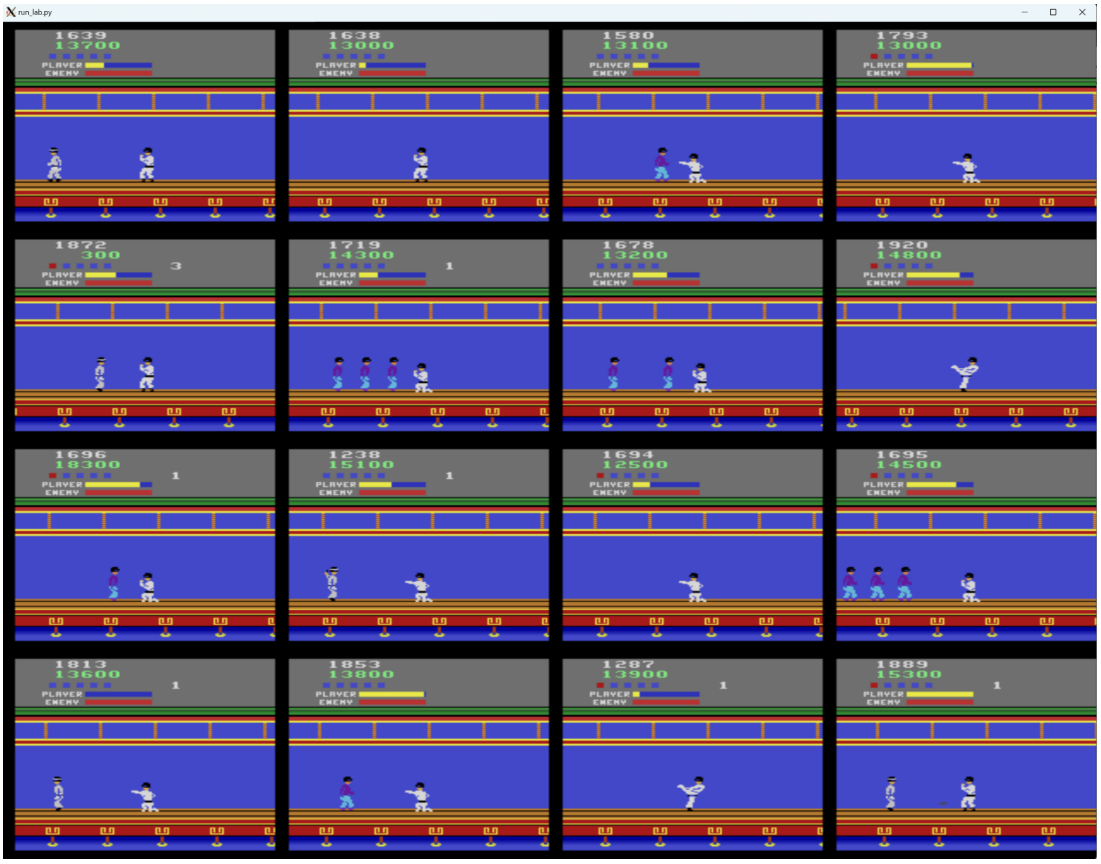
KungFuMaster-v0

Game description

KungFuMaster-v0 is an Atari game developed by programmer Dan Kitchen. It is an action-packed video game that challenges players to embody a skilled martial artist on a mission to rescue a kidnapped damsel in distress. The game is set in a retro-style environment, inspired by classic martial arts films of the 1980s.

In KungFuMaster-v0, players navigate through multiple levels, each filled with adversaries who must be defeated to progress further. The protagonist possesses various martial arts techniques, including kicks and punches, which can be strategically deployed to overcome enemies. Additionally, the character can perform impressive acrobatic maneuvers to dodge incoming attacks and gain an advantage in combat.

In SLM-Lab, the state space of the game is represented by a 4-channel image with a resolution of 84x84 pixels. The action space consists of 14 discrete actions the agent can choose from. These actions correspond to various martial arts moves and defensive maneuvers available to the agent.



Algorithms

This section provides a short overview on the algorithms used in training the KungFuMaster-v0 agent.

1. *REINFORCE*: The Reinforce algorithm utilizes policy gradients to optimize the agent's policy directly. It samples trajectories, computes the total rewards, and updates the policy parameters to maximize expected rewards.
2. *SARSA (State-Action-Reward-State-Action)*: SARSA is an on-policy algorithm that learns action values for each state-action pair. It estimates the Q-values through iterative updates based on observed rewards and the next action chosen by the agent.
3. *Deep Q-Networks (DQN)*: DQN combines deep neural networks with Q-learning to approximate the action-value function. It uses experience replay and a target network to stabilize training and improve convergence.
4. *Double Deep Q-Networks (DDQN)*: DDQN is an extension of DQN that addresses the overestimation bias issue. It employs two separate value networks, one for selecting actions and the other for evaluating their values.

5. *Actor-Critic (A2C)*: A2C is an on-policy algorithm that combines policy-based and value-based methods. It uses an actor network to select actions and a critic network to estimate state values, enabling more efficient learning.
6. *Proximal Policy Optimization (PPO)*: PPO is an on-policy algorithm that optimizes the policy by iteratively updating it while maintaining a constraint on the policy divergence. It balances exploration and exploitation to improve sample efficiency.
7. *Asynchronous Advantage Actor-Critic (A3C)*: A3C is a distributed variant of the A2C algorithm, where multiple agents run in parallel and update a shared model. It leverages asynchronous updates to accelerate learning and enhance exploration.

Experimental results

NOTE: All the configuration files content are stored in the configs directory. Kindly check them out there.

Each graph represents the total reward achieved during training using different reinforcement learning algorithms on the KungFuMaster-v0 game. The x-axis indicates the number of time steps, while the y-axis represents the corresponding reward.

1. Reinforce

Algorithm: The algorithm is REINFORCE (line: 6). Gamma (discount factor) is set to 0.99 (line: 11) and it makes use of a baseline for the reinforcing signal (line: 9). The encouragement of exploration is there, by adding entropy with a linear decay schedule to the loss. (line: 12-18).
Network Architecture: The network architecture is a ConvNet (line: 25). It consists of convolutional layers and a fully connected layer. The convolutional layers have the following configuration (line: 27-31):

1. 32 filters, 8 kernel size, 4 stride, 0 padding, 1 dilation
2. 64 filters, 4 kernel size, 2 stride, 0 padding, 1 dilation
3. 32 filters, 3 kernel size, 1 stride, 0 padding, 1 dilation

The fully connected layer has 256 units (line: 32). The activation function used is ReLU (line: 33). Normalization is enabled and batch normalization is disabled (line: 34-35). The weights are initialized using orthogonal initialization (line: 36). Gradient clipping is set to 0.5 (line: 37). The loss function used is Mean Squared Error (line: 38-40). The optimizer is RMSprop with a learning rate of 0.01 (line: 41-44). The learning rate is constant throughout, specified

by the null learning rate scheduler (line: 45). Training on GPU is enabled (line: 46).
Environment: The environment name is Atari KungFuMaster-v0 (line: 50)**.** It consists of frame concatenation operation for 4 consecutive frames, reward scaling using the sign function and it operates in 16 parallel environments (line: 52-55). Training length consists of 750,000 time steps (line: 56).
Evaluation: The training is not distributed across multiple instances (line: 63). The agent is evaluated every 10,000 time steps and logging also occurs after every 10,000 time steps (line: 64-65). During evaluation, only one episode is run (line: 67). Rigorous evaluation is disabled (line: 66).

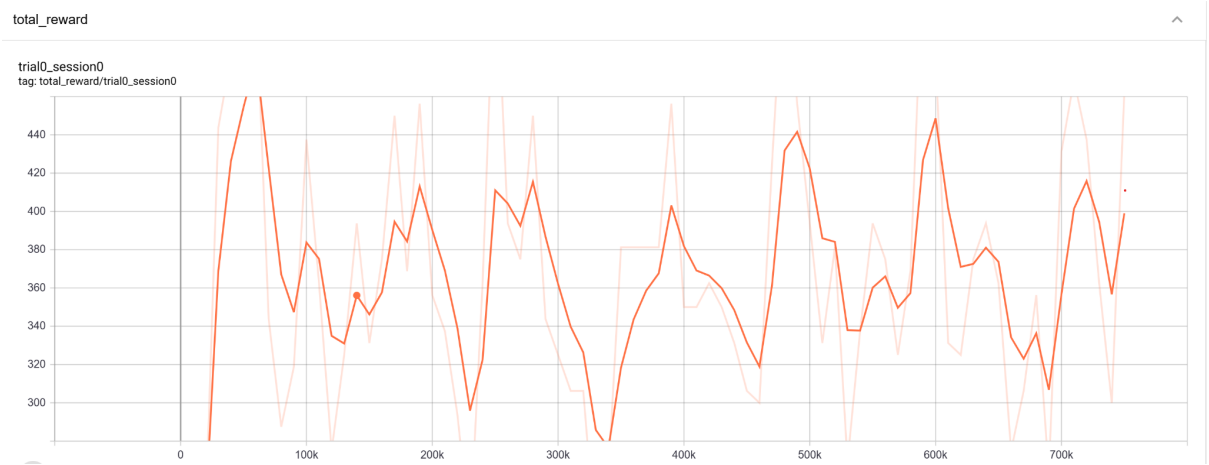


Figure 1: Total reward during REINFORCE training for KungFuMaster-v0

2. SARSA

Algorithm: The algorithm is SARSA (line: 6). The action probability type is argmax and the action policy is epsilon-greedy (line: 7-8) with linear decay (line: 9-15) of the exploration variable specification (epsilon is initialized to 0.8 and annealed to 0.1 between time steps 5,000 and 100,000) Gamma (discount factor) is set to 0.99 (line:16). The network will be trained every 32 steps because of the training_frequency parameter (line: 17).
Training frequency: Training is batch-wise, because we have selected OnPolicyBatchReplay memory (line: 20), and the batch size is 32 (line: 21) * 16 number of parallel environments (line: 54).
Network Architecture: The network architecture is a ConvNet (line:24). It consists of convolutional layers and a fully connected layer. The convolutional layers have the following configuration (line: 25-29):

1. 32 filters, 8 kernel size, 4 stride, 0 padding, 1 dilation
2. 64 filters, 4 kernel size, 2 stride, 0 padding, 1 dilation
3. 32 filters, 3 kernel size, 1 stride, 0 padding, 1 dilation

The fully connected layer has 256 units (line: 30). The activation function used is ReLU (line: 31). The weights are initialized using orthogonal initialization

(line: 32). Normalization is enabled and batch normalization is disabled (line: 33-34). The same optimizer is used throughout training (line: 35). Gradient clipping is set to 0.5 (line: 36). The loss function used is Mean Squared Error (line: 37-39). The optimizer is RMSprop with a learning rate of 0.0001 (line: 40-43). The learning rate is constant throughout, specified by the null learning rate scheduler (line: 44). Training on GPU is enabled (line:45).
Environment: The environment name is Atari KungFuMaster-v0 (line: 49)**.** It consists of frame concatenation operation for 4 consecutive frames, reward scaling using the sign function and it operates in 16 parallel environments (line: 51-54). Training length consists of 750,000 time steps (line: 55).
Evaluation: The training is not distributed across multiple instances (line: 62). The agent is evaluated every 100,000 time steps (line: 63). During evaluation, only one episode is run (line: 65). Rigorous evaluation is disabled (line: 66).

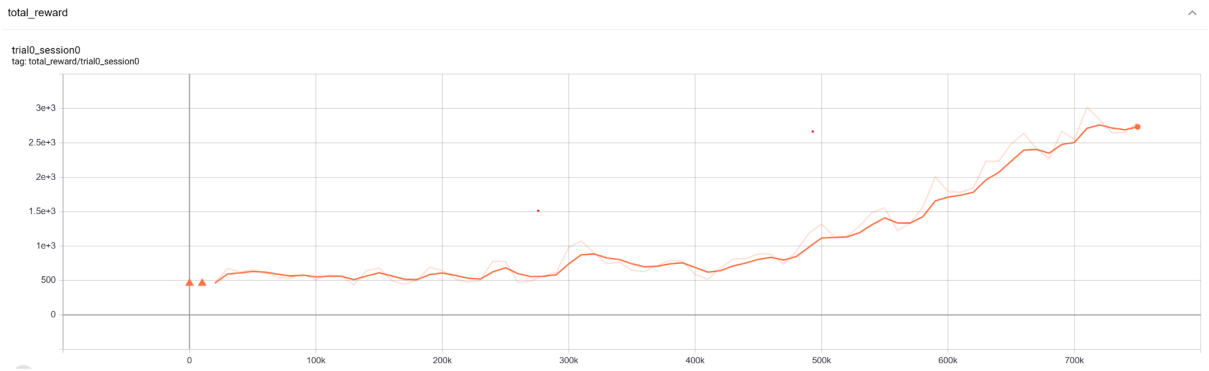


Figure 2: Total reward during SARSA training for KungFuMaster-v0

3. DQN

Algorithm: The algorithm is DQN (line: 6). The action probability type is argmax and the action policy is epsilon-greedy (line: 7-8) with linear decay (line: 9-15) of the exploration variable specification (epsilon is initialized to 1.0 and annealed to 0.01 between time steps 10,000 and 1,000,000). Gamma (discount factor) is set to 0.99 (line: 11).
Training frequency: Training starts after the agent has made 10,000 steps in the environment (line: 20) and occurs every 32 steps from then on (line: 19). It samples 1 batch from the Replay memory (line : 18) and makes only 1 parameter update (line: 17). Each batch has 32 elements (line: 24). A maximum of 200,000 samples are stored in the Replay memory (line: 25).
Network Architecture: The network architecture is a ConvNet (line: 29). It consists of convolutional layers and a fully connected layer. The convolutional layers have the following configuration (line: 30-34):

1. 32 filters, 8 kernel size, 4 stride, 0 padding, 1 dilation
2. 64 filters, 4 kernel size, 2 stride, 0 padding, 1 dilation
3. 32 filters, 3 kernel size, 1 stride, 0 padding, 1 dilation

The fully connected layer has 512 units (line: 35). The activation function used is ReLU (line: 36). The weights are initialized using orthogonal initialization (line: 37). Normalization is enabled and batch normalization is disabled (line: 38-39). Gradient clipping is set to 10.0 (line: 40). The loss function used is Smooth L1, a combination of MSE and MAE (line: 41-43). The optimizer is RMSprop with a learning rate of 0.0001 (line: 44-47). The learning rate is constant throughout, specified by the null learning rate scheduler (line: 48). The parameters are completely replaced with the updated ones after 1,000 steps (line: 49-50). Training on GPU is enabled (line: 51).
Environment: The environment name is Atari KungFuMaster-v0 (line: 55). It consists of frame concatenation operation for 4 consecutive frames, reward scaling using the sign function and it operates in 16 parallel environments (line: 56-59). Training length consists of 750,000 time steps (line: 61).
Evaluation: The training is not distributed across multiple instances (line: 68). The agent is evaluated every 10,000 time steps and logging also occurs after every 10,000 time steps (line: 69-70). During evaluation, only one episode is run (line: 72). Rigorous evaluation is disabled (line: 71).

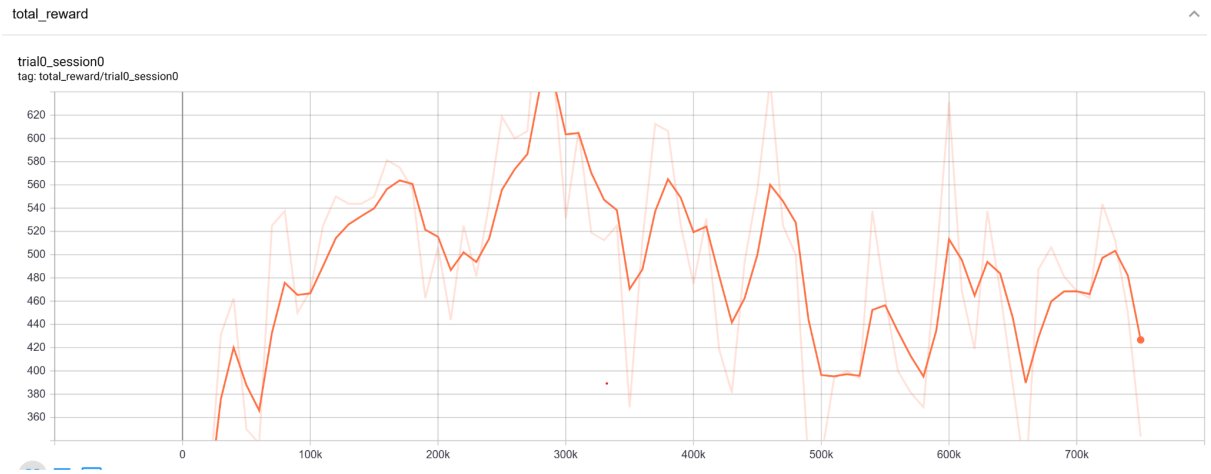


Figure 3: Total reward during DQN training for KungFuMaster-v0

4. DDQN

Algorithm: The algorithm is DDQN (line: 6). The action probability type is argmax and the action policy is epsilon-greedy (line: 7-8) with linear decay (line: 9-15) of the exploration variable specification (epsilon is initialized to 1.0 and annealed to 0.01 between time steps 10,000 and 1,000,000). Gamma (discount factor) is set to 0.99 (line: 11).
Training frequency: Training starts after the agent has made 10,000 steps in the environment (line: 20) and occurs every 32 steps from then on (line: 19). It samples 1 batch from the Replay memory (line : 18) and makes only 1 parameter update (line: 17). Each batch has 32 elements (line: 24). A maximum of 200,000 samples are stored in the Replay memory (line: 25).

Memory: The memory type is *PrioritizedReplay* (line: 23) and has a maximum of 200,000 experiences (line: 27). The prioritization parameter (alpha) has the value of 0.6 (line: 24) and the small constant epsilon is 0.0001 (line: 25).
Network Architecture: The network architecture is a ConvNet (line: 31). It consists of convolutional layers and a fully connected layer. The convolutional layers have the following configuration (line: 32-36):

1. 32 filters, 8 kernel size, 4 stride, 0 padding, 1 dilation
2. 64 filters, 4 kernel size, 2 stride, 0 padding, 1 dilation
3. 32 filters, 3 kernel size, 1 stride, 0 padding, 1 dilation

The fully connected layer has 512 units (line: 37). The activation function used is ReLU (line: 38). The weights are initialized using orthogonal initialization (line: 39). batch normalization is disabled (line: 40). Gradient clipping is set to 10.0 (line: 41). The loss function used is Mean Squared Error (line: 42-44). The optimizer is RMSprop with a learning rate of 0.0001 (line: 45-48). The learning rate is constant throughout, specified by the null learning rate scheduler (line: 49). The parameters are completely replaced with the updated ones after 1,000 steps (line: 50-51). Training on GPU is enabled (line: 52).
Environment: The environment name is Atari KungFuMaster-v0 (line: 56). It consists of frame concatenation operation for 4 consecutive frames, reward scaling using the sign function and it operates in 16 parallel environments (line: 57-60). Training length consists of 750,000 time steps (line: 62).
Evaluation: The training is not distributed across multiple instances (line: 69). The agent is evaluated every 10,000 time steps and logging also occurs after every 10,000 time steps (line: 70-71). During evaluation, only one episode is run (line: 73). Rigorous evaluation is disabled (line: 72).

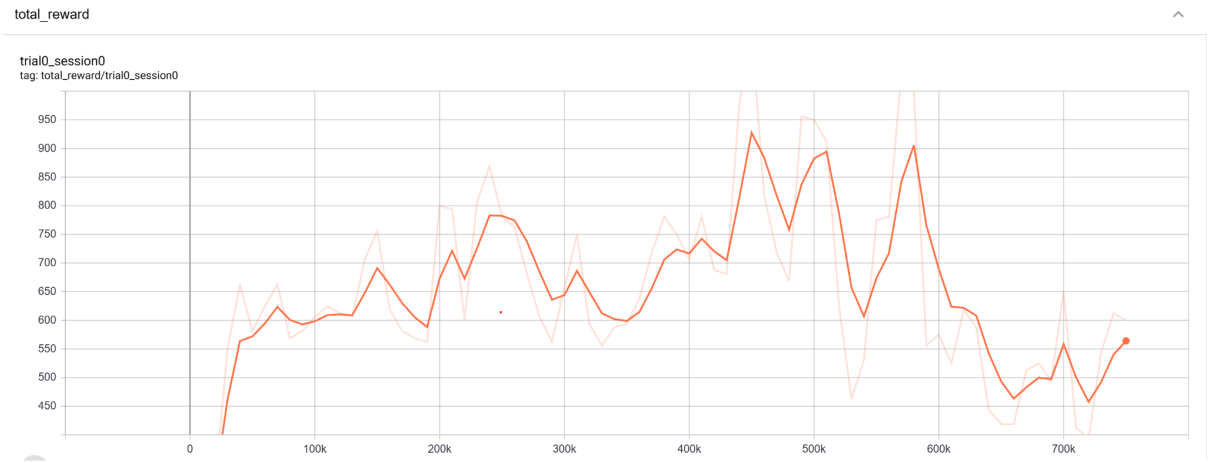


Figure 4: Total reward during DDQN training for KungFuMaster-v0

5. A2C

Algorithm: The algorithm is Actor-Critic (line: 6). The action policy is the default policy (line: 8) for discrete action space (Categorical distribution). Gamma (discount factor) is set to 0.99 (line: 11). The lambda parameter value is set to 0.95 (line: 11). The entropy coefficient and its decay during training are also specified (line: 9-15). The value loss coefficient is set to 0.5 (line: 20).
Training frequency: Training is batch-wise, because we have selected OnPolicyBatchReplay memory (line: 24), and the batch size is 32 controlled by the training_frequency (line: 21) * 16 number of parallel environments (line: 65).
Network Architecture: The network architecture is a ConvNet (line: 27). It consists of convolutional layers and a fully connected layer. The convolutional layers have the following configuration (line: 29-33):

1. 32 filters, 8 kernel size, 4 stride, 0 padding, 1 dilation
2. 64 filters, 4 kernel size, 2 stride, 0 padding, 1 dilation
3. 32 filters, 3 kernel size, 1 stride, 0 padding, 1 dilation

The fully connected layer has 256 units (line: 34). The activation function between layers is ReLU (line: 35). The weights are initialized using orthogonal initialization (line: 36). Normalization is enabled and batch normalization is disabled (line: 37-38). Gradient clipping is set to 0.5 (line: 39). The same optimizer is used differently during training (line: 40). The loss function used is Mean Squared Error (line: 41-43). The optimizer is RMSprop with a learning rate of 0.0007 (line: 44-49). If separated networks are used instead, it is possible to specify a different optimizer setting for the critic network (line: 50-55), by setting use_same_optim to false (line: 40). Since the network is shared in this case, it is not used. There is no learning rate decay (line: 56). Training on GPU is enabled (line: 57).
Environment: The environment name is Atari KungFuMaster-v0 (line: 61). It consists of frame concatenation operation for 4 consecutive frames, reward scaling using the sign function and it operates in 16 parallel environments (line: 62-65). Training length consists of 750,000 time steps (line: 67).
Evaluation: The training is not distributed across multiple instances (line: 74). The agent is evaluated every 10,000 time steps and logging also occurs after every 10,000 time steps (line: 75-76). During evaluation, only one episode is run (line: 78). Rigorous evaluation is disabled (line: 77).

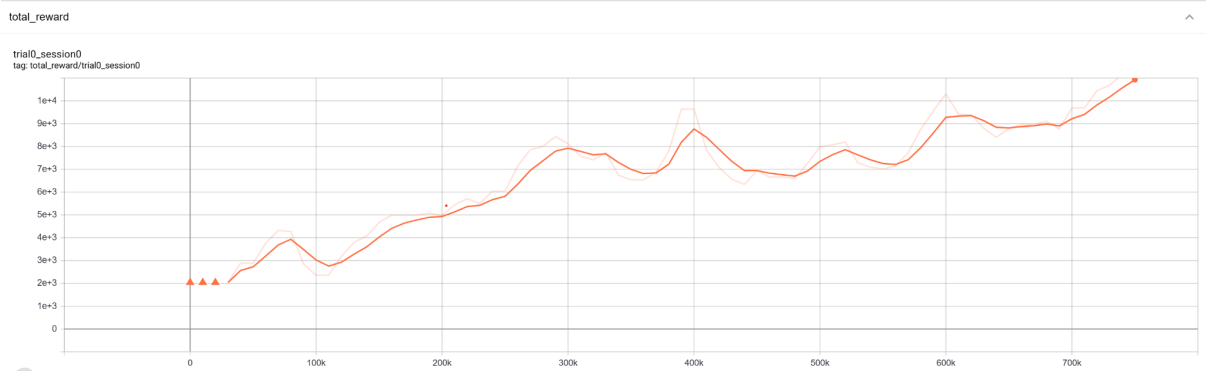


Figure 5: Total reward during A2C training for KungFuMaster-v0

6. PPO

Algorithm: The algorithm is Proximal Policy Optimization (line: 6). The action policy is the default policy (line: 8) for discrete action space (Categorical distribution). Gamma (discount factor) is set to 0.99 (line: 10). GAE is used to estimate advantages, with lam value set to 0.70 (line: 11). The clipping hyperparameter epsilon and its decay are specified (line: 12-18), and the entropy coefficient is similarly specified (line: 19:25). The value loss coefficient is set to 0.5 (line: 26).

Training frequency: Training is batch-wise, because we have selected OnPolicyBatchReplay memory (line: 32). The mini batch size is 256 (line: 28) and the number of training epochs is 4 (line: 29). The time horizon T is 128 (line: 27).

Network Architecture: The network architecture is a ConvNet (line: 35). It consists of convolutional layers and a fully connected layer. The convolutional layers have the following configuration (line: 37-41):

1. 32 filters, 8 kernel size, 4 stride, 0 padding, 1 dilation
2. 64 filters, 4 kernel size, 2 stride, 0 padding, 1 dilation
3. 32 filters, 3 kernel size, 1 stride, 0 padding, 1 dilation

The fully connected layer has 512 units (line: 42). The activation function between layers is ReLU (line: 43). The weights are initialized using orthogonal initialization (line: 44). Normalization is enabled and batch normalization is disabled (line: 45-46). Gradient clipping is set to 0.5 (line: 47). The same optimizer is used differently during training (line: 48). The loss function used is Mean Squared Error (line: 49-51). The actor and critic use a shared network (line: 36). The optimizer is Adam, with a learning rate of 0.0001 (line: 52-55). The learning rate has decayed to 0 over 10 million frames (line: 60-63).

Environment: The environment name is Atari KungFuMaster-v0 (line: 68). It consists of frame concatenation operation for 4 consecutive frames, reward scaling using the sign function and it operates in 16 parallel environments (line: 70-73). Training length consists of 750,000 time steps (line: 74).

Evaluation: The training is not distributed across multiple instances (line: 81). Rigorous evaluation is disabled (line: 82). During evaluation, only one episode is run (line: 83). The agent is evaluated every 10,000 time steps and logging also occurs after every 10,000 time steps (line: 86-87).

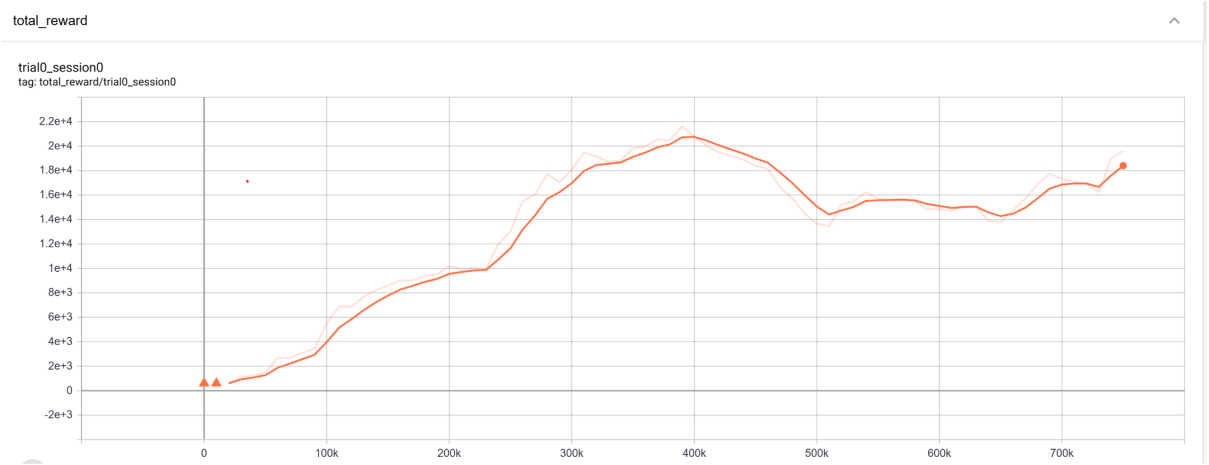


Figure 6: Total reward during PPO training for KungFuMaster-v0

7. A3C

Algorithm: The algorithm used is Asynchronous Advantage Actor-Critic (line: 6). The action policy and exploration variance specifications are set to default (line: 8-9). Gamma (discount factor) is set to 0.99 (line: 10). The lambda parameter (lam) for GAE is set to 0.95 (line: 11). The entropy coefficient is specified (line: 13:19). The value loss coefficient is set to 0.5 (line: 26).
Training frequency: Training is batch-wise, because we have selected OnPolicyBatchReplay memory (line: 24), and the batch size is 32 controlled by the training_frequency (line: 21) * 16 number of parallel environments (line: 65).
Network Architecture: The network architecture is a ConvNet (line: 27). It consists of convolutional layers and a fully connected layer. The convolutional layers have the following configuration (line: 29-33):

1. 32 filters, 8 kernel size, 4 stride, 0 padding, 1 dilation
2. 64 filters, 4 kernel size, 2 stride, 0 padding, 1 dilation
3. 32 filters, 3 kernel size, 1 stride, 0 padding, 1 dilation

The fully connected layer has 512 units (line: 34). The activation function between layers is ReLU (line: 35). The weights are initialized using orthogonal initialization (line: 36). Normalization is enabled and batch normalization is disabled (line: 37-38). Gradient clipping is set to 0.5 (line: 39). The same optimizer is used differently during training (line: 40). The loss function used is Mean Squared Error (line: 41-43). The optimizer is GlobalAdam with a learning rate of 0.0001 (line: 44-47). If separated networks are used instead, it

is possible to specify a different optimizer setting for the critic network (line: 48-51). There is no learning rate decay (line: 52). Training on GPU is enabled (line: 53).

Environment: The environment name is Atari KungFuMaster-v0 (line: 57)**.** It consists of frame concatenation operation for 4 consecutive frames, reward scaling using the sign function and it operates in 16 parallel environments (line: 58-61). Training length consists of 750,000 time steps (line: 63).

Evaluation: The training is performed in a synchronous manner (line: 70). The agent is evaluated every 10,000 time steps and logging also occurs after every 10,000 time steps (line: 71-72). Rigorous evaluation is disabled (line: 73). During evaluation, only one episode is run (line: 74).

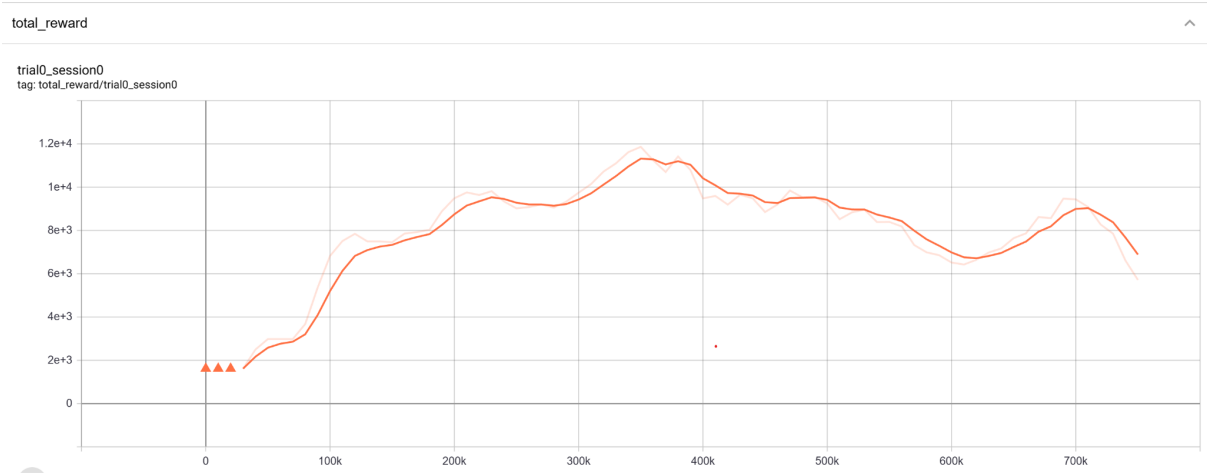


Figure 7: Total reward during A3C training for KungFuMaster-v0

Conclusions

Among the algorithms discussed, PPO stands out as the best performer, achieving a total reward close to 22,000. This indicates that PPO was able to learn effective strategies and perform well in the Atari KungFuMaster-v0 environment. The graph shows a consistent increase in reward over time, suggesting that further training may lead to even higher rewards.

A2C and A3C also demonstrated promising performance, with their final rewards ranging from 12,000 to 15,000. Both algorithms show potential for improvement with additional training. The graph implies that given more training time, A2C could achieve even better rewards.

SARSA, although not as successful as PPO, A2C and A3C, still performed decently with a reward of approximately 3,300. The graph indicates that training SARSA for a longer duration would likely yield better results.

On the other hand, REINFORCE, and the off-policy algorithms (DQN and DDQN) fared poorly in terms of reward, all falling below 1,000. The reward curves for

these algorithms exhibit significant fluctuations, suggesting that they struggled to converge to optimal policies in the given environment.

Full name: Glanda Darie-Teofil
Master: Machine Learning