META ARCADE: A CONFIGURABLE ENVIRONMENT SUITE FOR DEEP REINFORCEMENT LEARNING AND META-LEARNING

Edward W. Staley, Chace Ashcraft, Benjamin Stoler, Jared Markowitz,
Gautam Vallabha, Christopher Ratto, Kapil D. Katyal
Johns Hopkins University Applied Physics Laboratory
Laurel, MD 20723, USA
{edward.staley,chace.ashcraft,benjamin.stoler,jared.markowitz,
gautam.vallabha,christopher.ratto}@jhuapl.edu, kkatyal2@jhu.edu

ABSTRACT

Most approaches to deep reinforcement learning (DRL) attempt to solve a single task at a time. As a result, most existing research benchmarks consist of individual games or suites of games that have common interfaces but little overlap in their perceptual features, objectives, or reward structures. To facilitate research into knowledge transfer among trained agents (e.g. via multi-task and meta-learning), more environment suites that provide configurable tasks with enough commonality to be studied collectively are needed. In this paper we present Meta Arcade, a tool to easily define and configure custom 2D arcade games that share common visuals, state spaces, action spaces, game components, and scoring mechanisms. Meta Arcade differs from prior environments in that both task commonality and configurability are prioritized: entire sets of games can be constructed from common elements, and these elements are adjustable through exposed parameters. We include a suite of 24 predefined games that collectively illustrate the possibilities of this framework and discuss how these games can be configured for research applications. We provide several experiments that illustrate how Meta Arcade could be used, including single-task benchmarks of predefined games, sample curriculum-based approaches that change game parameters over a set schedule, and an exploration of transfer learning between games.

ACKNOWLEDGEMENTS

Primary development of Meta Arcade was funded by the DARPA Lifelong Learning Machines (L2M) Program. Additionally, this work relates to Department of Navy award N00014-20-1-2239 issued by the Office of Naval Research. The United States Government has a royalty-free license throughout the world in all copyrightable material contained herein.

1 INTRODUCTION

Early deep reinforcement learning (DRL) (Mnih et al., 2013; Schulman et al., 2015; Lillicrap et al., 2016; Silver et al., 2016; Schulman et al., 2017) algorithms focused on performance on very narrowly-defined tasks. Accordingly, existing benchmark environments for DRL tend to be narrow in scope. For example, DQN Mnih et al. (2013), one of the first works to combine deep neural networks and reinforcement learning, played individual Atari 2600 games using the Atari Learning Environment (ALE) (Bellemare et al., 2013). These games have little in common with each other, making them unsuitable for multitask learning or meta-learning (as noted in Yu et al. (2019)). Moving away from such narrow training domains remains a very difficult challenge, requiring environments that allow a graceful transition to multiple tasks and expose precise control over the underlying distributions.

In this work we present Meta Arcade, an attempt to address this need. Meta Arcade is a suite of lightweight environments designed for multi-task learning and meta-learning, with an emphasis on configurability. Game elements, colors, dynamics, and objectives are all parameterized with support for distribution sampling, such that a single game can be varied slightly or altered to the extent that it constitutes a new task. Common arcade game objectives like collecting blocks, bouncing balls, and avoiding obstacles are re-used between games. If multiple tasks are defined, these common objectives will result in clear overlap between the games, making them viable for multitask approaches.

The Meta Arcade framework and suite provides two major assets to researchers. First, an environment for RL that is fully parameterized so that it can be tuned to the needs of individual studies, including examinations of task variation or domain shift as a direct focus. Second, an entire space of arcade-inspired tasks that have common parameters, enabling research that requires sets of distinct but related tasks. We have defined 24 unique games that can be constructed using Meta Arcade, and describe how others may be built. Meta Arcade is available on github (https://github.com/darpa-l2m/meta-arcade) with accompanying documentation and examples. Additionally, we contribute a series of experiments that include acquiring single-task expertise, curriculum-based learning, and multitask learning.

This manuscript details the capabilities of Meta Arcade and the results of our experiments. In Section 2 we discuss other DRL environments and how they support the current research landscape. In Section 3 we describe Meta Arcade in detail and how it can be used. Benchmarks and other experimental results are presented in Section 4, and a discussion of future work is presented in Section 5.

2 Related Work

There are a variety of environments that currently support DRL research, both for single tasks and, to a lesser extent, for multiple tasks. In addition to the discrete setting of ALE, DDPG (Lillicrap et al., 2016) introduced several 2D continuous control tasks and used MuJoCo (Todorov et al., 2012) as a platform for building such environments. Later MuJoCo was used to build 3D environments, as introduced alongside generalized advantage estimation (Schulman et al., 2018). ALE, MuJoCo environments, classic control environments, and others are maintained in the OpenAI Gym (Brockman et al., 2016).

These benchmark environments tend to present narrow tasks, which lead to policies that are hyperfit to a specific problem (Zhang et al., 2018; Parisotto et al., 2016). This concern can be addressed algorithmically, but may be more naturally addressed by broadening the scope of the environments. One common avenue towards robustness is to envision a task not as a static problem but rather as a distribution of task parameters, from visual distortions to changing dynamics (Zhao et al., 2020). This general idea, termed domain randomization, was first presented for supervised learning in Tobin et al. (2017) and laid the groundwork for policy-based success in the robotic manipulation of a Rubik's Cube (OpenAI et al., 2019). Notably, Procgen (Cobbe et al., 2019), which includes the game Coin Run (Cobbe et al., 2018), defines tasks that are built procedurally and have key visuals randomized.

If a target environment is difficult to learn but can be modified to create more easily-learned variants, it can be useful to supply a learning curriculum (Bengio et al., 2009; MacAlpine & Stone, 2018; Narvekar et al., 2020; Luo et al., 2020), which may require some control over the underlying environment. ViZDoom (Kempka et al., 2016) and DeepMind Lab (Beattie et al., 2016) both provide tools for creating custom challenges. However, it is difficult to build out new tasks for these platforms, and ultimately all scenarios look more like variations of the original game than distinct tasks. Explicit simulation frameworks like Unity (Juliani et al., 2018) and MuJoCo provide the means to create entirely new environments or modify existing examples, but require full development efforts.

Multi-task learning examines training competent agents on several tasks simultaneously or sequentially, rather than solely on instances of a single task. Ideally, the different tasks have features or skills that overlap such that there is an advantage to learning them together Parisotto et al. (2016); Yu et al. (2019). While a suite of environments is needed for multi-task RL research, ensuring that tasks are sufficiently similar or have sufficiently overlapping skills to enable transfer is difficult. Works such as Meta-World (Yu et al., 2019) and the Sonic benchmark (Nichol et al., 2018b) have been designed with this requirement in mind. Suites which are based around a common game, such as the StarCraft II Learning Environment (Vinyals et al., 2017) minigames, tend to inherently include this task overlap. This overlap is also useful if the tasks will be learned in sequence, as in lifelong learning and its variations. Alchemy (Wang et al., 2021) takes these ideas further by re-sampling the underlying game structure each episode, such that every episode is a different draw from a common task distribution.

Finally, another approach to producing more generalizable agents is meta-learning, or “learning-to-learn” (Schmidhuber, 1987; Hospedales et al., 2020), which enables adaptation to new tasks in a structured way. The meta-learning objective may be to determine what knowledge is transferable between tasks, to learn how to transfer that knowledge, or to learn how to do so efficiently, with the goal of quickly becoming proficient at a new task. Current research questions in this area include how to apply common meta-learning techniques (Finn et al., 2017; Nichol et al., 2018a; Rakelly et al., 2019) to problems with discrete action spaces and to problems with sparse rewards. Meta-World (Yu et al., 2019) and Alchemy (mentioned above) are two recent environments that were specifically designed to support meta-learning research.

The goal of Meta Arcade is to enable further exploration of multitask and meta-learning by providing game environments that are useful for these burgeoning research areas. Meta Arcade has the variety, speed, and ease of use that accompanies arcade games as a research benchmark, while providing the rich configurability and shared task structure needed for multitask learning, meta-learning, and related topics.

3 Meta Arcade

3.1 Environment Suite

The Meta Arcade environment suite seeks to facilitate many of the above research directions through a highly flexible set of tools for training and evaluating DRL algorithms. A primary focus was to create a suite of arcade games with many areas of overlap, such that the collection has a diversity of tasks similar to ALE while having enough commonality among tasks to be suitable for multi-task learning and meta-learning. Each game is parameterized and can be easily modified or extended.

Meta Arcade includes 24 predefined games that have the following characteristics: All games present a 84x84x3 pixel-based state space, and all games have identical action spaces (6 discrete actions). Some games will not use all 6 actions, but will instead map unused actions to no-ops internally. A similar entity is controlled in all games, and the actions have identical effects in all games where they are used. Continuous actions are also supported as an option. An overview of the game display is shown in Figure 1. Meta Arcade was built with Pygame[1], a platform-agnostic Python API for video game development, making Meta Arcade immediately accessible to developers already using Python deep-learning frameworks. Meta Arcade can optionally be run headlessly.

Games are constructed from a common set of game mechanics, visuals, and scoring systems. Games have a variety of overlapping skills such that a policy to solve one game is potentially useful for others. A sampling of games with some highlighted commonalities can be seen in Figure 1. Across games, entities will have similar interactions with other objects and incur similar rewards. The maximum score for all games is 100 points and the minimum score is either 0 or -100, depending on whether it is possible to lose the game (as opposed to the game timing out, which happens after several thousand frames). Rewards are event-driven and sparse. Common scoring mechanisms are built around this scale and are summarized in the table below.

There are no lives in Meta Arcade, as past approaches to solving arcade games often manually split episodes among lives anyway (Mnih et al., 2013; 2015). Block-collecting games divide 100 points among some target number of blocks. This leads to some initially unintuitive results for single games in which a loss is heavily weighted against progress. For example, collecting 19 out of 20 blocks in breakout results in a score of -5 (+5 points x 19 and -100 for missing the ball). This does not prevent learning of the games but does require care when examining results. We believe this is a reasonable tradeoff in exchange for a common scoring system among all games, which may aid in the learning of value functions with similar magnitudes.

[1] `https://www.pygame.org`

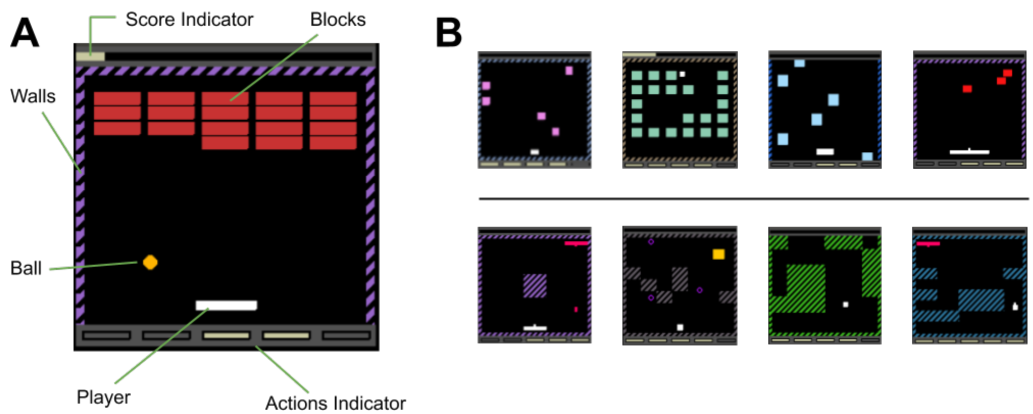


Figure 1: Some features of Meta Arcade. **A:** The game *breakout* with some highlighted elements, including the visual indicators that are outside of the game play area. **B:** Examples of common elements among games. The top row shows some games that feature solid-color collectable blocks, while the bottom row shows static walls that have a common visual texture. Several other elements are seen throughout the game suite, such as hazards, balls, or opponents.

Game Event	Score	Game Event	Score
Player passes opposite side	+100 (win)	Player hits collectable block	+ Block Value
Player hits hazard	-100 (loss)	Ball passes opponent side	+100 (win)
Ball passes player side	-100 (loss)	Ball hits collectable block	+ Block Value
Bullet hits collectable block	+ Block Value	Bullet hits opponent	+100 (win)
Bullet hits player	-100 (loss)	Block falls passed player	-100 (loss)

Table 1: Reward mechanisms in predefined Meta Arcade games

The predefined games in Meta Arcade present a range of difficulties. This spectrum is presented intentionally to provide easy games for rapid testing and challenge games to motivate future research. Not all games are necessarily solvable in isolation, and may require curricula, transfer learning, or other advanced methods to solve.

3.2 Capabilities for Modern DRL Research

While many environments are presented to the research community as static problems, there is often a close collaboration in the creation of new algorithms and new environments. Environments may need to be specially built to explore a research idea, or a novel environment may spark a new research direction. With this in mind, Meta Arcade games are editable and are designed to be tweaked as needed for a given research avenue. While many predefined games are supplied, Meta Arcade can be thought of as a game creation tool as much as a suite of specific challenges. New games can easily be created from scratch or by editing an existing game.

Each game is entirely defined by a JSON file (see Figure 5 in the appendix) which lists the relevant game mechanics and their attributes. The sizes, speeds, colors, and behaviors of any game can be directly changed through these files, or modified using environment tools from python scripts. Furthermore, any value in the game definitions can be replaced with a distribution to be sampled from rather than a static value. Gaussian distributions, uniform distributions, and special color distributions are supported. This enables exploration of task distributions, domain randomization, and domain shifts. Some examples of parameter changes can be found in Figure 2. Options are also provided for image-post processing, including hue, value, and saturation shifts, color inversion, and image rotation.

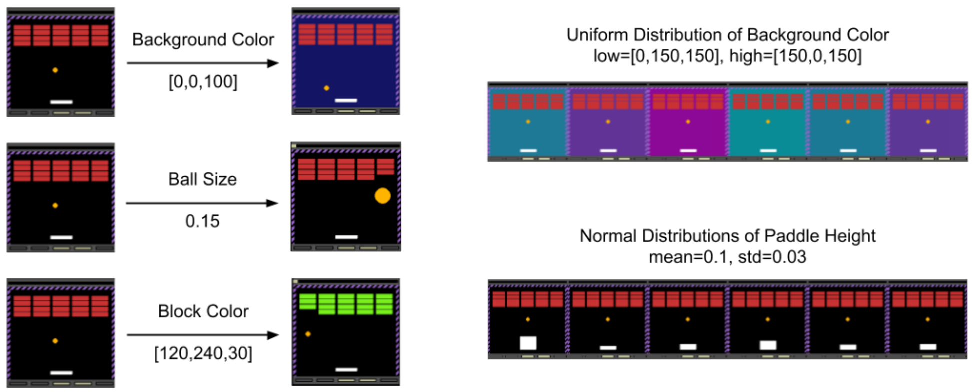


Figure 2: Parameter changes to breakout. All colors, sizes, speeds, and other behaviors can be easily modified to customize games for experimentation. Some static value changes are shown on the left, some of which are purely visual while one changes the gameplay itself. Almost all parameters can be replaced with distributions which are re-sampled each episode. The effects of using distributions can be seen in the examples on the right.

Additionally, Meta Arcade includes some simple tools for building training curricula based on multiple tasks, or many copies of a task with shifted parameters. Games may be grouped together in a pool of tasks to be sampled, repeated for a specific number of episodes or steps, or interpolated between in the configuration parameter space. Together with access to game parameters, this enables the construction of tasks which may change over time, may be interwoven with other tasks, or may be combined in other manners. This type of functionality is essential to explore areas like continual learning, transfer learning, and curriculum learning. Some examples of the environment's behavior under simple curricula can be seen in Figure 3.

The most powerful and nuanced feature of the curriculum tools is the ability to linearly interpolate between game parameters, including those defined by distributions. Over a set number of episodes, two games defined by identically keyed JSON fields can be interpolated between at each reset call. If a distribution is specified, the parameters of that distribution are interpolated. For example, the standard deviation of an object's size could be changed over time, or the number of blocks could be sampled from a uniform distribution that has changing bounds. This is incredibly powerful for slowly changing the difficulty of a game or controlling domain randomization as a function of time.

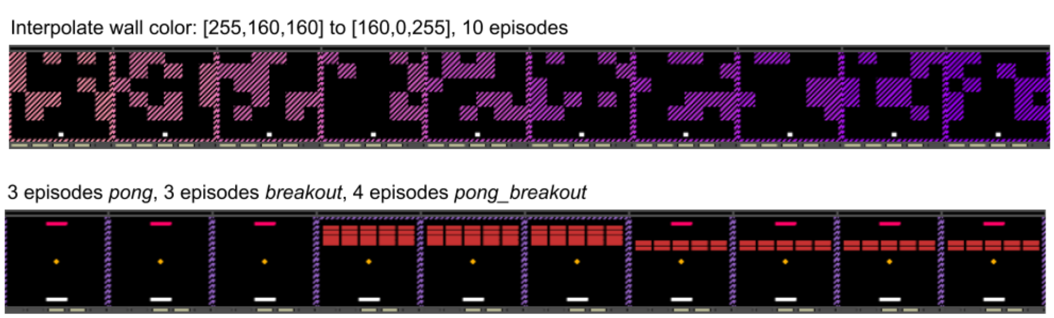


Figure 3: Simple example curricula using built-in tools. A variety of curriculum features are available to create training regimens from Meta Arcade. The examples above are limited to 10 episodes for demonstration purposes, but long sequences of tasks and their variations can be constructed. Curricula can optionally be defined in terms of steps rather than episodes.

4 EXPERIMENTS AND RESULTS

We present the results of several experiments to demonstrate the range of features available in Meta Arcade and their applications to DRL research. These experiments can be broken into three sections: (1) Single-task benchmarking of all predefined games using PPO (Schulman et al., 2017) (This also serves as a summary of all available games), (2) Selected curricula-based approaches for challenging games and the exploration of domain distributions, and (3) Transfer learning experiments on subsets of the games.

4.1 PPO BENCHMARKING

To demonstrate the solvability of predefined games and provide a point of reference for other researchers, we used Proximal Policy Optimization (PPO) (Schulman et al., 2017) to try to learn each game separately. This is useful not only for understanding the difficulty of a given game but also proved useful for designing the games themselves and even debugging. In an effort to provide a range of reasonable difficulty, several games were tweaked in an iterative process with PPO benchmarking. In some respects, this experiment could be considered part of the environment design process. A full table of the composition of each predefined game can be found in the appendix (Table 6).

We trained PPO once on each game for 10 million frames, using 8 workers distributed with the Message Passing Interface (MPI) (Forum, 1994; Tesser, 2016). Gradients are computed in each worker and averaged in the main process. A standard network architecture for Atari was used (Mnih et al., 2013). Specific hyperparameters can be found in the appendix, Section A.1.

The results from benchmarking all 24 games can be clearly split into those that were successful and unsuccessful, with ”unsuccessful” meaning that trained agent failed to consistently reach a positive score. 18 of the 24 were successfully learned, often with nearly maximal performance. The training results of the successful games are summarized in Table 2. The only games which were not clearly successful in this category are Juggling, Keep Ups, and Pong Breakout. However, performance was high enough to be considered successful learning. The six unsuccessfully learned games can be viewed in Table 3. These games tended to converge to average scores around -100 or 0, corresponding to complete failure or avoiding play entirely. These games appear to be challenging because it is very hard to discover rewards naturally or without incurring some penalty, which ultimately discourages learning the needed behaviors for success.

4.2 CURRICULA

In an effort to resolve the learning of some of the challenging problems seen in benchmarking and demonstrate some advanced features of the Meta Arcade environment suite, we designed learning curricula for Dungeon, Freeway, Invasion, Lava Maze, and Tunneler. These curricula span the original 10 million training frames but introduce the games’ full complexities over time. Haunted Hallway is omitted because it was unclear how to create a curriculum that did not change the game’s design. The detailed results of the curriculum experiments can be found in Table 5 in the Appendix.

The general form of each curriculum was to start with what we believe to be an easier set of parameters that would allow early success (in other words, parameters that allow reward discovery with random actions). Over the first 4 million frames, this parameter distribution is gradually expanded and interpolated to include increasingly difficult parameters during training, but without necessarily removing the easier versions from the set of possibilities. From 4 million to 8 million frames, the easy parameters are slowly excluded to focus on the specific parameters that describe the full game. Some difficulty may still be introduced during this period to completely match the full game. From 8 million steps onward, the original game is played with no alteration. The last entries in Table 5 illustrate how the games may change during training according to the high-level schedule described above.

The curricula approach proved very successful in training most of the challenging games. All the games show final performances that reflect a high level of skill, with average scores far above their PPO benchmark counterparts. The key trend during training seems to be that the policy quickly reaches strong play on the easy game parameters, and then tries to maintain high performance as the

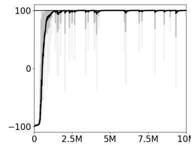
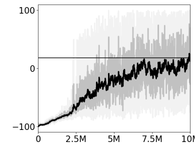
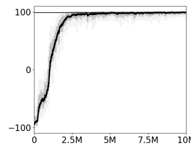
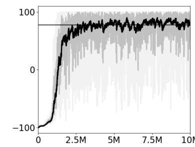
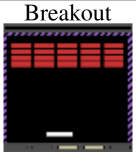
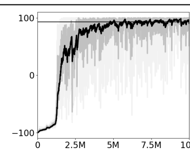
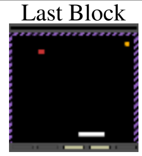
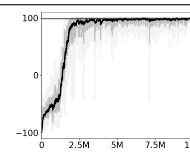
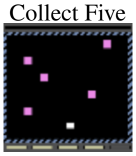
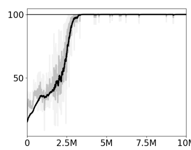
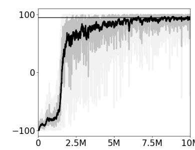
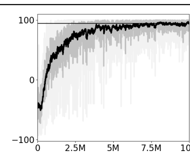
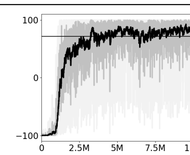
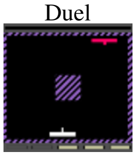
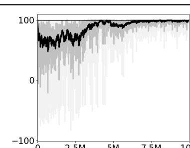
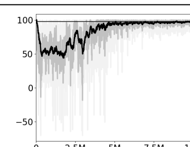
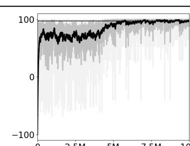
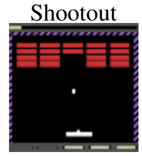
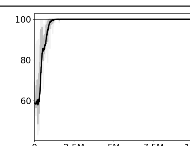
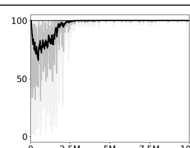
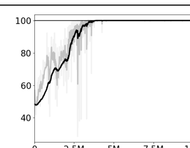
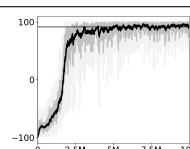
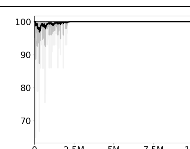
Game	Training Curve	Score	Game	Training Curve	Score
Avalanche 		100.00 ± 0.00	Juggling 		17.98 ± 68.33
Battle Pong 		99.03 ± 3.00	Keep Ups 		77.60 ± 51.14
Breakout 		93.02 ± 28.06	Last Block 		98.02 ± 7.03
Collect Five 		100.00 ± 0.00	Pong 		94.32 ± 22.35
Dodgeball Duel 		94.21 ± 13.43	Pong Breakout 		71.33 ± 53.18
Duel 		99.23 ± 3.80	Seek Destroy 		97.65 ± 5.32
Erosion 		97.43 ± 10.33	Shootout 		100.00 ± 0.00
Hedge Maze 		100.00 ± 0.00	Sweeper 		100.00 ± 0.00
Interference 		91.12 ± 15.64	Target Practice 		100.00 ± 0.00

Table 2: Predefined games that are clearly solvable with PPO, and their training performance. Different shades of gray show different amounts of smoothing.

game parameters become more difficult. Note that the Invasion curriculum leads to a slow ramp-up in score since the early games have fewer blocks than are needed to end the game (a score of 100 is

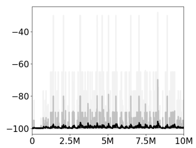
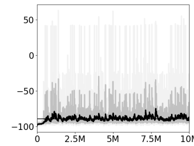
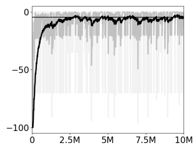
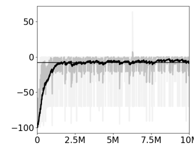
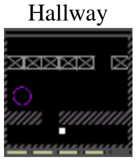
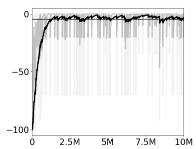
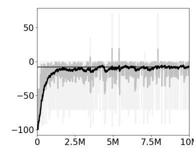
Game	Training Curve	Score	Game	Training Curve	Score
Dungeon 		-99.50 ± 3.50	Invasion 		-89.17 ± 16.91
Freeway 		-4.50 ± 20.12	Lava Maze 		-8.00 ± 27.13
Haunted Hallway 		-4.50 ± 20.12	Tunneler 		-8.00 ± 26.19

Table 3: Challenge games and their single-task PPO performance.

not actually possible until further into the curriculum). Several games also show a performance dip that coincides with a switch to the final game, which may indicate that the transition to the full game should have been drawn out over more frames for stability.

4.3 Transfer Learning for Multiple Games

While the curriculum experiments explored variations within a single task, our final set of experiments serve as exploratory examples of transfer learning between different tasks in Meta Arcade. We focus on predefined games with a common color scheme to ease transfer. Future work could examine transfer between color schemes or domain randomization over colors – both are easily implemented given the configurability of the environment.

We first select three predefined games which include skills that could readily transfer to other games. The three games are learned together with a single PPO policy (multitask learning) for 50 million steps. We use a model checkpoint from 30 million steps as a starting point for additional games that may have common skills or features. The decision to use the 30 million step checkpoint was inspired by Parisotto et al. (2016), which notes that training for a long time may lead to overfit policies that that do not transfer well between tasks. Unlike Parisotto et al. (2016), we did not re-initialize the final layer or change the model size, as this experiment is exploratory. When attempting transfer, the new games were trained for 1 million steps, as in Nichol et al. (2018b).

The first experiment examined transfer between paddle-based games. The three multitask games were Breakout, Duel, and Pong, which demonstrate a variety of game objectives that may include a paddle for the player. These three tasks were trained jointly to create a single policy that performed well on all three games. Transfer from this policy was attempted to Battle Pong, Erosion, Pong Breakout, and Shootout independently. These results can be seen in Figure 4 (Top).

The second experiment examined recombination of more diverse skills, including two-dimensional movement. The three multitask games were Collect Five (demonstrating block collection), Hedge Maze (demonstrating the traversal objective), and Erosion (demonstrating shooting). Transfer was attempted to Avalanche, Dungeon (without hazards), Seek Destroy, and Tunneler. These results can be seen in Figure 4 (Bottom).

The paddle-based experiments showed clear transfer among the games using a bouncing ball. This mechanism appears in both Breakout and Pong, which were closely tied during multitask training, indicating that this common skill was possibly learned in a generic way to satisfy both games. When transferring to Battle Pong and Pong Breakout, there was a significant benefit over starting from scratch. There was little benefit when transferring to Erosion or Shootout, which are shooting-based

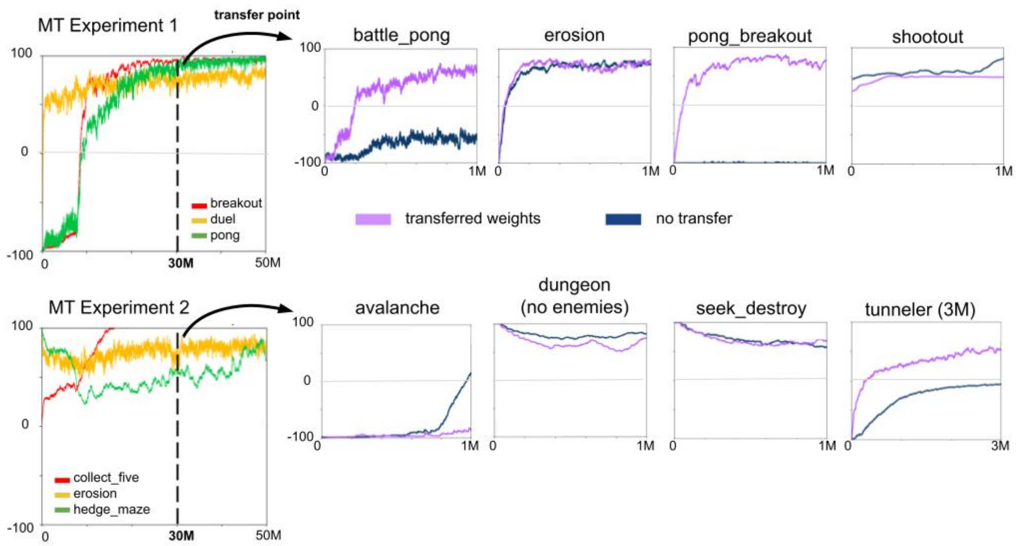


Figure 4: **Top:** Multitask learning and transfer among selected paddle-based games. **Bottom:** Multitask learning and transfer among selected games demonstrating a range of skills

games. However, these games are learned incredibly quickly and perhaps a more difficult game was needed to visualize the importance of transfer.

The second experiment using a more diverse range of skills had less clear results. There was no observed benefit to transfer aside from the game Tunneler, which was trained for 3 million steps instead of 1 million to flush out this result. This was surprising, as Tunneler is one of the more challenging predefined games in the suite, appearing unsolvable by PPO in baselining. Presumably, the combination of Hedge Maze (reach the other side) and Erosion (destroy blocks) allowed the transfer learner to more readily experience the goal, which lead to successful training. How to best choose what games to include in a meta-learning curriculum is an area of research worthy of further exploration with Meta Arcade.

5 Conclusions and Future Work

In this paper we presented Meta Arcade, a tool to create 2D arcade games that are built from common mechanisms, such that skills or learned features from one game may be applicable to others. The games are highly customizable by changing a common set of parameters, enabling a vast array of variations for any one game. We provide a set of single-task learner benchmarks on the games using PPO, and then show how taking advantage of Meta Arcade's features and curriculum learning enable successful learning on tasks where single-task learning failed. Future work formalizing this into pedagogy would be beneficial for problems where such configurability is possible. We also summarized initial results demonstrating successful policy transfer from sets of Meta Arcade games to other games in the suite. Future work could use Meta Arcade to characterize such transfer between tasks, explore when transfer is possible, or look to continual learning for sequences of tasks. While we focused on multitask learning, the application of meta-learning algorithms to Meta Arcade transfer experiments would also be of considerable interest.

Disclaimer

Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the Office of Naval Research. The views, opinions, and/or findings expressed are those of the author(s) and should not be interpreted as representing the official views or policies of the Department of Defense or the U.S. Government.

REFERENCES

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. *CoRR*, abs/1612.03801, 2016. URL http://arxiv.org/abs/1612.03801.

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.

Yoshua Bengio, Jérôme Louradour, Ronan Collobert, and Jason Weston. Curriculum learning. In *Proceedings of the 26th annual international conference on machine learning*, pp. 41–48, 2009.

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, Oleg Klimov, Christopher Hesse, Taehoon Kim, and John Schulman. Quantifying generalization in reinforcement learning. *CoRR*, abs/1812.02341, 2018. URL http://arxiv.org/abs/1812.02341.

Karl Cobbe, Christopher Hesse, Jacob Hilton, and John Schulman. Leveraging procedural generation to benchmark reinforcement learning. *CoRR*, abs/1912.01588, 2019. URL http://arxiv.org/abs/1912.01588.

Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. *CoRR*, abs/1703.03400, 2017. URL http://arxiv.org/abs/1703.03400.

Message P Forum. Mpi: A message-passing interface standard. Technical report, USA, 1994.

Timothy Hospedales, Antreas Antoniou, Paul Micaelli, and Amos Storkey. Meta-Learning in Neural Networks: A Survey. *arXiv e-prints*, art. arXiv:2004.05439, April 2020.

Arthur Juliani, Vincent-Pierre Berges, Esh Vckay, Yuan Gao, Hunter Henry, Marwan Mattar, and Danny Lange. Unity: A general platform for intelligent agents. *CoRR*, abs/1809.02627, 2018. URL http://arxiv.org/abs/1809.02627.

Michal Kempka, Marek Wydmuch, Grzegorz Runc, Jakub Toczek, and Wojciech Jaskowski. Vizdoom: A doom-based AI research platform for visual reinforcement learning. *CoRR*, abs/1605.02097, 2016. URL http://arxiv.org/abs/1605.02097.

Timothy P. Lillicrap, Jonathan J. Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. Continuous control with deep reinforcement learning. In Yoshua Bengio and Yann LeCun (eds.), *ICLR*, 2016. URL http://dblp.uni-trier.de/db/conf/iclr/iclr2016.html#LillicrapHPHETS15.

S. Luo, H. Kasaei, and L. Schomaker. Accelerating reinforcement learning for reaching using continuous curriculum learning. In *2020 International Joint Conference on Neural Networks (IJCNN)*, pp. 1–8, 2020. doi: 10.1109/IJCNN48605.2020.9207427.

Patrick MacAlpine and Peter Stone. Overlapping layered learning. *Artificial Intelligence*, 254:21 – 43, 2018. ISSN 0004-3702. doi: https://doi.org/10.1016/j.artint.2017.09.001. URL http://www.sciencedirect.com/science/article/pii/S0004370217301066.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin A. Riedmiller. Playing atari with deep reinforcement learning. *CoRR*, abs/1312.5602, 2013. URL http://arxiv.org/abs/1312.5602.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A. Rusu, Joel Veness, Marc G. Bellemare, Alex Graves, Martin Riedmiller, Andreas K. Fidjeland, Georg Ostrovski, Stig Petersen, Charles Beattie, Amir Sadik, Ioannis Antonoglou, Helen King, Dharshan Kumaran, Daan Wierstra, Shane Legg, and Demis Hassabis. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533, February 2015. ISSN 00280836. URL http://dx.doi.org/10.1038/nature14236.

Sanmit Narvekar, Bei Peng, Matteo Leonetti, Jivko Sinapov, Matthew E Taylor, and Peter Stone. Curriculum learning for reinforcement learning domains: A framework and survey. *Journal of Machine Learning Research*, 21(181):1–50, 2020.

Alex Nichol, Joshua Achiam, and John Schulman. On first-order meta-learning algorithms. *CoRR*, abs/1803.02999, 2018a. URL http://arxiv.org/abs/1803.02999.

Alex Nichol, Vicki Pfau, Christopher Hesse, Oleg Klimov, and John Schulman. Gotta learn fast: A new benchmark for generalization in RL. *CoRR*, abs/1804.03720, 2018b. URL http://arxiv.org/abs/1804.03720.

OpenAI, Ilge Akkaya, Marcin Andrychowicz, Maciek Chociej, Mateusz Litwin, Bob McGrew, Arthur Petron, Alex Paino, Matthias Plappert, Glenn Powell, Raphael Ribas, Jonas Schneider, Nikolas Tezak, Jerry Tworek, Peter Welinder, Lilian Weng, Qiming Yuan, Wojciech Zaremba, and Lei Zhang. Solving rubik's cube with a robot hand. *CoRR*, abs/1910.07113, 2019. URL http://arxiv.org/abs/1910.07113.

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

Kate Rakelly, Aurick Zhou, Deirdre Quillen, Chelsea Finn, and Sergey Levine. Efficient off-policy meta-reinforcement learning via probabilistic context variables. *CoRR*, abs/1903.08254, 2019. URL http://arxiv.org/abs/1903.08254.

Jurgen Schmidhuber. Evolutionary principles in self-referential learning. on learning now to learn: The meta-meta-meta...-hook. Diploma thesis, Technische Universitat Munchen, Germany, 14 May 1987. URL http://www.idsia.ch/~juergen/diploma.html.

John Schulman, Sergey Levine, Pieter Abbeel, Michael Jordan, and Philipp Moritz. Trust region policy optimization. In *International conference on machine learning*, pp. 1889–1897. PMLR, 2015.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

John Schulman, Philipp Moritz, Sergey Levine, Michael Jordan, and Pieter Abbeel. High-dimensional continuous control using generalized advantage estimation, 2018.

David Silver, Aja Huang, Chris J Maddison, Arthur Guez, Laurent Sifre, George Van Den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, et al. Mastering the game of go with deep neural networks and tree search. *nature*, 529(7587):484–489, 2016.

Federico Tesser. Distributed message passing with mpi4py. In *Euroscipy 2016*, 2016.

Joshua Tobin, Rachel Fong, Alex Ray, Jonas Schneider, Wojciech Zaremba, and Pieter Abbeel. Domain randomization for transferring deep neural networks from simulation to the real world. *CoRR*, abs/1703.06907, 2017. URL http://arxiv.org/abs/1703.06907.

Emanuel Todorov, Tom Erez, and Yuval Tassa. Mujoco: A physics engine for model-based control. In *IROS*, pp. 5026–5033. IEEE, 2012. ISBN 978-1-4673-1737-5. URL http://dblp.uni-trier.de/db/conf/iros/iros2012.html#TodorovET12.

Oriol Vinyals, Timo Ewalds, Sergey Bartunov, Petko Georgiev, Alexander Sasha Vezhnevets, Michelle Yeo, Alireza Makhzani, Heinrich Küttler, John P. Agapiou, Julian Schrittwieser, John Quan, Stephen Gaffney, Stig Petersen, Karen Simonyan, Tom Schaul, Hado van Hasselt, David Silver, Timothy P. Lillicrap, Kevin Calderone, Paul Keet, Anthony Brunasso, David Lawrence, Anders Ekermo, Jacob Repp, and Rodney Tsing. Starcraft II: A new challenge for reinforcement learning. *CoRR*, abs/1708.04782, 2017. URL http://arxiv.org/abs/1708.04782.

Jane X. Wang, Michael King, Nicolas Porcel, Zeb Kurth-Nelson, Tina Zhu, Charlie Deck, Peter Choy, Mary Cassin, Malcolm Reynolds, Francis Song, Gavin Buttimore, David P. Reichert, Neil Rabinowitz, Loic Matthey, Demis Hassabis, Alexander Lerchner, and Matthew Botvinick. Alchemy: A structured task distribution for meta-reinforcement learning, 2021.

Tianhe Yu, Deirdre Quillen, Zhanpeng He, Ryan Julian, Karol Hausman, Chelsea Finn, and Sergey Levine. Meta-world: A benchmark and evaluation for multi-task and meta reinforcement learning. *CoRR*, abs/1910.10897, 2019. URL `http://arxiv.org/abs/1910.10897`.

Chiyuan Zhang, Oriol Vinyals, Rémi Munos, and Samy Bengio. A study on overfitting in deep reinforcement learning. *CoRR*, abs/1804.06893, 2018. URL `http://arxiv.org/abs/1804.06893`.

Wenshuai Zhao, Jorge Peña Queralta, and Tomi Westerlund. Sim-to-real transfer in deep reinforcement learning for robotics: a survey, 2020.

A Appendix

A.1 PPO Implementation and Hyperparameters

In the benchmarking and curriculum experiments, the following hyperparameters were used for PPO:

Parameter	Value
Trajectory Length	128
Parallel Trajectories	8
Epochs	3
Clipping Parameter	0.1
Adam Learning Rate	0.00025
Entropy Coefficient	0.01
Value Coefficient	1.0
Maximum Gradient Norm	0.5
Gamma	0.99
Lambda	0.95

Table 4: PPO Hyperparameters

Parameters such as the clipping parameter and entropy coefficient were held fixed during training, and not annealed towards zero as in some implementations.

A.2 Curriculum Results

In Table 5, we show the learning curves and explanatory diagrams of the curricula described in Section 4.2.

A.3 Task Boundary Discussion

When the parameters of a task are randomized or modifiable, as in Meta Arcade, there is a gray area between varying the parameters of the game and introducing a separate task. The configurablility of the environment suite makes it easier to investigate this boundary. In our experiments, this was especially relevant to curriculum learning: while the task slowly changed over time, the start and end configurations often looked quite different from each other. Most curricula for games with the objective to cross the screen had a common starting point – the possibility of a blank play area with no obstacles. If this were defined as a separate task then these curricula could be considered multi-task problems with at least one task in common. Similarly, the curriculum for Invasion started with blocks that were collectable instead of harmful, in order to encourage positioning the paddle beneath the falling blocks. This mechanism is also found in the predefined game Avalanche. In general, it is useful to think of the predefined games as data points in the space of possible tasks, and it is up to future researchers to define metrics for similarity and uniqueness.

A.4 Predefined Games Characteristics

For reference, we provide a list of predefined games and which environment components they include (Table 6). This is especially useful for determining if any two (or more) games may be candidates for transfer learning.

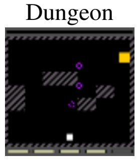
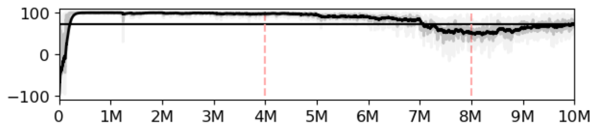
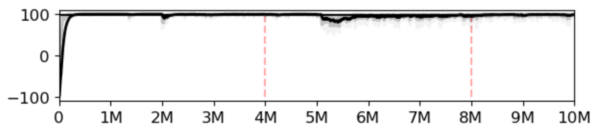
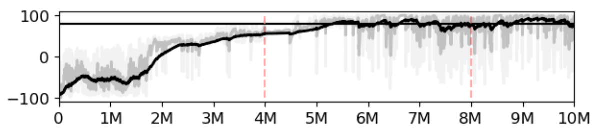
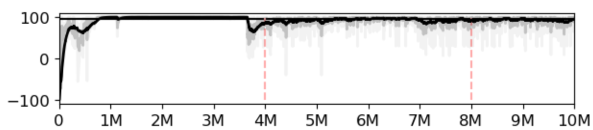
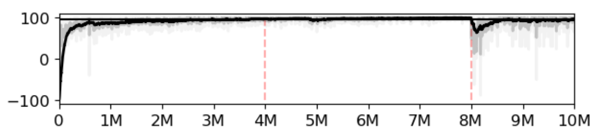
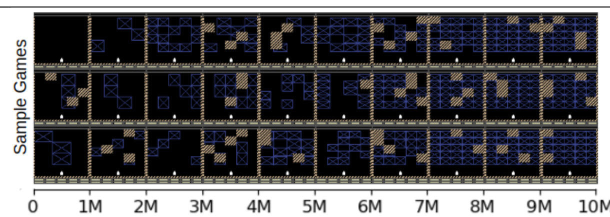
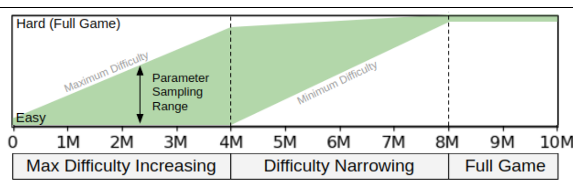
Game	Curriculum Training Curve	Final Score
Dungeon 		72.08 ± 17.51
Freeway 		98.67 ± 4.88
Invasion 		79.56 ± 53.86
Lava Maze 		95.52 ± 13.55
Tunneler 		95.95 ± 10.26
Tunneler Example Frames		-
Curriculum Sampling Strategy		-

Table 5: Successful curricula training on several games in which single-task PPO failed. **Second from bottom:** Example frames sampled from the Tunneler curriculum, showing variety on the y-axis and an increase in difficulty along the x-axis. **Bottom:** The general approach to curriculum parameter sampling, with a widening of the sampling envelope for 4M frames followed by a shrinking of the envelope to focus on the full game.

Game	Movement			Included Components						Game Objective				
(Red Denotes a Challenging Game)	Paddle	Full Movement	Shooting	Static Obstacles	Bounce Ball	Collectable Blocks	Hazardous Ball	Hazardous Blocks	Opponent	Get to the top of the screen	Destroy Blocks (Direct)	Destroy Blocks (Indirect)	Hit enemy with bullet	Get ball passed enemy
Avalanche	X					X					X			
Battle_Pong	X		X		X				X				X	X
Breakout	X				X	X						X		
Collect_Five		X				X					X			
Dodgeball_Duel	X		X				X		X				X	
Duel	X		X	X					X				X	
Dungeon		X		X		X	X				X			
Erosion	X		X					X	X				X	
Freeway		X						X		X				
Haunted_Hallway		X		X				X		X				
Hedge_Maze		X		X						X				
Interference	X				X			X	X					X
Invasion	X		X					X				X		
Juggling	X				X	X						X		
Keep_Ups	X				X	X						X		
Last_Block_On_The_Left	X				X	X						X		
Lava_Maze		X						X		X				
Pong	X				X				X					X
Pong_Breakout	X				X	X			X					X
Seek_Destroy		X	X	X					X				X	
Shootout	X		X			X						X		
Sweeper		X				X					X			
Target_Practice	X		X			X						X		
Tunneler		X	X	X				X		X				

Table 6: Summary of skills and elements common to the set of predefined games.

A.5 Example Game JSON Definition


```
{
    "meta":{
        "description":"Catch 50 falling blocks in a row."
    },
    "actions":{
        "up":false,
        "down":false,
        "left":true,
        "right":true,
        "fire":false
    },
    "game_elements":{
        "top_wall":false,
        "bottom_wall":false,
        "ball":false,
        "opponent":false,
        "blocks":true,
        "static_barriers":false
    },
    "display_settings":{
        "background_color":[0,0,0],
        "ui_color":[80,80,80],
        "indicator_color_1":[200,200,160],
        "indicator_color_2":[0,0,0]
    },
    "player_settings":{
        "width":0.15,
        "height":0.05,
        "speed":0.012,
        "color":[255,255,255],
        "steering":0.5
    },
    "opponent_settings":{},
    "ball_settings":{},
    "blocks_settings":{
        "creation_area":[0.05,-1.0,0.9,1.0],
        "rows":6,
        "cols":6,
        "per_row":1,
        "spacing":0.4,
        "color":[162, 219, 252],
        "static_weave_fall":"fall",
        "speed":0.003,
        "harmful":false,
        "points":2
    },
    "static_barrier_settings":{
        "color":[38, 101, 209]
    },
    "image_settings":{
        "color_inversion":false,
        "rotation":0,
        "hue_shift":0.0,
        "saturation_shift":0.0,
        "value_shift":0.0
    }
}
```


Figure 5: Example JSON of the game *Avalanche*.