ASH: Agents that Self-Hone via Embodied Learning

Benjamin Schneider [1] **Xavier Schneider** [1] **Victor Zhong** [1] **Sun Sun** [2]

Abstract

Long-horizon embodied tasks remain a fundamental challenge in AI, as current methods rely on hand-engineered rewards or action-labeled demonstrations, neither of which scales. We introduce **ASH**, an agentic system that learns an embodied policy from unlabeled, noisy internet video, without reward shaping or expert annotation. ASH follows a self-improvement loop; when it gets stuck, ASH learns an Inverse Dynamics Model (IDM) from its own trajectories, and uses its IDM to extract supervision from relevant internet video. ASH uses unsupervised learning to identify key moments from large-scale internet video and retains them as long-term memory — allowing it to tackle long-horizon problems. We evaluate ASH on two complementary environments demanding multi-hour planning: *Pokémon Emerald*, a turn-based RPG, and *The Legend of Zelda: The Minish Cap*, a real-time action-adventure game. In both games, behavioral cloning, retrieval-augmented and zero-shot foundation-model baselines plateau, while ASH sustains progression across our 8-hour evaluation. ASH reaches an average of $11.2/12$ milestones in Pokémon Emerald and $9.9/12$ in Legend of Zelda, while the strongest baseline gets stuck in both environments at an average of $6.5/12$ and $6.0/12$ milestones, respectively. We demonstrate that self-improving agents are a scalable recipe for long-horizon embodied learning.

1. Introduction

Developing agents capable of mastering long-horizon, open-ended tasks remains a fundamental challenge in artificial intelligence (Huang et al., 2026; Hu et al., 2024). Unlike agents for short-horizon planning, long-horizon agents must navigate complex, open-ended worlds over millions of steps. This requires multi-hour planning, fine-grained motor control, and the ability to adapt to new visual contexts.

Despite progress in embodied learning, current approaches struggle in these long-horizon settings. Behavioral Cloning (BC) methods (Torabi et al., 2018a) typically learn a policy from massive offline datasets. At this scale, assembling high-quality data is a significant challenge, requiring domain knowledge to curate data and extensive human annotation (Baker et al., 2022). Moreover, much of the available data may be irrelevant to the agent's current objectives, increasing both training compute and susceptibility to noise. Long-horizon tasks exacerbate these challenges (Ross & Bagnell, 2010): a single trajectory may span hours, making it intractable to model dependencies across the full sequence. As a result, BC policies typically condition on short-term memory and struggle when a current decision depends on old information (Lifshitz et al., 2023).

While Reinforcement Learning (RL) has demonstrated success in complex environments, it remains heavily dependent on dense reward engineering (Pleines et al., 2025; Hafner et al., 2025; Baker et al., 2022). In a domain spanning tens of hours with diverse sub-tasks, manually designing a reward function for every intermediate milestone is brittle, domain-specific, and labor intensive. Like BC, modeling dependencies across hour-long sequences using RL is intractable. Instead, a common approach is to use short-term memory combined with "next sub-objective" conditioning (Hafner et al., 2025; Baker et al., 2022), which amounts to a human manually laying out a path for the agent to follow. These approaches have difficulty learning skills and knowledge beyond their human-engineered objectives, making them unsuitable for open-ended environments.

To address these limitations, we introduce **ASH**, an agentic system that follows a **dynamic bootstrap** loop. When ASH gets stuck, it searches for helpful demonstrations on the internet, and learns from them. ASH uses its current trajectories to retrieve relevant examples from online video and to update its Inverse Dynamics Model (IDM). ASH uses unsupervised learning to identify key moments from the retrieved video. ASH bootstraps its policy using these key moments as **long-term memory** and the retrieved video demonstrations labeled by the IDM.

[1]Cheriton School of Computer Science, University of Waterloo [2]National Research Council Canada. Correspondence to: Benjamin Schneider <benjamin.schneider@uwaterloo.ca>.

Proceedings of the 43rd International Conference on Machine Learning, Seoul, South Korea. PMLR 306, 2026. Copyright 2026 by the author(s).

Figure 1. ASH self-improves over the course of a multi-hour playthrough by retrieving and learning from relevant internet video. In Pokémon Emerald, ASH learns to battle trainers, catch Pokémon, and heal its team at Pokémon centers. In Zelda, ASH discovers how to wield a sword and apply game mechanics to solve puzzles. These skills compound, allowing ASH to achieve sustained progression.

We evaluate ASH in two complementary long-horizon environments: *Pokémon Emerald*, a turn-based RPG combining strategic battling with open-ended exploration, and *The Legend of Zelda: The Minish Cap*, a real-time action-adventure game that requires puzzle solving and fine-grained motor control. Both require dozens of hours of decision-making across millions of distinct visual states, which makes them useful as long-horizon testbeds. ASH interacts with the games as a human does: by observing the screen and pressing buttons.

In our evaluations, ASH progresses further than behavioral cloning (Baker et al., 2022), retrieval-augmented (Zhu et al., 2024), and zero-shot foundation-model (Qwen Team, 2026) baselines. Baselines plateau on intermediate milestones, while ASH continues to acquire new skills—battling trainers, catching Pokémon, exploring areas, and defeating enemies—across multi-hour horizons. After 8 hours, ASH achieves an average of $11.2/12$ progression milestones in Pokémon Emerald and $9.9/12$ milestones in Legend of Zelda. The strongest baseline gets stuck in both environments, achieving an average of 6.5 and 6.0 milestones, respectively.

ASH's main contributions are as follows: **(1)** We introduce dynamic bootstrapping for long-horizon embodied learning. **(2)** We propose long-term memory created from salient moments in large-scale internet video. **(3)** Through experimentation, we demonstrate that dynamic bootstrapping and long-term memory create a scalable recipe for embodied learning — capable of multi-hour planning, fine-grained motor control, and the ability to adapt to new visual contexts.

2. Related Work

Internet-scale behavioral cloning. A large body of work trains embodied policies via BC on observation-only video at scale. VPT (Baker et al., 2022) creates pseudo-actions for thousands of hours of Minecraft video with an IDM trained on contractor-collected gameplay (Torabi et al., 2018a), and remains the strongest reported Minecraft BC backbone (Li et al., 2025). A parallel *imitation-from-observation* literature recovers actions without an explicit IDM (Torabi et al., 2018b; Edwards et al., 2019; Schmidt & Jiang, 2024; Bruce et al., 2024). ASH departs from these approaches in two ways: it does not rely on a single statically curated training corpus or a fixed action-labeling model. Instead, ASH is an agentic system that uses the trajectories of its agents to update its IDM and retrieve data for policy bootstrapping.

Retrieval-augmented decision-making. Retrieval has been used to give agents relevant context (Lewis et al., 2020; Yao et al., 2022; Shinn et al., 2023), to condition policies on prior experience (Humphreys et al., 2022; Goyal et al., 2022), and to guide planning (Zhang et al., 2023). In these methods, retrieval selects a small number of relevant documents, trajectories, or examples to condition the agent during inference.

ASH uses retrieval differently: it retrieves a task-relevant training set from a much larger internet-scale corpus, then updates its weights using that data.

Self-improvement from model-generated data. A growing line of work bootstraps a learner from data produced by an earlier version of itself: STaR (Zelikman et al., 2022) and ReST (Gulcehre et al., 2023) for reasoning, self-play for games (Anthony et al., 2017; Silver et al., 2017), and self-distillation in robotics (Ahn et al., 2024). ASH fits this template, but its improvement comes from retrieved internet video rather than from an external verifier or reward model, which are often unavailable in open-ended environments (Fan et al., 2022).

Long-horizon control in embodied agents. Long-horizon control requires agents to retain information that may become useful much later. However, full-history conditioning is computationally intractable, while short fixed-length memory is efficient but can lose information needed for planning (Lifshitz et al., 2023). Embodied BC and RL are dominated by bounded-memory policies, including recurrent policies that compress history into a fixed-size state and transformers that condition on a fixed context window (Chen et al., 2021; Baker et al., 2022; Hafner et al., 2025). Memory-augmented language architectures (Wu et al., 2022; Bulatov et al., 2022; Borgeaud et al., 2022) and skill-library agents such as Voyager (Wang et al., 2023) offer alternatives. Rather than being authored by an LLM or hand-engineered, ASH autonomously discovers key moments from internet data, and retains those moments as long-term memory when it encounters them in the environment.

RL in open-ended environments. RL can achieve strong results when dense rewards, self-play objectives, or programmatic verifiers are available (Mnih et al., 2013; Silver et al., 2017; Vinyals et al., 2019; OpenAI et al., 2019; Hafner et al., 2025). However, these assumptions are hard to satisfy in open-ended environments. For example, Pleines et al. (2025) train a PPO agent on Pokémon Red by manually creating a trail of rewards for the agent to follow. They report high failure rates when the agent encounters situations not anticipated by the reward engineer, and must continually adapt their rewards to correct for these failure modes. By contrast, ASH's training signal comes from unlabeled demonstrations, offering a scalable approach to open-ended embodied learning.

Foundation-model agents and scaffolded play. A separate line of work uses pretrained foundation models with extensive scaffolding as agents. Voyager (Wang et al., 2023) uses an LLM and a learned skill library in a Minecraft code-as-action loop, and Karten et al. (2025); ClaudePlaysPokemon (2026) apply pretrained models to Pokémon Emerald with scaffolds that read underlying game memory. These systems demonstrate emergent behavior but rely on privileged state access to make the games interpretable by an LLM (team et al., 2025). Unlike these approaches, ASH offers a path to embodied learning beyond environments where privileged state is available.

Long-horizon games as embodied testbeds. Prior testbeds for long-horizon embodied learning include Minecraft (Baker et al., 2022; Hafner et al., 2025; Wang et al., 2023; Fan et al., 2022), NetHack (Hambro et al., 2023), and Pokémon (Pleines et al., 2025; Karten et al., 2025). We add *The Legend of Zelda: The Minish Cap* as a complement to Pokémon Emerald (Section 4). Zelda contains real-time action and puzzle solving, while still offering long-horizon objectives. This contrasts with the turn-based, strategic nature of Pokémon.

3. ASH

Preliminary. An agent interacts with an environment $\mathcal{E}$ that, given action $a \in \mathcal{A}$, returns observation $o \in \mathcal{O}$. We write *observation-action trajectories* as $\tau = (o_1, a_1, \ldots, o_T, a_T)$ and *observation-only trajectories* as $\tau^{obs} = (o_1, \ldots, o_T)$. A policy specifies a distribution over the agent's next action given the information available to it, written $\pi \sim p_\theta(a_t \mid \cdot)$. We use end-exclusive slicing $A_{i:j} = (A_i, \ldots, A_{j-1})$. A detailed symbol glossary appears in Appendix I.

Behavioral cloning (BC) (Torabi et al., 2018a) trains π via supervised learning using cross-entropy loss on observation-action trajectories. However, scalable observation-only sources such as YouTube (Baker et al., 2022) lack action labels. We use an IDM to supply pseudo-actions from these observation-only trajectories; $\text{IDM}(\tau^{obs}) = (\tilde{a}_1, \tilde{a}_2, \ldots, \tilde{a}_T)$. The IDM is trained via supervised learning on a dataset of observation-action trajectories. We note that training an IDM is strictly easier than training a policy, as the IDM can use future information to predict pseudo-actions (Baker et al., 2022).

We use HDBSCAN (Campello et al., 2013) clustering to discover recurring key moments because (1) the number of clusters (key moments) is unknown a priori, (2) it rejects outliers as noise rather than forcing assignment, and (3) HDBSCAN's approximate-predict supports online assignment without refitting.

Overview. ASH is analogous to a human that plays the game, gets stuck, and finds helpful video on the internet. We deploy N agents under a single shared policy π conditioned on short- and long-term memory, and collect trajectories $\tau_{1\sim N}$. A key-moment classifier $\mathcal{K}(\tau^{obs}, t)$, derived from internet data, is used to determine if the observation at t is a new key moment. When an observation is classified as a new key moment, it gets added to an agent's long-term memory and resets a per-agent stuck-timer. When any

timer exceeds threshold Δ, ASH *dynamically bootstraps*: it uses the agents' trajectories to update the IDM, and to retrieve visually similar internet video. These observation-only videos are used to update $\mathcal{K}$. The IDM then labels the videos with pseudo-actions, producing trajectories that are used to update π. Full ASH pseudo-code can be found in Appendix F.

Algorithm 1 ASH

Inputs: initial policy π, IDM, key-moment classifier $\mathcal{K}$, internet corpus $\mathcal{D}^I$

Step 1: Inference

1: Infer N agents in parallel under shared policy π, conditioning each action on per-agent short-term and long-term memory.
2: On each step, use $\mathcal{K}$ to detect new key moments and append them to that agent's long-term memory.
3: Halt when any agent fails to find a new key moment for Δ steps (*stuck*).

Step 2: Retrieval

4: Embed agent trajectories $\tau_{1\sim N}$ with DINOv2.
5: Retrieve top-k visually similar videos from $\mathcal{D}^I$ per trajectory, union into $\mathcal{D}^R$.

Step 3: Bootstrapping

6: Update $\mathcal{K}$ on $\mathcal{D}^R$ to discover new key moments.
7: Update IDM on $\tau_{1\sim N}$.
8: Update π on $\mathcal{D}^R$, using the IDM for pseudo-actions and $\mathcal{K}$ to construct long-term memory.
9: `GoTo` **Step 1**.

Architecture. Due to the long-horizon setting, a decision can hinge on information from many steps ago. ASH resolves this with a *dual memory*: a short-term memory of the w_s most recent observation-action pairs for reactive control, and a long-term memory of observations ρ that are the w_l most recent *key moments*. The policy is therefore $\pi \sim p_\theta(a_t \mid \tau_{t-w_s:t},\, \rho)$. ASH uses an image tokenizer ϕ (SigLIP (Zhai et al., 2023)), an IDM with VPT's bidirectional-attention architecture (Baker et al., 2022), and the causal transformer block architecture from Yang et al. (2025), repeated 28 times, for $\mathcal{T}$.

Training. We train the policy π via gradient descent on observation-only trajectories τ^{obs} of internet data. Each τ^{obs} is divided into non-overlapping samples of length w_s, denoted $\tau^{obs}_{t:t+w_s} = (o_t, o_{t+1}, \ldots, o_{t+w_s-1})$.

$$\tilde{\boldsymbol{A}} = \mathrm{IDM}(\tau^{obs}_{t:t+w_s}), \qquad \boldsymbol{T}^s, \boldsymbol{T}^\ell = \phi(\tau^{obs}_{t:t+w_s}, \rho)$$

$$\boldsymbol{H} = \mathcal{T}\Big(\Big[\boldsymbol{T}^\ell,\ \boldsymbol{T}^s_1, \tilde{\boldsymbol{A}}_1,\ \ldots,\ \boldsymbol{T}^s_{w_s}, \tilde{\boldsymbol{A}}_{w_s}\Big]\Big)$$

$$\mathcal{L} = -\sum_{j=0}^{w_s} \tilde{\boldsymbol{A}}_j^\top \log \mathrm{softmax}(\boldsymbol{W}_o \boldsymbol{H}_{w_l+2j+1})\,.$$

First, we use the IDM to create pseudo-actions for each observation in the sample $\tau^{obs}_{t:t+w_s}$, which are encoded as one-hot vectors. These actions are used for both teacher-forcing and output labels while training π. ϕ is used to tokenize short-term ($\tau^{obs}_{t:t+w_s}$) and long-term (ρ) observations to create $\boldsymbol{T}^s$ and $\boldsymbol{T}^\ell$, respectively. The resulting tokens are concatenated into a unified sequence where the long-term memories $\boldsymbol{T}^\ell$ act as a prefix to the interleaved short-term observations and pseudo-actions. This combined sequence is passed through the causal transformer $\mathcal{T}$, allowing the model to attend to important memories while predicting local actions. We use linear layers to project $\boldsymbol{T}^s, \boldsymbol{T}^\ell$ and $\tilde{\boldsymbol{A}}$ into the hidden dimension of $\mathcal{T}$. The output layer $\boldsymbol{W}_o$ maps the hidden output at the observation token position of the transformer $\boldsymbol{H}$ to the action classes. The policy π is trained using $\mathcal{L}$, the cross-entropy loss between predictions generated by ASH and pseudo-actions generated by our IDM.

Algorithm 2 Inference

Require: Policy π, trajectories $\tau_{1\sim N}$, memory banks $\mathcal{M}_{1\sim N}$, classifier $\mathcal{K}$, memory lengths w_s, w_l, threshold Δ, environments $\mathcal{E}_{1\sim N}$

```
C_{1~N} ← 0
while max_n C_n < Δ do
    for n = 1 to N do
        ρ ← M_{n,−w_l:}
        a ~ π(· | τ_{n,−w_s:}, ρ)
        o ← E_n(a)
        τ_n ← τ_n ∪ {(o, a)}
        if K(τ_n, |τ_n|) then
            M_n ← M_n ∪ {o}
            C_n ← 0
        end if
        C_n ← C_n + 1
    end for
end while
return τ_{1~N}, M_{1~N}
```

Inference. Algorithm 2 describes ASH's inference loop. Each agent n maintains a per-agent stuck timer C_n, initialized to 0, that counts how many consecutive steps have elapsed since the agent last observed a new key moment. At each step, agent n reads the w_l most recent key moments from its memory bank $\mathcal{M}_n$ as ρ (long-memory), combines them with the last w_s observation-action pairs in τ_n (short-memory), and samples an action from π. After executing the action, the agent appends the resulting observation-action pair to τ_n. The classifier $\mathcal{K}$ determines whether the observation is a new key moment. If so, the observation is appended to $\mathcal{M}_n$ and the timer C_n is reset to zero; otherwise C_n is incremented. The loop runs until any agent's timer exceeds Δ, at which point ASH treats that agent as stuck, halts inference for all agents, and bootstraps their shared policy. After the bootstrap, agents resume inference from their current observation.

Retrieval. When an agent gets stuck, ASH needs training

data targeted at its *current roadblock*. To find this data, ASH retrieves a dataset $\mathcal{D}^R$ from internet video $\mathcal{D}^I$ by selecting videos that are similar to the agent's current observations.

Algorithm 3 Retrieval

Require: Trajectories $\tau_{1\sim N}$, indexed corpus $\mathcal{D}^I$, window w_r, number of videos retrieved k

$\mathcal{D}^R \leftarrow \emptyset$
for $n = 1$ to N **do**
  $\boldsymbol{Q} \leftarrow \textsc{Embed}(\tau_n)$
  scores $\leftarrow \emptyset$
  for $\tau^{\text{obs}} \in \mathcal{D}^I$ **do**
    $\boldsymbol{E} \leftarrow \textsc{Embed}(\tau^{\text{obs}})$
    $e \leftarrow \textsc{Rows}(\boldsymbol{E})$
    $\boldsymbol{S} \leftarrow \boldsymbol{Q}\boldsymbol{E}^\top$
    $z^\star \leftarrow -\infty$
    for $t \in \{1, \ldots, e - w_r + 1\}$ **do**
      $s \leftarrow \boldsymbol{S}_{:,\, t:t+w_r}$
      $z \leftarrow 0$
      for $j = 1$ to w_r **do**
        $r^\star \leftarrow \arg\max_r s_{r,j}$
        $z \leftarrow z + s_{r^\star,j}$
        $s_{r^\star,:} \leftarrow -\infty$
      end for
      $z^\star \leftarrow \max(z^\star, z)$
    end for
    scores $\leftarrow$ scores $\cup \{(\tau^{\text{obs}}, z^\star)\}$
  end for
  $\mathcal{D}^R \leftarrow \mathcal{D}^R \cup \{\text{top-k(scores)}\}$
end for
return $\mathcal{D}^R$

We index our internet dataset using DINOv2 (Oquab et al., 2024) by embedding video frames sampled at 2 second intervals. Each internet video is represented as a sequence of L2-normalized visual embeddings, forming a matrix $\boldsymbol{E} \in \mathbb{R}^{e\times d}$, where e denotes the number of sampled frames and d is the embedding dimension. For each agent trajectory τ_n, we similarly compute a trajectory embedding matrix $\boldsymbol{Q} \in \mathbb{R}^{q\times d}$. For each internet video τ^{obs}, we compute a similarity matrix $\boldsymbol{S} = \boldsymbol{Q}\boldsymbol{E}^\top$. Retrieval searches for the most similar contiguous segment of length w_r in each internet video. We score each temporal window $\boldsymbol{S}_{:,\, t:t+w_r}$ by greedy one-to-one matching — each internet video frame in the window claims its highest-similarity trajectory frame, and each trajectory frame is consumed at most once. An internet video's score is its best temporal window score. We retrieve the top-k videos per agent trajectory and union them into $\mathcal{D}^R$. This matching scheme is compared to simpler baselines in Section 4.4.

Discovering key moments. $\mathcal{K}$ is built by applying the HDBSCAN (Campello et al., 2013) clustering algorithm over the embeddings of $\mathcal{D}^R$, and all previously clustered embeddings. We further filter the resulting clusters by requiring each cluster contain observations from a minimum of $c_{\min}$ distinct trajectories in $\mathcal{D}^R$. This ensures that the clusters capture key visual moments that are commonly occurring between many trajectories. Each individual cluster therefore aims to be a *single key moment* in the environment. Details on cluster quality can be found in Appendix B.

Key moments to long-term memory. $\mathcal{K}(\tau^{obs}, t)$ is a binary classifier: it returns *true* if the observation at timestep t is a *new* key moment—one whose corresponding cluster has not yet been matched by any earlier observation in τ^{obs}—and *false* otherwise. This is done by embedding τ_t^{obs} using DINOv2 (Oquab et al., 2024) and applying HDBSCAN's approximate predict function to determine if it belongs to a cluster. These new key moments serve a dual purpose: firstly, they are appended to an agent's memory bank $\mathcal{M}_n$, and the last w_l entries are used as long-term memory for the agent. Secondly, they are used for our dynamic bootstrapping algorithm, where a bootstrap is triggered when an agent does not encounter a new key moment for Δ steps.

Algorithm 4 Bootstrapping

Require: Agent trajectories $\tau_{1\sim N}$, retrieved dataset $\mathcal{D}^R$, memory lengths w_s, w_l

Update $\mathcal{K}$ using $\mathcal{D}^R$
Update IDM on $\tau_{1\sim N}$
$\mathcal{D}^\pi \leftarrow \emptyset$
for $\tau^{obs} \in \mathcal{D}^R$ **do**
  for $t = 1$ to $|\tau^{obs}| - w_s + 1$ step w_s **do**
    $\tilde{\boldsymbol{A}} \leftarrow \text{IDM}(\tau^{obs}_{t:t+w_s})$
    $\bar{\rho} \leftarrow [\, \tau_i^{obs} \,:\, i < t,\ \mathcal{K}(\tau^{obs}, i)\,]$
    $\mathcal{D}^\pi \leftarrow \mathcal{D}^\pi \cup \{(\tau^{obs}_{t:t+w_s},\ \tilde{\boldsymbol{A}},\ \bar{\rho}_{-w_l:})\}$
  end for
end for
Update π on $\mathcal{D}^\pi$
return π, IDM, $\mathcal{K}$

Bootstrapping. Each bootstrap (Algorithm 4) updates $\mathcal{K}$, the IDM and π. First, $\mathcal{K}$ is updated using the newly retrieved $\mathcal{D}^R$. For IDM data, ASH uses the observations from $\tau_{1\sim N}$ as IDM inputs, and the corresponding actions as labels. To ensure coverage of new environmental dynamics, we supplement the IDM training dataset with samples obtained from the environment using a random policy. ASH uses $\mathcal{K}$ and the IDM to update the policy π. Since τ^{obs} is internet video, it only contains a sequence of observations, not actions. Therefore, we use the IDM to create pseudo-actions for each sample from τ^{obs}. Then, $\mathcal{K}(\tau^{obs}, k)$ is used to construct memories (ρ) from timesteps $k < t$—before the start of the sample—which are used for training π's long memory. The resulting sample, consisting of w_s observation-action pairs and w_l long-term memories, is added to π's training dataset $\mathcal{D}^\pi$. π is then trained using $\mathcal{D}^\pi$. For both π and the IDM, we use 10% of samples as a held-out set to prevent overfitting. Once π has been bootstrapped, ASH resumes agent inference.

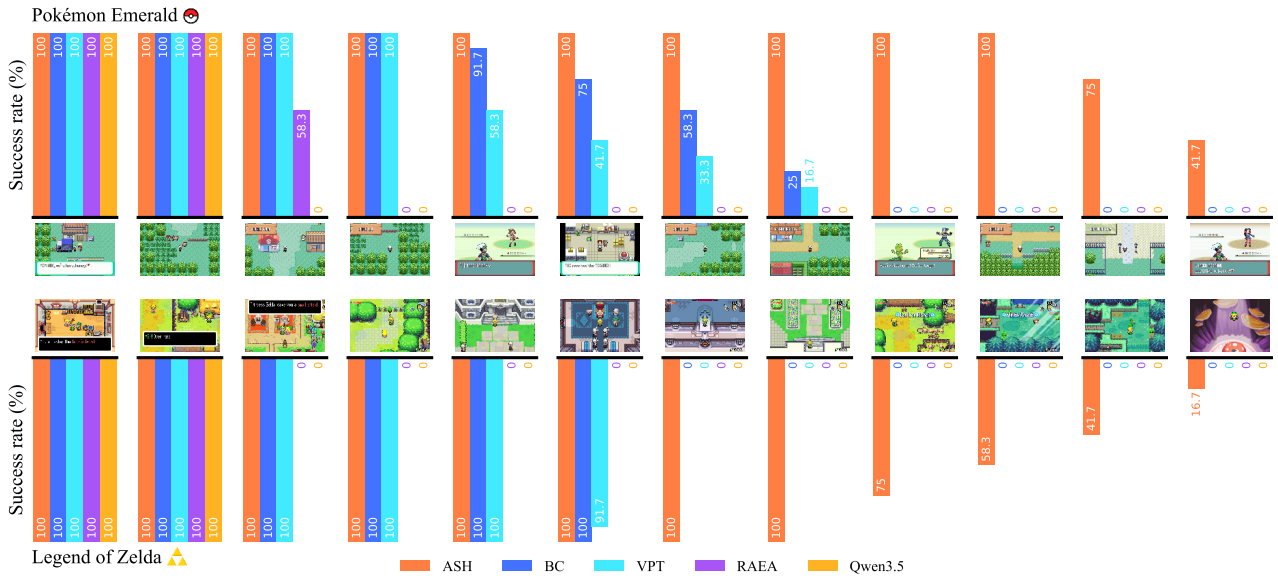


Figure 2. Agent progress in **Pokémon Emerald** (top) and **Legend of Zelda** (bottom), measured using milestone completion rates. ASH is able to adapt and continue progressing throughout the 8-hour gameplay period. While all methods are able to complete early milestones, only ASH can adapt to new areas, objectives, and mechanics. See Appendix D for standard deviation.

4. Experiments

We evaluate ASH on two complementary long-horizon environments: Pokémon Emerald, a strategic turn-based RPG with a deep skill tree and dozens of hours of content, and *The Legend of Zelda: The Minish Cap*, an action-adventure game that demands real-time dexterity and puzzle solving. Both span millions of distinct environment observations. We provide additional background on our environments in Appendix E. Through our experiments, we investigate the following:

- How does ASH compare to other adaptive and non-adaptive baselines? (Section 4.1)
- What effect do long-term memory and dynamic bootstrapping have on ASH's policy? (Section 4.2)
- What does bootstrapping do to the IDM? (Section 4.3)
- Does ASH retrieve relevant, high-quality examples from internet data? (Section 4.4)

Milestones. We evaluate our agents using *milestones*, which are human-defined markers of progress in our environments. We derive our milestones from online walkthroughs (Bulbapedia, 2025; Zelda Dungeon, 2026), descriptions of each milestone can be found in Appendix G.

Settings and implementation details. In our experiments, ASH uses $N = 4$ agents. We use a timestep of 0.25 seconds to sample observations from the environment and input actions. We set w_s and w_l to 64 and 10, respectively. At each timestep, we use w_s previous observations, with a sampling interval of 0.5 seconds, as an agent's short-term memory. This allows the agent to maintain a short-term memory of 32 seconds, while interacting with the environment every 0.25 seconds. For retrieval, we set $k = 100$ and $w_r = 540$. We set our HDBSCAN (Raschka et al., 2020) hyperparameter, $c_{\min}$, to 30. We scrape our internet datasets ($\mathcal{D}^I$) from YouTube using a list of keywords (Appendix K). Our dataset for Pokémon Emerald contains approximately 22,000 unique videos, whereas our Legend of Zelda dataset contains 17,000 videos. For dynamic bootstrapping, we set Δ to 4800 which is 20 minutes of real-time (the median video length in $\mathcal{D}^I$). ASH contains 1.2 billion parameters, 100 million for vision encoder ϕ and 1.1 billion for the transformer $\mathcal{T}$. We use fp32 precision for ϕ and bf16 for $\mathcal{T}$. We find that initializing each run from random weights is inefficient, so we use checkpoints trained for 3 and 9 GPU hours to initialize the IDM and π. The data for these short training runs is obtained from the environment's starting state using a uniformly random policy over the action space (for the IDM) and Algorithm 3 (for π). We detail architectural, inference, and training hyperparameters for ASH in Appendix H.

Baselines. We compare ASH with the following baselines. **Qwen3.5** (Qwen Team, 2026): A zero-shot multimodal reasoning model (35B params, 3B active). As with ASH, we provide Qwen3.5 with 64 memory frames, sampled at 0.5 second intervals. **VPT** (Baker et al., 2022): VPT requires an existing IDM, so we use the strongest IDM checkpoint produced by ASH, as measured by performance on a validation set; see Section 4.3. We find that direct offline training on $\mathcal{D}^I$ creates a weak policy that does not progress past the first

milestones, so we manually curate 400 hours of high-quality training data. **Offline BC**: ASH's architecture without long-term memory or bootstrapping, trained using the same IDM and data as VPT. **Retrieval-augmented Embodied Agent (RAEA)** (Zhu et al., 2024): We follow the recipe of Zhu et al. (2024) and augment offline BC with cross-attention on retrieved examples. We split our manually curated data into 350 hours of training data and 50 hours of retrievable examples, labeled by our IDM. For both inference and training, we run ASH retrieval every 32 seconds with $w_r = 16$ and $k = 1$ but return the best matching segment, rather than the whole video. Therefore, at every step, RAEA is conditioned on 32 seconds of context and a 32 second policy example. For compute parity, we allocate up to 288 GPU hours (4 days on 4×H100 SXM 80GB GPUs) total for each method, or until overfitting is detected on held-out data.

4.1. Performance on Milestones

For our baselines, we train one policy for each environment and perform 12 runs each. Since ASH infers multiple agents at once, we perform 3 training runs per environment with $N = 4$. This gives us 12 trajectories per-method. We report average milestone completion rate in Figure 2. We find that:

Self-improvement is necessary for sustained progression. Over the 8-hour evaluation, ASH reaches milestone 12 in both environments, while no baseline exceeds milestone 8 in Pokémon or 6 in Zelda. VPT and offline BC plateau once the games introduce dynamics that are underrepresented in their training distribution (for example, seldom encountered random events, or areas that are only briefly visited by players). RAEA, designed for short-horizon tasks, struggles in our long-horizon setting since it retrieves a different policy example every 32 seconds, causing instability. Finally, despite its size, Qwen3.5 hallucinates — running into walls, imagining conversations with NPCs, etc.

ASH adapts to changing dynamics. The Zelda castle entrance is the cleanest example of a distribution-shift moment (Figure 3): the visual dynamics change abruptly, and VPT and offline BC, frozen at training time, devolve into random button-mashing and never exit the castle. ASH's policy also falters, but after a bootstrap is able to continue making progress. The same pattern appears in Pokémon the moment the agent leaves the starting town and encounters the Pokémon battling interface.

Figure 3. Outside (top) vs. inside (bottom) of the Zelda castle: offline policies collapse on this dynamics shift; ASH bootstraps and continues.

Bootstrapping evolves new skills. ASH's initial checkpoint does not exhibit any ability to battle with Pokémon, and therefore gets stuck on the second milestone (which requires battling). However, through bootstrapping, ASH becomes an adept battler. After hour 4, ASH wins 87% of the Pokémon battles it engages in. In the Legend of Zelda, ASH kills enemies with its sword in 75% of trajectories. This capability is absent from its initial checkpoint and is not observed in any of our baselines. We provide examples of ASH evolving skills in Appendix A.

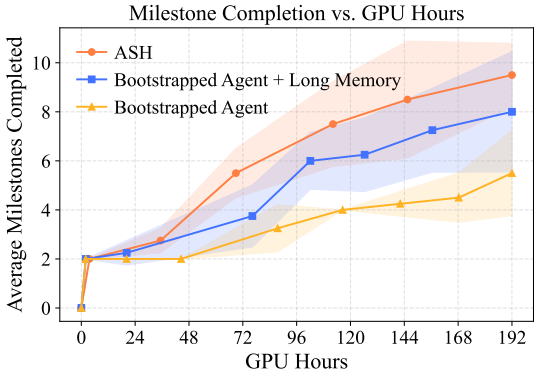
Across both environments, embodied play has a horizon beyond which a fixed policy struggles to succeed: the environments continually introduce new mechanics, regions, and visual contexts. Methods without an update mechanism plateau at that horizon. However, ASH's self-improvement loop pushes it through.

4.2. Component ablations

In Figure 4 (left), we ablate across the components of ASH and measure their contribution to milestone completion in Pokémon Emerald. **Bootstrapped Agent** removes both the long-term memory and dynamic bootstrapping, fixing the bootstrap interval to Δ. **Bootstrapped Agent + Long Memory** reintroduces the agent's long-term memory but keeps the fixed bootstrapping schedule. **ASH** adds dynamic bootstrapping. Because each variant uses different amounts of compute for inference and training, we plot milestone completion against *total GPU hours of online training*. Each method is averaged over 4 trajectories, and the shaded regions show the standard deviation. We find that the introduction of each component of ASH results in a clear increase in performance. By the end of the evaluation, ASH's long-memory component contributes 2.5 average milestones. Finally, dynamic bootstrapping allows ASH to deploy compute more efficiently by only training when it gets stuck, contributing an additional 1.5 milestones.

4.3. IDM adaptation

As an agent progresses through the environment, new visual dynamics emerge, and the IDM must correctly label these new dynamics for policy bootstrapping to be effective. To measure how well the IDM captures evolving dynamics, we take the checkpoint from each bootstrap of an ASH playthrough of Pokémon Emerald, and evaluate it on a test set of trajectories from a separate ASH playthrough. We report the per-bootstrap IDM accuracy in Figure 4. We find that the IDM becomes better at labeling environment dynamics across bootstraps. The initialization IDM checkpoint (no bootstrapping baseline) has just 60% accuracy on our test set, whereas our final IDM checkpoint achieves



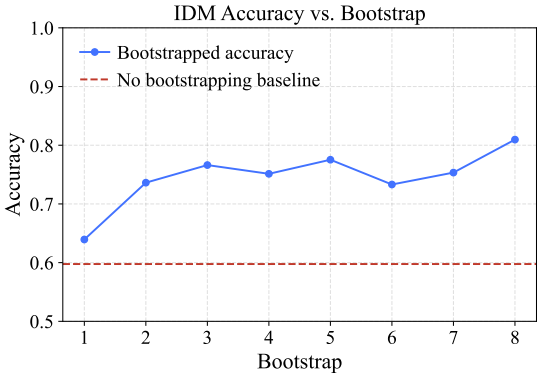


Figure 4. **(Left)** Component ablation on Pokémon Emerald: each addition (long-term memory, dynamic bootstrapping) yields a clear gain in milestones completed per GPU hour of online training. Shaded regions are one standard deviation over 4 trajectories per method. **(Right)** IDM accuracy across bootstraps, evaluated on a test set. The dashed line is the pre-bootstrap initialization checkpoint.

81%. We limit our analysis to the four (up, down, left, right) movement action classes since some actions like the `A` and `B` buttons are often visually indistinguishable.

We find that IDM bootstrapping is essential for progression. In Pokémon Emerald, training the VPT and BC baselines using our curated dataset and the no bootstrapping baseline IDM yielded 0% completion rates on milestones 3 and above.

4.4. Retrieval quality

	ASH	MSM	All2All	Rand
Binary Rel. ↑	**94%**	69%	3%	8%
Ave. Rank ↓	**1.35**	1.81	3.63	3.29

Table 1. Annotator judgments of ASH retrieval and baselines: binary relevance (yes/no) and average rank across methods.

For bootstrapping to be effective, it is crucial that ASH retrieval creates a *relevant and low-noise* training dataset $\mathcal{D}^R$ from our corpus of YouTube data, $\mathcal{D}^I$. To measure the effectiveness of ASH retrieval, we conduct a blind evaluation. For each query trajectory, an annotator sees the videos retrieved by each method and performs two tasks. First, they label each retrieved video as relevant training data (yes/no). Second, given one video per method, they rank the four videos using the following criteria, in priority order:

1. Does the retrieved video correspond to the same part of the game as the trajectory?
2. Is the retrieved video visually similar to the trajectory?
3. Does the retrieved video contain future milestones?

Using the above criteria, the annotators rank the videos from 1 (most relevant) to 4 (least relevant). For each method considered, we use 10 trajectories sampled from bootstrapping runs of Pokémon Emerald (Section 4.1). We judge the top 20 videos per trajectory, for a total of 800 videos judged.

We compare **ASH retrieval** with three baselines. **Random** selects videos uniformly from the entire dataset. **All-to-All** ranks videos by the average similarity between all trajectory and internet video embeddings. **Max Single Match** replaces the sliding window score of ASH retrieval with the maximum score of any given trajectory embedding with an internet video embedding. We report our findings in Table 1. As Pokémon is a long-horizon game, we find a randomly retrieved video has only an 8% chance of being relevant training data, highlighting the challenges of naively training a policy over $\mathcal{D}^I$. By contrast, ASH retrieves relevant videos 94% of the time, achieving an average rank of 1.35 between methods. We find that Max Single Match occasionally makes spurious matches, for example matching a black screen between videos, and All-to-All matching seems to lose all signal, consistently matching to the same internet video, irrespective of the query.

5. Limitations and Conclusion

We introduced ASH, an agent that self-improves by retrieving and learning from relevant internet video, without reward engineering or human annotation. Across Pokémon Emerald and The Legend of Zelda, ASH continues to adapt beyond the point where behavioral cloning, retrieval-augmented, and foundation-model baselines stall, developing new skills such as battling, catching Pokémon, and using a sword and shield. These results show that self-improving agents are a scalable path toward open-ended, long-horizon embodied learning.

Although ASH improves over fixed policies in long-horizon settings, it depends on the availability of useful internet video and may be harder to apply in environments where demonstrations are scarce. It also relies on pseudo-actions from an Inverse Dynamics Model, which will struggle to label actions that are difficult to infer from pixels. Finally, since our approach is not specific to Pokémon or Zelda, we hope ASH can inspire future work in creating long-horizon, embodied agents in other domains and environments.

References

Ahn, M., Dwibedi, D., Finn, C., Arenas, M. G., Gopalakrishnan, K., Hausman, K., Ichter, B., Irpan, A., Joshi, N., Julian, R., Kirmani, S., Leal, I., Lee, E., Levine, S., Lu, Y., Leal, I., Maddineni, S., Rao, K., Sadigh, D., Sanketi, P., Sermanet, P., Vuong, Q., Welker, S., Xia, F., Xiao, T., Xu, P., Xu, S., and Xu, Z. Autort: Embodied foundation models for large scale orchestration of robotic agents, 2024.

Anthony, T., Tian, Z., and Barber, D. Thinking fast and slow with deep learning and tree search. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, NIPS'17, pp. 5366–5376, Red Hook, NY, USA, 2017. Curran Associates Inc. ISBN 9781510860964.

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, 2022. URL https://arxiv.org/abs/2206.11795.

Borgeaud, S., Mensch, A., Hoffmann, J., Cai, T., Rutherford, E., Millican, K., Van Den Driessche, G. B., Lespiau, J.-B., Damoc, B., Clark, A., De Las Casas, D., Guy, A., Menick, J., Ring, R., Hennigan, T., Huang, S., Maggiore, L., Jones, C., Cassirer, A., Brock, A., Paganini, M., Irving, G., Vinyals, O., Osindero, S., Simonyan, K., Rae, J., Elsen, E., and Sifre, L. Improving language models by retrieving from trillions of tokens. In Chaudhuri, K., Jegelka, S., Song, L., Szepesvari, C., Niu, G., and Sabato, S. (eds.), *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pp. 2206–2240. PMLR, 17–23 Jul 2022. URL https://proceedings.mlr.press/v162/borgeaud22a.html.

Bruce, J., Dennis, M. D., Edwards, A., Parker-Holder, J., Shi, Y., Hughes, E., Lai, M., Mavalankar, A., Steigerwald, R., Apps, C., Aytar, Y., Bechtle, S. M. E., Behbahani, F., Chan, S. C., Heess, N., Gonzalez, L., Osindero, S., Ozair, S., Reed, S., Zhang, J., Zolna, K., Clune, J., Freitas, N. D., Singh, S., and Rocktäschel, T. Genie: Generative interactive environments. In Salakhutdinov, R., Kolter, Z., Heller, K., Weller, A., Oliver, N., Scarlett, J., and Berkenkamp, F. (eds.), *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 4603–4623. PMLR, 21–27 Jul 2024. URL https://proceedings.mlr.press/v235/bruce24a.html.

Bulatov, A., Kuratov, Y., and Burtsev, M. Recurrent memory transformer. In Oh, A. H., Agarwal, A., Belgrave, D., and Cho, K. (eds.), *Advances in Neural Information Processing Systems*, 2022. URL https://openreview.net/forum?id=Uynr3iPhksa.

Bulbapedia. Walkthrough:pokémon emerald, Jun 2025. URL https://bulbapedia.bulbagarden.net/wiki/Walkthrough:Pok%C3%A9mon_Emerald.

Campello, R. J. G. B., Moulavi, D., and Sander, J. Density-based clustering based on hierarchical density estimates. In Pei, J., Tseng, V. S., Cao, L., Motoda, H., and Xu, G. (eds.), *Advances in Knowledge Discovery and Data Mining*, pp. 160–172, Berlin, Heidelberg, 2013. Springer Berlin Heidelberg. ISBN 978-3-642-37456-2.

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, 2021. URL https://arxiv.org/abs/2106.01345.

ClaudePlaysPokemon. ClaudePlaysPokemon. https://www.twitch.tv/claudeplayspokemon, 2026. [Accessed 01-05-2026].

Edwards, A., Sahni, H., Schroecker, Y., and Isbell, C. Imitating latent policies from observation. In Chaudhuri, K. and Salakhutdinov, R. (eds.), *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pp. 1755–1763. PMLR, 09–15 Jun 2019. URL https://proceedings.mlr.press/v97/edwards19a.html.

Fan, L., Wang, G., Jiang, Y., Mandlekar, A., Yang, Y., Zhu, H., Tang, A., Huang, D.-A., Zhu, Y., and Anandkumar, A. Minedojo: Building open-ended embodied agents with internet-scale knowledge. In *Thirty-sixth Conference on Neural Information Processing Systems Datasets and Benchmarks Track*, 2022. URL https://openreview.net/forum?id=rc8o_j8I8PX.

Goyal, A., Friesen, A., Banino, A., Weber, T., Ke, N. R., Badia, A. P., Guez, A., Mirza, M., Humphreys, P. C., Konyushova, K., Valko, M., Osindero, S., Lillicrap, T., Heess, N., and Blundell, C. Retrieval-augmented reinforcement learning. In Chaudhuri, K., Jegelka, S., Song, L., Szepesvari, C., Niu, G., and Sabato, S. (eds.), *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pp. 7740–7765. PMLR, 17–23 Jul 2022. URL https://proceedings.mlr.press/v162/goyal22a.html.

Gulcehre, C., Paine, T. L., Srinivasan, S., Konyushkova, K., Weerts, L., Sharma, A., Siddhant, A., Ahern, A., Wang, M., Gu, C., Macherey, W., Doucet, A., Firat, O., and

de Freitas, N. Reinforced self-training (rest) for language modeling, 2023. URL https://arxiv.org/abs/2308.08998.

Hafner, D., Yan, W., and Lillicrap, T. Training agents inside of scalable world models, 2025. URL https://arxiv.org/abs/2509.24527.

Hambro, E., Raileanu, R., Rothermel, D., Mella, V., Rocktäschel, T., Küttler, H., and Murray, N. Dungeons and data: A large-scale nethack dataset, 2023. URL https://arxiv.org/abs/2211.00539.

Hu, Y., Xie, Q., Jain, V., Francis, J., Patrikar, J., Keetha, N., Kim, S., Xie, Y., Zhang, T., Fang, H.-S., Zhao, S., Omidshafiei, S., Kim, D.-K., akbar Agha-mohammadi, A., Sycara, K., Johnson-Roberson, M., Batra, D., Wang, X., Scherer, S., Wang, C., Kira, Z., Xia, F., and Bisk, Y. Toward general-purpose robots via foundation models: A survey and meta-analysis, 2024. URL https://arxiv.org/abs/2312.08782.

Huang, W.-C., Zhang, W., Liang, Y., Bei, Y., Chen, Y., Feng, T., Pan, X., Tan, Z., Wang, Y., Wei, T., Wu, S., Xu, R., Yang, L., Yang, R., Yang, W., Yeh, C.-Y., Zhang, H., Zhang, H., Zhu, S., Zou, H. P., Zhao, W., Wang, S., Xu, W., Ke, Z., Hui, Z., Li, D., Wu, Y., He, L., Wang, C., Xu, X., Huang, B., Tan, J., Heinecke, S., Wang, H., Xiong, C., Metwally, A. A., Yan, J., Lee, C.-Y., Zeng, H., Xia, Y., Wei, X., Payani, A., Wang, Y., Ma, H., Wang, W., Wang, C., Zhang, Y., Wang, X., Zhang, Y., You, J., Tong, H., Luo, X., Liu, X., Sun, Y., Wang, W., McAuley, J., Zou, J., Han, J., Yu, P. S., and Shu, K. Rethinking memory mechanisms of foundation agents in the second half: A survey, 2026. URL https://arxiv.org/abs/2602.06052.

Humphreys, P. C., Guez, A., Tieleman, O., Sifre, L., Weber, T., and Lillicrap, T. P. Large-scale retrieval for reinforcement learning. In Oh, A. H., Agarwal, A., Belgrave, D., and Cho, K. (eds.), *Advances in Neural Information Processing Systems*, 2022. URL https://openreview.net/forum?id=Ya9lATuQ3gg.

Karten, S., Grigsby, J., Milani, S., Vodrahalli, K., Zhang, A., Fang, F., Zhu, Y., and Jin, C. The pokeagent challenge: Competitive and long-context learning at scale. In *NeurIPS Competition Track*, April 2025.

Lewis, P., Perez, E., Piktus, A., Petroni, F., Karpukhin, V., Goyal, N., Küttler, H., Lewis, M., Yih, W.-t., Rocktäschel, T., Riedel, S., and Kiela, D. Retrieval-augmented generation for knowledge-intensive nlp tasks. In *Proceedings of the 34th International Conference on Neural Information Processing Systems*, NIPS '20, Red Hook, NY, USA, 2020. Curran Associates Inc. ISBN 9781713829546.

Li, Z., Xie, Y., Shao, R., Chen, G., Jiang, D., and Nie, L. Optimus-2: Multimodal minecraft agent with goal-observation-action conditioned policy. In *2025 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, 2025.

Lifshitz, S., Paster, K., Chan, H., Ba, J., and McIlraith, S. A. STEVE-1: A generative model for text-to-behavior in minecraft. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023. URL https://openreview.net/forum?id=YkBDJWerKg.

Loshchilov, I. and Hutter, F. Decoupled weight decay regularization, 2019. URL https://arxiv.org/abs/1711.05101.

Mnih, V., Kavukcuoglu, K., Silver, D., Graves, A., Antonoglou, I., Wierstra, D., and Riedmiller, M. Playing atari with deep reinforcement learning, 2013. URL https://arxiv.org/abs/1312.5602.

OpenAI, :, Berner, C., Brockman, G., Chan, B., Cheung, V., Dębiak, P., Dennison, C., Farhi, D., Fischer, Q., Hashme, S., Hesse, C., Józefowicz, R., Gray, S., Olsson, C., Pachocki, J., Petrov, M., d. O. Pinto, H. P., Raiman, J., Salimans, T., Schlatter, J., Schneider, J., Sidor, S., Sutskever, I., Tang, J., Wolski, F., and Zhang, S. Dota 2 with large scale deep reinforcement learning, 2019. URL https://arxiv.org/abs/1912.06680.

Oquab, M., Darcet, T., Moutakanni, T., Vo, H., Szafraniec, M., Khalidov, V., Fernandez, P., Haziza, D., Massa, F., El-Nouby, A., Assran, M., Ballas, N., Galuba, W., Howes, R., Huang, P.-Y., Li, S.-W., Misra, I., Rabbat, M., Sharma, V., Synnaeve, G., Xu, H., Jegou, H., Mairal, J., Labatut, P., Joulin, A., and Bojanowski, P. Dinov2: Learning robust visual features without supervision, 2024. URL https://arxiv.org/abs/2304.07193.

Pleines, M., Addis, D., Rubinstein, D., Zimmer, F., Preuss, M., and Whidden, P. Pokemon red via reinforcement learning, 2025.

Qwen Team. Qwen3.5: Accelerating productivity with native multimodal agents, February 2026. URL https://qwen.ai/blog?id=qwen3.5.

Raschka, S., Patterson, J., and Nolet, C. Machine learning in python: Main developments and technology trends in data science, machine learning, and artificial intelligence. *arXiv preprint arXiv:2002.04803*, 2020.

Ross, S. and Bagnell, D. Efficient reductions for imitation learning. In Teh, Y. W. and Titterington, M. (eds.), *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, volume 9 of *Proceedings of Machine Learning Research*, pp. 661–668, Chia Laguna Resort, Sardinia, Italy, 13–15 May

2010. PMLR. URL https://proceedings.mlr.press/v9/ross10a.html.

Schmidt, D. and Jiang, M. Learning to act without actions. In *The Twelfth International Conference on Learning Representations (ICLR)*, 2024.

Shinn, N., Cassano, F., Gopinath, A., Narasimhan, K. R., and Yao, S. Reflexion: language agents with verbal reinforcement learning. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023. URL https://openreview.net/forum?id=vAElhFcKW6.

Silver, D., Hubert, T., Schrittwieser, J., Antonoglou, I., Lai, M., Guez, A., Lanctot, M., Sifre, L., Kumaran, D., Graepel, T., Lillicrap, T., Simonyan, K., and Hassabis, D. Mastering chess and shogi by self-play with a general reinforcement learning algorithm, 2017. URL https://arxiv.org/abs/1712.01815.

team, S., Bolton, A., Lerchner, A., Cordell, A., Moufarek, A., Bolt, A., Lampinen, A., Mitenkova, A., Hallingstad, A. O., Vujatovic, B., Li, B., Lu, C., Wierstra, D., Sawyer, D. P., Slater, D., Reichert, D., Vercelli, D., Hassabis, D., Hudson, D. A., Williams, D., Hirst, E., Pardo, F., Hill, F., Besse, F., Openshaw, H., Chan, H., Soyer, H., Wang, J. X., Clune, J., Agapiou, J., Reid, J., Marino, J., Kim, J., Gregor, K., Sridhar, K., McKinney, K., Kampis, L., Zhang, L. M., Matthey, L., Wang, L., Raad, M. A., Loks-Thompson, M., Engelcke, M., Kecman, M., Jackson, M., Gazeau, M., Purkiss, O., Knagg, O., Stys, P., Mendolicchio, P., Hadsell, R., Ke, R., Faulkner, R., Chakera, S., Baveja, S. S., Legg, S., Kashem, S., Terzi, T., Keck, T., Harley, T., Scholtes, T., Roberts, T., Mnih, V., Liu, Y., Wang, Z., and Ghahramani, Z. Sima 2: A generalist embodied agent for virtual worlds, 2025. URL https://arxiv.org/abs/2512.04797.

Torabi, F., Warnell, G., and Stone, P. Behavioral cloning from observation. In *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence, IJCAI-18*, pp. 4950–4957. International Joint Conferences on Artificial Intelligence Organization, 7 2018a. doi: 10.24963/ijcai.2018/687. URL https://doi.org/10.24963/ijcai.2018/687.

Torabi, F., Warnell, G., and Stone, P. Behavioral cloning from observation, 2018b. URL https://arxiv.org/abs/1805.01954.

Vinyals, O., Babuschkin, I., Czarnecki, W. M., Mathieu, M., Dudzik, A. J., Chung, J., Choi, D., Powell, R., Ewalds, T., Georgiev, P., Oh, J., Horgan, D., Kroiss, M., Danihelka, I., Huang, A., Sifre, L., Cai, T., Agapiou, J. P., Jaderberg, M., Vezhnevets, A. S., Leblond, R., Pohlen, T., Dalibard, V., Budden, D., Sulsky, Y., Molloy, J., Paine, T. L., Gulcehre, C., Wang, Z., Pfaff, T., Wu, Y., Ring, R., Yogatama, D., Wünsch, D., McKinney, K., Smith, O., Schaul, T., Lillicrap, T. P., Kavukcuoglu, K., Hassabis, D., Apps, C., and Silver, D. Grandmaster level in starcraft ii using multi-agent reinforcement learning. *Nature*, 575:350 – 354, 2019. URL https://api.semanticscholar.org/CorpusID:204972004.

Wang, G., Xie, Y., Jiang, Y., Mandlekar, A., Xiao, C., Zhu, Y., Fan, L., and Anandkumar, A. Voyager: An open-ended embodied agent with large language models, 2023. URL https://arxiv.org/abs/2305.16291.

Wang, Z., Yang, E., Shen, L., and Huang, H. A comprehensive survey of forgetting in deep learning beyond continual learning. *IEEE Trans. Pattern Anal. Mach. Intell.*, 47(3):1464–1483, March 2025. ISSN 0162-8828. doi: 10.1109/TPAMI.2024.3498346. URL https://doi.org/10.1109/TPAMI.2024.3498346.

Wu, Y., Rabe, M. N., Hutchins, D., and Szegedy, C. Memorizing transformers. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=TrjbxzRcnf-.

Yang, A., Li, A., Yang, B., Zhang, B., Hui, B., Zheng, B., Yu, B., Gao, C., Huang, C., Lv, C., Zheng, C., Liu, D., Zhou, F., Huang, F., Hu, F., Ge, H., Wei, H., Lin, H., Tang, J., Yang, J., Tu, J., Zhang, J., Yang, J., Yang, J., Zhou, J., Zhou, J., Lin, J., Dang, K., Bao, K., Yang, K., Yu, L., Deng, L., Li, M., Xue, M., Li, M., Zhang, P., Wang, P., Zhu, Q., Men, R., Gao, R., Liu, S., Luo, S., Li, T., Tang, T., Yin, W., Ren, X., Wang, X., Zhang, X., Ren, X., Fan, Y., Su, Y., Zhang, Y., Zhang, Y., Wan, Y., Liu, Y., Wang, Z., Cui, Z., Zhang, Z., Zhou, Z., and Qiu, Z. Qwen3 technical report, 2025. URL https://arxiv.org/abs/2505.09388.

Yao, S., Zhao, J., Yu, D., Du, N., Shafran, I., Narasimhan, K., and Cao, Y. React: Synergizing reasoning and acting in language models. *arXiv preprint arXiv:2210.03629*, 2022.

Zelda Dungeon. The minish cap walkthrough. https://www.zeldadungeon.net/the-minish-cap-walkthrough/, 2026. [Accessed 30-04-2026].

Zelikman, E., Wu, Y., Mu, J., and Goodman, N. STar: Bootstrapping reasoning with reasoning. In Oh, A. H., Agarwal, A., Belgrave, D., and Cho, K. (eds.), *Advances in Neural Information Processing Systems*, 2022. URL https://openreview.net/forum?id=_3ELRdg2sgI.

Zhai, X., Mustafa, B., Kolesnikov, A., and Beyer, L. Sigmoid loss for language image pre-training, 2023. URL https://arxiv.org/abs/2303.15343.

Zhang, D., Chen, L., Zhang, S., Xu, H., Zhao, Z., and Yu, K. Large language models are semi-parametric reinforcement learning agents. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023. URL https://openreview.net/forum?id=ZcJa1R6j3v.

Zhu, Y., Ou, Z., Mou, X., and Tang, J. Retrieval-Augmented Embodied Agents . In *2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 17985–17995, Los Alamitos, CA, USA, June 2024. IEEE Computer Society. doi: 10.1109/CVPR52733.2024.01703. URL https://doi.ieeecomputersociety.org/10.1109/CVPR52733.2024.01703.

A. ASH Playthrough Examples

Examples of how ASH uses **dynamic bootstrapping** and **long-memory** to overcome roadblocks.

Figure 5. **Dynamic bootstrapping example.** To complete milestone 2, the player must rescue the Professor from a wild Zigzagoon **(Panel 1)**. To accomplish this, the player must use their starter Pokémon to defeat the Zigzagoon in battle **(Panel 2)**. However, ASH's initial policy does not know how to use the battle interface to command their Pokémon. After being stuck for Δ steps (20 minutes), ASH **dynamically bootstraps**, and finds relevant examples of battling from $\mathcal{D}^I$ **(Panel 3)**. After updating its policy, ASH is able to attack and defeat the Zigzagoon **(Panels 4 & 5)**. After learning how to battle, ASH is **unstuck** and continues on its journey **(Panel 6)**.

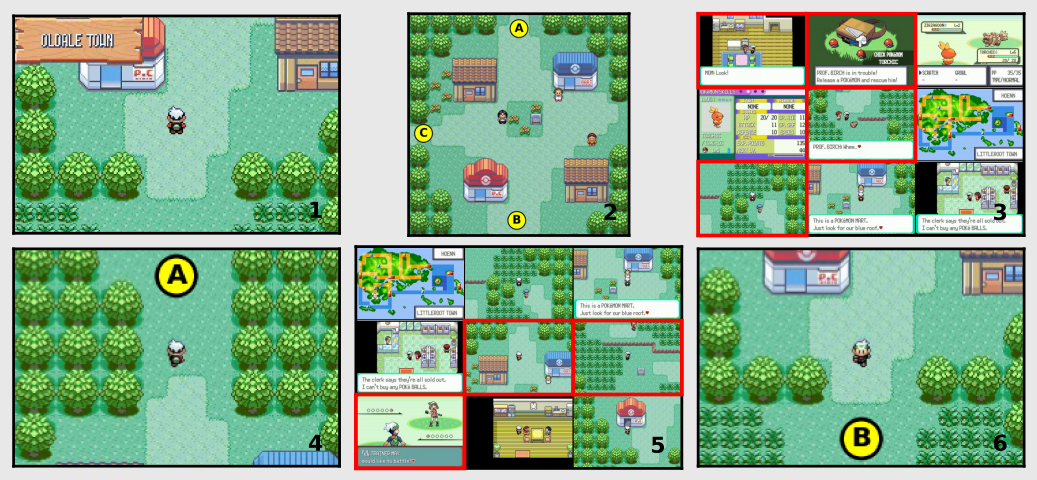


Figure 6. **Long-term memory example.** When the player arrives in Oldale town **(Panel 1)**, they are presented with 3 possible next paths. **Option `A`:** The player heads north to Route 103 to meet their rival, May. This is the correct choice if the player has just obtained their starter Pokémon from the Professor and been tasked with bringing May back to the lab. **Option `B`:** The player has already met May, and should head back to the lab to meet the professor. Or **Option `C`:** The player has obtained the Pokédex at the lab from May and the Professor, and should proceed to Route 102. However, **short-term memory is insufficient for this decision**, as short-term observations are nearly identical. However, from inspecting the agent's **long-term memory (Panel 3)**, it is clear that the player has just obtained their starter Pokémon, and therefore ASH chooses **Option `A`** unambiguously **(Panel 4)**. After meeting May, the player's long-term memory has been updated with their encounter **(Panel 5)**, so it knows to choose **Option `B`** and head back to the lab **(Panel 6)**.

B. Cluster Analysis

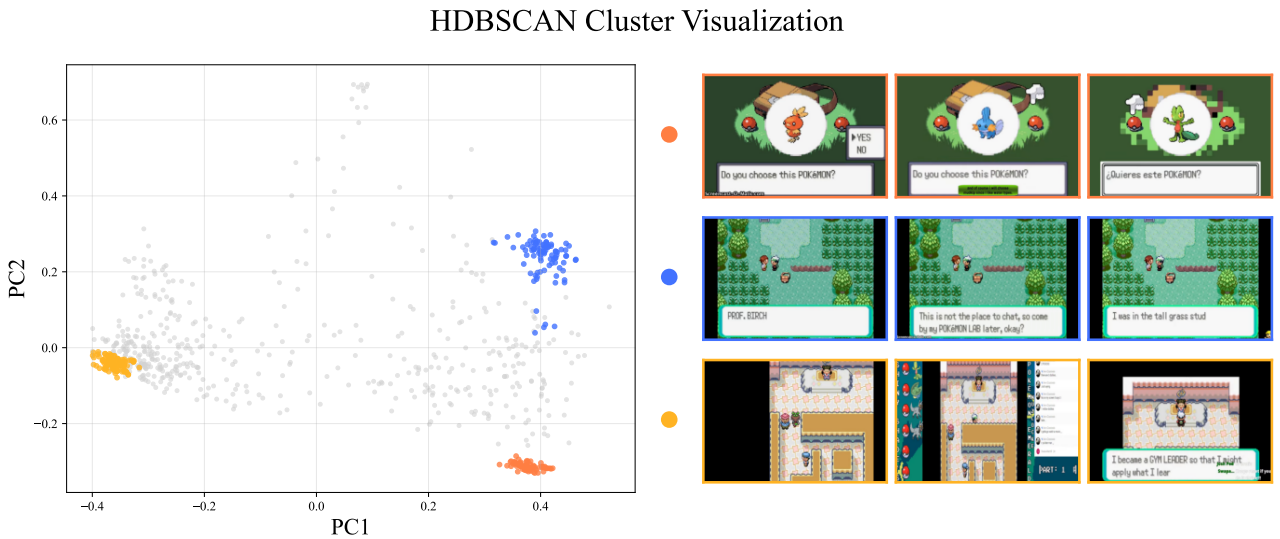


Figure 7. Visualization of 3 HDBSCAN (Campello et al., 2013) clusters, as well as 500 uniformly sampled outlier points reduced to 2 dimensions via principal component analysis. Each of these clusters represents a key moment in Pokémon Emerald; choosing a starter Pokémon (orange), saving the professor (blue) and challenging a gym leader (yellow). Grey dots are outliers that are not assigned to a cluster by HDBSCAN (Campello et al., 2013).

The effectiveness of ASH's long-term memory depends on the quality of the clusters produced by HDBSCAN (Campello et al., 2013). A cluster should be composed of data from a single *key moment* — an observation that can be used to contextualize a player's actions within the broader trajectory. Examples of key moments include reaching a milestone, selecting your starter Pokémon, or battling a particular enemy.

To evaluate how well HDBSCAN (Campello et al., 2013) clusters these key moments, we evaluated 400 clusters sampled uniformly at random from a pool of 2,200 clusters generated by ASH across all bootstraps during an eight-hour run in Pokémon Emerald. For each cluster, annotators were shown five of its constituent internet video frames and answered two yes/no questions:

1. Are all frames *key moments*?
2. Do all frames correspond to the *same* key moment?

If the answer to both questions is yes, we consider the cluster to be a key moment. Based on this analysis, we report that 48% of clusters identified by HDBSCAN (Campello et al., 2013) correspond to key moments. The effectiveness of HDBSCAN in identifying these key moments helps explain the performance increase observed in Section 4.2.

C. Offline Replay

Catastrophic forgetting is a phenomenon in lifelong learning where an agent will forget previously known skills and knowledge when its policy is updated (Wang et al., 2025). The result is an agent that can progress through the latter stages of an environment but can no longer accomplish early milestones. We examine whether ASH's final policy is able to play through the environment from the beginning without bootstrapping, or if it exhibits catastrophic forgetting.

We compare the final ASH checkpoint (offline ASH) with the online run that created it (online ASH). Both methods are run for 1.3 million environment steps (6 hours) in Pokémon Emerald (Figure 8). Results are averaged over 4 trajectories and the shaded regions show the standard deviation. We find that ASH does not exhibit catastrophic

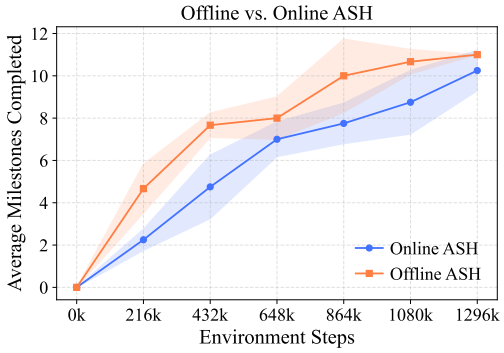


Figure 8. Offline replay of the final ASH checkpoint vs. the original online run.

forgetting. As the offline ASH checkpoint has already learned (and remembers) the environment, it progresses through milestones faster than the online learning agent. This is analogous to a player replaying the environment, who, due to accumulated expertise, is faster than the beginner.

D. Average Milestone Results

In Table 2, we report the average number of milestones reached (out of 12) in both environments at the end of the 8-hour evaluation.

Method	Pokémon Emerald	Legend of Zelda
ASH	$\mathbf{11.17 \pm 0.83}$	$\mathbf{9.92 \pm 1.51}$
BC	6.50 ± 1.31	6.00 ± 0.00
VPT	5.50 ± 1.62	5.92 ± 0.29
RAEA	2.58 ± 0.51	2.00 ± 0.00
Qwen3.5	2.00 ± 0.00	2.00 ± 0.00

Table 2. Average milestones reached (out of 12) at the end of the 8-hour evaluation, reported as mean $\pm$ one standard deviation.

E. Background on Environments

Pokémon Emerald is a third-generation *Pokémon* role-playing game for the Game Boy Advance (GBA). The game follows a largely linear-but-explorable progression: the player travels between towns and routes, completes story events, captures and trains creatures ("Pokémon"), and defeats a sequence of gym leaders to earn badges, culminating in the Elite Four and Champion battles.

Gameplay consists of real-time overworld navigation (movement, interactions with NPCs, and menu-driven inventory/team management) interleaved with turn-based battles. Battles require selecting discrete actions (moves, items, switching Pokémon, or attempting capture) under uncertainty due to hidden information (e.g., opponent move choice, variable damage rolls, status effects). Progress is gated by narrative triggers and abilities acquired over time (e.g., key items and field moves), producing long-horizon dependencies, sparse success signals, and frequent context switches between exploration, dialogue, menus, and combat. Bulbapedia (2025) provides a comprehensive overview of progression, game mechanics and storyline.

Pokémon Emerald is a demanding test of long-horizon embodied intelligence: it must map high-dimensional visual observations to low-level button sequences while coping with partial observability, delayed consequences, and sparse success signals. The outcomes of many actions are unclear over even long time horizons. For example, your starter Pokémon choice may be a crucial factor in winning a battle dozens of hours into a playthrough.

Legend of Zelda: The Minish Cap is a Game Boy Advance game that combines real-time combat, puzzle solving, and RPG mechanics. The player controls Link, a young hero tasked with rescuing Princess Zelda after she is turned to stone by the evil sorcerer Vaati during the annual Picori Festival. To restore Zelda and prevent Vaati from unleashing further chaos upon the kingdom of Hyrule, Link embarks on a journey to reforge the sacred Picori Blade.

The core gameplay loop consists of exploring a diverse set of zones collecting items to save the princess, defeating enemies with the player's sword and shield, and solving increasingly difficult puzzles using the game's RPG mechanics. The game's central mechanic is the ability to use the Minish Cap to shrink Link. This mechanic (in conjunction with other game mechanics) is combined with environmental puzzles to create settings where the player must reason how to use Link's ability to grow and shrink to progress through novel areas and terrains. This mechanic is also central to the combat in the game: when Link shrinks enemies that were once trivial become large and imposing. Zelda Dungeon (2026) provides a detailed walkthrough of the game, including areas, game mechanics, and the storyline of Link's quest.

F. ASH

Algorithm 5 ASH

Require: Number of agents N, short-term memory length w_s, long-term memory length w_l, max steps without new key moment Δ, retrieval match length w_r, number of videos retrieved k, internet dataset $\mathcal{D}^I$

Step 0: Initialization
$\pi \leftarrow$ Initialize ASH policy
$\mathcal{K} \leftarrow$ Initialize new key moment classifier
IDM $\leftarrow$ Initialize IDM
$\mathcal{E}_{1\sim N} \leftarrow$ Initialize environments {All agents start at the beginning of the environment.}
$\tau_{1\sim N} \leftarrow \emptyset$ {Per agent environment trajectories.}
$\mathcal{M}_{1\sim N} \leftarrow \emptyset$ {Key moment memory per agent.}

Step 1: Inference
$C_{1\sim N} \leftarrow 0$
while $\max_n C_n < \Delta$ **do**
  for $n = 1$ to N **do**
    $\rho \leftarrow \mathcal{M}_{n,\,-w_l:}$ {Retrieve w_l most recent long-memories from the agent's memory bank.}
    $a \sim \pi(\cdot \mid \tau_{n,\,-w_s:},\, \rho)$ {Predict action using short and long-term memory.}
    $o \leftarrow \mathcal{E}_n(a)$
    $\tau_n \leftarrow \tau_n \cup \{(o, a)\}$
    if $\mathcal{K}(\tau_n,\, |\tau_n|)$ **then** {Add observation to memory bank if it is a *new key moment*.}
      $\mathcal{M}_n \leftarrow \mathcal{M}_n \cup \{o\}$
      $C_n \leftarrow 0$ {Reset steps since last new key moment counter.}
    end if
    $C_n \leftarrow C_n + 1$
  end for
end while

Step 2: Retrieval
$\mathcal{D}^R \leftarrow \emptyset$
for $n = 1$ to N **do**
  $\boldsymbol{Q} \leftarrow \textsc{Embed}(\tau_n)$
  scores $\leftarrow \emptyset$
  for $\tau^{\text{obs}} \in \mathcal{D}^I$ **do**
    $\boldsymbol{E} \leftarrow \textsc{Embed}(\tau^{\text{obs}})$ {$\boldsymbol{E}$ is cached for efficiency.}
    $e \leftarrow \textsc{Rows}(\boldsymbol{E})$
    $\boldsymbol{S} \leftarrow \boldsymbol{Q}\boldsymbol{E}^\top$
    $z^\star \leftarrow -\infty$
    for $t \in \{1, \ldots, e - w_r + 1\}$ **do** {Compute scores over windows.}
      $s \leftarrow \boldsymbol{S}_{:,\,t:t+w_r}$
      $z \leftarrow 0$
      for $j = 1$ to w_r **do**
        $r^\star \leftarrow \arg\max_r s_{r,j}$
        $z \leftarrow z + s_{r^\star,j}$
        $s_{r^\star,:} \leftarrow -\infty$
      end for
      $z^\star \leftarrow \max(z^\star, z)$
    end for
    scores $\leftarrow$ scores $\cup \{(\tau^{\text{obs}}, z^\star)\}$
  end for
  $\mathcal{D}^R \leftarrow \mathcal{D}^R \cup \{\text{top-k(scores)}\}$ {Unionize videos into a dataset.}
end for

Step 3: Policy Bootstrapping
Update $\mathcal{K}$ using $\mathcal{D}^R$
Update IDM on $\tau_{1\sim N}$
$\mathcal{D}^\pi \leftarrow \emptyset$
for $\tau^{obs} \in \mathcal{D}^R$ **do**
  for $t = 1$ to $|\tau^{obs}| - w_s + 1$ step w_s **do**
    $\tilde{\boldsymbol{A}} \leftarrow \text{IDM}(\tau^{obs}_{t:t+w_s})$ {Use IDM to create pseudo-actions for internet data item.}
    $\bar{\rho} \leftarrow [\,\tau^{obs}_i \;:\; i < t,\, \mathcal{K}(\tau^{obs}, i)\,]$ {Find key moments in τ^{obs}.}
    $\mathcal{D}^\pi \leftarrow \mathcal{D}^\pi \cup \left\{\left(\tau^{obs}_{t:t+w_s},\, \tilde{\boldsymbol{A}},\, \bar{\rho}_{-w_l:}\right)\right\}$ {Add sample to dataset.}
  end for
end for
Update π on $\mathcal{D}^\pi$
GoTo **Step 1**

G. Milestones

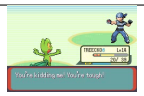
Milestone	Icon	Description
Home		Before setting off on their journey, the player must talk to their mother and set the clock in their home.
Save the professor		The player must obtain their starter Pokémon and save the professor from an encounter with a wild Pokémon.
Oldale Town		After progressing through Route 101, the player reaches the first town after their hometown. Oldale Town contains many important optional objectives such as healing your Pokémon at the Pokémon Center and obtaining items at the Poké Mart.
Route 103		Route 103 is the area above Oldale Town. This milestone is achieved when the player first enters the route.
Rival Battle		After meeting them on route 103, the player must defeat a trainer known as the "rival" in a Pokémon battle. This milestone is completed when the player successfully defeats their rival in a battle.
Obtain Pokédex		Upon defeating their rival, the player must backtrack to their hometown to obtain the Pokédex item from the professor. This milestone is completed when the player obtains the Pokédex.
Route 102		With the Pokédex in hand, the player can return to Oldale Town and continue on their journey through route 102. This milestone is achieved when the player first enters route 102.
Arrive in Petalburg City		After traveling through Route 102, the player arrives in Petalburg City, where they meet key characters such as the gym leader (their father) and progress the main story.
Defeat Team Aqua in Petalburg Woods		While passing through Petalburg Woods, the player encounters a Team Aqua member threatening a researcher. This milestone is completed when the player defeats the Team Aqua member in battle.
Route 104		Route 104 connects Petalburg Woods to Rustboro City. This milestone is achieved when the player first enters Route 104 after exiting the forest.
Rustboro City		After traveling through Route 104, the player arrives in Rustboro City, home to the first Gym challenge and several important locations such as the Trainer's School and Devon Corporation.
Defeat the Gym Leader		In Rustboro City, the player can battle the gym leader *Roxanne*. Gym leaders are powerful trainers who act as bosses that the trainer must defeat to progress. The milestone is marked as completed *after* the player wins the battle.

Table 3. Milestones, icons and corresponding descriptions for the Pokémon Emerald environment.

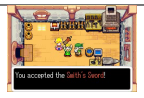
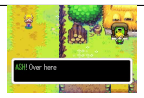
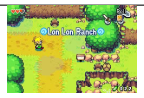
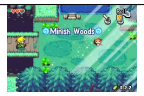
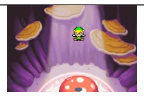
Milestone	Icon	Description
Obtain the sword from the smith		The player must speak to the smith to obtain a sword.
Meet with Zelda		After obtaining the sword, the player must meet up with Zelda to head to the festival. This milestone is marked as completed when the player 'groups up' with Zelda and she follows the player to the festival.
Obtain the small shield		At the festival, the player must win the small shield item as a prize in a contest. The milestone is completed when the player obtains the shield.
Reflect the seed projectile with the shield		After leaving the festival, the player encounters a *Deku Scrub* who is firing seeds and blocking the path. To progress, the player must use their shield to deflect a seed back at the Deku Scrub to stun it.
Enter Hyrule Castle		The player arrives at the entrance to Hyrule Castle. This milestone is achieved when the player reaches the castle exterior.
Talk with the king		The player enters the throne room and has an audience with the King of Hyrule.
Leave the castle		After speaking with the king, the player must leave the castle. This milestone is marked as completed once the player exits the castle. We allow the player to use any of the castle exits for this milestone.
Leave the palace grounds		This milestone is completed when the player leaves the palace grounds area.
Enter Lon Lon Ranch		To get to the Minish Forest, the player must go through the *Lon Lon Ranch* zone, this milestone is completed when the player first enters this new area.
Enter the Minish Woods		After navigating through Lon Lon Ranch, the player can enter the Minish Forest, an important location for the player's quest. This milestone is completed when the player enters the forest.
Obtain the Minish Cap		In the Minish Forest, the player obtains the *Minish Cap*. A new item that allows the player to shrink in size. When the player places the Minish Cap on their head, this milestone is completed.
Solve the Minish Cap Puzzle		Using the new shrinking game mechanic (from the Minish Cap), the player must solve a puzzle that requires using the Minish Cap to fit through narrow spaces (such as the interior of a log). This milestone is completed when the player finishes the puzzle.

Table 4. Milestones, icons and corresponding descriptions for the Legend of Zelda environment.

H. Hyperparameters and Training Settings

Hyperparameter	Setting
Optimizer	AdamW (Loshchilov & Hutter, 2019)
Learning Rate	5×10^{-5}
LR Scheduling	Linear
Warm-up Ratio	5%
Weight Decay	1×10^{-5}
Max Epochs	3
Batch Size	4
Trajectories (N)	4
Retrieved Videos per Agent k (for training π)	100
Sampling Temperature	1
Seconds between actions	0.25
Observations in Short-term Memory (w_s)	64
Seconds between observations	0.5
$\mathcal{T}$ hidden dimension	1024
ϕ hidden dimension	768
HDBSCAN minimum samples	30

Table 5. Hyperparameters and Settings.

I. Notation Glossary

Symbol	Description
Spaces and environment	
$\mathcal{E}$	Environment
$\mathcal{A}$	Action space
$\mathcal{O}$	Observation space
$a \in \mathcal{A}$	Action
$o \in \mathcal{O}$	Observation
Trajectories	
τ	Observation-action trajectory $(o_1, a_1, \ldots, o_T, a_T)$
τ^{obs}	Observation-only trajectory $(o_1, \ldots, o_T)$
τ_n	Trajectory of agent n
$\tau_{1\sim N}$	Trajectories all N agents: $(\tau_1, \ldots, \tau_N)$
$\tau^{obs}_{t:t+w_s}$	Short-term sample $o_{t:t+w_s}$
Policy and models	
$\pi \sim p_\theta(a_{t+1} \mid \cdot)$	Policy
ϕ	Image tokenizer (SigLIP)
$\mathcal{T}$	Causal transformer backbone
IDM	Inverse dynamics model
$\tilde{\boldsymbol{A}}$	Pseudo-action labels from IDM
$\boldsymbol{W}_o$	Output projection matrix
$\boldsymbol{H}$	Hidden states of $\mathcal{T}$
$\mathcal{L}$	Training loss
Memory	
w_s	Short-term memory window size (number of observations)
w_l	Long-term memory window size (number of key moments)
ρ	Long-term memory (most recent w_l key moments)
$\boldsymbol{T}^s$	Short-term memory tokens from ϕ
$\boldsymbol{T}^\ell$	Long-term memory tokens from ϕ
$\mathcal{K}(\tau^{obs}, t)$	Key-moment classifier
$\mathcal{M}_n$	Memory bank of agent n
Δ	Stuck-timer threshold (steps before bootstrap)
Data and retrieval	
$\mathcal{D}^I$	Internet video dataset
$\mathcal{D}^R$	Retrieved video dataset
$\mathcal{D}^\pi$	Policy update dataset
$\boldsymbol{E} \in \mathbb{R}^{e\times d}$	Per-video DINOv2 embedding matrix
$\boldsymbol{Q}$	Per-trajectory query embedding matrix
$\boldsymbol{S} = \boldsymbol{Q}\boldsymbol{E}^\top$	Similarity matrix
w_r	Retrieval window size
k	Number of retrieved videos per agent
$c_{\min}$	Minimum unique trajectories in a cluster
ASH	
N	Number of parallel agents

Table 6. Notation glossary.

J. Prompts for Zero-shot Foundation Model Baselines

Pokémon Emerald

```
You are playing Pokemon Emerald on a Game Boy Advance emulator.
You will be shown a sequence of up to 64 screenshots captured every 0.5
seconds of game time, ordered from oldest to most recent. The last image
is the current frame. Use the sequence to understand what has been
happening and choose the single best next action.
You must call the press_button tool every step.

Button reference:
- up / down / left / right: Move the character on the overworld; navigate
  cursor in menus and battle.
- a: Confirm a menu selection; interact with NPCs, signs, and objects;
  advance dialogue; select a move in battle.
- b: Cancel or close a menu; hold to run (after obtaining Running Shoes);
  skip or speed through dialogue.
- start: Open the main pause menu (Pokemon, Bag, Save, etc.).
- select: Register an item for quick use (after unlocking); swap menu
  shortcuts.
- none: Do nothing this frame (use when waiting for an animation or
  cutscene to finish).

{visual context}

The 64 image(s) above show the last 32 seconds of gameplay.
What is the best action to take next?
```

Legend of Zelda

```
You are playing The Legend of Zelda on a Game Boy Advance emulator.
You will be shown a sequence of up to 64 screenshots captured every 0.5
seconds of game time, ordered from oldest to most recent. The last image
is the current frame. Use the sequence to understand what has been
happening and choose the single best next action.
You must call the press_button tool every step.

Button reference:
- up / down / left / right: Move Link in that direction on the overworld
  or navigate menus.
- up+right / up+left / down+right / down+left: Move Link diagonally.
- a: Attack with sword; confirm menu selection; interact with NPCs and
  objects; advance dialogue.
- b: Use the currently equipped item (bombs, bow, etc.); cancel menus.
- up+r / right+r / left+r / down+r: Dash in that direction (hold shield
  while dashing).
- start: Open the pause/map menu.
- select: Open the item select screen or switch equipped items.
- none: Do nothing this frame (use when waiting for an animation or
  cutscene to finish).

{visual context}

The 64 image(s) above show the last 32 seconds of gameplay.
What is the best action to take next?
```

K. Search Keywords

Table 7 lists the search terms used to create our internet dataset. To ensure that we index multi-part walkthroughs, we replace {n} with the values 1-80.

Pokémon Emerald	**Legend of Zelda: The Minish Cap**
pokemon emerald full walkthrough	the minish cap full walkthrough
pokemon emerald longplay	legend of zelda the minish cap full walkthrough
pokemon emerald walkthrough	the legend of zelda the minish cap full walkthrough
pokemon emerald gameplay	the minish cap longplay
pokemon emerald playthrough	legend of zelda the minish cap longplay
pokemon emerald full game	the minish cap walkthrough
pokemon emerald full gameplay	legend of zelda the minish cap walkthrough
pokemon emerald all parts	the minish cap gameplay
pokemon emerald complete	legend of zelda the minish cap gameplay
pokemon emerald full version	the minish cap playthrough
pokemon emerald no commentary	legend of zelda the minish cap playthrough
pokemon emerald 100 percent	the minish cap full game
pokemon emerald part {n}	legend of zelda the minish cap full game
pokemon emerald longplay part {n}	the minish cap full gameplay
pokemon emerald let's play part {n}	legend of zelda the minish cap full gameplay
pokemon emerald episode {n}	the minish cap all parts
pokemon esmeralda guía completa	legend of zelda the minish cap all parts
pokemon esmeralda longplay	the minish cap complete
pokemon esmeralda guía	legend of zelda the minish cap complete
pokemon esmeralda gameplay	the minish cap full version
pokemon esmeralda recorrido	legend of zelda the minish cap full version
pokemon esmeralda juego completo	the minish cap no commentary
pokemon esmeralda gameplay completo	legend of zelda the minish cap no commentary
pokemon esmeralda todas las partes	the minish cap 100 percent
pokemon esmeralda completo	legend of zelda the minish cap 100 percent
pokemon esmeralda versión completa	the minish cap 100%
pokemon esmeralda sin comentarios	legend of zelda the minish cap 100%
pokemon esmeralda 100%	the minish cap part {n}
pokemon esmeralda parte {n}	the minish cap longplay part {n}

Table 7. Search terms used for creating internet dataset.

L. Broader Impact

Positive. By enabling agents to acquire new skills from widely available observational data, methods such as ASH could reduce the cost of developing embodied systems for domains including robotics, assistive technologies, simulation-based training, and autonomous exploration, where hand-engineered rewards or large labeled datasets are often impractical. More broadly, scalable long-horizon learning may accelerate the development of adaptive systems that can operate in complex real-world environments.

Negative. Systems capable of autonomously acquiring behaviors from internet-scale demonstrations may inherit biases, unsafe behaviors, or undesirable strategies present in online data. In addition, increasingly capable embodied agents could be applied in ways that reduce human oversight in safety-critical settings or be adopted for harmful applications. Our experiments are limited to video game environments, which provide a safe and controlled testbed, but future deployment in physical domains will require robust safeguards.