ASH: Agents that self-hone via Embodied Learning

by

Benjamin Schneider

A thesis
presented to the University of Waterloo
in fulfillment of the
thesis requirement for the degree of
Master of Mathematics
in
Computer Science

Waterloo, Ontario, Canada, 2026

© Benjamin Schneider 2026

Author's Declaration

This thesis consists of material all of which I authored or co-authored: see Statement of Contributions included in the thesis. This is a true copy of the thesis, including any required final revisions, as accepted by my examiners.

I understand that my thesis may be made electronically available to the public.

Statement of Contributions

This thesis is adapted from a preprint [1]. I contributed the original idea, writing/figures, and co-implemented with Xavier Schneider. Prof. Victor Zhong and Sun Sun contributed feedback and (a lot of) project guidance.

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.

Acknowledgements

Thank you to all the folks who ahve supported me throughout my reasearch with their time, effort and talent. In particular, Prof. Florian Kerschbaum for giving me my first research epxerience. Prof. Wenhu Chen for introducing me to the wonderful world of multimodal learning. Sun Sun and Prof. Victor Zhong for their supervision and feedback on the work that eventually became this thesis. Lastly, to my mom and dad, for always supporting me on my adventures, even if they involve spending a year training AI models to play Pokémon.

Table of Contents

Author's Declaration ii

Statement of Contributions iii

Abstract iv

Acknowledgements v

List of Figures ix

List of Tables xi

List of Abbreviations xiii

List of Symbols xv

1 Introduction 1

2 Related Work 4

2.1 Behavioral cloning and internet-scale demonstrations 4

2.2 Retrieval-augmented decision-making 5

2.3 Self-improvement from model-generated data 6

2.4 Inverse dynamics models 6

2.5 Long-horizon control and agent memory . . . 7
2.6 RL in open-ended environments . . . 8
2.7 Catastrophic forgetting in continual learning . . . 9

3 ASH **10**
3.1 Inference . . . 13
3.2 Retrieval . . . 14
3.3 Key Moments and Long-term Memory . . . 15
3.4 Bootstrapping . . . 16

4 Environments and Milestones **18**
4.1 Background on Environments . . . 18
4.2 Milestones . . . 20

5 Experiments **23**
5.1 Performance on Milestones . . . 25
5.2 Component ablations . . . 27
5.3 IDM adaptation . . . 28
5.4 Retrieval quality . . . 29
5.5 Cluster Analysis . . . 31
5.6 Offline Replay . . . 31

6 Qualitative Results **33**

7 Limitations and Conclusion **36**

References **37**

APPENDICES **52**

A ASH **53**

B Hyperparameters and Training Settings **55**

C Notation Glossary **57**

D Prompts for Zero-shot Foundation Model Baselines **60**

E Search Keywords **62**

F Broader Impact **64**

List of Figures

1.1 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. 2

3.1 An overview of ASH's self-improvement loop. 10

5.1 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 Section 5.1 for standard deviation. . 24

5.2 Outside (top) vs. inside (bottom) of the Zelda castle: offline policies collapse on this dynamics shift; ASH bootstraps and continues. 26

5.3 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. 27

5.4 IDM accuracy across bootstraps, evaluated on a test set. The dashed line is the pre-bootstrap initialization checkpoint. 28

5.5 Visualization of 3 HDBSCAN [64] 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 [64]. 30

5.6 Offline replay of the final ASH checkpoint vs. the original online run. . . . 32

6.1 **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)**. . 34

6.2 **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)**. 35

List of Tables

4.1 Milestones, icons and corresponding descriptions for the Pokémon Emerald environment. 21

4.2 Milestones, icons and corresponding descriptions for the Legend of Zelda environment. 22

5.1 Average milestones reached (out of 12) at the end of the 8-hour evaluation, reported as mean ± one standard deviation. 26

5.2 Annotator judgments of ASH retrieval and baselines: binary relevance (yes/no) and average rank across methods (lower is better). 29

B.1 Hyperparameters and Settings. 56

C.1 Notation glossary (part 1 of 2). 58

C.2 Notation glossary (part 2 of 2). 59

E.1 Search terms used for creating internet dataset. 63

List of Abbreviations

AI Artificial Intelligence

BC Behavioral Cloning

GBA Game Boy Advance

GPU Graphics Processing Unit

HDBSCAN Hierarchical Density-Based Spatial Clustering of Applications with Noise

IDM Inverse Dynamics Model

LLM Large Language Model

NPC Non-Player Character

PPO Proximal Policy Optimization

RAEA Retrieval-augmented Embodied Agent

RL Reinforcement Learning

RPG Role-Playing Game

VPT Video PreTraining

List of Symbols

$\mathcal{E}$ Environment

$\mathcal{A}$ Action space

$\mathcal{O}$ Observation space

$a \in \mathcal{A}$ Action

$o \in \mathcal{O}$ Observation

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 $o_{t:t+w_s}$

$\pi \sim p_\theta(a_{t+1} \mid \cdot)$ 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

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)

$\mathcal{D}^I$ 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

N Number of parallel agents

Chapter 1

Introduction

Developing agents capable of mastering long-horizon, open-ended tasks remains a fundamental challenge in artificial intelligence [2, 3]. 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 [4] 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 [5]. 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 [6]: 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 [7].

While Reinforcement Learning (RL) has demonstrated success in complex environments, it remains heavily dependent on dense reward engineering [8, 9, 5]. 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 [9, 5], 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.



Figure 1.1: 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.

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.

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 [5], retrieval-augmented [10], and zero-shot foundation-model [11] 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.

Chapter 2

Related Work

2.1 Behavioral cloning and internet-scale demonstrations

Behavioral cloning (BC) trains a policy via supervised learning to predict expert actions from observations. When paired with high-capacity policies and large action-labeled datasets, BC has produced strong results in robotic manipulation [12, 13], multi-task generalist agents [14], and vision-language-action models that transfer pretrained representations into control [15, 16]. Recent work pools teleoperated demonstrations across embodiments and tasks to amortize the cost of collection [17, 18]. BC is also known to suffer from compounding errors when the learner drifts off the expert distribution, which DAgger and its descendants address by iteratively querying the expert on states the learner visits [6]. The central bottleneck for all of these methods, however, is data: collecting action-labeled trajectories at the scale needed for open-ended, long-horizon tasks is prohibitively expensive, since every demonstration requires a controlled rig or a paid expert.

To address this data bottleneck, recent methods train on *observation-only* video harvested from the internet, where actions are absent but visual data is abundant. VPT [5] creates pseudo-actions for thousands of hours of Minecraft video with an Inverse Dynamics Model (IDM) trained on contractor-collected gameplay [4], and remains the strongest reported Minecraft BC backbone [19]. SIMA extends this recipe to a curated mixture of 3D games, training a single instruction-following agent across diverse environments [20, 21]. A parallel *imitation-from-observation* literature recovers actions without an explicit IDM, instead learning latent action spaces directly from video [22, 23, 24, 25], an idea later extended to generative world models that synthesize controllable trajectories from video alone [26].

ASH departs from these approaches: 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. This dynamic loop allows ASH to continually address gaps in its current policy at training time, rather than committing upfront to which demonstrations are likely to be useful.

2.2 Retrieval-augmented decision-making

Retrieval-augmented methods couple a parametric model with an external datastore that is queried at inference time, allowing the model to access information that is too large or domain-specific to fit in its weights. This idea took shape in language modeling with RAG [27], REALM [28], RETRO [29], and Atlas [30], which condition a generator on passages retrieved from a large text corpus to ground its predictions on external knowledge. The same recipe underpins modern LLM agents that operate over open-ended tasks: ReAct [31] interleaves reasoning steps with retrieved tool outputs, and Reflexion [32] retrieves verbalized reflections on prior failures to steer subsequent attempts.

Closer to the embodied setting, retrieval has been used to condition policies on prior experience rather than text. Humphreys et al. [33] and Goyal et al. [34] augment value-based and policy-gradient RL agents with a learned retriever over past trajectories, demonstrating gains in multi-task and Atari domains. RAEA [10] extends this idea to robotic manipulation, equipping policies with a shared memory bank that is retrieved over multi-modal inputs. For LLM agents, retrieval over self-generated experience underlies ExpeL [35], Voyager [36], and generative agents [37], which build a memory of insights, skills, or observations indexed by relevance, recency, and importance. Zhang et al. [38] cast LLM agents as semi-parametric RL agents whose policy depends jointly on weights and a retrieved memory buffer.

In all of 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. Where prior work treats the retrieved set as an inference-time context window, ASH uses retrieval for a targeted parametric update, allowing it to learn a broad set of new, contextually relevant behaviors.

2.3 Self-improvement from model-generated data

Self-improvement methods bootstrap a learner from data produced by an earlier version of itself, sidestepping the need for additional human supervision. Anthony et al. [39] and Silver et al. [40] use self-play to create models capable of super-human level Go and chess play. In language modeling, STaR [41] and ReST [42] bootstrap reasoning by fine-tuning on model-generated rationales filtered by correctness, and SPIN [43] shows that a single self-play loop can convert a weak instruction-tuned model into a stronger one without additional preference data.

In embodied and agentic settings, self-improvement faces an additional challenge: there is rarely a clean verifier of trajectory quality. Existing approaches address this by introducing one. AutoRT [44] uses VLMs to score robot rollouts before distilling them back into the policy, Eureka [45] has an LLM author and iteratively refine reward code that drives downstream RL, and Voyager [36] relies on Minecraft's programmatic action feedback to validate skills before adding them to its library. Each of these systems is bottlenecked by the availability of a reliable external signal — a learned reward model, a coded verifier, or a structured environment API — and degrades in open-ended environments where such signals are unavailable or expensive to engineer [46].

ASH's improvement comes from retrieved internet video rather than from an external verifier or reward model. By selecting human gameplay relevant to the agent's current trajectory and re-labeling it with an updated IDM, ASH produces new training data without requiring any reward, verifier, or hand-crafted skill library.

2.4 Inverse dynamics models

An inverse dynamics model (IDM) predicts the action that connects two consecutive states, $a_t = \text{IDM}(s_t, s_{t+1})$, providing a way to recover labels for trajectories where actions were not recorded. IDMs were popularized in deep RL by Pathak et al. [47], who paired an IDM with a forward model to provide a curiosity reward from prediction error. Subsequent work repurposed them as a way to recover action labels for observation-only trajectories: Torabi et al. [22] trained an IDM on random rollouts to label demonstrations and showed it could match expert-action BC on continuous control, and Edwards et al. [23] jointly learned a latent policy and an inverse model from observation sequences alone.

In recent embodied settings, IDMs have become the standard mechanism for converting internet video into supervision. VPT [5] trains an IDM on a few thousand hours of

contractor-collected Minecraft gameplay, then uses it to label hundreds of thousands of hours of YouTube video, producing the strongest reported Minecraft BC backbone [19]. LAPO [24] sidesteps the action-labeled bootstrap altogether by learning a *latent* action space from video alone, and Genie [26] pushes this further by training a generative world model whose latent actions are interpretable as control signals. Common to all of these is that the IDM is treated as a one-shot artifact: trained once on a static collection of action-labeled gameplay, then frozen for downstream labeling.

ASH treats the IDM as a moving target rather than a fixed pre-processor. Its IDM is updated on the agent's own trajectories as the policy explores new regions of the game's state space, so the labels available to downstream BC grow more accurate in exactly the regions the agent is currently struggling with. This couples the IDM's training distribution to the policy's, addressing a known failure mode where a statically trained IDM mislabels states the contractor data did not cover.

2.5 Long-horizon control and agent memory

Long-horizon control requires agents to retain information that may become useful much later. Full-history conditioning is computationally intractable, while short fixed-length memory is efficient but can lose information needed for planning [7]. Most embodied BC and RL approaches fall back on bounded-memory policies: recurrent architectures that compress history into a fixed-size hidden state, and transformers that condition on a fixed context window of recent observations and actions [48, 5, 9].

External memory augmentation provides an alternative to bounded-context policies, attaching a queryable store that can be read or written at any timestep. In RL, MERLIN [49] learns to write predictive representations into an episodic memory and read them back when partially observable tasks demand recall across long durations. The same idea has been ported into language modeling: Memorizing Transformers [50], the Recurrent Memory Transformer [51], and RETRO [29] attach a retrievable cache of prior keys or passages to a standard attention stack, while MemGPT [52] treats the context window itself as a managed memory hierarchy, paging information between an in-context working memory and an out-of-context disk memory under the model's own control.

When an agent acts in an environment over time, memory must also store *episodic* traces: timestamped observations, prior plans, and outcomes of past attempts. Generative agents [37] maintain a stream of natural-language observations indexed by recency, relevance, and importance, and periodically reflect over the stream to consolidate higher-level memories. ExpeL [35] accumulates a library of insights from successful and failed

trajectories, Voyager [36] stores reusable code-as-action skills, and Reflexion [32] maintains verbalized self-reflections that are prepended to subsequent attempts. Across these systems, the memory contents are either authored by the LLM itself or shaped by a hand-engineered curator that decides what is worth keeping.

Rather than being authored by an LLM or hand-engineered, ASHautonomously discovers "key moments" from internet data by unsupervised clustering of trajectory embeddings, and retains those moments as long-term memory when the agent re-encounters them in the environment. This sidesteps the need for either a hand-designed task abstraction or a learned write-controller, and grounds the memory in observed human behavior rather than self-generated text.

2.6 RL in open-ended environments

RL has produced strong results in settings where the reward structure is unambiguous: Atari arcade scores [53], self-play objectives in board games [40], league play in real-time strategy [54], team rewards in multiplayer games [55], and model-based optimization of an environment-given return [9]. These successes share a common feature: a scalar feedback signal that can be optimized at every step or rollout, either provided by the environment or constructed from a well-defined adversarial objective.

Open-ended environments rarely supply such a signal. Progress is distributed across long horizons and not naturally captured by a scalar, so practitioners typically fall back on reward shaping. Pleines et al. [8] train a PPO agent on Pokémon Red by manually creating a trail of intermediate 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. Reward shaping is also vulnerable to specification gaming, where agents satisfy the proxy without solving the underlying task. Verifier-based RL extends this approach by relying on programmatic correctness signals — for instance, Shao et al. [56] train math-reasoning policies against a unit-test-style verifier — but this assumes that solutions are cheap to check, which is rarely true for embodied behaviour.

By contrast, ASH's training signal comes from unlabeled demonstrations rather than a designed reward, offering a scalable approach to open-ended embodied learning that does not require the engineer to anticipate the agent's failure modes in advance.

2.7 Catastrophic forgetting in continual learning

Neural networks trained sequentially on new tasks tend to overwrite the knowledge required for earlier ones, a phenomenon known as *catastrophic forgetting* [57]. Continual-learning methods address this by constraining how the network updates. Elastic Weight Consolidation [58] penalizes movement of parameters important for past tasks; Learning without Forgetting [59] distills the previous model's outputs into the new one to anchor old behaviors; and replay-based methods such as iCaRL [60] interleave a small stored set of past examples with the new data. These approaches trade off plasticity (the ability to learn the new task) against stability (the retention of old tasks), and the right balance depends on whether the deployment phase requires uniform competence across all tasks or growing competence on the latest one.

Continual learning is especially fragile in RL, where the data distribution is non-stationary by construction and a brief failure mode can undo many episodes of progress. CLEAR [61] stabilizes off-policy continual RL by mixing replayed experiences from earlier tasks with on-policy rollouts, and Progress & Compress [62] pairs a fast task-specific learner with a slow knowledge-base network that distills past skills. Survey work on continual RL highlights that, unlike supervised continual learning, agents also face shifting transition dynamics and reward functions across tasks, which compounds the forgetting problem [63].

ASH faces a milder version of this problem: its policy is fine-tuned iteratively on freshly retrieved demonstrations, which risks overwriting skills relevant to earlier portions of the game. The long-term memory of key moments offers some natural mitigation, since memory entries persist across iterations and the policy is repeatedly exposed to earlier-stage observations whenever the agent re-encounters them in the environment. A systematic study of forgetting dynamics in ASH is left to future work.

Chapter 3

ASH

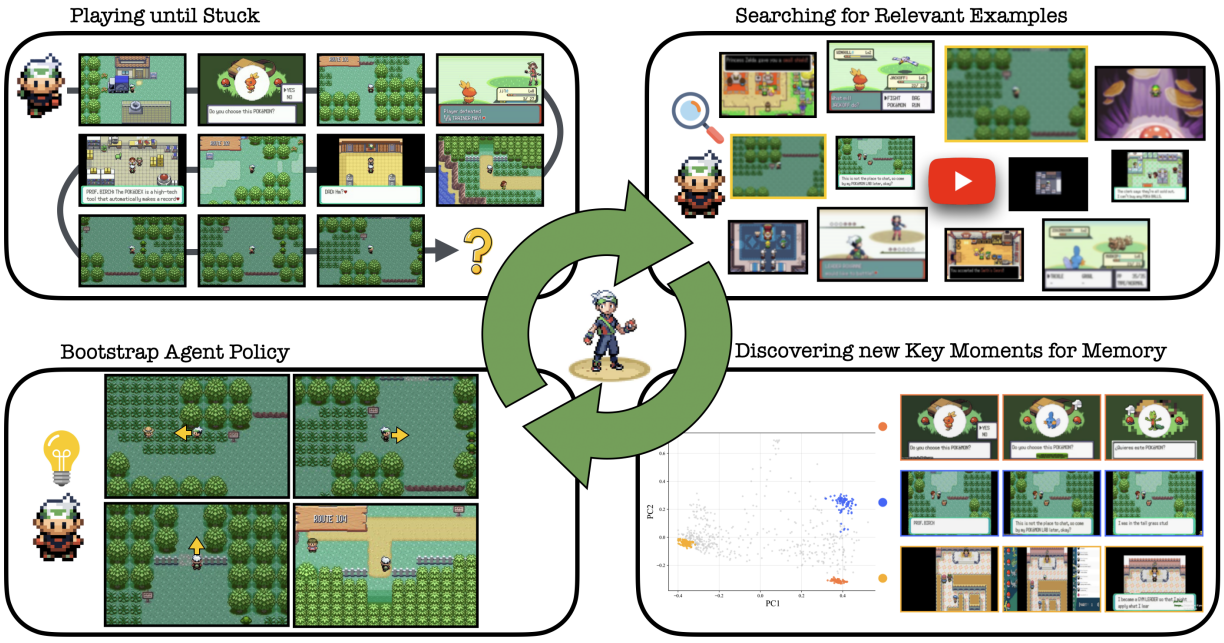


Figure 3.1: An overview of ASH's self-improvement loop.

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 Chapter C.

Behavioral cloning (BC) [4] trains π via supervised learning using cross-entropy loss on observation-action trajectories. However, scalable observation-only sources such as YouTube [5] 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 [5].

We use HDBSCAN [64] 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.

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**.

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 Chapter A.

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 [65]), an IDM with VPT's bidirectional-attention architecture [5], and the causal transformer block architecture from Yang et al. [66], 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 $\tau^{obs}_{t:t+w_s} = (o_t, o_{t+1}, \ldots, o_{t+w_s-1})$.

$$\tilde{\boldsymbol{A}} = \mathrm{IDM}(\tau^{obs}_{t:t+w_s}), \tag{3.1}$$

$$\boldsymbol{T}^s, \boldsymbol{T}^\ell = \phi(\tau^{obs}_{t:t+w_s}, \rho) \tag{3.2}$$

$$\boldsymbol{H} = \mathcal{T}\Big(\Big[\boldsymbol{T}^\ell,\ \boldsymbol{T}^s_1, \tilde{\boldsymbol{A}}_1,\ \ldots,\ \boldsymbol{T}^s_{w_s}, \tilde{\boldsymbol{A}}_{w_s}\Big]\Big) \tag{3.3}$$

$$\mathcal{L} = -\sum_{j=0}^{w_s} \tilde{\boldsymbol{A}}_j^\top \log \mathrm{softmax}(\boldsymbol{W}_o \boldsymbol{H}_{w_l+2j+1})\,. \tag{3.4}$$

First, we use the IDM to create pseudo-actions for each observation in the sample $\tau^{obs}_{t:t+w_s}$ (Equation (3.1)), 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 (Equation (3.2)). 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 $\mathcal{T}$ (Equation (3.3)), 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 $\mathcal{L}$ (Equation (3.4)), the cross-entropy loss between predictions generated by ASH and pseudo-actions generated by our IDM.

3.1 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.

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 $\mathcal{E}_{1\sim N}$

$C_{1\sim N} \leftarrow 0$ `# per-agent stuck timers`
while $\max_n C_n < \Delta$ **do**
  for $n = 1$ to N **do**
    $\rho \leftarrow \mathcal{M}_{n,-w_l:}$ `# last` w_l `key moments`
    $a \sim \pi(\cdot \mid \tau_{n,-w_s:}, \rho)$ `# sample action from policy`
    $o \leftarrow \mathcal{E}_n(a)$ `# step the environment`
    $\tau_n \leftarrow \tau_n \cup \{(o, a)\}$
    if $\mathcal{K}(\tau_n, |\tau_n|)$ **then** `# new key moment?`
      $\mathcal{M}_n \leftarrow \mathcal{M}_n \cup \{o\}$ `# append to memory bank`
      $C_n \leftarrow 0$ `# reset stuck timer`
    end if
    $C_n \leftarrow C_n + 1$
  end for
end while
return $\tau_{1\sim N}, \mathcal{M}_{1\sim N}$

3.2 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. We index our internet dataset using DINOv2 [67] 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 5.4.

Algorithm 3 Retrieval

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

$\mathcal{D}^R \leftarrow \emptyset$ # retrieved dataset
for $n = 1$ to N **do**
  $\boldsymbol{Q} \leftarrow \textsc{Embed}(\tau_{n,-\delta:})$ # query embedding (DINOv2)
  scores $\leftarrow \emptyset$
  for $\tau^{\text{obs}} \in \mathcal{D}^I$ **do**
    $\boldsymbol{E} \leftarrow \textsc{Embed}(\tau^{\text{obs}})$ # internet video embedding
    $e \leftarrow \textsc{Rows}(\boldsymbol{E})$
    $\boldsymbol{S} \leftarrow \boldsymbol{Q}\boldsymbol{E}^\top$ # similarity matrix
    $z^\star \leftarrow -\infty$
    for $t \in \{1, \dots, e - w_r + 1\}$ **do** # sliding window of length w_r
      $s \leftarrow \boldsymbol{S}_{:,\,t:t+w_r}$
      $z \leftarrow 0$
      for $j = 1$ to w_r **do** # greedy 1-to-1 match
        $r^\star \leftarrow \arg\max_r s_{r,j}$
        $z \leftarrow z + s_{r^\star,j}$
        $s_{r^\star,:} \leftarrow -\infty$
      end for
      $z^\star \leftarrow \max(z^\star, z)$
    end for
    scores $\leftarrow$ scores $\cup \{(\tau^{\text{obs}}, z^\star)\}$
  end for
  $\mathcal{D}^R \leftarrow \mathcal{D}^R \cup \{\text{top-k(scores)}\}$ # keep top-k per agent
end for
return $\mathcal{D}^R$

3.3 Key Moments and Long-term Memory

Discovering key moments. $\mathcal{K}$ is built by applying the HDBSCAN [64] 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 Section 5.5.

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 [67] 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.

3.4 Bootstrapping

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 $\mathcal{D}^R$ `# re-cluster key moments`
Update IDM on $\tau_{1\sim N}$ `# adapt to new dynamics`
$\mathcal{D}^\pi \leftarrow \emptyset$ `# policy training set`
for $\tau^{obs} \in \mathcal{D}^R$ **do**
  for $t = 1$ to $|\tau^{obs}| - w_s + 1$ step w_s **do** `# non-overlapping samples`
    $\tilde{\boldsymbol{A}} \leftarrow \text{IDM}(\tau^{obs}_{t:t+w_s})$ `# pseudo-actions`
    $\bar{\rho} \leftarrow [\, \tau_i^{obs} : i < t, \mathcal{K}(\tau^{obs}, i)\,]$ `# key moments before` t
    $\mathcal{D}^\pi \leftarrow \mathcal{D}^\pi \cup \{(\tau^{obs}_{t:t+w_s}, \tilde{\boldsymbol{A}}, \bar{\rho}_{-w_l:})\}$
  end for
end for
Update π on $\mathcal{D}^\pi$ `# train policy`
return π, IDM, $\mathcal{K}$

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.

Chapter 4

Environments and Milestones

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.

4.1 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 [68] 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 [69] provides a detailed walkthrough of the game, including areas, game mechanics, and the storyline of Link's quest.

Legend of Zelda: The Minish Cap is a fascinating environment for several reasons: (1) As with our previous environment, it is long-horizon, with progression spanning many hours and distinct areas. (2) *Progressing requires that the player demonstrates a variety of skills.* The Legend of Zelda is not a game where the player can simply "walk to the finish line," progressing through each zone requires demonstrating *escalating levels of mastery* over the game's core combat mechanics (centered around a sword and shield), the ability to traverse movement challenges, and utilize newly-introduced game mechanics to solve puzzles. (3) Lastly, *the environment rewards open-ended exploration.* As the player is tasked with a series of progressively harder challenges, they can explore the environment to find power-ups that make these challenges easier. For example, the player can obtain *heart containers* which increase the amount of damage the player can take without dying in combat.

4.2 Milestones

We evaluate our agents using *milestones*, which are human-defined markers of progress in our environments. We derive our milestones from online walkthroughs [68, 69]. Per-environment milestones, icons, and descriptions are listed in Tables 4.1 and 4.2.

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 4.1: 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).

Table 4.2: Milestones, icons and corresponding descriptions for the Legend of Zelda environment.

Chapter 5

Experiments

We evaluate ASH on the two long-horizon environments described in Chapter 4: Pokémon Emerald and *The Legend of Zelda: The Minish Cap*. Through our experiments, we investigate the following:

- How does ASH compare to other adaptive and non-adaptive baselines? (Section 5.1)
- What effect do long-term memory and dynamic bootstrapping have on ASH's policy? (Section 5.2)
- What does bootstrapping do to the IDM? (Section 5.3)
- Does ASH retrieve relevant, high-quality examples from internet data? (Section 5.4)
- Does ASH's clustering identify high-quality key moments? (Section 5.5)
- Can ASH's final policy replay the environment offline? (Section 5.6)

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 [70] hyperparameter, $c_{\min}$, to 30. We scrape our internet datasets ($\mathcal{D}^I$) from YouTube using a list of keywords (Chapter E). 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

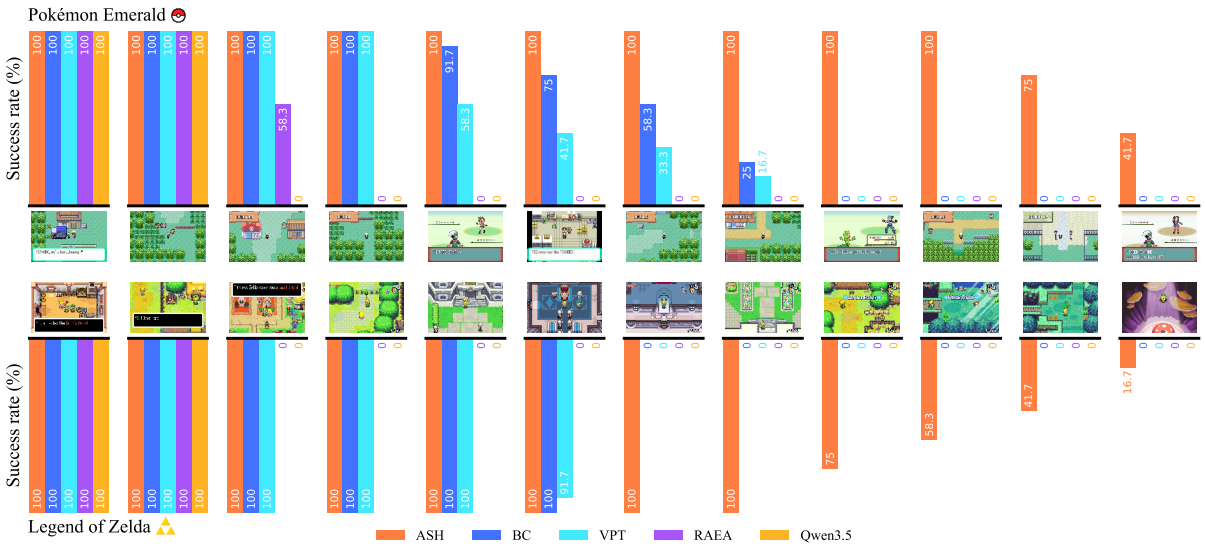


Figure 5.1: Agent progress in **Pokémon Emerald** (top) and **Legend of Zelda** (bottom), measured using milestone completion rates. ASH is able to adapt and continue progressing throughout the 8-hour gameplay period. While all methods are able to complete early milestones, only ASH can adapt to new areas, objectives, and mechanics. See Section 5.1 for standard deviation.

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 Chapter B.

Baselines. We compare ASH with the following baselines. **Qwen3.5** [11]: 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** [5]: 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 5.3. 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)** [10]: We follow the recipe of Zhu et al. [10] 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.

5.1 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 5.1. 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 5.2): 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 5.2: 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 Chapter 6.

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.

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 5.1: Average milestones reached (out of 12) at the end of the 8-hour evaluation, reported as mean ± one standard deviation.

Table 5.1 reports the average number of milestones reached (out of 12) at the end of the 8-hour evaluation. ASH clears nearly the entire milestone set in both environments, averaging 11.17 ± 0.83 in Pokémon Emerald and 9.92 ± 1.51 in The Legend of Zelda. The closest baselines, offline BC and VPT, stall around the halfway mark (5.5–6.5 milestones in

either environment), reflecting their inability to adapt once new game dynamics appear. RAEA and Qwen3.5 fail to progress beyond the earliest milestones, averaging roughly 2/12 in both games. The five-milestone gap between ASH and the strongest baseline quantifies the cost of running a fixed policy in long-horizon environments: every plateau visible in Figure 5.1 translates directly into uncovered milestones at the eight-hour mark. The Qwen and RAEA baselines consistently fail on early game milestones, resulting in low standard standard deviation.

5.2 Component ablations

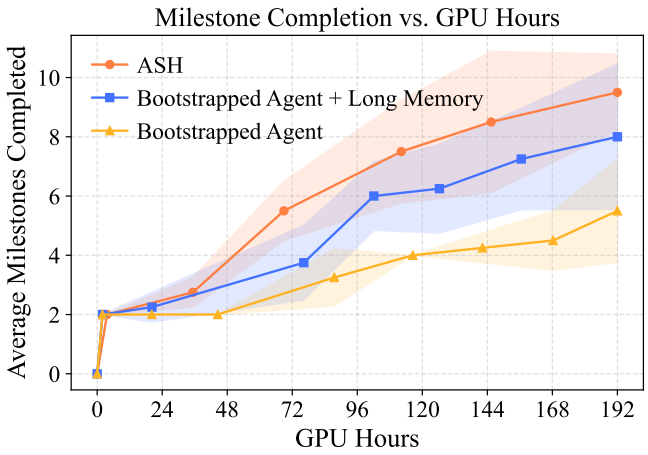


Figure 5.3: Component ablation on Pokémon Emerald: each addition (long-term memory, dynamic bootstrapping) yields a clear gain in milestones completed per GPU hour of online training. Shaded regions are one standard deviation over 4 trajectories per method.

In Figure 5.3, 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.

5.3 IDM adaptation

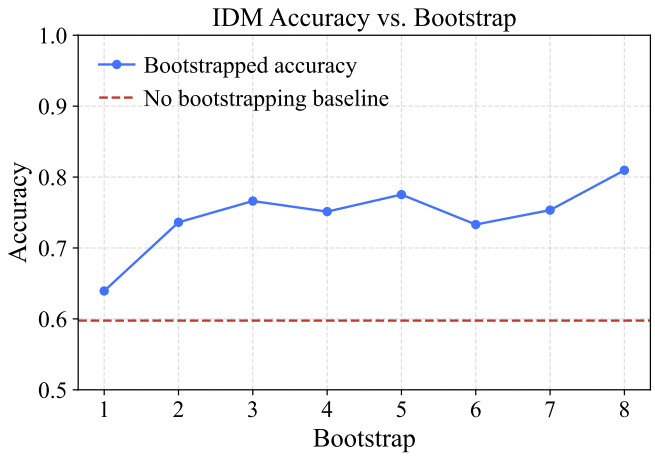


Figure 5.4: IDM accuracy across bootstraps, evaluated on a test set. The dashed line is the pre-bootstrap initialization checkpoint.

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 5.4. 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 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.

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

Table 5.2: Annotator judgments of ASH retrieval and baselines: binary relevance (yes/no) and average rank across methods (lower is better).

5.4 Retrieval quality

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 5.1). 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 5.2. 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.

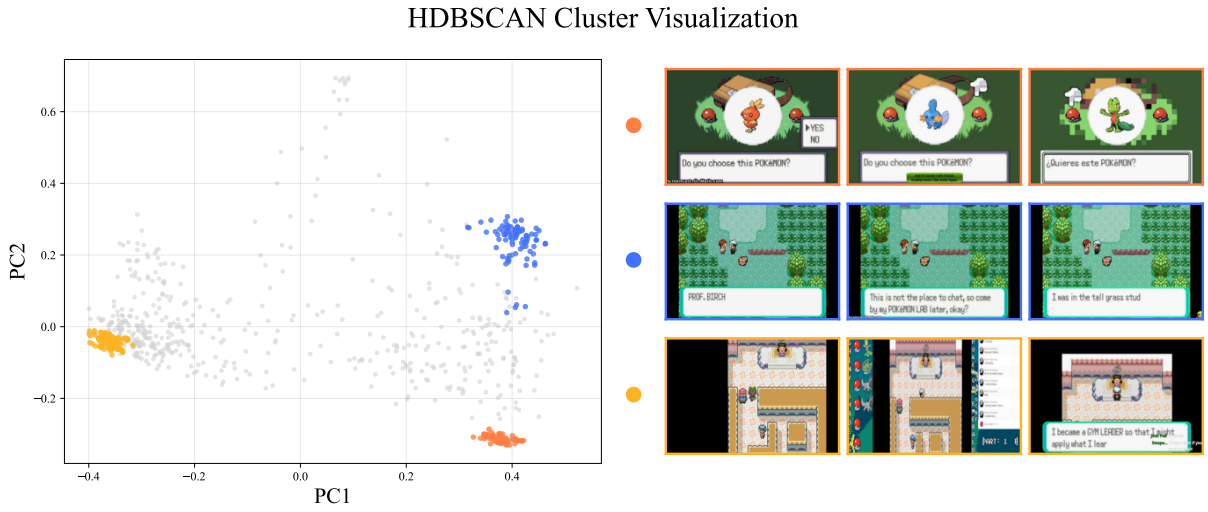


Figure 5.5: Visualization of 3 HDBSCAN [64] clusters, as well as 500 uniformly sampled outlier points reduced to 2 dimensions via principal component analysis. Each of these clusters represents a key moment in Pokémon Emerald; choosing a starter Pokémon (orange), saving the professor (blue) and challenging a gym leader (yellow). Grey dots are outliers that are not assigned to a cluster by HDBSCAN [64].

5.5 Cluster Analysis

The effectiveness of ASH's long-term memory depends on the quality of the clusters produced by HDBSCAN [64]. 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. We use principal componenet analysis to visualize key-moments discovered by HDBSCAN in Figure 5.5. To evaluate how well HDBSCAN [64] 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 [64] correspond to key moments. The effectiveness of HDBSCAN in identifying these key moments helps explain the performance increase observed in Section 5.2.

5.6 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 [71]. 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 5.6). Results are averaged over 4 trajectories and the shaded regions show the standard deviation. We find that ASH does not exhibit catastrophic 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.

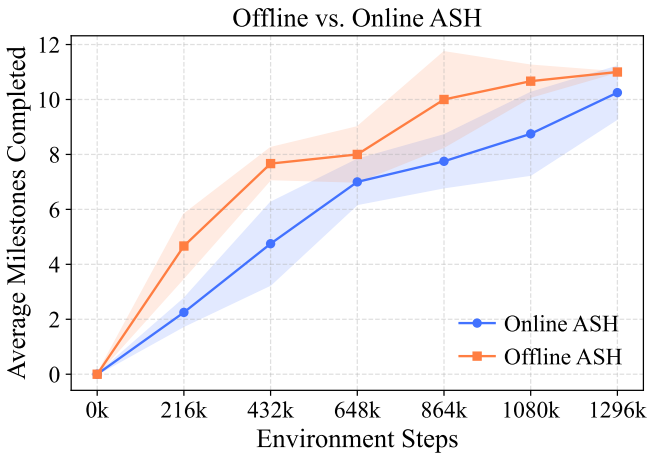


Figure 5.6: Offline replay of the final ASH checkpoint vs. the original online run.

Chapter 6

Qualitative Results

The experiments in Chapter 5 measure ASH in aggregate: milestone completion curves, component ablations, IDM accuracy, retrieval quality, and cluster purity. These results establish that **dynamic bootstrapping** and **long-term memory**, the two mechanisms introduced in Chapter 3, are responsible for ASH's ability to continue progressing where fixed-policy baselines plateau. They do not, however, show *how* either mechanism behaves within a single playthrough. This chapter presents two case studies in Pokémon Emerald that walk through these moments step-by-step. The first (Figure 6.1) shows ASH encountering a roadblock it cannot solve with its initial policy, retrieving relevant gameplay from $\mathcal{D}^I$, and bootstrapping to unstick itself. The second (Figure 6.2) shows the agent at a navigational decision point where short-term observations are insufficient, and demonstrates how the long-term memory bank provides the context required to choose correctly.

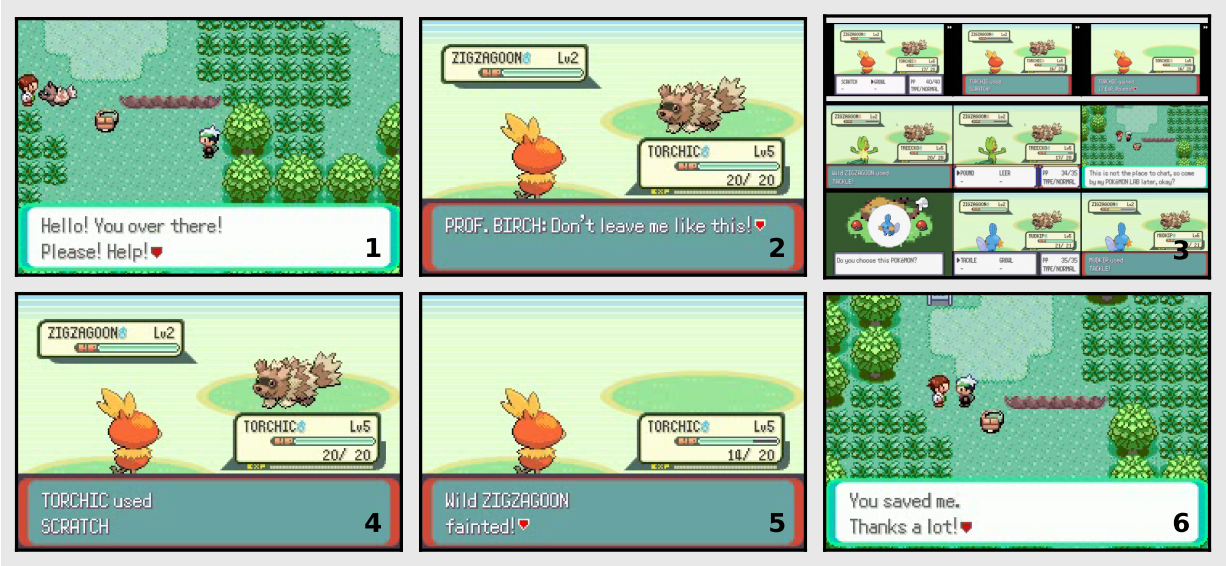


Figure 6.1: **Dynamic bootstrapping example.** To complete milestone 2, the player must rescue the Professor from a wild Zigzagoon **(Panel 1)**. To accomplish this, the player must use their starter Pokémon to defeat the Zigzagoon in battle **(Panel 2)**. However, ASH's initial policy does not know 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)**.

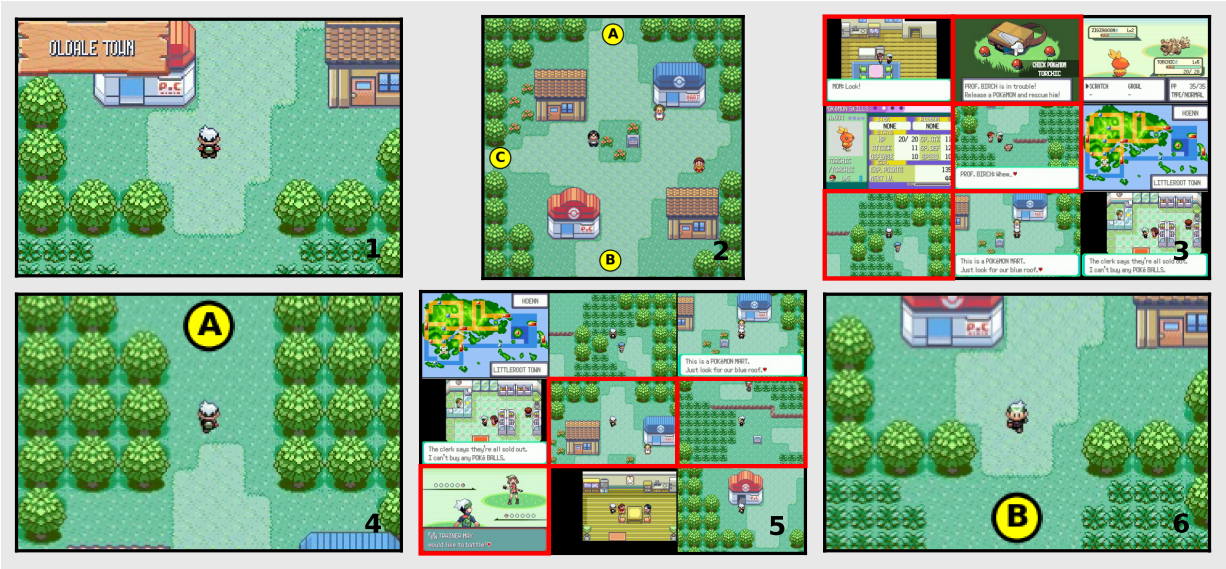


Figure 6.2: **Long-term memory example.** When the player arrives in Oldale town **(Panel 1)**, they are presented with 3 possible next paths. **Option `A`:** The player heads north to Route 103 to meet their rival, May. This is the correct choice if the player has just obtained their starter Pokémon 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)**.

Chapter 7

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

[1] Benjamin Schneider, Xavier Schneider, Victor Zhong, and Sun Sun. Ash: Agents that self-hone via embodied learning, 2026. URL https://arxiv.org/abs/2605.14211.

[2] Wei-Chieh Huang, Weizhi Zhang, Yueqing Liang, Yuanchen Bei, Yankai Chen, Tao Feng, Xinyu Pan, Zhen Tan, Yu Wang, Tianxin Wei, Shanglin Wu, Ruiyao Xu, Liangwei Yang, Rui Yang, Wooseong Yang, Chin-Yuan Yeh, Hanrong Zhang, Haozhen Zhang, Siqi Zhu, Henry Peng Zou, Wanjia Zhao, Song Wang, Wujiang Xu, Zixuan Ke, Zheng Hui, Dawei Li, Yaozu Wu, Langzhou He, Chen Wang, Xiongxiao Xu, Baixiang Huang, Juntao Tan, Shelby Heinecke, Huan Wang, Caiming Xiong, Ahmed A. Metwally, Jun Yan, Chen-Yu Lee, Hanqing Zeng, Yinglong Xia, Xiaokai Wei, Ali Payani, Yu Wang, Haitong Ma, Wenya Wang, Chenguang Wang, Yu Zhang, Xin Wang, Yongfeng Zhang, Jiaxuan You, Hanghang Tong, Xiao Luo, Xue Liu, Yizhou Sun, Wei Wang, Julian McAuley, James Zou, Jiawei Han, Philip S. Yu, and Kai Shu. Rethinking memory mechanisms of foundation agents in the second half: A survey, 2026. URL https://arxiv.org/abs/2602.06052.

[3] Yafei Hu, Quanting Xie, Vidhi Jain, Jonathan Francis, Jay Patrikar, Nikhil Keetha, Seungchan Kim, Yaqi Xie, Tianyi Zhang, Hao-Shu Fang, Shibo Zhao, Shayegan Omidshafiei, Dong-Ki Kim, Ali akbar Agha-mohammadi, Katia Sycara, Matthew Johnson-Roberson, Dhruv Batra, Xiaolong Wang, Sebastian Scherer, Chen Wang, Zsolt Kira, Fei Xia, and Yonatan Bisk. Toward general-purpose robots via foundation models: A survey and meta-analysis, 2024. URL https://arxiv.org/abs/2312.08782.

[4] Faraz Torabi, Garrett Warnell, and Peter Stone. Behavioral cloning from observation. In *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence, IJCAI-18*, pages 4950–4957. International Joint Conferences on Artificial Intelligence Organization, 7 2018. doi: 10.24963/ijcai.2018/687. URL https://doi.org/10.24963/ijcai.2018/687.

[5] Bowen Baker, Ilge Akkaya, Peter Zhokhov, Joost Huizinga, Jie Tang, Adrien Ecoffet, Brandon Houghton, Raul Sampedro, and Jeff Clune. Video pretraining (vpt): Learning to act by watching unlabeled online videos, 2022. URL https://arxiv.org/abs/2206.11795.

[6] Stephane Ross and Drew Bagnell. Efficient reductions for imitation learning. In Yee Whye Teh and Mike Titterington, editors, *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, volume 9 of *Proceedings of Machine Learning Research*, pages 661–668, Chia Laguna Resort, Sardinia, Italy, 13–15 May 2010. PMLR. URL https://proceedings.mlr.press/v9/ross10a.html.

[7] Shalev Lifshitz, Keiran Paster, Harris Chan, Jimmy Ba, and Sheila A. McIlraith. 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.

[8] Marco Pleines, Daniel Addis, David Rubinstein, Frank Zimmer, Mike Preuss, and Peter Whidden. Pokemon red via reinforcement learning, 2025.

[9] Danijar Hafner, Wilson Yan, and Timothy Lillicrap. Training agents inside of scalable world models, 2025. URL https://arxiv.org/abs/2509.24527.

[10] Yichen Zhu, Zhicai Ou, Xiaofeng Mou, and Jian Tang. Retrieval-Augmented Embodied Agents . In *2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 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.

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

[12] Pete Florence, Corey Lynch, Andy Zeng, Oscar Ramirez, Ayzaan Wahid, Laura Downs, Adrian Wong, Johnny Lee, Igor Mordatch, and Jonathan Tompson. Implicit behavioral cloning, 2021. URL https://arxiv.org/abs/2109.00137.

[13] Eric Jang, Alex Irpan, Mohi Khansari, Daniel Kappler, Frederik Ebert, Corey Lynch, Sergey Levine, and Chelsea Finn. Bc-z: Zero-shot task generalization with robotic imitation learning, 2022. URL https://arxiv.org/abs/2202.02005.

[14] Scott Reed, Konrad Zolna, Emilio Parisotto, Sergio Gomez Colmenarejo, Alexander Novikov, Gabriel Barth-Maron, Mai Gimenez, Yury Sulsky, Jackie Kay, Jost Tobias

Springenberg, Tom Eccles, Jake Bruce, Ali Razavi, Ashley Edwards, Nicolas Heess, Yutian Chen, Raia Hadsell, Oriol Vinyals, Mahyar Bordbar, and Nando de Freitas. A generalist agent, 2022. URL https://arxiv.org/abs/2205.06175.

[15] Anthony Brohan, Noah Brown, Justice Carbajal, Yevgen Chebotar, Xi Chen, Krzysztof Choromanski, Tianli Ding, Danny Driess, Avinava Dubey, Chelsea Finn, Pete Florence, Chuyuan Fu, Montse Gonzalez Arenas, Keerthana Gopalakrishnan, Kehang Han, Karol Hausman, Alexander Herzog, Jasmine Hsu, Brian Ichter, Alex Irpan, Nikhil Joshi, Ryan Julian, Dmitry Kalashnikov, Yuheng Kuang, Isabel Leal, Lisa Lee, Tsang-Wei Edward Lee, Sergey Levine, Yao Lu, Henryk Michalewski, Igor Mordatch, Karl Pertsch, Kanishka Rao, Krista Reymann, Michael Ryoo, Grecia Salazar, Pannag Sanketi, Pierre Sermanet, Jaspiar Singh, Anikait Singh, Radu Soricut, Huong Tran, Vincent Vanhoucke, Quan Vuong, Ayzaan Wahid, Stefan Welker, Paul Wohlhart, Jialin Wu, Fei Xia, Ted Xiao, Peng Xu, Sichun Xu, Tianhe Yu, and Brianna Zitkovich. Rt-2: Vision-language-action models transfer web knowledge to robotic control, 2023. URL https://arxiv.org/abs/2307.15818.

[16] Moo Jin Kim, Karl Pertsch, Siddharth Karamcheti, Ted Xiao, Ashwin Balakrishna, Suraj Nair, Rafael Rafailov, Ethan Foster, Grace Lam, Pannag Sanketi, Quan Vuong, Thomas Kollar, Benjamin Burchfiel, Russ Tedrake, Dorsa Sadigh, Sergey Levine, Percy Liang, and Chelsea Finn. Openvla: An open-source vision-language-action model, 2024. URL https://arxiv.org/abs/2406.09246.

[17] Embodiment Collaboration, Abby O'Neill, Abdul Rehman, Abhinav Gupta, Abhiram Maddukuri, Abhishek Gupta, Abhishek Padalkar, Abraham Lee, Acorn Pooley, Agrim Gupta, Ajay Mandlekar, Ajinkya Jain, Albert Tung, Alex Bewley, Alex Herzog, Alex Irpan, Alexander Khazatsky, Anant Rai, Anchit Gupta, Andrew Wang, Andrey Kolobov, Anikait Singh, Animesh Garg, Aniruddha Kembhavi, Annie Xie, Anthony Brohan, Antonin Raffin, Archit Sharma, Arefeh Yavary, Arhan Jain, Ashwin Balakrishna, Ayzaan Wahid, Ben Burgess-Limerick, Beomjoon Kim, Bernhard Schölkopf, Blake Wulfe, Brian Ichter, Cewu Lu, Charles Xu, Charlotte Le, Chelsea Finn, Chen Wang, Chenfeng Xu, Cheng Chi, Chenguang Huang, Christine Chan, Christopher Agia, Chuer Pan, Chuyuan Fu, Coline Devin, Danfei Xu, Daniel Morton, Danny Driess, Daphne Chen, Deepak Pathak, Dhruv Shah, Dieter Büchler, Dinesh Jayaraman, Dmitry Kalashnikov, Dorsa Sadigh, Edward Johns, Ethan Foster, Fangchen Liu, Federico Ceola, Fei Xia, Feiyu Zhao, Felipe Vieira Frujeri, Freek Stulp, Gaoyue Zhou, Gaurav S. Sukhatme, Gautam Salhotra, Ge Yan, Gilbert Feng, Giulio Schiavi, Glen Berseth, Gregory Kahn, Guangwen Yang, Guanzhi Wang, Hao Su, Hao-Shu Fang, Haochen Shi, Henghui Bao, Heni Ben Amor, Henrik I Christensen, Hiroki Furuta, Homanga Bharadhwaj, Homer

Walke, Hongjie Fang, Huy Ha, Igor Mordatch, Ilija Radosavovic, Isabel Leal, Jacky Liang, Jad Abou-Chakra, Jaehyung Kim, Jaimyn Drake, Jan Peters, Jan Schneider, Jasmine Hsu, Jay Vakil, Jeannette Bohg, Jeffrey Bingham, Jeffrey Wu, Jensen Gao, Jiaheng Hu, Jiajun Wu, Jialin Wu, Jiankai Sun, Jianlan Luo, Jiayuan Gu, Jie Tan, Jihoon Oh, Jimmy Wu, Jingpei Lu, Jingyun Yang, Jitendra Malik, João Silvério, Joey Hejna, Jonathan Booher, Jonathan Tompson, Jonathan Yang, Jordi Salvador, Joseph J. Lim, Junhyek Han, Kaiyuan Wang, Kanishka Rao, Karl Pertsch, Karol Hausman, Keegan Go, Keerthana Gopalakrishnan, Ken Goldberg, Kendra Byrne, Kenneth Oslund, Kento Kawaharazuka, Kevin Black, Kevin Lin, Kevin Zhang, Kiana Ehsani, Kiran Lekkala, Kirsty Ellis, Krishan Rana, Krishnan Srinivasan, Kuan Fang, Kunal Pratap Singh, Kuo-Hao Zeng, Kyle Hatch, Kyle Hsu, Laurent Itti, Lawrence Yunliang Chen, Lerrel Pinto, Li Fei-Fei, Liam Tan, Linxi "Jim" Fan, Lionel Ott, Lisa Lee, Luca Weihs, Magnum Chen, Marion Lepert, Marius Memmel, Masayoshi Tomizuka, Masha Itkina, Mateo Guaman Castro, Max Spero, Maximilian Du, Michael Ahn, Michael C. Yip, Mingtong Zhang, Mingyu Ding, Minho Heo, Mohan Kumar Srirama, Mohit Sharma, Moo Jin Kim, Muhammad Zubair Irshad, Naoaki Kanazawa, Nicklas Hansen, Nicolas Heess, Nikhil J Joshi, Niko Suenderhauf, Ning Liu, Norman Di Palo, Nur Muhammad Mahi Shafiullah, Oier Mees, Oliver Kroemer, Osbert Bastani, Pannag R Sanketi, Patrick "Tree" Miller, Patrick Yin, Paul Wohlhart, Peng Xu, Peter David Fagan, Peter Mitrano, Pierre Sermanet, Pieter Abbeel, Priya Sundaresan, Qiuyu Chen, Quan Vuong, Rafael Rafailov, Ran Tian, Ria Doshi, Roberto Martín-Martín, Rohan Baijal, Rosario Scalise, Rose Hendrix, Roy Lin, Runjia Qian, Ruohan Zhang, Russell Mendonca, Rutav Shah, Ryan Hoque, Ryan Julian, Samuel Bustamante, Sean Kirmani, Sergey Levine, Shan Lin, Sherry Moore, Shikhar Bahl, Shivin Dass, Shubham Sonawani, Shubham Tulsiani, Shuran Song, Sichun Xu, Siddhant Haldar, Siddharth Karamcheti, Simeon Adebola, Simon Guist, Soroush Nasiriany, Stefan Schaal, Stefan Welker, Stephen Tian, Subramanian Ramamoorthy, Sudeep Dasari, Suneel Belkhale, Sungjae Park, Suraj Nair, Suvir Mirchandani, Takayuki Osa, Tanmay Gupta, Tatsuya Harada, Tatsuya Matsushima, Ted Xiao, Thomas Kollar, Tianhe Yu, Tianli Ding, Todor Davchev, Tony Z. Zhao, Travis Armstrong, Trevor Darrell, Trinity Chung, Vidhi Jain, Vikash Kumar, Vincent Vanhoucke, Vitor Guizilini, Wei Zhan, Wenxuan Zhou, Wolfram Burgard, Xi Chen, Xiangyu Chen, Xiaolong Wang, Xinghao Zhu, Xinyang Geng, Xiyuan Liu, Xu Liangwei, Xuanlin Li, Yansong Pang, Yao Lu, Yecheng Jason Ma, Yejin Kim, Yevgen Chebotar, Yifan Zhou, Yifeng Zhu, Yilin Wu, Ying Xu, Yixuan Wang, Yonatan Bisk, Yongqiang Dou, Yoonyoung Cho, Youngwoon Lee, Yuchen Cui, Yue Cao, Yueh-Hua Wu, Yujin Tang, Yuke Zhu, Yunchu Zhang, Yunfan Jiang, Yunshuang Li, Yunzhu Li, Yusuke Iwasawa, Yutaka Matsuo, Zehan Ma, Zhuo Xu, Zichen Jeff Cui, Zichen Zhang, Zipeng Fu, and Zipeng Lin. Open x-embodiment: Robotic learning

datasets and rt-x models, 2025. URL https://arxiv.org/abs/2310.08864.

[18] NVIDIA, :, Johan Bjorck, Fernando Castañeda, Nikita Cherniadev, Xingye Da, Runyu Ding, Linxi "Jim" Fan, Yu Fang, Dieter Fox, Fengyuan Hu, Spencer Huang, Joel Jang, Zhenyu Jiang, Jan Kautz, Kaushil Kundalia, Lawrence Lao, Zhiqi Li, Zongyu Lin, Kevin Lin, Guilin Liu, Edith Llontop, Loic Magne, Ajay Mandlekar, Avnish Narayan, Soroush Nasiriany, Scott Reed, You Liang Tan, Guanzhi Wang, Zu Wang, Jing Wang, Qi Wang, Jiannan Xiang, Yuqi Xie, Yinzhen Xu, Zhenjia Xu, Seonghyeon Ye, Zhiding Yu, Ao Zhang, Hao Zhang, Yizhou Zhao, Ruijie Zheng, and Yuke Zhu. Gr00t n1: An open foundation model for generalist humanoid robots, 2025. URL https://arxiv.org/abs/2503.14734.

[19] Zaijing Li, Yuquan Xie, Rui Shao, Gongwei Chen, Dongmei Jiang, and Liqiang Nie. 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.

[20] SIMA Team, Maria Abi Raad, Arun Ahuja, Catarina Barros, Frederic Besse, Andrew Bolt, Adrian Bolton, Bethanie Brownfield, Gavin Buttimore, Max Cant, Sarah Chakera, Stephanie C. Y. Chan, Jeff Clune, Adrian Collister, Vikki Copeman, Alex Cullum, Ishita Dasgupta, Dario de Cesare, Julia Di Trapani, Yani Donchev, Emma Dunleavy, Martin Engelcke, Ryan Faulkner, Frankie Garcia, Charles Gbadamosi, Zhitao Gong, Lucy Gonzales, Kshitij Gupta, Karol Gregor, Arne Olav Hallingstad, Tim Harley, Sam Haves, Felix Hill, Ed Hirst, Drew A. Hudson, Jony Hudson, Steph Hughes-Fitt, Danilo J. Rezende, Mimi Jasarevic, Laura Kampis, Rosemary Ke, Thomas Keck, Junkyung Kim, Oscar Knagg, Kavya Kopparapu, Rory Lawton, Andrew Lampinen, Shane Legg, Alexander Lerchner, Marjorie Limont, Yulan Liu, Maria Loks-Thompson, Joseph Marino, Kathryn Martin Cussons, Loic Matthey, Siobhan Mcloughlin, Piermaria Mendolicchio, Hamza Merzic, Anna Mitenkova, Alexandre Moufarek, Valeria Oliveira, Yanko Oliveira, Hannah Openshaw, Renke Pan, Aneesh Pappu, Alex Platonov, Ollie Purkiss, David Reichert, John Reid, Pierre Harvey Richemond, Tyson Roberts, Giles Ruscoe, Jaume Sanchez Elias, Tasha Sandars, Daniel P. Sawyer, Tim Scholtes, Guy Simmons, Daniel Slater, Hubert Soyer, Heiko Strathmann, Peter Stys, Allison C. Tam, Denis Teplyashin, Tayfun Terzi, Davide Vercelli, Bojan Vujatovic, Marcus Wainwright, Jane X. Wang, Zhengdong Wang, Daan Wierstra, Duncan Williams, Nathaniel Wong, Sarah York, and Nick Young. Scaling instructable agents across many simulated worlds, 2024. URL https://arxiv.org/abs/2404.10179.

[21] SIMA team, Adrian Bolton, Alexander Lerchner, Alexandra Cordell, Alexandre Mo-

ufarek, Andrew Bolt, Andrew Lampinen, Anna Mitenkova, Arne Olav Hallingstad, Bojan Vujatovic, Bonnie Li, Cong Lu, Daan Wierstra, Daniel P. Sawyer, Daniel Slater, David Reichert, Davide Vercelli, Demis Hassabis, Drew A. Hudson, Duncan Williams, Ed Hirst, Fabio Pardo, Felix Hill, Frederic Besse, Hannah Openshaw, Harris Chan, Hubert Soyer, Jane X. Wang, Jeff Clune, John Agapiou, John Reid, Joseph Marino, Junkyung Kim, Karol Gregor, Kaustubh Sridhar, Kay McKinney, Laura Kampis, Lei M. Zhang, Loic Matthey, Luyu Wang, Maria Abi Raad, Maria Loks-Thompson, Martin Engelcke, Matija Kecman, Matthew Jackson, Maxime Gazeau, Ollie Purkiss, Oscar Knagg, Peter Stys, Piermaria Mendolicchio, Raia Hadsell, Rosemary Ke, Ryan Faulkner, Sarah Chakera, Satinder Singh Baveja, Shane Legg, Sheleem Kashem, Tayfun Terzi, Thomas Keck, Tim Harley, Tim Scholtes, Tyson Roberts, Volodymyr Mnih, Yulan Liu, Zhengdong Wang, and Zoubin Ghahramani. Sima 2: A generalist embodied agent for virtual worlds, 2025. URL https://arxiv.org/abs/2512.04797.

[22] Faraz Torabi, Garrett Warnell, and Peter Stone. Behavioral cloning from observation, 2018. URL https://arxiv.org/abs/1805.01954.

[23] Ashley Edwards, Himanshu Sahni, Yannick Schroecker, and Charles Isbell. Imitating latent policies from observation. In Kamalika Chaudhuri and Ruslan Salakhutdinov, editors, *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 1755–1763. PMLR, 09–15 Jun 2019. URL https://proceedings.mlr.press/v97/edwards19a.html.

[24] Dominik Schmidt and Minqi Jiang. Learning to act without actions. In *The Twelfth International Conference on Learning Representations (ICLR)*, 2024.

[25] Jake Bruce, Michael D Dennis, Ashley Edwards, Jack Parker-Holder, Yuge Shi, Edward Hughes, Matthew Lai, Aditi Mavalankar, Richie Steigerwald, Chris Apps, Yusuf Aytar, Sarah Maria Elisabeth Bechtle, Feryal Behbahani, Stephanie C.Y. Chan, Nicolas Heess, Lucy Gonzalez, Simon Osindero, Sherjil Ozair, Scott Reed, Jingwei Zhang, Konrad Zolna, Jeff Clune, Nando De Freitas, Satinder Singh, and Tim Rocktäschel. Genie: Generative interactive environments. In Ruslan Salakhutdinov, Zico Kolter, Katherine Heller, Adrian Weller, Nuria Oliver, Jonathan Scarlett, and Felix Berkenkamp, editors, *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pages 4603–4623. PMLR, 21–27 Jul 2024. URL https://proceedings.mlr.press/v235/bruce24a.html.

[26] Jake Bruce, Michael Dennis, Ashley Edwards, Jack Parker-Holder, Yuge Shi, Edward Hughes, Matthew Lai, Aditi Mavalankar, Richie Steigerwald, Chris Apps, Yusuf Aytar,

Sarah Bechtle, Feryal Behbahani, Stephanie Chan, Nicolas Heess, Lucy Gonzalez, Simon Osindero, Sherjil Ozair, Scott Reed, Jingwei Zhang, Konrad Zolna, Jeff Clune, Nando de Freitas, Satinder Singh, and Tim Rocktäschel. Genie: Generative interactive environments, 2024. URL https://arxiv.org/abs/2402.15391.

[27] Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, Sebastian Riedel, and Douwe Kiela. 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.

[28] Kelvin Guu, Kenton Lee, Zora Tung, Panupong Pasupat, and Mingwei Chang. REALM: Retrieval-augmented language model pre-training. In *Proceedings of the 37th International Conference on Machine Learning*, 2020.

[29] Sebastian Borgeaud, Arthur Mensch, Jordan Hoffmann, Trevor Cai, Eliza Rutherford, Katie Millican, George Bm Van Den Driessche, Jean-Baptiste Lespiau, Bogdan Damoc, Aidan Clark, Diego De Las Casas, Aurelia Guy, Jacob Menick, Roman Ring, Tom Hennigan, Saffron Huang, Loren Maggiore, Chris Jones, Albin Cassirer, Andy Brock, Michela Paganini, Geoffrey Irving, Oriol Vinyals, Simon Osindero, Karen Simonyan, Jack Rae, Erich Elsen, and Laurent Sifre. Improving language models by retrieving from trillions of tokens. In Kamalika Chaudhuri, Stefanie Jegelka, Le Song, Csaba Szepesvari, Gang Niu, and Sivan Sabato, editors, *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pages 2206–2240. PMLR, 17–23 Jul 2022. URL https://proceedings.mlr.press/v162/borgeaud22a.html.

[30] Gautier Izacard, Patrick Lewis, Maria Lomeli, Lucas Hosseini, Fabio Petroni, Timo Schick, Jane Dwivedi-Yu, Armand Joulin, Sebastian Riedel, and Edouard Grave. Atlas: Few-shot learning with retrieval augmented language models. *Journal of Machine Learning Research*, 24(251):1–43, 2023.

[31] Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. React: Synergizing reasoning and acting in language models. *arXiv preprint arXiv:2210.03629*, 2022.

[32] Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik R Narasimhan, and Shunyu Yao. Reflexion: language agents with verbal reinforcement learning. In *Thirty-*

seventh Conference on Neural Information Processing Systems, 2023. URL https://openreview.net/forum?id=vAElhFcKW6.

[33] Peter Conway Humphreys, Arthur Guez, Olivier Tieleman, Laurent Sifre, Theophane Weber, and Timothy P Lillicrap. Large-scale retrieval for reinforcement learning. In Alice H. Oh, Alekh Agarwal, Danielle Belgrave, and Kyunghyun Cho, editors, *Advances in Neural Information Processing Systems*, 2022. URL https://openreview.net/forum?id=Ya9lATuQ3gg.

[34] Anirudh Goyal, Abram Friesen, Andrea Banino, Theophane Weber, Nan Rosemary Ke, Adrià Puigdomènech Badia, Arthur Guez, Mehdi Mirza, Peter C Humphreys, Ksenia Konyushova, Michal Valko, Simon Osindero, Timothy Lillicrap, Nicolas Heess, and Charles Blundell. Retrieval-augmented reinforcement learning. In Kamalika Chaudhuri, Stefanie Jegelka, Le Song, Csaba Szepesvari, Gang Niu, and Sivan Sabato, editors, *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pages 7740–7765. PMLR, 17–23 Jul 2022. URL https://proceedings.mlr.press/v162/goyal22a.html.

[35] Andrew Zhao, Daniel Huang, Quentin Xu, Matthieu Lin, Yong-Jin Liu, and Gao Huang. ExpeL: LLM agents are experiential learners. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2024.

[36] Guanzhi Wang, Yuqi Xie, Yunfan Jiang, Ajay Mandlekar, Chaowei Xiao, Yuke Zhu, Linxi Fan, and Anima Anandkumar. Voyager: An open-ended embodied agent with large language models, 2023. URL https://arxiv.org/abs/2305.16291.

[37] Joon Sung Park, Joseph C. O'Brien, Carrie J. Cai, Meredith Ringel Morris, Percy Liang, and Michael S. Bernstein. Generative agents: Interactive simulacra of human behavior. In *Proceedings of the 36th Annual ACM Symposium on User Interface Software and Technology*, 2023.

[38] Danyang Zhang, Lu Chen, Situo Zhang, Hongshen Xu, Zihan Zhao, and Kai Yu. 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.

[39] Thomas Anthony, Zheng Tian, and David Barber. Thinking fast and slow with deep learning and tree search. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, NIPS'17, page 5366–5376, Red Hook, NY, USA, 2017. Curran Associates Inc. ISBN 9781510860964.

[40] David Silver, Thomas Hubert, Julian Schrittwieser, Ioannis Antonoglou, Matthew Lai, Arthur Guez, Marc Lanctot, Laurent Sifre, Dharshan Kumaran, Thore Graepel, Timothy Lillicrap, Karen Simonyan, and Demis Hassabis. Mastering chess and shogi by self-play with a general reinforcement learning algorithm, 2017. URL https://arxiv.org/abs/1712.01815.

[41] Eric Zelikman, Yuhuai Wu, Jesse Mu, and Noah Goodman. STar: Bootstrapping reasoning with reasoning. In Alice H. Oh, Alekh Agarwal, Danielle Belgrave, and Kyunghyun Cho, editors, *Advances in Neural Information Processing Systems*, 2022. URL https://openreview.net/forum?id=_3ELRdg2sgI.

[42] Caglar Gulcehre, Tom Le Paine, Srivatsan Srinivasan, Ksenia Konyushkova, Lotte Weerts, Abhishek Sharma, Aditya Siddhant, Alex Ahern, Miaosen Wang, Chenjie Gu, Wolfgang Macherey, Arnaud Doucet, Orhan Firat, and Nando de Freitas. Reinforced self-training (rest) for language modeling, 2023. URL https://arxiv.org/abs/2308.08998.

[43] Zixiang Chen, Yihe Deng, Huizhuo Yuan, Kaixuan Ji, and Quanquan Gu. Self-play fine-tuning converts weak language models to strong language models. In *Proceedings of the 41st International Conference on Machine Learning*, 2024.

[44] Michael Ahn, Debidatta Dwibedi, Chelsea Finn, Montse Gonzalez Arenas, Keerthana Gopalakrishnan, Karol Hausman, Brian Ichter, Alex Irpan, Nikhil Joshi, Ryan Julian, Sean Kirmani, Isabel Leal, Edward Lee, Sergey Levine, Yao Lu, Isabel Leal, Sharath Maddineni, Kanishka Rao, Dorsa Sadigh, Pannag Sanketi, Pierre Sermanet, Quan Vuong, Stefan Welker, Fei Xia, Ted Xiao, Peng Xu, Steve Xu, and Zhuo Xu. Autort: Embodied foundation models for large scale orchestration of robotic agents, 2024.

[45] Yecheng Jason Ma, William Liang, Guanzhi Wang, De-An Huang, Osbert Bastani, Dinesh Jayaraman, Yuke Zhu, Linxi Fan, and Anima Anandkumar. Eureka: Human-level reward design via coding large language models. In *International Conference on Learning Representations*, 2024.

[46] Linxi Fan, Guanzhi Wang, Yunfan Jiang, Ajay Mandlekar, Yuncong Yang, Haoyi Zhu, Andrew Tang, De-An Huang, Yuke Zhu, and Anima Anandkumar. 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.

[47] Deepak Pathak, Pulkit Agrawal, Alexei A. Efros, and Trevor Darrell. Curiosity-driven exploration by self-supervised prediction. In *Proceedings of the 34th International Conference on Machine Learning*, 2017.

[48] Lili Chen, Kevin Lu, Aravind Rajeswaran, Kimin Lee, Aditya Grover, Michael Laskin, Pieter Abbeel, Aravind Srinivas, and Igor Mordatch. Decision transformer: Reinforcement learning via sequence modeling, 2021. URL https://arxiv.org/abs/2106.01345.

[49] Greg Wayne, Chia-Chun Hung, David Amos, Mehdi Mirza, Arun Ahuja, Agnieszka Grabska-Barwinska, Jack Rae, Piotr Mirowski, Joel Z. Leibo, Adam Santoro, et al. Unsupervised predictive memory in a goal-directed agent, 2018.

[50] Yuhuai Wu, Markus Norman Rabe, DeLesley Hutchins, and Christian Szegedy. Memorizing transformers. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=TrjbxzRcnf-.

[51] Aydar Bulatov, Yuri Kuratov, and Mikhail Burtsev. Recurrent memory transformer. In Alice H. Oh, Alekh Agarwal, Danielle Belgrave, and Kyunghyun Cho, editors, *Advances in Neural Information Processing Systems*, 2022. URL https://openreview.net/forum?id=Uynr3iPhksa.

[52] Charles Packer, Sarah Wooders, Kevin Lin, Vivian Fang, Shishir G. Patil, Ion Stoica, and Joseph E. Gonzalez. MemGPT: Towards LLMs as operating systems, 2023.

[53] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. Playing atari with deep reinforcement learning, 2013. URL https://arxiv.org/abs/1312.5602.

[54] Oriol Vinyals, Igor Babuschkin, Wojciech M. Czarnecki, Michaël Mathieu, Andrew Joseph Dudzik, Junyoung Chung, David Choi, Richard Powell, Timo Ewalds, Petko Georgiev, Junhyuk Oh, Dan Horgan, Manuel Kroiss, Ivo Danihelka, Aja Huang, L. Sifre, Trevor Cai, John P. Agapiou, Max Jaderberg, Alexander Sasha Vezhnevets, Rémi Leblond, Tobias Pohlen, Valentin Dalibard, David Budden, Yury Sulsky, James Molloy, Tom Le Paine, Caglar Gulcehre, Ziyun Wang, Tobias Pfaff, Yuhuai Wu, Roman Ring, Dani Yogatama, Dario Wünsch, Katrina McKinney, Oliver Smith, Tom Schaul, Timothy P. Lillicrap, Koray Kavukcuoglu, Demis Hassabis, Chris Apps, and David Silver. Grandmaster level in starcraft ii using multi-agent reinforcement learning. *Nature*, 575:350 – 354, 2019. URL https://api.semanticscholar.org/CorpusID:204972004.

[55] OpenAI, :, Christopher Berner, Greg Brockman, Brooke Chan, Vicki Cheung, Przemysław Dębiak, Christy Dennison, David Farhi, Quirin Fischer, Shariq Hashme, Chris Hesse, Rafal Józefowicz, Scott Gray, Catherine Olsson, Jakub Pachocki, Michael Petrov, Henrique P. d. O. Pinto, Jonathan Raiman, Tim Salimans, Jeremy Schlatter, Jonas Schneider, Szymon Sidor, Ilya Sutskever, Jie Tang, Filip Wolski, and Susan Zhang. Dota 2 with large scale deep reinforcement learning, 2019. URL https://arxiv.org/abs/1912.06680.

[56] Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, Y. K. Li, Y. Wu, and Daya Guo. Deepseekmath: Pushing the limits of mathematical reasoning in open language models, 2024. URL https://arxiv.org/abs/2402.03300.

[57] Michael McCloskey and Neal J. Cohen. Catastrophic interference in connectionist networks: The sequential learning problem. In *Psychology of Learning and Motivation*, volume 24, pages 109–165. Academic Press, 1989.

[58] James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A. Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, et al. Overcoming catastrophic forgetting in neural networks. *Proceedings of the National Academy of Sciences*, 114(13):3521–3526, 2017.

[59] Zhizhong Li and Derek Hoiem. Learning without forgetting. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 40(12):2935–2947, 2017.

[60] Sylvestre-Alvise Rebuffi, Alexander Kolesnikov, Georg Sperl, and Christoph H. Lampert. iCaRL: Incremental classifier and representation learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[61] David Rolnick, Arun Ahuja, Jonathan Schwarz, Timothy Lillicrap, and Greg Wayne. Experience replay for continual learning. In *Advances in Neural Information Processing Systems*, 2019.

[62] Jonathan Schwarz, Wojciech Czarnecki, Jelena Luketina, Agnieszka Grabska-Barwinska, Yee Whye Teh, Razvan Pascanu, and Raia Hadsell. Progress & compress: A scalable framework for continual learning. In *Proceedings of the 35th International Conference on Machine Learning*, 2018.

[63] Khimya Khetarpal, Matthew Riemer, Irina Rish, and Doina Precup. Towards continual reinforcement learning: A review and perspectives. *Journal of Artificial Intelligence Research*, 75:1401–1476, 2022.

[64] Ricardo J. G. B. Campello, Davoud Moulavi, and Joerg Sander. Density-based clustering based on hierarchical density estimates. In Jian Pei, Vincent S. Tseng, Longbing Cao, Hiroshi Motoda, and Guandong Xu, editors, *Advances in Knowledge Discovery and Data Mining*, pages 160–172, Berlin, Heidelberg, 2013. Springer Berlin Heidelberg. ISBN 978-3-642-37456-2.

[65] Xiaohua Zhai, Basil Mustafa, Alexander Kolesnikov, and Lucas Beyer. Sigmoid loss for language image pre-training, 2023. URL https://arxiv.org/abs/2303.15343.

[66] An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, Chujie Zheng, Dayiheng Liu, Fan Zhou, Fei Huang, Feng Hu, Hao Ge, Haoran Wei, Huan Lin, Jialong Tang, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jing Zhou, Jingren Zhou, Junyang Lin, Kai Dang, Keqin Bao, Kexin Yang, Le Yu, Lianghao Deng, Mei Li, Mingfeng Xue, Mingze Li, Pei Zhang, Peng Wang, Qin Zhu, Rui Men, Ruize Gao, Shixuan Liu, Shuang Luo, Tianhao Li, Tianyi Tang, Wenbiao Yin, Xingzhang Ren, Xinyu Wang, Xinyu Zhang, Xuancheng Ren, Yang Fan, Yang Su, Yichang Zhang, Yinger Zhang, Yu Wan, Yuqiong Liu, Zekun Wang, Zeyu Cui, Zhenru Zhang, Zhipeng Zhou, and Zihan Qiu. Qwen3 technical report, 2025. URL https://arxiv.org/abs/2505.09388.

[67] Maxime Oquab, Timothée Darcet, Théo Moutakanni, Huy Vo, Marc Szafraniec, Vasil Khalidov, Pierre Fernandez, Daniel Haziza, Francisco Massa, Alaaeldin El-Nouby, Mahmoud Assran, Nicolas Ballas, Wojciech Galuba, Russell Howes, Po-Yao Huang, Shang-Wen Li, Ishan Misra, Michael Rabbat, Vasu Sharma, Gabriel Synnaeve, Hu Xu, Hervé Jegou, Julien Mairal, Patrick Labatut, Armand Joulin, and Piotr Bojanowski. Dinov2: Learning robust visual features without supervision, 2024. URL https://arxiv.org/abs/2304.07193.

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

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

[70] Sebastian Raschka, Joshua Patterson, and Corey Nolet. Machine learning in python: Main developments and technology trends in data science, machine learning, and artificial intelligence. *arXiv preprint arXiv:2002.04803*, 2020.

[71] Zhenyi Wang, Enneng Yang, Li Shen, and Heng Huang. 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.

[72] S. Lloyd. Least squares quantization in pcm. *IEEE transactions on information theory*, 28(2):129–137, 1982. ISSN 0018-9448.

[73] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms, 2017. URL https://arxiv.org/abs/1707.06347.

[74] Eric Hambro, Roberta Raileanu, Danielle Rothermel, Vegard Mella, Tim Rocktäschel, Heinrich Küttler, and Naila Murray. Dungeons and data: A large-scale nethack dataset, 2023. URL https://arxiv.org/abs/2211.00539.

[75] Seth Karten, Jake Grigsby, Stephanie Milani, Kiran Vodrahalli, Amy Zhang, Fei Fang, Yuke Zhu, and Chi Jin. The pokeagent challenge: Competitive and long-context learning at scale. In *NeurIPS Competition Track*, April 2025.

[76] Gemini Team. Gemini 2.5: Pushing the frontier with advanced reasoning, multimodality, long context, and next generation agentic capabilities, 2025. URL https://arxiv.org/abs/2507.06261.

[77] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization, 2019. URL https://arxiv.org/abs/1711.05101.

[78] Loïc Magne, Anas Awadalla, Guanzhi Wang, Yinzhen Xu, Joshua Belofsky, Fengyuan Hu, Joohwan Kim, Ludwig Schmidt, Georgia Gkioxari, Jan Kautz, Yisong Yue, Yejin Choi, Yuke Zhu, and Linxi "Jim" Fan. Nitrogen: An open foundation model for generalist gaming agents, 2026. URL https://arxiv.org/abs/2601.02427.

[79] Richard S. Sutton and Andrew G. Barto. *Reinforcement learning : an introduction*. Adaptive computation and machine learning. The MIT Press, Cambridge, Massachusetts, second edition. edition, 2018. ISBN 9780262039246.

[80] Balls Play Pokémon. Can plinko beat pokémon emerald? #1. https://www.youtube.com/watch?v=rrbc{W}u4{G}{V}yc&list={P}{L}fqgdq9_5frv{I}{F}k{K}w{O}q0{Y}w{S}s_ap2n{J}o{W}f, 2025. [Accessed 23-04-2026].

[81] Minecraft Wiki. Diamond. https://minecraft.wiki/w/{D}iamond, 2026. [Accessed 23-04-2026].

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

[83] Weihao Tan, Wentao Zhang, Xinrun Xu, Haochong Xia, Ziluo Ding, Boyu Li, Bohan Zhou, Junpeng Yue, Jiechuan Jiang, Yewen Li, Ruyi An, Molei Qin, Chuqiao Zong, Longtao Zheng, Yujie Wu, Xiaoqiang Chai, Yifei Bi, Tianbao Xie, Pengjie Gu, Xiyun Li, Ceyao Zhang, Long Tian, Chaojie Wang, Xinrun Wang, Börje F. Karlsson, Bo An, Shuicheng Yan, and Zongqing Lu. Cradle: Empowering foundation agents towards general computer control, 2024. URL https://arxiv.org/abs/2403.03186.

[84] Thomas Anthony, Zheng Tian, and David Barber. Thinking fast and slow with deep learning and tree search. In *Advances in Neural Information Processing Systems*, 2017.

[85] Qizhe Xie, Minh-Thang Luong, Eduard Hovy, and Quoc V. Le. Self-training with noisy student improves ImageNet classification. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020.

[86] Alexander Sasha Vezhnevets, Simon Osindero, Tom Schaul, Nicolas Heess, Max Jaderberg, David Silver, and Koray Kavukcuoglu. FeUdal networks for hierarchical reinforcement learning. In *Proceedings of the 34th International Conference on Machine Learning*, 2017.

[87] Marc G. Bellemare, Sriram Srinivasan, Georg Ostrovski, Tom Schaul, David Saxton, and Rémi Munos. Unifying count-based exploration and intrinsic motivation. In *Advances in Neural Information Processing Systems*, 2016.

[88] Yuri Burda, Harrison Edwards, Amos Storkey, and Oleg Klimov. Exploration by random network distillation. In *International Conference on Learning Representations*, 2019.

[89] Adrien Ecoffet, Joost Huizinga, Joel Lehman, Kenneth O. Stanley, and Jeff Clune. First return, then explore. *Nature*, 590(7847):580–586, 2021.

[90] Jakob Bauer, Kate Baumli, Feryal Behbahani, Avishkar Bhoopchand, Nathalie Bradley-Schmieg, Michael Chang, Natalie Clay, Adrian Collister, Vibhavari Dasagi, Lucy Gonzalez, et al. Human-timescale adaptation in an open-ended task space. In *Proceedings of the 40th International Conference on Machine Learning*, 2023.

[91] Marc G. Bellemare, Yavar Naddaf, Joel Veness, and Michael Bowling. The arcade learning environment: An evaluation platform for general agents. *Journal of Artificial Intelligence Research*, 47:253–279, 2013.

[92] Heinrich Küttler, Nantas Nardelli, Alexander H. Miller, Roberta Raileanu, Marco Selvatici, Edward Grefenstette, and Tim Rocktäschel. The NetHack learning environment. In *Advances in Neural Information Processing Systems*, 2020.

APPENDICES

Appendix A

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**

Appendix B

Hyperparameters and Training Settings

Hyperparameter	**Setting**
Optimizer	AdamW [77]
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 B.1: Hyperparameters and Settings.

Appendix C

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

Table C.1: Notation glossary (part 1 of 2).

Symbol	Description
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 C.2: Notation glossary (part 2 of 2).

Appendix D

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?
```

Appendix E

Search Keywords

Table E.1 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 E.1: Search terms used for creating internet dataset.

Appendix F

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.