## # Experiment 2 (Vicky & Jiayue)

Mainly trained on the hyperparameters. Here's what we did and didn't complete and some suggestions.

### ##Understanding the training state

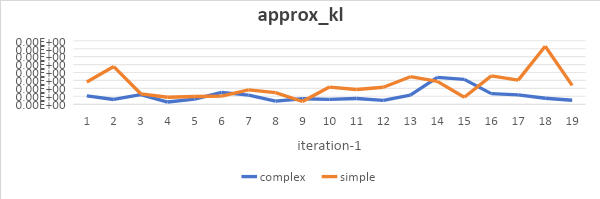
### \*\*Time:\*\* ~2 hours.

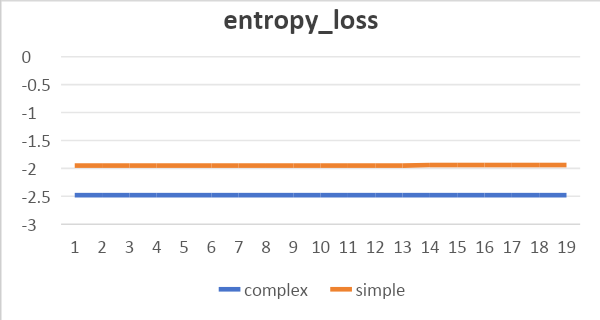
* approx\_kl: Approximate Kullback-Leibler divergence. It's a measure of how different the new policy is from the old policy. A lower value indicates less difference between the policies
* clip\_fraction: The fraction of gradients that were clipped. In PPO, this is related to how often the gradient clipping is applied.
* clip\_range: The range for clipping in PPO.
* entropy\_loss: The loss related to entropy, used to encourage exploration and prevent premature convergence to suboptimal policies. A larger negative value could indicate more exploration.
* explained\_variance: It is a metric used to assess how accurately a model predicts rewards:
  + Close to 1: This indicates that the model's predictions are very close to the actual rewards, suggesting the model has learned well.
  + Close to 0: This means there's a large discrepancy between the model's predictions and the actual rewards, indicating the model has not learned effectively.
  + Negative values: These suggest that the model is performing worse than even a basic benchmark, which might indicate fundamental issues.
* learning\_rate: The current learning rate for the training.
* loss: The loss value is an indicator of how wrong the model's predictions are. It is calculated using a loss function that compares the model's predictions with the actual observed data. The goal during training is to minimize this loss value.
* n\_updates: It refers to the number of times the policy parameters have been updated, indicating how often the model's decision-making strategy has been adjusted based on the feedback from the environment.
* policy\_gradient\_loss: The loss incurred during the optimization of the policy using policy gradient methods. A lower value suggests that the policy is performing well, requiring minor adjustments. It implies that the agent's actions are closely aligned with the expected optimal actions. A higher value indicates a greater discrepancy between the agent's current policy and the optimal policy, necessitating more significant adjustments to the policy.
* value\_loss: The loss for the value function, which is part of the critic in actor-critic methods like PPO. A higher value loss indicate a large difference between predicted and actual returns.

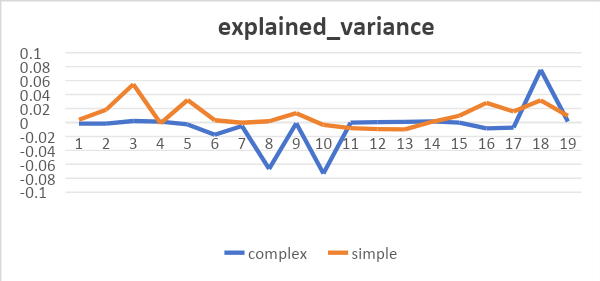
### ## Experiment On hyperparameter --Jiayue

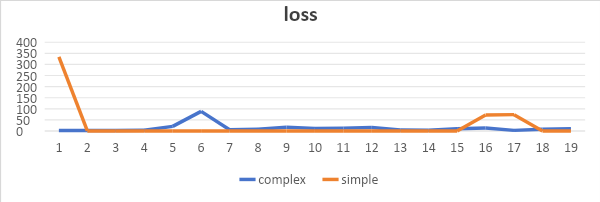
### 1.Original SIMPLE\_MOVEMENT v.s. COMPLEX\_MOVEMENT(win):

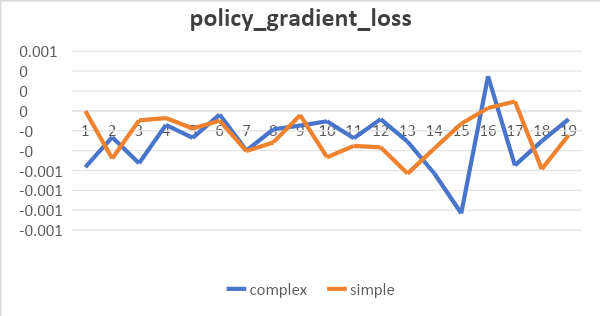
### \*\*Time:\*\* ~2 hours.

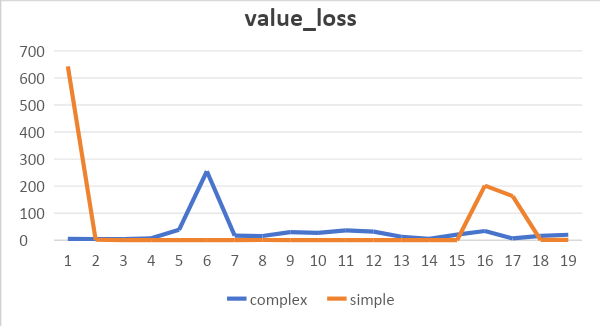








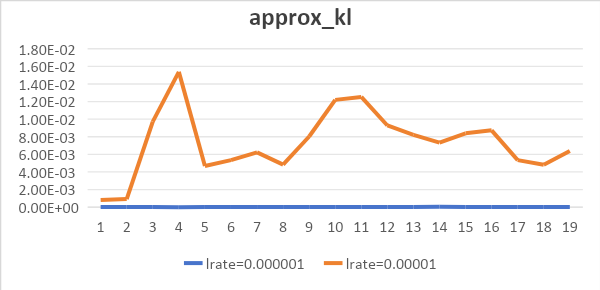


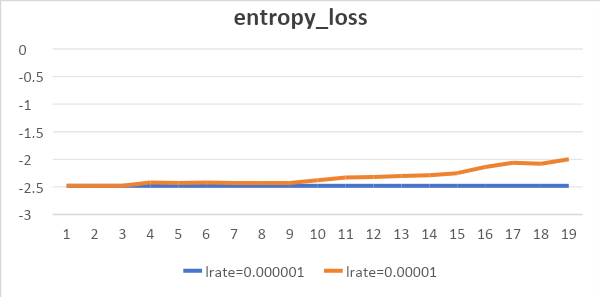


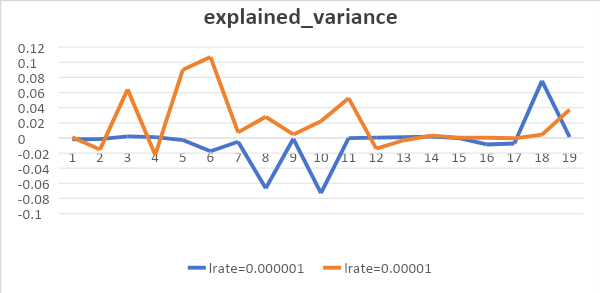
Complex movement outperforms simple one with a more steady performance and smaller losses across most indicators.

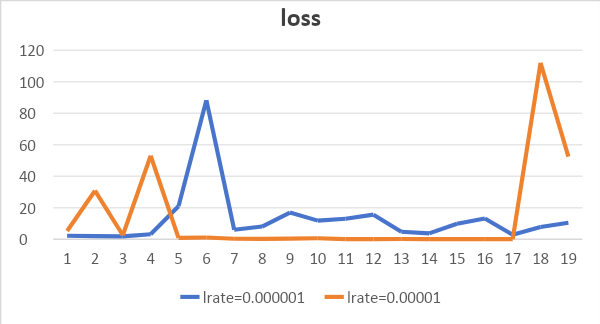
### PPO parameter: learning\_rate=0.000001(win) vs. learning\_rate=0.00001:

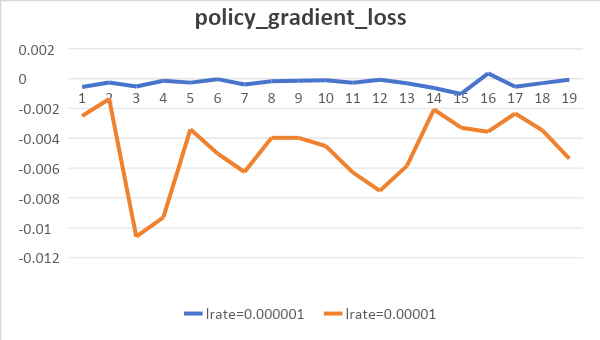
### \*\*Time:\*\* ~2 hours.

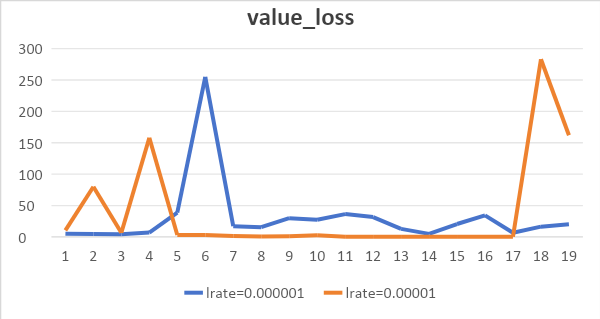








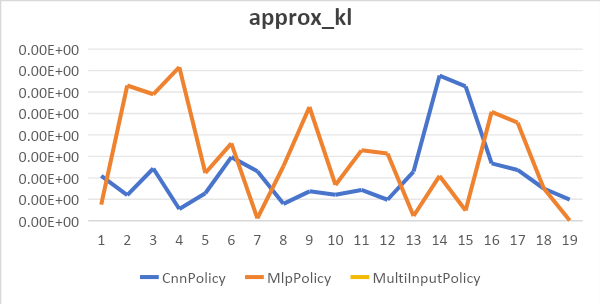


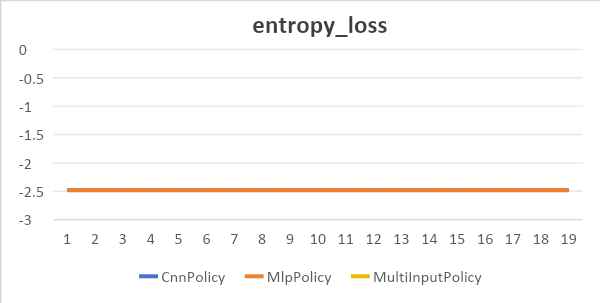


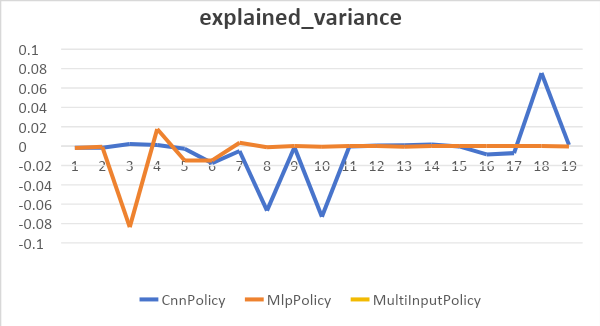
While both showed unstable peak values in value\_loss and loss, only policy\_gradient\_loss favored learning\_rate=0.00001. The other losses indicated faster improvement with smaller learning rate, suggesting it could be explored further.

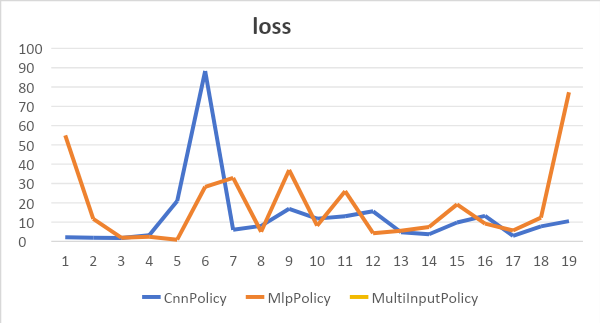
### PPO parameter："CnnPolicy" v.s. 'MlpPolicy' v.s. "MultiInputPolicy"

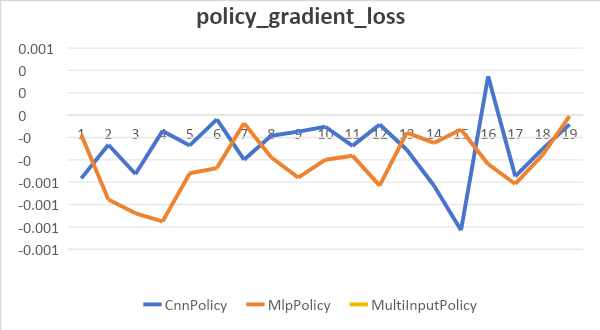
### \*\*Time:\*\* ~3 hours.

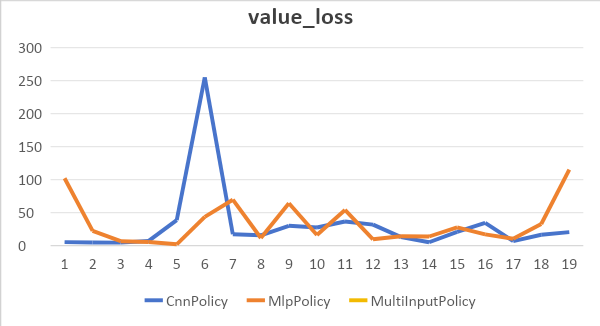












Please note that the MultiInputPolicy parameter may result in an error message:

Error: for key, subspace in observation\_space.spaces.items():

"AttributeError: 'Box' object has no attribute 'spaces'"

This suggests a compatibility issue between the MultiInputPolicy and the observation space type, given that the spaces attribute isn't present in the Box object. A potential solution is to employ strategies matched with the observation space, like MlpPolicy or CnnPolicy, depending on its type. Changes to the environment's configuration may be required to evaluate the MultiInputPolicy effect if necessary.

Comparing MlpPolicy and CnnPolicy, considerable variance exists in various parameters, suggesting no clear winner. I believe CnnPolicy suffices.

### ##Functionality of Logging Rewards --Vicky

### \*\*Time:\*\* ~20 hours.

I was thinking of adding a function to track the rewards during training in order to analyze the model's effectiveness, but unfortunately my coding skills are terrible, so I'm sorry to say that even studying a lot of tutorials and examples didn't allow me to complete this function. I will continue to study and may refine the code in the future if I could.

### 

### ## Recommendations for Next Experiments

1. In my recent research, I realized the challenge of learning both the implementation and training aspects of reinforcement learning in a short period of time. To speed up the process, I found some existing Mario PPO reinforcement learning programs online. We can use them directly or as a reference. We can then focus on understanding the RL and PPO algorithms, as well as analyzing how hyperparameters affect model training, which seems more manageable.

relative links:

https://www.youtube.com/watch?v=2eeYqJ0uBKE

<https://github.com/yumouwei/super-mario-bros-reinforcement-learning>

1. The six output parameters provided seem excessive, potentially conflicting (some metrics favor Parameter A, while others prefer Parameter B). It makes sense to define a metric to assess the model's quality or rank these loss functions according to their significance, to determine the model's performance.
2. Adjustments to the environment's configuration may be necessary to evaluate the MultiInputPolicy effect, if necessary.
3. Due to time constraints, some parameters haven’t undergone experiments yet, such as：

env = DummyVecEnv([lambda: env])

# env = SubprocVecEnv([lambda: env], start\_method="spawn") # EXPERIMENT: Try to change how we run the simulations

# # 6. Normalize the observation/rewards/both

# env = VecNormalize(env, norm\_obs=True) # EXPERIMENT: See if this normalization (on observation, or on rewards with norm\_rewards=True) changes performance of Agent

model.learn(

    total\_timesteps=10000, callback=callback

)  # EXPERIMENT: train for longer periods of time to see how it improves over that time span

also the learning rate which needs further testing.