Deep Q-Learning for Tree Chopping in Craftium with Reward Shaping and Visual Augmentation

By: Haijun Si, Ash Sze, and Cheng Xi Tsou

Introduction and Background

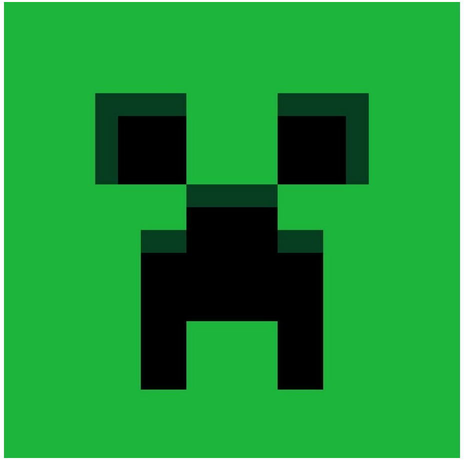
Problem Statement: How does reward shaping and visual augmentation affect an RL agent ability to learn in high-dimensional environments such as Craftium?

We train a **deep Q-learning** model to “chop a tree” in minecraft and modify the visual space by applying “blur”, “crop” and “edge mapping”.

Minecraft: open-world, sandbox, voxel-based game.

We will demonstrate what we have learned about reward shaping, our methodology, and the success of our models.

Environment Set-up



Observation Space:
Box(0, 255, (64, 64, 3), uint8)

Actions: Discrete(8) (nop, forward, jump, dig (used to chop), mouse x+, mouse x-, mouse y+, mouse y-)

Chopping Trees

Reinforcement Learning Methods

Select action with epsilon-greedy policy and execute action

Store transition in replay

Compute TD error and update policy network

Algorithm 1 Deep Q-Learning Algorithm (Mnih et. al 2013)

Initialize replay memory $\mathcal{D}$ with capacity N
Initialize action-value function $Q(s, a; \theta)$ with random weights θ
for each episode **do**
  Initialize state s_0
  for each step in the episode **do**
    Select action a_t using ϵ-greedy policy:

$$a_t = \begin{cases} \text{random action} & \text{with probability } \epsilon \\ \arg\max_a Q(s_t, a; \theta) & \text{otherwise} \end{cases}$$

    Execute action a_t and observe reward r_t and next state s_{t+1}
    Store transition (s_t, a_t, r_t, s_{t+1}) in $\mathcal{D}$
    Sample random minibatch of transitions (s_j, a_j, r_j, s_{j+1}) from $\mathcal{D}$
    Compute target y_j:

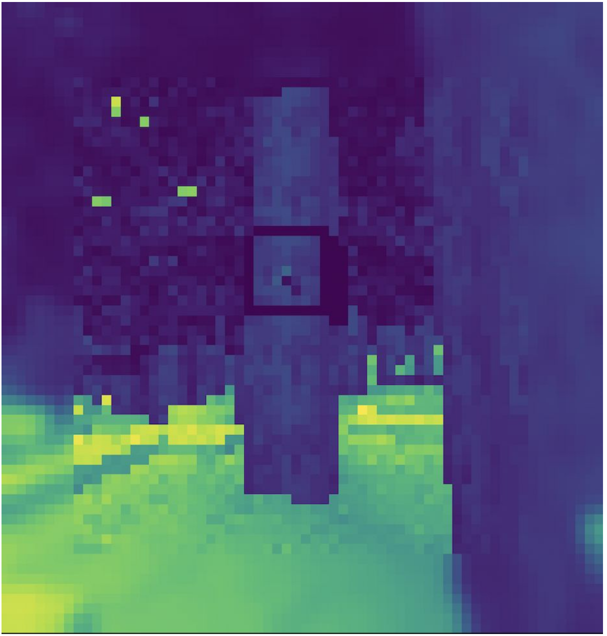
$$y_j = \begin{cases} r_j & \text{if terminal state} \\ r_j + \gamma \max_{a'} Q'(s_{j+1}, a'; \theta^-) & \text{otherwise} \end{cases}$$

    Perform gradient descent step on loss:

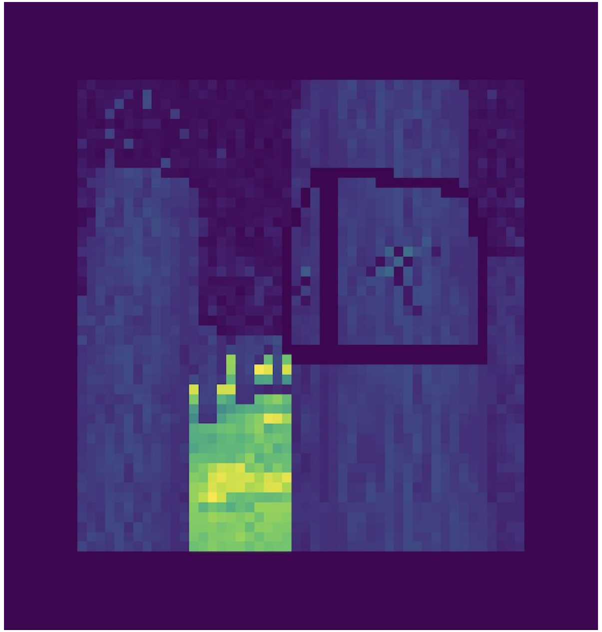
$$L(\theta) = \mathbb{E}[(y_j - Q(s_j, a_j; \theta))^2]$$

  end for
end for

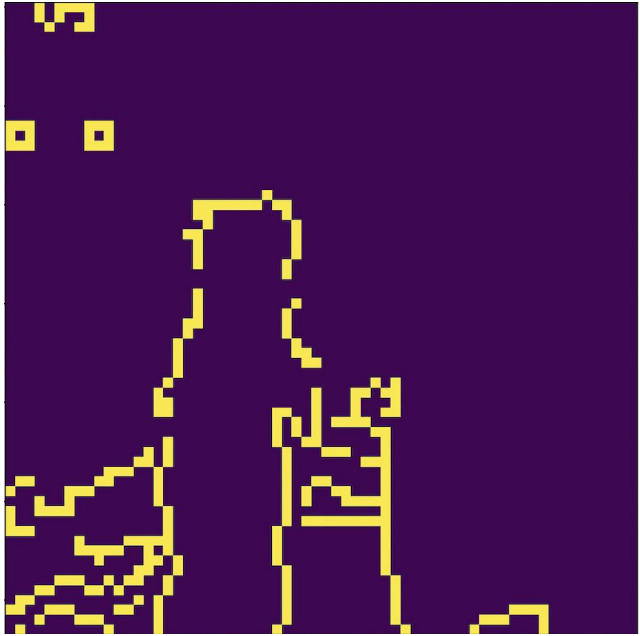
Visual Augmentation



75% edge Blurred



75% center crop



Edge map

Experimental Set-up

Tech Stack

Programming: Python 3.12 and Lua

RL Framework: Craftium (Minetest)

Libraries: PyTorch, numpy, gymnasium, matplotlib

Agent and Environment

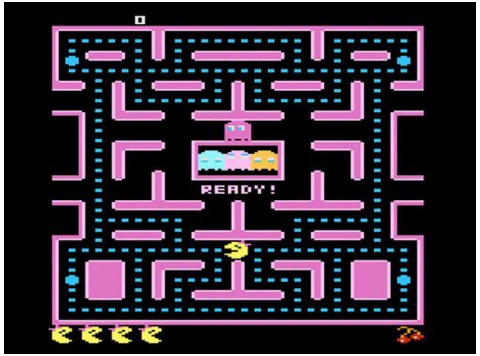
DQN agent with CNN based Q-network

Grayscale 64-by-64 pixel observation

Discrete actions: forward, dig, mouse x+, x-, y+, y-

1000 timesteps per episode

4 frames for framestack with 4 frameskips



Training Details and Reward Shaping

- Baseline with simple reward shaping
- Baseline with complex reward shaping
- Blur (gaussian), crop (center crop) and edgemap (canny edge) with simple reward shaping

200-300 episodes of training

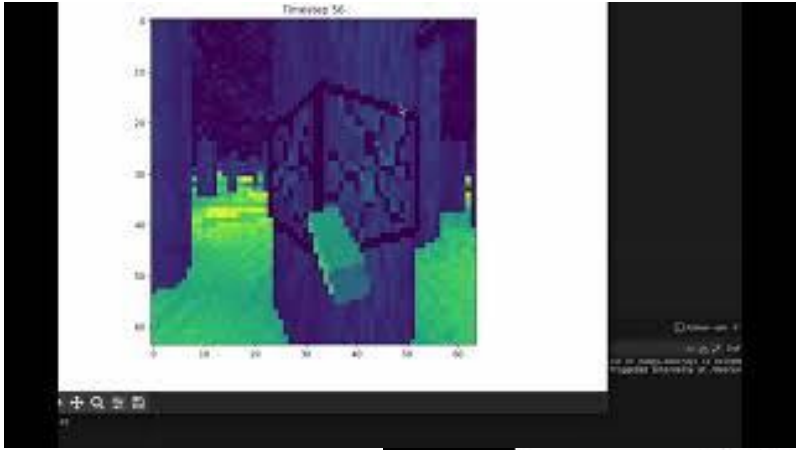
Epsilon decay from 0.9 -> 0.05 over 3000 steps

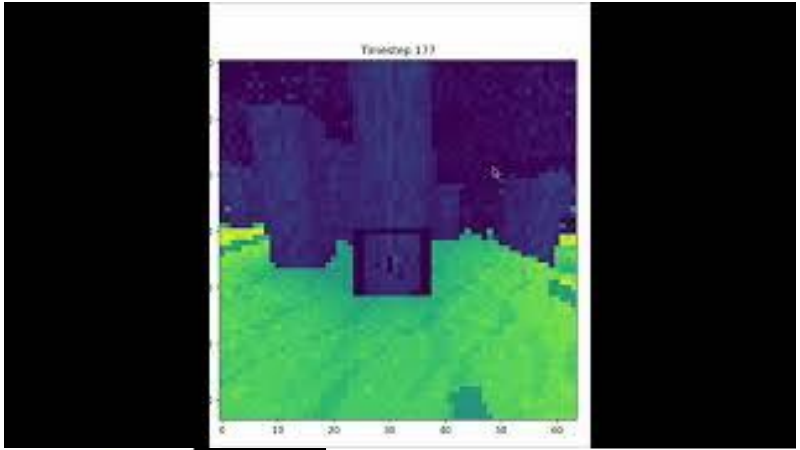
Optimization with AdamW, Learning rate 1e-4 and 128 batch size

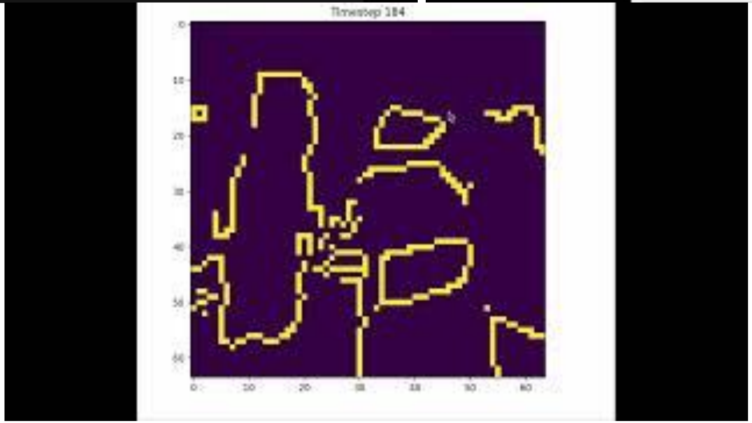
Simple rewards:

- +15.0 for digging a tree block
- −5.0 for digging a non-tree block
- +0.5 per timestep of sustained digging (LMB held)

Chopping a tree

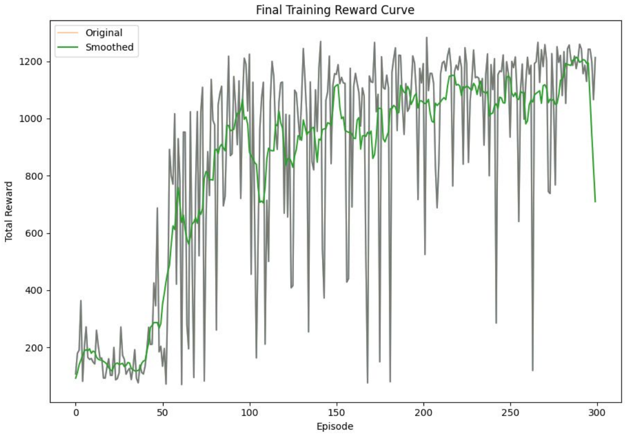


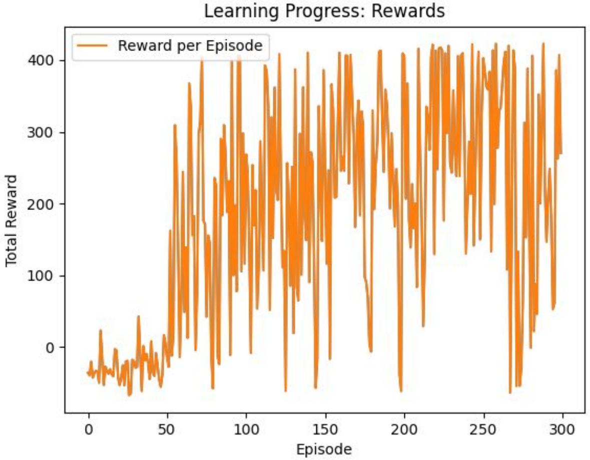


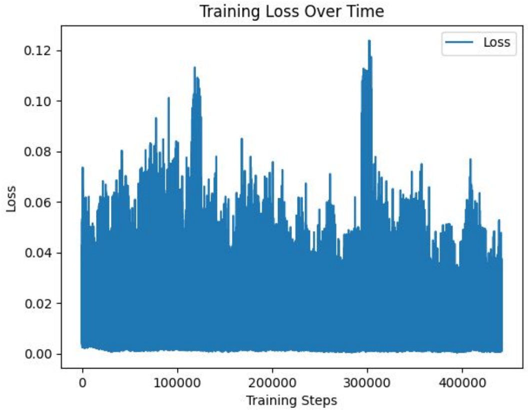


Results

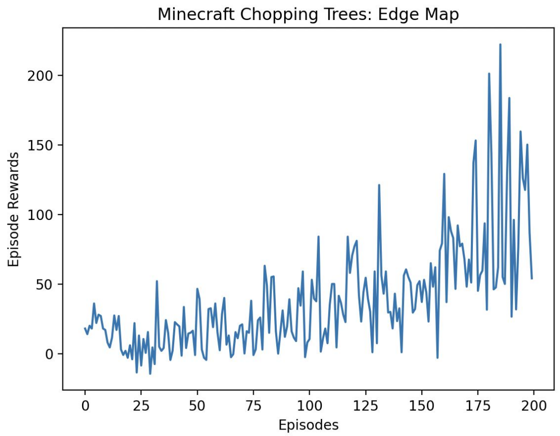
Metrics for evaluation are reward per episode, Q-network loss per gradient update and training time to convergence (in episodes).

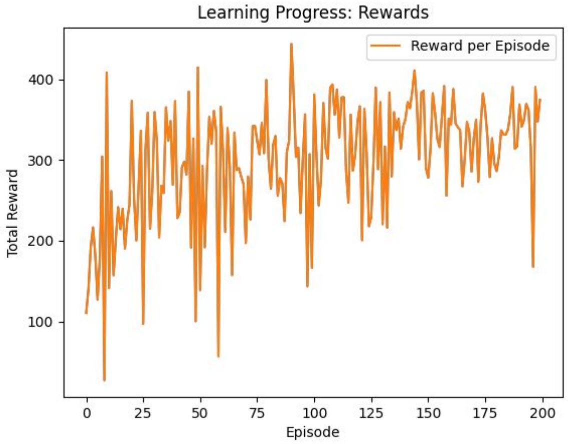


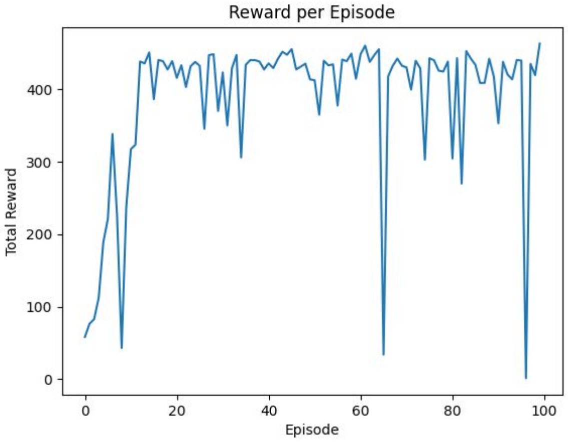




Results pt 2


Minecraft Chopping Trees: Edge Map
Episode Rewards
Episodes


Learning Progress: Rewards
Reward per Episode
Total Reward
Episode


Reward per Episode
Total Reward
Episode

Conclusion

What we learned

- Integrate open-source projects and external APIs with RL Agents
- Implement a custom DQN with tailored reward shaping
- Visual alternation can **stabilize learning** in pixel-based environments.

Future Direction

- Extend the framework to more complex problems
- Develop more reward shaping strategies for multi-task setups.
- Explore additional forms of visual degradation:
  - Dynamic blur!
  - Hybrid visual alterations