In-Context Reinforcement Learning for Variable Action Spaces

Viacheslav Sinii [1 2] **Alexander Nikulin** [3 4 *] **Vladislav Kurenkov** [3 *] **Ilya Zisman** [3 *] **Sergey Kolesnikov** [1]

Abstract

Recently, it has been shown that transformers pre-trained on diverse datasets with multi-episode contexts can generalize to new reinforcement learning tasks in-context. A key limitation of previously proposed models is their reliance on a predefined action space size and structure. The introduction of a new action space often requires data re-collection and model re-training, which can be costly for some applications. In our work, we show that it is possible to mitigate this issue by proposing the Headless-AD model that, despite being trained only once, is capable of generalizing to discrete action spaces of variable size, semantic content and order. By experimenting with Bernoulli and contextual bandits, as well as a gridworld environment, we show that Headless-AD exhibits significant capability to generalize to action spaces it has never encountered, even outperforming specialized models trained for a specific set of actions on several environment configurations. Implementation is available at: `https://github.com/corl-team/headless-ad`.

1. Introduction

The transformer architecture, first introduced by Vaswani et al. (2017), has been widely adopted in key areas of machine learning, including natural language processing (Radford et al., 2018; Devlin et al., 2018), computer vision (Dosovitskiy et al., 2020) and sequential decision-making (Chen et al., 2021). One major feature of transformers is in-context learning (ICL), which makes it possible for them to adapt to new tasks after extensive pre-training (Brown et al., 2020; Liu et al., 2023). Recent developments, such as Algorithm Distillation (AD) by Laskin et al. (2022) and Decision Pretrained Transformer (DPT) by Lee et al. (2023), have successfully employed transformer ICL abilities in sequential decision-making. These models are capable of predicting the next action based on a query state and history of environment interactions, which inform them about task objectives and environment dynamics. While effective at generalizing across various reward distributions (Laskin et al., 2022; Lee et al., 2023) and transition functions (Raparthy et al., 2023), their adaptability to new action spaces remains unexplored and limited by architectural constraints.

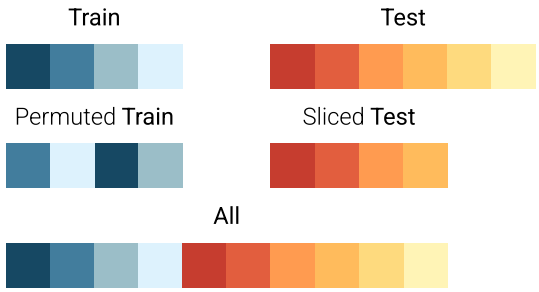

Figure 1. **Variable Action Spaces:** We consider four types of novel action spaces different from the one used during training. *Permuted Train Actions* maintains the action set contents but reorders its elements. *Test Actions* introduces a completely new action set with an increased size. It is important to consider that some models may be architecturally limited to a fixed action set size. To evaluate the performance of such models on unseen actions, we adjust the size of a new set to be compatible with the model output. Therefore, we slice the first actions from the *Test Actions* set. Lastly, a new action space might include both the seen *Train* and unseen *Test* actions, depicted as the *All Actions* set.

Creating models that can adapt to new action spaces is essential for building the foundation of decision-making systems in order to enable large-scale pretraining across various environments and address real-world problems (Jain et al., 2020; Chandak et al., 2020; London & Joachims, 2020; Jain et al., 2021). With this in mind, our research focuses on variable discrete action spaces, with the notion of variability illustrated in the Figure 1. In our study, we reveal the limitations of the Algorithm Distillation model from prior work, such as its diminished performance upon changes in action semantics as well as architectural constraints when handling varying action space sizes (see Figure 3).

[*]Work done while at Tinkoff [1]Tinkoff, Moscow, Russia [2]Innopolis University [3]AIRI, Moscow, Russia [4]MIPT. Correspondence to: Viacheslav Sinii <v.siniy@tinkoff.ai>.

Proceedings of the 41st International Conference on Machine Learning, Vienna, Austria. PMLR 235, 2024. Copyright 2024 by the author(s).

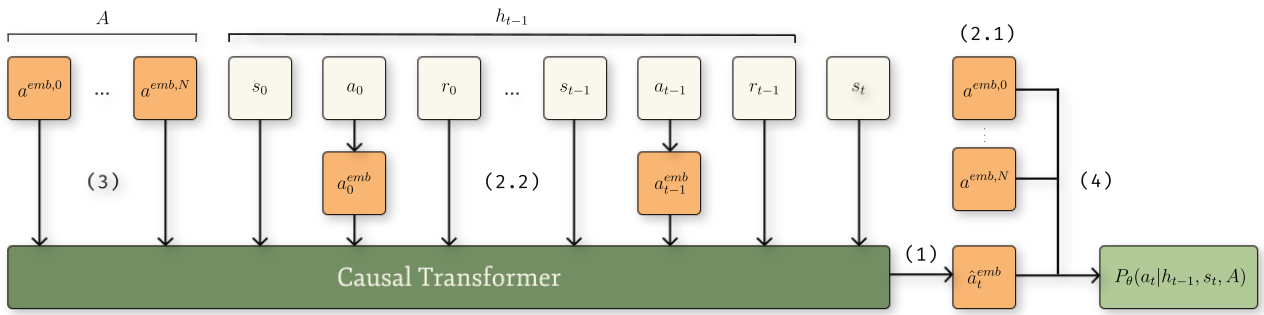

Figure 2. **Headless-AD Architecture:** Compared to AD, Headless-AD introduces four new components. (1) We remove the output linear head, making the model directly predict the action embedding. That allows us to avoid a direct connection between the model and action space size, contents and ordering. (2.1) At each training step, we generate random action embeddings for each action in the action set. (2.2) We convert actions in the context into their embeddings and pass them as the model input. This prepares the model for unseen actions, forcing it to infer action semantics from the context. (3) As the model loses prior knowledge about action space structure, we pass the generated action embeddings as a prompt to aid the model in sensible action selection. (4) We convert a prediction vector into a distribution over actions based on the similarities between the prediction and previously generated action embeddings. To increase the probability of correct actions, we use contrastive loss instead of cross-entropy.

Our solution, **Headless-AD**, is an architecture and training methodology tailored to effective generalization on new action spaces. We employ an approach similar to Wolterpinger (Dulac-Arnold et al., 2015) and Headless-LLM (Godey et al., 2023) by encoding actions with random embeddings (Kirsch et al., 2023) and directly predicting these embeddings. This way, we remove the direct connection between the model output layer and the action space structure.

Through experiments using Bernoulli and contextual bandits, and a darkroom environment with changing action spaces, we demonstrate that Headless-AD is capable of matching the performance of the original data generation algorithm and scaling to action spaces up to **5x** larger than those seen during training. We also observed that Headless-AD can even outperform AD when they are both trained for the same action space, especially when evaluated on larger action sets. To summarize, our contributions are as follows:

- We show that AD struggles with generalization on novel action spaces (Section 2).
- We extend AD with a modified model architecture and a training strategy, called **Headless-AD**, for it to acquire the ability to adapt to new discrete action spaces (Section 3). We demonstrate the strong generalization capabilities of Headless-AD on Bernoulli and contextual bandits, and darkroom environments (Section 4).
- We perform ablations on the loss and the prompt format to highlight the importance of Headless-AD's design choices (Section 5).

2. Algorithm Distillation Struggles with Novel Action Spaces

Algorithm Distillation (AD) (Laskin et al., 2022) is a transformer model trained to autoregressively predict the next action given the history of previous environment interactions and a current state. Formally, the history is defined as:

$$h_t = (o_0, a_0, r_0, \ldots, o_t, a_t, r_t)\,,$$

where o are the observations, a are the actions and r are the rewards. The probabilities of each action are given by a model $P_\theta(A = a_t^{(n)}|h_{t-1}, o_t)$, where n is the action index, t is a timestamp and θ are the model weights. AD is pretrained on data logged by a training agent, and the context size should be sufficiently large to span multiple episodes in order to capture policy improvement. This way, AD learns an improvement operator that increases performance entirely in-context when applied to novel tasks.

The model output is a probability distribution across the action set, derived from a linear projection and a softmax function. As highlighted in Figure 3, this structure causes a fundamental limitation in AD's adaptability to new action spaces, as the output dimension is predetermined. To accommodate an action set of a different size than the one used during training, the model's final layer must be redefined and the model retrained. Moreover, even with a constant action space size, the model's efficacy diminishes if the action semantics are altered. The reason for this is the classifier nature of the model, which associates each dimension with a particular meaning of an action. Since augmenting the dataset with permuted action sets does not lead to improvement, it signifies that action set invariance should be enforced from a model design standpoint.

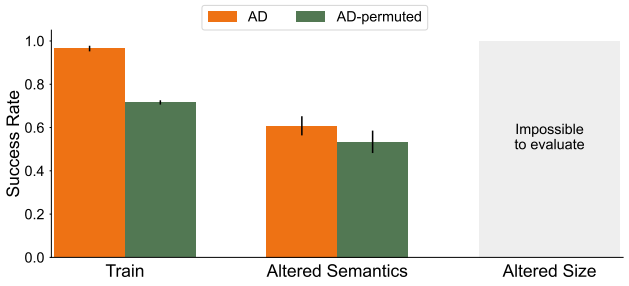

Figure 3. **Algorithm Distillation Struggles with Novel Action Spaces:** Despite its good results on the train action set, AD's performance diminishes when the action semantics change, either due to a permutation or substitution. It is important to note that augmenting the training data with permuted action sets does not lead to increased performance, signifying that action set invariance should be enforced from a model design standpoint. Additionally, it is impossible to apply a trained AD model to a larger action set. On the graph, the bars are the success rate values on the Darkroom environment (described in Section 4.3) obtained after evaluating each of the action sets visualized in Figure 1, averaged over 5 runs. *Altered Semantics* aggregate the values from the *Permuted Train Actions* and *Sliced Test Actions* sets. *Altered Size* aggregates the values from *Test Actions* and *All Actions*. See Section 4.3 for more information about the construction of the action sets.

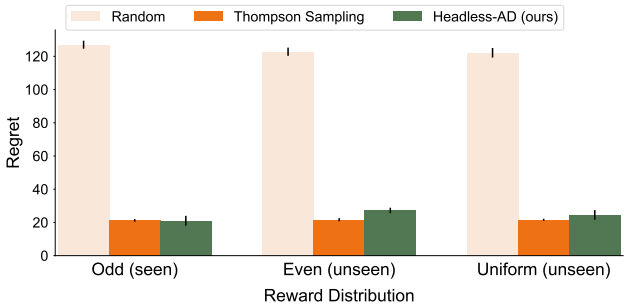

Figure 4. **Algorithm Regret under Variable Reward Distributions in Bernoulli Bandit:** The graph compares regret for Random, Thompson Sampling, and Headless-AD across distinct reward distributions in the Bernoulli Bandit environment, averaged from five seeds. During training, the high reward was 95% more likely to distribute across the odd arms. During testing, it either switched to the even arms or a uniform distribution. Note that Headless-AD maintains high performance in all configurations, proving its ICL capabilities at generalizing to novel tasks, represented by changes in reward distribution. Data is aggregated from bandit problems with $4 - 20$ arms, reflecting the training conditions.

3. Headless-AD

To mitigate the action space limitations of AD, we propose Headless-AD, a new architecture that improves on AD by omitting the final linear layer and incorporating three key modifications. The Headless-AD architecture and data flow are visualized in Figure 2.

Random Action Embeddings: To remove the dependence of the model on the pretrained action embeddings, we employed a dynamic mapping function $g : \mathbb{A} \to \mathbb{R}^n$, which produces a unique random encoding for each action in a batch at the start of every training step. The mapping is shared across all batch instances and is consistent along the context sequence, i.e., actions with index i in their respective action sets will all be mapped to the same embedding. During inference, we generated a single set of embeddings at the beginning and used them throughout the evaluation.

The core intuition behind employing random action embeddings is to eliminate any prior knowledge about the structure of the action space within our model. This approach stems from our observation that using learnable embeddings for actions becomes impractical when encountering new actions not seen during training. A new action would lack a pretrained embedding, and assigning an arbitrary embedding could introduce an undesirable domain shift, as the model would not recognize it. Moreover, allowing the model to learn new embeddings on-the-fly would necessitate extra gradient steps, diverging from our goal of maintaining a zero-shot learning framework. The usage of random action embeddings ensures that the model does not depend on extracting any information from the embeddings themselves but rather relies on interpreting the context provided by historical interactions with the environment. Moreover, employing random embeddings enhances data variety, which has been demonstrated to boost in-context learning for RL agents (Kirsch et al., 2023; Lu et al., 2023).

We further refined this strategy by constraining the random embeddings to lie on a unit sphere and ensuring their orthogonality (see Appendix H). The choice of a unit sphere normalizes the scale of the embeddings, while orthogonality allows the model to independently adjust the probability assigned to each action. This condition is crucial for preventing unintended probability mass allocation to multiple actions when the model's prediction vector aligns with one embedding vector. A similar concept is explored by (Elhage et al., 2022) in the context of feature interference.

Direct Prediction of Action Embeddings: The model output is modified to yield an action embedding $\hat{a}_t^{emb}$ rather than a probability distribution over actions. This alteration makes the model independent of action set size and order, granting it permutation invariance. The InfoNCE Contrastive Loss (Oord et al., 2018), diverging from its usual role in representation learning (Jaiswal et al., 2020), serves as a regression objective to reinforce the similarity between the model prediction and the subsequent action in the data. All other actions are treated as negative samples. Thus, the

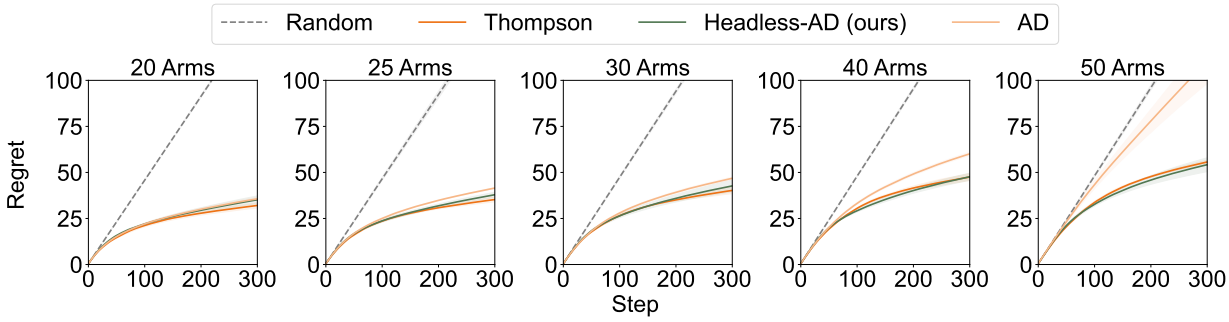

Figure 5. **Algorithm Regret under Increasing Amount of Arms in Bernoulli Bandit:** This series of plots shows the regret of Thompson Sampling, AD, and Headless-AD algorithms over evaluation steps in environments with $20-50$ arms, averaged from five seeds with 100 bandits each. Although Headless-AD has been trained on bandits with up to 20 arms, it performs well, matching or outperforming other algorithms in larger arm settings without additional training. Note that AD was retrained from scratch for each task with a different number of arms.

objective is

$$L = -\mathbb{E}\left[\log \frac{f(\hat{a}_t^{emb}, a_t^{emb})/\tau}{\sum_{a\in A} f(\hat{a}_t^{emb}, a^{emb})/\tau}\right],$$

where $a^{emb} = g(a)$ and τ is a temperature parameter. We used dot-product as the similarity function f.

Action Set Prompt: To address the model's lack of awareness of the action space structure caused by the two previous changes, we prepend the input with a sequence of embeddings for all available actions.

The modified input format is thus represented as

$$h_t = \left(a^{emb,0}, \ldots, a^{emb,N}, o_{<t}, a_{<t}^{emb}, r_{<t}, o_t\right),$$

where N is the action set size. An illustrative code snippet with Headless-AD's training procedure can be found in Appendix L.

We suggest two methods to select actions during inference: (1) nearest neighbor selection:

$$a = \arg\max_{a\in A} f(\hat{a}^{emb}, a^{emb})$$

and (2) probabilistic sampling based on the similarity to the predicted action embedding:

$$P(a_i) = \frac{f(\hat{a}^{emb}, a_i^{emb})}{\sum_{a\in A} f(\hat{a}^{emb}, a^{emb})}.$$

We treat the specific choice of method as a hyperparameter.

4. Experiments

As Headless-AD extends and improves on AD, we checked it in two different aspects. Firstly, it should maintain In-Context Learning abilities and thus generalize well to new tasks. Secondly, it should show high performance on action spaces different from the one seen during training. All of the following environments are designed specifically to check both of the above aspects. In our experiments, we used the TinyLLaMa (Zhang et al., 2024) implementation of the transformer model and AdamW optimizer (Loshchilov & Hutter, 2017). All environment specific hyperparameters are listed in Appendix J.

4.1. Bernoulli Bandit

Motivation: This experiment checked Headless-AD's abilities on a toy task, where the environment did not have a notion of state and returned binary rewards.

Setup 1: In our first experiment, we examined the model's robustness to distributional shifts in rewards. We used a Bernoulli bandit where each arm is associated with a mean μ and the reward after pulling an arm i is generated by Bernoulli(μ). The training dataset consisted of bandits with $4-20$ arms, i.e., different action set sizes. Additionally, to evaluate the ICL capabilities of models, the reward distribution over the arms differed between train and test (Laskin et al., 2022). Specifically, the training data consisted of bandits where in 95% of cases the odd-numbered arms were assigned a random $\mu \in [0.5, 1]$ and even-numbered arms received $\mu \in [0, 0.5]$. For other 5% of bandits, the ranges were swapped between odd and even arms. The test distribution also consisted of bandits with $4-20$ arms, but the reward distribution was switched either to even arms or was uniform, i.e., all arms were assigned $\mu \in [0, 1]$. Learning histories were generated using Thompson Sampling algorithm for a total of $10,000$ bandit instances with 300 steps each. The evaluation was performed on 100 bandits in each reward distribution, included 5 seeds, and the algorithms were rolled out for 300 steps.

Results and Discussion 1: As depicted in Figure 4, the

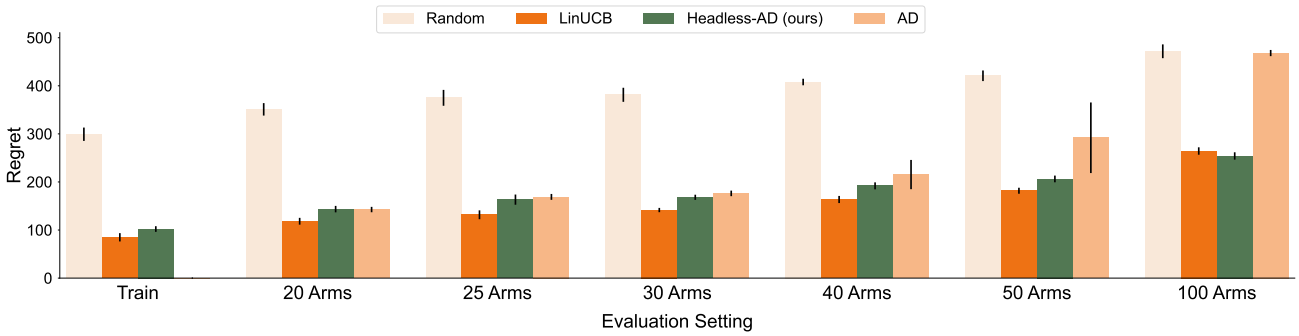

Figure 6. **Contextual Bandit Regret Comparison:** The *Train* set consists of trajectories of LinUCB trained for 300 steps on contextual bandits with $4-20$ arms. All models are also evaluated for 300 steps. The results are averaged over 5 seeds, each seed containing 100 environments. AD requires retraining on each new action set and, while showing good performance on lower space sizes, it fails to converge on larger ones. Due to its variable-size action sets, AD was not tested on the *Train* set. Conversely, Headless-AD, trained exclusively on the *Train* set, is successful at both learning effectively within this environment and generalizing to new action sets.

Headless-AD model demonstrates strong generalization capabilities by almost reaching the performance results set by the traditional Thompson Sampling algorithm under each test distribution.

Setup 2: In our second experiment, we evaluated the transferability of Headless-AD to new action set sizes. Training distribution remained the same as in the previous experiment. Each evaluation dataset consisted of a fixed amount of 20, 25, 30, 40 and 50 arms and a uniform distribution of rewards. AD was trained on fixed-size bandits corresponding to each evaluation dataset. The training and evaluation reward distributions were the same as for Headless-AD.

Results and Discussion 2: Figure 5 shows that Headless-AD can effectively maintain its performance as the action space grows, without necessitating any retraining. Moreover, Headless-AD even outperforms a specially trained AD, especially on larger action sets. Note that Headless-AD's performance curves resemble TS's performance curves, signifying that it has indeed learned a policy improvement operator generic enough to apply to unseen action set sizes.

4.2. Contextual Bandit

Motivation: To validate the sustained performance of Headless-AD, we progressed to a more complex Multi-Armed Bandit (MAB) extension that integrated states and real-valued rewards.

Setup: Each time step presented a context with arm features modeled as two-dimensional vectors, and rewards were generated with a standard deviation $\sigma = 1$. The data were created using a LinUCB (Li et al., 2010) algorithm trained over 300 steps, on bandits with $4-20$ arms. We used 100 contextual bandits, 5 seeds and 300 evaluation steps.

Results and Discussion: As shown in Figure 6, Headless-AD's performance is on par with LinUCB across varied arm counts. While AD also reaches the performance of LinUCB on lower space sizes, it has problems converging on larger action space sizes, in addition to requiring a specific retraining. This serves to highlight the advantages of Headless-AD for use in even more complex environments than toy Bernoulli bandits.

4.3. Darkroom

Motivation: In this experiment, we delve into a more sophisticated Markov Decision Process (MDP) framework, constructing five distinct action spaces aimed at demonstrating the architectural constraints of AD. We then show how Headless-AD's architecture is engineered to navigate the complexities presented by each of these diverse action sets.

Setup: The Darkroom environment, inspired by Chevalier-Boisvert et al. (2018) and Jain et al. (2020), consists of a $N \times N$ grid where the agent needs to reach a specific cell for a reward. In our experiment, the action space consisted of 3-step sequences of 5 atomic actions: *up*, *down*, *left*, *right*, *noop*. As a result, the environment offered 5^3 possible actions. The agent earned a reward of 1 if the trajectory induced by the action sequence passed through a goal cell, after which the episode finished. Otherwise, the reward was 0. As an observation, the environment offered only the current coordinates of the agent, so the goal information could only be obtained from the agent's memory. We divided the goals into disjoint sets used for training and testing in order to evaluate Headless-AD's in-context learning abilities. Furthermore, the action set was randomly split into train and test sets, each including 50 and 75 actions respectively, to create five distinct spaces for assessing various generalization aspects. These action sets are visualized in Figure 1.

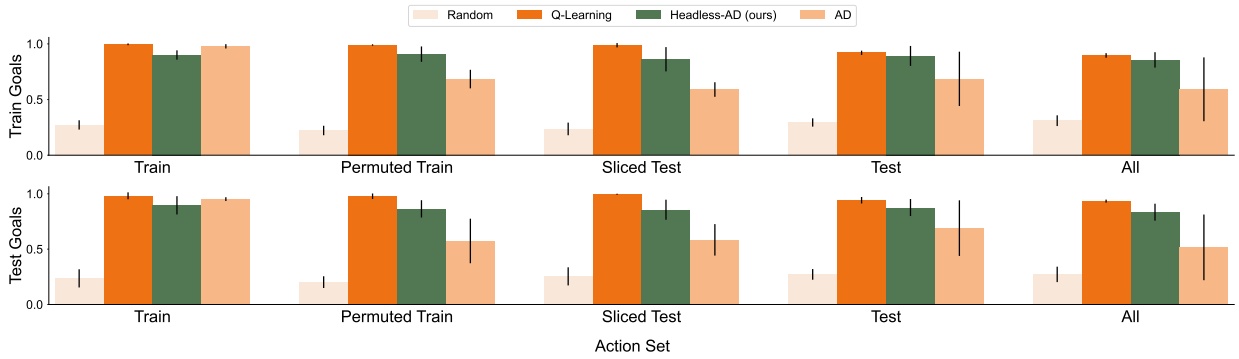

Figure 7. **Darkroom Environment Success Rate:** The chart displays mean success rates and their standard deviations from five training seeds, comparing performance in fixed and variable action spaces, along with the models' adaptability to new goals. *Train Actions* refers to the fixed-size action set used for training. *Test Actions* include exclusively unseen actions, while *All Actions* combine both *Train* and *Test*, with set sizes expanded to 75 and 125 respectively. Since the output dimension changes, AD requires retraining. To assess AD's adaptability to the changed action semantics, we either permute *Train Actions* or replace it with the first 50 actions from the *Test* set. In this case, AD does not require retraining, but it still exhibits diminished performance. Conversely, Headless-AD, while trained solely on *Train Actions*, delivers strong and stable performance across all action set variants, and even surpasses specially trained AD on larger set sizes.

Train Actions. Comprising the training split, this set represented the actions encountered during model training.

Test Actions. Comprising the test split, this set assessed Headless-AD's generalization on novel and larger action sets.

All Actions. Combining both training and testing actions, this set contained 125 actions and was 2.5 times larger than the training set. Its aim was to challenge Headless-AD to effectively integrate seen and unseen actions.

Permuted Train Actions. Shuffled training set, meant to test the model's adaptability to reordered action spaces that comprised of the same actions.

Sliced Test Actions. Tailored to match the training set size, this set contained a slice of the first 50 actions from the test set. Its aim was to check the models' generalization abilities on unseen actions while maintaining the action set size.

In scenarios where the action space exceeded the size of the training set, we analyzed the performance of an AD model retrained from scratch. The data generation algorithm was Q-learning, executed over 200 episodes for each environment. Further details on model hyperparameters are available in Appendix J.

Results and Discussion: Figure 7 illustrates the generalization abilities of both AD and Headless-AD. First, note that both models maintain their performance when transitioning to novel tasks, thus fulfilling their purpose as ICL-RL models. However, this environment was mainly designed to challenge the models' generalization abilities on novel action spaces. Following the scope of the action space novelty we set earlier, we studied the models' ability to address changed action semantics and variable action set sizes.

While AD achieves high performance on the train action set, its limitations become evident when the action space is changed. The first limitation is AD's action set size constraint. The linear layer at the end of its network fixes the size of the output dimension, something that cannot be modified once the model is trained. Increasing the amount of options, as was done in the *Test Actions* and *All Actions* sets, requires reinitializing the output dimension and thus retraining the model, which demands additional time and resources. In contrast, once trained, Headless-AD easily adapts to changes in the action set size without losing performance.

The second limitation is AD's reliance on a stationary action space structure. Due to the classifier nature of AD's network, it learns to associate each output dimension with a specific action meaning. When action semantics change either due to a permutation of seen actions, as in *Permuted Train Actions*, or due to a substitution of completely new actions, as in *Sliced Test Actions*, AD's performance degrades. We checked whether this problem may be solved by biasing the training distribution to have a structure resembling the one during testing. We trained AD on permuted action sets while leaving the data and the model architecture the same. However, as one can see in Figure 3, this training procedure resulted only in a slightly decreased performance of AD in the *Permuted Train* setting. An additional graph depicting the performance on each of the (action set, goal type) pairs can be found in Appendix C. Meanwhile, *Permuted Train Actions* does not pose a challenge for Headless-AD as, by design, it is invariant to specific action order. Most importantly, Headless-AD maintains its performance even

Table 1. **Ablations:** Table compares the performance of Headless-AD with its ablated versions. Columns 'Bandit' and 'Darkroom' show the performance averaged along all action sets. Bernoulli Bandit performance is normalized, where 0 denotes a random agent and 1 denotes the Thompson Sampling algorithm. Columns 'Bandit. Arms Used' and 'Darkroom. Arms Used' show the amount of actions tried by the model during evaluation, also averaged along all the action sets. Columns 'Bandit. *N* Arms' show the performance of models on the Bernoulli Bandit environment for each respective number of arms during evaluation without averaging. The results are aggregated over 5 random seeds. As the table shows, changing each component of Headless-AD's architecture greatly damages the model's ability either to utilize the action set effectively or to perform well.

Setting	Bandit	Bandit. Arms Used	Darkroom	Darkroom. Acts Used	Bandit. 20 Arms	Bandit. 30 Arms	Bandit. 40 Arms	Bandit. 50 Arms
Headless-AD	0.98 ± 0.02	19.27 ± 9.26	0.87 ± 0.03	57.21 ± 21.89	0.97 ± 0.02	0.98 ± 0.02	1.0 ± 0.02	1.02 ± 0.04
Prompt Ablation	0.81 ± 0.12	22.28 ± 13.11	0.83 ± 0.03	54.69 ± 18.63	0.91 ± 0.03	0.81 ± 0.05	0.66 ± 0.03	0.55 ± 0.03
Loss Ablation	0.63 ± 0.08	2.4 ± 0.22	0.25 ± 0.04	70.0 ± 29.15	0.6 ± 0.22	0.64 ± 0.27	0.7 ± 0.22	0.8 ± 0.25
Embed Ablation	0.79 ± 0.14	23.78 ± 12.54	0.81 ± 0.03	55.72 ± 19.29	0.85 ± 0.01	0.74 ± 0.03	0.62 ± 0.04	0.52 ± 0.04

when completely new actions are introduced, despite having never seen them in training. We attribute this feature to our success in making the model infer action meaning from the context.

5. Ablations

The hyperparameters tuned for each ablation can be found in Appendix J.

5.1. Action Set Prompt

We assessed the impact of omitting the action embedding enumeration from the model context.

Table 1 reveals that removing the action set prompt did not significantly alter the number of unique actions attempted by each model. This suggests that models without the prompt continue to sample a diverse range of actions, comparable to the original setup. However, the presence of the prompt did affect the performance. We hypothesize that, without the action set prompt, the model lacked explicit knowledge of the action space, which may have led to a more randomized sampling within the embedding space. Conversely, with the prompt in place, the model could directly target specific action embeddings.

The absence of the prompt was more detrimental in the Bernoulli Bandit environment, where each decision has a direct impact on the final outcome due to the environment's single-episode structure. However, in a multi-episode environment such as the Darkroom, the early lack of action space information becomes less impactful over time, as the model eventually encounters all actions through the context. This divergence highlights the prompt's critical role in enabling the model to make informed choices in environments where each option has immediate and lasting consequences. Note that the impact of action set prompt ablation on the performance got more pronounced as the number of arms grows.

5.2. Contrastive Loss

In principle, the prediction of action embedding can be facilitated by various ways. Here, we considered the performance of a model that was asked to directly copy the corresponding embedding from its context to the output. To achieve it, we used mean squared error (MSE) loss instead of contrastive loss.

In this case, probabilistic interpretation became less meaningful, which is why the nearest neighbor of the predicted vector was chosen as an action index, without sampling.

Table 1 illustrates a significant decline in the variety of actions attempted by the model under this configuration within the bandit environment, showing that the model concentrated only on a fraction of the action set. Though the final regrets for this model were far from random, we point out that this result is because the model skipped exploration phase and went directly to an exploitation of suboptimal actions. That allowed it not to waste time on even more suboptimal ones via exploration. Appendix G shows the in-context curves of both models, providing more insight into MSE's effect on model quality.

Conversely, on the Darkroom environment, the loss substitution led to a marginal increase in the amount of attempted actions. However, this did not lead to an increased performance. In fact, the performance of this model on the Darkroom environment was similar to a random behavior.

In summary, although the employed neural network architecture was capable of learning an improvement operator and demonstrating ICL capabilities, the implemented loss function played a crucial role in the success of this learning process. We emphasize the importance of our design choice to use contrastive loss by showing that a more naive approach of directly copying the action embeddings, as incentivized by MSE loss, resulted in underperformance.

5.3. Orthonormal Action Embeddings

In this ablation study, we underscored the significance of employing orthonormal vectors for action embeddings by contrasting them with vectors derived from a standard normal

distribution. Our preference for the former stemmed from their ability to simplify the approximation of action probabilities because of their property of independence. Unlike embeddings from a standard normal distribution, orthonormal vectors ensure that assigning a probability weight to one vector does not unintentionally influence another (we illustrate it in Appendix K). This concept echoes the principle of *superposition*, observed when models incorporate more features than available dimensions, leading to feature interference (Elhage et al., 2022).

Table 1 illustrates the consequences of using linearly dependent action embeddings, manifesting as diminished performance across Bernoulli bandits and Darkroom scenarios. Notably, these results bear a striking resemblance to "Prompt Ablation", with both indicating a slight performance dip in Darkroom, a more noticeable decline in Bernoulli Bandits, and a general downtrend as the number of bandit arms grows. This parallel underscores a shared objective between these design choices: refining the model's capacity for precise action selection.

6. Related Work

Here, we discuss previous research in adapting RL to environments with variable action spaces. For an extended Literature Review, see Appendix B.

Recent research by Chandak et al. (2020); Ye et al. (2023) has focused on scenarios where the amount of available actions grows during evaluation by introducing new actions to an existing set. However, their model requires fine-tuning when new actions are introduced, while our model does not require parameter updates. Additionally, our research explores a broader spectrum of dynamic action spaces, encompassing the addition, removal, and substitution of actions. Lastly, we work in a setting where the action set remains constant throughout model evaluation.

Kirsch et al. (2022) present the concept of SymLA, a methodology that ensures resilience to changes in input and output sizes and permutations. This is achieved by integrating symmetries into a neural network model by representing each neuron with uniformly structured RNNs and data flow with message-passing connections. However, we suggest a simpler approach by extending the well-known transformer architecture. Additionally, we demonstrate the high performance of our model on more complex environments.

Further developments by Jain et al. (2020) include a specific module designed to generate action representations informed by observed trajectories after action application. Additionally, Jain et al. (2021) delve into variable action spaces, placing a specific emphasis on modeling the interconnections among actions to improve the quality of action representations. Our work, however, adopts a more implicit approach to inferring action-related information.

In a similar work, Lu et al. (2023); Kirsch et al. (2023) employ random projections for action encoding, a technique we also utilize. This method facilitates training across multiple domains with actions of varying sizes. Nonetheless, their focus is predominantly on continuous spaces, while our focus is on discrete action spaces.

To the best of our knowledge, Headless-AD is the first study to explore variable discrete action spaces for in-context reinforcement learning.

7. Conclusion

In our work, we have introduced a new architecture that extends Algorithm Distillation (AD) for environments with variable action spaces, achieving invariance to their structure and size. Our approach consists of discarding the last linear layer, granting invariance to the action space structure, and making the model infer action semantics from the context, preparing it for the introduction of novel actions. We demonstrated Headless-AD's capability to generalize across new action spaces on a set of environments. We also observed its performance gains over vanilla AD, especially on larger action spaces. We hypothesize that this is due to the augmentation of the dataset by random embeddings, which was shown to improve the generalization abilities of agents (Kirsch et al., 2023). Headless-AD marks the progress toward versatile foundational models in RL, ones capable of operating across an expanded range of environments. We hope that Headless-AD inspires further development of models that can adapt to any action space beyond discrete ones.

Limitations. Headless-AD shares AD's limitation of fixed sequence lengths, which may limit its effectiveness in environments with long episodes. A unique constraint of Headless-AD is its limit on the number of actions, dictated by the dimensionality of the action embeddings – only as many actions as there are dimensions can be orthogonally represented. Going over this limit causes embeddings to become linearly dependent, unintentionally distributing probability across multiple actions. While action spaces in RL environments typically do not become excessively large, and increasing the dimension of embeddings could mitigate this issue, it remains a point for consideration.

Future Work. In our study, we demonstrated the ability of our algorithm to generalize to new action spaces, as shown by its performance on elementary tasks. To extend and validate these findings, future research should focus on more complex environments. This will offer a deeper insight into the algorithm's versatility and robustness in diverse and complex settings.

Additionally, to ensure the broader applicability and adaptability of our approach, it is essential to examine its compatibility and performance with various models beyond Algorithm Distillation (AD) (Laskin et al., 2022), such as the Decision Pretrained Transformer (DPT) (Lee et al., 2023). This exploration will provide a more comprehensive understanding of the algorithm's strengths and limitations, potentially leading to further improvements and a wider scope of applications in different contexts.

Impact Statement

Strong safeguards are necessary to prevent unauthorized users from manipulating the model, such as adding harmful action embeddings that could lead to negative outcomes.

Another concern is how the model handles out-of-distribution data. If the model encounters a new action that is significantly different from the training actions, it may take a while to understand its effects. Since our model learns by trying out actions, there is a risk it might perform harmful actions before learning they are inappropriate.

Any application of Headless-AD in real-life scenarios should be aware of these potential risks.

References

Biewald, L. Experiment tracking with weights and biases, 2020. URL https://www.wandb.com/. Software available from wandb.com.

Brown, T., Mann, B., Ryder, N., Subbiah, M., Kaplan, J. D., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33: 1877–1901, 2020.

Chandak, Y., Theocharous, G., Nota, C., and Thomas, P. Lifelong learning with a changing action set. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pp. 3373–3380, 2020.

Chen, L., Lu, K., Rajeswaran, A., Lee, K., Grover, A., Laskin, M., Abbeel, P., Srinivas, A., and Mordatch, I. Decision transformer: Reinforcement learning via sequence modeling. *Advances in neural information processing systems*, 34:15084–15097, 2021.

Chevalier-Boisvert, M., Willems, L., and Pal, S. Minimalistic gridworld environment for openai gym. 2018.

Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

Dorfman, R., Shenfeld, I., and Tamar, A. Offline meta reinforcement learning–identifiability challenges and effective data collection strategies. *Advances in Neural Information Processing Systems*, 34:4607–4618, 2021.

Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S., et al. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020.

Duan, Y., Schulman, J., Chen, X., Bartlett, P. L., Sutskever, I., and Abbeel, P. Rl ^{2}: Fast reinforcement learning via slow reinforcement learning. *arXiv preprint arXiv:1611.02779*, 2016.

Dulac-Arnold, G., Evans, R., van Hasselt, H., Sunehag, P., Lillicrap, T., Hunt, J., Mann, T., Weber, T., Degris, T., and Coppin, B. Deep reinforcement learning in large discrete action spaces. *arXiv preprint arXiv:1512.07679*, 2015.

Elhage, N., Nanda, N., Olsson, C., Henighan, T., Joseph, N., Mann, B., Askell, A., Bai, Y., Chen, A., Conerly, T., DasSarma, N., Drain, D., Ganguli, D., Hatfield-Dodds, Z., Hernandez, D., Jones, A., Kernion, J., Lovitt, L., Ndousse, K., Amodei, D., Brown, T., Clark, J., Kaplan, J., McCandlish, S., and Olah, C. A mathematical framework for transformer circuits. *Transformer Circuits Thread*, 2021. https://transformer-circuits.pub/2021/framework/index.html.

Elhage, N., Hume, T., Olsson, C., Schiefer, N., Henighan, T., Kravec, S., Hatfield-Dodds, Z., Lasenby, R., Drain, D., Chen, C., Grosse, R., McCandlish, S., Kaplan, J., Amodei, D., Wattenberg, M., and Olah, C. Toy models of superposition. *Transformer Circuits Thread*, 2022. URL https://transformer-circuits.pub/2022/toy_model/index.html.

Godey, N., de la Clergerie, É., and Sagot, B. Headless language models: Learning without predicting with contrastive weight tying. *arXiv preprint arXiv:2309.08351*, 2023.

Gu, A., Goel, K., and Ré, C. Efficiently modeling long sequences with structured state spaces. *arXiv preprint arXiv:2111.00396*, 2021.

Hafner, D., Lillicrap, T., Ba, J., and Norouzi, M. Dream to control: Learning behaviors by latent imagination. *arXiv preprint arXiv:1912.01603*, 2019.

Hu, S., Shen, L., Zhang, Y., Chen, Y., and Tao, D. On transforming reinforcement learning by transformer: The development trajectory. *arXiv preprint arXiv:2212.14164*, 2022.

Jain, A., Szot, A., and Lim, J. J. Generalization to new actions in reinforcement learning. *arXiv preprint arXiv:2011.01928*, 2020.

Jain, A., Kosaka, N., Kim, K.-M., and Lim, J. J. Know your action set: Learning action relations for reinforcement learning. In *International Conference on Learning Representations*, 2021.

Jaiswal, A., Babu, A. R., Zadeh, M. Z., Banerjee, D., and Makedon, F. A survey on contrastive self-supervised learning. *Technologies*, 9(1):2, 2020.

Janner, M., Li, Q., and Levine, S. Offline reinforcement learning as one big sequence modeling problem. *Advances in neural information processing systems*, 34: 1273–1286, 2021.

Kirsch, L., Flennerhag, S., van Hasselt, H., Friesen, A., Oh, J., and Chen, Y. Introducing symmetries to black box meta reinforcement learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, pp. 7202–7210, 2022.

Kirsch, L., Harrison, J., Freeman, C. D., Sohl-Dickstein, J., and Schmidhuber, J. Towards general-purpose in-context learning agents. In *NeurIPS 2023 Workshop on Distribution Shifts: New Frontiers with Foundation Models*, 2023.

Laskin, M., Wang, L., Oh, J., Parisotto, E., Spencer, S., Steigerwald, R., Strouse, D., Hansen, S., Filos, A., Brooks, E., et al. In-context reinforcement learning with algorithm distillation. *arXiv preprint arXiv:2210.14215*, 2022.

Lee, J. N., Xie, A., Pacchiano, A., Chandak, Y., Finn, C., Nachum, O., and Brunskill, E. Supervised pretraining can learn in-context reinforcement learning. *arXiv preprint arXiv:2306.14892*, 2023.

Lee, K.-H., Nachum, O., Yang, M. S., Lee, L., Freeman, D., Guadarrama, S., Fischer, I., Xu, W., Jang, E., Michalewski, H., et al. Multi-game decision transformers. *Advances in Neural Information Processing Systems*, 35: 27921–27936, 2022.

Li, L., Chu, W., Langford, J., and Schapire, R. E. A contextual-bandit approach to personalized news article recommendation. In *Proceedings of the 19th international conference on World wide web*, pp. 661–670, 2010.

Li, L., Yang, R., and Luo, D. Focal: Efficient fully-offline meta-reinforcement learning via distance metric learning and behavior regularization. *arXiv preprint arXiv:2010.01112*, 2020.

Li, L., Huang, Y., Chen, M., Luo, S., Luo, D., and Huang, J. Provably improved context-based offline meta-rl with attention and contrastive learning. *arXiv preprint arXiv:2102.10774*, 2021.

Li, W., Luo, H., Lin, Z., Zhang, C., Lu, Z., and Ye, D. A survey on transformers in reinforcement learning. *arXiv preprint arXiv:2301.03044*, 2023.

Lin, L., Bai, Y., and Mei, S. Transformers as decision makers: Provable in-context reinforcement learning via supervised pretraining. *arXiv preprint arXiv:2310.08566*, 2023.

Lin, Q., Liu, H., and Sengupta, B. Switch trajectory transformer with distributional value approximation for multi-task reinforcement learning. *arXiv preprint arXiv:2203.07413*, 2022.

Liu, P., Yuan, W., Fu, J., Jiang, Z., Hayashi, H., and Neubig, G. Pre-train, prompt, and predict: A systematic survey of prompting methods in natural language processing. *ACM Computing Surveys*, 55(9):1–35, 2023.

London, B. and Joachims, T. Offline policy evaluation with new arms. 2020.

Loshchilov, I. and Hutter, F. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*, 2017.

Lu, C., Schroecker, Y., Gu, A., Parisotto, E., Foerster, J., Singh, S., and Behbahani, F. Structured state space models for in-context reinforcement learning. *arXiv preprint arXiv:2303.03982*, 2023.

Mirchandani, S., Xia, F., Florence, P., Ichter, B., Driess, D., Arenas, M. G., Rao, K., Sadigh, D., and Zeng, A. Large language models as general pattern machines. *arXiv preprint arXiv:2307.04721*, 2023.

Nikulin, A., Kurenkov, V., Zisman, I., Sinii, V., Agarkov, A., and Kolesnikov, S. XLand-minigrid: Scalable meta-reinforcement learning environments in JAX. In *Intrinsically-Motivated and Open-Ended Learning Workshop, NeurIPS2023*, 2023. URL https://openreview.net/forum?id=xALDC4aHGz.

Oord, A. v. d., Li, Y., and Vinyals, O. Representation learning with contrastive predictive coding. *arXiv preprint arXiv:1807.03748*, 2018.

Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., Desmaison, A., Kopf, A., Yang, E., DeVito, Z., Raison, M., Tejani, A., Chilamkurthy, S., Steiner, B., Fang, L., Bai, J., and Chintala, S. Pytorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems 32*, pp. 8024–8035. Curran Associates, Inc., 2019.

Radford, A., Narasimhan, K., Salimans, T., Sutskever, I., et al. Improving language understanding by generative pre-training. 2018.

Rajeswaran, A., Lowrey, K., Todorov, E. V., and Kakade, S. M. Towards generalization and simplicity in continuous control. *Advances in Neural Information Processing Systems*, 30, 2017.

Raparthy, S. C., Hambro, E., Kirk, R., Henaff, M., and Raileanu, R. Generalization to new sequential decision making tasks with in-context learning. *arXiv preprint arXiv:2312.03801*, 2023.

Saxe, A. M., McClelland, J. L., and Ganguli, S. Exact solutions to the nonlinear dynamics of learning in deep linear neural networks. *arXiv preprint arXiv:1312.6120*, 2013.

Schroff, F., Kalenichenko, D., and Philbin, J. Facenet: A unified embedding for face recognition and clustering. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 815–823, 2015.

Sutton, R. S. and Barto, A. G. *Reinforcement learning: An introduction*. MIT press, 2018.

Team, O. E. L., Stooke, A., Mahajan, A., Barros, C., Deck, C., Bauer, J., Sygnowski, J., Trebacz, M., Jaderberg, M., Mathieu, M., et al. Open-ended learning leads to generally capable agents. *arXiv preprint arXiv:2107.12808*, 2021.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., and Polosukhin, I. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

Wang, J., Blaser, E., Daneshmand, H., and Zhang, S. Transformers learn temporal difference methods for in-context reinforcement learning, 2024.

Wang, J. X., Kurth-Nelson, Z., Tirumala, D., Soyer, H., Leibo, J. Z., Munos, R., Blundell, C., Kumaran, D., and Botvinick, M. Learning to reinforcement learn. *arXiv preprint arXiv:1611.05763*, 2016.

Wang, X., Wang, W., Cao, Y., Shen, C., and Huang, T. Images speak in images: A generalist painter for in-context visual learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 6830–6839, 2023.

Weinberger, K. Q. and Saul, L. K. Distance metric learning for large margin nearest neighbor classification. *Journal of machine learning research*, 10(2), 2009.

Xu, M., Shen, Y., Zhang, S., Lu, Y., Zhao, D., Tenenbaum, J., and Gan, C. Prompting decision transformer for few-shot policy generalization. In *international conference on machine learning*, pp. 24631–24645. PMLR, 2022.

Ye, J., Li, X., Wu, P., and Wang, F. Action pick-up in dynamic action space reinforcement learning. *arXiv preprint arXiv:2304.00873*, 2023.

Zhang, A., Ballas, N., and Pineau, J. A dissection of overfitting and generalization in continuous reinforcement learning. *arXiv preprint arXiv:1806.07937*, 2018.

Zhang, P., Zeng, G., Wang, T., and Lu, W. Tinyllama: An open-source small language model, 2024.

Zisman, I., Kurenkov, V., Nikulin, A., Sinii, V., and Kolesnikov, S. Emergence of in-context reinforcement learning from noise distillation. *arXiv preprint arXiv:2312.12275*, 2023.

A. Background

Partially Observable Markov Decision Process. A Markov Decision Process (MDP) is defined by a tuple (S, A, P, R), where $s_t \in S$ denotes a state, $a_t \in A$ an action, $p(s'|s_t = s, a_t = a)$ the transition probability from state s_t to s_{t+1} after taking action a_t, and $R(s, a)$ the reward for action a in state s (Sutton & Barto, 2018). An agent π observes the state s_t, selects action $a_t \sim \pi(\cdot|s_t)$, and receives the subsequent state $s_{t+1} \sim P(\cdot|s_t, a_t)$ and reward $R(s_t, a_t)$. In POMDPs, the agent receives an observation o_t instead of the full state s, which contains partial information about the MDP's real state. In the context of our work, o_t may lack goal information, requiring inference from the agent's memory.

Multi-Armed Bandits. A Bernoulli multi-armed bandit (MAB) environment consists of N arms $a_i \in A$, each associated with a mean μ_i (Sutton & Barto, 2018). Pulling an arm a_i yields a reward $r_i \sim \text{Bernoulli}(\mu_i)$. The agent's objective is to identify the arm with the highest μ_i. Performance is measured using regret, calculated as $\sum_t(\mu_{a^*} - \mu_{\hat{a}_t})$. Unlike MDPs, MABs lack states.

In a Contextual MAB, each arm a has a feature vector $x_a \in R^d$ (Sutton & Barto, 2018). At each step t, the agent observes a context state s_t. Selecting action a_i results in a reward from a normal distribution with mean $\mu_t = \langle s_t, a_i \rangle$ and standard deviation σ. Here, unlike in MDPs, the actions influence immediate rewards but not future states.

In-Context Learning. In-Context Learning describes the capability of a model to infer its task from the context it is given. For instance, the GPT-3 model (Brown et al., 2020) can be prompted in natural language to perform a variety of functions such as text classification, summarization, and translation, despite not being explicitly trained for these specific tasks. One of the possible prompts, which is used in our paper in some form, is a list of example pairs (x_i, y_i) ending with a query x_q for which the model is expected to generate a corresponding prediction $\hat{y}_q$.

Contrastive Learning. Contrastive learning focuses on creating representations where "similar" examples are close together in the feature space, while "dissimilar" ones are far apart (Weinberger & Saul, 2009; Schroff et al., 2015). This concept may be encapsulated in a triplet loss formula, where the similarity between an anchor and a positive example is maximized, and that between an anchor and a negative example is minimized, expressed as $L = sim(x, x^+) - sim(x, x^-)$. Oord et al. (2018) has developed a variant of contrastive loss called InfoNCE. This loss was lately widely adopted for representation learning (Jaiswal et al., 2020).

B. Related Work

B.1. Transformers in Reinforcement Learning

According to the survey by Li et al. (2023), Transformers (Vaswani et al., 2017) are increasingly utilized in reinforcement learning (RL) for various tasks, including representation learning of individual observations and their histories, as well as model learning, as seen in Dreamer by Hafner et al. (2019). In our research, we focus on the application of Transformers in sequential decision-making and in developing generalist agents. Same as AD (Laskin et al., 2022), Headless-AD also utilizes transformers as the base model for our approach.

The incorporation of transformers as a sequence modeling tool in RL began with the Decision Transformer (DT) by (Hu et al., 2022), which is trained autoregressively on offline datasets of state, action, and return-to-go tuples. Unlike conventional RL, which focuses on return maximization, DT generates appropriate actions during inference by conditioning on specified return-to-go values. The Trajectory Transformer by Janner et al. (2021) is an alternative approach using beam search to bias trajectory samples based on future cumulative rewards. Building on this, the Multi-Game Decision Transformer (MGDT) by Lee et al. (2022) improves upon DT with enhanced transfer learning capabilities for new games, eliminating the need for manual return specification. Similarly, the Switch Trajectory Transformer by (Lin et al., 2022) expands on the Trajectory Transformer to facilitate multi-task training. However, when transitioning to novel tasks, these approaches require fine-tuning the model.

B.2. Offline Meta-RL

Traditional RL agents, tailored to specific environments, struggle with novel tasks. In contrast, by training across a diverse array of tasks, Meta-RL equips agents with adaptable exploration strategies and an understanding of common environmental patterns (Duan et al., 2016; Wang et al., 2016; Rajeswaran et al., 2017; Zhang et al., 2018; Team et al., 2021; Nikulin et al., 2023). Offline Meta-RL, a subset of this approach, trains agents solely on pre-existing datasets, without direct environmental interaction. A critical challenge here is MDP ambiguity, where task-specific policies misinterpret data due to dataset biases.

Dorfman et al. (2021) propose a data collection method to mitigate this issue, and suggest an approach that treats Meta-RL as a Bayesian RL problem for optimal exploration in new tasks. Li et al. (2020) introduce FOCAL, a framework separating task identification from control. However, their method relies on a strict mapping assumption that fails in certain scenarios, such as those with sparse rewards. Li et al. (2021) enhance FOCAL with an attention mechanism and improved metric learning, showing greater robustness in scenarios with sparse rewards and domain shifts. However, most Offline Meta-RL methods rely on explicit task modeling, which can introduce limiting biases. Alternatively, In-Context Learning implicitly infers tasks from environmental interactions, offering a potentially more flexible approach.

B.3. In-Context Learning in RL

In-Context Learning (ICL) is an ability of a pretrained model to adapt and perform a new task given a context with examples $\{x_i, f(x_i)\}^n$, where f is a function that gives ground truth targets (Brown et al., 2020; Wang et al., 2023). Previous work on ICL (Mirchandani et al., 2023) showed that Large Language Models operate as General Pattern Machines and are able to complete, transform and improve token sequences, even when the sequences consist of randomly sampled tokens. This ability supports our design choice to randomly encode the actions of agents. The research on Transformer Circuits (Elhage et al., 2021) gives evidence of Transformers' ability to copy tokens, either literally or on a more abstract level, explaining their ICL capabilities. This is particularly useful for our model, which explicitly incentivizes an abstract copying of the action embeddings.

Efforts to blend In-Context Learning (ICL) abilities of transformers with reinforcement learning (RL) are gaining traction, promising to create adaptable RL agents for real-world scenarios with varying conditions. Attempts to transfer ICL features to RL include Prompt-Based Decision Transformers (Xu et al., 2022), which leverage task-specific demonstration datasets as prompts, exhibiting strong few-shot learning without weight updates. Laskin et al. (2022) introduced Algorithm Distillation (AD) that trains a policy improvement operator using data from agent-environment interactions of learning RL algorithm, predicting the next action autoregressively. A key strategy here is across-episode training for capturing policy improvement. Lee et al. (2023) developed Decision Pretrained Transformer (DPT) using supervised training with unordered environmental interactions as context to predict optimal actions. Under certain conditions, DPT can be proved to implement posterior sampling, resulting in near-optimal exploration. However, DPT's reliance on an optimal policy during training, not always available, is a limitation, similar to AD's constraint on action space structure, restricting their use as fully generalist agents. Lin et al. (2023) provided theoretical analysis of AD and DPT, offering guarantees about learning from base algorithms. Wang et al. (2024) also provided a theoretical analysis showing that transformers may implement several RL algorithm in-context. Zisman et al. (2023) propose a method for distilling the improvement operator from a demonstrator whose actions are initially noisy, with the noise level decreasing progressively throughout the data collection phase. This approach simplifies the data generation compared to AD as it does not require logging the training process of RL models and relaxes DPT's limitation because the demonstrator may be suboptimal. Kirsch et al. (2023) demonstrated that data augmentation, through random projection of observations and actions, enhances task distribution and generalization on new domains, boosting ICL capabilities. Raparthy et al. (2023) study the effect of model size and dataset properties on the success of in-context learning. Other research, acknowledging Transformers' limitations when it comes to long sequences, explores alternative models such as S4 (Gu et al., 2021). Lu et al. (2023) adapted S4 for RL, showing that it is capable of surpassing LSTM in performance and Transformers in runtime, indicating potential in RL ICL applications.

In contrast to previous works that tackle changing reward distributions or environment dynamics as changing goals, we expand the application of ICLRL to changing action spaces. Our use of random embeddings, motivated by the preparation of the model to unseen actions, may also improve ICL abilities by data augmentation, as suggested by Kirsch et al. (2023).

B.4. Discarding the Linear Layer

The Headless LLM approach, introduced by Godey et al. (2023), utilizes Contrastive Learning to eliminate the language head from the model architecture. Rather than generating a probability distribution over tokens, it directly predicts token embeddings. This strategy aims to enhance training and inference speeds by discarding a substantial linear layer. While in the RL context, where action spaces are typically small, this might not significantly impact runtime, it does offer a key advantage, as the model is no longer dependent on the number of actions and can therefore handle variable-length action spaces. Unlike Headless LLM, our approach does not rely on contrastive loss for learning action representations. Instead, we use fixed action embeddings, and the model is tasked to output the embedding of the next action. During inference, an action is chosen from a categorical distribution, where the logits are the similarities between the predicted embedding and available actions. Thus, the role of contrastive loss is to enhance the likelihood of selecting the correct action while

diminishing the probabilities of other actions.

Wolpertinger by Dulac-Arnold et al. (2015) also employs a similar approach of removing the linear head to improve the train and inference speeds. The authors suggest associating each action with an embedding and performing the training in a continuous action space. A specific action index is chosen as a nearest neighbor of the predicted embedding, and its effect is treated as environment dynamics. Most importantly, this nearest neighbor selection process can be optimized using an approximate algorithm. This leads to logarithmic time complexity, which is particularly advantageous in environments with large action spaces. Thus, we believe that Headless-AD is also capable of performing well on large action spaces, benefiting from runtime gains of approximate nearest neighbor lookup. However, Wolpertinger's usage of fixed action embeddings prevents introduction of new actions, highlighting the significance of Headless-AD's usage of randomized action embeddings.

C. Algorithm Distillation on Permuted Train Sets

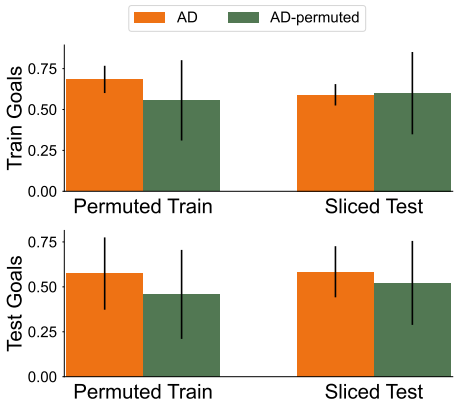

Figure 8. **AD Performance on Permuted Actions:** This figure shows the models' success rates on Darkroom environment, averaged over 5 seeds. We evaluated AD's ability to adapt to action sets with varied semantics by training it on a permuted dataset. Except for this, the training and testing are the same as in the vanilla setting and can be found in Section 4.3. Contrary to expectations, this tailored training did not enhance performance when compared to the AD model trained on standard datasets.

D. Across-Environment Generalization

We evaluated Headless-AD's ability to generalize skills learned in one domain to another. We trained Headless-AD on Contextual Bandits and subsequently assessed its performance on Bernoulli Bandits. To align with the Contextual Bandit setting, we introduced a random vector to serve as the state in the Bernoulli Bandit environment, effectively transforming it into a single-context Contextual Bandit scenario with Bernoulli-distributed rewards.

Figure 9 presents the results of these cross-domain experiments. Headless-AD not only adapted to the new domain but also maintained a satisfactory performance level, highlighting the model's capacity for cross-environment application.

E. Visual Darkroom

We adapted the Darkroom environment to produce high-dimensional visual observations to demonstrate Headless-AD's capability to generalize to new action spaces in more complex settings. In this modified version, the grid is visually represented, with the agent's position indicated in red (see Figure 10). This change introduces a more complex observation space. For the model to handle these high-dimensional visual observations, we integrated a convolutional network that converts the visual input into an embedding of size 'token_dim'.

Both Headless-AD and AD were configured with the identical hyperparameters previously applied in the Darkroom experiments. The results at Figure 11 show that Headless-AD maintains its superior performance in this more complex observational setting, whereas AD experiences a notable drop in performance. This discrepancy likely stems from our decision to retain the original hyperparameters without tuning them for the new environment. Based on our experience,

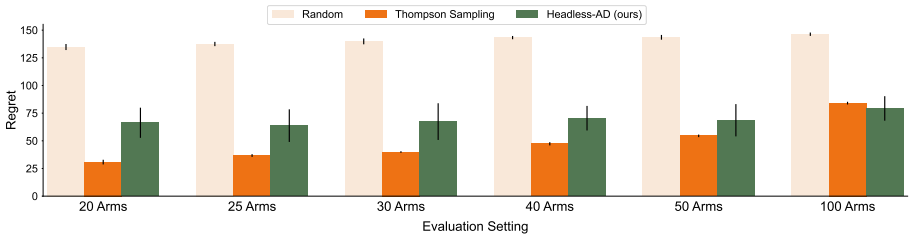

Figure 9. **Evaluation of Across-Environment Generalization:** This graph illustrates the performance of the Headless-AD model, which was initially trained on a Contextual Bandit setting and then evaluated on a Bernoulli Bandit environment. The experiment aimed to assess the model's generalization capabilities across novel environments. As shown, Headless-AD achieves a decent performance across various configurations with different numbers of arms, demonstrating its potential for across-environment usage.

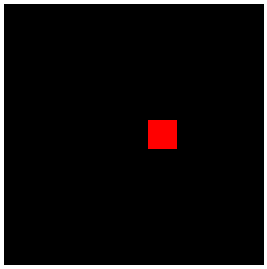

Figure 10. **Observation in the Visual Darkroom Environment.**

AD's performance is particularly sensitive to hyperparameter settings.

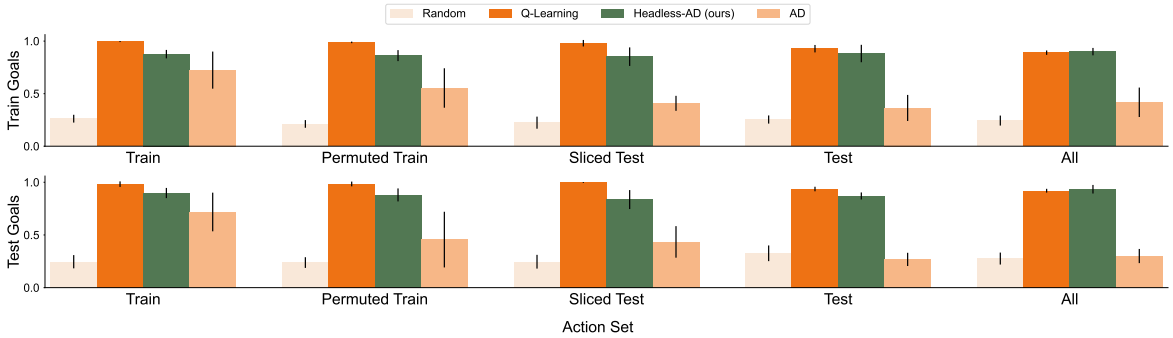

Figure 11. **Visual Darkroom:** Darkroom environment was modified to emit visual observations. Headless-AD consistently outperforms AD in this more complex observational setting.

F. Darkroom. Alternative Split

A random split of actions in Darkroom environment may lead to a data leakage as multiple action sequences result in the same endpoint. To address this, we conducted an experiment with distinct, non-overlapping action sets for training and testing. We trained Headless-AD with actions that cause movements of 0, 1, and 3 cells, and tested on movements of 2 cells, introducing scenarios not encountered during training.

Figure 12 shows that although Headless-AD experienced a slight decrease in performance on the test action set, it still remained significantly above the random level and outreaching the AD-level, underscoring its capability to generalize beyond the training action space. This experiment intends to make the capabilities and limitations of Headless-AD clearer.

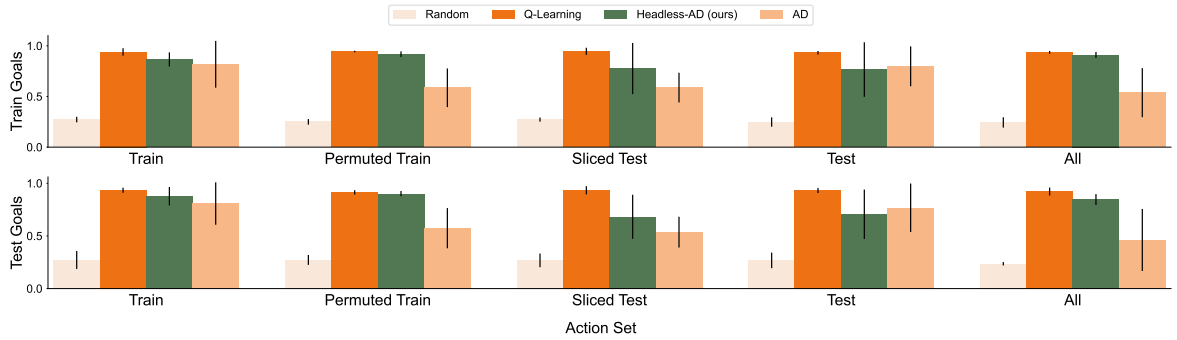

Figure 12. **Darkroom. Alternative Split:** In this experiment the actions in Darkroom were split into non-overlapping in terms of the distance of the endpoint from the initial position. Train split consisted of actions with length 0, 1 and 3, test split - 2. Headless-AD, though with a slight drop, maintained its performance on levels seen during the previous experiment with a random split.

G. MSE-Headless-AD In-Context Curves on Bernoulli Bandit

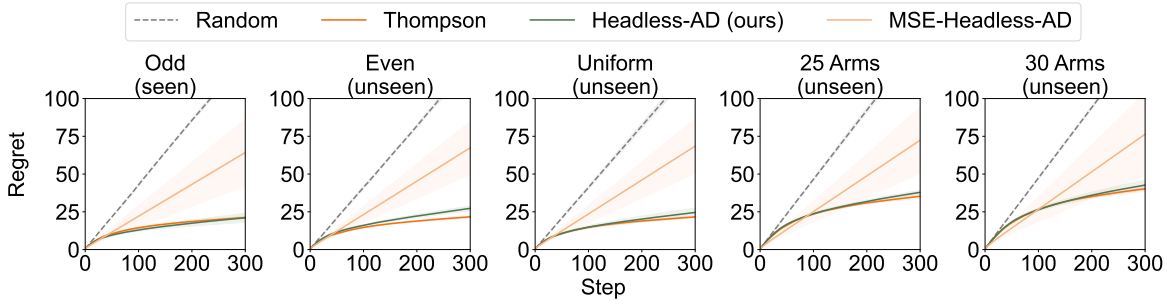

Figure 13. **MSE-Headless-AD In-Context Curves on Benroulli Bandit:** Though Table 1 shows that a variant of Headless-AD with MSE loss has a far from random regret, this graph shows that this result is due to the exploitation of several actions and the lack of exploration. We make this conclusion by observing that the shape of MSE-Headless-AD's training curve is a straight line, showing the evidence that no learning is being present. All curves are averaged over 5 seeds.

H. Sampling of Orthonormal Vectors

To sample the orthonormal vectors used as action embeddings, we use the `torch.nn.init.orthogonal` function from PyTorch (Paszke et al., 2019) that utilizes a specific algorithm from Saxe et al. (2013).

I. Algorithms' Training Times

Table 2. **Algorithm Training Times:** Here we list the training times (in hours) of Headless-AD and AD on each environment. All experiments were performed on A100 GPUs. Note that times here include both training and evaluation steps. Though Headless-AD requires more time for completion, we point out that it is evaluated on more environment sets compared to AD.

	Headless-AD	AD
Bernoulli Bandit	5.5	1.5
Contextual Bandit	10	7
Darkroom	27	19

J. Model Hyperparameters

In this section, we detail the hyperparameters for our models, each tuned for specific environments and design configurations. The tuning process utilized Bayesian optimization via the wandb sweep tool (Biewald, 2020). The optimization objective

was chosen to maximize both the performance score achieved by the model and the efficiency in the number of actions utilized. The objective function was structured as follows:

$$n_a/N + \text{final_normalized_return},$$

where n_a represents the total number of actions attempted by the model during evaluation, and N signifies the amount of possible actions within an environment. Moreover, we took the return from the final episode in evaluation and normalized it to the $[0, 1]$ range, with the lower bound 0 corresponding to the performance of a random agent, and the upper bound 1 denoting the efficiency of our data generation algorithm. Darkroom environment already has the returns in the range $[0, 1]$, so we did not perform normalization for this environment.

Table 3. **Headless-AD's Environment-Specific Hyperparameters:** For certain instances, hyperparameters underwent optimization within the specified ranges in the *Sweep Values* column, utilizing the Bayesian search method facilitated by the wandb sweep tool (Biewald, 2020). This process was employed to identify the optimal set of hyperparameters for enhanced performance and fair comparisons.

Hyperparameter	Bernoulli Bandit	Darkroom	Contextual Bandit	Sweep Values
Number of Layers	4	4	8	[1,2,4,6,8]
Number of Heads	64	64	8	[2,4,8,16,32,64
Model Dim.	512	512	1024	[128, 256, 512, 1024, 2048]
Sequence Length	300	100	300	
τ	1.34	7.8	3.0	[1e-5, 10]
Learning Rate	3.1e-04	1.7e-04	5.0e-05	[1e-5, 1e-2]
Weight Decay	4.4e-03	1.7e-02	5.0e-02	[1e-6, 5]
β_1	0.68	0.72	0.5	[0, 1]
Attention Dropout Rate	0.22	0.7	0.3	[0, 1]
Dropout Rate	0.12	0.19	0.25	[0, 1]
In-Context Episodes	300	1000	300	
Action Selection	sample	sample	mode	[sample, mode]

Table 4. **AD's Environment-Specific Hyperparameters:** For certain instances, hyperparameters underwent optimization within the specified ranges in the *Sweep Values* column, utilizing the Bayesian search method facilitated by the wandb sweep tool (Biewald, 2020). This process was employed to identify the optimal set of hyperparameters for enhanced performance and fair comparisons.

Hyperparameter	Bernoulli Bandit	Darkroom	Contextual Bandit	Sweep Values
Number of Layers	4	2	8	[1,2,4,6,8]
Number of Heads	64	8	8	[2,4,8,16,32,64
Model Dim.	512	512	1024	[128, 256, 512, 1024, 2048]
Sequence Length	300	100	300	
Label Smoothing	5.0e-03	5.0e-02	0.2	[0, 1]
Learning Rate	1.1e-05	4.0e-04	1.3e-05	[1e-5, 1e-2]
Weight Decay	1.5e-04	8.0e-06	0.25	[1e-6, 5]
β_1	0.82	0.95	0.58	[0, 1]
Attention Dropout Rate	0.1	3.0e-02	0.28	[0, 1]
Dropout Rate	0.1	1.0e-02	0.45	[0, 1]
In-Context Episodes	300	1000	300	
Action Selection	sample	mode	mode	[sample, mode]

Table 5. **Headless-AD's Environment-Specific Hyperparameters for Prompt Ablation:** For certain instances, hyperparameters underwent optimization within the specified ranges in the *Sweep Values* column, utilizing the Bayesian search method facilitated by the wandb sweep tool (Biewald, 2020). This process was employed to identify the optimal set of hyperparameters for enhanced performance and fair comparisons.

Hyperparameter	Bernoulli Bandit	Darkroom	Sweep Values
Number of Layers	4	4	[1,2,4,6,8]
Number of Heads	64	64	[2,4,8,16,32,64
Model Dim.	512	512	[128, 256, 512, 1024, 2048]
Sequence Length	300	100	
τ	0.92	8.5	[1e-5, 10]
Learning Rate	5.0e-04	3.5e-04	[1e-5, 1e-2]
Weight Decay	2.6e-05	4.7e-03	[1e-6, 5]
β_1	0.78	0.99	[0, 1]
Attention Dropout Rate	0.28	0.4	[0, 1]
Dropout Rate	0.16	0.16	[0, 1]
In-Context Episodes	300	1000	
Action Selection	sample	sample	[sample, mode]

Table 6. **Headless-AD's Environment-Specific Hyperparameters for Loss Ablation:** For certain instances, hyperparameters underwent optimization within the specified ranges in the *Sweep Values* column, utilizing the Bayesian search method facilitated by the wandb sweep tool (Biewald, 2020). This process was employed to identify the optimal set of hyperparameters for enhanced performance and fair comparisons.

Hyperparameter	Bernoulli Bandit	Darkroom	Sweep Values
Number of Layers	4	4	[1,2,4,6,8]
Number of Heads	64	64	[2,4,8,16,32,64
Model Dim.	512	512	[128, 256, 512, 1024, 2048]
Sequence Length	300	100	
Learning Rate	5.0e-05	1.3e-05	[1e-5, 1e-2]
Weight Decay	0.17	2.5	[1e-6, 5]
β_1	0.91	0.75	[0, 1]
Attention Dropout Rate	1.0e-03	8.0e-02	[0, 1]
Dropout Rate	0.14	0.75	[0, 1]
In-Context Episodes	300	1000	
Action Selection	mode	mode	

Table 7. **Headless-AD's Environment-Specific Hyperparameters for Action Embeddings Ablation:** For certain instances, hyperparameters underwent optimization within the specified ranges in the *Sweep Values* column, utilizing the Bayesian search method facilitated by the wandb sweep tool (Biewald, 2020). This process was employed to identify the optimal set of hyperparameters for enhanced performance and fair comparisons.

Hyperparameter	Bernoulli Bandit	Darkroom	Sweep Values
Number of Layers	4	4	[1,2,4,6,8]
Number of Heads	64	64	[2,4,8,16,32,64
Model Dim.	512	512	[128, 256, 512, 1024, 2048]
Sequence Length	300	100	
Learning Rate	1.1e-03	6.6e-04	[1e-5, 1e-2]
Weight Decay	1.2e-03	9.1e-04	[1e-6, 5]
β_1	0.94	0.6	[0, 1]
Attention Dropout Rate	0.62	0.53	[0, 1]
Dropout Rate	0.17	3.0e-02	[0, 1]
In-Context Episodes	300	1000	
Action Selection	sample	sample	

K. Linear Dependence of Different Types of Action Embeddings

We illustrate how the probability mass can unintentionally be put on unwanted actions through an analysis of cosine similarities between action embeddings, comparing orthonormal vectors to those from a standard normal distribution. Figure 14 demonstrates non-zero cosine similarities for the standard normal distribution, indicating that a vector perfectly aligned with one action embedding may erroneously attribute non-zero probabilities to other actions.

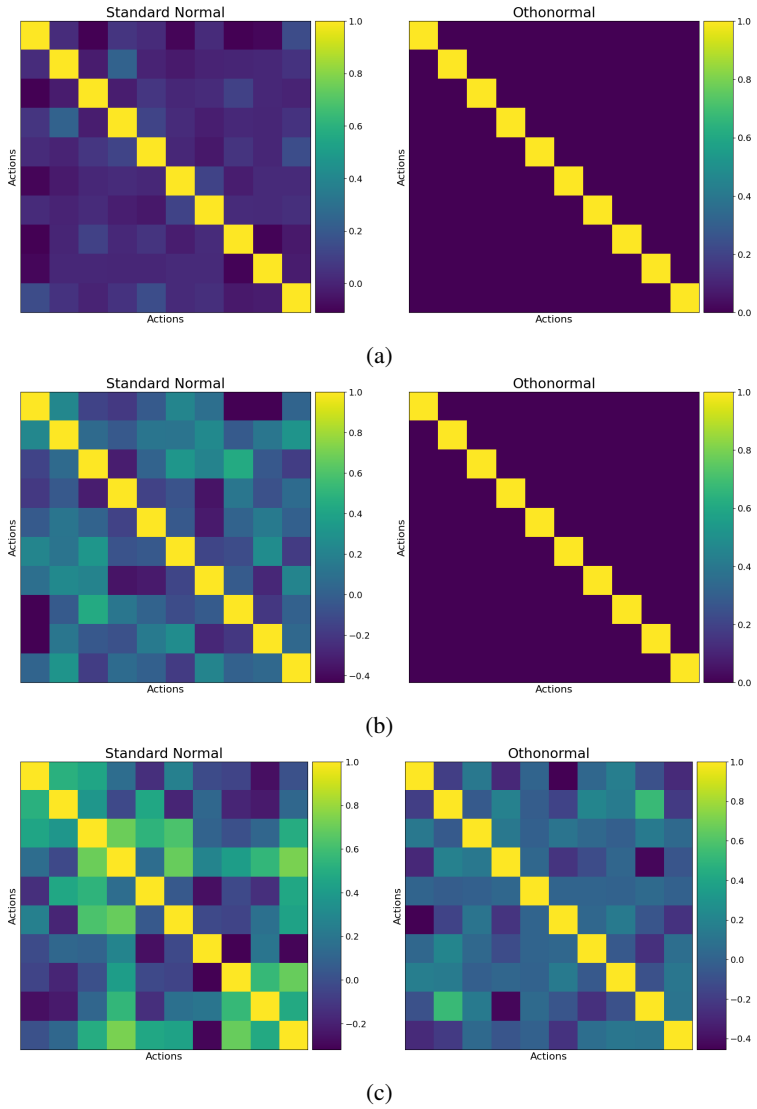

Figure 14. These plots illustrate the pairwise cosine similarities between action embeddings for orthonormal versus standard normal distributions. (a) In a 128-dimensional space with 10 actions, orthonormal embeddings exhibit perfect decorrelation, whereas standard normal embeddings display non-zero similarities among them. (b) With 10 actions in a 16-dimensional space, the similarity between embeddings increases as the dimensionality of the space decreases, indicating denser correlations. (c) For 10 actions in an 8-dimensional space, despite the action count exceeding the space's dimensionality, orthonormal vectors (referenced in Appendix: Orthonormal Vectors) maintain lower similarities compared to those from a standard normal distribution, emphasizing the robustness of orthonormal embeddings in constrained dimensions.

L. Code Sample

```python
from typing import Tuple

import torch
import torch.nn as nn


class ActionMapper(nn.Module):
    def __init__(
        self,
        action_embed_dim: int,
        num_actions: int,
        device: str,
    ):
        super().__init__()
        self.action_embed_dim = action_embed_dim
        self.num_actions = num_actions
        self.device = device

        self.action_map = torch.empty((num_actions, action_embed_dim), device=device)
        self.action_map.requires_grad = False

    @torch.no_grad()
    def regenerate(self) -> None:
        torch.nn.init.orthogonal_(self.action_map, gain=1)

    @torch.no_grad()
    def __call__(self, actions: torch.Tensor) -> torch.Tensor:
        embeds = self.action_map[actions]

        return embeds

    @torch.no_grad()
    def get_action(
        self,
        embeds: torch.Tensor,
    ) -> Tuple[torch.Tensor, torch.Tensor]:
        sims = embeds @ self.action_map.T

        dist = torch.distributions.Categorical(logits=sims)
        actions_sample, actions_argmax = dist.sample(), dist.probs.argmax(-1)

        return actions_sample, actions_argmax

    @torch.no_grad()
    def _get_action_map_as_context(self, batch_size: int):
        actions = torch.tile(self.action_map.unsqueeze(0), (batch_size, 1, 1))

        return actions


act_mapper = ActionMapper(...)

for states, actions, rewards in dataloader:
    act_mapper.regenerate()
    action_embeds = act_mapper(actions)

    actions_list = act_mapper._get_action_map_as_context(batch_size=states.shape[0])

    pred = model(
        states=states,
        act_emb=action_embeds,
        rewards=rewards,
        actions_list=actions_list,
    )

    act_sample, act_mode = act_mapper.get_action(pred)
```

Listing 1: Code that demonstrates the Headless-AD training procedure. Note that this snippet is intended for illustration purposes only. The complete code can be found in Headless-AD's repository.