Reinforcement Learning Project: Continuous Cartpole

Done by:

Simon Ging, Abdelrahman Younes, Naya Baslan

Supervised by:

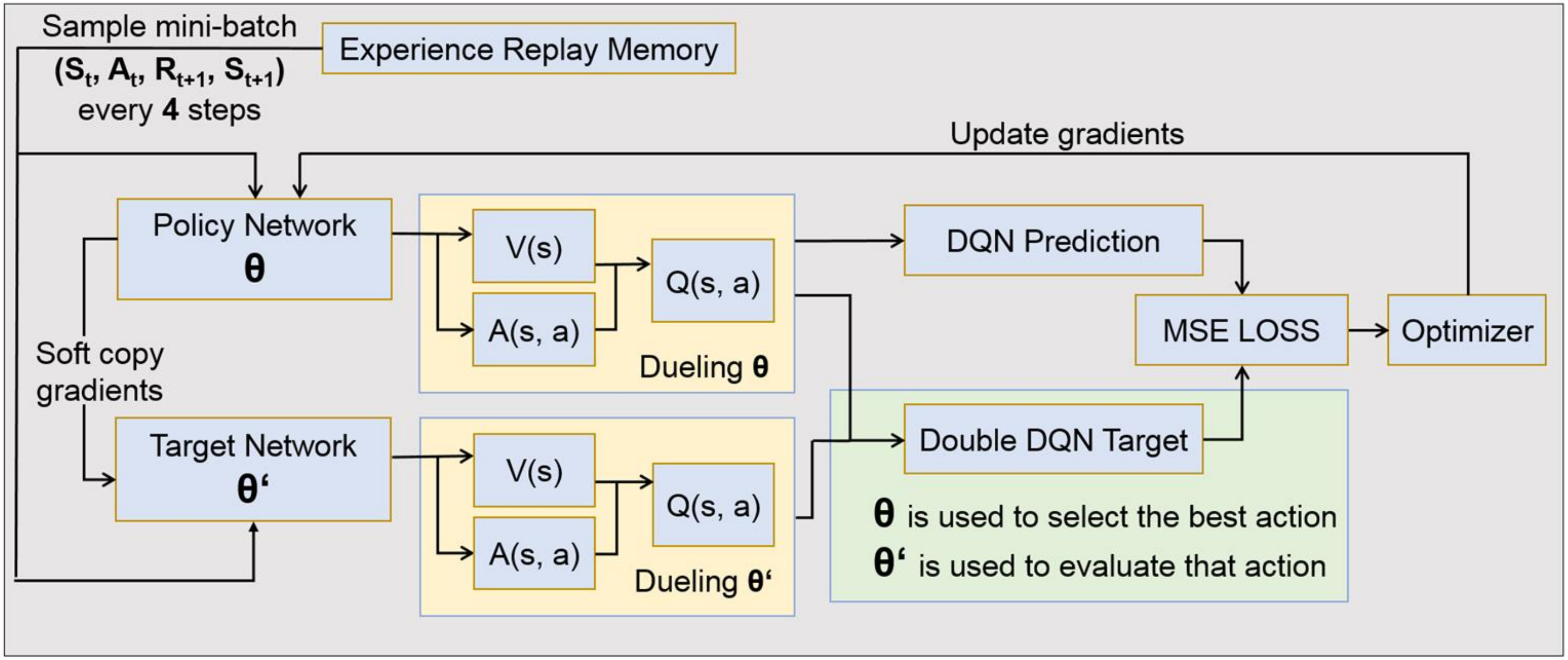
Neurorobotics Lab, Uni Freiburg

Agenda

- **Overview**
  - Choice of Algorithm
  - Performance analysis
- **Details**
  - Performance visualization
  - Project process
  - Algorithm details and improvements
  - Reward function analysis
  - Success metrics discussion
  - Visual comparison of result differences
  - Epsilon scheduler analysis
  - Runtime Analysis
  - HPO configuration space
  - All results in one plot
  - Best agent showcase
  - Qualitative evaluation of 2 other environments
- **Outlook**
- **References**

Choice of Implemented Algorithm

- Choice: DQN[1] (discrete action space)
  – Popular algorithm, can solve Atari games
  – Lots of possible improvements[3][4][5]
  – Agents with discretized actions can solve given problem



DQN Target: $Y_t^{\text{DoubleQ}} \equiv R_{t+1} + \gamma Q(S_{t+1}, \operatorname{argmax} Q(S_{t+1}, a; \boldsymbol{\theta}_t); \boldsymbol{\theta}'_t)$

Figure 1: Schematic diagram of DQN Algorithm[1] with Double Q Learning[3] and Dueling Networks[4]

Performance Analysis

Experiment	K steps	Episodes	Evaluation (1000 ep)	Train score (last 100 ep)	Success metric	Converged / Total
Baseline	688.07±161.59	2851.13±714.40	303.42±177.98	191.71±126.34	Eval ≥ 400 (every 20K steps)	10/15
Best (eval @5K)	**189.02±28.41**	**879.73±80.56**	414.96±18.02	59.81±45.44	Eval ≥ 400 (every 5K steps)	15/15
Best (eval @20K)	197.41±35.29	893.93±110.19	414.85±12.16	81.16±57.91	Eval ≥ 400 (every 20K steps)	15/15
Best (train score)	512.08±162.60	1521.27±210.33	**420.25±3.10**	**400.42±0.75**	Score of last 100 episodes ≥ 400	14/15
Best (shaped reward)	256.03±54.99	1027.00±163.02	409.688±40.87	13.23K±71.80K	Eval ≥ 400 (every 5K steps)	15/15

Table 1: Quantitative Analysis of 5 selected experiments

- Careful HP optimization improved the results
- Different success metric have their advantages
- High variance, repeated experiments needed
- Highly time efficient
  - **~5min** GPU runtime for experiment *Best (eval @5K)*

Performance visualization

Baseline improved significantly, eval success metric outperforms train metric

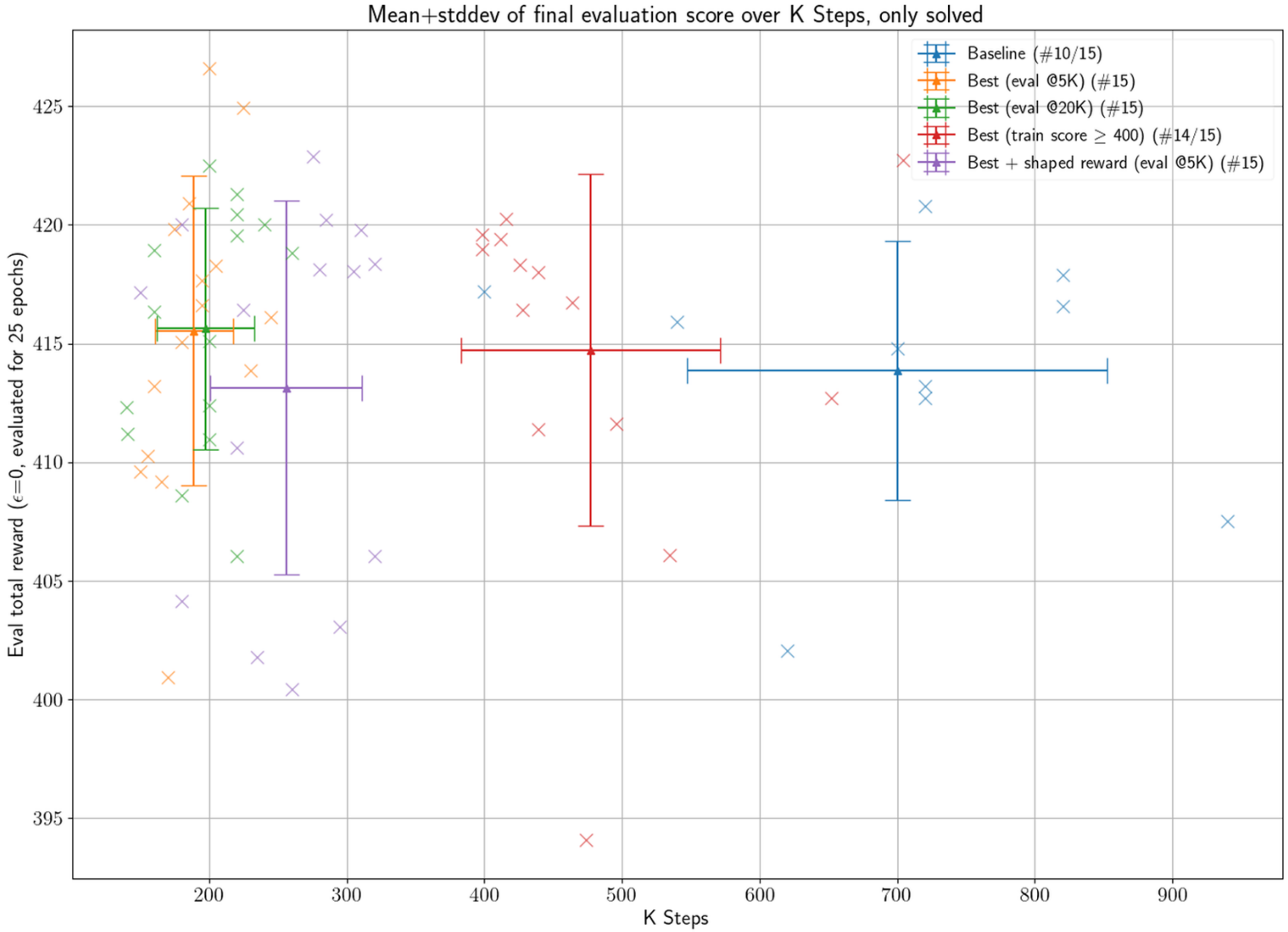


Figure 2: Convergence speed comparison of 5 selected experiments

Project Process Details

- Baseline: DQN with local and target network[1]
  - Using soft target network update[2]
  - Converges after ~700k steps, but unstable
- Experiment setup
  - High variance: At least 5 runs per experiment
  - Default success metric: Eval score ≥ 400
    - ϵ=0, 25 episodes, evaluated every 20K steps
- Algorithm improvements
  - Double Q Learning[3]
  - Dueling Network Architecture[4]
- Local search for better hyperparameters
- In-depth evaluation

DQN Details[1]

- Experiences are stored in a buffer and randomly sampled
- Uses two Networks
  - Online net predicts state-action-values $y^O = Q(s_t, a_t \mid \Phi^O)$
  - Target net predicts targets for TD Error with weights Φ^T
- Every K=4 steps:
  - Update Φ^O with gradient of MSELoss(prediction, target)
  - Soft update[2] $\Phi^T = (1 - \tau)\,\Phi^T + \tau\,\Phi^O$
- Act ϵ-greedily w.r.t $Q(s_t, a_t \mid \Phi^O)$
  - Decay ϵ during training, evaluate with low or zero ϵ

DQN Improvements - Double Q Learning

- DQN loss[1]

$$L_i(\theta_i) = \mathbb{E}_{s,a,s',r\sim D}\left(\underbrace{r + \gamma \max_{a'} Q(s', a'; \theta_i^-)}_{\text{target}} - Q(s,a;\theta_i)\right)^2$$

- DQN target[3]

$$Y_t^{\text{DQN}} \equiv R_{t+1} + \gamma \max_a Q(S_{t+1}, a; \boldsymbol{\theta}_t^-)$$

  – Target network both chooses and evaluates target action

- Double Q-learning[3] target
  – Decouple action selection from evaluation

$$Y_t^{\text{DoubleQ}} \equiv R_{t+1} + \gamma Q(S_{t+1}, \operatorname*{argmax}_a Q(S_{t+1}, a; \boldsymbol{\theta}_t); \boldsymbol{\theta}_t')$$

  – Choose best action for target with online network
  – Estimate its value with target network

DQN Improvements - Dueling Networks

- DQN[1]: predict Q(s, a) directly
- Dueling Networks Architecture[4]:
  - Predict state-value V and advantage function A:

$$A^{\pi}(s,a) = Q^{\pi}(s,a) - V^{\pi}(s)$$

  - Compute Q values:

$$Q(s,a;\theta,\alpha,\beta) = V(s;\theta,\beta) + \left(A(s,a;\theta,\alpha) - \frac{1}{|\mathcal{A}|} \sum_{a'} A(s,a';\theta,\alpha) \right)$$

  - Network graphs with image input
  - We replaced convs with fully-connected layers
  - We use a single hidden layer instead one for each V and A

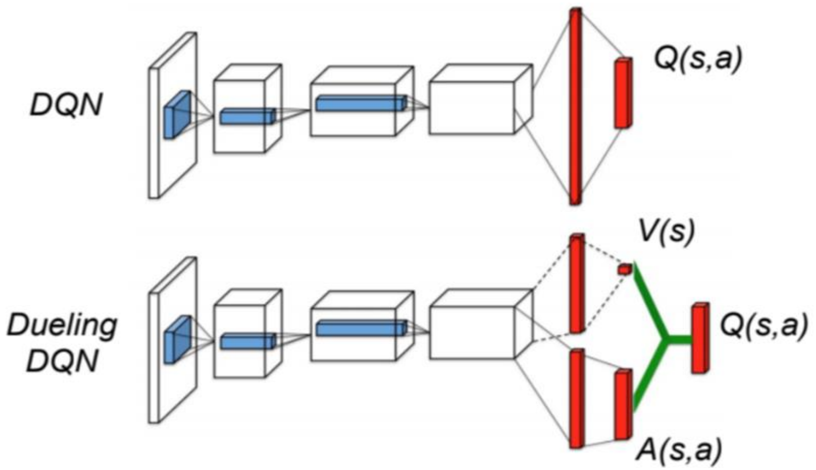


Figure 3: DQN vs Dueling Networks[4]

Reward Function Analysis

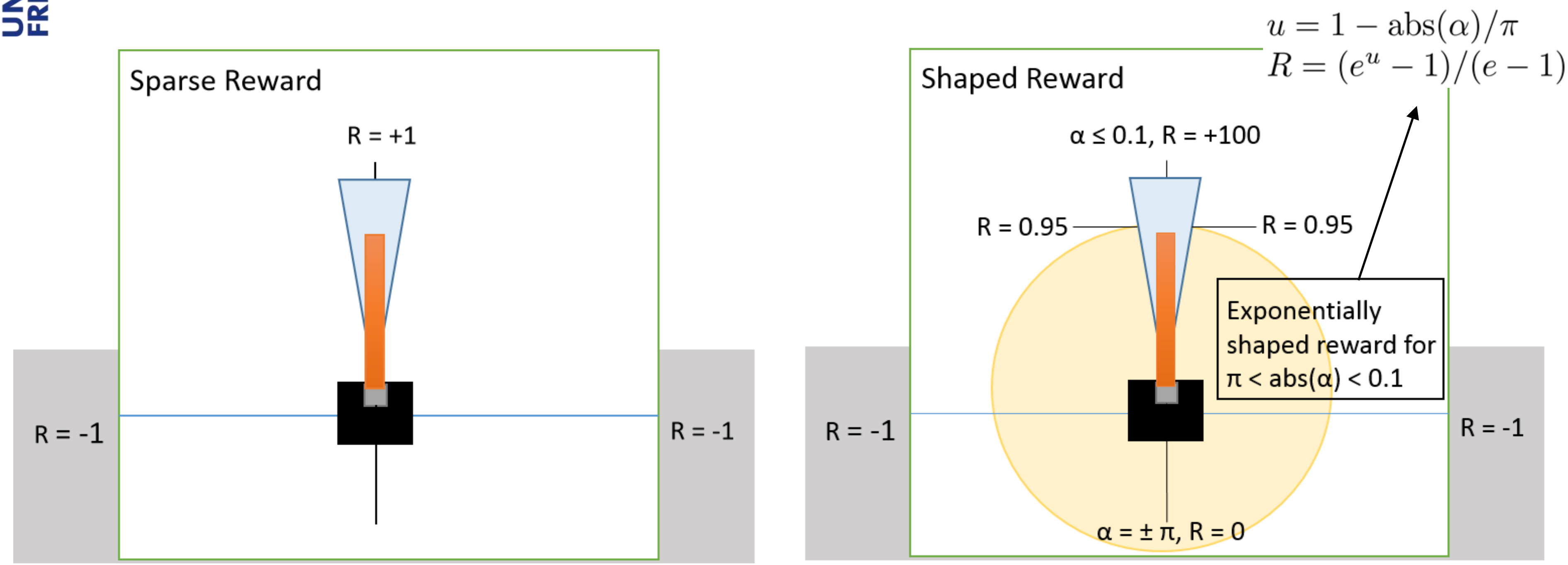


Figure 4: Visualization of sparse and shaped reward

- Environment has sparse rewards
- Shaped reward tested as above
  - Agent performance dropped significantly
- Final configuration uses sparse reward

Different success metrics

- High evaluation result was our goal
  – Evaluate every 20k (or 5k) steps for 25 episodes
  – Mean score ≥ 400: Success
  – Sparse reward - independent of shaped reward choice
  – $\epsilon = 0$ - independent of exploration strategy
  – Overfits the agent to the problem
- Train metric for comparison
  – Mean train reward of last 100 episodes ≥ 400: Success
  – Must be tuned for shaped reward
  – Harder to achieve / can be unstable
  – Probably needs more fine-tuning of ϵ-decay
  – Very dependent on optimization process
  – Agents obtained this way are performing slightly better

Best config, eval metric, evaluation score

Algorithm is stable, all 15 runs converged after 250K steps

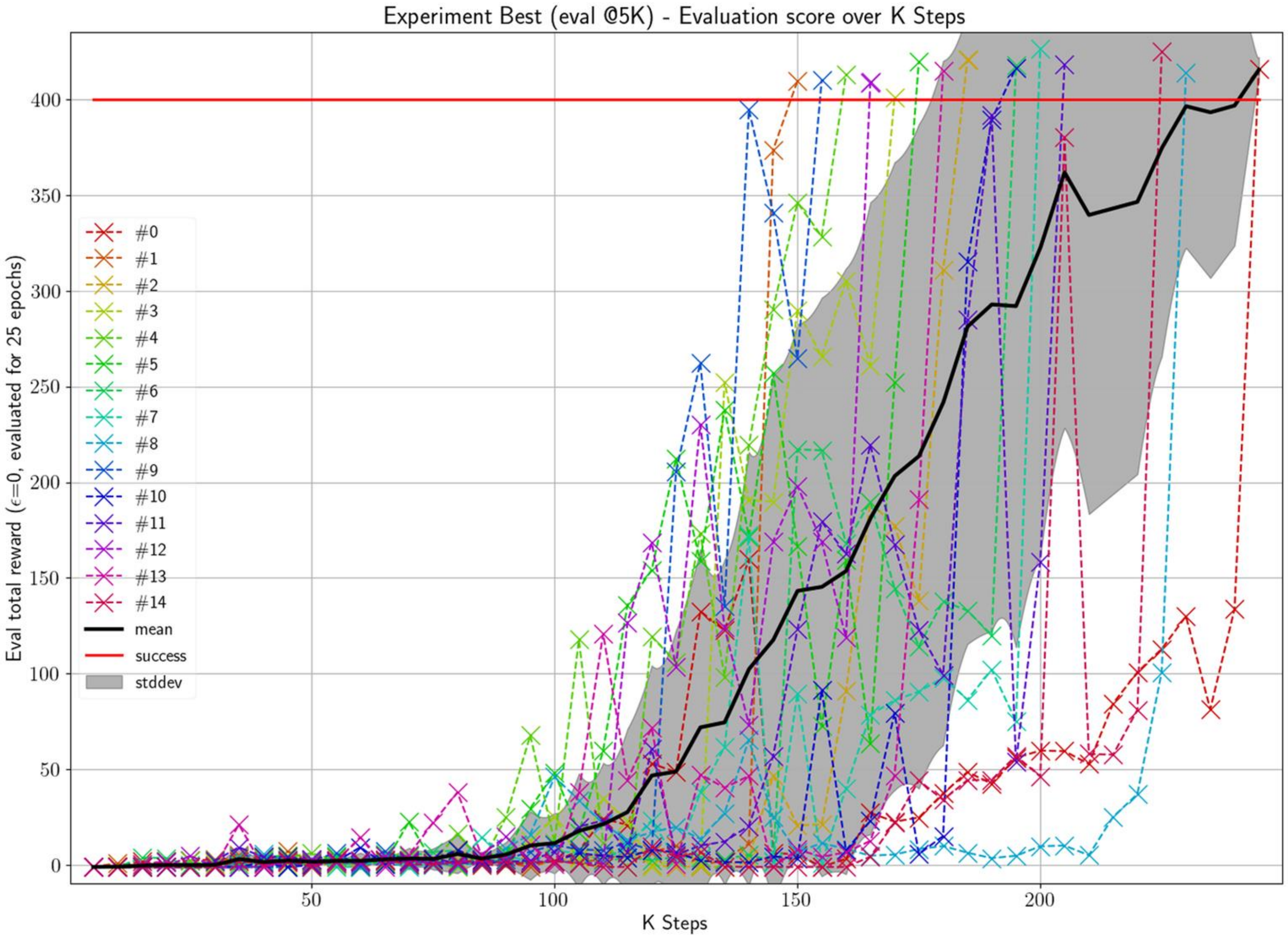


Figure 5: Evaluation curve of best config with eval metric

Best config, train metric, evaluation score

Train score metric is unstable. Too high ϵ?

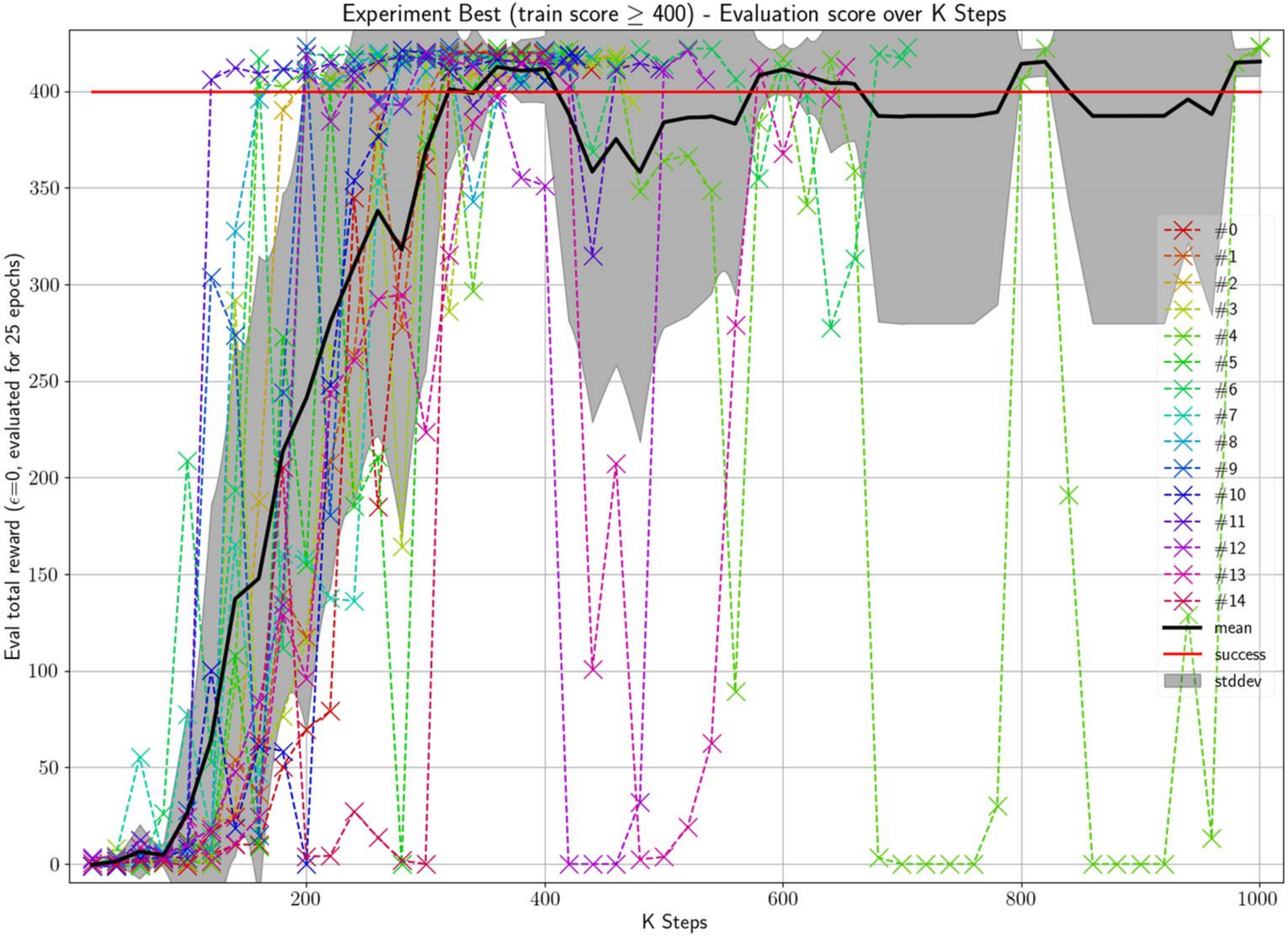


Figure 6: Evaluation curve of best config with train metric

Best config, eval metric, training score

Agents converge before training score is high

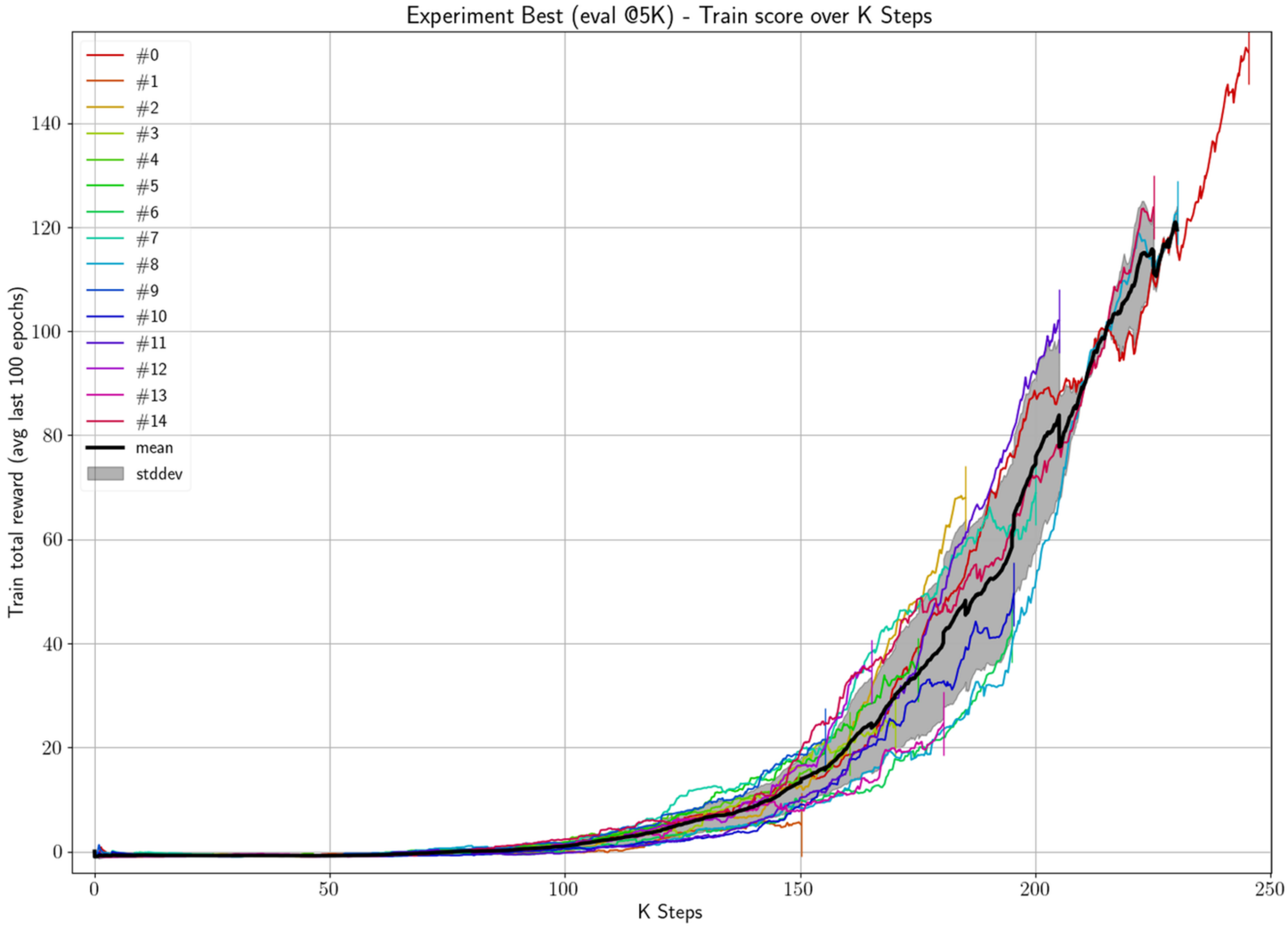


Figure 7: Training curve of best config with eval metric

Best config, train metric, training score

Reaching ≥ 400 train reward over last 100 episodes seems difficult

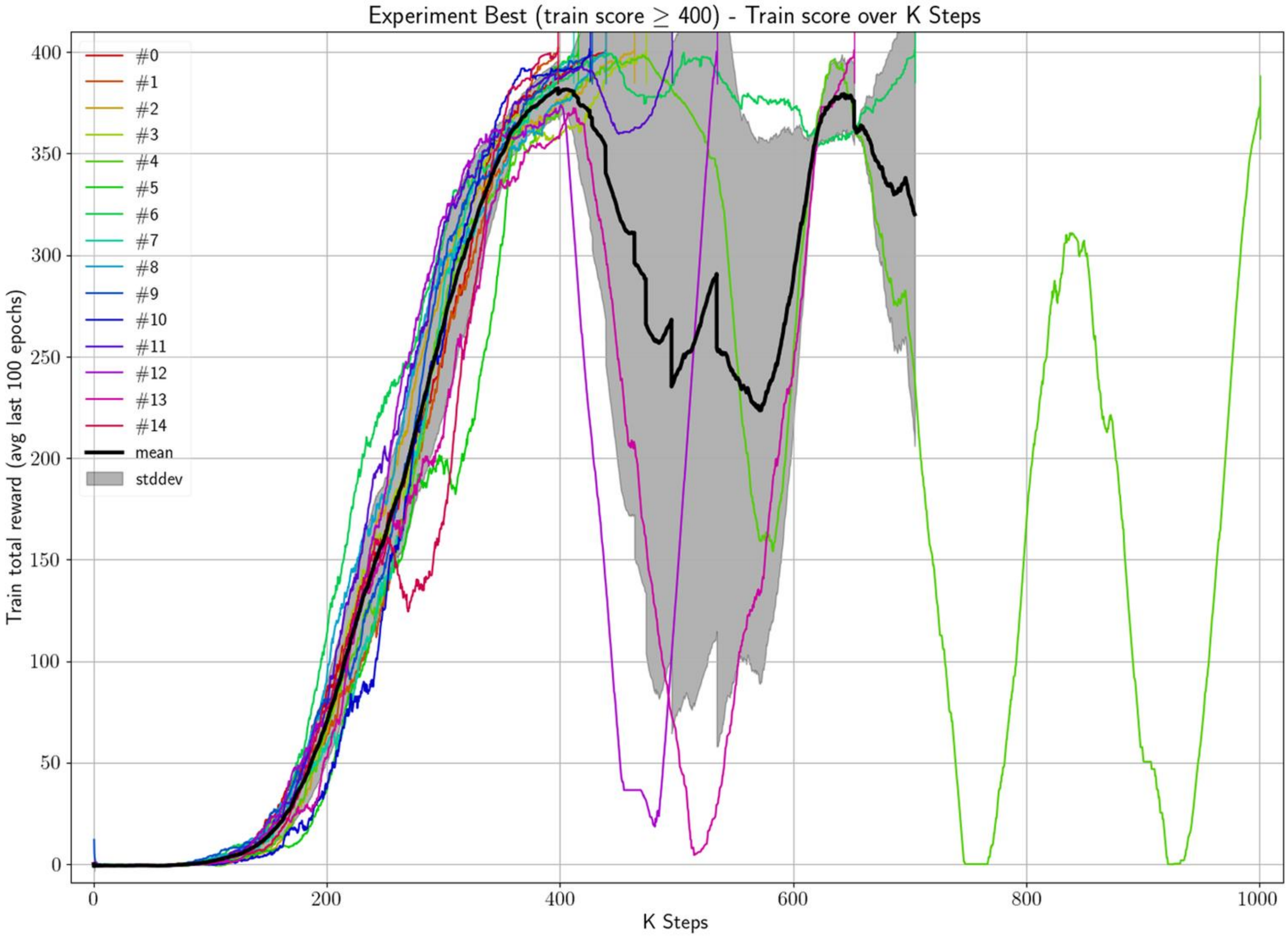


Figure 8: Training curve of best config with train metric

Best config, eval metric, quality test

Agents are very good but sometimes still fail (tested for 1000 episodes)

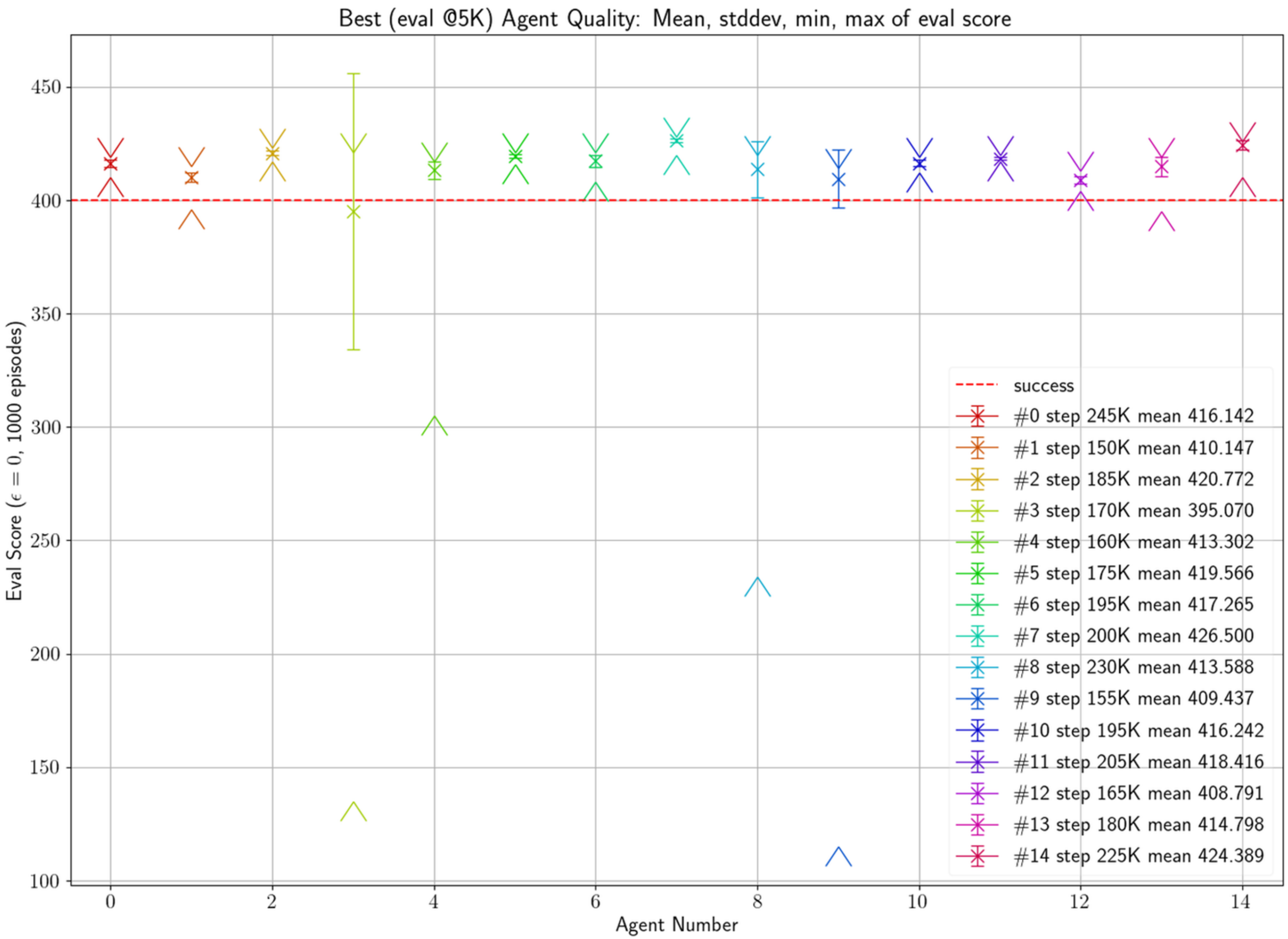


Figure 9: Quality test of best config with eval metric

Best config, train metric, quality test

Train metric is unstable and slower but provides much better agents

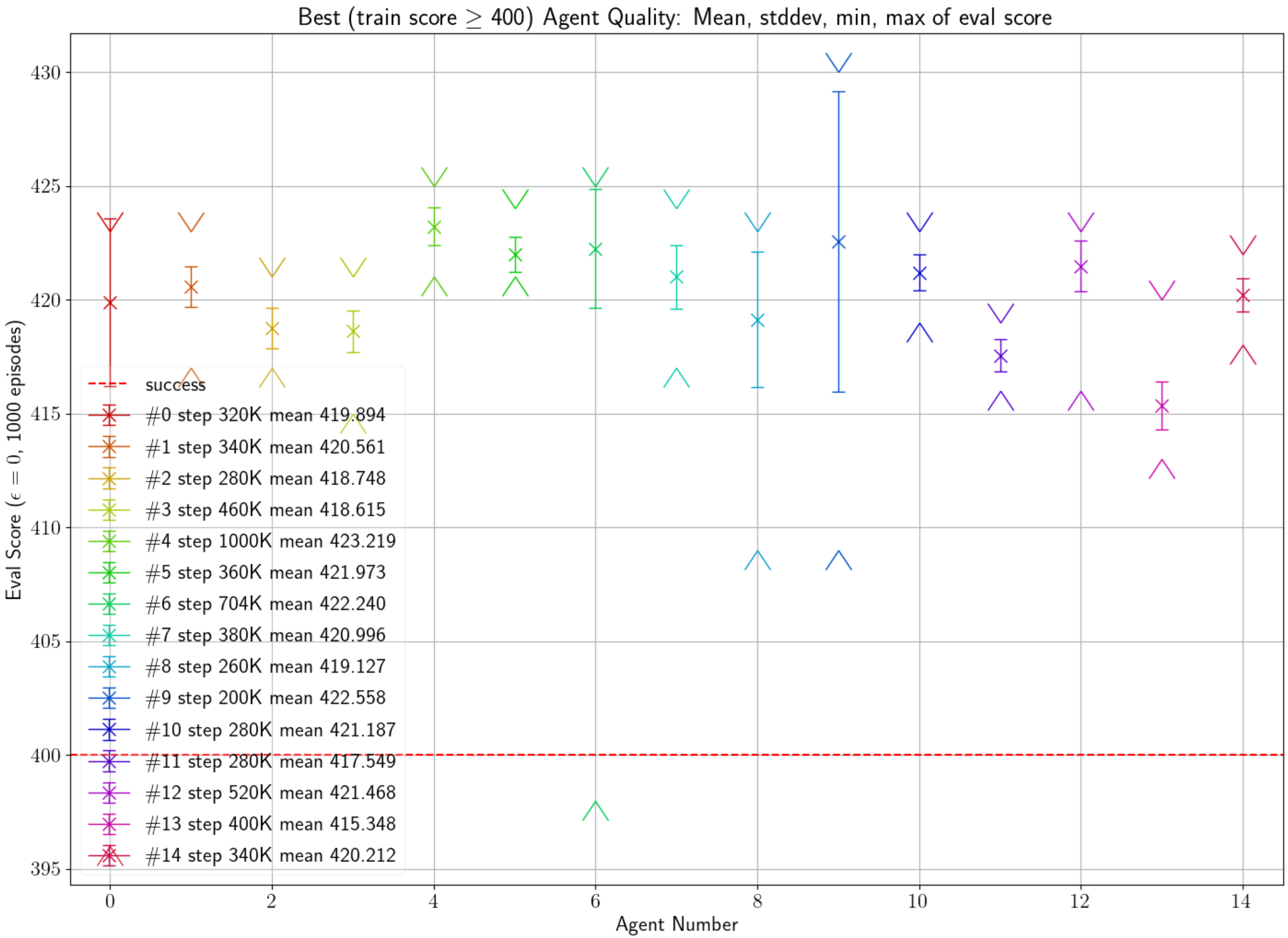


Figure 10: Quality test of best config with train metric

Epsilon decay scheduler

Agents are already converging at high ϵ values

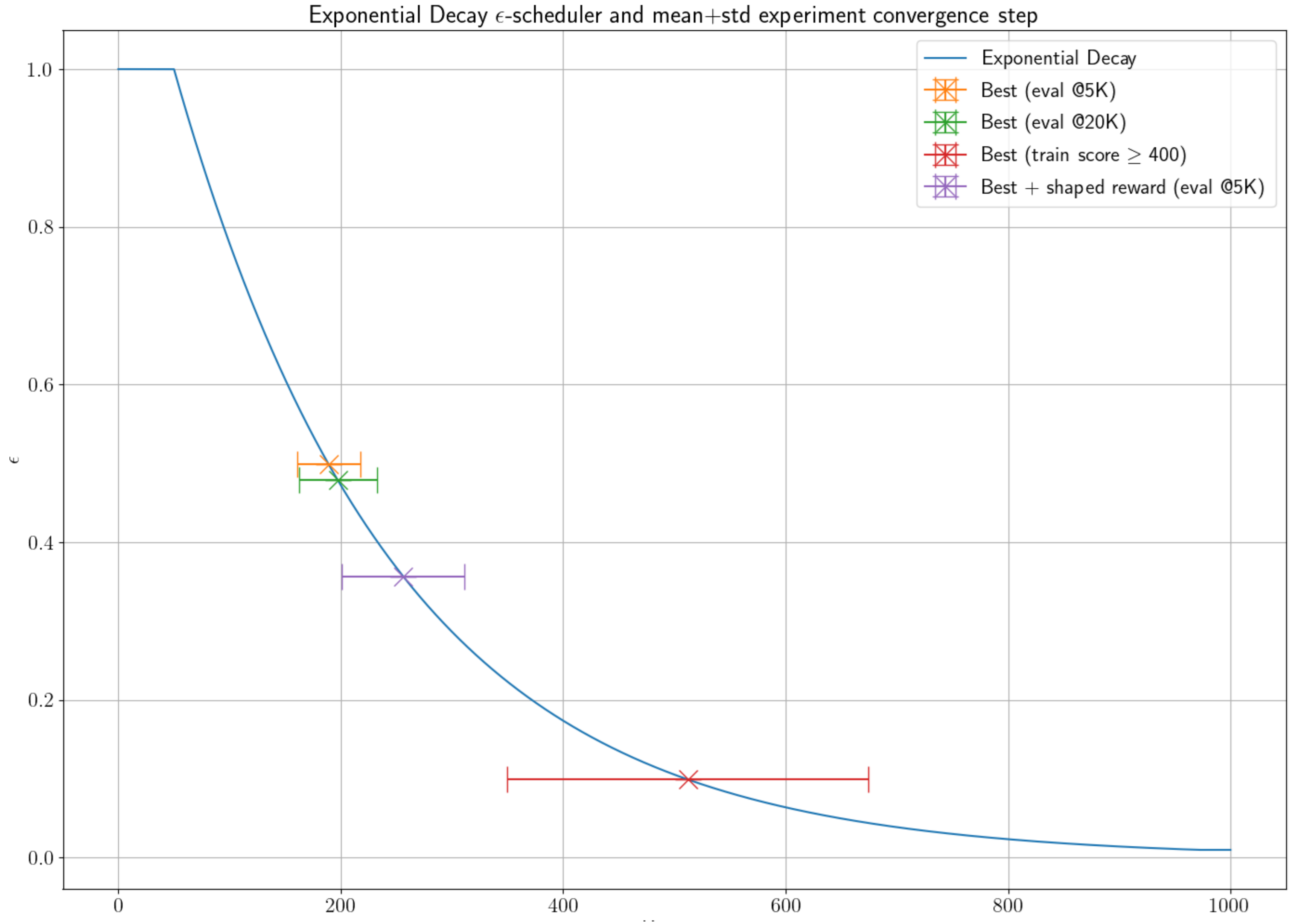


Figure 11: Convergence speeds and epsilon values of 4 selected experiments

Runtime Analysis

Experiment	K steps	Episodes	Approx CPU Time	Approx GPU Time	Success metric	Converged / Total
Baseline	688.07±161.59	2851.13±714.40	799.62±187.79	636.84±149.56	Eval ≥ 400 (every 20K steps)	10/15
Best (eval @5K)	**189.02±28.41**	**879.73±80.56**	**585.58±88.03**	**284.59±42.78**	Eval ≥ 400 (every 5K steps)	15/15
Best (eval @20K)	197.41±35.29	893.93±110.19	606.49±108.42	297.52±53.19	Eval ≥ 400 (every 20K steps)	15/15
Best (train score)	512.08±162.60	1521.27±210.33	1584.34±503.07	758.81±240.94	Score of last 100 episodes ≥ 400	14/15
Best (shaped reward)	256.03±54.99	1027.00±163.02	720.96±154.84	326.83±70.19	Eval ≥ 400 (every 5K steps)	15/15

Table 2: Runtime Analysis of 5 selected experiments

- Runtime approximation process:
  - Run each experiment 5 times for 10000 steps
  - Calculate mean time per step
  - Multiply K Steps with mean time per step

Configuration space for HPO

<u>Underlined</u> values: Baseline

Bold values: best configuration (lowest steps to convergence)

- Success metric: <u>Eval every 20k steps</u>, **eval every 5k steps**, avg train score last 100 ep
- Quantizer: how many actions (**2**, 3, 5, <u>16</u>)
- Batch size (<u>64</u>, 128, **256**, 512)
- Discount: <u>**0.99**</u>
- Reward: <u>**sparse**</u>, shaped
- Optimizer/LR: <u>**Adam/5e-4**</u>, SGD/1e-2, RMSProp/1e-2, AdamW/5e-4
- Epsilon scheduler (first 5% steps ϵ=1): <u>**Exponential Decay**</u>, Cosine Decay
- DQN: (<u>baseline</u>, double Q, **dueling**, both)
- Tau (soft update): <u>1e-3</u>, **2e-3**
- Replay Buffer Size: <u>100000</u>, **500000**
- Nonlinearity: <u>**ReLU**</u>, SELU, LeakyReLU
- Regularization: Batchnorm, <u>**None**</u>
- Network Linear Layers: <u>[64, 64]</u>, **[256, 256]**
- State Space Transformation: <u>None</u>, **sincos transform on pole angle**

All experiments (only solved runs)

Experiments sorted by mean convergence step

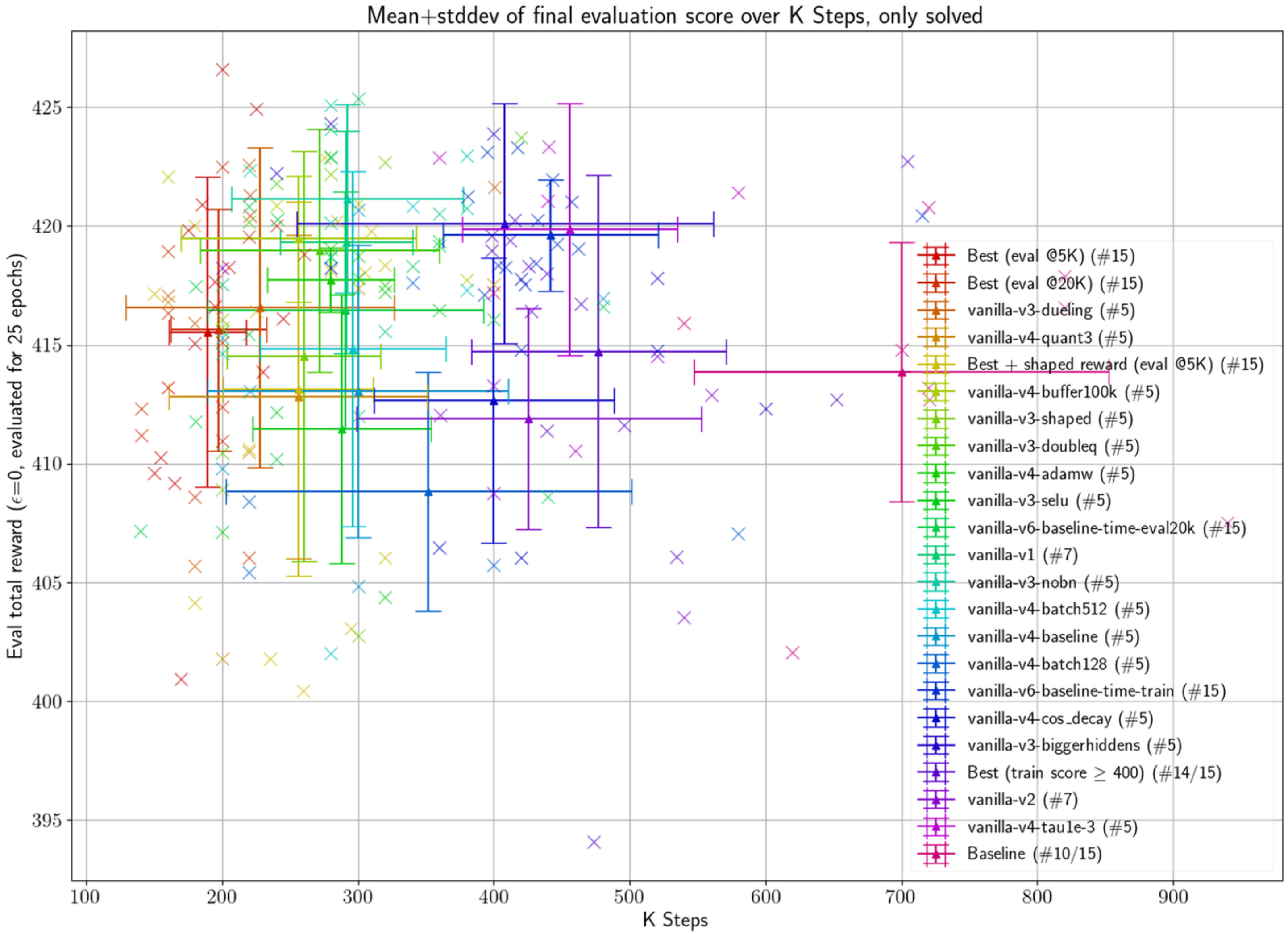


Figure 12: Convergence speed of all experiments (only converged runs)

Best agent over all experiments

- Our best single agent has a reward of 426.5±0.84
  - Tested over 1000 episodes
  - See _*media/agent.mp4* for the video

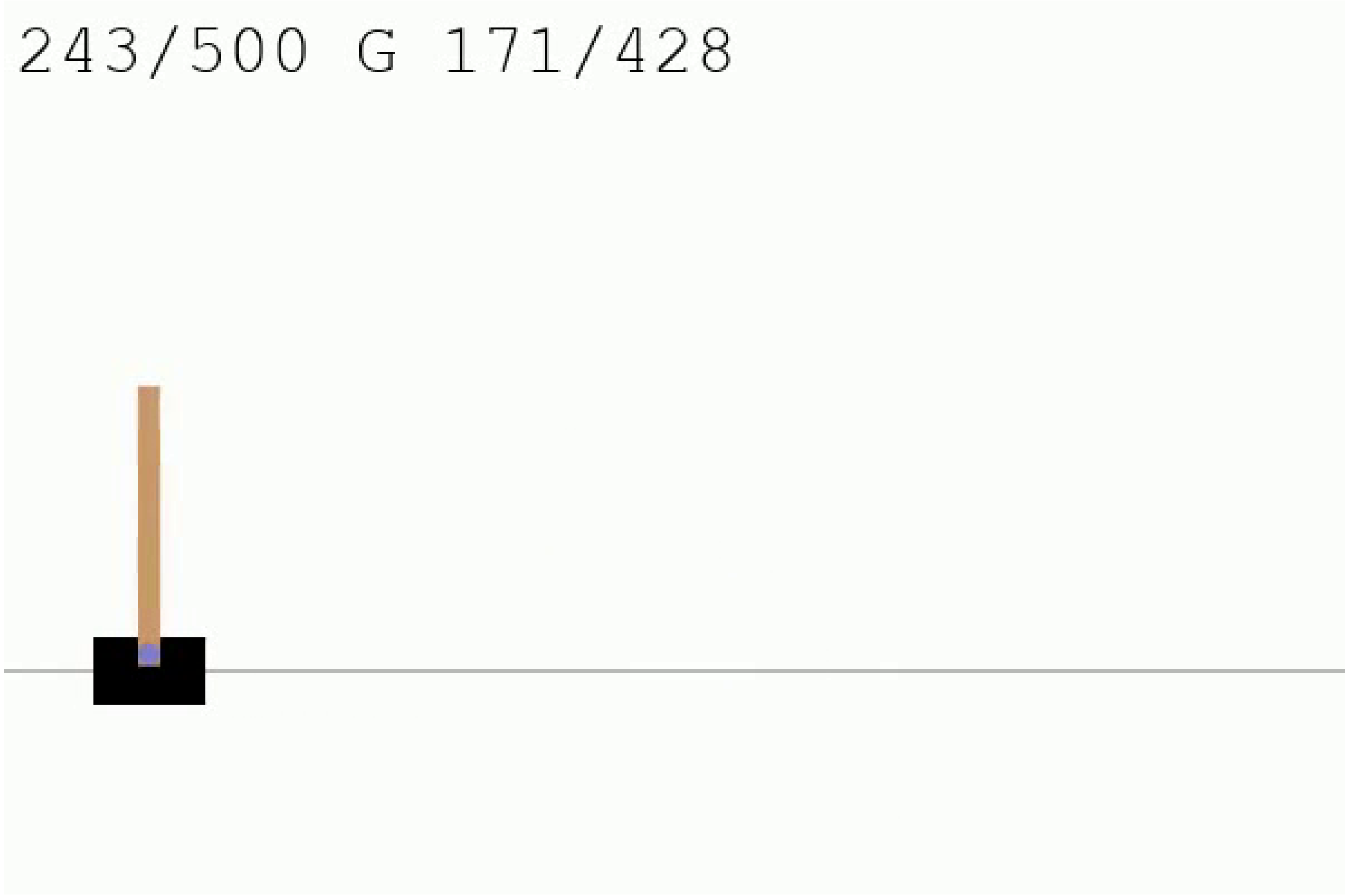


Figure 13: Best Agent in the Continuous Cartpole environment

Qualitative analysis on other environments #1

- Our algorithm solves the Lunar Lander problem
  - 520K steps (~2400 episodes)
  - See *_media/lander.mp4* for the video

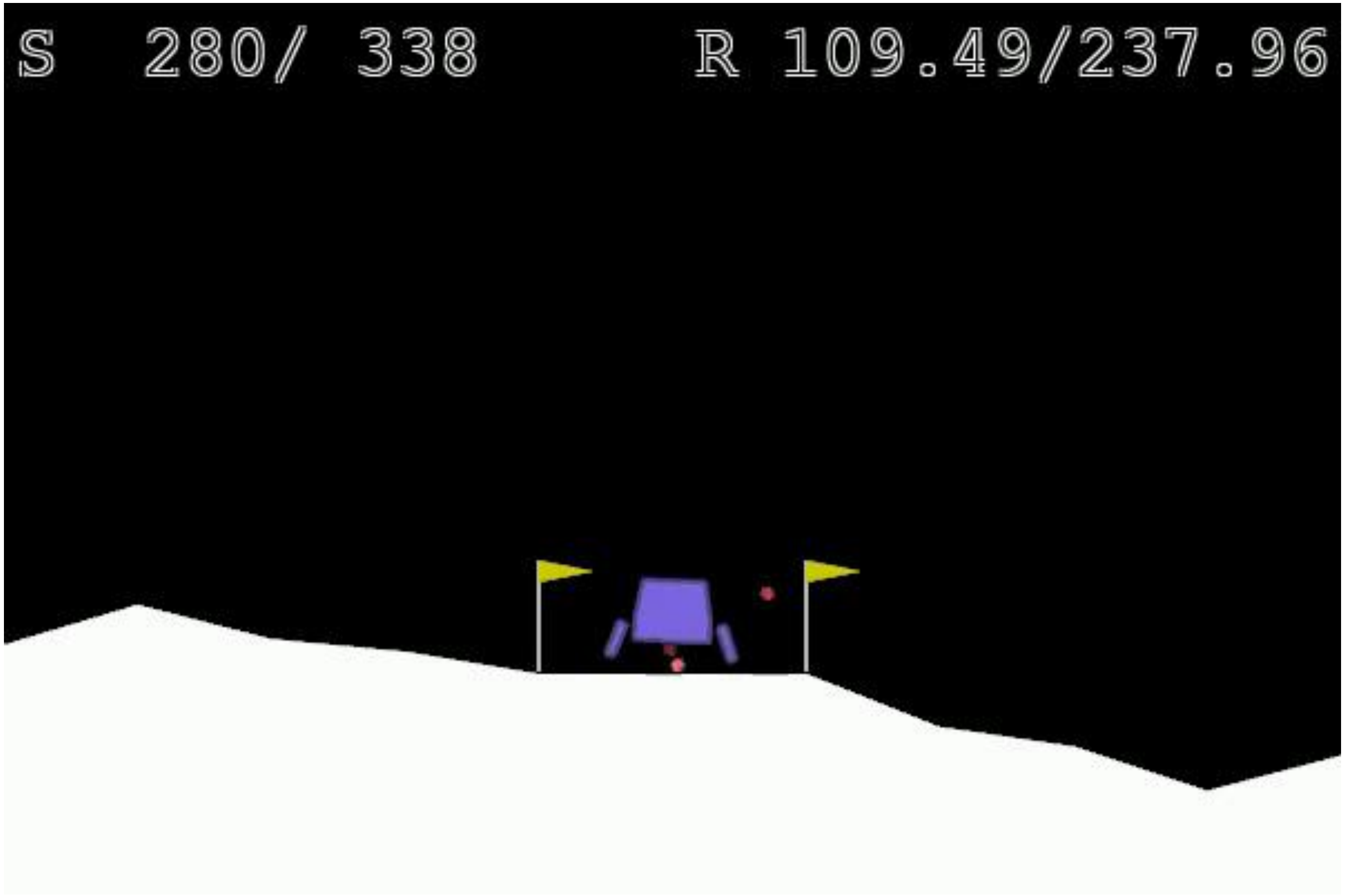


Figure 14: DQN Agent in the Lunar Lander environment

Qualitative analysis on other environments #2

- Bipedal Walker is not solved after 1M steps
  - More punishment for applying stronger torque to the joints
  - Action space is 4-dimensional and continuous
  - Discretization is simply a bad strategy here
  - See _*media/walker.mp4* for the video

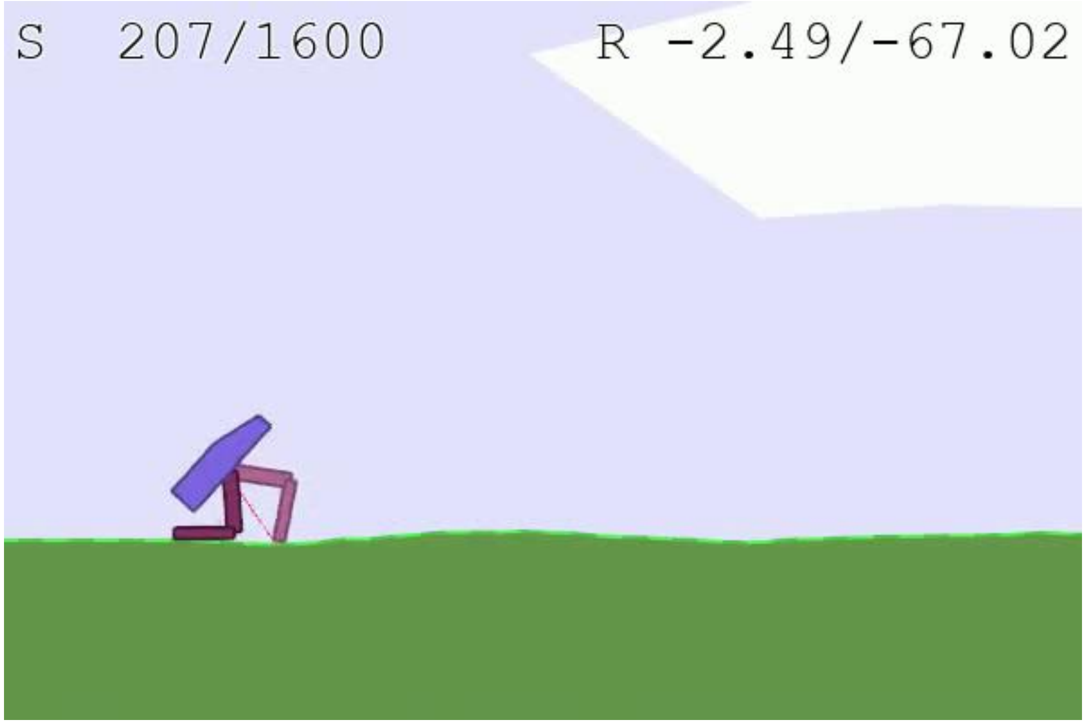


Figure 15: DQN Agent in the Bipedal Walker environment

Outlook, possible improvements

- Rainbow DQN[5] methods to improve the algorithm
- Improve epsilon decay schedule
  - Good for baseline, possibly too high ϵ for better configs
  - Try Reward Based Epsilon Decay[6]
- Stabilize the train success metric experiments
  - Possibly related to the epsilon scheduler
  - Optimizer tuning / LR decay may be helpful
- Improve the eval success metric experiments
  - Use exponential moving average of weights
  - Increase time spent evaluating to get more accurate
- Automated HP search (e.g. Bayesian Optimization)
- Test more environments (Atari, classic control, ...)
- Compare results to other algorithms

References

- [1] Mnih, V. et al (2015), Human Level Control Through Deep Reinforcement Learning
https://web.stanford.edu/class/psych209/Readings/MnihEtAlHassibis15NatureControlDeepRL.pdf
- [2] Lillicrap, T. et al (2015), Continuous control with deep reinforcement learning
https://arxiv.org/abs/1509.02971
- [3] van Hasselt, H. et al (2016), Deep Reinforcement Learning with Double Q-learning
https://arxiv.org/abs/1509.06461
- [4] Wang, Z. et al (2015), Dueling Network Architectures for Deep Reinforcement Learning
https://arxiv.org/abs/1511.06581
- [5] Hessel, M. et al (2017), Rainbow: Combining Improvements in Deep Reinforcement Learning
https://arxiv.org/abs/1710.02298
- [6] Maroti, A., RBED: Reward Based Epsilon Decay
https://arxiv.org/abs/1910.13701