

A Continual Offline Reinforcement Learning Benchmark for Navigation Tasks

Anthony Kobanda^{1,2}, Odalric-Ambrym Maillard¹, Rémy Portelas²

¹ Inria, Univ. Lille, CNRS, Centrale Lille, UMR 9198-CRISAL, F-59000 Lille, France

² Ubisoft La Forge, Bordeaux, France

{anthony.kobanda,remy.portelas}@ubisoft.com , odalric.maillard@inria.fr

Abstract—Autonomous agents operating in domains such as robotics or video game simulations must adapt to changing tasks without forgetting about the previous ones. This process called Continual Reinforcement Learning poses non-trivial difficulties, from preventing catastrophic forgetting to ensuring the scalability of the approaches considered. Building on recent advances, we introduce a benchmark providing a suite of video-game navigation scenarios, thus filling a gap in the literature and capturing key challenges : catastrophic forgetting, task adaptation, and memory efficiency. We define a set of various tasks and datasets, evaluation protocols, and metrics to assess the performance of algorithms, including state-of-the-art baselines. Our benchmark is designed not only to foster reproducible research and to accelerate progress in continual reinforcement learning for gaming, but also to provide a reproducible framework for production pipelines – helping practitioners to identify and to apply effective approaches.

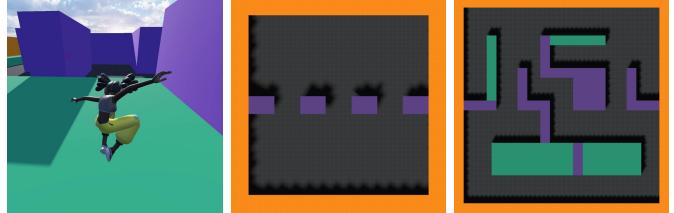
<https://sites.google.com/view/continual-nav-bench>

Index Terms—Deep Learning, Offline Reinforcement Learning, Continual Learning, Bots, Benchmarks, Human Generated Data.

I. INTRODUCTION

The human learning process is inherently cumulative : we constantly master skills through new experiences without discarding what we already know, instead we leverage previously acquired knowledge. In contrast, most classical **Reinforcement Learning (RL)** [1], [2] methods depend on extensive interactions and are prone to catastrophic forgetting when faced with a series of different challenges. In this context, **Continual Reinforcement Learning (CRL)** [3], [4] aim to bridge this gap by enabling agents to learn incrementally, thus emulating the adaptive, lifelong learning process of humans. This incremental paradigm not only helps mitigate forgetting but also facilitates the transfer of learned skills.

An interesting subset of tasks within the CRL framework is **Goal-Conditioned RL (GCRL)** [5], [6] allowing policies to be conditioned to specific target states, which is particularly valuable in navigation tasks. Modern video game environments, with their evolving layouts require efficiently learning and adapting bots. Moreover, in production pipelines, rigorous evaluation of these methods is critical to ensure that algorithms are effective and computationally efficient. These attributes make GCRL particularly promising for applications that demand adaptive and robust decision-making strategies.



(A) A human playing. (B) 20 m × 20 m map. (C) 60 m × 60 m map.

Fig. 1: Visualization of an environment and a human playing : (a) **Human Player Interaction.** A third-person perspective of a human playing within a large maze ; (b) **Overview of a Small Map.** A top-down view of a small maze, showcasing the simple layout ; (c) **Overview of a Large Map.** A top-down view of a larger maze, highlighting a complex layout requiring better planning and actions.

Offline RL (ORL) [7], [8] offers a compelling framework by leveraging collections of pre-recorded gameplay or simulation data, reducing the cost and risks of live data acquisition. However, despite ORL’s progress, standardized benchmarks for CRL in video game-inspired navigation tasks remain scarce. Current benchmarks [9], [10] rarely address the flexibility and memory requirements of production pipelines.

In this work, we introduce **Continual NavBench** : a novel benchmark for Continual Offline Reinforcement Learning focused on navigation in virtual game worlds. Continual NavBench bridges research and production needs by offering a suite of scenarios – including Godot engine [11] tailor made environments with human-generated data – for both research and real-world applications. Our benchmark includes :

- **Standardized Offline Datasets :** Sourced from 10 hours of human gameplay (around 3000 episodes) in diverse Godot mazes, to capture different navigation strategies.
- **Evaluation Protocols and Metrics :** We consider standard continual metrics – the *Performance*, the *Backward Transfer*, the *Forward Transfer*, the *Relative Model Size*, the *Training and Inference Costs* – to assess a model performance and efficiency, ensuring relevance for video game production.
- **Reproducible Benchmarking Tools :** Our open - source framework provides code, datasets, and evaluation protocols to ensure reproducible experiments and facilitate comparisons of methods for streamlined integration into research pipelines.

II. RELATED WORK

Reinforcement Learning (RL) has achieved notable successes in domains such as robotics and video games [1], [2]. Benchmarks, such as Atari 2600 [12], Procgen [13], and VizDoom [14], evaluate RL agents in dynamic and rich environments. However, classical methods rely suffer from catastrophic forgetting when agents are faced with sequential tasks. **Continual Reinforcement Learning (CRL)** [3], [15] address these issues by enabling agents to learn incrementally, better approximating the lifelong learning exhibited by humans.

In particular, **Goal-Conditioned RL (GCRL)** [5], [6] shows promise for navigation tasks by conditioning policies on desired goal states. Despite these advances, most CRL research has focused on the online setting. In contrast, **Offline RL** [7], [16] leverages fixed datasets of pre-recorded gameplay or simulation traces, bypassing the risks and costs of live data collection. Nevertheless, standardized benchmarks for Offline CRL in video game-inspired navigation tasks remain scarce. Existing benchmarks, such as Continual World [17] and CORA [18], primarily target online learning and rarely address production requirements and constraints like inference speed, memory efficiency, and model scalability.

CRL methods address the challenge of sequential task learning through various strategies : **Replay-Based** methods mitigate forgetting by storing past experiences and replaying them [19], [20], however they can be impractical due to significant storage requirements and potential privacy concerns, especially in industrial applications ; **Regularization** techniques such as Elastic Weight Consolidation (EWC) [21], or simply L2 regularization [22] introduce constraints on parameter updates, but they may struggle with highly diverse tasks. **Architectural** strategies, such as Progressive Neural Networks (PNNs) [23] or [24], modify a networks architecture to accommodate new tasks, nevertheless they can become memory-intensive and may lack scalability.

III. PRELIMINARIES

In this section, we present the theoretical background of CRL. These foundational concepts provide the basis for understanding and comparing the performance of different approaches.

A. Reinforcement Learning

We consider a Markov Decision Process (MDP) framework $\mathcal{M} = (\mathcal{S}, \mathcal{A}, \mathcal{P}_{\mathcal{S}}, \mathcal{P}_{\mathcal{S}}^{(0)}, \mathcal{R}, \gamma)$, which provides a formal framework for RL, where \mathcal{S} is a state space, \mathcal{A} an action space, $\mathcal{P}_{\mathcal{S}} : \mathcal{S} \times \mathcal{A} \rightarrow \Delta(\mathcal{S})$ a transition function, $\mathcal{P}_{\mathcal{S}}^{(0)} \in \Delta(\mathcal{S})$ an initial distribution over the states, $\mathcal{R} : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \rightarrow \mathbb{R}$ a deterministic reward function, and $\gamma \in]0, 1]$ a discount factor. An agent's behavior follows a policy $\pi_{\theta} : \mathcal{S} \rightarrow \Delta(\mathcal{A})$, parameterized by $\theta \in \Theta$. The objective is to learn optimal parameters $\theta^*_{\mathcal{M}}$, or a set of such parameters, maximizing the expected cumulative reward $J_{\mathcal{M}}(\theta)$ or the success rate $\sigma_{\mathcal{M}}(\theta)$.

B. Offline Goal-Conditioned Reinforcement Learning

We extend the MDP to include a goal space \mathcal{G} , an initial state-goal distribution $\mathcal{P}_{\mathcal{S}, \mathcal{G}}^{(0)}$, a function mapping each state to the goal it represents $\phi : \mathcal{S} \rightarrow \mathcal{G}$, and $d : \mathcal{G} \times \mathcal{G} \rightarrow \mathbb{R}^+$ a distance metric on \mathcal{G} . Then, the policy $\pi_{\theta} : \mathcal{S} \times \mathcal{G} \rightarrow \Delta(\mathcal{A})$ and the reward function $\mathcal{R} : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \times \mathcal{G} \rightarrow \mathbb{R}$ now depend on a goal $g \in \mathcal{G}$. We consider sparse rewards allocated when agents reach a set goal : $\mathcal{R}(s_t, a_t, s_{t+1}, g) = \mathbb{1}(\|d(\phi(s_{t+1}), g) \leq \epsilon\|)$, and given a pre-collected dataset $\mathcal{D} = \{(s, a, r, s', g)\}$ [7], [25], the policy is then optimized to reach specified goals.

C. Continual Reinforcement Learning

In CRL, an agent follows a sequence of tasks, or stream, $\mathcal{T} = (T_1, \dots, T_N)$, with $T_k = (\mathcal{M}_k, \mathcal{D}_k)$. We note as θ_k the parameters of a policy after learning on the k -th task. As the agent learns new skills, it must preserve (to prevent forgetting) and enhance (to encourage backward transfer) its performance on tasks already learned, while ideally having a relatively low number of parameters. To quantitatively compare CRL methods, we adopt standard metrics commonly used in the literature [15], [26] : **Performance (PER)** : $\frac{1}{N} \sum_{k=1}^N \sigma_{\mathcal{M}_k}(\theta_N)$; **Backward Transfer (BWT)** : $\frac{1}{N} \sum_{k=1}^N (\sigma_{\mathcal{M}_k}(\theta_N) - \sigma_{\mathcal{M}_k}(\theta_k))$; **Forward Transfer (FWT)** : $\frac{1}{N} \sum_{k=1}^N (\sigma_{\mathcal{M}_k}(\theta_k) - \sigma_{\mathcal{M}_k}(\tilde{\theta}_k))$; **Relative Model Size (MEM)** : $\frac{|\theta_N|}{|\theta_{\text{ref}}|}$; **Model Inference Cost (INF)**;

Model Training Cost (TRN). PER measures the success rate across all tasks. BWT indicates how learning a new task affects previous ones, while FWT measures the transfer of knowledge, using $\tilde{\theta}_k$ as randomly initialized parameters. MEM compares the memory load of the model to a reference one associated to parameters θ_{ref} . INF is the duration required for policy inference across tasks, and TRN is the overall training duration.

IV. ENVIRONMENTS, TASKS & DATASETS

A. Navigation Environments & Mazes

We propose video game-inspired navigation environments, which offer easy to use 3D maze configurations via our provided open-source code¹. Our benchmark comprises two families of mazes designed to evaluate CRL agents : **SimpleTown**, a collection of 8 relatively simple mazes ($20 \text{ m} \times 20 \text{ m}$), and **AmazeVille**, a more complex set of 8 mazes ($60 \text{ m} \times 60 \text{ m}$). Table I and Table II display the available outputs and inputs.

| Action | Dimensions | Type | Human Controls / Keys |
|----------------|------------|-------|-----------------------|
| Move Forwards | 1 | bool | Z |
| Move Backwards | 1 | bool | S |
| Move Left | 1 | bool | Q |
| Move Right | 1 | bool | D |
| Jump | 1 | bool | Space |
| Turn | 1 | float | Mouse |

TABLE I: **Available Action Keys.** This table details the control commands available to the agents in our environments. Actions include directional movement, running, jumping, and turning. The *Keys* indicate the human input used during data collection.

¹GitHub Repository Link.

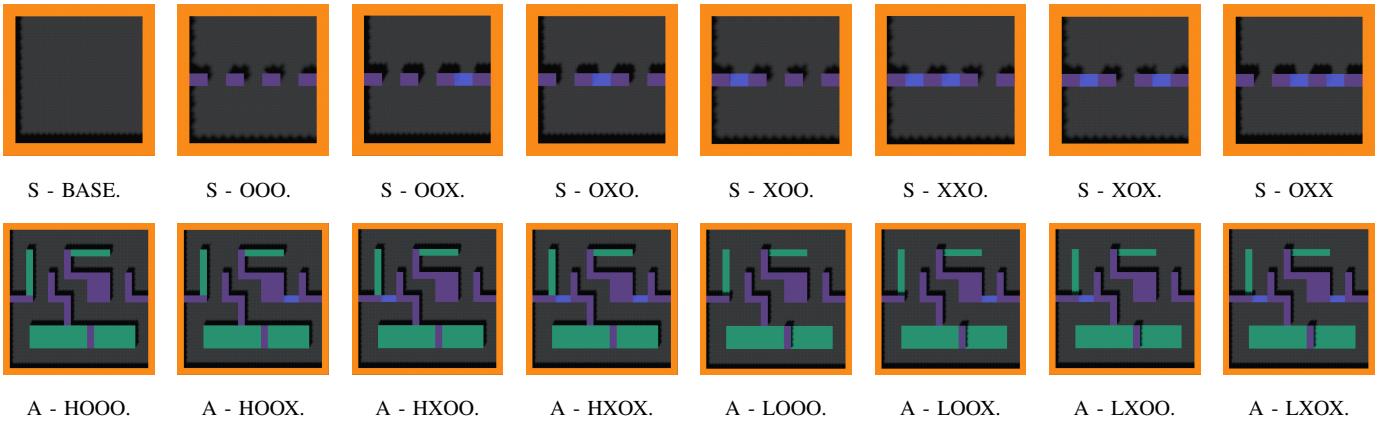


Fig. 2: **SimpleTown** mazes are relatively simple, with a size of 20×20 meters. The starting positions are randomly sampled on one side, and the goal positions are on the other side ; **AmazeVille** mazes, of 60×60 meters, are more challenging. They have a finite set of start and goal positions, with two subsets of maps : some with high blocks, i.e. not jumpable obstacles ; others with low blocks, i.e. jumpable ones. The naming convention of the tasks use a prefix for the maze family (S for SimpleTown, and A for AmazeVille), the subsequent characters encode key layout features where “O” and “X” are respectively open and closed doors, and “H” and “L” denote high and low blocks.

| Feature | Size | Type | Feature | Size | Type |
|---------------|-------------|-------|--------------|----------|-------|
| Position | 3 | float | Orientation | 3 | float |
| Goal Position | 3 | float | Velocity | 3 | float |
| RGB Image | (3, 64, 64) | float | Depth Image | (11, 11) | float |
| Floor Contact | 1 | bool | Wall Contact | 1 | bool |
| Goal Contact | 1 | bool | Timestep | 1 | int |
| Up Direction | 3 | float | - | - | - |

TABLE II: Available observation features. Position data are in m , while the Orientation is in rad , and the Velocity in $m \cdot s^{-1}$. The RGB Images offer a visualization of the agent’s viewpoint, and the Depth Image is generated from 11×11 raycasts thrown towards nearest obstacles. Although our experiments only use positions and depth images, we also provide the pixel-based observations with all datasets.

B. Task Streams

Our benchmark defines task streams with different maze configurations. We distinguish two types of streams :

Random Streams – In AmazeVille we have :

Stream AR1 : A-LOOX \rightarrow A-HXOX \rightarrow A-LXXO \rightarrow A-HXOX ;
Stream AR2 : A-HXOO \rightarrow A-HOOX \rightarrow A-LOOX \rightarrow A-LXXO.

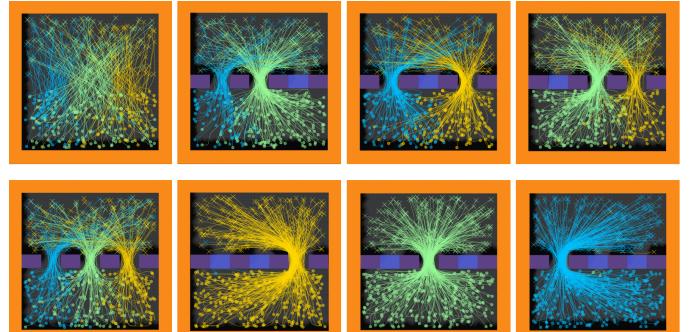
Topological Streams – Designed with common changes in maze structure. We have in both environments :

Stream ATI : A-HOOX \rightarrow A-HXOX \rightarrow A-HXOX \rightarrow A-HOOX ;
Stream AT2 : A-LOOO \rightarrow A-LOOO \rightarrow A-LXXO \rightarrow A-LXXO ;
Stream ST1 : S-BASE \rightarrow S-OXO \rightarrow S-BASE \rightarrow S-OOX ;
Stream ST2 : S-BASE \rightarrow S-OXX \rightarrow S-XOO \rightarrow S-OXX.

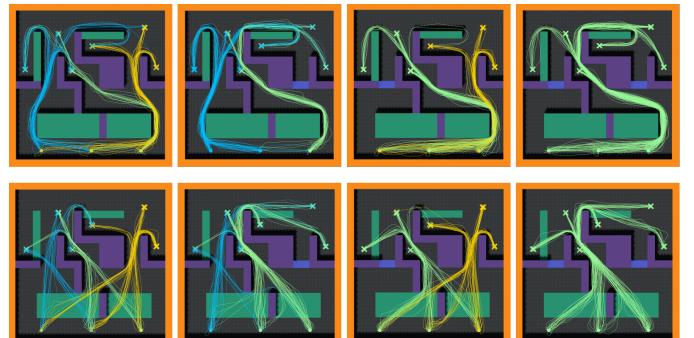
Notably, some tasks reoccur within a stream, offering an opportunity to evaluate whether an algorithm can recognize and reuse previously learned strategies without increasing model size or compromising performance. We encourage future users to tailor task streams to address specific research or production goals. Additionally, these environments are well-suited for developing and benchmarking GCRL algorithms.

C. Human Generated Datasets

We gathered 2800 trajectories over 10 hours of human plays, with 250 episodes per maze in SimpleTown and 100 episodes in AmazeVille. These datasets capture strategies that are valuable for training bots to exhibit human-like behavior.



(A) Trajectories generated within SimpleTown mazes.



(B) Trajectories generated within AmazeVille mazes.

Fig. 3: Visualization of the generated trajectories

To further support the research community, we open source the data alongside our benchmark framework. This ensures that other practitioners can readily access, reproduce, and build upon our work, fostering collaborative advancements in CRL, and more generally GCRL, for video game bots development.

V. BASELINE DETAILS

A. Hierarchical Imitation Learning as Backbone Algorithm

We consider Hierarchical Imitation Learning (HGCBC) [27] to address GCRL by decomposing tasks into more manageable ones. Given a MDP $\mathcal{M} = (\mathcal{S}, \mathcal{A}, \mathcal{P}_{\mathcal{S}}, \mathcal{P}_{\mathcal{S}, \mathcal{G}}^{(0)}, \mathcal{R}, \gamma, \mathcal{G}, \phi, d)$ and a pre-collected dataset $\mathcal{D} = \{(s_t^i, a_t^i, r_t^i, s_{t+1}^i, g^i)\}$, the end-to-end policy $\pi_\theta(s, g)$ is split into two components :

- **A High-Level Policy** $\pi_{\theta_h}^h(s, g)$, which selects intermediate sub-goals that serve as waypoints towards the final goal ;
- **A Low-Level Policy** $\pi_{\theta_l}^l(s, g)$, which generates actions to move the agent towards the provided sub-goal.

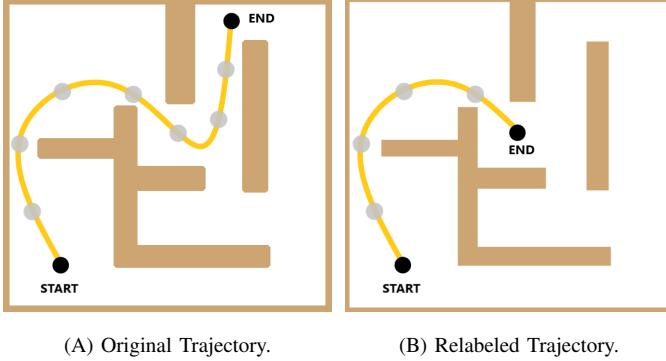
The high-level policy is optimized by minimizing :

$$\mathcal{L}_D^h(\theta_h) = \mathbb{E}_{(s_t^i, s_{t+k}^i, g^i) \sim \mathcal{D}} \left[-\log(\pi_{\theta_h}^h(\phi(s_{t+k}^i) | s_t^i, g^i)) \right],$$

while the low-level policy minimizes :

$$\mathcal{L}_D^l(\theta_l) = \mathbb{E}_{(s_t^i, a_t^i, s_{t+1}^i, s_{t+k}^i, g^i) \sim \mathcal{D}} \left[-\log(\pi_{\theta_l}^l(a_t^i | s_t^i, \phi(s_{t+k}^i))) \right].$$

Hence, the hyperparameter $k \in \mathbb{N}^*$ defines the temporal gap.



(A) Original Trajectory.

(B) Relabeled Trajectory.

Fig. 4: **Hindsight Experience Replay (HER) Illustration.**

Figure 4 illustrates *Hindsight Experience Replay (HER)* as a data augmentation strategy, leveraging the hierarchical structure. By relabeling transitions with alternative sub-goals, HER can enrich the dataset, especially in low or sparse data regimes.

B. Continual Reinforcement Learning Baselines

We evaluate a range of baseline methods designed to learn from a task stream. These methods span several categories :

Naive Methods :

Naive Learning Strategy or From Scratch (SC1 & SCN) : in SC1, a single policy is learned from the latest dataset and then applied unchanged to all tasks. In SCN, a new policy is trained for each task, improving performance at the cost of a memory load. **Freeze Strategy (FRZ) :** here, a single policy is trained on the first task and applied without modification to all subsequent tasks. **Finetuning Strategy (FT1 & FTN) :** the Finetuning Strategy involves adapting a policy learned from the initial task to each subsequent task, either by continuously updating a single policy (FT1) or by copying and then updating the policy for each new task (FTN), allowing for a better adaptation.

Replay-Based Method :

Experience Replay Strategy (RPL) : this method aggregates all task datasets, up to the current one, into a single replay buffer and trains a single policy on the combined data, ensuring continuous exposure to earlier experiences. Such a process may allow to avoid catastrophic forgetting, and eventually build common skills and knowledge accross tasks.

Weight Regularization Methods :

Elastic Weight Consolidation (EWC) [21] : this strategy mitigates catastrophic forgetting by selectively slowing down learning on certain weights based on their importance to previously learned tasks. This importance is measured by the Fisher Information Matrix, which quantifies the sensitivity of the output function to changes in the parameters. EWC introduces a quadratic penalty, constraining the parameters close to their values from previous tasks, where the strength of the penalty is proportional to each parameter's importance.

L2-Regularization Finetuning (L2) [22] : This strategy also mitigates catastrophic forgetting by adding an L2 penalty to the loss, discouraging large weight changes during training. This helps preserve knowledge from previous tasks by promoting stability in the learned representations. As with EWC, L2-regularization struggles in CRL for navigation tasks, especially when actions or dynamics change drastically. The method limits the network's flexibility by forcing small weight updates, making it difficult to adapt to tasks that require distinct actions for similar states, which is critical in evolving environments.

Architectural Methods :

Progressive Neural Networks (PNN) [23] : This framework introduce a new column layer for each task, freezing previous weights to preserve knowledge. Lateral connections allow feature transfer, leveraging prior experience while avoiding interference. By integrating previously learned features, PNN builds richer compositional representations, ensuring that prior knowledge is preserved and used throughout the feature hierarchy. Nevertheless the model grows with each task, limiting scalability for long streams tasks or limited memory contexts. **Hierarchical Subspace of Policies (HiSPO) [24]**: this strategy partitions the model into distinct subspaces of neural networks [28] for high-level and low-level control. Initially, anchor parameters for both subspaces are trained on the first task to form a foundational knowledge reservoir. For each subsequent task, new anchors are introduced and optimized using the task's dataset. The algorithm then samples anchor weight configurations to explore the previous subspace, and it selects the configuration that minimizes the loss. Finally, by comparing the loss of the extended subspace with that of the previous configuration, under a threshold ϵ , HiSPO either prunes the new anchor or retains it. This process integrates new information while preserving and reusing prior knowledge, ensuring scalability and adaptability across tasks.

C. Implementation Details

We base our configuration on the approach described in [29]. In all our experiments, we use Residual MLPs [30] with Layer Normalization [31] for the hidden layers. All networks use 3×256 hidden units, GELU activations, and are initialized with a variance scaling strategy [32] (with a scale of 0.1).

We set the batch size to 64, the learning rate to 3×10^{-4} , the number of gradient steps to 1×10^5 , and the sub-goal distance to $k = 10$ for AmazeVille and $k = 5$ for SimpleTown. HER sampling temperatures are respectively set to 100.0 and 15.0.

For both EWC and L2, we considered five regularization strengths $\lambda \in \{1 \times 10^{-2}, 1 \times 10^{-1}, 1, 1 \times 10^1, 1 \times 10^2\}$ and selected the best-performing model for each stream. Similarly, for HiSPO, we experimented with acceptance thresholds $\epsilon_h, \epsilon_l \in \{5 \times 10^{-2}, 1 \times 10^{-1}\}$ and weights for the similarity loss as $\lambda_h, \lambda_l \in \{5 \times 10^{-2}, 1 \times 10^{-1}\}$, for functional diversity.

We used a compute cluster featuring Intel(R) Xeon(R) CPU E5-1650 and Intel Cascade Lake 6248 processors. For most models, 4 cores were sufficient, but due to PNN's growing memory requirements, we allocated 6 cores for its experiments.

VI. BENCHMARK STUDY

A. Offline Reinforcement Learning

We first compare the hierarchical backbone (HGCBC) with a Goal-Conditioned Behavioral Cloning (GCBC) [5] baseline, which is the non-hierarchical supervised baseline approach.

| Task | Success Rate | | Episode Length | |
|----------|-------------------|-------------------|---------------------|--------------------|
| | GCBC | HGCBC | GCBC | HGCBC |
| S - BASE | 98.9 ± 0.8 | 97.6 ± 1.7 | 50.9 ± 2.3 | 50.8 ± 2.2 |
| S - OOO | 97.6 ± 1.2 | 97.1 ± 2.1 | 55.8 ± 1.4 | 55.0 ± 1.7 |
| S - OOX | 95.5 ± 1.5 | 96.6 ± 2.0 | 59.4 ± 1.6 | 57.4 ± 1.8 |
| S - OXO | 91.3 ± 3.0 | 97.5 ± 1.3 | 60.9 ± 2.5 | 55.8 ± 1.8 |
| S - XOO | 93.8 ± 0.9 | 97.6 ± 1.3 | 59.5 ± 1.4 | 56.1 ± 1.2 |
| S - X XO | 92.4 ± 3.0 | 97.4 ± 1.2 | 68.8 ± 1.8 | 66.1 ± 1.3 |
| S - XOX | 94.2 ± 2.5 | 98.5 ± 0.9 | 60.3 ± 1.5 | 57.2 ± 1.5 |
| S - OXX | 92.0 ± 1.8 | 95.8 ± 1.8 | 67.8 ± 2.0 | 65.8 ± 2.6 |
| A - HOOO | 80.2 ± 3.2 | 91.3 ± 3.3 | 193.4 ± 6.8 | 173.5 ± 4.3 |
| A - HOOX | 73.2 ± 7.5 | 78.1 ± 5.2 | 222.3 ± 7.8 | 215.0 ± 9.7 |
| A - HXOO | 92.7 ± 2.1 | 97.9 ± 2.3 | 185.7 ± 5.3 | 181.4 ± 5.0 |
| A - HXOX | 79.4 ± 2.8 | 85.4 ± 3.4 | 239.4 ± 2.0 | 234.2 ± 3.1 |
| A - LOOO | 92.7 ± 3.8 | 88.5 ± 3.0 | 138.0 ± 5.0 | 142.0 ± 4.5 |
| A - LOOX | 85.5 ± 7.9 | 82.7 ± 5.7 | 171.8 ± 12.6 | 173.0 ± 9.7 |
| A - LXOO | 89.2 ± 3.5 | 90.1 ± 3.8 | 145.2 ± 6.0 | 141.3 ± 7.2 |
| A - LXOX | 93.4 ± 2.2 | 93.4 ± 2.2 | 160.4 ± 3.8 | 161.8 ± 3.9 |

TABLE III: Performance of GCBC and HGCBC (10 seeds).

HGCBC consistently outperforms GCBC in both success rate and episode length across most mazes. In some of the SimpleTown ones, the differences between HGCBC and GCBC are negligible, as these tasks are easier to learn and provide limited room for improvement with a hierarchical approach.

B. Continual Offline Reinforcement Learning

1) *Performance* : Table IV presents the performance metric of each CRL method on the task streams. To a certain extent, PER simultaneously tells about generalization and forgetting.

| CRL Methods | AR1 | AR2 | AT1 | AT2 | ST1 | ST2 |
|-------------|-------------|-------------|-------------|-------------|-------------|-------------|
| SC1 | 75.8 | 32.4 | 46.8 | 74.4 | 90.6 | 74.8 |
| SCN | 84.3 | 84.0 | 77.2 | 87.8 | 96.9 | 96.2 |
| FT1 | 72.1 | 35.3 | 56.3 | 85.2 | 94.1 | 81.2 |
| FTN | 83.8 | 86.9 | 81.3 | 91.0 | 97.5 | 96.8 |
| FRZ | 33.4 | 65.1 | 46.8 | 70.1 | 84.6 | 64.1 |
| RPL | 86.7 | 75.5 | 61.2 | 90.2 | 97.8 | 91.8 |
| EWC | 72.4 | 39.6 | 56.5 | 89.6 | 94.8 | 83.2 |
| L2 | 72.0 | 20.3 | 48.0 | 82.6 | 93.9 | 81.1 |
| PNN | 90.1 | 94.4 | 85.0 | 93.4 | 98.8 | 98.1 |
| HiSPO | 85.7 | 88.3 | 75.2 | 83.5 | 97.0 | 97.1 |

TABLE IV: Performance of the CRL Methods (3 seeds). PNN attains the highest scores overall, reflecting its strong ability to retain and reuse knowledge. In simpler settings, methods like FTN can achieve competitive performance, but they lag behind more adaptive approaches (RPL or HiSPO) in complex tasks.

2) *Backward & Forward Transfer* : Figure 5 presents the BWT (5.A) and FWT (5.B) metrics, thus focusing on the generalization and forgetting phenomena across the streams.

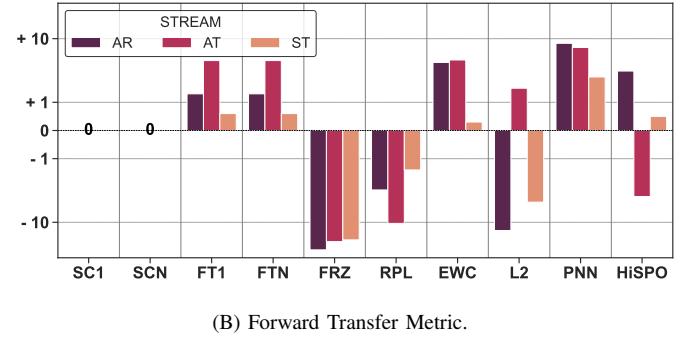
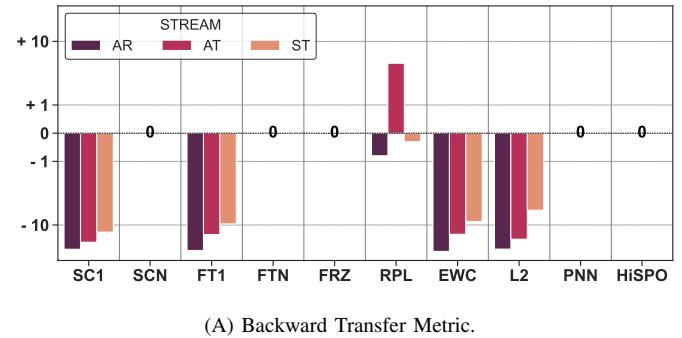


Fig. 5: Backward and Forward Transfer Metrics. Architectural or separate-policy methods (PNN, HiSPO, SCN, FTN) typically balance old-task retention and new-task adaptation, whereas single-policy or regularized methods (SC1, FRZ, EWC, L2) risk higher forgetting or reduced forward transfer in complex navigation scenarios.

3) *Relative Model Size* : On one hand, Figure 6 shows how much each model grows relative to a reference network (*SC1*) at inference time. This metric is crucial for production settings, with storage or runtime overheads to minimize. On the other hand, Figure 7 shows the relative size during training. This metric is relevant to track the scalability of a learning method.

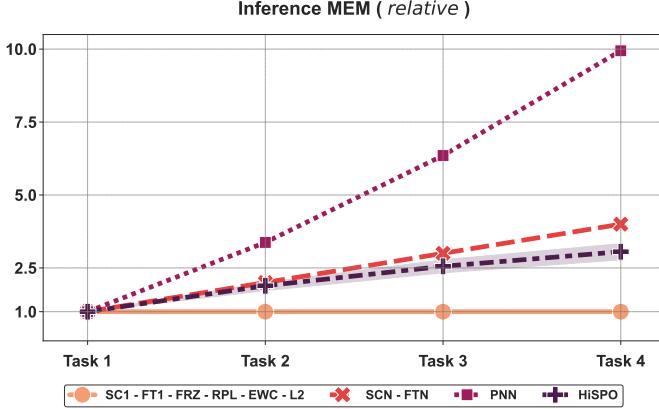


Fig. 6: **Average Relative MEM Metric at Inference.** PNN exhibits the fastest growth, as each new task introduces an additional column and layer interconnections to all previously learned features maps. SCN and FTN grow linearly, allocating a new policy for each new task. In contrast, SC1, FRZ, EWC, L2, and RPL keep a single model, so their size remains constant for any stream of tasks. Finally, HiSPO increases subspace parameters only when necessary, leading to sublinear or moderate growth relative to purely additive approaches.

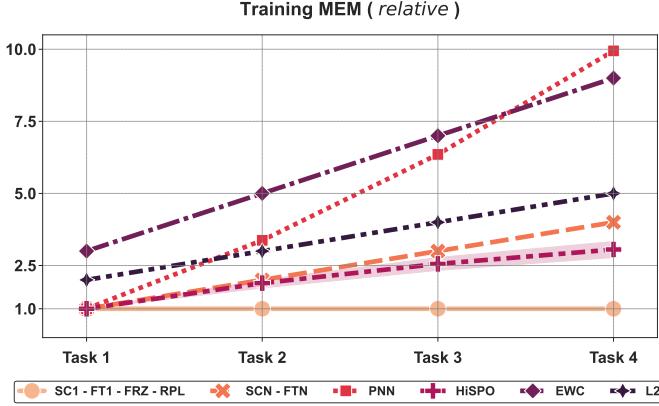


Fig. 7: **Average Relative MEM Metric at Training.** Most methods preserve a single working model, so their training overhead remains similar to the inference one. However, EWC and L2 store additional parameter snapshots (old weights for both, and Fisher Information for EWC) to constrain updates and reduce forgetting. This extra storage can raise memory demands when many tasks are encountered.

These results highlight a core trade-off : methods like *PNN* or *SCN & FTN* can improve continual learning but incur greater memory overhead, while all the single-model approaches (such as *SC1, FRZ, RPL*) remain lightweight but risk higher forgetting. Balancing memory cost and adaptation is thus key to deploying continual offline reinforcement learning in practical pipelines.

4) *Training & Inference Costs* : Continual Reinforcement Learning methods differ not only in memory overhead but also in computational demands during both training and inference. Figure 9 reports TRN, the average models training cost metric (in minutes), and Figure 8 shows INF, the average models inference cost metric (in ms).

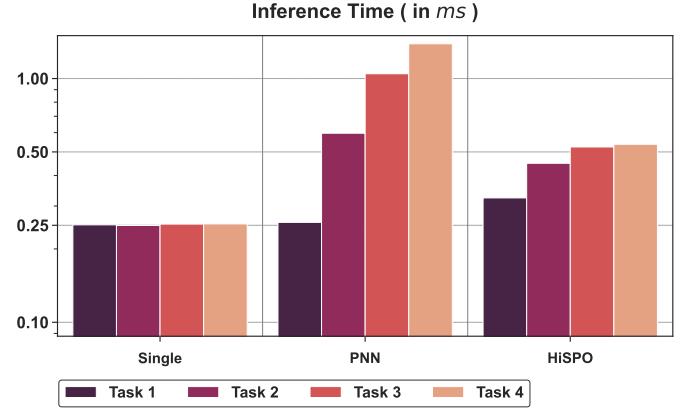


Fig. 8: **Average INF (model inference cost duration in ms).** We measure inference time averaging on 100,000 forward passes on CPU with a batch size of 64, using the AR1 stream (seed 100). At inference, single-model approaches (*SC1-N, FT1-N, FRZ, RPL, EWC*, and *L2*) incur minimal overhead, requiring a single forward pass per step. In contrast architectural strategies (*PNN* and *HiSPO*) may demand additional computations (such as selecting the right sub-policy or combining anchors), leading to higher inference times.

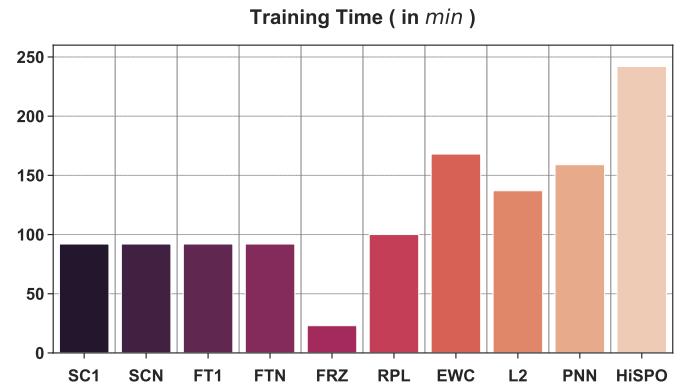


Fig. 9: **Average TRN (training duration in min).** Single-model approaches train quickly by updating a single parameter set, while RPL is slightly slower due to repeated dataset sampling. EWC and L2 add overhead by storing parameters snapshots. Architectural methods also take longer to train, as PNN shows non-linear growth by adding new columns and lateral connections per task, and HiSPO introduces new anchors and also computes cosine-similarity regularization weights.

Overall, these findings show that single-model methods maintain lighter overhead during both training and inference but can suffer from higher forgetting. Conversely, multi-policy or subspace-based strategies better preserve knowledge at the cost of increased computational demands, a crucial consideration in large-scale or time-sensitive applications.

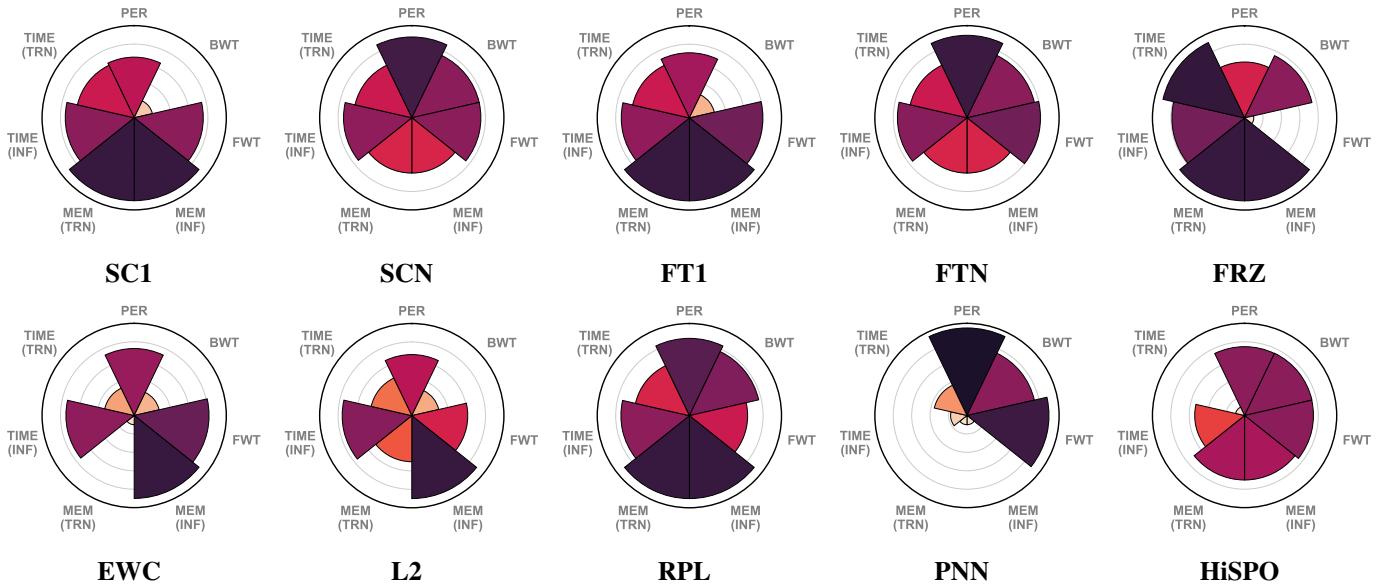


Fig. 10: **Radar Plots of all Metrics for each CRL Method considered.** Each sub-figure shows a *radar chart* summarizing the measured metrics in Section VI, as defined in Section III, for each CRL method : **SC1**, **SCN** (Naive strategies), **FT1**, **FTN** (Finetuning), **FRZ** (Freeze), **RPL** (Replay-based), **EWC**, **L2** (Regularization), **PNN** (Progressive Neural Networks), and **HiSPO** (Hierarchical Subspace of Policies). For each metric, higher radial values indicate better performance, with color intensity reflecting the normalized score. This visualization highlights the trade-offs among memory usage, computational overhead, performance, and transfer capabilities across different approaches.

5) *Benchmark Summary* : Our evaluation reveals that, overall, CRL generally outperform single-task baselines in terms of maintaining performance across sequential tasks. Metrics such as PER, BWT, and FWT indicate that approaches incorporating architectural expansion or hierarchical decomposition tend to retain and even enhance previously learned skills better.

PNN and HiSPO demonstrate good performances, with PNN achieving the highest PER however with a high MEM, TRN, and INF costs. In contrast, simpler approaches such as SC1 and FRZ, though computationally efficient, suffer from significant forgetting, as indicated by negative BWT. RPL, EWC and L2 offer moderate gains but may incur additional TRN overhead.

In summary, our benchmark illustrates that while no single method excels across all metrics, architectural approaches (PNN, HiSPO, and FTN) present promising strategies for Offline CRL in navigation tasks. These findings underscore the trade-offs between memory cost, computational demand, and learning performance, which is an essential consideration for real-world productions and research in the field.

VII. DISCUSSION & FUTURE WORK

We introduced **Continual NavBench**, a new benchmark designed for *Continual Offline Reinforcement Learning* in video game-inspired navigation tasks. Our experiments show that while approaches like *PNN* and *HiSPO* are good at preserving and reusing skills across tasks, they may incur either significant memory or computational costs. In contrast, single-model or regularized methods (*SC1*, *FRZ*, *EWC*, *L2*) remain lighter in resource consumption but often exhibit higher forgetting. These findings highlight the inherent trade-offs between scalability, performance, and efficiency in Offline CRL.

Looking ahead, we believe there are several promising paths for future research. First, *enlarging the environment suite* (e.g., multi-level mazes, dynamic obstacles, or different agents) could further stress the agent’s ability to adapt in complex scenarios. Second, *exploring hybrid approaches* (e.g., combining subspace methods with replay or regularization) may yield more balanced solutions that preserve prior knowledge without increasing memory or training costs. Third, incorporating *domain transfer*, from offline learning to online finetuning, to shed light on how efficiently each method generalizes beyond using datasets.

Ultimately, we hope **Continual NavBench** spurs more rigorous and reproducible evaluations of continual learning strategies for navigation and beyond. By offering diverse tasks, open-source tools, and well-defined metrics, our benchmark lowers the barrier for future work to compare methods, refine architectures, and propose novel solutions. We believe that systematically addressing memory, training, and inference constraints will be pivotal in bringing *lifelong autonomy* to both realistic research prototypes and real-world production pipelines.

REFERENCES

- [1] R. S. Sutton and A. G. Barto, *Reinforcement Learning : An Introduction, Second Edition*. MIT Press, 2018.
- [2] K. Arulkumaran, M. P. Deisenroth, M. Brundage, and A. A. Bharath, “Deep reinforcement learning : A brief survey,” *IEEE Signal Processing Magazine*, 2017.
- [3] K. Khetarpal, M. Riemer, I. Rish, and D. Precup, “Towards continual reinforcement learning : A review and perspectives,” *Journal of Artificial Intelligence Research*, 2022.
- [4] L. Wang, X. Zhang, H. Su, and J. Zhu, “A comprehensive survey of continual learning : Theory, method and application,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2024.
- [5] Y. Ding, C. Florensa, M. Phielipp, and P. Abbeel, “Goal-conditioned imitation learning,” *Advances in Neural Information Processing Systems*, 2019.
- [6] M. Liu, M. Zhu, and W. Zhang, “Goal-conditioned reinforcement learning : Problems and solutions,” *IJCAI*, 2022.
- [7] S. Levine, A. Kumar, G. Tucker, and J. Fu, “Offline reinforcement learning : Tutorial, review, and perspectives on open problems,” 2020.
- [8] A. Kobanda, C. Valliappan, J. Romoff, and L. Denoyer, “Learning computational efficient bots with costly features,” in *2023 IEEE Conference on Games (CoG)*, IEEE, 2023.
- [9] V. Volz and B. Naujoks, “Towards game-playing ai benchmarks via performance reporting standards,” in *2020 IEEE Conference on Games (CoG)*, IEEE, 2020.
- [10] G. Macaluso, A. Sestini, and A. D. Bagdanov, “A benchmark environment for offline reinforcement learning in racing games,” in *2024 IEEE Conference on Games (CoG)*, pp. 1–4, IEEE, 2024.
- [11] Godot, “Godot game engine,” 2020.
- [12] V. Kurin, S. Nowozin, K. Hofmann, L. Beyer, and B. Leibe, “The atari grand challenge dataset,” *arXiv preprint arXiv:1705.10998*, 2017.
- [13] K. Cobbe, C. Hesse, J. Hilton, and J. Schulman, “Leveraging procedural generation to benchmark reinforcement learning,” in *International conference on machine learning*, pp. 2048–2056, PMLR, 2020.
- [14] M. Kempka, M. Wydmuch, G. Runc, J. Toczek, and W. Jaśkowski, “Vizdoom: A doom-based ai research platform for visual reinforcement learning,” in *2016 IEEE conference on computational intelligence and games (CIG)*, pp. 1–8, IEEE, 2016.
- [15] N. Díaz-Rodríguez, V. Lomonaco, D. Filliat, and D. Maltoni, “Don’t forget, there is more than forgetting : new metrics for continual learning,” 2018.
- [16] J. Liu, W. Li, X. Yue, S. Zhang, C. Chen, and Z. Wang, “Continual offline reinforcement learning via diffusion-based dual generative replay,” *arxiv*, 2024.
- [17] M. Wolczyk, M. Zajac, R. Pascanu, L. Kucinski, and P. Milos, “Continual world : A robotic benchmark for continual reinforcement learning,” 2021.
- [18] S. Powers, E. Xing, E. Kolve, R. Mottaghi, and A. Gupta, “Cora : Benchmarks, baselines, and metrics as a platform for continual reinforcement learning agents,” in *Conference on Lifelong Learning Agents*, PMLR, 2022.
- [19] D. Rolnick, A. Ahuja, J. Schwarz, T. Lillicrap, and G. Wayne, “Experience replay for continual learning,” *Advances in neural information processing systems*, 2019.
- [20] Y. Huang, K. Xie, H. Bharadhwaj, and F. Shkurti, “Continual model-based reinforcement learning with hypernetworks,” in *International Conference on Robotics and Automation (ICRA)*, IEEE, 2021.
- [21] J. Kirkpatrick, R. Pascanu, N. Rabinowitz, J. Veness, G. Desjardins, A. A. Rusu, K. Milan, J. Quan, T. Ramalho, A. Grabska-Barwinska, *et al.*, “Overcoming catastrophic forgetting in neural networks,” *Proceedings of the national academy of sciences*, 2017.
- [22] S. Kumar, H. Marklund, and B. V. Roy, “Maintaining plasticity in continual learning via regenerative regularization,” 2023.
- [23] A. A. Rusu, N. C. Rabinowitz, G. Desjardins, H. Soyer, J. Kirkpatrick, K. Kavukcuoglu, R. Pascanu, and R. Hadsell, “Progressive neural networks,” *arXiv preprint*, 2016.
- [24] A. Kobanda, R. Portelas, O.-A. Maillard, and L. Denoyer, “Hierarchical subspaces of policies for continual offline reinforcement learning,” *arXiv preprint arXiv:2412.14865*, 2024.
- [25] E. Biré, A. Kobanda, L. Denoyer, and R. Portelas, “Efficient active imitation learning with random network distillation,” *arXiv preprint arXiv:2411.01894*, 2024.
- [26] R. Kemker, M. McClure, A. Abitino, T. Hayes, and C. Kanan, “Measuring catastrophic forgetting in neural networks,” in *Proceedings of the AAAI conference on artificial intelligence*, 2018.
- [27] A. Gupta, V. Kumar, C. Lynch, S. Levine, and K. Hausman, “Relay policy learning: Solving long-horizon tasks via imitation and reinforcement learning,” *arXiv preprint arXiv:1910.11956*, 2019.
- [28] M. Wortsman, M. C. Horton, C. Guestrin, A. Farhadi, and M. Rastegari, “Learning neural network subspaces,” in *International Conference on Machine Learning*, 2021.
- [29] D. Ghosh, C. A. Bhateja, and S. Levine, “Reinforcement learning from passive data via latent intentions,” in *International Conference on Machine Learning*, PMLR, 2023.
- [30] K. He, X. Zhang, S. Ren, and J. Sun, “Deep residual learning for image recognition,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016.
- [31] J. L. Ba, J. R. Kiros, and G. E. Hinton, “Layer normalization,” *arXiv preprint arXiv:1607.06450*, 2016.
- [32] M. V. Narkhede, P. P. Bartakke, and M. S. Sutaone, “A review on weight initialization strategies for neural networks,” *Artificial intelligence review*, vol. 55, no. 1, pp. 291–322, 2022.