SKILLHACK: A BENCHMARK FOR SKILL TRANSFER IN OPEN-ENDED REINFORCEMENT LEARNING

Anonymous authors
Paper under double-blind review

ABSTRACT

Practising and honing skills forms a fundamental component of how humans learn, yet artificial agents are rarely specifically trained to perform them. Instead, they are usually trained end-to-end, with the hope being that useful skills will be implicitly learned in order to maximise discounted return of some extrinsic reward function. In this paper, we investigate how skills can be incorporated into the training of reinforcement learning (RL) agents in complex environments with large state-action spaces and sparse rewards. To this end, we created SkillHack, a benchmark of tasks and associated skills based on the game of NetHack. We evaluate a number of baselines on this benchmark, as well as our own novel skill-based method Hierarchical Kickstarting (HKS), which is shown to outperform all other evaluated methods. Our experiments show that learning with a prior knowledge of useful skills can significantly improve the performance of agents on complex problems. We ultimately argue that utilising predefined skills provides a useful inductive bias for RL problems, especially those with large state-action spaces and sparse rewards.

1 INTRODUCTION

The acquisition and execution of skills form a fundamental component of how humans learn. Consider learning to play the game of football. Rather than learning by simply playing successive matches, large parts of training would be commonly devoted to developing specific skills, such as passing, shooting, footwork, and general fitness. Since humans seem to benefit from explicitly breaking down a complex task into constituent skills, we hypothesise that reinforcement learning (RL) agents can benefit from doing the same.

Most existing methods that incorporate some form of skill-based learning do so by *learning the skills during training*, simultaneously with the policy that makes use of these skills (Bacon et al., 2016; Frans et al., 2017; Vezhnevets et al., 2017). These two layers of learning often result in instability, although recent approaches have shown success in limiting this (Nachum et al., 2018; Levy et al., 2019).

In this work, we consider only the problem of learning with the aid of predefined skills. We assume access to a set of expert policies, one for each skill we have defined. These experts could be obtained automatically by performing some search for diverse policies (Lehman & Stanley, 2008; Eysenbach et al., 2018), be hand coded with some heuristic policy, or be trained with RL on a set of skill-specific environments to a level that is deemed sufficient. While the methods used in this paper could also work with a hand coded heuristic policy, we chose the option of producing a set of skill acquisition environments. We believe this allows the benchmark to be useful for a much wider variety of skill transfer methods, namely those that require training on the skill domains and not just access to expert policies, for instance (Rusu et al., 2016; Parisotto et al., 2016).

The manifestation of this is SkillHack[1], a new benchmark for skill-based learning using the MiniHack (Samvelyan et al., 2021) framework. The benchmark consists of 8 unique task environments, each of which has an associated set of skill acquisition environments for mastering the individual skills necessary for completing each task. The tasks are designed to be of a difficulty such that not utilising the relevant skill environments makes them very hard to solve. Completing the suite of

[1]Code available at https://github.com/skillhack-env/skillhack

tasks requires a broad range of skills in the NetHack environment (Kuttler et al., 2020) including navigation, combat and manipulation of in-game items. The large space of items, as well as actions for manipulating them, results in a state-action space that is difficult to explore under a random exploration regime. The large number of unique and diverse skills in the game of NetHack, combined with its speed and ease of use, make SkillHack a convenient benchmark for studying policy transfer in RL, as well as problems such as continual learning and unsupervised RL.

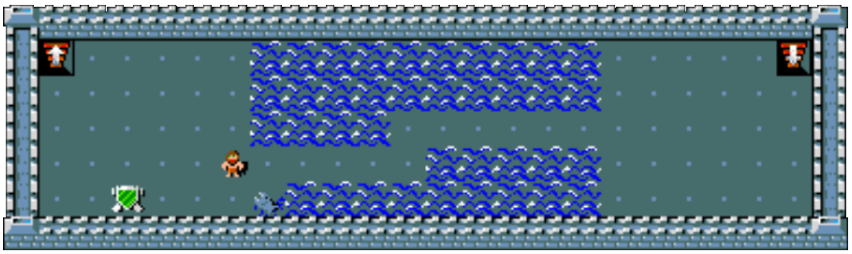
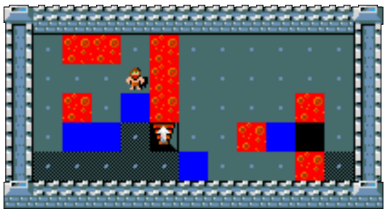
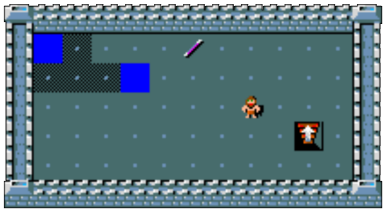


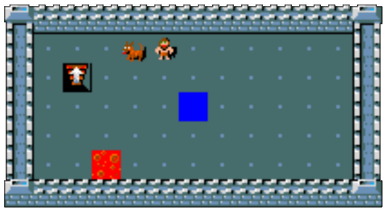
Figure 1: SEA MONSTERS task. The agent must reach the staircase on the opposite bank without dying to the monsters in the lake. To do this it must use of the powerful suit of armour that has spawned on the near bank. The agent can most easily learn to complete this task by transferring knowledge gained from its 4 associated skill environments.



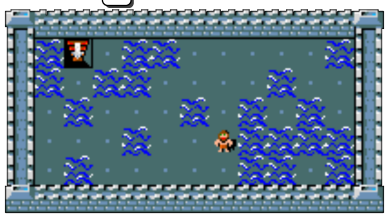
(a) `TakeOff` skill environment. The agent must take off the clothes it starts with by pressing Shift + t.



(b) `PickUp` skill environment. The agent must navigate to the item and pick it up with ,.



(c) `Wear` skill environment. The agent must press Shift + w and then select the item of clothing that is in its inventory.



(d) `NavigateWater` skill environment. The agent must reach the staircase, receiving a reward penalty every time it tries to walk into the water. Note that the agent can move diagonally.

Figure 2: Skill environments for the skills associated with the SEA MONSTERS task.

To evaluate SkillHack, we propose Hierarchical Kickstarting (HKS), a new policy transfer approach which distills knowledge acquired from expert policies trained on the skill acquisition environments. We refer to these expert policies as *teachers*. HKS learns a policy-over-teachers π_H, which outputs a weighting for each teacher at every timestep. These are then used to calculate a weighted average of the action distribution of the teachers. The agent policy π then incurs a loss proportional to the divergence of its own action distribution from this weighted average. This incentivises the agent to behave more similarly to highly weighted teachers, potentially accelerating learning by nudging the agent to behave more similarly to a useful policy. Simultaneously, the policy-over-teachers is incentivised to pick teachers that match the agent. After a burn-in time, these two networks should be in sync, with the policy-over-teachers recommending useful teachers based on the current state of the agent, while the agent behaves similarly to the chosen teachers. This method can be considered

as a more generalised version of Kickstarting (Schmitt et al., 2018), where we add the network π_H so that different skills can be prioritised for knowledge transfer at different points within an episode.

We perform an empirical evaluation of HKS as well as several baselines across all SkillHack environments. Our results show that learning with a prior knowledge of useful skills can significantly improve performance of agents on complex problems. We demonstrate that while HKS outperforms other baselines on the overall suite of SkillHack tasks, there is still significant room for methods to improve.

In summary, this paper makes the following contributions: (i) we present SkillHack, a new benchmark for skill transfer in RL, (ii) we propose HKS, a new method for policy transfer based on skill specific expert policies, (iii) we provide an evaluation and discussion of HKS and 3 other baseline methods on the SkillHack benchmark.

2 Background

2.1 Reinforcement Learning

We use the standard formalism of Markov Decision Process (MDP) defined as a tuple $M = \langle \mathcal{S}, \mathcal{A}, \mathcal{T}, \mathcal{R}, \gamma \rangle$ representing state space $\mathcal{S}$, action space $\mathcal{A}$, transition probability $\mathcal{T} : \mathcal{S} \times \mathcal{A} \rightarrow \mathcal{P}(\mathcal{S})$, reward function $\mathcal{R} : \mathcal{S} \times \mathcal{A} \rightarrow \mathbb{R}$, and discount factor γ. The agent chooses actions according to a stochastic policy $\pi : \mathcal{S} \rightarrow \mathcal{P}(\mathcal{S})$ in order to maximise expected episodic reward, defined as $\mathbb{E}\left[\sum_{t=0}^{T} \gamma^t r_t\right]$ where T is the final timestep and r_t is reward at timestep t. The value function of policy π is defined as $v_\pi(s) = \mathbb{E}_\pi\left[\sum_{t=0}^{T} \gamma^t r_t | s_t = s\right]$. Model-free approaches to RL aim to learn the optimal policy without explicitly learning the dynamics of the environment. Actor-critic algorithms (Konda & Tsitsiklis, 2000) make use of both a policy (actor) for control and an estimated value function (critic) for updating the policy of the agent.

2.2 Policy Transfer

Training RL agents from scratch can often be prohibitively expensive and time-consuming, while making use of existing knowledge can improve sample efficiency and training speed. *Policy transfer* approaches aim to assist the learning process of the agent by utilizing pretrained policies on related tasks. We refer to the policy designed for the target task M_T as the *student*, which can leverage knowledge from *teacher* policies $\pi_{E_1}, \pi_{E_2}, \ldots, \pi_{E_K}$ trained on a set of source domains $\mathcal{M}_1, \mathcal{M}_2, \ldots, \mathcal{M}_K$, respectively.

The Options Framework One way to perform policy transfer is to use the framework of options (Sutton et al., 1999) representing temporally-extended actions that can span multiple timesteps. An option ω is a 3-tuple $\langle \mathcal{I}, \pi, \beta \rangle$ representing the states the option can be initiated from, the option policy and the option termination function. In a given state s, a policy-over-options selects an option π_ω for which $s \in \mathcal{I}_\omega$. Control over actions is then ceded to this option for a number of timesteps. Once the termination condition β_ω is stochastically met, the option finishes and cedes control back to the policy-over-options, which then selects a new option to execute.

Kickstarting RL Kickstarting (Schmitt et al., 2018) is a policy transfer method that uses one or more expert teachers to guide a student policy. Multi-teacher Kickstarting works by adding an auxiliary loss to the student during training, defined as $l_{\text{kick}} = \mathcal{H}(\sum_k \lambda_k \pi_k(\cdot|s_t) \parallel \pi(\cdot|s_t))$, where $\mathcal{H}(\cdot \parallel \cdot)$ is cross-entropy, π_k are the teacher policies, π is the student policy and λ_k are the per-teacher Kickstarting coefficients. This means that the agent not only optimises for extrinsic reward, but is also incentivised to behave similarly to the teachers. The λ_k can be varied over training (although remain constant within an episode) either with Population-Based Training (Jaderberg et al., 2017) or with a manual schedule.

NetHack & MiniHack The NetHack Learning Environment (Kuttler et al., 2020) is an RL environment based off the classic game of *NetHack*. NetHack is a dungeon crawler game, notorious for its difficulty, where the player must work their way through dozens of procedurally generated levels,

with hundreds of unique enemies and objects. The game is turn based and stochastic, with a large action space. MiniHack (Samvelyan et al., 2021) is an extension of the NetHack Learning environment that allows for customisation of levels, rewards and termination conditions, while retaining access to the full set of entities and environment dynamics from NetHack.

Table 1: Summary of SkillHack tasks. Different styles are used to differentiate TASKS, `Skills` and `Items/Entities`.

Task	Description
BATTLE	`PickUp` a randomly placed `Sword`, `Wield` the `Sword` and finally `Fight` and kill a `Monster`.
PREPARE FOR BATTLE	`PickUp` a ration of `Food` and a piece of `Armour`. `Wear` the `Armour` and `Eat` the `Food`.
TARGET PRACTICE	Either a set of `Daggers` or a `Wand of Death` will spawn on the floor. `PickUp` the item and either `ZapWandOfDeath` or `Throw` the `Daggers` in order to kill the `Minotaur`.
MEDUSA	`PickUp` and then `PutOn` the `Towel` that is placed in your starting room, thus blindfolding yourself. Open the door to the second room where `Medusa` is caged, but can still instantly kill you with her deadly gaze should you enter with your vision intact. You must `NavigateBlind` in this second room in order to find the stairway out.
SEA MONSTERS	`TakeOff` your starting `Armour` and then `PickUp` and `Wear` the strong suit of `Armour` that spawns on the near bank of the lake. Whilst wearing this `Armour` you can `NavigateWater` across the bridge without dying to the `Piranhas`. Make it to the other side and reach the staircase to complete the task.
FROZEN LAVA CROSS	Either a `Wand of Cold` or a `Frost Horn` will spawn on the near side of a river of lava. `PickUp` the item and then create a bridge across the lava with either `ZapWandOfCold` or by `ApplyFrostHorn`. Finally, `NavigateLava` across your newly made bridge to reach the staircase on the other side.
IDENTIFY MIMIC	`Mimics` are monsters that can camouflage themselves. This task generates 3 `Statues`, 2 of which are real and one of which is actually a `Mimic` that you must safely identify. To do this you can `PickUp` a stack of `Daggers` and `Throw` them at the `Statues`. Hitting the `Mimic` will reveal it and complete the task. Alternatively you can `NavigateLavaToAmulet` by following the bridge across the lava lake and then `PickUp` and `PutOn` the `Amulet of ESP` which will reveal the `Mimic`. Revealing the `Mimic` in an unsafe way, for instance by walking into it, will cause you to fail the task.
A LOCKED DOOR	`PickUp` the `Skeleton Key` and use it to unlock the `Door`, before proceeding to `NavigateLava` to the exit.

3 SKILLHACK: A BENCHMARK FOR SKILL TRANSFER

To serve as a benchmark for skill transfer we developed *SkillHack*: a set of MiniHack environments containing 8 procedurally generated task environments and 16 skill acquisition environments. The task environments are designed to be tricky to solve with vanilla RL, due to the large state-action space and sparse rewards. For this reason, each task environment has an associated set of relevant skill environments. By transferring knowledge gained from solving these skill environments, the task environment will become easier to solve.

Each task environment is set up such that the agent receives a $+1$ reward for successfully completing the task and a -1 reward for failing the task and/or dying. The skill environments are set up with bespoke rewards to encourage an RL agent to learn the skill quickly and in a generalised manner.

A motivating example is the SEA MONSTERS task (Figure 1). In this task, the agent is faced with a narrow, procedurally generated bridge over a monster infested lake, with the goal to reach the staircase on the opposite bank. If the agent simply walks over the bridge, it will be killed by one of the monsters before it reaches the other side. However, on the near side of the lake a powerful piece of armour will be randomly generated and placed on the floor. The agent must `Take Off` the clothes it starts off wearing, `Pick Up` and then `Wear` the armour, before finally `Navigating Water` to reach the staircase on the other side of the lake.

The four skills associated with the task each have their own respective environments (Figure 2), in which skill-specific teacher agents can learn expert policies. The environments are all procedurally generated, with distractions in the form of random terrain and entities, in order to prevent overfitting (Cobbe et al., 2020). As well as providing reward and termination when the skill is successfully performed, a constant negative reward is applied every timestep in order to encourage the agent to perform the skill as fast as possible.

It should be noted that these task and skill acquisition environments only form a fraction of the possible behaviours in NetHack. However, the primary aim of SkillHack is not necessarily to directly lead to a more competent NetHack agent, but to facilitate research into skill transfer.

The full list of tasks is summarised in Table 1. More information on the skill acquisition and task environments can be found in appendices C and D respectively.

4 HIERARCHICAL KICKSTARTING

In this section, we propose Hierarchical Kickstarting (HKS), a new approach for policy transfer that combines the strengths of two other policy transfer approaches: Kickstarting and the Options Framework.

To perform policy transfer from multiple teachers, HKS utilises a hierarchical policy network π_H which at every timestep weights the relevance of each teacher π_k based on the current state. Rather than ceding control to one of the teacher policies π_k as in the Options Framework, this weighting is used to calculate a weighted average of the teacher action distributions. We then formulate an additional loss, proportional to the cross-entropy of this weighted average with the agent action distribution. The policy-over-teachers π_H therefore does not directly affect the action selection but only the loss incurred by the agent. As a result, the student π is optimised to behave similarly to the average of the teachers π_k as weighted by π_H, as well as maximising external rewards. Conversely, π_H is optimised to choose teachers that match the behaviour of π, forming a cyclic relationship between the two levels of policies.

The additional auxiliary loss of the student policy in HKS is computed as follows:

$$l_{\text{HKS}} = \lambda \mathcal{H}\left(\sum_k \pi_H(k|s_t)\pi_k(\cdot|s_t) \;\middle\|\; \pi(\cdot|s_t) \right) \tag{1}$$

HKS can be seen as a combination of Kickstarting and the Options Framework. It generalises Kickstarting by using a hierarchical policy-over-teachers which adjusts how much to kickstart from each teacher at each timestep, rather than distilling a fixed amount from each teacher. Since skills often have to be performed sequentially to complete a task, it is common for only a single skill to be immediately relevant at a given point in time, making this a useful ability.

Furthermore, HKS also addresses the weakness of the Options Framework. Pretrained options are effective when the transition dynamics and initial state distribution of the MDP of the target domain matches those used during the training of the option. However, skill-level options can be highly unstable when executed directly on a target task which is meaningfully different from the skill environment encountered during the training.

5 EXPERIMENTAL SETUP

All experiments are run with the IMPALA (Espeholt et al., 2018) framework, using the open-source TorchBeast implementation (Küttler et al., 2019). The agent architecture is an adapted version of the one used in (Kuttler et al., 2020) and (Samvelyan et al., 2021).

The inputs to the network include the full matrix of *glyphs* that serve as the main screen of the game, along with an encoding of the most recently received in-game message ("*You hit the orc!*"), the players inventory and a set of relevant statistics (health, armour level, etc.). The network uses CNNs for the spatial inputs, as well as LSTM units to incorporate memory. The action space consists of 32 actions (16 movement actions and 16 command actions). For the full action space see Appendix A and for a full listing of hyperparameters see Appendix F. For more detail on the observation space and network architecture see (Kuttler et al., 2020) and (Samvelyan et al., 2021).

An expert teacher policy is produced via vanilla RL for each of the 16 skill environments. The results of training the experts can be seen in Appendix E. We train vanilla RL, the Options Framework, Kickstarting and HKS on all 8 tasks for 2×10^8 timesteps, repeated over at least 3 random seeds. For the 3 skill-based methods, only teachers relevant to the target task are used. For instance, only the `TakeOff`, `PickUp`, `Wear` and `NavigateWater` teachers are used when training on the SEA MONSTERS task.

5.1 BASELINES

Options Framework We use the pretrained experts as options with frozen weights, while training a policy-over-options network to direct them. The option networks have no LSTM units and take only the instantaneous observation as input. We consider a degenerate form of the Options Framework where options can be initialised from any state and always terminate in one step $\mathcal{I}_\omega = \mathcal{S}$, $\beta(\cdot)_\omega = 1$.

Kickstarting For the Kickstarting agent, we use the experts as teachers and similarly remove the LSTMs. We keep all λ_k equal and therefore refer to them all as $\lambda = \lambda_k$. The schedule for λ was manually set to start at $\lambda = 10$ and linearly decay to $\lambda = 1$ after 10^7 timesteps. Retaining a small Kickstarting coefficient was found to increase the stability of the method, whereas if λ was allowed to decay to 0 there would be occasional massive drops in performance (Appendix B).

Hierarchical Kickstarting The LSTM removal and λ schedule are the same as for Kickstarting. The policy-over-teachers was implemented as an extra head on the student network in the form of an additional single fully connected layer with softmax activations. Preliminary testing of the method found that it would often quickly get stuck in local minima where the policy-over-teachers would always weight the same teacher very highly, resulting in poor performance. To mitigate this, an additional loss was added proportional to the negative entropy of the policy-over-teachers $l_E = -\kappa\mathcal{H}(\pi_H)$. This encourages the policy-over-teachers to give a more uniform distribution, preventing it from falling into the described local minima. The coefficient κ was set with a manual schedule, starting at $\kappa = 20$ and linearly decaying to $\kappa = 0$ after 2×10^7 timesteps.

6 RESULTS AND DISCUSSION

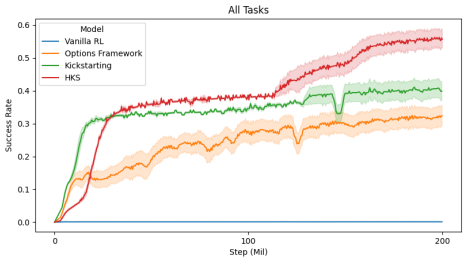


Figure 3: Mean success rate of baselines across all SkillHack tasks. The shaded area denotes 1 standard error.

Figure 3 plots the success rate of the 4 methods averaged over all 8 of the SkillHack tasks. The complete lack of progress by vanilla RL, in contrast to the progress by the 3 skill-based methods, shows that the SkillHack tasks are suitably designed to be a useful benchmark for policy transfer. HKS performs best in terms of final performance, achieving 56% success rate, as opposed to 41% and 32% for Kickstarting and the Options Framework, respectively. It does however take marginally longer than the other methods to converge. HKS overtakes the Options Framework and Kickstarting in success rate at

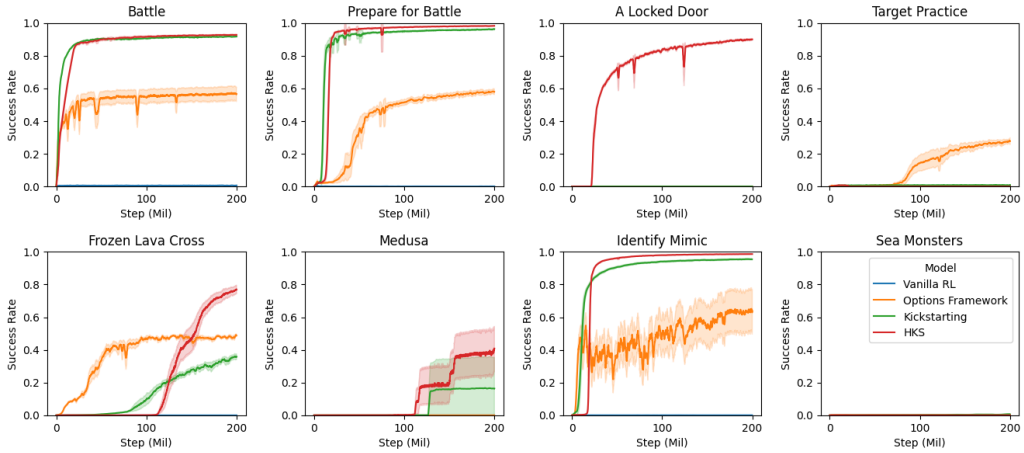


Figure 4: Success rate for all methods on the 8 SkillHack tasks, averaged over the repeats of each experiment. The shaded area denotes 1 standard error.

around 2×10^7 and 3×10^7 timesteps respectively. For a fair comparison against HKS, the Kickstarting score was increased by around 2%, for reasons discussed in Appendix B.

Figure 4 splits out the results for each of the SkillHack tasks, painting a more nuanced picture of the results. There are some environments where only a subset of the skill-based methods managed to achieve any significant performance, namely A LOCKED DOOR (HKS), TARGET PRACTICE (Options Framework) and MEDUSA (Kickstarting & HKS). In terms of the final success rate, HKS always at least equals Kickstarting, although notably loses to the Options Framework on TARGET PRACTICE. Interestingly, the only two tasks where the Options Framework outperforms Kickstarting (TARGET PRACTICE & FROZEN LAVA CROSS) are the only two tasks that cannot always be solved using the same method for each episode. Both tasks need to be solved differently depending on a random object spawned at the start of the episode (See Table 1). The SEA MONSTERS task was the only one to remain unsolved by all 4 methods, making it a good candidate for future research into skill-based methods.

7 RELATED WORK

Policy Transfer Policy transfer is defined as the paradigm where the learning process of an agent is supplemented by utilizing pretrained policies on related tasks. It is a paradigm that becomes more prevalent as environments develop larger state-action spaces, sparser rewards and more open-ended sets of possible behaviours. *Knowledge distillation* is a popular technique in supervised learning that ensembles knowledge from many teacher models into a student model (Hinton et al., 2015). Recent work in RL has used the distillation of policies to transfer knowledge from teacher policies to the student. (Rusu et al., 2016) propose *policy distillation* where the student policy is trained by minimising the divergence between the action distributions of the teacher and student policies over the trajectories sampled via the teacher policies. (Czarnecki et al., 2019) performs policy distillation using the trajectories that are sampled with the student policy, rather than teachers'. Another objective for policy transfer is minimising the cross-entropy between the teacher and student policies, as is done in Kickstarting RL (Schmitt et al., 2018). The Actor-Mimic algorithm (Parisotto et al., 2016) takes a slightly different approach to Kickstarting. First, the student is pretrained to master all the skill environments, but with the coefficient governing this loss function held constant. The resultant weights are used to initialise a network that then trains on the target environment, without any supervision from the teachers. A different perspective for achieving transfer is maximising the probability that the student and teacher policies will visit the same trajectories. An example of such an approach is *Distral* (Teh et al., 2017) where a central "distilled" policy is shared across many tasks in order to capture common behaviour. Each teacher is trained in separation to solve its own task while being constrained to stay close to the shared policy.

Hierarchical Reinforcement Learning While we choose to use the Options Framework as a baseline and as a building block of HKS, many other HRL approaches have been developed throughout the years, such as Feudal Networks (Dayan & Hinton, 1992; Vezhnevets et al., 2017), Hierarchical Deep Q-Networks (Kulkarni et al., 2016) and Data-Efficient HRL (Nachum et al., 2018).

Benchmarks We consider a range of RL benchmarks and their viability to the problem of skill transfer. Classic environments such as those in OpenAI gym (Brockman et al., 2016) are too simple to be broken down into constituent skills. Common video game environments such as Atari (Bellemare et al., 2013), DeepMind Lab (Beattie et al., 2016) and ViZDoom (Kempka et al., 2016) are complex enough for skill-based learning to be warranted, but much of the focus of learning in these environments must be put towards interpreting the high dimensional pixel input. The CORA benchmarka (Powers et al., 2021) for continual learning proposes schedules of environments which are each shown to the agent for some large number of iterations, with one of the primary goals being to *forward transfer* knowledge from previously seen environments to new ones.

The MineRL (Guss et al., 2021) benchmark challenges the RL agent to mine a diamond in the popular video game Minecraft. Using vanilla RL to do this would be intractable, so the benchmark comes with a dataset of over 60 million state-action tuples of recorded human demonstrations. It should be noted that while similar to the use of expert teachers in SkillHack, new samples cannot be dynamically generated from the teachers, meaning this dataset could not be used by any of the methods we investigate. The OBTAINDIAMOND task set out by this benchmark can be broken down into a large graph of dependent skills (`GatherWood`, `CreateWoodPickaxe`, `MineStone`, ...) and the benchmark provides 6 additional environments for completing chosen subsets of these skills. However, even these smaller tasks are highly complex and training will inevitably still lean heavily on the provided expert human trajectories. This makes MineRL moreso a challenge in imitation learning and interpreting the high dimensional pixel input, although with a more fine-grained breakdown of skill environments it could become an interesting benchmark for combining skill transfer and imitation learning.

Crafter (Hafner, 2022) is a 2D open world survival game, with similar game dynamics to Minecraft but with a lower dimensional input space and faster runtime. The paper formulates a taxonomy of 22 achievements and their dependencies on each other. By interpreting achievements with dependencies as tasks and achievements with no dependencies as skills, Crafter could be phrased as a skill transfer benchmark. However, the vast majority of the tasks would only have 1 associated skill and many skills would have no associated task.

8 CONCLUSION

This paper presents SkillHack: a NetHack-based benchmark for skill transfer in RL. SkillHack features 16 skill environments for acquiring knowledge necessary to solve 8 target problems, all of which have been shown to be intractable for vanilla RL. Our experimental results show that the SkillHack tasks form a difficult and varied suite of challenges, with methods performing markedly differently across the range of tasks. We also propose Hierarchical Kickstarting (HKS): a new method for skill transfer that combines Kickstarting and the Options Framework. Evaluating HKS on SkillHack shows that it improves the final performance on the overall benchmark by 15% over the next best baseline (Kickstarting).

In the near future, we aim to conduct additional experiments to compare the methods by conditioning on experts for all 16 skill environments, rather than the subset of relevant experts for each task. We would also like to investigate extending our baseline for the Options Framework to allow for options to learn during training (Bacon et al., 2016). Finally, we plan to use SkillHack to study other policy transfer approaches including inter-task mapping, representation transfer and transfer learning approaches for the framework of unsupervised RL.

We hope that SkillHack can serve as a useful benchmark in the community for evaluating skill transfer and that HKS can find applications where traditional RL methods struggle. We are open sourcing both the SkillHack benchmark and the HKS method and look forward to contributions from the community.

REFERENCES

Pierre-Luc Bacon, Jean Harb, and Doina Precup. The option-critic architecture. *CoRR*, abs/1609.05140, 2016. URL http://arxiv.org/abs/1609.05140.

Charles Beattie, Joel Z. Leibo, Denis Teplyashin, Tom Ward, Marcus Wainwright, Heinrich Küttler, Andrew Lefrancq, Simon Green, Víctor Valdés, Amir Sadik, Julian Schrittwieser, Keith Anderson, Sarah York, Max Cant, Adam Cain, Adrian Bolton, Stephen Gaffney, Helen King, Demis Hassabis, Shane Legg, and Stig Petersen. Deepmind lab, 2016.

M. G. Bellemare, Y. Naddaf, J. Veness, and M. Bowling. The arcade learning environment: An evaluation platform for general agents. *Journal of Artificial Intelligence Research*, 47:253–279, Jun 2013. ISSN 1076-9757. doi: 10.1613/jair.3912. URL http://dx.doi.org/10.1613/jair.3912.

Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. Openai gym. *arXiv preprint arXiv:1606.01540*, 2016.

Karl Cobbe, Christopher Hesse, Jacob Hilton, and John Schulman. Leveraging procedural generation to benchmark reinforcement learning, 2020.

Wojciech M. Czarnecki, Razvan Pascanu, Simon Osindero, Siddhant Jayakumar, Grzegorz Swirszcz, and Max Jaderberg. Distilling policy distillation. In Kamalika Chaudhuri and Masashi Sugiyama (eds.), *Proceedings of the Twenty-Second International Conference on Artificial Intelligence and Statistics*, volume 89 of *Proceedings of Machine Learning Research*, pp. 1331–1340. PMLR, 16–18 Apr 2019.

Peter Dayan and Geoffrey E. Hinton. Feudal reinforcement learning. In *Advances in Neural Information Processing Systems 5, [NIPS Conference]*, pp. 271–278, San Francisco, CA, USA, 1992. Morgan Kaufmann Publishers Inc. ISBN 1558602747.

Lasse Espeholt, Hubert Soyer, Remi Munos, Karen Simonyan, Volodymir Mnih, Tom Ward, Yotam Doron, Vlad Firoiu, Tim Harley, Iain Dunning, Shane Legg, and Koray Kavukcuoglu. Impala: Scalable distributed deep-rl with importance weighted actor-learner architectures, 2018.

Benjamin Eysenbach, Abhishek Gupta, Julian Ibarz, and Sergey Levine. Diversity is all you need: Learning skills without a reward function, 2018.

Kevin Frans, Jonathan Ho, Xi Chen, Pieter Abbeel, and John Schulman. Meta learning shared hierarchies. *CoRR*, abs/1710.09767, 2017. URL http://arxiv.org/abs/1710.09767.

William H. Guss, Mario Ynocente Castro, Sam Devlin, Brandon Houghton, Noboru Sean Kuno, Crissman Loomis, Stephanie Milani, Sharada Mohanty, Keisuke Nakata, Ruslan Salakhutdinov, John Schulman, Shinya Shiroshita, Nicholay Topin, Avinash Ummadisingu, and Oriol Vinyals. The minerl 2020 competition on sample efficient reinforcement learning using human priors, 2021.

Danijar Hafner. Benchmarking the spectrum of agent capabilities, 2022.

Geoffrey E. Hinton, Oriol Vinyals, and Jeffrey Dean. Distilling the knowledge in a neural network. *ArXiv*, abs/1503.02531, 2015.

Max Jaderberg, Valentin Dalibard, Simon Osindero, Wojciech M. Czarnecki, Jeff Donahue, Ali Razavi, Oriol Vinyals, Tim Green, Iain Dunning, Karen Simonyan, Chrisantha Fernando, and Koray Kavukcuoglu. Population based training of neural networks, 2017.

Michał Kempka, Marek Wydmuch, Grzegorz Runc, Jakub Toczek, and Wojciech Jaśkowski. Vizdoom: A doom-based ai research platform for visual reinforcement learning, 2016.

Vijay Konda and John Tsitsiklis. Actor-critic algorithms. In S. Solla, T. Leen, and K. Müller (eds.), *Advances in Neural Information Processing Systems*, volume 12. MIT Press, 2000. URL https://proceedings.neurips.cc/paper/1999/file/6449f44a102fde848669bdd9eb6b76fa-Paper.pdf.

Tejas D. Kulkarni, Karthik R. Narasimhan, Ardavan Saeedi, and Joshua B. Tenenbaum. Hierarchical deep reinforcement learning: Integrating temporal abstraction and intrinsic motivation, 2016.

Heinrich Küttler, Nantas Nardelli, Thibaut Lavril, Marco Selvatici, Viswanath Sivakumar, Tim Rocktäschel, and Edward Grefenstette. TorchBeast: A PyTorch Platform for Distributed RL. *arXiv preprint arXiv:1910.03552*, 2019. URL https://github.com/facebookresearch/torchbeast.

Heinrich Kuttler, Nantas Nardelli, Alexander H. Miller, Roberta Raileanu, Marco Selvatici, Edward Grefenstette, and Tim Rocktäschel. The nethack learning environment, 2020.

Joel Lehman and Kenneth Stanley. Exploiting open-endedness to solve problems through the search for novelty. *Artificial Life - ALIFE*, 01 2008.

Andrew Levy, G. Konidaris, Robert W. Platt, and Kate Saenko. Learning multi-level hierarchies with hindsight. In *ICLR*, 2019.

Ofir Nachum, Shixiang Gu, Honglak Lee, and Sergey Levine. Data-efficient hierarchical reinforcement learning, 2018.

Emilio Parisotto, Lei Jimmy Ba, and Ruslan Salakhutdinov. Actor-mimic: Deep multitask and transfer reinforcement learning. In *ICLR*, 2016.

Sam Powers, Eliot Xing, Eric Kolve, Roozbeh Mottaghi, and Abhinav Gupta. Cora: Benchmarks, baselines, and metrics as a platform for continual reinforcement learning agents, 2021.

Andrei A. Rusu, Sergio Gomez Colmenarejo, Çaglar Gülçehre, Guillaume Desjardins, James Kirkpatrick, Razvan Pascanu, Volodymyr Mnih, Koray Kavukcuoglu, and Raia Hadsell. Policy distillation. In *ICLR*, 2016.

Mikayel Samvelyan, Robert Kirk, Vitaly Kurin, Jack Parker-Holder, Minqi Jiang, Eric Hambro, Fabio Petroni, Heinrich Kuttler, Edward Grefenstette, and Tim Rocktäschel. Minihack the planet: A sandbox for open-ended reinforcement learning research, 2021. URL https://openreview.net/forum?id=skFwlyefkWJ.

Simon Schmitt, Jonathan J. Hudson, Augustin Zídek, Simon Osindero, Carl Doersch, Wojciech M. Czarnecki, Joel Z. Leibo, Heinrich Küttler, Andrew Zisserman, Karen Simonyan, and S. M. Ali Eslami. Kickstarting deep reinforcement learning. *CoRR*, abs/1803.03835, 2018. URL http://arxiv.org/abs/1803.03835.

Richard S. Sutton, Doina Precup, and Satinder Singh. Between mdps and semi-mdps: A framework for temporal abstraction in reinforcement learning. *Artif. Intell.*, 112(1–2):181–211, August 1999. ISSN 0004-3702. doi: 10.1016/S0004-3702(99)00052-1. URL https://doi.org/10.1016/S0004-3702(99)00052-1.

Yee Whye Teh, Victor Bapst, Wojciech Marian Czarnecki, John Quan, James Kirkpatrick, Raia Hadsell, Nicolas Heess, and Razvan Pascanu. Distral: Robust multitask reinforcement learning, 2017.

Alexander Sasha Vezhnevets, Simon Osindero, Tom Schaul, Nicolas Heess, Max Jaderberg, David Silver, and Koray Kavukcuoglu. Feudal networks for hierarchical reinforcement learning, 2017.

Name	Key
PickUp	,
PutOn	Shift + p
Zap	z
TakeOff	Shift + t
Wear	Shift + w
Throw	t
Escape	Esc
Eat	e
Apply	a
Wield	w
Quaff	q
Gold	$
Inv1	f
Inv2	g
Inv3	h

Table 2: Command Actions

A ACTION SPACE

NetHack takes ASCII characters as its inputs. These can be modified with certain keys like Ctrl and Shift. We use a reduced action space of size 32, only allowing actions that have been deemed relevant for completing the benchmark. This includes 16 movement actions consisting of the 4 cardinal directions, the 4 diagonal directions each with a single and long movement option. The remaining actions are summarised in Table 2.

B KICKSTARTING COEFFICIENT DISCUSSION

The Kickstarting coefficient λ was set with a manual schedule defined in Section 5.1. Notably, we never let the coefficient drop to 0, but instead specify a minimum value of 0.1, meaning that the Kickstarting loss is always present. The reason for this was twofold. Firstly, we saw that large spikes in performance would often occur after the coefficient had reached its minimum value of 0.1. Therefore, keeping a small minimum allowed Kickstarting and HKS to work without going through the lengthy process of tuning the schedule for each environment. Secondly, when the coefficient was allowed to drop to 0, we would see infrequent massive drops in performance with both Kickstarting and HKS (Around 1 in 2 runs would at some point lose around half of their performance after converging). The reason for this is not entirely clear, but is an area of current investigation.

This artificial minimum on the coefficient effectively imposes a cap on the performance of Kickstarting, as it is always slightly being optimised away from following extrinsic reward. HKS is not as susceptible to this effect, as the policy-over-teachers can converge such that the weighted average of teacher action distributions closely matches that which would maximise extrinsic reward anyway. This can be seen in the plot of model performances (Figure 4) for the tasks BATTLE, PREPARE FOR BATTLE and IDENTIFY MIMIC. In all of these tasks, HKS converges to a slightly higher success rate than Kickstarting. To account for this discrepancy, we take the average percentage difference in the final performances of Kickstarting and HKS on these three tasks, which works out to be 2.2%. The final performance for Kickstarting averaged over all tasks is 39.9%. We therefore increase this value by 2.2% giving us 40.8%, giving us a fairer comparison against HKS.

C SKILL LISTING

The list of skills is shown in Table 3. Each skill has an environment where it is learned in isolation, shown in Figures 5 and 6. It should be noted that many of the environments visually look the same.

Name	Description
ApplyFrostHorn	Use a frost horn to freeze some lava.
Eat	Eat an apple.
Fight	Hit a monster.
NavigateBlind	Reach the staircase while blinded.
NavigateLava	Reach the staircase past random lava patches.
NavigateLavaToAmulet	Reach an amulet past random lava patches.
NavigateWater	Reach the staircase past random water patches.
PickUp	Pick up a random item.
PutOn	Put on an amulet or towel.
TakeOff	Take off clothes.
Throw	Throw daggers at a statue or at a monster.
Unlock	Use a key to unlock a locked door.
Wear	Wear a robe.
Wield	Wield a sword.
ZapCold	Use a wand of cold to freeze lava.
ZapWoD	Use a wand of death to kill a monster.

Table 3: Skills contained in the benchmark.

This is because many environments focus on manipulating items in the players inventory, with the rest of the world only serving as a distraction which the agent must learn to reliably ignore.

D TASK LISTING

The task descriptions can be found in Table 1. The environments are shown in Figures 7 and 8.

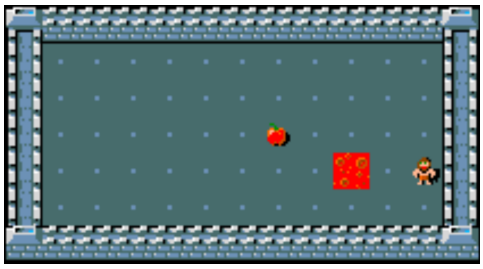
E FURTHER RESULTS

The experts were all trained on the skill acquisition environments for 7.5×10^7 timesteps. The results are shown in Figure 9.

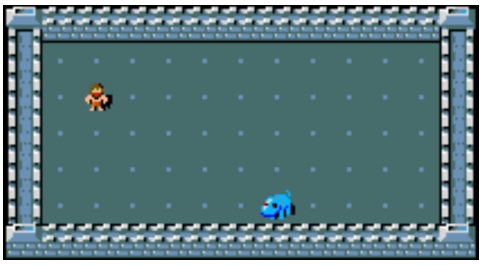
F HYPERPARAMETERS

The hyperparameters for the IMPALA agent are summarised in Table 4.

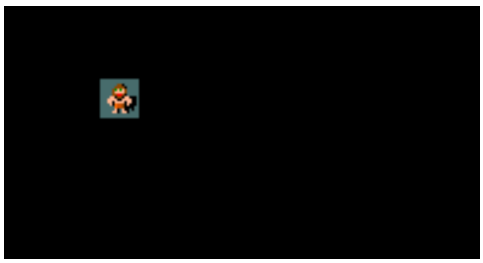
(a) ApplyFrostHorn



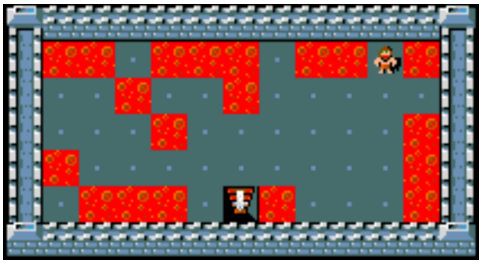
(b) Eat



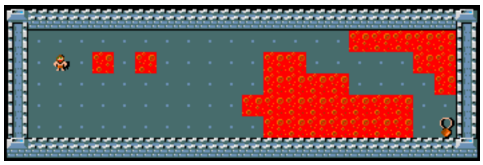
(c) Fight



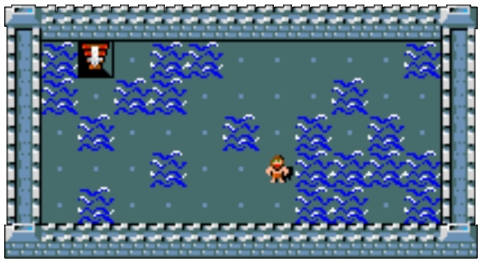
(d) NavigateBlind



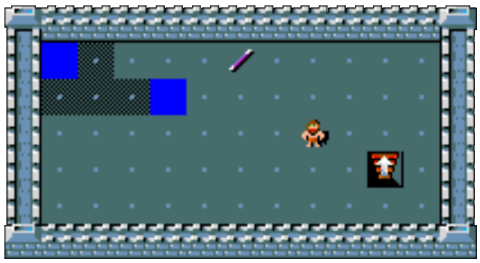
(e) NavigateLava



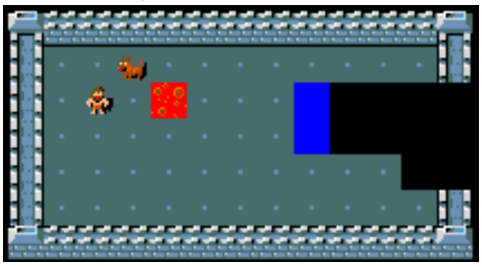
(f) NavigateLavaToAmulet



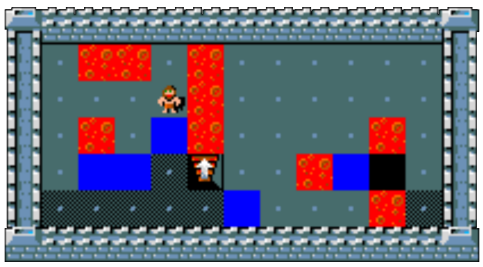
(g) NavigateWater



(h) PickUp

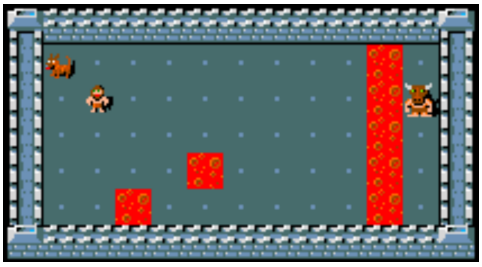


(i) PutOn

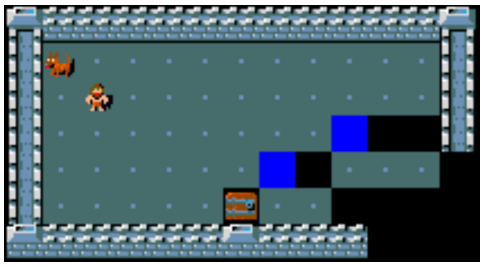


(j) TakeOff

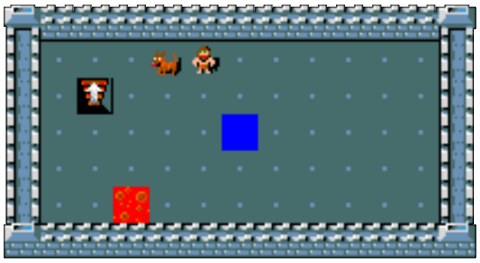
Figure 5: Skill environments.



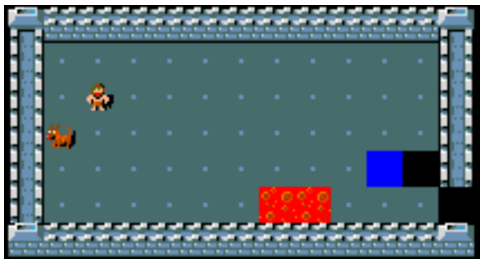
(a) Throw



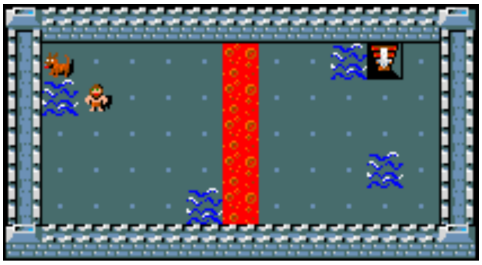
(b) Unlock



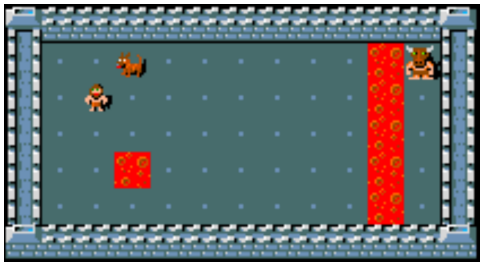
(c) Wear



(d) Wield

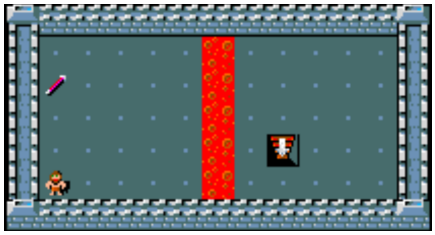


(e) ZapWandOfCold

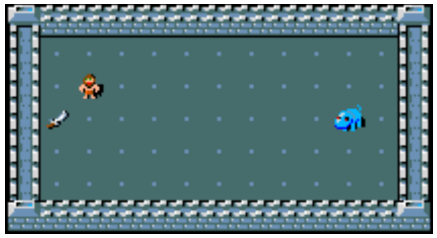


(f) ZapWandOfDeath

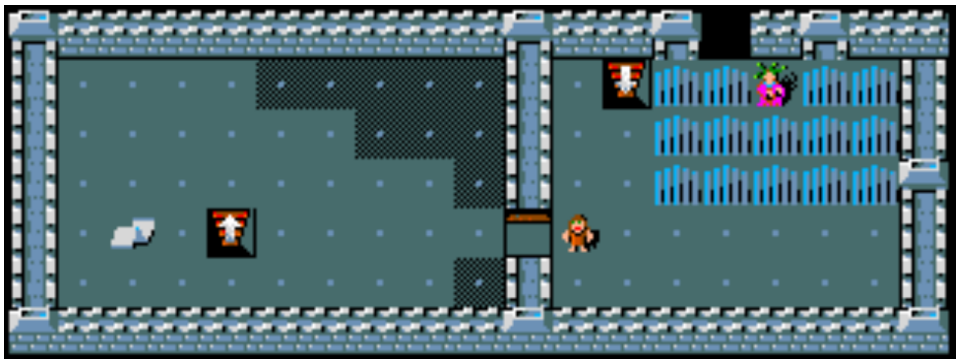
Figure 6: Skill environments (continued).



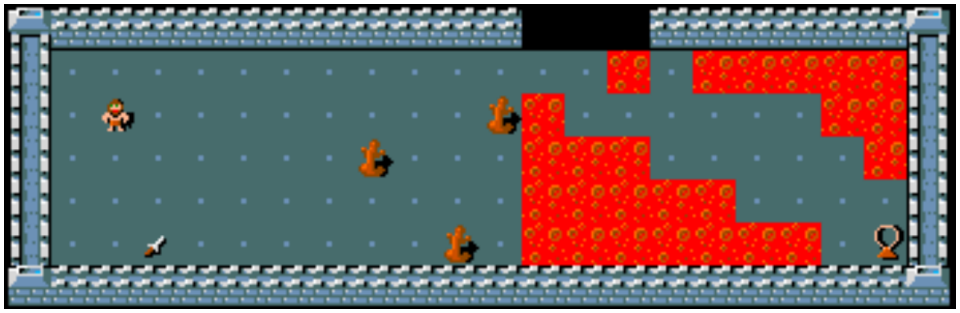
(a) FROZEN LAVA CROSS



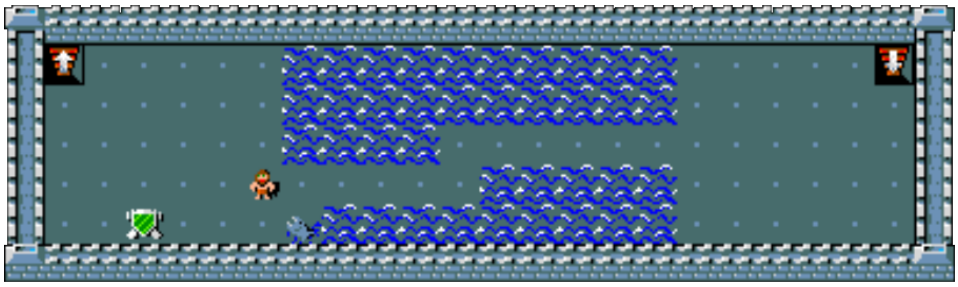
(b) BATTLE



(c) MEDUSA

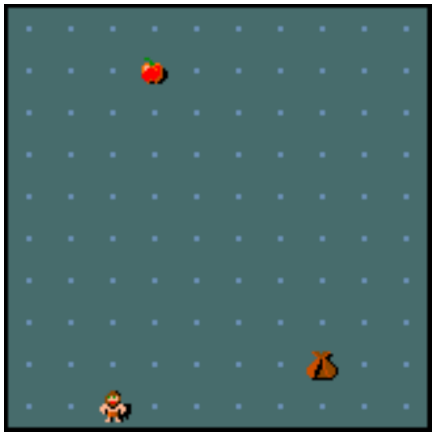


(d) IDENTIFY MIMIC

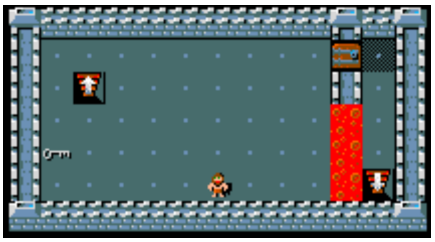


(e) SEA MONSTERS

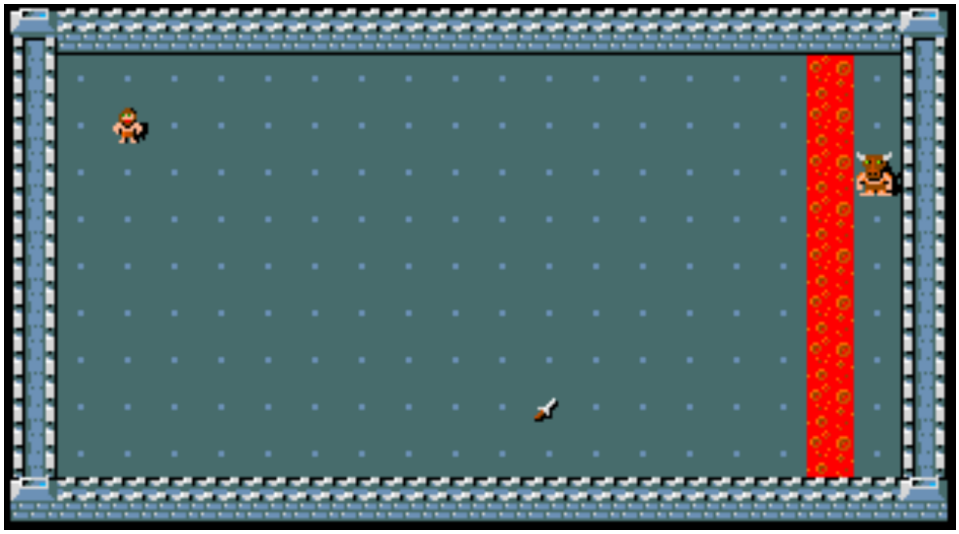
Figure 7: Task environments.



(a) PREPARE FOR BATTLE



(b) A LOCKED DOOR



(c) TARGET PRACTICE

Figure 8: Task environments (continued).

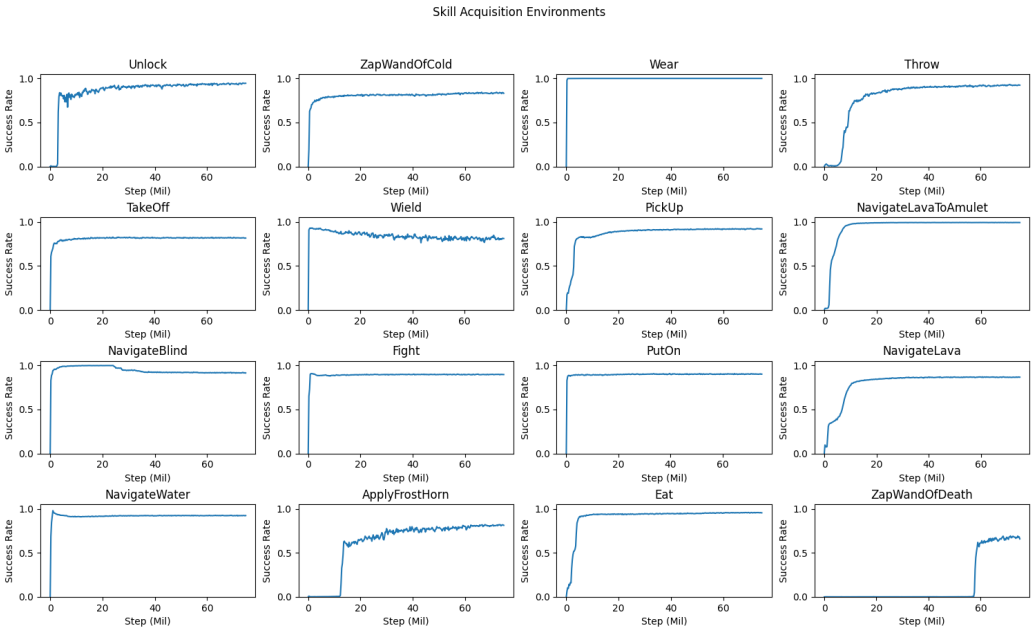


Figure 9: Success rate for training skill experts on the 16 skill acquisition environments

Name	Value
Training Settings	
num_actors	256
batch_size	32
unroll_length	80
Model Settings	
hidden_dim	256
embedding_dim	64
glyph_type	all_cat
equalize_input_dim	false
layers	5
crop_model	cnn
crop_dim	9
use_index_select	true
max_learner_queue_size	1024
Loss Settings	
entropy_cost	0.001
baseline_cost	0.5
discounting	0.999
reward_clipping	none
normalize_reward	true

Name	Value
Optimizer Settings	
learning_rate	0.0002
grad_norm_clipping	40
Intrinsic Reward Settings	
int.twoheaded	true
int.input	full
int.intrinsic_weight	0.1
int.discounting	0.99
int.baseline_cost	0.5
int.episodic	true
int.reward_clipping	none
int.normalize_reward	true

Table 4: Hyperparameters of the IMPALA agent.