Using Human Gameplay to Augment Reinforcement Learning Models for Crypt of the NecroDancer

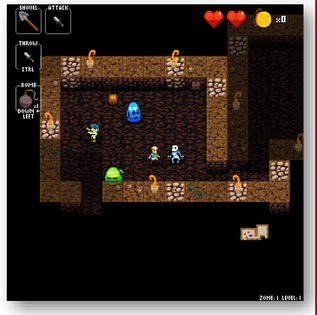
Buck Bukaty, Dillon Kanne
CS 221 - Department of Computer Science, Stanford University

Motivation

- In games, reinforcement learning algorithms must learn visual information and environmental dynamics from the ground up by taking random actions.
- Recent work shows how this lack of prior understanding puts RL algorithms at a distinct disadvantage in games intended for humans [1].
- We explore ways to use human gameplay to bootstrap the environmental understanding of deep reinforcement learning models.
- Specifically, we utilize convolutional network weights learned from human gameplay to initialize Deep Q-Network [2] and Policy Gradient [3] models.

Game Environment

- Our domain is the game *Crypt of the NecroDancer.*
- Turn based, takes place on a grid of tiles.
- Clear analogy to Markov Decision Process – take discrete action [left, right, up, down] and observe environment response.



Infrastructure

- Created software to capture screenshots of the game window during human play and label them with keyboard inputs.
- Recorded playthroughs of 100 randomly generated levels.
- Result: labeled dataset of ~12,000 game frames and corresponding button presses.

RL Environment, Challenges

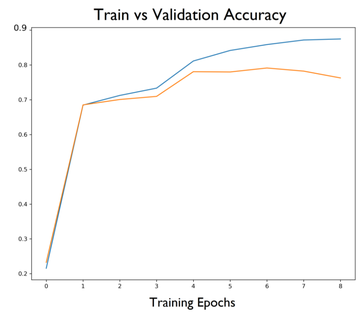
- No access to game state; manually scan life meter, gold meter, and level indicator for use in reward function.
- RL algorithms must train in real time, severely limiting training epochs.

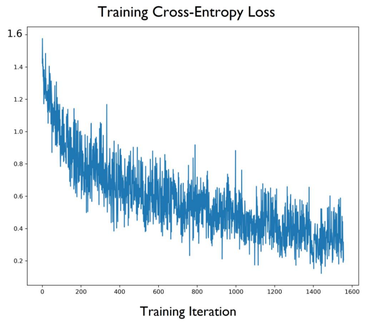
Behavioral Cloning

- Trained and tested convolutional networks for predicting human-like inputs in response to game situations.
- Saw improvement when input included additional time dimension, stacked 4 images as input in the same manner as Mnih [3].
- Tested network architectures, consideration given to the game-grid-aligned nature of input images.
- Transfer learning with ResNet not effective – the game's pixelated features were too dissimilar with real life curves and lines.
- Recorded gold acquired in 5 random test levels.

Architecture	% Val Accuracy	1	2	3	4	5	Mean
Simple – Single Image Input	73	16	14	0	35	0	13
Deeper – Single Image Input	77	3	0	16	35	34	17.6
Resnet – Single Image Input	78	41	31	23	18	2	23
Simple – Four Image Input	74	8	16	9	24	2	11.8
Deeper – Four Image Input	79	9	37	13	43	0	20.4
Resnet – Four Image Input	80	9	34	39	12	46	28
Optimal Human Play (Oracle)		154	185	144	200	169	170.4

- Compared with a ResNet34, our architecture was much less deep with comparable performance. This was useful for intensive reinforcement learning training, when the necessary information for gradient updates for the ResNet34 overflowed GPU memory.

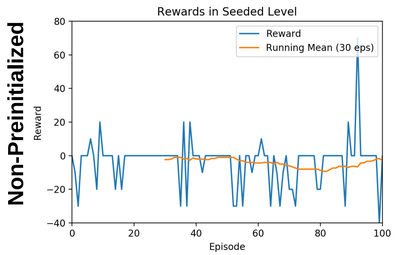


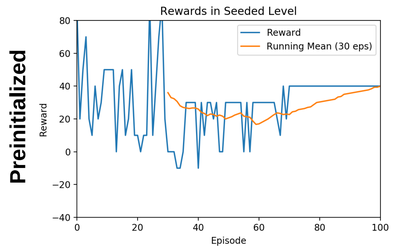


Policy Gradients (REINFORCE)

Note: All reinforcement learning training seen below is performed on the same random level, functionally an MDP whose state we can only interpret visually.

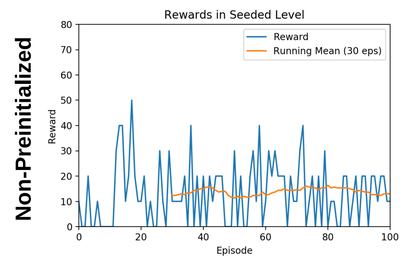
We interpret our best Behavioral Cloning network as a stochastic policy network, and perform gradient updates on it according to the REINFORCE algorithm. [5]

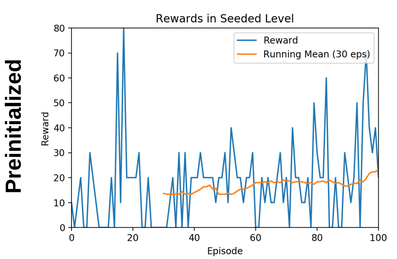




Deep Q-Network (DQN)

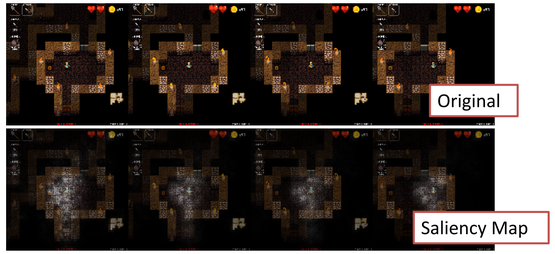
We interpret our best Behavioral Cloning network as a Q-function approximator, and perform gradient updates on it according to the Q-Learning algorithm with experience replay and freezing target networks. [6]





Feature Visualization

The dark spot over the low health blue slime indicates that if it were removed from the original image, the score given to the 'left' class would decrease significantly, as expected. Interestingly, the corresponding white spot on the green slime to the right indicates that its decision to move left would be easier if it weren't there.



Here, the human player has defeated all the enemies in their current room, and is deciding where to go next. The saliency map indicates that the empty squares around the player are primarily contributing to its decision making. More interestingly, there is also some weight given to the minimap in the bottom right

Discussion and Future Work

- Though somewhat unstable throughout training, our preinitialized reinforcement learning models reasonably outperformed their non-preinitialized counterparts.
- All the models we tested had difficulty primarily with navigating through levels. Our best behavioral cloning model is quite good at fighting through rooms of enemies, though, and its play is at times difficult to distinguish from that of a human.
- Given more time to train and tweak our preinitialized reinforcement learning models, we believe they will be able to consistently outperform our best behavioral cloning model, as they did at some points throughout training. We plan to focus effort on finding good values for initial and final epsilon for the epsilon-greedy policy of the DQN model.

References

[1] R. Dubey, P. Agrawal, D. Pathak, T. L. Griffiths, and A. A. Efros. Investigating human priors for playing video games. *arXiv preprint arXiv:1802.10217*, 2018.

[2] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. Riedmiller. Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*, 2013.

[3] A. Karpathy. Deep reinforcement learning: Pong from pixels. http://karpathy.github.io/2016/05/31/rl/. 2016.

[4] G. V. Cruz Jr, Y. Du, and M. E. Taylor. Pre-training neural networks with human demonstrations for deep reinforcement learning. *arXiv preprint arXiv:1709.04083*, 2017.

[5] Implemented in PyTorch based on https://github.com/pytorch/examples/blob/master/reinforcement_learning/reinforce.py

[6] Implemented in PyTorch based on https://pytorch.org/tutorials/intermediate/reinforcement_q_learning.html