Solving challenging video games in human-like ways

NYU Center for Data Science

Brian Pennisi
Jiawen Wu
Adeet Patel
Sarvesh Patki

Background & Motivation

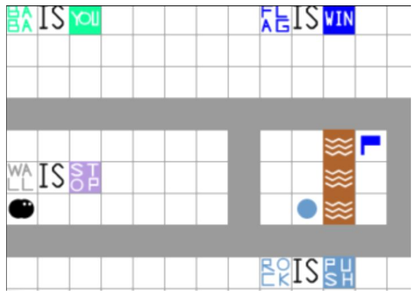
- **"Baba is You"** is a puzzle video game which centers around the manipulation of "rules"—represented in the play area by movable objects with words written on them—in order to allow the player character, usually the titular "Baba", to reach a specified goal.
- **Deep Reinforcement Learning** is a machine learning training method based on rewarding desired behaviors and/or punishing undesired ones for addressing sequential decision tasks in which the agent has only limited environmental feedback, while incorporating neural network architectures (ie. CNN) to encode the game state (input) and determine the next best action (prediction).
- **Curriculum Learning** is a type of transfer learning. In reinforcement learning, it is used to accelerate or improve learning by incrementally exposing the agent to a set of experiences over time on 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?***

Methodology

- For many reinforcement learning problems, each batch of data observations is one or more playthroughs (episodes) of the game. Unlike other supervised machine learning tasks such as classification, labelled datasets of images or texts for gameplay do not exist. Rather, the data has to be generated via multiple playthrough simulations, where the data is generated based on the sequence of steps taken by the agent and can change throughout training. The target variable that we want to maximize is the **episodic return**, which depends on the objective of the game.
- 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 **vectorized environments** that asynchronously play multiple games.
- To ensure our training data has sufficient variance, we use **data augmentation** techniques by randomly initializing the Baba agent and the flag goal in each game. This helps teach Baba the importance of the flag and prevent it from merely memorizing moves required to beat a level.

Key Experiments

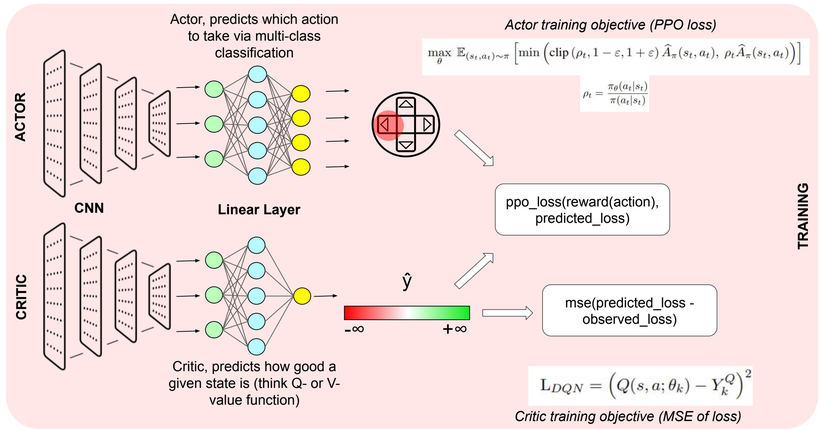
- **Experiment 1: Teaching Baba different skills**
  - Skills include flag is important, how to walk through walls, etc. (this is what we mean by curriculum!)

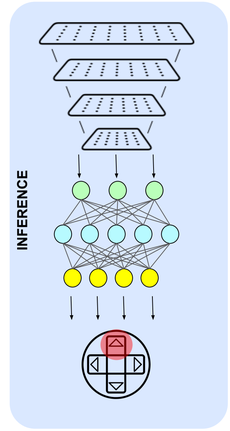


walk-through-walls.level

- **Experiment 2: Teaching Baba to beat walk-through-walls.level**
  - In this experiment Baba needed to synthesize skills learned in the curriculum implemented in experiment 1 in order to beat this level
  - In this level, Baba had to change the rule to allow it walk through walls so that it could eventually reach the flag.

Model Architecture & Implementation





We use **PPO** (Proximal Policy Optimization) as a way to compute the best policy for Baba to follow in the game. Furthermore, to encourage Baba to learn more new paths to the flag (exploration vs. exploitation), we use an "***entropy trick***" that lets Baba forget a little bit of what it's learned (exploited), so that it can explore more.
Our deep reinforcement learning model uses an **actor-critic framework.** Both the actor and the critic are convolutional neural networks (CNNs) with linear layers that work together to help Baba learn from the environment. We use two networks because it gives Baba more parameters to learn, thus encoding more information about the level.

- 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 a location (state), produces an estimate of how good it is; for example states closer to the flag generally receive a higher score.

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

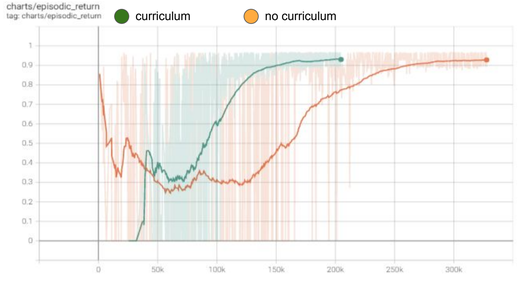
Model training is highly sensitive to hyperparameter selection. As such, we adapted our model hyperparameters to learn more effectively. Table on the left summarizes the hyperparameter values that provided best results.

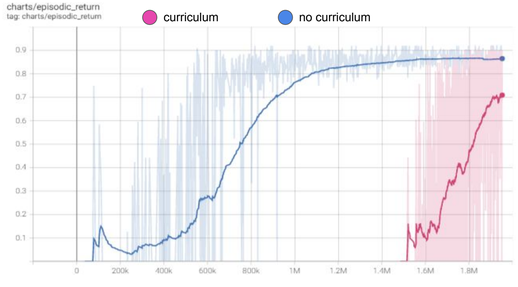
Results

Metrics Used

- **Episodic Return** - quantifies how well Baba is performing on a level. Achieving a sustained average episodic return above some threshold determines whether we beat a level ("Beat Level" column in table)
- **Steps** - number of *sequential* training examples required to beat a level, also a measure of time

The chart below compares the episodic return of a curriculum model (green) to a no-curriculum model (orange) trained on a level which requires Baba to have mastered finding the flag. While both models beat the level (achieve an average return above 0.9), the curriculum is able to learn in about 150k steps 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.





The chart above 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. Both models achieve the threshold for beating a level, which involved exceeding an average episodic return of at least 0.7. The curriculum model has a steeper slope, which illustrates how its previous curriculum helps it to learn the level more quickly. The no-curriculum model requires about 300k additional steps before reaching the return threshold, however because it spends around 4x as much time training on the level, it achieves a higher overall return.

Sr. No.	Strategy	Steps to Beat Level (k)	Total Steps (k)	Beat Level
1	no curriculum (baseline)	500.0	500.0	✕
2	curriculum	500.0	1950.0	✕
3	no curriculum + dual encoder	718.4	718.4	✓
4	curriculum + dual encoder	500.0	1950.0	✕
5	no curriculum + dual encoder + entropy trick	718.4	718.4	✓
6	curriculum + dual encoder + entropy trick	427.3	1877.3	✓

The table above shows a summary of the different strategies used to train Baba on more difficult levels, the results for each, the number of training steps (level-only and overall) required, whether Baba learned to beat the level (based on sustained episodic return over some threshold)
We draw the following conclusions from this table:

1. Comparing rows 1 and 2 to 3 and 4, we see that using a dual encoder architecture is the difference in beating a level.
2. Comparing rows 4 and 6, 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.
3. Comparing rows 5 and 6, we see that a curriculum-based model learns faster than a no-curriculum model, however, the latter returns a higher episodic return since it spends more time in learning.

Conclusion

- Curriculum learning helps the model learn to beat the level faster than no curriculum, but requires more total steps (our curriculum required 1.45m steps before attempting the level).
- Dual encoder architecture for the actor-critic pair reduces the number of timesteps by a factor of almost **50%**.
- Trade-off between exploration and exploitation is very important in reinforcement learning. Incentivizing Baba to explore the environment (by changing entropy penalty) helped it to learn new things.
- Model training is highly sensitive to hyperparameter selections - the entropy trick is required for models using curriculum.
- Based on our experiments with an AI agent, we speculate that while human and machine learning may both be improved using curricula, humans are able to learn faster due to prior knowledge in adapting to new situations without need for entropy

Future Work

- **Pixel encoding:** How well does our agent learn when using pixels as input compared to our object-based encoding?
- **Maximum potential:** Is the agent capable of beating levels which the experimenters do not know how to beat? How much help is required? What percentage of the current levels is the agent able to beat without training? With training?
- **More parameters:** Are larger neural networks (1M, 10M parameters) able to beat challenging levels which our current model (0.5M) cannot?
- **Variable size levels:** All experiments were run in levels contained in a 14x14 grid and CNNs do not naturally lend themselves to their input size varying over time. However, can we come up with a way to transfer knowledge learned on the 14x14 grid to other, perhaps larger, levels?

References

- Andrychowicz, M., Raichuk, A., Stańczyk, P., Orsini, M., Girgin, S., Marinier, R., Hussenot, L., Geist, M., Pietquin, O., Michalski, M., Gelly, S., & Bachem, O. (2020). What Matters In On-Policy Reinforcement Learning? A Large-Scale Empirical Study. *arXiv*. https://doi.org/10.48550/arXiv.2006.05990
- Engstrom, L., Ilyas, A., Santurkar, S., Tsipras, D., Janoos, F., Rudolph, L., & Madry, A. (2020). Implementation Matters in Deep Policy Gradients: A Case Study on PPO and TRPO. *arXiv*. https://doi.org/10.48550/arXiv.2005.12729
- Mnih, V., Kavukcuoglu, K., Silver, D., Graves, A., Antonoglou, I., Wierstra, D., & Riedmiller, M. (2013). Playing Atari with Deep Reinforcement Learning. *arXiv*. https://doi.org/10.48550/arXiv.1312.5602
- Narvekar, S., Peng, B., Leonetti, M., Sinapov, J., Taylor, M. E., & Stone, P. (2020). Curriculum Learning for Reinforcement Learning Domains: A Framework and Survey. *arXiv*. https://doi.org/10.48550/arXiv.2003.04960
- Schulman, J., Wolski, F., Dhariwal, P., Radford, A., & Klimov, O. (2017). Proximal Policy Optimization Algorithms. *arXiv*. https://doi.org/10.48550/arXiv.1707.06347