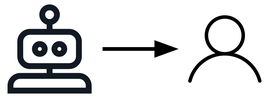
Using Modularized RL to Create a Minecraft PvP Agent

Alesandro Gopal, Xiaojun Ge, Ashley Hu, Annabel Qin

Purpose

Minecraft's complex 3D world and real-time combat make it a strong environment for AI, requiring agents to perceive, plan, and act under noisy conditions. Traditional bots rely on hand-crafted rules, but reinforcement learning offers the potential for more creative, human-like behavior.

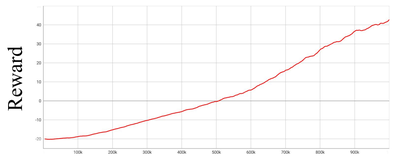


Current systems fail to capture the flexible strategies players develop through experience. By breaking combat into primitive skills, we aim to evaluate whether learned agents can match rule-based performance while also displaying more human-like tactics and adaptation.

Aiming Subskill

Controls: Camera movement (Yaw/Horizontal and Pitch/Vertical)
Target Behavior: Random movement at a minimum distance.
Rewards / Penalties:
+ **Alignment Progress**, reward for reducing angular difference to target across timesteps.
+ **Alignment Accuracy**, reward for accurately looking at the target.
– **Excess camera movement**, penalty proportional to Yaw/Pitch adjustment (discourages jittering/wild swings).

Figure 1: Aiming Reward

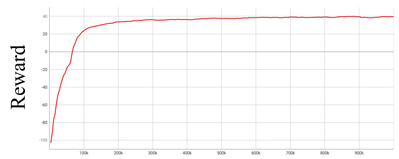


Attacking Subskill

Controls: Attack button.

Target Behavior: Random movement within certain distance.
Rewards / Penalties:
+ **Successful hit**, with bonus for high-precision aim.
+ Small bonus for **withholding attack** when target is out of range.
– **Missed attack** (in range but fails to connect).
– **Not attacking** when target is in ideal range.

Figure 2: Clicking Reward

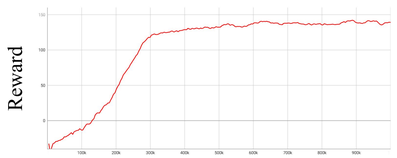


Movement Subskill

Controls: WASD, Sprint, Jump.

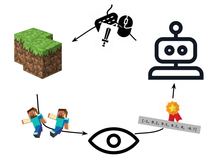
Target Behavior: Random movement with impact forces.
Rewards / Penalties:
+ **Moving toward target**, while maintaining optimal distance.
+ **Optimal distance**, distance of 2-3.5 blocks.
+ **Dodging**, reward for not being in targets direct line-of-sight.
– **Inactivity**, no WASD input
– **Jumping** (not optimal for PvP)

Figure 3: Movement Reward



PPO and Models

PPO is an actor critic network with extra modifications to enforce learning stability. As usual, actor network outputs the action probabilities and critic estimates rewards for the current state and given action.
Clipped Objectives limits how much the policy can change in one update to prevent overly large updates that disrupts skills that the agent has learned, ensuring stability during learning and making it more suitable for complex tasks.

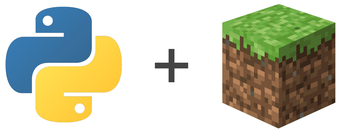


Model Summary:
- The modularized skills are each built with two hidden layers of 64 neurons with Tanh activations (movement has 128 to allow for more complex behavior)
- Small exploration rate that selects random actions during training to allow the agent to discover new possible strategies

Environment

A custom Python simulation that mimics Minecraft PvP in an infinite flat 3D space was built. The agent spawns within a 2-block area at the origin, and the target appears 6–12 blocks away. The agent and target control movement (forward / back / strafe), jumping, sprinting, attacking, and continuous camera orientation.

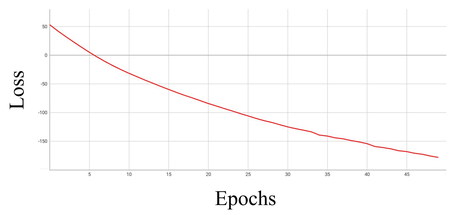
To support modularized learning, the agent is only allowed access to a portion of the controls, enabling mastery of individual skills before combining them.

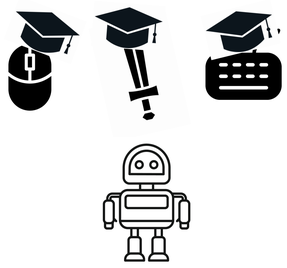


Combined Model

Skill Distillation via Behavior Cloning:
The three subskill experts (movement, aiming, and clicking) generated expert **demonstrations**. These demonstrations were merged into a single action space and used to train a **distilled student model** through behavior cloning using **negative log-likelihood loss**. Effectively, the student learns to **maximize the likelihood of the actions taken by experts**. The policy network was frozen to prevent learning the loss function as the reward function.

Figure 4: Distillation Loss



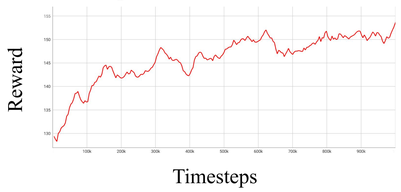


Comparison and Fine Tuning: When compared to a combination of the three experts, the distilled agent was on equal footing. The distilled agent was then put into a combined environment that rewarded winning vs losing. When fighting a conventional PvP bot opponent, the distilled bot won 97% of the time.

Raw Agent

Purpose: A comparison agent trained directly on the full PvP task without modularization. This is to compare the results of modularized RL vs conventional RL.
Rewards / Penalties: A combination of normalized rewards from each subskills.

Figure 5: Raw Task Reward



Results

Subskill Performance:
Each modularized subskill agent (movement, aiming, and clicking) converged reliably and achieved expert-level performance within its subskill task.

Skill Distillation:
The resulting distilled agent performed on equal footing with a spliced together combination of the three expert agents. This indicates that the student model successfully captured the behaviors of all three experts. The distilled agent had a 97% win rate against conventional PvP bots.

Comparison to Non-Modularized Agent:
While the rewards curve suggested apparent learning progress, qualitative evaluation showed that the non-modularized agent failed to learn any meaningful behavior. In contrast, the modularized and distilled agent consistently exhibited structured movement, accurate aim, and effective attack timing, demonstrating that straightforward reward shaping alone is insufficient for this task without skill decomposition.

Future Work

While the modularized RL agent effectively learns the core Minecraft PvP mechanics, future work will focus on creating more human-like strategic behavior. One pathway to do so is incorporating simulated server latency into the environment, as many competitive techniques, such as w-tapping and hit selection, rely on delayed interactions.
Additionally, adding more diverse opponents may lead to more interesting learning. Human strategy is built upon adapting and exploiting the opponent's mistakes. When put into a scenario against a bot that is algorithmically designed to play the same way every time, there are no mistakes to exploit thus no strategy. Future environments could include differing skill opponents, self play, and adaptive opponents.

References

- PPO: [1707.06347] Proximal Policy Optimization Algorithms
- Modular RL: [2207.00429] Modular Lifelong Reinforcement Learning via Neural Composition
- Imitation Learning: Imitating Language via Scalable Inverse Reinforcement Learning

Acknowledgments

Thanks to Eric Ewing and our TA advisor Matthew Prenovitz