$	$1\%_{(0.1)}$	$0.13_{(0.7)}$	$1\%_{(0.1)}$	$1\%_{(0.1)}$	$1.71_{(0.1)}$
	AGLO	$\mathbf{46}\%_{(0.3)}$	$\mathbf{20}\%_{(0.2)}$	$\mathbf{2.42}_{(0.2)}$	$\mathbf{44}\%_{(0.5)}$	$\mathbf{11}\%_{(0.3)}$	$\mathbf{1.56}_{(0.3)}$	$\mathbf{34}\%_{(0.4)}$	$\mathbf{7}\%_{(0.1)}$	$\mathbf{3.03}_{(0.8)}$
9 Obs.	VAE	$30\%_{(0.1)}$	$6\%_{(0.3)}$	$0.96_{(0.3)}$	$8\%_{(0.1)}$	$1\%_{(0.1)}$	$0.26_{(0.2)}$	$8\%_{(0.1)}$	$0\%_{(0.0)}$	$1.13_{(1.2)}$
	HVAE	$25\%_{(0.3)}$	$5\%_{(0.1)}$	$0.79_{(0.1)}$	$1\%_{(0.1)}$	$0\%_{(0.0)}$	$0.09_{(0.1)}$	$1\%_{(0.1)}$	$0\%_{(0.0)}$	$0.16_{(0.1)}$
	AGLO	$\mathbf{47}\%_{(0.1)}$	$\mathbf{21}\%_{(0.4)}$	$\mathbf{2.52}_{(0.4)}$	$\mathbf{50}\%_{(0.1)}$	$\mathbf{16}\%_{(0.1)}$	$\mathbf{2.05}_{(0.1)}$	$\mathbf{33}\%_{(0.4)}$	$\mathbf{8}\%_{(0.1)}$	$\mathbf{3.19}_{(1.1)}$

the action set for each episode is randomly sampled from the test set. The representations of test actions are inferred from their corresponding observations using the action representation learning module. We report the mean and standard deviation across three runs for the policy learning module, each with 200 evaluation episodes, for three key evaluation metrics: episode target hit, episode goal hit, and episode reward. The target hit metric indicates whether the target ball moved, while the goal hit metric reflects whether the goal was reached.

5.2 Implementation Details

We utilize the g_{co} and g implementations provided in (Jain et al., 2020), along with the HVAE implementation from (Edwards and Storkey, 2017), with an action embedding size of 128. The message passing function g_{re} consists of two Graph Convolution Network layers (Kipf and Welling, 2017), each followed by a ReLU activation function. The sparsity threshold (ϵ) of the action observation adjacency matrix A is set to 0.95. The action observation classifier ρ is composed of two fully connected layers, each followed by a ReLU activation function. The contrastive loss $\mathcal{L}_{\text{cont}}$ uses a temperature value of $\kappa = 0.5$, a negative observation set size of $K' = 5$, and a hyper-parameter λ_{cont} with a value of $1e^{-1}$. The hyper-parameter λ_{ce} for the observation classification loss is set to $1e^{-3}$. The action representation learning module is trained using the RAdam optimizer with a learning rate of $1e^{-3}$, over 10,000 epochs, and a batch size of $K = 32$.

For the policy learning module, we adopt the implementations of the state encoder, utility function, and auxiliary function from (Jain et al., 2020). We utilize the Proximal Policy Optimization (PPO) method (Schulman et al., 2017) to learn the policy function π, and extend the specific PPO implementation from (Jain et al., 2020). The number of sampled actions is set to $K = 50$ and the maximum episode length is set to 30 timesteps. The mixing rate λ for generating augmented actions representations is sampled from a Beta$(0.4, 0.4)$ distribution. The policy is trained using the Adam optimizer with a learning rate of $1e^{-3}$ for $6e^{7}$ steps and a batch size of 3,072. The entropy coefficient β is set to $5e^{-3}$.

5.3 Comparison Results

We evaluate the performance of our AGLO framework on the zero-shot action generalization problem with limited action observations. Specifically, we conduct experiments using three different numbers of observations per action: 5, 7 and 9 observations. These correspond to 10%, 15% and 20% of the observations per action used in prior work (Jain et al., 2020). We compare our AGLO against two zero-shot action generalization methods: Hierarchical Variational Auto-Encoder (HVAE) (Jain et al., 2020) and Variational Auto-Encoder (VAE) (Jain et al., 2020). The comparison results for the Push, Navigate and Obstacle tasks are presented in Table 1.

Table 1 shows that our AGLO framework consistently and significantly outperforms both the VAE and HVAE methods across all three tasks, for all three numbers of action observations, and across all evaluation metrics. The performance improvements on the Push task are substantial, with gains exceeding 23%, 25% and 17% in terms of Target Hit, and 18%, 17% and 15% in terms of Goal Hit for 5, 7 and 9 action observations, respectively. Moreover, AGLO demonstrates superior performance in terms of average rewards collected on the Push task, achieving remarkable improvements of approximately 1.99 (248% relative gain), 1.84 (317% relative gain) and 1.56 (162% relative gain) with 5, 7 and 9 action observations, respectively. The performance gains on the more challenging Navigate and Obstacle tasks are even more pronounced, where both VAE and HVAE struggle to learn generalizable policies. On the Navigate task, AGLO achieves significant gains in terms of Target Hit with improvements of around 32%, 42% and 42% for 5,

7 and 9 observations, respectively. Similarly, on the Obstacle task, AGLO outperforms both VAE and HAVE in terms of Target Hit by 35%, 33% and 25% for 5, 7 and 9 action observations, respectively. These results highlight the superior performance of our framework compared to existing zero-shot action generalization methods, particularly in scenarios where the number of available action observations is limited, across three challenging sequential decision-making tasks.

5.4 Ablation Study

We conducted an ablation study to investigate the contributions of three novel components in our framework: the observation action classification loss $\mathcal{L}_{ce}$ and the action observation contrastive learning loss $\mathcal{L}_{\text{cont}}$ in the action representation learning module, as well as the action representation augmentation (denoted as Action Aug.) in the policy learning module. Specifically, we consider five variants of our proposed AGLO framework, where in each variant we remove one or two of the three novel components. The evaluation is carried out on the CREATE Push task with 5 observations per action, and the results are presented in Table 2.

The results in Table 2 demonstrate performance degradations in all variants across the three evaluation metrics compared to the full proposed framework. The significant performance drops are observed in variants that do not include action representation augmentation, highlighting the critical role of action representation augmentation in improving the generalization of the learned policy. Moreover, the notable performance drops when $\mathcal{L}_{\text{cont}}$ and $\mathcal{L}_{ce}$ are removed underscore the importance of both of these losses in learning discriminative and cohesive action representations, particularly as reflected in the reward and goal hit metrics. The consistent performance degradations across all metrics for all variants highlight the essential contribution of each novel component in our proposed framework.

5.5 Analysis of Policy Learning

We also investigate the dynamics of the policy learning process within our AGLO framework. Specifically, we sample 200 training and 200 testing episodes, where the actions in the training episodes are drawn from the set of training actions, and the actions in the testing episodes are drawn from the set of testing actions. We evaluate the performance of our framework on both sets of episodes throughout the policy learning process by measuring the mean and standard deviation of Target hit and collected rewards across three runs. To assess its robustness, we compare the performance of our AGLO framework against both VAE and HVAE methods on the Push task with 5 and 7 observations per action, and present the results in Figure 2.

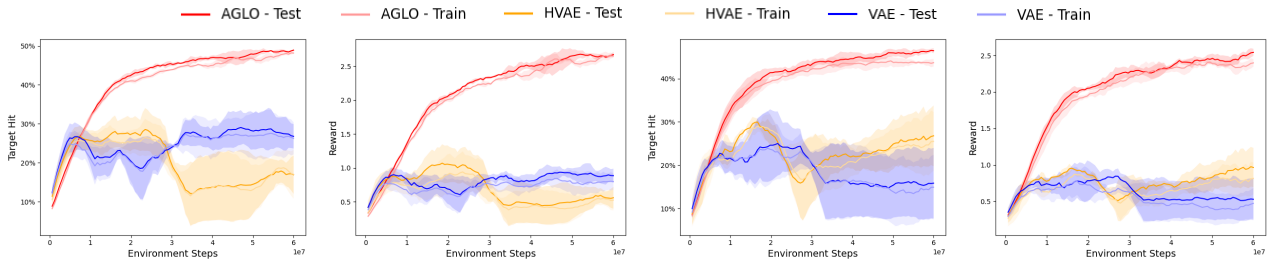
Table 2: Ablation study results (standard deviation within brackets) on the Push task in the CREATE environment with 5 observations per action.

$\mathcal{L}_{ce}$	$\mathcal{L}_{\text{cont}}$	Action Aug.	Target Hit	Goal Hit	Reward
✓	✓	✓	**48**%$_{(0.1)}$	**23**%$_{(0.1)}$	**2.79**$_{(0.1)}$
✓	✓		27%$_{(0.8)}$	7%$_{(0.1)}$	1.03$_{(0.1)}$
	✓	✓	46%$_{(0.3)}$	19%$_{(0.2)}$	2.37$_{(0.2)}$
✓		✓	**48**%$_{(0.1)}$	21%$_{(0.1)}$	2.53$_{(0.1)}$
	✓		21%$_{(0.1)}$	6%$_{(0.1)}$	0.82$_{(0.5)}$
✓			26%$_{(0.3)}$	7%$_{(0.2)}$	0.98$_{(0.2)}$

Figure 2 demonstrates that while all three methods perform similarly in the early stage of policy learning, the performance of AGLO framework diverges significantly from that of VAE and HVAE as learning progresses. In particular, AGLO exhibits steady and consistent improvement in terms of both rewards and Target hit metrics as the number of environment steps increases, across both numbers of action observations. This improvement is observed in both the training and testing episodes, highlighting that our framework effectively learns stable and generalizable policies that perform well on both seen and unseen actions. In contrast, HVAE and VAE methods either converge to suboptimal policy solutions—indicative of a lack of exploration during learning—or develop unstable policies where performance fluctuates or drops significantly during policy learning. Our AGLO framework also exhibits smaller standard deviations across different runs compared to both the HVAE and VAE methods, highlighting the robustness of its training process. These results underscore the superior performance of AGLO in learning stable policies that generalize well to unseen actions, while also highlighting the limitations of current methods.

6 CONCLUSION

In this paper, we introduced a novel AGLO framework to tackle the challenging problem of zero-shot action generalization in Reinforcement Learning, specifically targeting scenarios with limited action observations. Our proposed framework consists of two key components: an action representation learning module and a policy learning module. The action representation learning module encodes the limited set of action observations to generate representative action embeddings. The policy learning module then leverages the learned action embeddings to augment the representations of seen actions with synthetic action representations. The policy is subsequently trained on both the original rep-



(a) 5 Observations, Target Hit (b) 5 Observations, Reward (c) 7 Observations, Target Hit (d) 7 Observations, Reward

Figure 2: Policy learning analysis of VAE, HVAE and our AGLO on the Push task, with 5 and 7 observations per action, evaluated using the Target hit and reward metrics.

resentations of seen actions and their synthetic counterparts, enabling the learning of a policy that generalizes effectively to unseen actions. Our extensive experiments on three benchmark tasks demonstrate the superior zero-shot generalization capabilities of the policies learned by our proposed AGLO framework when applied to unseen actions with limited observations.

References

Yash Chandak, Georgios Theocharous, Chris Nota, and Philip Thomas. Lifelong learning with a changing action set. In *AAAI Conference on Artificial Intelligence (AAAI)*, 2020.

Karl Cobbe, Oleg Klimov, Chris Hesse, Taehoon Kim, and John Schulman. Quantifying generalization in reinforcement learning. In *International Conference on Machine Learning (ICML)*, 2019.

Harrison Edwards and Amos Storkey. Towards a neural statistician. In *International Conference on Learning Representations (ICLR)*, 2017.

Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In *International Conference on Machine Learning (ICML)*, 2017.

Ayush Jain, Andrew Szot, and Joseph J Lim. Generalization to new actions in reinforcement learning. In *International Conference on Machine Learning (ICML)*, 2020.

Minqi Jiang, Edward Grefenstette, and Tim Rocktäschel. Prioritized level replay. In *International Conference on Machine Learning (ICML)*, 2021.

Thomas N. Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. In *International Conference on Learning Representations (ICLR)*, 2017.

Robert Kirk, Amy Zhang, Edward Grefenstette, and Tim Rocktäschel. A survey of zero-shot generalisation in deep reinforcement learning. *Journal of Artificial Intelligence Research*, 76:201–264, 2023.

Jens Kober, J Andrew Bagnell, and Jan Peters. Reinforcement learning in robotics: A survey. *The International Journal of Robotics Research*, 32(11): 1238–1274, 2013.

Bogdan Mazoure, Ilya Kostrikov, Ofir Nachum, and Jonathan J Tompson. Improving zero-shot generalization in offline reinforcement learning using generalized similarity functions. *Advances in Neural Information Processing Systems (NeurIPS)*, 2022.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533, 2015.

Seungyong Moon, JunYeong Lee, and Hyun Oh Song. Rethinking value function learning for generalization in reinforcement learning. *Advances in Neural Information Processing Systems (NeurIPS)*, 2022.

Lerrel Pinto, James Davidson, Rahul Sukthankar, and Abhinav Gupta. Robust adversarial reinforcement learning. In *International conference on machine learning (ICML)*, 2017.

Roberta Raileanu and Rob Fergus. Decoupling value and policy for generalization in reinforcement learning. In *International Conference on Machine Learning (ICML)*, 2021.

Roberta Raileanu, Max Goldstein, Arthur Szlam, and Rob Fergus. Fast adaptation to new environments via policy-dynamics value functions. In *International Conference on Machine Learning (ICML)*, 2020.

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

David Silver, Aja Huang, Chris J Maddison, Arthur Guez, Laurent Sifre, George Van Den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, et al. Mastering the

game of go with deep neural networks and tree search. *nature*, 529(7587):484–489, 2016.

Viacheslav Sinii, Alexander Nikulin, Vladislav Kurenkov, Ilya Zisman, and Sergey Kolesnikov. In-context reinforcement learning for variable action spaces. *arXiv preprint arXiv:2312.13327*, 2023.

Xingyou Song, Yiding Jiang, Stephen Tu, Yilun Du, and Behnam Neyshabur. Observational overfitting in reinforcement learning. In *International Conference on Learning Representations (ICLR)*, 2020.

Richard S Sutton, David McAllester, Satinder Singh, and Yishay Mansour. Policy gradient methods for reinforcement learning with function approximation. *Advances in Neural Information Processing Systems (NIPS)*, 1999.

Jiaqi Ye, Xiaodong Li, Pangjing Wu, and Feng Wang. Action pick-up in dynamic action space reinforcement learning. *arXiv preprint arXiv:2304.00873*, 2023.

Chao Yu, Jiming Liu, Shamim Nemati, and Guosheng Yin. Reinforcement learning in healthcare: A survey. *ACM Computing Surveys (CSUR)*, 55(1):1–36, 2021.

Hanping Zhang and Yuhong Guo. Generalization of reinforcement learning with policy-aware adversarial data augmentation. In *Decision Awareness in Reinforcement Learning Workshop at ICML*, 2022.

Checklist

1. For all models and algorithms presented, check if you include:
   (a) A clear description of the mathematical setting, assumptions, algorithm, and/or model. [Yes]
   (b) An analysis of the properties and complexity (time, space, sample size) of any algorithm. [No]
   (c) (Optional) Anonymized source code, with specification of all dependencies, including external libraries. [No]

2. For any theoretical claim, check if you include:
   (a) Statements of the full set of assumptions of all theoretical results. [Not Applicable]
   (b) Complete proofs of all theoretical results. [Not Applicable]
   (c) Clear explanations of any assumptions. [Not Applicable]

3. For all figures and tables that present empirical results, check if you include:
   (a) The code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL). [No]
   (b) All the training details (e.g., data splits, hyperparameters, how they were chosen). [Yes]
   (c) A clear definition of the specific measure or statistics and error bars (e.g., with respect to the random seed after running experiments multiple times). [Yes]
   (d) A description of the computing infrastructure used. (e.g., type of GPUs, internal cluster, or cloud provider). [No]

4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets, check if you include:
   (a) Citations of the creator If your work uses existing assets. [Yes]
   (b) The license information of the assets, if applicable. [Not Applicable]
   (c) New assets either in the supplemental material or as a URL, if applicable. [Not Applicable]
   (d) Information about consent from data providers/curators. [Not Applicable]
   (e) Discussion of sensible content if applicable, e.g., personally identifiable information or offensive content. [Not Applicable]

5. If you used crowdsourcing or conducted research with human subjects, check if you include:
   (a) The full text of instructions given to participants and screenshots. [Not Applicable]
   (b) Descriptions of potential participant risks, with links to Institutional Review Board (IRB) approvals if applicable. [Not Applicable]
   (c) The estimated hourly wage paid to participants and the total amount spent on participant compensation. [Not Applicable]