Group 30 - Project 56
Solving challenging video games in human-like ways

Adeet Patel
Center for Data Science
New York University
ajp756@nyu.edu

Sarvesh Patki
Center for Data Science
New York University
ssp6603@nyu.edu

Brian Pennisi
Center for Data Science
New York University
bp2221@nyu.edu

Jiawen Wu
Center for Data Science
New York University
jw1562@nyu.edu

Abstract

As progress in machine learning advances, it is natural to compare an artificial intelligence agent's ability to adapt to new environments and "learn" to that of a human. The game "Baba is You" is a natural environment to experiment and test the performance of AI agents vs humans because of its unique feature of manipulating the rules of the game. We use deep reinforcement learning and curriculum learning to train an AI agent on the game "Baba is You". We provide evidence that curriculum learning is advantageous not just for humans, but for artificial agents as well. We also find that using a dual encoder architecture and dynamically adapting entropy ("entropy trick") are keys to improving performance over standard PPO-based actor-critic techniques.

1 Introduction

An often-cited goal of machine learning is to create an artificial general intelligence - one that has the ability to mimic human understanding by "learning" through experience and feedback to adapt to different environments. In recent years, reinforcement learning has emerged as a powerful technique towards this goal as a method for training agents to solve a wide range of tasks, from playing games to controlling robots. In this paper, we explore the use of reinforcement learning to train an agent to play the game "Baba is You." We describe the game and the reinforcement learning algorithm we used to train the agent, and present the results of our experiments. Our results show that reinforcement learning can be used to successfully train an agent to play this challenging game, achieving performance that rivals that of human players. They also show that standard models can be improved upon to handle the idiosyncracies of the game. This work has potential applications in a variety of domains, including game design and artificial intelligence.

"Baba is You" is a puzzle video game with the unique feature that the player agent is able to manipulate the "rules" of the game dynamically during playtime, represented in the play area by movable word blocks, in order to allow the player character, usually the titular "Baba", to reach a specified goal. Reinforcement learning is a type of machine learning training method developed by Sutton [1984] where an agent learns to interact with its environment for addressing sequential decision tasks in which the agent has only limited environmental feedback. Desired behaviors or actions are rewarded and/or undesired ones are penalized. Narvekar et al. [2020] developed curriculum learning as as a type of transfer learning to use in reinforcement learning to accelerate or improve training by incrementally exposing the agent to a pre-specified set of tasks or data samples prepared in a particular sequence.

By using a curriculum learning-based approach to train a deep reinforcement learning network, we attempt to answer the question: How do curricula improve deep reinforcement learning for the game “Baba is You”, and what does that teach us about the differences in problem-solving between humans and machines? Since reinforcement learning aids in decision making by teaching the agent to learn optimal behavior in an environment in order to maximize rewards, our hypothesis was that it could be applied to problem solving in the context of video games and puzzles, among many other applications.

Curriculum learning has been a foundational component on top of which we built our research, and we derived inspiration from related work conducted in this domain. Since the research question involved the study of differences in problem-solving between humans and machines, a curriculum learning based approach was apt. When children learn to solve problems, their instructors break the problem into smaller parts and teach them to solve the simpler, more fundamental tasks first before gradually moving to harder ones. For example, humans learn to add, subtract, multiply and divide two numbers individually before learning to solve complex algebraic equations. In this project, we replicate this approach by teaching the agent (which we named Baba, after the character in the game) to first learn the fundamental and easy tasks and gradually move on to harder tasks. The details have been discussed in the subsequent sections. A number of experiments were conducted and different model architectures were used to teach Baba each of these sub-tasks. We tried various combinations of a single and dual encoder architecture, curriculum and no-curriculum based learning, and tweaking various parameters such as entropy.

The experiments showed that using a curriculum learning based approach with a dual encoder architecture along with the “entropy trick” gave the best results. Moreover, this design shares similarities with the way humans learn. Still, there are levels that humans can beat that our models cannot. We speculate that humans are able to learn faster due to adapting prior knowledge from the same or other domains to new situations. The experiments and the results are discussed in greater detail in the subsequent sections. As a secondary takeaway, we share our findings on how model training for “Baba is You” is particularly sensitive to the experiment design and hyperparameter tuning. This may be useful since the game’s rewards are sparse and difficult to achieve through random exploration alone.

2 Related work

Deep reinforcement learning incorporates techniques and architectures from deep learning neural network models, such as CNNs, developed by LeCun et al. [2015], Schmidhuber [2015], Goodfellow et al. [2016], and others into reinforcement learning frameworks to encode the game state as an input to determine the next best action as a prediction. In the past, significant research has been conducted on using various reinforcement learning methods, such as Q-learning, to train agents that can solve games. Mnih et al. [2013] found success in using deep reinforcement learning methods to teach an AI agent to beat Atari games. Similarly, Silver et al. [2016] used the deep reinforcement learning techniques of value and policy networks to train an AI agent to win the game Go. The game "Baba is You" is unique in that the "rules" of the game are not fixed, so we want to see if an artificial agent can learn to manipulate them to its advantage.

Mnih et al. [2013] first introduced the idea of using convolutional neural networks (CNNs) with linear layers to predict the best action for an agent to take given a game state and train an agent to complete a variety of tasks based on different levels of Atari games, which have a 2D decision space with discrete action spaces. Although similar in some aspects, “Baba is You” is more difficult than the Atari games due to both the mutable nature of objects in the level (based on the arrangement of the "word" objects) and the sparsity in the rewards. We therefore attempt to enhance the model architecture design by incorporating several other techniques. First, we use proximal policy optimization (PPO) which was developed by Schulman et al. [2017] as the objective function for our action agent which uses a clipping function to constrain the size of the policy updates to ensure improvements in policy. Second, we use an actor-critic architecture, first introduced by Konda and Tsitsiklis [1999], which moves the Q-learning task to a network called the critic, helping to produce more useful rewards as input to the PPO-based actor network. Both of these modifications align with more modern RL methods and are discussed further the "Algorithms" section. We use curriculum learning as developed by Narvekar et al. [2020] to understand how pre-training on simple tasks helps with more complicated ones, which can be especially useful when working with sparse rewards. Finally, many classic hyperparameters

Table 1: Optimal hyperparameters

Hyperparameter	Value
num-parallel-envs	64
batch-size	4096
learning-rate	1.25e-4
value-loss-clipping	False
entropy-trick	True
epochs-per-batch	4
mini-batch-size	4

such as batch size and learning rate were importantly tuned to our domain and hardware specifications, as detailed in Table 1.

3 Problem definition and algorithm

3.1 Task

In order to study how deep learning techniques and curriculum learning improves the performance of an artificial agent on the game "Baba is You", we need to experiment with various permutations of both model architectures and sequence of curricula. By analyzing how well each works on different levels of the game, we can draw ideas about how "machines" learn.

At the atomic level, our model (described in more detail in the "Algorithm" section) input was a "game state" which was a point in time summary of all of the objects in the level, including Baba, and their locations. The output was the action that Baba should take, which was any of "Up", "Down", "Left", or "Right". Given the games state and the actions that Baba could take, we experimented with the size and structure of our neural networks, the curriculum which was taught to the artificial agent, and evaluation function (i.e. type of loss, level of entropy, etc) to see how well Baba would learn and perform in each successive simulation of the game.

We were ultimately able to test our agent on two main levels shown in Figure 1 - "finding the flag" which is a square grid consisting of just Baba and the flag and "walking through walls" which incorporated obstacles and the ability to change the rules and properties of the level. The baseline case is simply allow the artificial agent to try to beat these levels without any prior training or curricula, but simply by trying multiple simulations on the level as is. For our experiment, we studied how good various training strategies were at beating the main levels of the game. We constructed a number of easier "lessons" as part of the curricula to "teach" the agent certain skills, such as the importance of the flag or that certain properties can be changed, and to see if these skills would transfer to the main levels. The artificial agent would then be trained first using the curriculum before attempting the main level. We also experimented with different model architectures. We found that the dual encoder structure for the actor and critic allowed the agent to learn more effectively compared to the single encoder structure for both the actor and the critic. This is because it removes the conflict between the policy and value function during training. We also experimented with different levels of entropy, and observed that lower levels of entropy allowed the agent to carry over skills learned in previous levels, but made it more rigid and difficult to adapt to new environments. On the other hand, higher levels of entropy allowed the agent to "unlearn" skills that it had previous learned and explore the environment more.

3.2 Algorithm

Our deep reinforcement learning model uses an actor-critic framework. The actor looks at Baba's current location (state) in the level and determines the next move to make out of the four possible options: up, down, left, or right. The critic estimates the Q-value function, which, given an observation (state), produces an estimate of how good it is. For example, states closer to the flag (reward) are generally considered to be of higher value. This dual actor-critic network architecture allows the model to have some sense for which actions are better than others. For example, consider an agent which makes an optimal trip to the flag by taking a direct path with no redundant steps, but then,

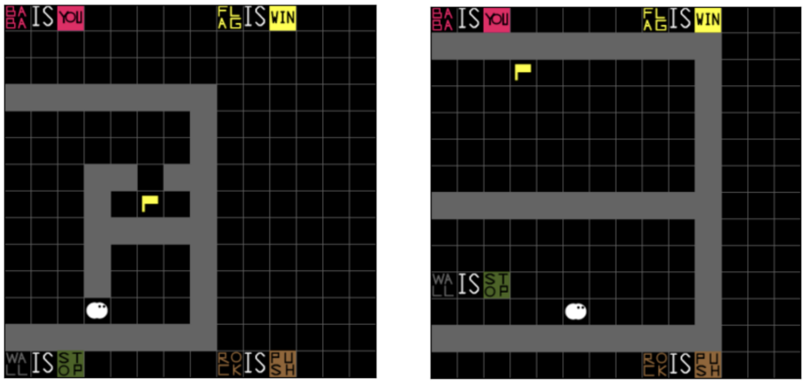

Figure 1: find-the-flag.level (left) and walk-through-walls.level (right). Baba (white) is shown towards of bottom of each 14 by 14 grid. For both levels, the goal is to occupy to same space as ("capture") the flag. For the first level, Baba needs to navigate around the maze of walls (grey.) On the second level, Baba needs to manipulate the rules by moving the wall word block (green.) This will allow it to walk through walls and capture the flag.

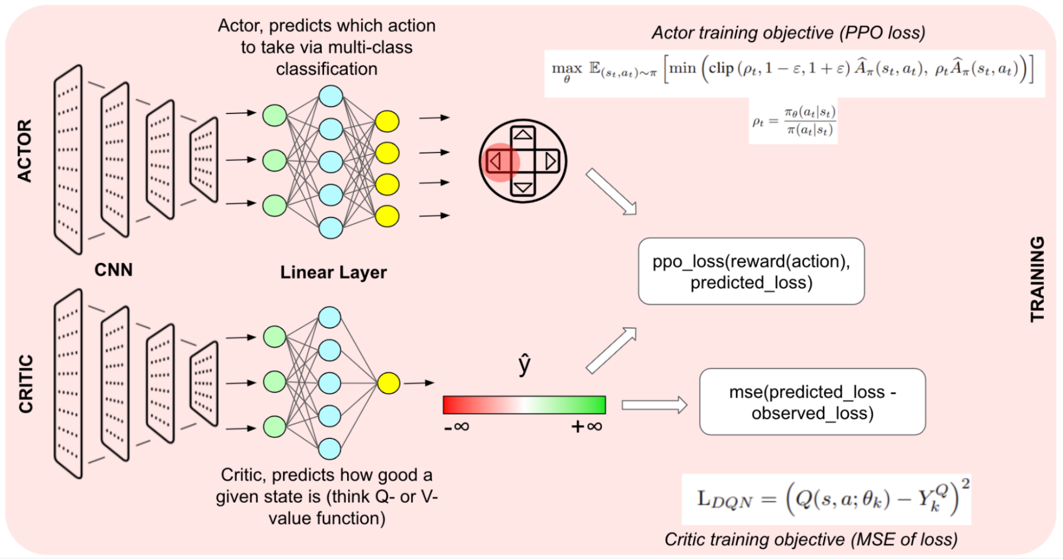
Actor, predicts which action to take via multi-class classification
ACTOR
CNN
Linear Layer
Actor training objective (PPO loss)
$\max_{\theta} \mathbb{E}_{(s_t,a_t)\sim\pi}\left[\min\left(\text{clip}(\rho_t, 1-\varepsilon, 1+\varepsilon)\,\widehat{A}_\pi(s_t,a_t),\ \rho_t\widehat{A}_\pi(s_t,a_t)\right)\right]$
$\rho_t = \frac{\pi_\theta(a_t|s_t)}{\pi(a_t|s_t)}$
ppo_loss(reward(action), predicted_loss)
TRAINING
CRITIC
ŷ
-∞
+∞
mse(predicted_loss - observed_loss)
Critic, predicts how good a given state is (think Q- or V-value function)
$L_{DQN} = \left(Q(s,a;\theta_k) - Y_k^Q\right)^2$
Critic training objective (MSE of loss)

Figure 2: Model architecture, training graph, and loss functions (PPO, MSE)

later in the game, it reaches the flag after only taking superfluous steps around it. The actor-critic framework can reward optimal paths of reaching the flag in an efficient manner, while still providing some reward for reaching the flag even if the path taken was less efficient. Both the actor and the critic are networks with a convolutional neural network (CNN) encoder plus linear layers that work together to help Baba learn from the environment.

Figure 2 and 3 illustrate our model architecture for training and inference respectively. In training, the actor network uses proximal policy optimization (PPO) to learn the best policy for the artificial agent Baba to follow in the game. Trust region policy optimization (TRPO) as described by Schulman et al. [2015] is a classic objective function in RL and is designed to update model parameters in a way that doesn't change the distribution too dramatically each time. PPO attempts to enforce the same behavior in a way that reduces the computational complexity and allows for multiple data samples per parameter update. In order to properly contextualize the "goodness" of a given action, it requires input ("predicted loss") from the critic network. The critic network's objective is the mean squared error of the current state. The overall loss function is shown below and includes entropy which is

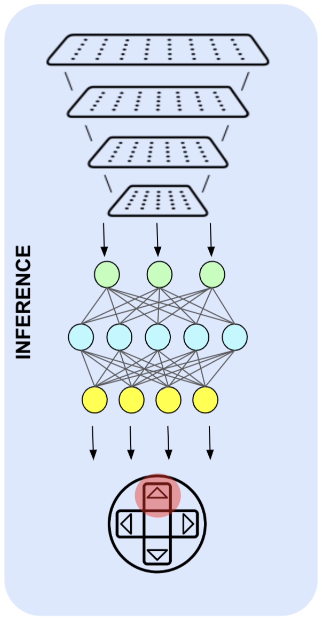

Figure 3: Model inference, this is the Actor network from Figure 2

designed to help encourage exploration, especially early on. Note β_1, β_2, and β_3 are the configurable coefficient weights for PPO, MSE, and entropy respectively.

$$\ell_{model} = \beta_1 * \ell_{PPO} + \beta_2 * \ell_{MSE} - \beta_3 * \varepsilon_{entropy}$$

As part of our curriculum learning experiments, we want to train the model to learn incremental skills. However, the model tends to overfit to the earlier skills that it learns such that it would be difficult to learn newer skills later on. For example, we would first teach Baba that it can earn a reward by navigating the game space to co-locate itself with the flag. When trying to teach it how to beat a level that required this co-location skill as well as learning how to walk through walls, Baba would attempt to reach the flag while banging into the impeding wall. The built-in exploration mechanism was not effective because the agent at this point was very confident that it needed to take the shortest path to the flag. In order to fix this problem, we needed a way to incentivize exploration despite the model's confidence and "unlearn" what it had learned on previous levels. We used what we call the "entropy trick", which is an algorithm that doubles the value of the entropy coefficient in the loss function. This is applied if our agent achieves 0 return, when it does not reach the flag within a certain number of steps, for at least three consecutive gradient updates. This prevented Baba from becoming too confident or rigid and encourages the AI agent to be curious and explore more, allowing it to learn in new environments instead of simply exploiting what it already knows. Once Baba learned to explore sufficiently and earned a reward, we scaled this coefficient back down. Section 4 shows an example of how this works in practice.

Another important consideration was the network architecture. Originally, the CNN encoder was shared by the actor and critic. However, in our experiments, we found using separate (dual) CNNs encoders to be more useful since it removed the tendency for the model to overfit to the value function (critic) early on. Similar observations have been noted in experiments on the famous RL archetype "CartPole" Huang et al. [2022a].

4 Experimental evaluation

4.1 Data

For many reinforcement learning problems, each batch of data is one or more playthroughs (episodes) of the game. Unlike other supervised machine learning tasks such as classification, labeled datasets

Algorithm 1 Actor-critic framework for training artificial agent

```
initialize tensor for storing the game's next state
for each update do
    for each step do
        run actor network
        run critic network
        take an action based on network outputs
    end for
    update the gradients for both the actor and critic networks
    if Baba does not learn for 3 consecutive iterations then
        apply the "entropy trick"
    end if
end for
```

of images or texts for gameplay do not exist. Rather, the data has to be generated via playthrough simulations, where the data is generated based on the sequence of steps taken by the agent and can change throughout training. The data generated are based on the actions taken by the model, which changes throughout training as the parameters are updated and the agent explores different parts of the level.

The input to the model is the game state at some point in time, which completely describes the level and the location of all the objects, including the agent. The model outputs an action based on this input and the game simulator returns the next state and a reward, if any. The target variable that we want to maximize is the episodic return, which is a time-discounted reward and depends on the objective of the game. For the purpose of our experiments, the agent earns a reward only when being colocated with ("capturing") the flag. To simulate game play, we used OpenAI Gym environment (v0.23) Brockman et al. [2016], which is a standard API for reinforcement learning and is compatible with Python programming language.

For model set up, data generation, and training, we adapt a script from CleanRL Huang et al. [2022b]. To simulate "Baba is You", we use a custom implementation created by our mentors which uses OpenAI gym interface. The CleanRL script by default uses a preprocessed version of raw pixels as input to the model. Since our "Baba is You" implementation contains a full representation of the objects and their location, we created a dense representation of the game which used a binary encoding of the game grid. In order to allow the model to learn how to interact with different objects in different ways, each of the 29 objects had its own channel. All levels were designed in a 14 by 14 grid.

When generating our data, we randomly initialized the Baba agent and the flag in each game. This forced Baba to learn the importance of the flag and prevented it from merely memorizing moves required to beat a level. Throughout the paper, we refer to this as data augmentation. Given the lack of initial training data, the ability to simulate the game is paramount to generating training data. To accelerate the data generation and training, we use k asynchronous vectorized environments that play multiple games in parallel using multiple CPUs, where k controls our batch size.

4.2 Methodology

Our primary goal was to understand how useful curriculum learning is in teaching deep reinforcement learning agents to beat the game "Baba is You". Our hypothesis was that curriculum learning would allow the agent to be able to learn complex levels more effectively.

In order to evaluate whether curriculum learning is a useful approach, we created new levels which requires using simple skills in order to win. The skills required were: 1) learning to occupy the same space as the flag and 2) how to walk through walls. We then ran experiments with two different agents: one where the agent had been pretrained with a curriculum and another where the agent was initialized from scratch (no curriculum.) The curriculum agent was trained on a sequence of simple levels in order to learn the aforementioned skills. We collected the average episodic return and how long it took for the agent to “beat” the level.

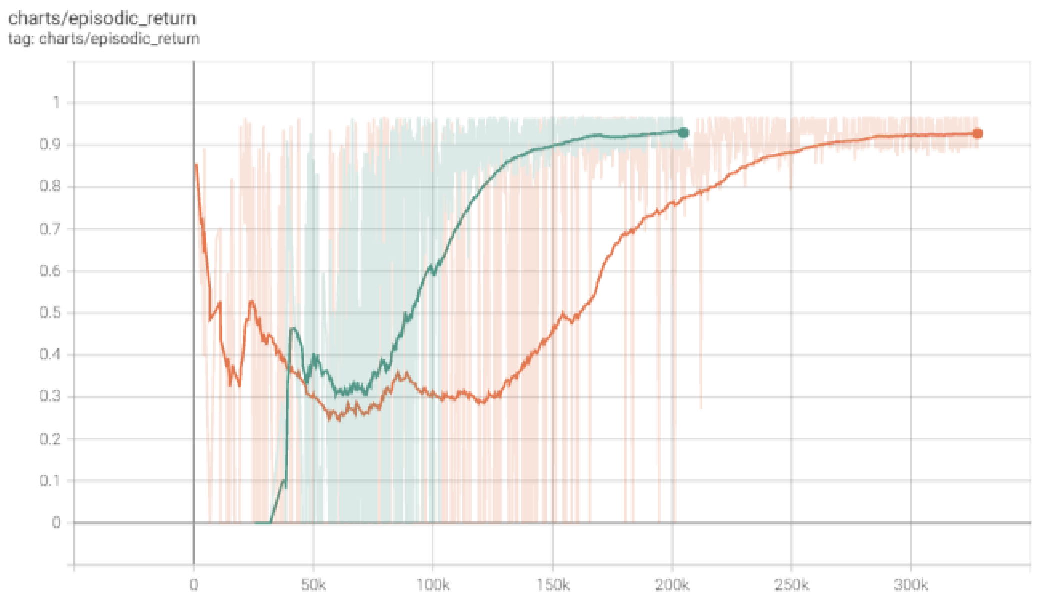

Figure 4: Performance on find-the-flag.level. The curriculum model (green) gets "stuck" at 0 return early on because it is using the knowledge from the curriculum up until that point on a level where it needs to learn new skills. However once it unlearns enough and start to earn a reward, it learns the skills necessary to beat the level much faster than the no curriculum model (orange.)

As a secondary goal, we experimented with different architectures and configurations to understand which implementation details were unique to the sparse reward game that is "Baba is You". The levels discussed in our "Results" section below were not possible without several improvements to default architectures recommended by the CleanRL paper, as discussed in the "Algorithms" and "Related Work" section.

Two key metrics were used. The first is "episodic return", which quantifies how well Baba is performing per episode, or one gameplay simulation. Achieving a sustained average episodic return above some threshold determines whether we beat a level ("Beat Level" column in table.) The second metric is the number of "steps", which is the number of sequential training examples required to beat a level and is also a measure of time. This is primarily used to understand how many simulations of the game are necessary to achieve sufficient performance and compare the rate at which a particular model architecture and curriculum learns. If our curriculum model is able to learn how to beat the level more at a faster rate than the no curriculum model or is able to consistently achieve a higher average episodic return, this would suggest that our approach has value.

While episodic return, or extrinsic reward, is a common metric used to compare experiments in reinforcement learning, it is not comparable across games. We are not aware of existing published research which uses reinforcement learning to play "Baba is You" to use as a benchmark.

4.3 Results

Figure 4 compares the episodic return of a curriculum model (green line) to a no-curriculum model (orange line) trained on a level which requires Baba to have mastered finding the flag. The level, find-the-flag.level, can be found in Figure 1. While both models beat the level (achieve an average return above 0.9), the curriculum is able to learn in about 150k steps of the level compared to 260k for the no-curriculum model. Unlike the no-curriculum model, the curriculum model gets "stuck" at 0 for about 40k steps so it requires the "entropy trick" to unlearn some from the previous levels. Note that in this chart, we do not depict steps that the curriculum model trained on prior to this level in order to provide a comparison of the training required for both models to succeed in this level.

Figure 5 compares the episodic return of a curriculum model (pink, row 6 in table) to a no-curriculum model (blue, row 5 in table) trained on the difficult walk-through-walls.level, also found in Figure 1. Unlike the previous chart, we show global step counts on the x-axis. Since the curriculum spent its

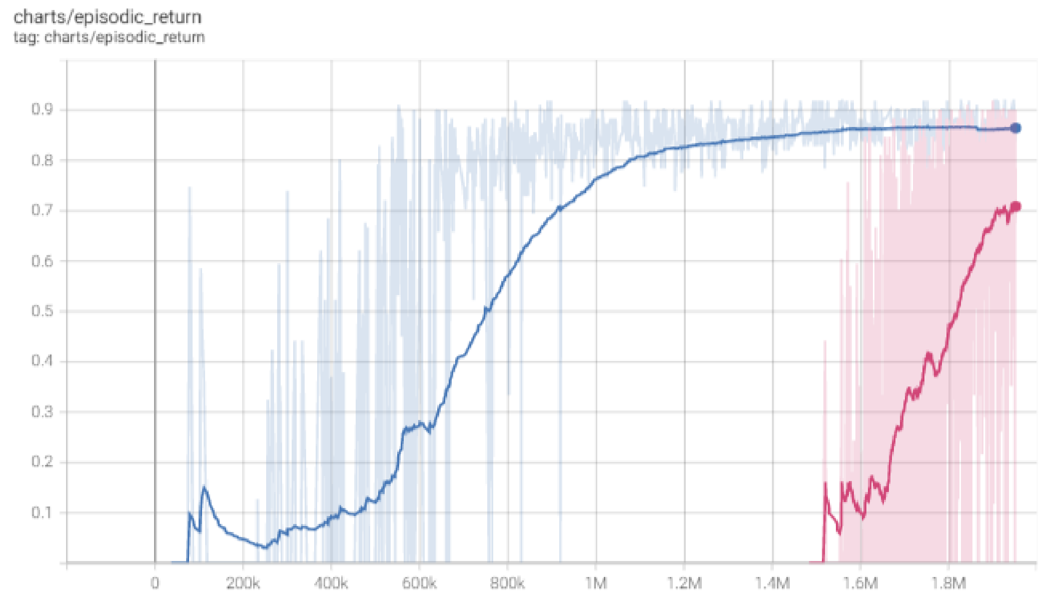

Figure 5: Performance on walk-through-walls.level. Here the curriculum model (pink) learns to beat the level very quickly. However, it was trained on 1.45 million steps of gameplay before attempting the level. Compared to the no curriculum model (blue) which attempts this level for the entirely of its steps, it learns faster but has a lower episodic return.

Table 2: Training results for walk-through-walls.level

Sr. No	Strategy	Steps to beat level (k)	Total steps (k)	Success?
1	No curriculum (baseline)	500.0	500.0	No
2	Curriculum	500.0	1950.0	No
3	No curriculum + dual encoder	718.4	718.4	Yes
4	Curriculum + dual encoder	500.0	1950.0	No
5	No curriculum + dual encoder + entropy trick	718.4	718.4	Yes
6	Curriculum + dual encoder + entropy trick	427.3	1877.3	Yes

first 1.45m steps learning other levels, we show no return during that period. The curriculum model was first exposed to this level at 1,450k steps whereas the no curriculum model began at 0 steps. Both models achieve the goal of beating the level, which involved exceeding an average episodic return of at least 0.7. The curriculum model has a steeper slope, which illustrates how its curriculum allows it to learn the level more quickly. The no-curriculum model requires about 300k additional steps of the level before reaching the return threshold. However, because it spends around 4x as much time training on the level, the no-curriculum model achieves a higher overall return.

4.4 Discussion

Table 2 shows a summary of the different strategies used to train Baba on the more difficult walk-through-walls.level, the results for each, the number of training steps (level-only and global) required, whether Baba learned to beat the level (based on sustained episodic return over some threshold.) When comparing the results from adding dual encoders to strategies that did not, we see that the use of a dual encoder architecture contributes to allowing the agent to beat a level.

When incorporating the "entropy trick" in addition to curriculum and dual encoders, we see the “entropy trick” is critical to ensuring our curriculum model is able to transfer its knowledge from past levels (the curriculum) to a new level, since our models are sensitive to overfitting and thus need a means of “unlearning” some information from the past. The difference in including a curriculum or not in the final model with dual encoders and the "entropy trick" (Sr. No 6 & 5 respectively) is the rate at which the model learns. We see that a curriculum-based model learns faster than a no-curriculum model, although the latter achieves a higher episodic return since it spends more time learning.

5 Conclusions

We can conclude from the results of our experiments that curriculum learning helps the model learn to beat our difficult walk-through-walls.level level faster than no curriculum. However, curriculum learning requires more total steps for training. Moreover, using a dual encoder architecture for the actor-critic pair reduces the number of timesteps by a factor of almost 50% and is critical in allowing our agent to beat this level. This, as well as the efficacy of the "entropy trick" for curriculum models highlights how the success of our RL algorithm was highly dependent on our model architecture, hyperparameters, and experiment set up. The trade-off between exploration and exploitation is always very important in reinforcement learning, but it was especially important in this case. Throughout our experiments, we found that incentivizing the agent to explore the environment (by dynamically changing the entropy penalty) was crucial in helping Baba to adapt to new environments and succeed at more difficult levels. We found that this "entropy trick" was a significant requirement for curriculum models when new skills were introduced.

Based on our experiments with an AI agent, we note that machines can benefit from curriculum learning which is used similarly to train humans. Further, we speculate that while human and machine learning may both be improved using these curricula, teenage to adult humans are able to learn faster due to prior knowledge in adapting to new situations. One perspective is that adult humans are akin to complex pretrained models which encode a significant amount of information about how to play video games based on life experience. In this lens, we can draw further similarities to the way machines and humans learn.

While "Baba is You" is a unique game with sparse rewards and complex levels, we believe that our experimentation in training an agent to learn how to play this game can be applied to many other applications towards other challenging games. puzzles, or even other domains. This research is the only known set of experiments to apply these deep learning techniques to "Baba is You" specifically. We hypothesize that many of the modeling techniques such as random initialization of objects for data augmentation, the entropy trick, and our hyperparameter selection could be useful in similarly challenging games.

5.1 Future Work

There are a number of additional experiments which would be prudent to explore in order to further understand how machines learn. Perhaps our biggest shortcoming was getting the agent to effectively explore on difficult levels. Difficult levels require a highly specific sequence of steps and were therefore extremely unlikely for our agent to randomly execute. It is possible to beat these levels with heavily engineered curricula, but this becomes tightly coupled to the specific task or level and is limited to the ability of the human teacher. Methods which incentivize the agent to explore more effectively Burda et al. [2018], Parisi et al. [2021] could allow for learning to outpace that of a human teacher.

Other interesting topics include looking at how our dense encoding compares to the less efficient pixel encoding, which is the same input humans use when playing the level. At some point, as the level get more difficult, it may be useful to see whether there are larger neural networks (1M, 10M parameters) able to beat challenging levels which our current model (0.5M) cannot. Furthermore, all experiments were run in levels contained in a 14 by 14 grid and CNNs do not naturally lend themselves to their input size varying over time. However, it would be interesting to see if we come up with or learn a transfer function to allow our network to take game states of varying sizes as network input.

6 Lessons learned

There were a number of lessons learned in this project. We learned first-hand how fickle reinforcement learning experiments can be, especially using the popular PPO objective function. We spent the better part of a month running experiments in which our agent was not able to successfully learn to beat easier versions of find-the-flag.level. After discussing this issue with our mentors in various stages, we were ultimately able to use data augmentation to solve this problem. Getting our agent to beat more difficult levels involved many hours of trial and error. Our model would fail to beat a level despite a myriad of hyperparameter changes. Then, after discussing with our mentors and doing a review of relevant literature, we would come up with some new ideas. One example of this was that our curriculum models, having been trained to do specific tasks, were overfitting on unseen levels. Our mentors helped create a short list of potential solutions. In this case, we discovered technical documentation on the nuances of PPO Huang et al. [2022a] which allowed us to develop an understanding of how our particular implementation of the loss function used entropy to incentivize exploration. Once we understood this, we were able to add custom logic to the loss function to allow the model to unlearn enough to start generating rewards for the new level (the “entropy trick”).

Another issue that we found which is common in reinforcement learning is the sparsity of the reward signals made it difficult for the artificial agent to learn initially. Going forward, it would be beneficial to engineer some intermediate rewards (e.g., for getting closer to the goal) to increase the rate of learning. We also saw the benefits of communicating to outside stakeholders using visual examples, such as replays of our agent attempting the level of demoing different levels to showcase areas where our agent struggled, to more efficiently convey our talking points. Actively working on the feedback provided by our mentors and by redirecting our approach we were able to effectively overcome some of the major roadblocks. They also helped bring to light the importance of curiosity objectives, discussed in "Future Work", to combat this issue.

While a lot of the speculation around what our experiments tell us about how humans and machine learn is left to future work, we believe our experiments were effective in overcoming some of the idiosyncrasies of “Baba is You”. Given that the game had not been attempted with reinforcement learning before, we offer useful experiments and code which can guide a number of future avenues of work on the game and implications for human cognition.

One key takeaway for future projects would be that sometimes to get past an obstacle, it is necessary to step back, see the entire picture rather than focusing only on certain intricacies, and then redirect the approach to find alternative methods to achieve the objective. Moreover, insights from domain experts are often helpful and can open up new perspectives.

Acknowledgments and Disclosure of Funding

We would like to thank Professor Todd Gureckis, Mark Ho, Emily Liquin, and Pat Little for their advice and supervision on this project.

References

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

Yuri Burda, Harrison Edwards, Deepak Pathak, Amos J. Storkey, Trevor Darrell, and Alexei A. Efros. Large-scale study of curiosity-driven learning. *CoRR*, abs/1808.04355, 2018. URL `http://arxiv.org/abs/1808.04355`.

Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep learning*. MIT press, 2016.

Shengyi Huang, Rousslan Fernand Julien Dossa, Antonin Raffin, Anssi Kanervisto, and Weixun Wang. The 37 implementation details of proximal policy optimization. In *ICLR Blog Track*, 2022a. URL `https://iclr-blog-track.github.io/2022/03/25/ppo-implementation-details/`. https://iclr-blog-track.github.io/2022/03/25/ppo-implementation-details/.

Shengyi Huang, Rousslan Fernand Julien Dossa, Chang Ye, Jeff Braga, Dipam Chakraborty, Kinal Mehta, and João G.M. Araújo. Cleanrl: High-quality single-file implementations of deep reinforcement learning algorithms. *Journal of Machine Learning Research*, 23(274):1–18, 2022b. URL `http://jmlr.org/papers/v23/21-1342.html`.

Vijay Konda and John Tsitsiklis. Actor-critic algorithms. *Advances in neural information processing systems*, 12, 1999.

Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *nature*, 521(7553):436–444, 2015.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*, 2013.

Sanmit Narvekar, Bei Peng, Matteo Leonetti, Jivko Sinapov, Matthew E Taylor, and Peter Stone. Curriculum learning for reinforcement learning domains: A framework and survey. *arXiv preprint arXiv:2003.04960*, 2020.

Simone Parisi, Victoria Dean, Deepak Pathak, and Abhinav Gupta. Interesting object, curious agent: Learning task-agnostic exploration. *CoRR*, abs/2111.13119, 2021. URL `https://arxiv.org/abs/2111.13119`.

Jürgen Schmidhuber. Deep learning in neural networks: An overview. *Neural networks*, 61:85–117, 2015.

John Schulman, Sergey Levine, Pieter Abbeel, Michael Jordan, and Philipp Moritz. Trust region policy optimization. In *International conference on machine learning*, pages 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.

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.

Richard Stuart Sutton. *Temporal credit assignment in reinforcement learning*. University of Massachusetts Amherst, 1984.

Student contributions

Brian, Jiawen, Adeet and Sarvesh surveyed the academic literature on deep reinforcement learning and reinforcement learning applied to games, especially with curriculum learning. Brian and Jiawen adapted and wrote code to adapt OpenAI Gym for "Baba is You" and for CleanRL to optimize experimentation. Brian, Adeet, and Sarvesh set up the environment on the HPC clusters to run experiments. Brian, Jiawen, and Adeet ran experiments to train the AI agent. Brian, Jiawen, Adeet and Sarvesh worked on creating and formatting the presentations, report, and poster.