Investigating Pre-Training Objectives for Generalization in Vision-Based Reinforcement Learning

Donghu Kim [*1] **Hojoon Lee** [*1] **Kyungmin Lee** [*1] **Dongyoon Hwang** [1] **Jaegul Choo** [1]

Abstract

Recently, various pre-training methods have been introduced in vision-based Reinforcement Learning (RL). However, their generalization ability remains unclear due to evaluations being limited to in-distribution environments and non-unified experimental setups. To address this, we introduce the Atari Pre-training Benchmark (Atari-PB), which pre-trains a ResNet-50 model on 10 million transitions from 50 Atari games and evaluates it across diverse environment distributions. Our experiments show that pre-training objectives focused on learning task-agnostic features (e.g., identifying objects and understanding temporal dynamics) enhance generalization across different environments. In contrast, objectives focused on learning task-specific knowledge (e.g., identifying agents and fitting reward functions) improve performance in environments similar to the pre-training dataset but not in varied ones. We publicize our codes, datasets, and model checkpoints at `https://github.com/dojeon-ai/Atari-PB`.

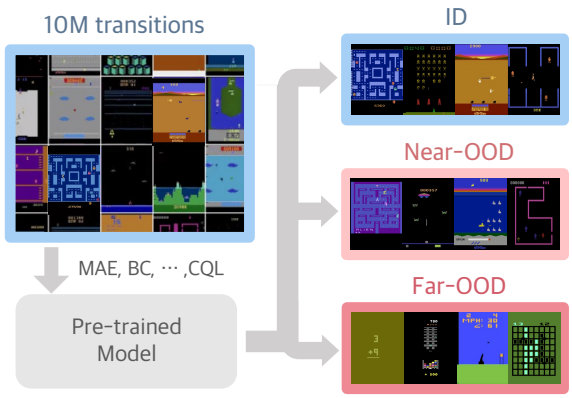


Figure 1: **Overview of Atari-PB.** The ResNet-50-based model is pre-trained from 10M interactions with a given pre-training algorithm. The pre-trained model is then evaluated by fine-tuning to In-Distribution (ID), Near-Out-of-Distribution (Near-OOD), and Far-Out-of-Distribution (Far-OOD) environments.

1. Introduction

The pretrain-then-finetune approach has become a standard practice in computer vision (CV) and natural language processing (NLP), renowned for its robust generalization across diverse tasks (He et al., 2022; Bubeck et al., 2023). In vision-based Reinforcement Learning (RL), this approach is now gaining attraction as well (Levine et al., 2020; Yang et al., 2023), driven by the increasing availability of large-scale offline datasets (Grauman et al., 2022; Padalkar et al., 2023).

In vision-based RL, various pre-training methods are designed to capture unique features from different data types. Image-based methods, for instance, focus on extracting spatial characteristics such as object sizes and shapes (Laskin et al., 2020b; Seo et al., 2023a). In contrast, video-based approaches delve into the temporal dynamics of environments like objects' movement and direction (Schwarzer et al., 2020b; Nair et al., 2022; Gupta et al., 2023). Those learning from demonstrations prioritize extracting task-relevant features, distinguishing agents from irrelevant elements like backgrounds (Pomerleau, 1991; Christiano et al., 2016; Islam et al., 2022). Finally, trajectory-based methods further concentrate on learning task-specific features by estimating the rewards associated with different states and actions (Kumar et al., 2020; Chen et al., 2021; Fujimoto et al., 2019).

Despite their advancements, the generalization capabilities of these methods remain underexplored as evaluations are typically confined to environments akin to their pre-training datasets (Schwarzer et al., 2021b; Lee et al., 2023). Several studies have probed the generalization ability of the agents by changing visual and task attributes, such as object colors (Hansen & Wang, 2021), shapes (Yuan et al., 2023), and physics (Taiga et al., 2022). Yet, these variations are relatively minor and may not sufficiently mirror

[*]Equal contribution [1]KAIST. Correspondence to: Donghu Kim <quagmire@kaist.ac.kr>.

Proceedings of the 41st International Conference on Machine Learning, Vienna, Austria. PMLR 235, 2024.

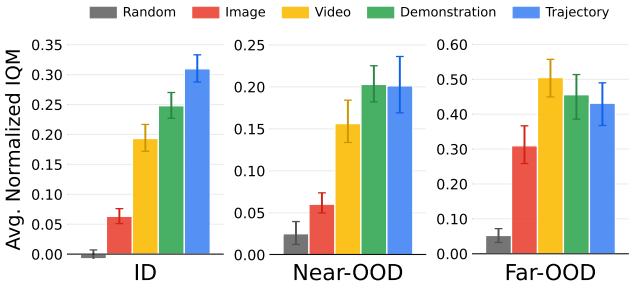


Figure 2: **Results Overview.** The pre-training methods are evaluated by their performance after fine-tuning to environments in three groups: In-Distribution, Near-Out-of-Distribution, and Far-Out-of-Distribution. Here, we report the results of fine-tuning via behavior cloning (i.e., replicating expert behavior) and average the scores of each algorithm category for a comprehensive analysis.

the complexities found in real-world scenarios. More recent research has begun to investigate the pre-trained models' generalization capabilities under larger distribution shifts (Nair et al., 2022; Xiao et al., 2022; Parisi et al., 2022; Majumdar et al., 2023a). However, these studies use models pre-trained on different data sources and with varying architectures, complicating the understanding of how learning objectives affect generalization performance.

To address this gap, we introduce the Atari Pre-training Benchmark (Atari-PB), which investigates the generalization capability of various learning objectives using a unified dataset and architecture. Our benchmark begins by pre-training a ResNet-50 encoder (He et al., 2016) on a dataset containing 10 million interactions from 50 different Atari environments (Agarwal et al., 2020). The pre-trained models are then fine-tuned across three groups of environments: In-Distribution (ID), Near-Out-of-Distribution (Near-OOD), and Far-Out-of-Distribution (Far-OOD). The ID group includes environments identical to those in the pre-training dataset. The Near-OOD group consists of similar tasks (e.g., shooting or tracking) but with different visual characteristics (e.g., object shape or speed). The Far-OOD group contains environments with entirely different tasks (e.g., solving math puzzles or color matching).

Our findings, illustrated in Figure 2, show that pre-training methods aimed at learning task-agnostic features, such as extracting spatial characteristics from images and temporal dynamics from videos, enhance generalization across various distribution shifts. In contrast, methods that learn task-specific knowledge, such as identifying agents from demonstrations and fitting reward functions from trajectories, enhance performance in the same environments but hinder generalization under distribution shifts.

Takeaways:

- Pre-training objectives that learn task-agnostic features, such as identifying objects from images and understanding temporal dynamics from videos, consistently improve generalization across various distribution shifts.
- Pre-training objectives that learn task-specific features, such as focusing on agents from demonstrations and fitting the reward function from trajectories, improve performance in similar tasks but lose effectiveness as task distribution shifts increase.

2. Related Work

2.1. Evaluating Generalization in Vision-Based RL

As Deep Reinforcement Learning has achieved notable success in specific games or tasks (Hafner et al., 2023; Schwarzer et al., 2023), research interest is moving towards the agent's ability to generalize on environments with visual and/or task distribution shifts.

For visual generalization, researchers have focused on developing environments with controllable visual elements. This includes changing the design of objects and walls (Lomonaco et al., 2020), adding background noise (Hansen & Wang, 2021; Stone et al., 2021), and applying realistic disturbances such as changes in lighting and camera perspectives (Dosovitskiy et al., 2017; Yuan et al., 2023).

Regarding task distribution shifts, numerous studies aimed to evaluate the agents on new tasks with similar dynamics but varying reward functions. For instance, Farebrother et al. (2018); Taiga et al. (2022) evaluated generalization abilities using different game modes and difficulty levels, while other works employed procedural generation (Cobbe et al., 2020; Zhu et al., 2020).

Despite the importance of these studies, they have primarily considered a single type of distribution shift, often minor in nature. Thus, we propose evaluating agent performance in environments with novel tasks and visual shifts (Far-OOD), those with similar tasks and visual shifts (Near-OOD), and those with same tasks (ID). This broader coverage of distribution shifts enables a comprehensive assessment of agents' generalization abilities under visual and task shifts.

2.2. Pre-training for Generalization in Vision-based RL

The pretrain-then-finetune approach is a well-established framework in CV and NLP for improving generalization ability. Similarly for visual RL, the development of large-scale offline datasets (Grauman et al., 2022; Padalkar et al., 2023) have spurred investigations into the efficacy of pre-training methods for enhancing generalization capabilities.

Studies by Parisi et al. (2022) and Hu et al. (2023) have shown that the use of pre-trained models, such as ResNet (He et al., 2016) and ViT (Dosovitskiy et al., 2020), enhances generalizability in various RL tasks. These models were pre-trained on ImageNet (Deng et al., 2009) and employed diverse training objectives, ranging from supervised (Radford et al., 2021) to self-supervised learning (He et al., 2020; Nair et al., 2022). Furthermore, Majumdar et al. (2023b) achieved improved generalization using a large, realistic dataset that closely resembles the ego-centric perspective of agents (Grauman et al., 2022), to pre-train a large ViT with masked image modeling (He et al., 2022).

Despite the advancements, these models were pre-trained with different datasets and architectures, making it challenging to identify the effectiveness of individual learning objectives for generalization. Our study addresses this gap by systematically evaluating the generalization capabilities of various pre-training objectives using a unified experimental setup, including an identical dataset, architecture, and downstream environments.

3. Preliminaries

3.1. Reinforcement Learning

In Reinforcement Learning (RL), we adopt a Partially Observable Markov Decision Process (POMDP) framework, which differs from standard MDP by limiting direct access to the true state s_t. Instead, agents receive a partially observable image o_t, generated by an emission function: $o_t \sim q(\cdot|s_t)$. At each timestep, t, an agent observes an observation o_t and chooses an action a_t according to its policy $a_t \sim \pi(\cdot|o_t)$. The agent then receives the next observation o_{t+1} and a reward r_t from the environment. Here, the goal of the agent is to learn a policy that maximizes the expected cumulative reward, $\mathbb{E}[\sum_{t=1}^{T} r_t]$.

3.2. Pre-training for Reinforcement Learning

Drawing insights from the CV and NLP field, the pretrain-then-finetune approach has emerged as a compelling method to improve generalization, paving the way for models to effectively adapt to diverse tasks and environments.

When pre-training for RL, the model learns from the transitions made by humans or agents, using different objectives such as Offline RL (Kumar et al., 2020) and self-supervised loss (Laskin et al., 2020b). These learning objectives are tightly related to the data type they leverage—image, video, demonstration, and trajectory—each introducing unique knowledge to the model. Based on this, we classify the pre-training algorithms into the following four categories. Here, N denotes the number of trajectories, T denotes the length of the trajectories, and NT denotes the total number of transitions in the dataset.

- **Image:** These methods learn the *spatial characteristics* of individual states from a set of images, $\bigcup_{i=1}^{NT} \{o_i\}$.
- **Video:** These methods learn the *temporal dynamics* of environments from a set of videos, $\bigcup_{i=1}^{N} \bigcup_{t=1}^{T} \{o_{i,t}\}$.
- **Demonstration:** These methods learn *task-relevant information* such as identifying agents and enemies from a set of observation-action pairs, $\bigcup_{i=1}^{N} \bigcup_{t=1}^{T} \{o_{i,t}, a_{i,t}\}$.
- **Trajectory:** These methods learn *richer task-relevant information* such as the value of states and actions from a set of observation-action-reward triplets, $\bigcup_{i=1}^{N} \bigcup_{t=1}^{T} \{o_{i,t}, a_{i,t}, r_{i,t}\}$.

4. Algorithms

This section outlines the pre-training algorithms we study, divided into four categories described in Section 3.2. Some algorithms were simplified to fit into our framework; we annotate such algorithms with a dagger(†) to avoid confusion. For further details, please refer to Appendix B.2.

4.1. No pre-training

Random: This approach involves fine-tuning a model that has been randomly initialized, with its encoder kept frozen throughout the fine-tuning process.

E2E: In contrast to the Random method, the End-to-End approach involves fine-tuning a randomly initialized model without any frozen components. In addition to ResNet-50, we also evaluate a 3-layer-CNN based model (Mnih et al., 2015), which is known to excel in many RL environments.

4.2. Learning from Image

CURL: Contrastive Unsupervised Reinforcement Learning (Laskin et al., 2020b) learns the spatial feature of images using augmentation functions and InfoNCE loss. It operates by ensuring that two augmented instances of the same image are encoded similarly in latent space.

MAE: Masked Autoencoder (He et al., 2022) learns the spatial structure of images by reconstructing masked images with transformer encoder-decoder architecture. Since we employ a convolutional architecture in this study, we adapt this approach by masking pixels in the convolutional feature map, inspired by Seo et al. (2023a); Xiao et al. (2021).

4.3. Learning from Video

ATC: Augmented Temporal Contrast (Stooke et al., 2021) learns the temporal dynamics of videos using InfoNCE loss.

This involves closely aligning an image with its future image in a latent space while maintaining distinctiveness from the other unrelated images.

SiamMAE: Siamese Masked Autoencoder (Gupta et al., 2023) is an extension of MAE to videos. This variant uses a Siamese architecture and an asymmetric masking strategy for temporal information extraction. In line with MAE, we apply masking to convolutional features.

R3M†: Reusable Representations for Robot Manipulation (Nair et al., 2022) uses multiple losses to learn from human demonstration videos with diverse tasks. In our study, we focus on the time contrastive loss of R3M. While akin to ATC, R3M gives an additional task of differentiating between 'near-future' and 'far-future' images.

4.4. Learning from Demonstration

BC: Behavioral Cloning (Pomerleau, 1991) learns to imitate the demonstrations by predicting actions from observations. This method acquires task-specific information from the dataset, as noted by Islam et al. (2022).

SPR: Self-Predictive Representations (Schwarzer et al., 2020b) learn the dynamics of environments by recursively predicting future observations in a latent space. We employ a recurrent neural network for future prediction and a momentum network for encoding target observations.

IDM: Inverse Dynamics Modeling (Christiano et al., 2016) learns to predict the action between consecutive observations. Similar to BC, IDM focuses on learning task-relevant information as it seeks to understand the cause-and-effect dynamics within the environments (Islam et al., 2022).

SPR+IDM: Inspired by Schwarzer et al. (2021b), we investigate whether the combination of SPR and IDM can further improve the agent's performance.

4.5. Learning from Trajectory

CQL: Conservative Q-Learning (Kumar et al., 2020) integrates temporal difference loss with conservative Q-learning loss. Its main objective is to accurately approximate the Bellman target while minimizing the overestimation of actions that are absent in the offline data. We consider two variants: CQL-M, using mean squared loss, and CQL-D, applying cross-entropy-based distributional backups (Bellemare et al., 2017a) for minimizing temporal differences.

DT: Decision Transformer (Chen et al., 2021) redefines RL as a sequence modeling problem. The network is trained to predict actions based on given states and desired cumulative rewards. During inference, the agent predicts the optimal actions needed to achieve a specified cumulative reward.

5. Atari Pre-training Benchmark

We introduce the Atari Pre-training Benchmark (Atari-PB), a benchmark designed to assess the generalization ability of pre-training methods in vision-based RL.

5.1. Dataset

The dataset for Atari-PB is derived from the DQN-Replay-Dataset (Agarwal et al., 2020). This dataset encompasses training logs from a DQN agent's experiences across 60 Atari games (Bellemare et al., 2013; Machado et al., 2018), documented over five distinct runs. Each run is divided into 50 evenly spaced checkpoints, ranging from the initial state (checkpoint 1) to the final state (checkpoint 50). This segmentation allows for precise control over the agent's performance level, thereby influencing the quality of the pre-training data.

To construct a pre-training dataset that reflects real-world complexities as a mixture of suboptimal and optimal behaviors, we selected the first 10 checkpoints from two separate runs. From each checkpoint, we sampled the first 10,000 transitions. Consequently, this process generated 200,000 transitions per game, culminating in a comprehensive dataset of 10 million transitions across 50 games. For more details, please refer to Appendix B.1.

5.2. Model

The network architecture of our model is composed of three main components: a backbone for encoding images into features, a neck for converting the features to a low-dimensional latent vector, and a head for mapping the latent vector into policy outputs.

Backbone, $f(\cdot)$: Our backbone utilizes a modified version of the ResNet-50 architecture (He et al., 2016). It is designed to process input images $\mathbf{o} \in \mathbb{R}^{C \times H \times W}$ and encode them into spatial feature maps $\mathbf{z} = f(\mathbf{o})$, where $\mathbf{z} \in \mathbb{R}^{D_z \times H_z \times W_z}$. Here, D_z represents the output dimension, while H_z and W_z are the dimensions of the feature map's height and width, respectively. We replaced the batch normalization (Ioffe & Szegedy, 2015) with group normalization (Wu & He, 2018), aiming to address discrepancies in data distributions between pre-training and fine-tuning phases, as discussed in Kumar et al. (2022a;b).

Neck, $g(\cdot)$: The neck module incorporates a game-specific spatial pooling strategy to manage the variability of in-game elements (Kumar et al., 2022a;b). It is followed by a 2-layer Multi-Layer Perceptron (MLP) which transforms the spatial features into a low-dimensional latent vector $\mathbf{q} = g(\mathbf{z})$, with $\mathbf{q} \in \mathbb{R}^{D_q}$ and D_q denoting the latent dimension.

Head, $h(\cdot)$: The head module employs a game-specific linear layer, allowing diverse policy outputs across different

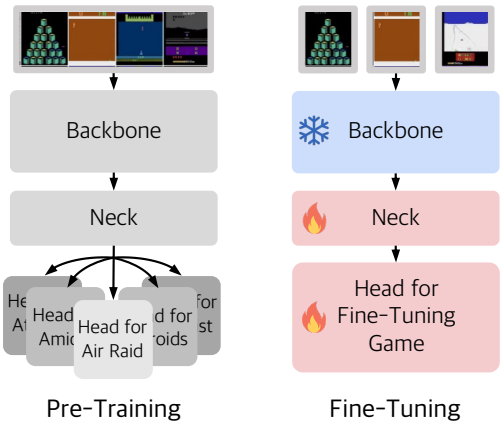


Figure 3: **Experimental Setup.** The model is pre-trained with 50 Atari games in a multi-headed fashion (left), then fine-tuned for each game individually (right). The snowflake symbol indicates freezing the weights, whereas the fire symbol represents re-initializing and fine-tuning the component.

games. This results in the final output $\mathbf{y} = h(\mathbf{q})$, where $\mathbf{y} \in \mathbb{R}^{|A|}$ and A denotes the action space.

5.3. Pre-Training

Following standard training protocols in Atari games, we pre-processed each image by down-sampling to 84×84 with grey-scaling, then stacked 4 consecutive frames (Mnih et al., 2015; Hessel et al., 2018). We applied two image augmentations: a random shift followed by intensity jittering (Schwarzer et al., 2020a; 2021a). Each model was pre-trained for 100 epochs using an AdamW optimizer (Loshchilov & Hutter, 2017) with a batch size of 512. We experimented with various learning rates, selecting from the range of $\{1e-3, 3e-4, ..., 3e-5, 1e-6\}$, and adjusted the weight decay within the range of $\{1e-4, 1e-5, 1e-6\}$.

5.4. Fine-tuning

For fine-tuning, we assessed the effectiveness of our pre-trained models in three different categories of environments:

- **In-Distribution (ID)**: This category includes the same 50 games used during the pre-training phase. This assessment aims to evaluate model performance in familiar settings. These games include the main task genres of Atari games, such as maze-based, tracking, vertical shooting, and horizontal shooting games.
- **Near-Out-of-Distribution (Near-OOD)**: This group consists of 10 games not used in pre-training but belonging to the same task genres as the ID games. While these games present tasks similar to those in ID, they assess the model's generalization ability on new visual elements and reward structures.
- **Far-Out-of-Distribution (Far-OOD)**: This group extends beyond Near-OOD, featuring 5 games with entirely novel task mechanics. For instance, HumanCannonball introduces projectile motion against gravity, while Klax focuses on color matching and stacking. These games serve as the baselines to understand the pre-trained models' generalization ability in entirely unfamiliar environments and tasks.

For these environments, we consider two common adaptation scenarios:

- **Offline BC**: In this scenario, the model undergoes fine-tuning through behavior cloning using 50,000 frames from expert demonstrations. For ID and Near-OOD games, expert demonstrations were sourced from the final checkpoint of the DQN-Replay-Dataset. For Far-OOD games, we used demonstrations from a Rainbow agent (Hessel et al., 2018) trained for 2 million steps. During this process, as depicted in Figure 3, we kept the backbone parameters frozen, while the neck and head components were re-initialized and then fine-tuned for 100 epochs.
- **Online RL**: In this scenario, the model is fine-tuned using the Rainbow algorithm (Hessel et al., 2018), with 50,000 interactions in each respective environments. Identical to the Offline BC approach, the backbone of the pre-trained model remains unaltered, while the neck and head are re-initialized and subsequently fine-tuned.

To ensure a reliable evaluation, we report the normalized game scores using the Inter Quantile Mean (IQM) methodology (Agarwal et al., 2021). This method involves stratified bootstrap sampling and is executed with three different random seeds. For normalizing the scores in ID and Near-OOD environments, we use the DQN scores as documented by Castro et al. (2018). In the case of Far-OOD scenarios, the normalization is based on the final scores of our Rainbow agent, which was trained for 2 million steps to provide the expert dataset for the Offline BC scenario.

6. Experimental Results

In this section, we present our main experimental results, as illustrated in Figure 4 and 5. Instead of merely providing a ranking of different pre-training methods, we focus on discerning specific trends and patterns that manifest across different downstream distributions and adaptation scenarios.

In summary, a distinctive pattern emerged between algorithms that learn task-agnostic information (using images or videos), and those that learn task-relevent information (using demonstrations or trajectories). While task-agnostic knowledge consistently improved performance across all

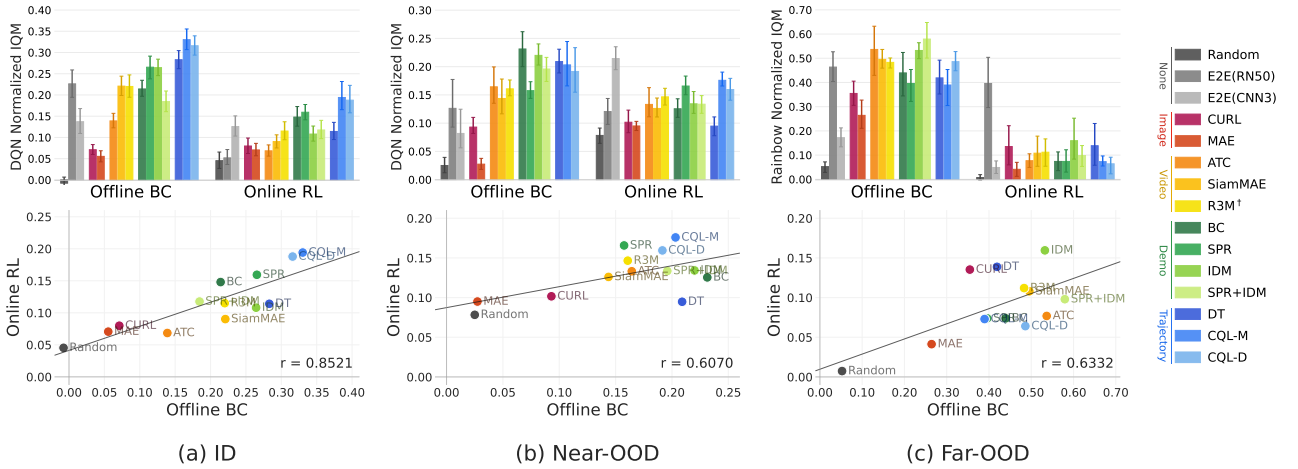


Figure 4: **Main Results.** Performance of each pre-training method after fine-tuning, in different distributions (ID, Near-OOD, Far-OOD) and adaptation scenarios (Offline BC, Online RL). We report the Inter Quantile Mean (IQM) of normalized scores across three seeds, along with a 95% confidence interval. The bars are grouped and color-coded by their categories described in Section 3.2 for ease of view.

distributions, task-specific knowledge showed limited robustness against task distribution shifts. We further elaborate on our findings through several key observations.

O1: Learning task-agnostic information from images and videos significantly enhance performance across ID, Near-OOD, and Far-OOD environments.

Our findings, as depicted in Figure 4, reveal that image-based pre-training methods like CURL and MAE consistently surpassed the Random baseline (i.e., a frozen backbone with randomly initialized weights). This highlights the importance of extracting spatial information (e.g., object size, shape, location) for an effective adaptation to ID environments and generalization to OOD environments.

Moreover, video-based pre-training methods (ATC, SiamMAE, R3M†) showed enhanced performance over image-based approaches, underscoring the significance of capturing both spatial and temporal information (e.g., objects' moving orientation and speed). Notably, these video-pretrained models slightly surpassed end-to-end (E2E) fine-tuning from scratch, despite having their backbone parameters frozen during fine-tuning. We believe that allowing end-to-end fine-tuning of video-based models will further enhance their performance, widening the performance gap compared to the E2E approach.

These findings support the emerging trend of incorporating images and videos in pre-training to achieve better generalization across various control tasks (Majumdar et al., 2023b; Bhateja et al., 2023).

O2: Learning task-relevant information from demonstrations further enhances ID and Near-OOD performance, but provides marginal improvements in Far-OOD performance.

Incorporating actions into pre-training objectives (BC, SPR, IDM, and SPR+IDM) led to mixed performance improvements. While task-specific knowledge benefited performance in ID and Near-OOD settings, a decline was noted in Far-OOD environments. This variation likely arose from the task similarities between ID and Near-OOD environments, in contrast to the distinct differences in tasks between ID and Far-OOD. Given that ID and Near-OOD games typically fall into the four main genres, it's plausible that the task knowledge from ID environments can be applied to Near-OOD environments, but not Far-OOD.

For example, as illustrated in the first column of Figure 5, the model pre-trained on SpaceInvaders (ID) learns task-specific knowledge such as agent and enemy locations as well as bullet movement direction, which are characteristic of the vertical shooting genre. This knowledge helps the model to quickly identify similar gameplay elements in Assault (Near-OOD), despite not having been exposed to the game during pre-training. Conversely, task-specific knowledge is rendered ineffective in Surround (Far-OOD), with its unique task of avoiding trails and distinct agent locations. This difference shows the difficulty in applying pre-learned task knowledge to environments with considerably different mechanisms.

O3: Learning reward-specific information from trajectories yields the best ID performance, while it shows limited generalization gains in Near-OOD and Far-OOD environments.

Integrating reward-specific information from trajectories (DT, CQL-M, CQL-D) led to superior success in ID environments. Nonetheless, their effectiveness diminished in

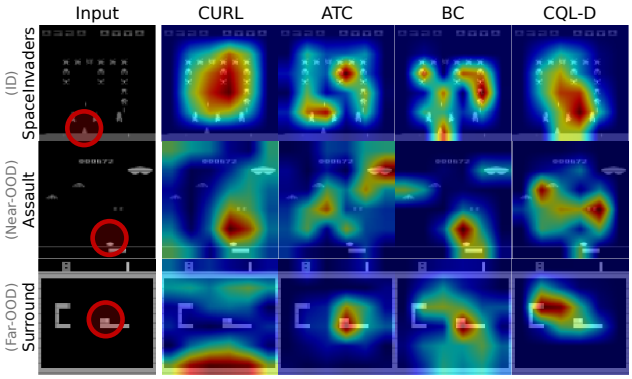


Figure 5: **Qualitative analysis of methods.** EigenCAM visualization of the pre-trained backbones in 3 games: SpaceInvaders (ID), Assault (Near-OOD), and Surround (Far-OOD). The agents are marked in red circles for each game (first column). We chose one representative method for each algorithm class.

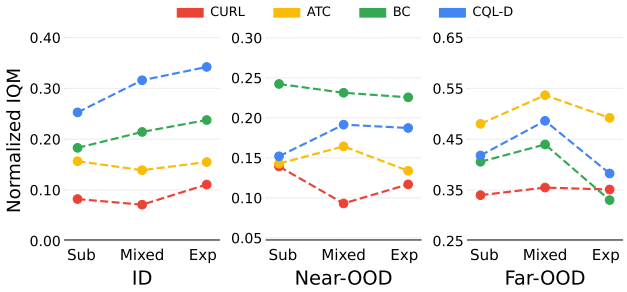


Figure 6: **Effect of data optimality.** Offline BC performance of algorithms after pre-training with datasets of differing optimality. We control the dataset optimality via the proficiency of the policy that created it, which in turn can be controlled by choosing different checkpoints of the DQN-Replay-Dataset (Agarwal et al., 2020).

Near-OOD and Far-OOD environments, where they lagged behind demonstration-based and video-based approaches. This shows that extracting reward-specific knowledge (e.g., rewarded for shooting enemies) enhances performance in familiar settings, but fails to generalize across environments with different reward functions (e.g., rewarded for shooting enemies but penalized for missing) or tasks (e.g., receiving rewards when surrounding the enemy), indicating their limitations for broader generalization.

To further validate our observations, we conducted a qualitative analysis of the pre-trained models with Eigen-CAM (Muhammad & Yeasin, 2020), which are illustrated in Figure 5. In ID games like SpaceInvaders, we found that unlike video-based methods (ATC), the demonstration or trajectory-based methods (BC, CQL-D) concentrated on capturing the agent, a critical object of the task. Interestingly, an identical pattern was observed in Near-OOD environments like Assault, fortifying our conjecture that the task-specific knowledge acquired during pre-training can be transferred to visually distinct environments. For an extended discussion and analysis, refer to Appendix C.3.

O4: Effective adaptation in one scenario correlates to effective adaptation in the other.

Figure 4 shows a strong correlation between performances in offline behavioral cloning and online reinforcement learning, with Pearson correlation coefficients of 0.85 in ID, 0.61 in Near-OOD, and 0.63 in Far-OOD environments.

This suggests that a well-pre-trained model can yield versatile representations applicable to a wide range of scenarios, not limited to specific adaptation algorithms (Parisi et al., 2022; Majumdar et al., 2023a).

O5: Miscellaneous Remarks

Inspired by Taiga et al. (2022), we have conducted an additional experiment to fine-tune our pre-trained models on ID environments with different game modes (Machado et al., 2018). Detailed protocol and results can be found in Appendix C.1. As shown in Figure 9, the results were similar to that of ID in our main experiments. We believe that simply changing the game mode did not sufficiently alter the reward functions and thus led to similar tendencies.

Additionally, we noticed that in end-to-end fine-tuning for Online RL, the smaller 3-layer CNN agent (CNN3) often outperforms the larger ResNet-50 agent (RN50). This finding is consistent with prior research, which suggests that deep and large models are prone to overfitting and are more vulnerable to the non-stationary nature of Online RL (Lee et al., 2024b). Many studies have proposed techniques to mitigate this issue (Nikishin et al., 2022; Lee et al., 2024a; Farebrother et al., 2024), suggesting that applying such methods could make larger architectures more competitive.

Finally, we have found an interesting relationship between object size and the performance of mask-based algorithms (MAE, SiamMAE). Compared to other pre-training methods like CURL and ATC, these methods excel in environments with large objects but underperform in games with tiny objects. For more details, please refer to Appendix C.2.

7. Ablation Studies

In our ablation studies, we assess the effects of variations in data optimality, size, and model size on pre-training methods. We selected CURL (image), ATC (video), BC (demonstration), and CQL-D (trajectory) as representative methods for comparison. Unless otherwise noted, the experimental setup remains identical to our main experiments.

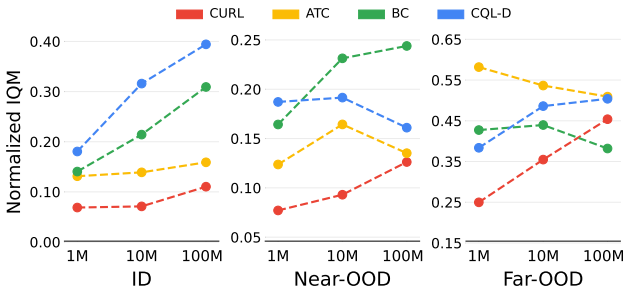


Figure 7: **Effect of Dataset Size.** We measure the Offline BC performance after pre-training with datasets of varying sizes. All datasets were derived from the same DQN-Replay-Dataset runs and checkpoints, with different numbers of transitions sampled based on dataset size.

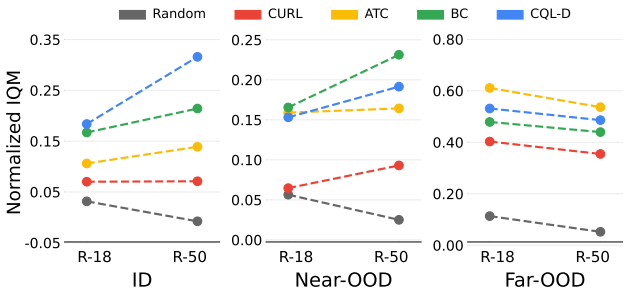


Figure 8: **Effect of model size.** Offline BC performance of algorithms utilizing different-sized models. We scaled the depth of models by the number of layers, and the width by the number of channels. Here, we also provide the Random performance of the modified models as a baseline.

7.1. Data Optimality

A key factor in building pre-training datasets for RL is data optimality. We investigated the impact of data optimality using three distinct datasets from the DQN-Replay-Dataset:

- **Suboptimal**: Data from the first and second checkpoint of each game, each with 50,000 interactions.
- **Mixed**: Data from the first ten checkpoints per game, each with 10,000 interactions (same as our main setup).
- **Expert**: Data from the ninth and tenth checkpoint of each game, each with 50,000 interactions.

O6: Using optimal data enhances ID adaptation but falters generalization in Near-OOD and Far-OOD environments.

Figure 6 shows that using optimal data does not guarantee improved performance in downstream environments. While moving from a Mixed dataset to an Expert dataset enhances performance in ID environments, its effectiveness was limited in Near-OOD and Far-OOD environments.

Optimal gameplay in Atari games usually follows repetitive patterns, resulting in limited diversity in Expert dataset. Such uniformity likely hindered the pre-trained model's ability to generalize to unfamiliar objects, by making it overly tailored to the objects encountered during pre-training. On the other hand, models pre-trained on datasets that blend optimal and suboptimal transitions (Mixed) showed enhanced generalization capabilities, particularly in Far-OOD environments. This improvement underscores the significance of dataset diversity for achieving effective generalization, a principle supported by recent research (Taiga et al., 2022).

7.2. Data Size

The scalability of pre-training methods can be significantly influenced by the size of the pre-training dataset. We explored this by using datasets of varying sizes: 1M, 10M, and 100M transitions, each derived from the same runs and checkpoints but adjusted to their respective sizes.

- **1M**: Consists of 1,000 initial transitions, sampled from the 10 initial checkpoints of each run.
- **10M**: Our standard setup, with 10,000 initial transitions from each checkpoint.
- **100M**: The largest dataset, containing 100,000 initial transitions from each checkpoint.

Models were pre-trained over 100 epochs for 1M and 10M datasets. However, due to the substantial size of the 100M dataset, pre-training was limited to 10 epochs.

O7: Larger dataset enhances ID adaptation but shows mixed effects on Near-OOD and Far-OOD generalization.

Figure 7 shows that larger datasets improved performance in ID settings, especially for methods involving demonstrations or trajectories. However, the benefits were less predictable in Near-OOD and Far-OOD environments, possibly because the model is relatively too small to fully encode all the information within large datasets in 10 epochs.

Consistent with our earlier findings in *O1*, *O2*, and *O3*, demonstration-based methods excelled in Near-OOD settings, while video-based methods were more effective in Far-OOD scenarios. This reiterates our findings that learning task-agnostic features improves generalization across distributions, but the benefits of learning task-relevant features diminish as task shifts increase.

7.3. Model Size

In the subsequent analysis, we explore how pre-training methods scale with changes in model size:

- **R-18**: A scaled-down model with reduced depth (18 layers) and halved channel widths, compared to R-50.
- **R-50**: A standard model architecture used in our study.

O8: Larger model provides consistent benefits in ID and Near-OOD, while its improvement is unclear in Far-OOD.

When comparing ResNet-18 to ResNet-50, we found that larger models improve results in ID and Near-OOD scenarios. While broader computer vision research suggests larger models usually offer better generalization to out-of-distribution (OOD) environments, this trend was not clearly observed in our Far-OOD experiments.

Nevertheless, the experimental results aligned with our prior observations (*O1*, *O2*, and *O3*), indicating that demonstration-based methods perform better in Near-OOD environments, and video-based methods excel in Far-OOD environments, irrespective to their model size.

8. Conclusion and Future Work

In this study, we examined how different pre-training objectives affect agents' generalization capabilities in vision-based reinforcement learning (RL). Our findings indicate that learning task-agnostic features, such as spatial features (e.g., object locations and shapes) and temporal features (e.g., speed and direction of moving objects), enhances generalization across various visual and task distribution shifts. In contrast, learning task-specific features, through actions (e.g., locations of agents and related objects) or reward structures (e.g., identifying beneficial or detrimental actions), improves performance in similar tasks but offers limited benefits in divergent task distributions.

Our results align with prior theoretical works demonstrating the benefits of learning temporal structures in environments for acquiring optimal feature representations (Bellemare et al., 2019; Lan et al., 2022). In this work, we provide empirical evidences supporting the notion that learning temporal structures within environments can improve generalization across various types of task shifts.

The RL community has recently introduced diverse datasets covering a wide range of human and robotic behaviors (Padalkar et al., 2023; Grauman et al., 2022). Our results suggest that pre-training with both task-agnostic and task-specific knowledge offers distinct benefits: one enhances generalization to different shifts, while the other excels in similar environments. Therefore, a promising future direction is to develop architectures or learning objectives that can decouple task-agnostic and task-specific features, allowing their use based on specific purposes (Wang et al., 2022; Bhateja et al., 2023). We hope our investigation provides valuable insights for advancing vision-based RL.

Acknowledgements

This work was supported by Institute for Information & communications Technology Promotion(IITP) grant funded by the Korea government(MSIT) (No.RS-2019-II190075 Artificial Intelligence Graduate School Program(KAIST)), the National Research Foundation of Korea (NRF) grant funded by the Korea government (MSIT) (No. NRF-2022R1A2B5B02001913).

Impact Statement

This paper investigates the relationship between pre-training objectives and their generalization capabilities across different downstream environments. Our findings provide valuable insights into how various learning objectives influence model generalization, aiding the development of large, foundational vision-based agents.

We also acknowledge the potential risks associated with rapid advancements in these foundational models, especially in robotic control. Discussions for preventing misuse and maintaining high ethical standards are essential to ensure these technologies benefit society and do not cause harm.

References

Agarwal, R., Schuurmans, D., and Norouzi, M. An optimistic perspective on offline reinforcement learning. *Proc. the International Conference on Machine Learning (ICML)*, 2020.

Agarwal, R., Schwarzer, M., Castro, P. S., Courville, A. C., and Bellemare, M. Deep reinforcement learning at the edge of the statistical precipice. *Proc. the Advances in Neural Information Processing Systems (NeurIPS)*, 34: 29304–29320, 2021.

Arora, S., Du, S., Kakade, S., Luo, Y., and Saunshi, N. Provable representation learning for imitation learning via bi-level optimization. In *International Conference on Machine Learning*, 2020.

Baker, B., Akkaya, I., Zhokhov, P., Huizinga, J., Tang, J., Ecoffet, A., Houghton, B., Sampedro, R., and Clune, J. Video pretraining (vpt): Learning to act by watching unlabeled online videos. *Proc. the Advances in Neural Information Processing Systems (NeurIPS)*, 2022.

Bellemare, M., Dabney, W., Dadashi, R., Ali Taiga, A., Castro, P. S., Le Roux, N., Schuurmans, D., Lattimore, T., and Lyle, C. A geometric perspective on optimal representations for reinforcement learning. *Advances in neural information processing systems*, 32, 2019.

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

Bellemare, M. G., Dabney, W., and Munos, R. A distributional perspective on reinforcement learning. In *Proc. the International Conference on Machine Learning (ICML)*, pp. 449–458. PMLR, 2017a.

Bellemare, M. G., Dabney, W., and Munos, R. A distributional perspective on reinforcement learning. 2017b.

Bhateja, C., Guo, D., Ghosh, D., Singh, A., Tomar, M., Vuong, Q., Chebotar, Y., Levine, S., and Kumar, A. Robotic offline rl from internet videos via value-function pre-training. *Proc. the Advances in Neural Information Processing Systems (NeurIPS)*, 2023.

Brandfonbrener, D., Nachum, O., and Bruna, J. Inverse dynamics pretraining learns good representations for multitask imitation. *Proc. the Advances in Neural Information Processing Systems (NeurIPS)*, 2023.

Bubeck, S., Chandrasekaran, V., Eldan, R., Gehrke, J., Horvitz, E., Kamar, E., Lee, P., Lee, Y. T., Li, Y., Lundberg, S., et al. Sparks of artificial general intelligence: Early experiments with gpt-4. *arXiv preprint arXiv:2303.12712*, 2023.

Caluwaerts, K., Iscen, A., Kew, J. C., Yu, W., Zhang, T., Freeman, D., Lee, K.-H., Lee, L., Saliceti, S., Zhuang, V., et al. Barkour: Benchmarking animal-level agility with quadruped robots. *arXiv preprint arXiv:2305.14654*, 2023.

Castro, P. S., Moitra, S., Gelada, C., Kumar, S., and Bellemare, M. G. Dopamine: A Research Framework for Deep Reinforcement Learning. 2018. URL http://arxiv.org/abs/1812.06110.

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. *Proc. the Advances in Neural Information Processing Systems (NeurIPS)*, 2021.

Chen, T., Kornblith, S., Norouzi, M., and Hinton, G. A simple framework for contrastive learning of visual representations. In *Proc. the International Conference on Machine Learning (ICML)*, pp. 1597–1607. PMLR, 2020a.

Chen, X. and He, K. Exploring simple siamese representation learning. In *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 15750–15758, 2021.

Chen, X., Fan, H., Girshick, R., and He, K. Improved baselines with momentum contrastive learning. *arXiv preprint arXiv:2003.04297*, 2020b.

Christiano, P., Shah, Z., Mordatch, I., Schneider, J., Blackwell, T., Tobin, J., Abbeel, P., and Zaremba, W. Transfer from simulation to real world through learning deep inverse dynamics model. *arXiv preprint arXiv:1610.03518*, 2016.

Cobbe, K., Hesse, C., Hilton, J., and Schulman, J. Leveraging procedural generation to benchmark reinforcement learning. In *Proc. the International Conference on Machine Learning (ICML)*, pp. 2048–2056. PMLR, 2020.

Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., and Fei-Fei, L. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pp. 248–255. Ieee, 2009.

Dosovitskiy, A., Ros, G., Codevilla, F., Lopez, A., and Koltun, V. Carla: An open urban driving simulator. In *Conference on robot learning*, pp. 1–16. PMLR, 2017.

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.

Farebrother, J., Machado, M. C., and Bowling, M. Generalization and regularization in dqn. *arXiv preprint arXiv:1810.00123*, 2018.

Farebrother, J., Orbay, J., Vuong, Q., Taïga, A. A., Chebotar, Y., Xiao, T., Irpan, A., Levine, S., Castro, P. S., Faust, A., et al. Stop regressing: Training value functions via classification for scalable deep rl. *arXiv preprint arXiv:2403.03950*, 2024.

Feichtenhofer, C., Li, Y., He, K., et al. Masked autoencoders as spatiotemporal learners. *Advances in neural information processing systems*, 35:35946–35958, 2022.

Fujimoto, S., Meger, D., and Precup, D. Off-policy deep reinforcement learning without exploration. In *Proc. the International Conference on Machine Learning (ICML)*, 2019.

Grauman, K., Westbury, A., Byrne, E., Chavis, Z., Furnari, A., Girdhar, R., Hamburger, J., Jiang, H., Liu, M., Liu, X., et al. Ego4d: Around the world in 3,000 hours of egocentric video. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 18995–19012, 2022.

Grill, J.-B., Strub, F., Altché, F., Tallec, C., Richemond, P., Buchatskaya, E., Doersch, C., Avila Pires, B., Guo, Z., Gheshlaghi Azar, M., et al. Bootstrap your own latent-a new approach to self-supervised learning. *Proc. the Advances in Neural Information Processing Systems (NeurIPS)*, 2020.

Gupta, A., Wu, J., Deng, J., and Fei-Fei, L. Siamese masked autoencoders. *arXiv preprint arXiv:2305.14344*, 2023.

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

Hafner, D., Lillicrap, T., Fischer, I., Villegas, R., Ha, D., Lee, H., and Davidson, J. Learning latent dynamics for planning from pixels. In *Proc. the International Conference on Machine Learning (ICML)*, 2019b.

Hafner, D., Pasukonis, J., Ba, J., and Lillicrap, T. Mastering diverse domains through world models. *arXiv preprint arXiv:2301.04104*, 2023.

Han, T., Xie, W., and Zisserman, A. Video representation learning by dense predictive coding. In *Proc. of the IEEE International Conference on Computer Cision (ICCV)*, pp. 0–0, 2019.

Hansen, N. and Wang, X. Generalization in reinforcement learning by soft data augmentation. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 13611–13617. IEEE, 2021.

He, K., Zhang, X., Ren, S., and Sun, J. Deep residual learning for image recognition. In *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

He, K., Fan, H., Wu, Y., Xie, S., and Girshick, R. Momentum contrast for unsupervised visual representation learning. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 9729–9738, 2020.

He, K., Chen, X., Xie, S., Li, Y., Dollár, P., and Girshick, R. Masked autoencoders are scalable vision learners. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 16000–16009, 2022.

Hessel, M., Modayil, J., Van Hasselt, H., Schaul, T., Ostrovski, G., Dabney, W., Horgan, D., Piot, B., Azar, M., and Silver, D. Rainbow: Combining improvements in deep reinforcement learning. In *Proc. the AAAI Conference on Artificial Intelligence (AAAI)*, 2018.

Hu, Y., Wang, R., Li, L. E., and Gao, Y. For pre-trained vision models in motor control, not all policy learning methods are created equal. *arXiv preprint arXiv:2304.04591*, 2023.

Ioffe, S. and Szegedy, C. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *Proc. the International Conference on Machine Learning (ICML)*, 2015.

Islam, R., Tomar, M., Lamb, A., Efroni, Y., Zang, H., Didolkar, A., Misra, D., Li, X., van Seijen, H., Combes, R. T. d., et al. Agent-controller representations: Principled offline rl with rich exogenous information. *Proc. the International Conference on Machine Learning (ICML)*, 2022.

Kaiser, Ł., Babaeizadeh, M., Miłos, P., Osiński, B., Campbell, R. H., Czechowski, K., Erhan, D., Finn, C., Kozakowski, P., Levine, S., et al. Model based reinforcement learning for atari. In *Proc. the International Conference on Learning Representations (ICLR)*, 2019.

Kumar, A., Zhou, A., Tucker, G., and Levine, S. Conservative q-learning for offline reinforcement learning. *Proc. the Advances in Neural Information Processing Systems (NeurIPS)*, 2020.

Kumar, A., Agarwal, R., Geng, X., Tucker, G., and Levine, S. Offline q-learning on diverse multi-task data both scales and generalizes. *Proc. the International Conference on Learning Representations (ICLR)*, 2022a.

Kumar, A., Singh, A., Ebert, F., Yang, Y., Finn, C., and Levine, S. Pre-training for robots: Offline rl enables learning new tasks from a handful of trials. arxiv e-prints, art. *arXiv preprint arXiv:2210.05178*, 2022b.

Lan, C. L., Tu, S., Oberman, A., Agarwal, R., and Bellemare, M. G. On the generalization of representations in reinforcement learning. *arXiv preprint arXiv:2203.00543*, 2022.

Laskin, M., Lee, K., Stooke, A., Pinto, L., Abbeel, P., and Srinivas, A. Reinforcement learning with augmented data. *Proc. the Advances in Neural Information Processing Systems (NeurIPS)*, 2020a.

Laskin, M., Srinivas, A., and Abbeel, P. Curl: Contrastive unsupervised representations for reinforcement learning. In *Proc. the International Conference on Machine Learning (ICML)*, 2020b.

Lee, H., Lee, K., Hwang, D., Lee, H., Lee, B., and Choo, J. On the importance of feature decorrelation for unsupervised representation learning in reinforcement learning. *arXiv preprint arXiv:2306.05637*, 2023.

Lee, H., Cho, H., Kim, H., Gwak, D., Kim, J., Choo, J., Yun, S.-Y., and Yun, C. Plastic: Improving input and label plasticity for sample efficient reinforcement learning. *Advances in Neural Information Processing Systems*, 36, 2024a.

Lee, H., Cho, H., Kim, H., Kim, D., Min, D., Choo, J., and Lyle, C. Slow and steady wins the race maintaining plasticity with hare and tortoise networks. 2024b.

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. *Proc. the Advances in Neural Information Processing Systems (NeurIPS)*, 2022a.

Lee, K.-H., Nachum, O., Zhang, T., Guadarrama, S., Tan, J., and Yu, W. Pi-ars: Accelerating evolution-learned visual-locomotion with predictive information representations. In *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2022b.

Levine, S., Kumar, A., Tucker, G., and Fu, J. Offline reinforcement learning: Tutorial, review, and perspectives on open problems. *arXiv preprint arXiv:2005.01643*, 2020.

Lomonaco, V., Desai, K., Culurciello, E., and Maltoni, D. Continual reinforcement learning in 3d non-stationary environments. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, pp. 248–249, 2020.

Loshchilov, I. and Hutter, F. Decoupled weight decay regularization. *Proc. the International Conference on Learning Representations (ICLR)*, 2017.

Ma, Y. J., Sodhani, S., Jayaraman, D., Bastani, O., Kumar, V., and Zhang, A. Vip: Towards universal visual reward and representation via value-implicit pre-training. 2022.

Machado, M. C., Bellemare, M. G., Talvitie, E., Veness, J., Hausknecht, M., and Bowling, M. Revisiting the arcade learning environment: Evaluation protocols and open problems for general agents. *Journal of Artificial Intelligence Research*, 2018.

Majumdar, A., Yadav, K., Arnaud, S., Ma, Y. J., Chen, C., Silwal, S., Jain, A., Berges, V.-P., Abbeel, P., Malik, J., et al. Where are we in the search for an artificial visual cortex for embodied intelligence? 2023a.

Majumdar, A., Yadav, K., Arnaud, S., Ma, Y. J., Chen, C., Silwal, S., Jain, A., Berges, V.-P., Abbeel, P., Malik, J., et al. Where are we in the search for an artificial visual cortex for embodied intelligence? *arXiv preprint arXiv:2303.18240*, 2023b.

Mendonca, R., Bahl, S., and Pathak, D. Structured world models from human videos. *arXiv preprint arXiv:2308.10901*, 2023.

Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., Graves, A., Riedmiller, M., Fidjeland, A. K., Ostrovski, G., et al. Human-level control through deep reinforcement learning. *Nature*, 2015.

Muhammad, M. B. and Yeasin, M. Eigen-cam: Class activation map using principal components. In *2020 International Joint Conference on Neural Networks (IJCNN)*. IEEE, 2020.

Nair, S., Rajeswaran, A., Kumar, V., Finn, C., and Gupta, A. R3m: A universal visual representation for robot manipulation. *6th Annual Conference on Robot Learning*, 2022.

Nakamoto, M., Zhai, Y., Singh, A., Mark, M. S., Ma, Y., Finn, C., Kumar, A., and Levine, S. Cal-ql: Calibrated offline rl pre-training for efficient online fine-tuning. *Proc. the Advances in Neural Information Processing Systems (NeurIPS)*, 2023.

Nikishin, E., Schwarzer, M., D'Oro, P., Bacon, P.-L., and Courville, A. The primacy bias in deep reinforcement learning. pp. 16828–16847, 2022.

Padalkar, A., Pooley, A., Jain, A., Bewley, A., Herzog, A., Irpan, A., Khazatsky, A., Rai, A., Singh, A., Brohan, A., et al. Open x-embodiment: Robotic learning datasets and rt-x models. *arXiv preprint arXiv:2310.08864*, 2023.

Parisi, S., Rajeswaran, A., Purushwalkam, S., and Gupta, A. The unsurprising effectiveness of pre-trained vision models for control. In *International Conference on Machine Learning*, pp. 17359–17371. PMLR, 2022.

Pomerleau, D. A. Efficient training of artificial neural networks for autonomous navigation. *Neural computation*, 3(1):88–97, 1991.

Radford, A., Kim, J. W., Hallacy, C., Ramesh, A., Goh, G., Agarwal, S., Sastry, G., Askell, A., Mishkin, P., Clark, J., et al. Learning transferable visual models from natural language supervision. In *International conference on machine learning*, pp. 8748–8763. PMLR, 2021.

Savva, M., Kadian, A., Maksymets, O., Zhao, Y., Wijmans, E., Jain, B., Straub, J., Liu, J., Koltun, V., Malik, J., et al. Habitat: A platform for embodied ai research. In *Proceedings of the IEEE/CVF international conference on computer vision*, pp. 9339–9347, 2019.

Schwarzer, M., Anand, A., Goel, R., Hjelm, R. D., Courville, A., and Bachman, P. Data-efficient reinforcement learning with self-predictive representations. *Proc. the International Conference on Learning Representations (ICLR)*, 2020a.

Schwarzer, M., Anand, A., Goel, R., Hjelm, R. D., Courville, A., and Bachman, P. Data-efficient reinforcement learning with self-predictive representations. In *Proc. the International Conference on Learning Representations (ICLR)*, 2020b.

Schwarzer, M., Rajkumar, N., Noukhovitch, M., Anand, A., Charlin, L., Hjelm, R. D., Bachman, P., and Courville, A. C. Pretraining representations for data-efficient reinforcement learning. *Proc. the Advances in Neural Information Processing Systems (NeurIPS)*, 2021a.

Schwarzer, M., Rajkumar, N., Noukhovitch, M., Anand, A., Charlin, L., Hjelm, R. D., Bachman, P., and Courville, A. C. Pretraining representations for data-efficient reinforcement learning. *Proc. the Advances in Neural Information Processing Systems (NeurIPS)*, 2021b.

Schwarzer, M., Ceron, J. S. O., Courville, A., Bellemare, M. G., Agarwal, R., and Castro, P. S. Bigger, better, faster: Human-level atari with human-level efficiency. In *International Conference on Machine Learning*, pp. 30365–30380. PMLR, 2023.

Seo, Y., Lee, K., James, S. L., and Abbeel, P. Reinforcement learning with action-free pre-training from videos. In *International Conference on Machine Learning*, pp. 19561–19579. PMLR, 2022.

Seo, Y., Hafner, D., Liu, H., Liu, F., James, S., Lee, K., and Abbeel, P. Masked world models for visual control. In *6th Annual Conference on Robot Learning*, pp. 1332–1344. PMLR, 2023a.

Seo, Y., Kim, J., James, S., Lee, K., Shin, J., and Abbeel, P. Multi-view masked world models for visual robotic manipulation. *Proc. the International Conference on Machine Learning (ICML)*, 2023b.

Sermanet, P., Lynch, C., Chebotar, Y., Hsu, J., Jang, E., Schaal, S., Levine, S., and Brain, G. Time-contrastive networks: Self-supervised learning from video. In *2018 IEEE international conference on robotics and automation (ICRA)*, pp. 1134–1141. IEEE, 2018.

Shah, R., Martín-Martín, R., and Zhu, Y. Mutex: Learning unified policies from multimodal task specifications. *arXiv preprint arXiv:2309.14320*, 2023.

Stone, A., Ramirez, O., Konolige, K., and Jonschkowski, R. The distracting control suite–a challenging benchmark for reinforcement learning from pixels. *arXiv preprint arXiv:2101.02722*, 2021.

Stooke, A., Lee, K., Abbeel, P., and Laskin, M. Decoupling representation learning from reinforcement learning. In *Proc. the International Conference on Machine Learning (ICML)*, 2021.

Taiga, A. A., Agarwal, R., Farebrother, J., Courville, A., and Bellemare, M. G. Investigating multi-task pretraining and generalization in reinforcement learning. In *Proc. the International Conference on Learning Representations (ICLR)*, 2022.

Tassa, Y., Doron, Y., Muldal, A., Erez, T., Li, Y., Casas, D. d. L., Budden, D., Abdolmaleki, A., Merel, J., Lefrancq, A., et al. Deepmind control suite. *arXiv preprint arXiv:1801.00690*, 2018.

Tong, Z., Song, Y., Wang, J., and Wang, L. Videomae: Masked autoencoders are data-efficient learners for self-supervised video pre-training. *Advances in neural information processing systems*, 35:10078–10093, 2022.

Walke, H. R., Black, K., Zhao, T. Z., Vuong, Q., Zheng, C., Hansen-Estruch, P., He, A. W., Myers, V., Kim, M. J., Du, M., et al. Bridgedata v2: A dataset for robot learning at scale. In *Conference on Robot Learning*, pp. 1723–1736. PMLR, 2023.

Wang, C., Luo, X., Ross, K., and Li, D. Vrl3: A data-driven framework for visual deep reinforcement learning. *Advances in Neural Information Processing Systems*, 35: 32974–32988, 2022.

Wu, P., Majumdar, A., Stone, K., Lin, Y., Mordatch, I., Abbeel, P., and Rajeswaran, A. Masked trajectory models for prediction, representation, and control. *arXiv preprint arXiv:2305.02968*, 2023.

Wu, Y. and He, K. Group normalization. In *Proc. of the European Conference on Computer Vision (ECCV)*, pp. 3–19, 2018.

Xiao, T., Singh, M., Mintun, E., Darrell, T., Dollár, P., and Girshick, R. Early convolutions help transformers see better. *Advances in neural information processing systems*, 34:30392–30400, 2021.

Xiao, T., Radosavovic, I., Darrell, T., and Malik, J. Masked visual pre-training for motor control. *arXiv preprint arXiv:2203.06173*, 2022.

Yang, S., Nachum, O., Du, Y., Wei, J., Abbeel, P., and Schuurmans, D. Foundation models for decision making: Problems, methods, and opportunities. *arXiv preprint arXiv:2303.04129*, 2023.

Ye, W., Zhang, Y., Abbeel, P., and Gao, Y. Become a proficient player with limited data through watching pure videos. In *The Eleventh International Conference on Learning Representations*, 2022.

Yu, T., Lan, C., Zeng, W., Feng, M., Zhang, Z., and Chen, Z. Playvirtual: Augmenting cycle-consistent virtual trajectories for reinforcement learning. *Advances in Neural Information Processing Systems*, 34:5276–5289, 2021.

Yu, T., Zhang, Z., Lan, C., Chen, Z., and Lu, Y. Mask-based latent reconstruction for reinforcement learning. *Proc. the Advances in Neural Information Processing Systems (NeurIPS)*, 2022.

Yuan, Z., Yang, S., Hua, P., Chang, C., Hu, K., Wang, X., and Xu, H. Rl-vigen: A reinforcement learning benchmark for visual generalization. *Proc. the Advances in Neural Information Processing Systems (NeurIPS)*, 2023.

Zang, H., Li, X., Yu, J., Liu, C., Islam, R., Combes, R. T. D., and Laroche, R. Behavior prior representation learning for offline reinforcement learning. *arXiv preprint arXiv:2211.00863*, 2022.

Zhang, W., GX-Chen, A., Sobal, V., LeCun, Y., and Carion, N. Light-weight probing of unsupervised representations for reinforcement learning. *arXiv preprint arXiv:2208.12345*, 2022.

Zheng, R., Wang, X., Sun, Y., Ma, S., Zhao, J., Xu, H., Daumé III, H., and Huang, F. Taco: Temporal latent action-driven contrastive loss for visual reinforcement learning. *arXiv preprint arXiv:2306.13229*, 2023.

Zhu, Y., Wong, J., Mandlekar, A., Martín-Martín, R., Joshi, A., Nasiriany, S., and Zhu, Y. robosuite: A modular simulation framework and benchmark for robot learning. *arXiv preprint arXiv:2009.12293*, 2020.

A. Extended Related Work

The availability of large-scale offline datasets tailored for robotics (Walke et al., 2023; Shah et al., 2023; Mendonca et al., 2023) has significantly influenced reinforcement learning (RL) research. This influence is particularly notable in adopting a pre-trained visual encoder for various downstream tasks (Ye et al., 2022; Ma et al., 2022; Lee et al., 2023).

A crucial aspect in the selection of pre-trained models is the type of data employed during pre-training. These algorithms are typically categorized by the data they use: Images, Videos, Demonstrations, and Trajectories. Each category utilizes different objectives to extract distinct features during the pre-training phase.

Image: Image-based learning encompasses two main strategies. The first focuses on augmentation-invariant representations, achieved by ensuring consistency in the latent space for different augmentations of the same image (Chen et al., 2020a; Laskin et al., 2020b; Chen & He, 2021; Chen et al., 2020b; Grill et al., 2020). The second strategy involves reconstructing heavily masked images, leveraging transformer architectures (He et al., 2022; Seo et al., 2023a;b).

Video: Video-based algorithms extend image-based techniques by integrating temporality. Temporal contrastive learning, for example, aims to closely encode temporally adjacent images to understand temporal dynamics (Nair et al., 2022; Ma et al., 2022; Sermanet et al., 2018; Han et al., 2019; Stooke et al., 2021). Other approaches include reconstructing masked images from temporally adjacent frames (Yu et al., 2022; Gupta et al., 2023; Tong et al., 2022; Feichtenhofer et al., 2022), or autoregressively predicting future frames based on past observations (Hafner et al., 2019b; Seo et al., 2022).

Demonstration: Learning from demonstrations has a wide range of methods. Inverse dynamics learning focuses on predicting actions, given consecutive states (Christiano et al., 2016; Islam et al., 2022; Brandfonbrener et al., 2023). Forward dynamics learning targets predicting future states from current state-action pairs (Schwarzer et al., 2020b; Yu et al., 2021; Lee et al., 2023; Zheng et al., 2023). Imitation learning, on the other hand, aims to replicate the behavior policy demonstrated in the data (Pomerleau, 1991; Zang et al., 2022; Arora et al., 2020; Baker et al., 2022; Caluwaerts et al., 2023). Some approaches combine multiple methods for enhanced robustness Yu et al. (2022); Zhang et al. (2022).

Trajectory: Trajectory-based methods maximize the use of reward information available in trajectories. These can be broadly categorized into two settings: online, where data is collected through real-time interaction with the environment (Mnih et al., 2015; Bellemare et al., 2017a; Hafner et al., 2019a; Laskin et al., 2020a), and offline, which involves learning from pre-existing datasets (Fujimoto et al., 2019; Kumar et al., 2020; Lee et al., 2022a; Nakamoto et al., 2023; Wu et al., 2023; Lee et al., 2022b).

Recent studies have evaluated the effectiveness of pre-trained visual representations in vision-based RL, employing a variety of pre-training methods (Parisi et al., 2022; Majumdar et al., 2023b; Hu et al., 2023). These investigations have used different models, including ResNet (He et al., 2016) and ViT (Dosovitskiy et al., 2020), which were pre-trained on a wide array of datasets (Deng et al., 2009; Savva et al., 2019; Tassa et al., 2018; Grauman et al., 2022). The findings indicate that while visual representations pre-trained on large and diverse datasets can enhance generalization in downstream tasks, no single pre-trained model consistently excels in generalization across all types of tasks.

B. Implementation Details for Atari-PB

B.1. Pre-training

In this section, we describe our pre-training stage in detail. We first depict the three components of our pre-training model (backbone, neck, and head) and explain how our pre-training dataset was curated. Note that the model description is our most fundamental form, and that adjustments are often made to align with each method's requirements. For such algorithm-specific elements, refer to Section B.2.

Backbone, $f(\cdot)$: A backbone network is a game-agnostic spatial feature extractor. We employ a widely used and sufficiently large ResNet-50, but replace batch normalization with group normalization following Kumar et al. (2022a). Given an $(4, 84, 84)$ input $\mathbf{o}$, the backbone encodes them into a $(2048, 6, 6)$ feature map $\mathbf{z} = f(\mathbf{o})$.

Neck, $g(\cdot)$: The main goal of neck is to encode the feature map into a 512-dimensional vector $\mathbf{q} = g(\mathbf{z})$, while applying learnable spatial embedding. Given the backbone output, each feature map is point-wise multiplied with its game-specific spatial embedding. Spatial pooling and instance normalization is then applied to obtain a 2048-dimensional vector, which is further encoded by a neural network. Unless stated otherwise, we use a 2-layer MLP with ReLU activation, hidden dimension of 1024, and output dimension of 512.

Head, $h(\cdot)$: The head makes the prediction $\mathbf{y} = h(\mathbf{q})$, where $\mathbf{y}$'s dimensionality depends on the task. We employ a multi-head architecture, meaning that multiple neural networks of identical architecture are trained, each one devoted for each game. Unless specified, we use a single linear layer that outputs the action prediction $\mathbf{y} \in \mathbb{R}^A$.

Dataset: The DQN-Replay-Dataset (Agarwal et al., 2020), a collection of DQN agent's training logs in 60 Atari games, provides 50 million transitions for each game collected across five different runs. These runs are subdivided into 50 checkpoints, each containing 1 million transitions of differing optimality. To fulfill our desiderata of "reflecting the diverse nature of real-world datasets" while keeping it computationally accessible, we choose to compile small segments from multiple runs and checkpoints. Our data creation procedure is thus the following: from the 50 games of our choice, we choose the first 2 runs of each game and the first 10 checkpoints of each run. From each of the 1,000 checkpoints, the initial 10,000 interactions are sampled, resulting in a 10 million dataset. We found the initial 10 checkpoints to be sufficient for covering both suboptimal and expert policies; as shown by the supplementary figures of Agarwal et al. (2020), 40 million steps (end of 10th checkpoint) is enough for DQN agents to achieve reasonable score in all games.

Table 1: Games categorized by distribution.

Distribution	Games
In-Distribution	AirRaid, Amidar, Asteroids, Atlantis, BankHeist, BattleZone, Berzerk, Bowling, Boxing, Breakout, Carnival, Centipede, ChopperCommand, CrazyClimber, DemonAttack, DoubleDunk, ElevatorAction, Enduro, FishingDerby, Freeway, Frostbite, Gopher, Gravitar, Hero, IceHockey, Jamesbond, Kangaroo, Krull, KungFuMaster, MontezumaRevenge, MsPacman, NameThisGame, Phoenix, Pitfall, PrivateEye, Qbert, RoadRunner, Robotank, Skiing, Solaris, SpaceInvaders, StarGunner, Tennis, TimePilot, Tutankham, UpNDown, VideoPinball, WizardOfWor, YarsRevenge, Zaxxon
Near-Out-of-Distribution	Alien, Assault, Asterix, BeamRider, JourneyEscape, Pong, Pooyan, Riverraid, Seaquest, Venture
Far-Out-of-Distribution	BasicMath, HumanCannonball, Klax, Othello, Surround

B.2. Baseline Implementations

We provide a more detailed description of each method in the following sections. Table 2 shows the universal hyperparameters across all pre-training methods; any method-specific hyperparameters are individually listed in the following tables.

Table 2: Global pre-training hyperparameters.

Hyperparameter	Value
Observation rendering	(84,84), Grayscale
Frames stacked	4
Reward discount factor (γ)	1.0 (DT) 0.99 (Rest)
Action space size ($\|A\|$)	18
Augmentation	[Random Shift, Intensity]
Random shift pad	4
Intensity scale	0.05
Learning rate scheduler	Cosine annealing with warmup
Warmup ratio	0.1
Initial learning rate ratio	0.1

B.2.1. CURL

Contrastive Unsupervised Representations for Reinforcement Learning (Laskin et al., 2020b) learns augmentation invariant representations using InfoNCE loss and momentum encoder. Given two augmented versions (two views) of an image (denoted $\mathbf{o}_1, \mathbf{o}_2$), one is passed through an 'online' encoder to get $\mathbf{q} = g(f(\mathbf{o}_1))$ and the other is passed through a coupled 'momentum' encoder to get a target $\mathbf{q}_+ = g'(f'(\mathbf{o}_2))$. To predict the target, $\mathbf{q}$ is passed through a predictor network to get $\mathbf{y} = h(\mathbf{q})$. The InfoNCE loss is then computed based on $\mathbf{y}$ and $\mathbf{q}_+$:

$$\mathcal{L}_{\text{CURL}} = -\sum_{b \in B} \log \frac{\exp(\mathbf{y}^b \cdot \mathbf{q}_+^b)}{\exp(\mathbf{y}^b \cdot \mathbf{q}_+^b) + \sum_{b' \in B - \{b\}} \exp(\mathbf{y}^b \cdot \mathbf{q}_+^{b'})}$$

As implied in the notations, we use backbone and neck as the encoder and head as the predictor. The momentum networks f', g' are updated every iteration with a coefficient of $\tau = 0.99$ that scales linearly up to 0.999. As for the similarity measure, we use dot product instead of bilinear product (Chen et al., 2020b).

Table 3: Hyperparameters for pre-training CURL.

Hyperparameter	Value
Epochs (early stop)	100 (20)
Base learning rate	3e-6
Weight decay	1e-5
Optimizer (β_1, β_2)	AdamW (0.9, 0.999)
Batch size	512
Momentum update ratio (τ)	[0.9, 0.999]
Representation dimensions	512
Similarity measure	Dot product

B.2.2. MAE

Masked Autoencoder (He et al., 2022) learns to reconstruct heavily masked images with transformer encoder-decoder architecture. As we use a convolutional network for our backbone, we naturally turn from masking patches of images to masking pixels of convolutional features, inspired by Seo et al. (2023a). The transformer encoder-decoder module is added at the end of our neck, and the head is used to predict the target pixels. In the neck, we perform masking after applying the game-wise spatial embedding, and pass the 2048-dimensional features through a 2 layer MLP into a sequence of 512-dimensional tokens. These are then processed with a transformer encoder, appended with mask tokens for prediction, and further processed with a transformer decoder. Finally, the head's game-wise linear layer is used to predict the pixels

of the target image. Because the backbone reduces an $(84, 84)$ image into a $(6, 6)$ spatial embedding (or 36 tokens), the game-wise linear layer maps each token to a 784-dimensional vector to predict a $(4, 14, 14)$ sized patch of the target image.

We follow the hyperparameters provided by He et al. (2022) but use a higher mask ratio $\rho = 0.9$. We use a 3-layer transformer encoder and a 4-layer transformer decoder, following Seo et al. (2023a).

Table 4: Hyperparameters for pre-training MAE.

Hyperparameter	Value
Epochs	100
Base learning rate	3e-4
Weight decay	5e-2
Optimizer (β_1, β_2)	AdamW (0.9, 0.95)
Batch size	512
Mask ratio (ρ)	0.9
Transformer embedding dimensions	512
Transformer MLP ratio	4
Transformer heads	4
Transformer encoder layers	3
Transformer decoder layers	4
(Head) Linear output dimensions	768

B.2.3. ATC

Augmented Temporal Contrast (Stooke et al., 2021) learns temporally predictive representations by maximizing the similarity between the representations of current states and their paired future states, using InfoNCE loss and momentum encoder. This means that the input image $\mathbf{o}_t$ is encoded by an 'online' encoder into $\mathbf{q}_t = g(f(\mathbf{o}_t))$, and its near-future target image $\mathbf{o}_{t+k}$ is encoded by a 'momentum' encoder into $\mathbf{q}_{t+k} = g'(f'(\mathbf{o}_{t+k}))$. With the two representations in hand, the predictor predicts the target $\mathbf{q}_{t+k}$ via $\mathbf{y}_t = h(\mathbf{q}_t)$ to minimize the InfoNCE loss:

$$\mathcal{L}_{\text{ATC}} = -\sum_{b \in B} \log \frac{\exp(\mathbf{y}_t^b \cdot \mathbf{q}_{t+k}^b)}{\exp(\mathbf{y}_t^b \cdot \mathbf{q}_{t+k}^b) + \sum_{b' \in B - \{b\}} \exp(\mathbf{y}_t^b \cdot \mathbf{q}_{t+k}^{b'})}$$

Same as CURL, we use the backbone and neck as our encoder and the head as our predictor. The momentum update is performed every iteration with a coefficient of $\tau = 0.99$ that scales linearly up to 0.999. We also use dot product instead of bilinear product as a similarity measure.

Table 5: Hyperparameters for pre-training ATC.

Hyperparameter	Value
Epochs	100
Base learning rate	3e-4
Weight decay	1e-5
Optimizer (β_1, β_2)	AdamW (0.9, 0.999)
Batch size	512
Steps to future state (k)	3
Momentum update ratio (τ)	[0.9, 0.999]
Representation dimensions	512
Similarity measure	Dot product

B.2.4. SIAMMAE

Siamese Masked Autoencoder (Gupta et al., 2023) extends MAE (He et al., 2022) to a temporal prediction task using siamese architecture and asymmetric masking strategy. Concretely, both current image $\mathbf{o}_t$ and future(target) image $\mathbf{o}_{t+k}$ are passed through the same network up until the transformer decoder. In our implementation, this includes the backbone, game-wise spatial embedding, and the transformer encoder. In the process, $\mathbf{o}_{t+k}$ is masked with an extremely high ratio($\rho = 0.95$) while $\mathbf{o}_t$ remains untouched. In order to retrieve the lost information, the transformer decoder refers to the current frame's

information via cross attention. For image reconstruction, we use the same head as our MAE. Overall, we follow the training procedure stated by Gupta et al. (2023) with some modifications. First, for the same reason as MAE, we use convolution feature masking instead of patch masking. Second, we use the transformer architecture used to train our MAE. Third, we reduce the future sampling window from $[4, 48]$ to $[1, 3]$.

Table 6: Hyperparameters for pre-training SiamMAE.

Hyperparameter	Value
Epochs	100
Base learning rate	3e-4
Weight decay	5e-2
Optimizer (β_1, β_2)	AdamW (0.9, 0.95)
Batch size	512
Steps to future state (k)	[1,3]
Mask ratio (ρ)	0.95
Transformer embedding dimensions	512
Transformer MLP ratio	4
Transformer heads	4
Transformer encoder layers	3
Transformer decoder layers	4
(Head) Linear output dimensions	768

B.2.5. R3M†

Reusable Representations for Robot Manipulation (Nair et al., 2022) uses multiple losses to learn from human demonstration videos with diverse tasks, but we focus our study on the time contrastive loss. Although similar to InfoNCE loss in ATC (Stooke et al., 2021), the main difference is that in addition to the future(target) image $\mathbf{o}_{t+k}$, a further-future image $\mathbf{o}_{t+k'}(k < k')$ is sampled as a 'hard' negative. Unfortunately, we find pre-training with the original time contrastive loss to be challenging. In order to preserve the aforementioned idea, we implement R3M as 'ATC with an additional hard negative' and use the following loss function:

$$\mathcal{L}_{\text{R3M}} = -\sum_{b \in B} \log \frac{\exp(\mathbf{y}_t^b \cdot \mathbf{q}_{t+k}^b)}{\exp(\mathbf{y}_t^b \cdot \mathbf{q}_{t+k}^b) + \sum_{b' \in B - \{b\}} \exp(\mathbf{y}_t^b \cdot \mathbf{q}_{t+k}^{b'}) + \exp(\mathbf{y}_t^b \cdot \mathbf{q}_{t+k'}^b)}$$

Naturally, we use the same momentum networks as ATC; all future images are passed through momentum backbone and neck, which are updated every iteration.

Table 7: Hyperparameters for pre-training R3M†.

Hyperparameter	Value
Epochs	100
Base learning rate	3e-4
Weight decay	1e-5
Optimizer (β_1, β_2)	AdamW (0.9, 0.999)
Batch size	512
Steps to future states (k, k')	3, 6
Momentum update ratio (τ)	[0.9, 0.999]
Output representation dimensions	512
Similarity measure	Dot product

B.2.6. BC

Behavioral cloning learns to predict the action $\mathbf{a}_t$ from its observation $\mathbf{o}_t$. Predictions are made by a single pass through the model $\mathbf{y}_t = h(g(f(\mathbf{o}_t)))$, of which goal is to minimize the cross entropy loss between $\mathbf{y}_t$ and $\mathbf{a}_t$.

Table 8: Hyperparameters for pre-training BC.

Hyperparameter	Value
Epochs	100
Base learning rate	3e-4
Weight decay	1e-5
Optimizer (β_1, β_2)	AdamW (0.9, 0.999)
Batch size	512

B.2.7. SPR

Self Predictive Representations (Schwarzer et al., 2020b) learn to recursively predict future states from a starting state and subsequent actions. Given an image $\mathbf{o}_t$ and a sequence of actions $\mathbf{a}_{t:t+K-1}$, the model's goal is to predict the consequent future images $\mathbf{o}_{t+1:t+K}$ in the latent space. The current image is encoded using an 'online' network to get $\mathbf{q}_t = f(g(\mathbf{o}_t))$, which is used in tandem with the actions to predict future representations $\mathbf{y}_{t+1:t+K} = h(\mathbf{q}_t, \mathbf{a}_{t:t+K-1})$. The target representations are obtained by encoding the future images with a 'momentum' encoder: $\mathbf{q}_i = f'(g'(\mathbf{o}_i)), i \in [t+1, t+K]$. In our experiments, we make the following modifications. First, we use RNN instead of recursive CNN, which is put in front of the linear layer in the head network. A game-wise action embedding for the RNN input, as each game uses a different set of actions. Second, we use contrastive loss instead of cosine similarity loss. Third, we discard the Q-learning loss. As a result, we use the following loss:

$$\mathcal{L}_{\text{SPR}} = -\sum_{b \in B} \sum_{k=1}^{K} \log \frac{\exp(\mathbf{y}_{t+k}^b \cdot \mathbf{q}_{t+k}^b)}{\sum_{b' \in B} \sum_{k'=1}^{K} \exp(\mathbf{y}_{t+k}^b \cdot \mathbf{q}_{t+k'}^{b'})}$$

Table 9: Hyperparameters for pre-training SPR.

Hyperparameter	Value
Epochs	25
Base learning rate	3e-4
Weight decay	1e-4
Optimizer (β_1, β_2)	AdamW (0.9, 0.999)
Batch size	128
Prediction sequence length (K)	4
Momentum update ratio (τ)	[0.9, 0.999]
Representation dimensions	512
Similarity measure	Dot product

B.2.8. IDM

Inverse Dynamics Modeling (Christiano et al., 2016) learns to predict the action $\mathbf{a}_t$ taken between two successive observations $\mathbf{o}_t, \mathbf{o}_{t+1}$. Both images are passed through the same backbone and neck to obtain $\mathbf{q}_t = g(f(\mathbf{o}_t))$ and $\mathbf{q}_{t+1} = g(f(\mathbf{o}_{t+1}))$, which are concatenated to make an action prediction via $\mathbf{y}_t = h(\mathbf{q}_t, \mathbf{q}_{t+1})$. Same as BC, we use the cross-entropy loss between $\mathbf{y}_t$ and $\mathbf{a}_t$.

B.2.9. SPR+IDM

In combining the two algorithms, we do not make any modifications, and simply use the sum of two losses to pre-train the model.

Table 10: Hyperparameters for pre-training IDM.

Hyperparameter	Value
Epochs (early stop)	100 (30)
Base learning rate	3e-4
Weight decay	1e-5
Optimizer (β_1, β_2)	AdamW (0.9, 0.999)
Batch size	512

Table 11: Hyperparameters for pre-training SPR+IDM.

Hyperparameter	Value
Epochs	25
Base learning rate	3e-5
Weight decay	1e-5
Optimizer (β_1, β_2)	AdamW (0.9, 0.999)
Batch size	128
Prediction sequence length (K)	4
Momentum update ratio (τ)	[0.9, 0.999]
Representation dimensions	512
Similarity measure	Dot product

B.2.10. CQL

Conservative Q-Learning (Kumar et al., 2020) is developed for learning Q values in an offline setting. We implement two distinct Q learning methods: Mean Squared Error (MSE) learning and Cross-entropy (Distributional) learning, both based on CQL. We choose ResNet50 as the backbone architecture, with a neck structure consistent with that in Behavioral Cloning (BC) for both MSE and Distributional.

MSE: In MSE-based Q-learning, we employ a game-wise head that yields an output in the shape of $\mathbb{R}^{B\times T\times A}$, where B denotes the batch size, T represents the number of time steps, and A signifies the size of the action space. We balance the MSE loss and CQL loss using a coefficient of 0.1.

Table 12: Hyperparameters for pre-training CQL-M.

Hyperparameter	Value
Epochs	100
Base learning rate	1e-4
Weight decay	1e-5
Optimizer (β_1, β_2)	AdamW (0.9, 0.95)
Batch size	512
CQL coefficient	0.1

Distributional: For cross-entropy-based distributional Q-learning, we utilize a game-wise head designed to produce an output dimension of $\mathbb{R}^{B\times T\times A\times N_A}$, with B, T, A having the same implications as in MSE, and N_A representing the number of atoms. We set a coefficient of 0.1 to balance cross-entropy loss and CQL loss, consistent with Kumar et al. (2022a). Additionally, we adopted a support set of [-10, 10] following the methodology in Bellemare et al. (2017b).

B.2.11. DT

Decision Transformer (DT) (Chen et al., 2021) adopts a unique perspective by treating reinforcement learning as a sequence modeling problem. It represents each transition within a trajectory as a triplet consisting of total return $\hat{R}_t$, observation o_t, and action a_t. DT is trained to autoregressively predict these sequences, given the transition history $\bigcup_{i=1}^{t-1}\{\hat{R}_i, o_i, a_i\}$. At inference, this knowledge is leveraged to predict the optimal actions necessary for achieving a desired cumulative reward.

In our implementation, observations are encoded by the backbone and a 2-layer MLP with game-specific spatial embedding, generating 512-dimensional tokens. Actions and returns are separately embedded into 512-dimensional tokens, resulting in

Table 13: Hyperparameters for pre-training CQL-D.

Hyperparameter	Value
Epochs	100
Base learning rate	1e-4
Weight decay	1e-5
Optimizer (β_1, β_2)	AdamW (0.9, 0.95)
Batch size	512
Support Set	[-10, 10]
CQL coefficient	0.1

a sequence of length $K \times 3$. This sequence is processed through a causal transformer to produce K outputs. The outputs are then utilized by the head layer to predict actions $a_{1:K}$. We employ cross-entropy loss for training the model.

Table 14: Hyperparameters for pre-training DT.

Hyperparameter	Value
Epochs	12
Base learning rate	1e-4
Weight decay	5e-2
Optimizer (β_1, β_2)	AdamW (0.9, 0.95)
Batch size	64
Sequence length (in tokens)	8×3
Reward scale	0.01
Transformer embedding dimensions	512
Transformer MLP ratio	4
Transformer heads	4
Transformer layers	4

B.3. Fine-tuning

After pre-training, we assess the pre-trained models on two downstream adaptation scenarios: Offline Behavioral Cloning (BC) and Online Reinforcement Learning (RL). Each task is conducted on three distinct sets of games (ID, Near-OOD, Far-OOD), including two sets unseen during pre-training. Results are compiled from three independent runs (seeds).

The backbone network is kept frozen to focus on the quality of representations, while other components are re-initialized. Any modifications revert to the standard architecture as detailed in Section 5.2 and Section B.1. This process is uniformly applied across all games.

B.3.1. Offline BC

In Offline BC, expert demonstrations are used for learning control. Specifically, the fine-tuning dataset for ID and Near-OOD games are sampled from DQN-Replay-Dataset. From the five runs, we use the last checkpoints and sample the initial 10,000 interactions. For Far-OOD games, we train a Rainbow agent on each environment for 2M steps and record the last 50,000 interactions. As a result, each fine-tuning stage is performed over a 50k dataset for 100 epochs. The final performance is evaluated by the average gameplay score of 100 trials.

Table 15: Hyperparameters for downstream Offline BC.

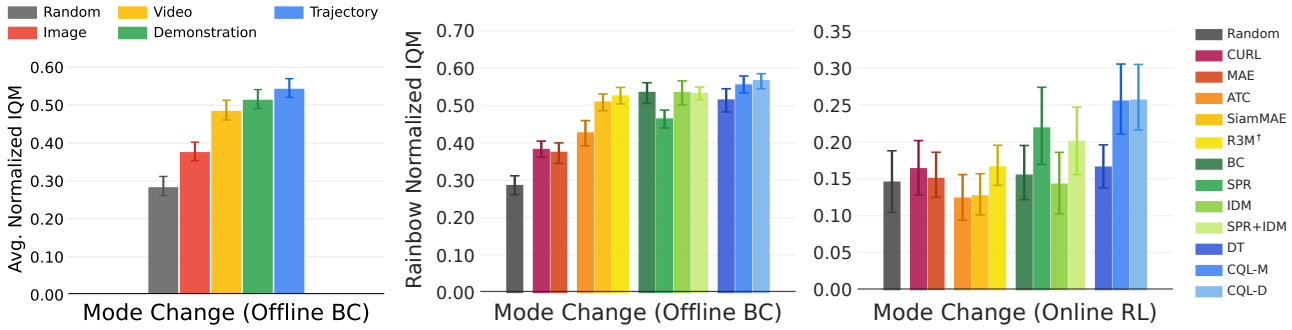
Hyperparameter	Value
Augmentation	[Random Shift, Intensity]
Random shift pad	4
Intensity scale	0.05
Learning rate scheduler	Cosine annealing with warmup
Warmup ratio	0.1
Initial learning rate ratio	0.1
Base learning rate	1e-3
Weight decay	1e-4
Optimizer (β_1, β_2)	AdamW (0.9, 0.999)
Batch size	512

B.3.2. ONLINE RL

Following the standard experimental setup from sample-efficient Atari benchmark (Kaiser et al., 2019; Schwarzer et al., 2021b), we train a Q-learning layer on top of the frozen encoder via the Rainbow algorithm (Hessel et al., 2018). For the sake of simplicity, we do not use noisy layers and use epsilon greedy for exploration. Similar to the Offline BC, we trained the model for 50k steps and the performance is measured by the average gameplay score over 100 attempts. Detailed hyperparameters are listed in Table 16.

Table 16: Hyperparameters for downstream Online RL.

Hyperparameter	Value
Augmentation	[Random Shift, Intensity]
Random shift pad	4
Intensity scale	0.05
Training steps	50k
Update	Distributional Q
Dueling	True
Support of Q-distribution	51
Discount factor γ	0.99
Batch size	32
Optimizer (β_1, β_2, ϵ)	Adam (0.9, 0.999, 0.000015)
Learning rate	0.0001
Max gradient norm	10
Priority exponent	0.5
Priority correction	$0.4 \rightarrow 1$
Exploration Schedule (start, end, steps)	Epsilon Greedy (50k, 1.0, 0.02)
Replay buffer size	50k
Min buffer size for sampling	2000
Replay per training step	1
Updates per replay step	2
Multi-step return length	10
Q-head hidden units	1024
Q-head non-linearity	ReLU
Evaluation trajectories	100



(a) Aggregate scores (b) Algorithm-wise Scores

Figure 9: The results of offline BC and online RL in environments where only the mode is changed from the pre-training game environments.

C. Extended Experimental Results

Here, we present additional experimental results that were not included in the Section 6.

C.1. Game-Mode Change

For Near-OOD in the main experiment (Figure 4), we fine-tuned with offline BC and online RL using games that were different from the pre-training but belonged to the same task genre. However, some might think that simply changing the modes of the games used in pre-training would be more suitable for Near-OOD (Machado et al., 2018). The reason we chose the Near-OOD environment in the main experiment is that changing the mode results in slight changes, such as alterations in the color or shape of objects. Therefore, we thought it was unrealistic because it is very similar to the pre-training environment, indicating that the fine-tuning environment is almost the same as the pre-training environment. For example, let's assume we are training a house-cleaning robot using the pretrain-then-finetune method. In this case, changing the mode is analogous to training the robot on different colored/shaped objects within the same house in the pre-training stage. Assuming that the fine-tuning environment is the same as the pre-training environment is unrealistic.

To support our selection, we conducted offline BC and online RL fine-tuning experiments by changing the modes of the games listed in Table 17. Additionally, to match the scale of the results for each game, we normalized with the final scores of the Rainbow agent trained with 2 million steps as in the Far-OOD setting of the main experiment. The experimental results are shown in Figure 9. Similar to Figure 4(a), learning reward-specific knowledge yields best performance even in the Game-Mode Change environment. This means that the reward function in the Game-Mode Change environment is very similar to the pre-training environment. In other words, simply changing the game mode is not significantly different from the ID.

Table 17: Games used in the game mode change experiment.

Games(Mode)	AirRaid(5), Asteroids(17), Atlantis(3), BankHeist(5), Berzerk(7), Breakout(7), CrazyClimber(3), DemonAttack(3), DoubleDunk(9), Freeway(5), Gravitar(3), Hero(3), Krull(3), MsPacman(3), PrivateEye(3), Skiing(6), SpaceInvaders(9), StarGunner(3), Tutankham(3), YarsRevenge(3) Zaxxon(3)

C.2. Relationship Between Object Size and Mask-Based Methods

One unique property of the Atari environments is the large variance of object sizes, ranging from few hundred pixels to mere 1-2 pixels. This led us to hypothesize that mask-based reconstruction methods may struggle to deal with small objects, as these objects can be entirely occluded by masking operation.

To investigate this, we categorized ID and Near-OOD environments by their object size (see Table 18) and compared the mask-based methods (MAE, SiamMAE) against latent reconstruction methods in the same category (CURL, ATC). As

illustrated in Figure 10, the results indeed support our proposition. Mask-based methods underperformed with small objects and excelled with larger objects, implying that the object size in environments can play a crucial role in these models.

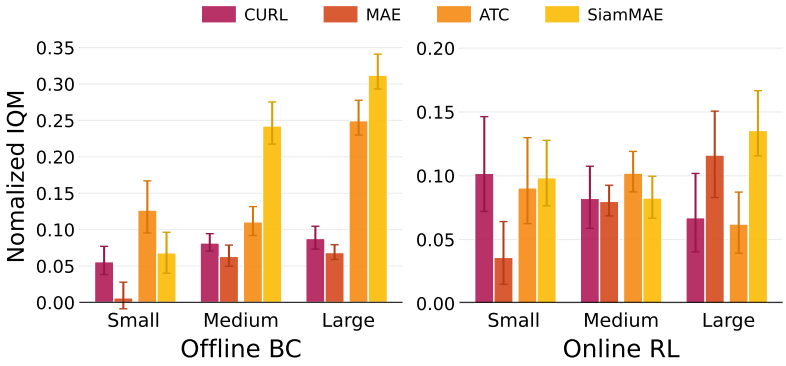


Figure 10: **Mask-based methods relative to object sizes.** A comparative analysis of mask-based versus non-mask methods in ID and Near-OOD games, classified by object size (small, medium, large). The results indicate that mask-based methods tend to excel more in games with larger objects compared to smaller ones.

Table 18: Games categorized by object size. Far-OOD games are not included.

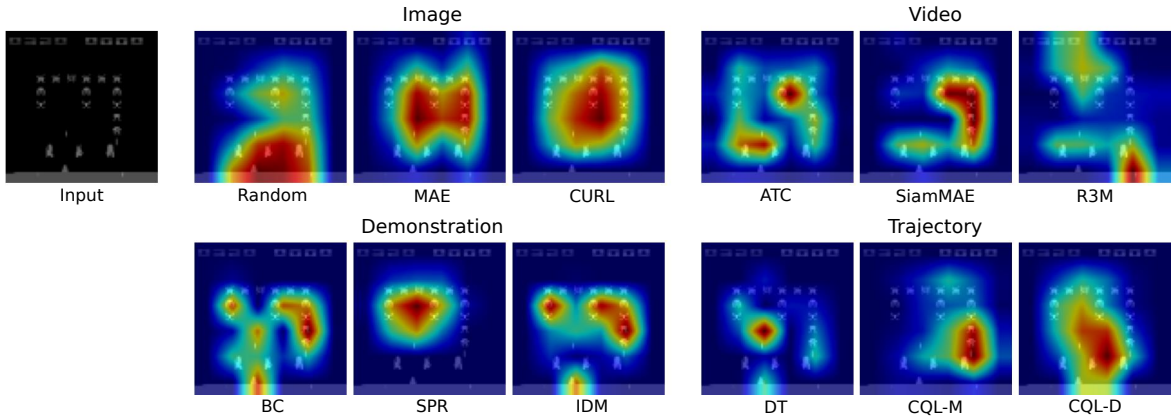
Object size	Games
Small	Amidar, BankHeist, Bowling, Breakout, Centipede, Gravitar, Jamesbond, Krull, MsPacman, Phoenix, Pitfall, Pong, Pooyan, Riverraid, Skiing, StarGunner, TimePilot, Tutankham, Venture, VideoPinball
Medium	Alien, Asterix, Asteroids, Atlantis, BeamRider, Berzerk, Carnival, ChopperCommand, CrazyClimber, DemonAttack, DoubleDunk, ElevatorAction, FishingDerby, Freeway, Frostbite, Hero, IceHockey, JourneyEscape, Kangaroo, MontezumaRevenge, NameThisGame, PrivateEye, Qbert, Robotank, Seaquest Solaris, SpaceInvaders, Tennis, WizardOfWor, YarsRevenge
Large	AirRaid, Assault, BattleZone, Boxing, Enduro, Gopher, KungFuMaster, RoadRunner, UpNDown, Zaxxon

C.3. Eigen-CAM Analysis

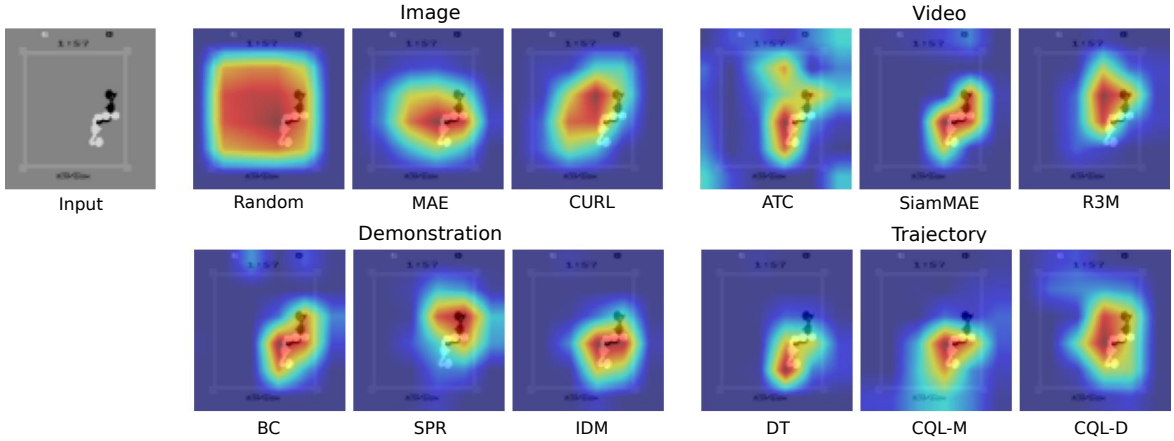
In this section, we analyze the pre-trained backbones based on what aspects of the observations they concentrate on. We employed Eigen-CAM (Muhammad & Yeasin, 2020), a visualization method frequently used in computer vision and deep learning. We randomly sampled an observation from two In-Distribution and two Near-Out-of-Distribution games, and applied Eigen-CAM on our main models in Section 6.

C.3.1. In-Distribution environment

For In-Distribution environments, we employ 'Space Invaders' and 'Boxing'. The outcomes of this analysis are illustrated in Figure 11. It was observed that when the encoder is pre-trained solely with images, it struggles to capture objects relevant to the task. However, when pre-trained with data enriched with additional information such as temporal dynamics and task-relevant information, and values of states and state-action pairs, the encoder effectively identifies meaningful objects.



(a) SpaceInvaders

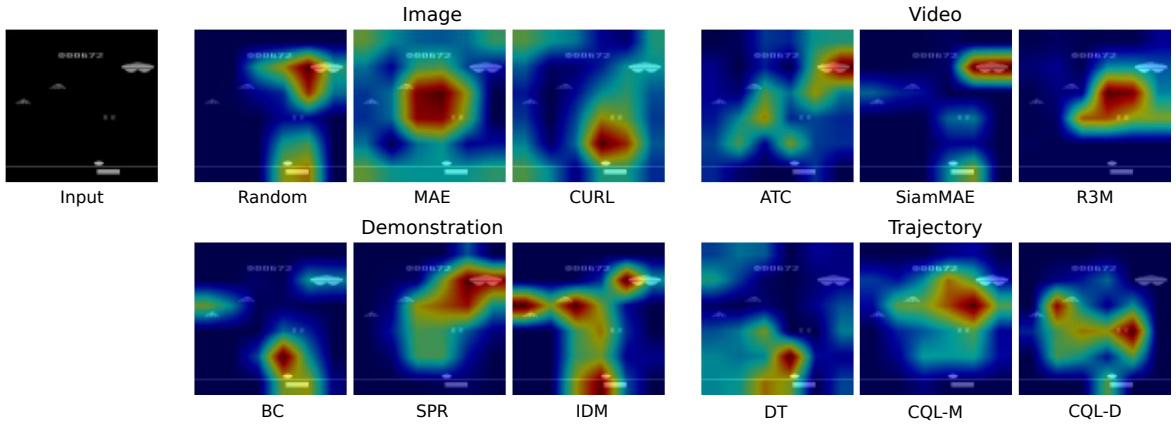


(b) Boxing

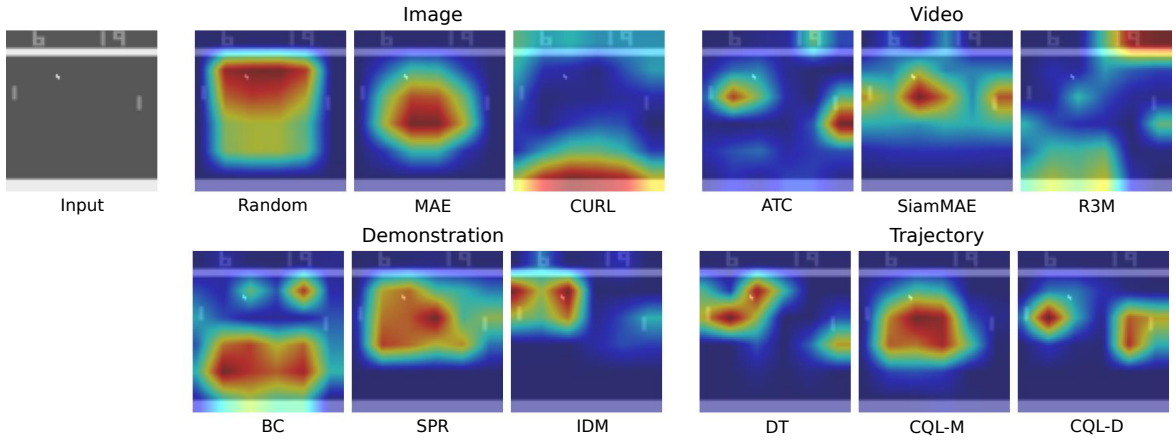
Figure 11: **Eigen-CAM analysis of ID games.** (a) Space Invaders is a game in which players are required to evade enemy attacks while eliminating as many adversaries as possible through vertical shooting. (b) Boxing is a game where the player aims to score more points than their opponent by striking the opponent's face as many as possible within a set time limit. When looking at Eigen-CAM images, it's evident that encoders pre-trained with temporal dynamics and task-related information such as video, action, and reward, rather than just spatial information from images, are more effective in identifying objects relevant to the task. Additionally, encoders trained using demonstration data, which includes task-relevant information, show improved ability in capturing and understanding important features for the task.

C.3.2. Near-Out-of-Distribution environment

We also analyze the pre-trained encoder in an out-of-distribution environment, which was not used during its pre-training phase. Specifically, 'Assault' and 'Pong' are utilized for this purpose. The outcomes are shown in Figure 12. Similar to the situation with an in-distribution environment, encoders pre-trained on images alone had difficulty in targeting significant objects, while those pre-trained on a richer dataset demonstrated effective identification of important objects.



(a) Assault



(b) Pong

Figure 12: **Eigen-CAM analysis of Near-OOD games.** (a) Assault is a game where players must avoid enemy assaults and eliminate as many opponents as possible through either vertical or horizontal shooting. (b) Pong is a game where the player who reaches a specified score wins by scoring points while preventing the opponent from hitting the ball. Similar to what we observe in ID, encoders that are trained on temporal dynamics, task-related information, and the value between states and actions using videos, demonstrations, and trajectories demonstrate superior ability in recognizing task-related objects, compared to those that only learn spatial aspects from images.

D. Full Results

We provide the individual game scores of all algorithms in our experiments. The scores are recorded at the end of fine-tuning and averaged over 3 seeds. We also include the mean scores of a randomly behaving agent (aliased *RndmAgnt*) and the DQN scores reported by Castro et al. (2018), which are used to compute the DQN normlized score with the formula: $s_{\text{DNS}} = (s_{\text{agent}} - s_{\text{min}})/(s_{\text{max}} - s_{\text{min}})$ where $s_{\text{max}} = \max(s_{DQN}, s_{RndmAgnt})$, $s_{\text{min}} = \min(s_{DQN}, s_{RndmAgnt})$, and s_{agent} is the score to be normalized. For Far-OOD games, instead of DQN, we report the scores of Rainbow (Hessel et al., 2018) trained for 2M steps and use them for normalization. In addition to IQM, we report the optimality gap of the normalized scores.

D.1. Main Results

Table 19: Mean episodic scores of Offline BC on In-Distribution games.

Game	*RndmAgnt*	*DQN*	Random	MAE	CURL	ATC	SiamMAE	R3M†	BC	SPR	IDM	DT	CQL-M	CQL-D
AirRaid	579.2	7479.5	1262.2	973.8	970.1	1479.6	3106.9	1422.1	4290.6	1521.2	2743.8	3497.9	3189.5	3754.3
Amidar	5.8	1207.7	55.5	60.1	28.9	60.3	34.8	44.8	65.0	58.0	56.8	58.6	69.4	78.5
Asteroids	719.1	698.4	668.2	603.3	635.1	693.6	826.6	800.4	671.1	872.8	724.8	812.2	794.3	998.3
Atlantis	12850.0	853640.0	7607.7	18895.3	12785.7	13607.0	49505.7	29909.7	24181.3	472835.3	48420.7	25266.7	22112.7	16246.3
BankHeist	14.2	601.8	0.0	18.2	32.9	8.5	21.7	23.8	23.4	40.9	26.1	30.6	62.3	70.8
BattleZone	2360.0	17784.8	3830.0	3040.0	2810.0	8196.7	5150.0	8220.0	6393.3	3780.0	7850.0	5086.7	9323.3	7003.3
Berzerk	123.7	487.5	231.5	311.1	246.1	361.1	393.9	365.3	314.1	371.4	296.0	336.7	422.1	374.4
Bowling	23.1	30.1	0.0	42.9	28.9	29.7	38.8	27.6	28.7	29.1	36.0	33.4	40.9	34.6
Boxing	0.1	78.0	9.5	14.9	54.7	59.4	60.1	52.3	32.3	82.5	49.5	67.0	72.2	45.3
Breakout	1.7	96.2	1.1	1.3	3.4	7.0	2.3	7.8	22.5	25.1	16.8	22.2	41.2	31.1
Carnival	700.8	4784.8	389.7	494.7	759.3	882.0	563.7	683.1	872.9	979.3	1192.1	1536.7	1426.5	1118.7
Centipede	2090.9	2583.0	1700.3	1588.6	1759.7	1471.7	1451.6	1788.7	1815.9	2243.2	1657.5	1821.8	2180.6	1999.3
ChopperCommand	811.0	2690.6	438.3	914.0	1044.3	1130.3	1393.7	1066.3	1092.3	1053.0	1268.3	1526.3	1548.7	1360.0
CrazyClimber	10780.5	104568.8	3548.3	7762.0	3066.7	3892.0	25284.7	55393.3	16002.3	57875.0	55794.7	44730.3	49646.3	82641.3
DemonAttack	152.1	6361.6	44.2	110.4	143.1	126.1	241.3	226.0	269.0	175.6	193.0	533.8	1154.5	1489.3
DoubleDunk	−18.6	−6.5	−15.3	−20.9	−19.1	−18.3	−19.6	−18.2	−19.8	−16.5	−17.7	−18.6	−16.9	−19.4
ElevatorAction	4387.0	439.8	0.3	303.7	378.3	617.3	523.7	538.7	870.0	1683.7	2978.0	3014.3	1005.7	1130.0
Enduro	0.0	628.9	5.1	5.4	0.0	15.0	48.0	182.0	152.6	150.9	61.6	141.3	100.3	274.0
FishingDerby	−91.7	0.6	−90.4	−81.1	−86.7	−80.1	−53.0	−74.7	−62.3	−86.8	−43.4	−56.7	−74.5	−75.2
Freeway	0.0	26.3	13.0	15.6	20.3	22.0	24.9	20.1	24.1	21.0	22.5	25.9	19.3	22.4
Frostbite	65.2	367.1	95.9	252.1	109.5	213.6	214.6	256.4	396.4	130.9	393.7	185.1	507.5	273.2
Gopher	257.6	5479.9	85.1	323.7	327.3	983.5	709.7	1908.5	1050.3	466.9	749.0	1868.8	1232.4	1064.6
Gravitar	173.0	330.1	77.5	202.3	213.8	242.8	227.2	262.5	236.2	232.8	273.5	222.2	196.3	230.7
Hero	1027.0	17325.4	4282.8	10054.2	7929.6	6222.6	11861.3	4379.3	7053.6	13418.1	7768.3	7817.3	12239.7	10207.0
IceHockey	−11.2	−5.8	−1.6	−10.8	−10.0	−8.6	−8.6	−8.1	−7.6	−9.3	−7.3	−8.3	−8.0	−8.5
Jamesbond	29.0	573.3	202.5	156.8	214.8	385.0	397.2	344.3	299.3	322.2	371.2	295.3	302.8	335.5
Kangaroo	52.0	11486.0	500.7	1012.0	954.7	1057.3	1697.0	868.3	1555.7	1147.3	1375.3	1255.0	951.3	1835.7
Krull	1598.0	6097.6	2404.5	2065.8	4232.7	2922.0	5806.2	3611.2	4684.5	6297.9	5915.6	5714.5	7356.0	7155.1
KungFuMaster	258.5	23435.4	1.0	1696.0	287.7	3970.7	9548.0	7484.3	10450.7	15001.3	4622.3	9729.3	10157.7	9852.7
MontezumaRevenge	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
MsPacman	307.3	3402.4	273.8	861.4	648.4	629.5	1361.2	1132.8	1334.1	1496.5	1475.3	1374.1	2205.8	2027.4
NameThisGame	2292.3	7278.6	2367.4	4046.7	3745.6	4074.7	5624.4	5028.9	6083.7	4317.9	5688.0	5890.2	6059.5	6209.7
Phoenix	761.4	4996.6	193.6	582.8	749.0	1144.7	1314.6	1886.4	1764.5	1794.7	1686.4	2278.5	2498.5	2245.8
Pitfall	−229.4	−73.8	−19.8	−131.8	−214.9	−77.8	−178.8	−56.3	−47.6	−36.2	−45.4	−88.8	−55.8	−88.8
PrivateEye	24.9	−16.0	−527.5	262.4	−924.8	−380.7	98.7	93.1	−703.8	−111.4	−635.1	−303.3	−138.7	−451.7
Qbert	163.9	10117.5	133.6	390.8	294.4	298.8	389.9	399.7	359.6	367.8	376.1	474.2	2110.8	2353.5
RoadRunner	11.5	36925.5	2456.7	2304.3	3373.0	6310.0	4098.3	3980.0	9162.7	5340.7	3706.3	3498.0	1104.0	4979.7
Robotank	2.2	59.8	11.5	5.4	6.2	9.6	13.0	12.3	11.5	9.3	10.9	15.2	11.7	10.2
Skiing	−17098.1	−15824.6	−29970.8	−29970.8	−29793.3	−29332.6	−29970.8	−29970.6	−29897.7	−29970.1	−29704.3	−29934.4	−29491.2	−29970.8
Solaris	1236.3	1436.4	971.5	1916.9	2076.9	2003.8	2022.0	2046.4	1843.2	2166.9	2017.7	2314.7	1592.7	1567.8
SpaceInvaders	148.0	1794.2	190.8	190.6	198.4	179.4	296.2	215.6	266.0	169.5	250.2	209.8	168.4	263.8
StarGunner	664.0	42165.2	874.7	726.0	1248.7	3664.3	2472.3	2610.7	2765.0	2183.7	2973.0	2406.3	3543.7	2748.3
Tennis	−23.8	−1.5	−17.9	−23.9	−5.7	−23.7	−18.1	−16.1	−11.8	−2.3	−18.2	−16.2	−8.3	−10.7
TimePilot	3568.0	3654.4	1851.3	3267.0	2805.0	3473.0	3240.3	2913.7	3112.0	2243.0	4027.7	2747.7	3633.0	3904.3
Tutankham	11.4	103.8	0.7	1.6	20.5	12.2	4.3	30.9	14.4	18.8	22.5	23.3	45.1	17.9
UpNDown	533.4	8488.3	27.3	2509.3	2151.1	3325.5	4018.2	3565.1	3120.2	5087.2	3478.9	3115.5	2549.3	4288.6
VideoPinball	16256.9	63406.1	1648.6	91.6	5.4	3158.2	1080.6	2512.5	1172.3	8900.4	678.6	1516.5	2120.1	298.2
WizardOfWor	563.5	2065.8	132.3	251.3	572.3	436.0	354.3	323.7	329.0	604.7	525.7	703.3	556.3	543.0
YarsRevenge	3092.9	23909.4	9073.7	8664.9	10317.8	10882.1	11535.6	9253.9	15613.9	8628.6	11851.9	12560.9	13291.8	15732.1
Zaxxon	32.5	4538.6	421.7	1870.3	2174.0	2774.3	3280.7	3386.3	2494.3	2262.0	2360.3	2460.7	3849.7	3291.0
IQM(DNS)	0.0	1.0	−0.0077	0.0554	0.0711	0.1389	0.2209	0.2201	0.2142	0.2655	0.2644	0.2830	0.3302	0.3160
Optimality Gap(DNS)	0.0	1.0	1.9248	1.2456	1.7360	1.2762	1.0006	1.1541	1.4445	1.2556	1.1981	1.2174	0.9126	1.0855

Table 20: Mean episodic scores of Offline BC on Near-Out-of-Distribution games.

Game	*RndmAgnt*	*DQN*	Random	MAE	CURL	ATC	SiamMAE	R3M†	BC	SPR	IDM	DT	CQL-M	CQL-D
Alien	227.8	2484.5	553.5	381.3	536.6	590.8	863.0	539.0	727.0	701.1	816.8	797.8	831.6	469.8
Assault	222.4	1525.2	414.6	134.0	396.8	623.0	726.7	618.4	758.5	550.0	751.1	795.0	738.2	702.7
Asterix	210.0	2711.4	281.5	236.8	294.8	365.3	349.5	392.8	432.7	347.5	487.7	470.0	421.7	521.2
BeamRider	363.9	5852.4	548.3	522.7	533.9	543.5	523.6	623.4	554.3	471.0	551.1	574.4	535.9	519.5
JourneyEscape	−19977.0	−3671.1	−8998.0	−12759.3	−14779.0	−11430.0	−11594.0	−12088.7	−11677.3	−14696.3	−13380.0	−11251.3	−12415.3	−12341.3
Pong	−20.7	16.6	−20.7	−19.9	−20.6	−6.9	−16.3	−3.2	−0.9	−2.3	3.0	4.2	−2.2	−4.7
Pooyan	371.2	3212.0	18.3	295.4	812.5	690.0	502.3	967.9	880.0	745.5	912.4	712.7	518.5	456.2
Riverraid	1338.5	11638.9	833.3	1638.8	2021.0	2877.5	2156.1	2579.3	2743.1	2667.6	2851.5	3002.0	2474.2	3181.3
Seaquest	68.4	1600.7	75.3	171.3	232.1	252.7	450.6	336.9	479.3	408.7	437.9	392.7	402.7	488.5
Venture	0.0	39.1	0.0	0.7	13.0	8.7	2.7	0.0	5.7	0.0	2.3	2.7	26.0	6.0
IQM(DNS)	0.0	1.0	0.0250	0.0274	0.0930	0.1643	0.1437	0.1608	0.2313	0.1575	0.2200	0.2090	0.2032	0.1915
Optimality Gap(DNS)	0.0	1.0	0.9141	0.9411	0.8682	0.7940	0.8172	0.7980	0.7476	0.8165	0.7510	0.7401	0.7529	0.7839

Table 21: Mean episodic scores of Offline BC on Far-Out-of-Distribution games.

Game	*RndmAgnt*	*Rainbow*	Random	MAE	CURL	ATC	SiamMAE	R3M†	BC	SPR	IDM	DT	CQL-M	CQL-D
BasicMath	0.1334	3.5700	0.6300	1.3700	2.4500	2.8633	3.4967	3.4000	2.7200	2.9467	3.5867	2.5167	2.8167	3.3300
HumanCannonball	1.0767	5.5600	0.7067	1.5033	1.1533	2.8133	1.0500	1.0267	1.8133	0.8167	1.1900	1.0600	1.1833	1.3433
Klax	0.3734	2124.7500	4.4000	3.4167	6.7667	12.8333	13.4333	66.8167	40.6667	23.1000	25.2667	47.8500	28.2833	21.4667
Othello	−21.3233	−2.0100	−0.0067	0.0167	−0.0200	−0.1500	0.1167	0.0767	0.0067	0.0667	0.0867	0.0933	0.1933	0.1333
Surround	−9.9833	−7.8100	−9.9967	−9.2533	−9.2800	−9.0533	−8.9067	−8.9833	−9.1100	−9.2300	−8.7600	−8.9567	−9.2400	−9.0233
IQM(RNS)	0.0	1.0	0.0524	0.2642	0.3547	0.5365	0.4955	0.4831	0.4396	0.3957	0.5322	0.4190	0.3896	0.4861
Optimality Gap(RNS)	0.0	1.0	0.7885	0.6415	0.5964	0.4769	0.5090	0.5138	0.5324	0.5766	0.4855	0.5631	0.5681	0.5117

Table 22: Mean episodic scores of Online RL on In-Distribution games.

Game	*RndmAgnt*	*DQN*	Random	MAE	CURL	ATC	SiamMAE	R3M†	BC	SPR	IDM	DT	CQL-M	CQL-D
AirRaid	579.2	7479.5	368.2	601.1	502.8	718.0	639.6	570.6	803.7	512.7	636.4	782.2	1087.8	848.9
Amidar	5.8	1207.7	131.0	110.3	105.3	109.0	105.8	95.7	165.3	83.1	111.2	105.5	116.5	128.3
Asteroids	719.1	698.4	1008.3	739.4	755.6	811.5	751.2	718.6	701.2	918.4	894.5	857.7	821.3	1061.6
Atlantis	12850.0	853640.0	10404.7	17912.0	29700.7	14236.7	18619.3	24911.0	19350.0	79626.0	18266.3	30214.7	31849.7	38407.3
BankHeist	14.2	601.8	9.7	40.9	37.3	39.9	54.8	56.1	357.1	57.7	63.9	47.8	300.3	569.5
BattleZone	2360.0	17784.8	2863.3	5056.7	2513.3	4493.3	7756.7	7810.0	8473.3	2593.3	4446.7	4583.3	5850.0	5446.7
Berzerk	123.7	487.5	503.5	419.9	468.3	468.9	340.6	438.7	478.3	410.0	391.6	353.4	484.6	420.1
Bowling	23.1	30.1	15.8	23.7	32.7	46.2	40.3	49.7	41.4	43.5	45.1	46.5	58.0	31.0
Boxing	0.1	78.0	−0.3	7.6	2.0	−2.4	−1.4	1.9	3.0	6.4	2.8	1.1	8.1	1.6
Breakout	1.7	96.2	2.1	3.4	3.7	6.5	3.3	7.0	13.2	18.5	8.9	9.8	33.4	26.5
Carnival	700.8	4784.8	453.2	515.0	660.5	690.0	667.1	692.1	857.8	662.9	716.6	648.7	869.1	856.3
Centipede	2090.9	2583.0	3196.4	2485.1	3060.6	3014.3	2409.1	2494.6	2519.1	2971.1	2216.9	2849.3	3087.5	2865.3
ChopperCommand	811.0	2690.6	697.7	799.3	459.7	425.7	711.0	909.3	822.3	1053.0	870.0	661.7	771.3	648.7
CrazyClimber	10780.5	104568.8	4764.3	6639.3	2468.3	2274.7	4712.7	6419.3	18996.7	29972.7	17333.0	18018.0	25023.7	40734.0
DemonAttack	152.1	6361.6	360.6	201.7	324.9	253.9	201.1	296.1	313.1	319.2	316.4	432.1	605.5	1930.4
DoubleDunk	−18.6	−6.5	−17.6	−19.9	−14.7	−18.0	−17.6	−16.8	−18.1	−18.1	−17.1	−16.7	−15.4	−18.3
ElevatorAction	4387.0	439.8	0.0	0.0	0.0	81.0	4.3	0.0	0.0	11.3	137.0	1041.7	0.0	19.7
Enduro	0.0	628.9	30.9	20.1	13.0	17.0	27.3	93.4	52.7	47.8	23.8	53.3	56.3	82.6
FishingDerby	−91.7	0.6	−94.4	−83.9	−93.5	−89.3	−87.8	−85.6	−86.7	−92.2	−86.7	−90.7	−88.0	−86.6
Freeway	0.0	26.3	21.9	24.9	21.8	25.0	24.7	23.7	24.0	21.5	24.2	22.4	22.0	22.4
Frostbite	65.2	367.1	458.8	595.8	291.3	219.9	329.6	231.5	583.1	203.6	234.4	226.1	1167.5	251.6
Gopher	257.6	5479.9	287.0	498.9	287.3	292.4	423.5	433.9	592.9	516.1	410.9	640.1	632.0	430.2
Gravitar	173.0	330.1	88.3	109.3	219.2	245.8	240.2	183.0	225.5	234.0	255.2	225.3	220.2	309.2
Hero	1027.0	17325.4	3064.9	3745.5	3079.1	4471.2	4502.6	4665.1	6241.9	4613.6	4021.9	5498.0	4622.4	6531.4
IceHockey	−11.2	−5.8	−10.7	−13.1	−14.0	−12.5	−12.7	−10.1	−11.1	−12.7	−10.3	−11.1	−10.8	−11.1
Jamesbond	29.0	573.3	109.5	193.3	205.8	170.3	121.8	161.8	176.8	196.7	191.5	140.0	248.8	201.8
Kangaroo	52.0	11486.0	569.7	1064.3	478.0	553.7	843.7	391.3	656.7	611.7	382.7	567.3	3158.3	1363.7
Krull	1598.0	6097.6	2287.0	3424.7	4114.6	2609.5	3240.5	3556.4	3623.2	4155.2	3655.0	3625.4	4281.5	5052.2
KungFuMaster	258.5	23435.4	1583.7	4777.3	5173.7	2168.0	5183.3	4927.7	6800.7	11493.3	5513.3	7797.7	13286.0	10442.7
MontezumaRevenge	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
MsPacman	307.3	3402.4	767.4	758.8	923.6	622.8	789.5	753.8	932.3	942.5	870.6	794.0	1080.6	946.2
NameThisGame	2292.3	7278.6	2603.1	2676.2	2580.8	2036.4	2420.6	2111.5	2978.6	3901.9	2166.0	2393.6	2765.5	2986.1
Phoenix	761.4	4996.6	659.6	566.1	733.6	673.0	748.7	1234.1	1447.0	1101.9	1282.9	1315.7	1818.7	1754.9
Pitfall	−229.4	−73.8	−3.7	−17.6	−9.2	−259.9	−71.9	−352.5	−418.2	−235.7	−726.4	−93.3	−33.3	−87.0
PrivateEye	24.9	−16.0	47.2	96.3	94.7	2568.0	527.4	178.6	603.5	83.7	19.5	25.9	81.3	99.7
Qbert	163.9	10117.5	306.8	532.0	442.2	594.4	845.8	457.0	1664.8	404.2	577.4	515.2	1063.7	978.5
RoadRunner	11.5	36925.5	1913.7	4417.0	3797.7	3613.3	3092.7	2944.7	5389.0	9028.7	2936.0	2606.3	10322.7	3643.3
Robotank	2.2	59.8	7.3	5.1	4.3	5.3	5.4	4.6	5.8	4.2	4.5	4.2	5.8	4.5
Skiing	−17098.1	−15824.6	−29970.8	−29970.8	−29859.5	−28785.2	−30028.2	−29970.8	−28532.4	−29970.1	−29633.2	−30059.1	−29970.8	−29970.8
Solaris	1236.3	1436.4	856.7	757.7	1138.9	1647.3	1222.3	2269.1	1423.9	2256.5	1888.7	1431.3	1661.9	1230.0
SpaceInvaders	148.0	1794.2	326.5	331.3	252.4	275.3	264.6	257.7	288.5	304.1	210.0	415.2	274.5	367.2
StarGunner	664.0	42165.2	812.0	873.3	1087.3	998.7	606.0	1124.0	930.3	1419.3	1095.0	935.7	1239.3	1744.7
Tennis	−23.8	−1.5	−4.5	−7.6	−18.4	−7.7	−22.8	−12.5	−9.1	−11.4	−14.2	−16.8	−10.3	−10.0
TimePilot	3568.0	3654.4	2620.0	1796.3	2307.7	3196.0	3046.3	2711.3	3215.7	2730.0	3156.3	2409.0	2592.0	2772.7
Tutankham	11.4	103.8	73.6	34.0	46.1	4.5	58.2	90.9	48.7	56.6	33.7	36.8	75.1	49.8
UpNDown	533.4	8488.3	1964.3	2287.2	2447.2	2359.9	2654.8	3079.5	2530.7	4988.3	3067.3	3400.8	2625.3	2902.1
VideoPinball	16256.9	63406.1	15487.9	10895.1	11280.4	8283.4	11055.4	14538.1	14761.6	28978.4	17023.7	19717.4	14876.2	19314.0
WizardOfWor	563.5	2065.8	252.0	448.3	505.3	428.0	637.7	696.0	900.3	899.7	858.7	839.3	690.7	640.0
YarsRevenge	3092.9	23909.4	8204.7	7125.4	9463.7	9095.8	7382.7	7295.3	6853.2	7784.7	7976.2	6639.3	7665.6	7820.2
Zaxxon	32.5	4538.6	921.0	860.7	779.3	221.3	1693.7	905.3	1957.3	0.0	2060.7	434.3	707.3	282.7
IQM(DNS)	0.0	1.0	0.0453	0.0706	0.0799	0.0685	0.0902	0.1151	0.1480	0.1594	0.1080	0.1141	0.1941	0.1878
Optimality Gap(DNS)	0.0	1.0	1.3154	1.4812	1.3200	1.1852	1.1472	1.2228	1.0846	1.1394	1.2301	1.3125	1.1299	1.1950

Table 23: Mean episodic scores of Online RL on Near-Out-of-Distribution games.

Game	*RndmAgnt*	*DQN*	Random	MAE	CURL	ATC	SiamMAE	R3M†	BC	SPR	IDM	DT	CQL-M	CQL-D
Alien	227.8	2484.5	444.2	641.9	595.0	659.7	566.0	667.2	772.9	604.3	546.3	538.0	640.8	621.0
Assault	222.4	1525.2	607.7	572.8	439.8	453.9	564.7	379.3	371.4	510.3	334.9	351.3	545.9	461.1
Asterix	210.0	2711.4	536.2	476.0	533.8	485.2	573.8	496.7	499.2	571.8	608.0	396.5	615.2	545.2
BeamRider	363.9	5852.4	581.1	479.0	597.5	542.4	575.9	701.5	598.2	510.2	562.7	560.6	674.4	591.5
JourneyEscape	−19977.0	−3671.1	−6439.0	−12185.3	−7828.3	−11409.0	−10866.3	−11459.3	−9623.0	−11564.3	−14326.0	−17966.0	−9327.0	−8893.0
Pong	−20.7	16.6	−20.2	−20.4	−18.3	−11.8	−17.6	−11.7	2.8	0.6	−9.0	−1.0	−12.0	−5.4
Pooyan	371.2	3212.0	349.1	410.1	402.4	767.8	931.8	1171.9	743.6	1049.1	1027.9	917.8	954.3	1026.2
Riverraid	1338.5	11638.9	1855.6	2185.2	2174.7	2268.1	2477.1	2358.6	2416.7	2423.5	2728.5	1999.8	2448.6	2343.7
Seaquest	68.4	1600.7	273.3	280.1	214.7	228.9	214.3	259.9	230.3	278.5	250.6	198.9	376.7	298.7
Venture	0.0	39.1	0.0	0.0	16.0	35.3	0.0	1.7	0.0	0.0	0.0	0.0	0.0	0.0
IQM(DNS)	0.0	1.0	0.0783	0.0950	0.1016	0.1332	0.1261	0.1466	0.1255	0.1657	0.1342	0.0948	0.1758	0.1595
Optimality Gap(DNS)	0.0	1.0	0.8418	0.8702	0.8168	0.8057	0.8360	0.8195	0.7909	0.7873	0.8432	0.8685	0.7950	0.7899

Table 24: Mean episodic scores of Online RL on Far-Out-of-Distribution games.

Game	*RndmAgnt*	*Rainbow*	Random	MAE	CURL	ATC	SiamMAE	R3M†	BC	SPR	IDM	DT	CQL-M	CQL-D
BasicMath	0.1334	3.5700	0.0000	0.2233	0.8700	0.0967	0.3367	0.2400	0.0900	0.2667	0.6533	0.9233	0.0700	0.0633
HumanCannonball	1.0767	5.5600	1.0833	1.0500	1.4633	1.6667	2.0400	1.6767	1.5033	1.3400	1.9767	1.4967	1.6667	1.4400
Klax	0.3734	2124.7500	5.1667	6.0500	10.8000	7.5000	10.1000	13.2667	11.5333	9.3167	9.5667	10.4500	10.0833	10.1500
Othello	−21.3233	−2.0100	0.0000	0.3400	−0.5067	0.0667	−0.0667	−0.1000	−0.6933	−0.1067	0.0200	−0.0400	−0.7400	−0.0667
Surround	−9.9833	−7.8100	−9.9433	−9.8633	−9.7933	−9.8067	−9.8800	−9.6867	−9.7200	−9.7467	−9.7400	−9.8133	−9.8167	−9.8333
IQM(RNS)	0.0	1.0	0.0074	0.0413	0.1353	0.0769	0.1080	0.1120	0.0739	0.0739	0.1595	0.1388	0.0730	0.0640
Optimality Gap(RNS)	0.0	1.0	0.8033	0.7844	0.7214	0.7589	0.7348	0.7385	0.7585	0.7579	0.7063	0.7187	0.7616	0.7731

D.2. Ablation Results

Table 25: Offline BC scores on In-Distribution games, after pre-training with datasets of differing optimality.

			Suboptimal				Expert			
Game	*RndmAgnt*	*DQN*	CURL	ATC	BC	CQL-D	CURL	ATC	BC	CQL-D
AirRaid	579.2	7479.5	1216.8	1699.8	2955.7	3120.6	1213.8	1393.0	4803.5	3953.7
Amidar	5.8	1207.7	44.1	37.5	68.6	63.2	47.7	44.6	67.1	92.5
Asteroids	719.1	698.4	767.5	725.5	614.5	833.9	735.8	742.2	593.4	849.4
Atlantis	12850.0	853640.0	24057.3	24262.3	16286.3	15749.3	15637.3	18773.7	22173.0	9151.0
BankHeist	14.2	601.8	18.2	22.1	12.4	20.9	18.0	18.1	15.8	49.3
BattleZone	2360.0	17784.8	3416.7	6936.7	3633.3	5323.3	4083.3	5783.3	5750.0	10216.7
Berzerk	123.7	487.5	285.9	402.9	354.6	349.3	297.7	423.9	367.2	366.2
Bowling	23.1	30.1	26.6	44.7	28.9	30.2	20.9	42.1	29.7	29.2
Boxing	0.1	78.0	44.1	35.4	39.9	50.8	45.9	61.9	27.6	57.3
Breakout	1.7	96.2	6.2	10.0	24.1	27.8	6.7	6.3	30.5	43.3
Carnival	700.8	4784.8	797.7	732.3	876.3	739.8	859.7	1082.3	1555.9	802.9
Centipede	2090.9	2583.0	1936.7	1779.4	2188.5	2279.9	2164.3	2053.9	1942.1	1989.8
ChopperCommand	811.0	2690.6	1169.7	1307.0	1163.3	1213.3	1228.3	1043.0	784.0	1678.7
CrazyClimber	10780.5	104568.8	11217.0	6319.3	29986.3	60791.0	14040.0	7754.7	40367.3	93106.7
DemonAttack	152.1	6361.6	193.0	167.0	269.8	332.5	156.3	146.1	347.6	480.2
DoubleDunk	−18.6	−6.5	−18.7	−18.6	−19.2	−19.9	−18.7	−17.9	−20.1	−19.6
ElevatorAction	4387.0	439.8	202.3	937.7	810.0	1634.3	899.3	116.7	560.0	1107.0
Enduro	0.0	628.9	0.9	25.3	74.7	507.2	0.0	15.7	137.9	434.1
FishingDerby	−91.7	0.6	−91.1	−79.1	−66.2	−71.3	−91.2	−79.8	−66.7	−59.0
Freeway	0.0	26.3	18.1	12.3	26.5	19.9	21.8	13.5	24.7	21.4
Frostbite	65.2	367.1	157.2	322.1	240.5	398.8	137.0	254.8	256.9	345.9
Gopher	257.6	5479.9	527.0	944.9	1065.7	433.8	681.7	736.8	985.3	1135.1
Gravitar	173.0	330.1	192.8	244.8	246.8	220.3	255.3	270.8	255.7	214.8
Hero	1027.0	17325.4	4787.0	6575.6	5147.0	8377.1	7361.9	5629.3	9667.0	13268.4
IceHockey	−11.2	−5.8	−11.9	−9.5	−9.1	−9.1	−10.0	−9.7	−9.5	−7.6
Jamesbond	29.0	573.3	192.5	411.2	232.8	183.3	126.2	316.0	336.0	305.2
Kangaroo	52.0	11486.0	1002.0	1341.3	1349.0	1113.0	877.0	1044.3	1076.7	1164.0
Krull	1598.0	6097.6	5056.4	3480.2	4143.7	5825.2	5194.4	3332.6	5709.1	4750.2
KungFuMaster	258.5	23435.4	3423.7	4667.3	5678.7	5188.3	4144.0	3210.0	9304.7	12541.0
MontezumaRevenge	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
MsPacman	307.3	3402.4	739.4	662.2	1105.1	1539.4	703.7	846.8	1363.7	1227.9
NameThisGame	2292.3	7278.6	4093.7	4330.7	5475.6	6598.1	4866.3	4784.4	6007.1	6486.4
Phoenix	761.4	4996.6	1179.9	1148.7	1281.0	2018.8	1078.0	1169.2	1910.4	1204.2
Pitfall	−229.4	−73.8	−124.3	−158.4	−55.8	−97.5	−40.8	−68.2	−59.5	−46.3
PrivateEye	24.9	−16.0	−271.4	−348.5	−541.4	747.5	−749.4	1729.7	−581.5	−10.0
Qbert	163.9	10117.5	263.3	334.1	312.0	942.8	404.1	188.3	1144.2	706.3
RoadRunner	11.5	36925.5	2965.3	4571.3	6576.3	2367.3	4028.7	4733.7	4466.7	9201.7
Robotank	2.2	59.8	6.1	11.2	10.5	7.1	6.4	11.6	14.2	12.2
Skiing	−17098.1	−15824.6	−29970.8	−29069.5	−29120.5	−29970.1	−29970.8	−29961.2	−29970.8	−29970.8
Solaris	1236.3	1436.4	2026.2	1795.9	1729.2	1676.6	2461.1	1795.5	1692.5	1583.8
SpaceInvaders	148.0	1794.2	184.7	168.9	205.6	309.6	218.3	160.5	283.0	215.0
StarGunner	664.0	42165.2	1254.7	2689.0	3698.3	2024.0	1190.7	4374.3	3532.0	2957.7
Tennis	−23.8	−1.5	−14.5	−8.7	−16.4	−17.4	−11.4	−4.6	−5.5	−20.7
TimePilot	3568.0	3654.4	2645.0	3343.7	3633.0	2846.0	3341.0	3117.3	3517.7	3422.0
Tutankham	11.4	103.8	9.3	21.2	4.6	28.1	17.8	18.7	8.7	19.5
UpNDown	533.4	8488.3	2085.7	2400.8	3389.2	3365.7	2435.0	2306.2	3377.7	3677.9
VideoPinball	16256.9	63406.1	222.6	513.6	538.7	2211.7	106.1	731.0	627.1	715.3
WizardOfWor	563.5	2065.8	638.7	366.0	518.3	323.7	581.0	281.7	167.0	385.0
YarsRevenge	3092.9	23909.4	11853.3	10337.0	15739.8	12818.0	7919.2	9642.5	12798.3	15447.9
Zaxxon	32.5	4538.6	1469.3	3143.7	3054.3	3372.0	1706.7	3168.3	3186.0	3480.7
IQM(DNS)	0.0	1.0	0.0822	0.1567	0.1829	0.2527	0.1107	0.1550	0.2376	0.3421
Optimality Gap(DNS)	0.0	1.0	1.5000	1.3059	1.3171	1.1301	1.4567	1.0506	1.3595	0.9363

Table 26: Offline BC scores on Near-Out-of-Distribution games, after pre-training with datasets of differing optimality.

			Suboptimal				Expert			
Game	*RndmAgnt*	*DQN*	CURL	ATC	BC	CQL-D	CURL	ATC	BC	CQL-D
Alien	227.8	2484.5	659.6	469.6	902.8	466.6	574.4	588.8	750.2	693.2
Assault	222.4	1525.2	485.7	698.3	680.1	578.0	501.4	656.0	801.7	680.7
Asterix	210.0	2711.4	409.2	346.7	510.8	542.5	384.2	404.5	487.0	455.7
BeamRider	363.9	5852.4	532.3	542.0	564.5	485.9	536.7	537.7	613.4	520.7
JourneyEscape	−19977.0	−3671.1	−14311.0	−13381.3	−12507.0	−12048.0	−15092.0	−13598.0	−10461.3	−11805.0
Pong	−20.7	16.6	−7.9	−6.3	0.2	−3.6	−2.7	−5.1	3.8	−0.9
Pooyan	371.2	3212.0	755.5	694.2	731.6	476.3	494.8	593.5	837.4	399.8
Riverraid	1338.5	11638.9	2137.9	2424.6	3493.0	2886.6	1982.1	2086.9	2350.0	3590.4
Seaquest	68.4	1600.7	217.7	224.5	378.0	313.8	216.4	187.7	368.9	348.3
Venture	0.0	39.1	7.3	4.7	18.0	2.3	10.7	5.0	12.0	2.3
IQM(DNS)	0.0	1.0	0.1392	0.1428	0.2423	0.1520	0.1169	0.1338	0.2257	0.1873
Optimality Gap(DNS)	0.0	1.0	0.8308	0.8210	0.7201	0.8113	0.8274	0.8231	0.7162	0.7812

Table 27: Offline BC scores on Far-Out-of-Distribution games, after pre-training with datasets of differing optimality.

			Suboptimal				Expert			
Game	*RndmAgnt*	*Rainbow*	CURL	ATC	BC	CQL-D	CURL	ATC	BC	CQL-D
BasicMath	0.1334	3.5700	2.0433	3.3667	2.3867	2.8467	2.3200	3.1400	2.2800	2.3767
HumanCannonball	1.0767	5.5600	1.6767	1.5600	1.4800	1.3667	1.0800	1.4067	1.4100	1.1767
Klax	0.3734	2124.7500	16.0167	19.5167	73.9000	23.6333	13.7000	20.7000	24.4000	20.0667
Othello	−21.3233	−2.0100	0.1000	0.0200	0.1267	0.1067	−0.0033	0.0267	0.1067	0.0467
Surround	−9.9833	−7.8100	−9.2900	−9.1433	−8.9600	−9.1233	−9.1100	−8.9000	−9.3500	−8.9767
IQM(RNS)	0.0	1.0	0.3398	0.4802	0.4055	0.4179	0.3509	0.4919	0.3301	0.3825
Optimality Gap(RNS)	0.0	1.0	0.5968	0.5112	0.5498	0.5478	0.5910	0.5087	0.5997	0.5705

Table 28: Offline BC scores on in-distribution games, across models pre-trained with datasets of different sizes.

			1M				100M			
Game	*RndmAgnt*	*DQN*	CURL	ATC	BC	CQL-D	CURL	ATC	BC	CQL-D
AirRaid	579.2	7479.5	1025.8	1398.0	2772.4	2799.7	1499.1	2232.8	3613.5	4359.5
Amidar	5.8	1207.7	14.5	46.7	43.8	45.1	35.6	39.5	77.9	130.2
Asteroids	719.1	698.4	607.3	730.3	817.8	747.6	801.9	732.3	873.8	994.1
Atlantis	12850.0	853640.0	16861.7	35155.3	9752.0	13624.0	20046.0	12788.0	41484.0	62210.7
BankHeist	14.2	601.8	17.4	29.1	28.9	10.3	21.7	27.9	78.5	20.9
BattleZone	2360.0	17784.8	2840.0	4193.3	3983.3	4683.3	2376.7	9906.7	5296.7	6690.0
Berzerk	123.7	487.5	260.6	317.6	308.4	295.3	310.2	403.1	389.3	403.8
Bowling	23.1	30.1	33.4	37.8	28.0	32.6	16.2	30.4	36.9	33.6
Boxing	0.1	78.0	58.8	32.0	37.4	45.3	25.1	62.6	66.2	65.5
Breakout	1.7	96.2	2.4	5.1	7.9	4.5	5.3	7.7	48.0	46.4
Carnival	700.8	4784.8	593.3	495.7	953.1	890.3	996.1	1043.3	1084.3	2121.8
Centipede	2090.9	2583.0	1919.5	1758.3	2037.0	2167.7	1932.5	1424.1	2277.2	2250.1
ChopperCommand	811.0	2690.6	873.3	1125.7	1138.7	798.3	1331.3	1281.0	1080.3	1271.7
CrazyClimber	10780.5	104568.8	3724.0	7310.7	6088.7	45641.0	11782.3	10822.7	40575.3	94226.7
DemonAttack	152.1	6361.6	116.7	138.0	143.9	177.2	211.0	152.0	902.4	1794.0
DoubleDunk	−18.6	−6.5	−20.2	−18.5	−18.8	−19.5	−18.1	−18.6	−19.1	−19.3
ElevatorAction	4387.0	439.8	76.3	984.7	1101.3	566.0	984.0	224.7	593.7	797.3
Enduro	0.0	628.9	0.0	19.0	37.1	76.3	0.2	3.2	398.0	674.1
FishingDerby	−91.7	0.6	−89.8	−78.3	−81.9	−82.4	−90.0	−80.8	−50.8	−45.9
Freeway	0.0	26.3	18.5	24.0	25.3	12.8	20.5	19.5	24.1	22.0
Frostbite	65.2	367.1	192.1	110.3	147.9	183.6	251.3	361.6	375.2	430.8
Gopher	257.6	5479.9	305.7	762.6	773.3	557.4	338.4	760.9	1025.6	619.3
Gravitar	173.0	330.1	164.8	255.3	201.0	183.7	269.3	253.7	216.7	200.3
Hero	1027.0	17325.4	8936.1	4842.0	4492.4	12995.9	5283.6	6461.2	6178.4	12951.6
IceHockey	−11.2	−5.8	−14.0	−9.0	−8.9	−8.2	−7.8	−9.5	−7.6	−7.3
Jamesbond	29.0	573.3	138.8	340.7	238.5	290.5	200.2	328.0	386.7	310.3
Kangaroo	52.0	11486.0	820.3	1109.7	895.0	934.0	1020.7	1036.0	1844.3	2087.0
Krull	1598.0	6097.6	4906.6	3441.3	3761.7	4603.8	4721.2	3624.8	6278.9	5852.1
KungFuMaster	258.5	23435.4	93.7	4214.3	5342.3	3685.3	2812.3	5568.7	9100.3	10273.0
MontezumaRevenge	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
MsPacman	307.3	3402.4	654.5	1141.8	1150.5	1047.0	828.7	782.2	1956.9	2043.5
NameThisGame	2292.3	7278.6	4146.3	4754.8	5508.2	5520.7	4453.8	4714.2	6042.8	6760.6
Phoenix	761.4	4996.6	1007.0	1299.9	1300.8	1030.0	1104.3	1564.4	2841.0	2607.2
Pitfall	−229.4	−73.8	−94.7	−169.0	−66.6	−22.9	−11.9	−206.4	−24.3	−23.7
PrivateEye	24.9	−16.0	177.8	−211.7	−358.6	1801.2	−880.4	−407.8	353.4	−530.1
Qbert	163.9	10117.5	274.3	361.8	349.4	319.8	375.5	298.6	318.6	4175.1
RoadRunner	11.5	36925.5	1209.7	4501.3	3177.0	2327.3	3482.3	3730.3	3067.0	4678.0
Robotank	2.2	59.8	6.7	10.0	8.6	8.1	6.6	9.6	13.1	10.7
Skiing	−17098.1	−15824.6	−29970.8	−29471.4	−27043.3	−29970.8	−29007.1	−28387.0	−29041.4	−29970.8
Solaris	1236.3	1436.4	1462.7	1992.1	2207.1	1994.1	2398.4	1835.3	1868.1	1338.5
SpaceInvaders	148.0	1794.2	180.1	270.5	169.5	278.0	153.2	167.5	280.9	582.4
StarGunner	664.0	42165.2	1598.0	2191.0	2034.0	2540.0	1349.0	5014.3	4431.0	3237.3
Tennis	−23.8	−1.5	−5.0	−16.7	−18.7	−14.0	−17.7	−13.0	−19.4	−4.5
TimePilot	3568.0	3654.4	3123.3	2877.0	2729.0	3313.7	2826.7	2661.0	3201.7	2927.7
Tutankham	11.4	103.8	25.3	18.9	3.8	24.7	19.1	23.3	20.3	17.3
UpNDown	533.4	8488.3	3498.4	2263.2	3205.7	3221.5	1980.7	2807.9	3319.1	3530.5
VideoPinball	16256.9	63406.1	16.9	954.9	2125.9	950.2	142.6	74.3	1153.4	6997.2
WizardOfWor	563.5	2065.8	415.7	707.0	655.0	375.0	568.3	531.7	369.7	341.0
YarsRevenge	3092.9	23909.4	10913.3	10005.9	10029.0	9772.2	8952.1	10161.5	13605.0	11366.6
Zaxxon	32.5	4538.6	2961.0	3352.7	2909.7	3202.7	1801.3	3134.7	3213.0	3865.3
IQM(DNS)	0.0	1.0	0.0688	0.1312	0.1405	0.1805	0.1104	0.1590	0.3092	0.3942
Optimality Gap(DNS)	0.0	1.0	1.2978	1.2950	1.3461	1.1784	1.5977	1.3689	0.9819	1.2247

Table 29: Offline BC scores on Near-Out-of-Distribution games across models pre-trained with datasets of different sizes.

			1M				100M			
Game	*RndmAgnt*	*DQN*	CURL	ATC	BC	CQL-D	CURL	ATC	BC	CQL-D
Alien	227.8	2484.5	463.6	584.2	603.9	500.9	537.0	729.0	639.0	530.5
Assault	222.4	1525.2	414.0	663.5	551.3	674.7	575.8	696.1	818.8	635.6
Asterix	210.0	2711.4	275.8	349.0	451.8	412.7	399.8	395.8	550.0	355.7
BeamRider	363.9	5852.4	542.3	592.4	525.4	468.9	519.3	549.3	572.3	451.4
JourneyEscape	−19977.0	−3671.1	−13211.3	−15709.7	−12576.7	−14429.0	−15281.0	−12775.0	−11893.7	−12395.7
Pong	−20.7	16.6	−9.6	−0.2	3.0	−1.8	−9.8	−8.2	5.5	−19.3
Pooyan	371.2	3212.0	357.7	563.7	813.1	280.1	536.8	507.3	992.3	360.2
Riverraid	1338.5	11638.9	1762.4	2458.8	2585.8	2980.1	2520.4	2004.3	3874.0	3226.0
Seaquest	68.4	1600.7	196.7	263.1	349.4	503.3	161.5	212.8	350.3	383.7
Venture	0.0	39.1	0.7	1.3	1.3	13.0	18.7	0.0	11.7	52.0
IQM(DNS)	0.0	1.0	0.0771	0.1237	0.1643	0.1871	0.1262	0.1352	0.2439	0.1610
Optimality Gap(DNS)	0.0	1.0	0.8840	0.8258	0.7870	0.7842	0.8290	0.8323	0.7042	0.7809

Table 30: Offline BC scores on Far-Out-of-Distribution games, across models pre-trained with datasets of different sizes.

			1M				100M			
Game	*RndmAgnt*	*Rainbow*	CURL	ATC	BC	CQL-D	CURL	ATC	BC	CQL-D
BasicMath	0.1334	3.5700	1.9300	3.4233	2.6400	2.5633	3.2167	3.3333	2.0967	3.3600
HumanCannonball	1.0767	5.5600	0.9200	2.8300	1.3667	1.4100	1.5367	1.6533	1.3833	1.4867
Klax	0.3734	2124.7500	5.0167	16.4667	23.0667	12.2667	12.5667	23.2167	51.0667	23.6833
Othello	−21.3233	−2.0100	0.0000	0.1667	−0.1000	−0.0533	−0.2800	0.3867	−0.0800	0.1867
Surround	−9.9833	−7.8100	−9.4967	−9.1200	−8.9233	−9.1800	−9.1967	−8.9833	−8.8833	−8.9367
IQM(RNS)	0.0	1.0	0.2498	0.5819	0.4273	0.3837	0.4539	0.5093	0.3819	0.5040
Optimality Gap(RNS)	0.0	1.0	0.6572	0.4494	0.5415	0.5687	0.5265	0.4958	0.5661	0.4981

Table 31: Offline BC scores of a smaller backbone(ResNet-18) on In-Distribution games.

			ResNet-18					
Game	*RndmAgnt*	*DQN*	Random	E2E	CURL	ATC	BC	CQL-D
AirRaid	579.2	7479.5	1550.9	4671.4	1100.7	1305.3	3180.2	3306.7
Amidar	5.8	1207.7	21.3	70.5	18.8	34.1	68.3	56.1
Asteroids	719.1	698.4	872.3	792.7	612.5	670.8	777.7	731.1
Atlantis	12850.0	853640.0	10249.0	133756.0	16177.0	17939.3	12289.7	16609.7
BankHeist	14.2	601.8	11.7	67.9	13.9	27.5	29.3	34.1
BattleZone	2360.0	17784.8	4216.7	930.0	5183.3	5280.0	4543.3	5643.3
Berzerk	123.7	487.5	251.9	364.5	291.8	373.5	340.1	348.5
Bowling	23.1	30.1	0.0	0.0	23.7	23.9	27.9	28.4
Boxing	0.1	78.0	24.7	54.3	39.8	31.6	63.4	65.5
Breakout	1.7	96.2	0.9	38.8	4.3	6.3	14.4	15.2
Carnival	700.8	4784.8	616.8	3077.2	735.3	1172.6	954.7	1081.6
Centipede	2090.9	2583.0	1808.4	2038.9	1833.0	1809.4	2243.5	2235.3
ChopperCommand	811.0	2690.6	867.0	1085.0	1161.7	1181.0	1163.7	1021.3
CrazyClimber	10780.5	104568.8	3715.0	68936.7	1226.7	16585.0	2238.0	3030.7
DemonAttack	152.1	6361.6	122.4	633.8	113.6	141.6	224.6	255.4
DoubleDunk	−18.6	−6.5	−20.3	−16.5	−19.2	−19.0	−19.2	−18.6
ElevatorAction	4387.0	439.8	79.7	230.0	1397.7	697.7	341.7	216.7
Enduro	0.0	628.9	1.4	230.7	0.0	10.6	107.4	113.9
FishingDerby	−91.7	0.6	−86.3	−66.7	−87.2	−80.2	−78.6	−74.9
Freeway	0.0	26.3	21.4	21.0	18.8	13.8	24.6	23.9
Frostbite	65.2	367.1	100.8	85.4	197.6	110.1	459.1	454.0
Gopher	257.6	5479.9	23.1	2166.1	329.5	413.3	487.7	602.1
Gravitar	173.0	330.1	89.2	195.8	187.8	229.0	211.3	165.5
Hero	1027.0	17325.4	3551.2	9714.4	8684.1	4570.2	6966.6	6800.8
IceHockey	−11.2	−5.8	−7.0	−3.3	−9.7	−8.7	−9.8	−9.2
Jamesbond	29.0	573.3	286.7	258.3	97.3	334.2	244.7	237.8
Kangaroo	52.0	11486.0	599.0	1327.0	793.7	1146.0	905.3	897.7
Krull	1598.0	6097.6	3162.1	5491.9	2033.4	3889.0	4994.1	4620.7
KungFuMaster	258.5	23435.4	146.0	7363.7	1433.7	1938.7	3357.7	3525.3
MontezumaRevenge	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
MsPacman	307.3	3402.4	294.5	683.3	563.9	691.7	1132.6	1108.3
NameThisGame	2292.3	7278.6	2713.0	6102.3	4173.7	3792.7	5193.2	5632.0
Phoenix	761.4	4996.6	440.9	1275.8	892.9	1163.2	1665.1	1504.0
Pitfall	−229.4	−73.8	−64.3	−89.5	−50.2	−190.6	−316.0	−93.1
PrivateEye	24.9	−16.0	−383.7	−347.4	342.3	−459.5	544.8	917.2
Qbert	163.9	10117.5	194.8	1506.0	214.4	449.4	390.8	347.2
RoadRunner	11.5	36925.5	3222.3	5673.7	1644.7	3784.3	5600.3	6063.7
Robotank	2.2	59.8	11.1	8.4	6.8	8.5	10.2	10.3
Skiing	−17098.1	−15824.6	−29732.9	−29812.3	−29970.8	−29838.1	−29953.4	−29875.5
Solaris	1236.3	1436.4	1195.2	1865.3	1955.7	2194.7	2014.3	1932.9
SpaceInvaders	148.0	1794.2	234.9	306.1	231.0	177.1	167.4	254.9
StarGunner	664.0	42165.2	901.0	7849.0	1062.0	2009.0	2460.7	2302.0
Tennis	−23.8	−1.5	−0.4	−14.3	−21.3	−14.1	−14.0	−7.1
TimePilot	3568.0	3654.4	2888.3	3301.0	3498.3	2679.3	3311.3	2590.7
Tutankham	11.4	103.8	28.2	24.8	13.5	15.9	9.5	10.3
UpNDown	533.4	8488.3	1050.1	3957.1	1962.5	2040.4	2987.8	2785.0
VideoPinball	16256.9	63406.1	634.1	1193.7	17.4	822.9	1498.8	786.8
WizardOfWor	563.5	2065.8	227.7	437.7	635.7	472.3	479.0	565.3
YarsRevenge	3092.9	23909.4	8732.5	13978.4	9780.6	10600.0	11365.6	10441.5
Zaxxon	32.5	4538.6	1454.3	3646.0	2314.7	2266.7	2672.7	3345.7
IQM(DNS)	0.0	1.0	0.0316	0.2514	0.0702	0.1061	0.1671	0.1837
Optimality Gap(DNS)	0.0	1.0	1.4950	1.2427	1.2298	1.5015	1.0698	1.1814

Table 32: Offline BC scores of a smaller backbone(ResNet-18) on Near-Out-of-Distribution games.

			ResNet-18					
Game	*RndmAgnt*	*DQN*	Random	E2E	CURL	ATC	BC	CQL-D
Alien	227.8	2484.5	542.8	618.0	402.0	739.7	777.9	711.6
Assault	222.4	1525.2	405.7	1004.2	410.1	641.8	677.4	668.7
Asterix	210.0	2711.4	352.3	436.2	293.3	414.8	443.2	406.7
BeamRider	363.9	5852.4	594.5	730.8	542.7	487.9	468.0	467.1
JourneyEscape	−19977.0	−3671.1	−8892.3	−5435.0	−15142.3	−16542.3	−13947.7	−12616.0
Pong	−20.7	16.6	−20.9	−21.0	−19.1	−2.1	4.1	2.6
Pooyan	371.2	3212.0	55.9	381.7	733.4	946.5	293.1	634.6
Riverraid	1338.5	11638.9	575.5	2437.1	1859.3	2276.7	2633.3	2649.2
Seaquest	68.4	1600.7	209.0	641.1	193.4	176.6	299.5	226.1
Venture	0.0	39.1	30.7	13.3	2.0	20.0	0.7	0.0
IQM(DNS)	0.0	1.0	0.0565	0.1557	0.0647	0.1590	0.1655	0.1532
Optimality Gap(DNS)	0.0	1.0	0.8707	0.7368	0.9063	0.7923	0.7994	0.7946

Table 33: Offline BC scores of a smaller backbone(ResNet-18) on Far-Out-of-Distribution games.

			ResNet-18				
Game	*RndmAgnt*	*Rainbow*	Random	CURL	ATC	BC	CQL-D
BasicMath	0.1334	3.5700	1.1500	2.7667	3.6667	2.9133	3.3833
HumanCannonball	1.0767	5.5600	0.9900	1.1367	2.6800	1.1400	1.6867
Klax	0.3734	2124.7500	3.6667	6.9833	27.1667	20.0667	23.3167
Othello	−21.3233	−2.0100	0.0000	0.5667	0.0333	0.0600	0.2733
Surround	−9.9833	−7.8100	−9.8967	−9.0700	−9.0000	−8.7333	−8.8700
IQM(RNS)	0.0	1.0	0.1134	0.4029	0.6110	0.4790	0.5313
Optimality Gap(RNS)	0.0	1.0	0.7364	0.5594	0.4409	0.5185	0.4790