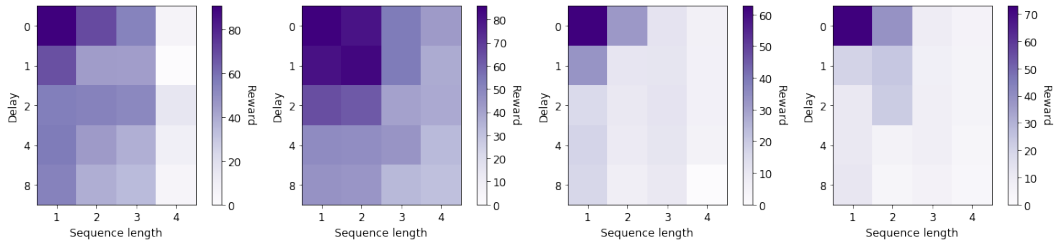
# A Additional Reward Plots




(a) DQN (b) Rainbow (c) A3C (d) A3C + LSTM

Figure 1: Mean episodic reward for evaluation rollouts (limited to 100 timesteps) at the end of training for the different algorithms **when varying delay and specific sequences**. Please note the different colorbar scales.

## B Additional Learning Curves

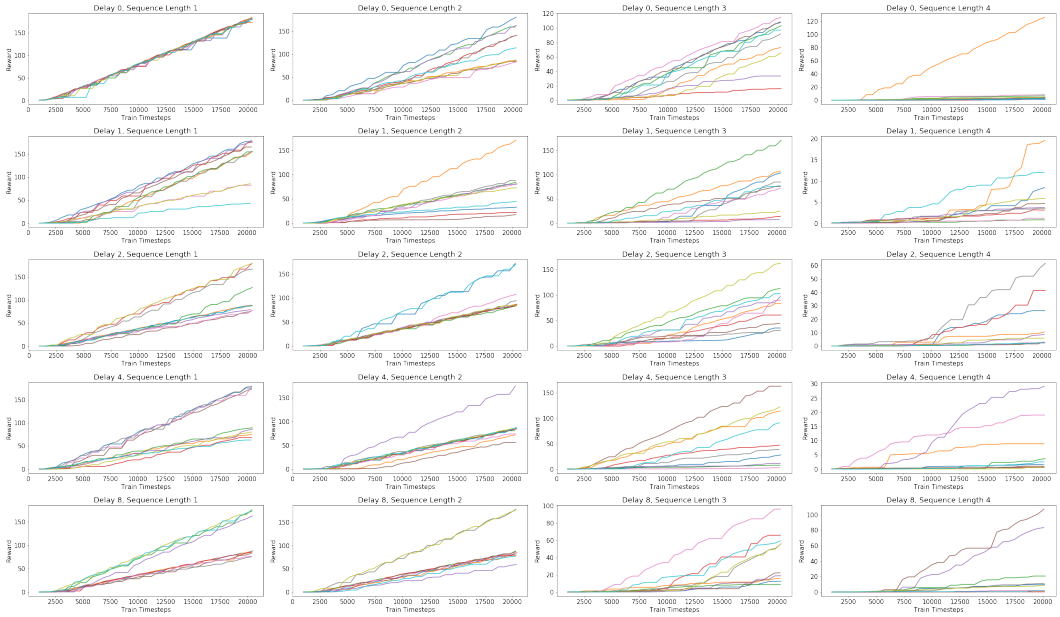



Figure 2: Training Learning Curves for DQN **when varying delay and specific sequences**. Please note the different colorbar scales.

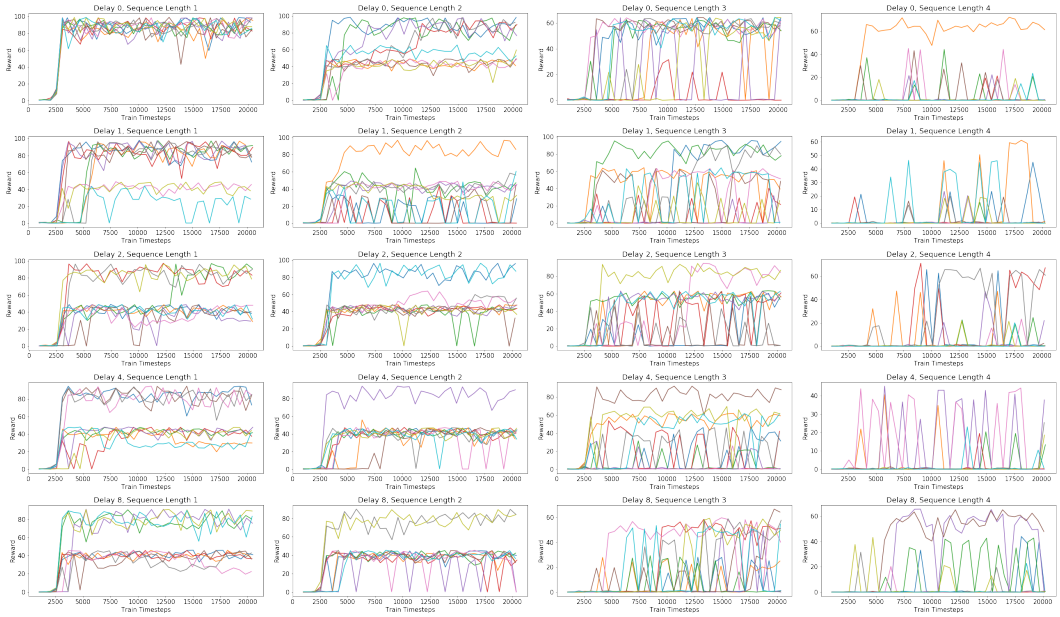



Figure 3: Evaluation Learning Curves for DQN **when varying delay and specific sequences**. Please note the different colorbar scales.

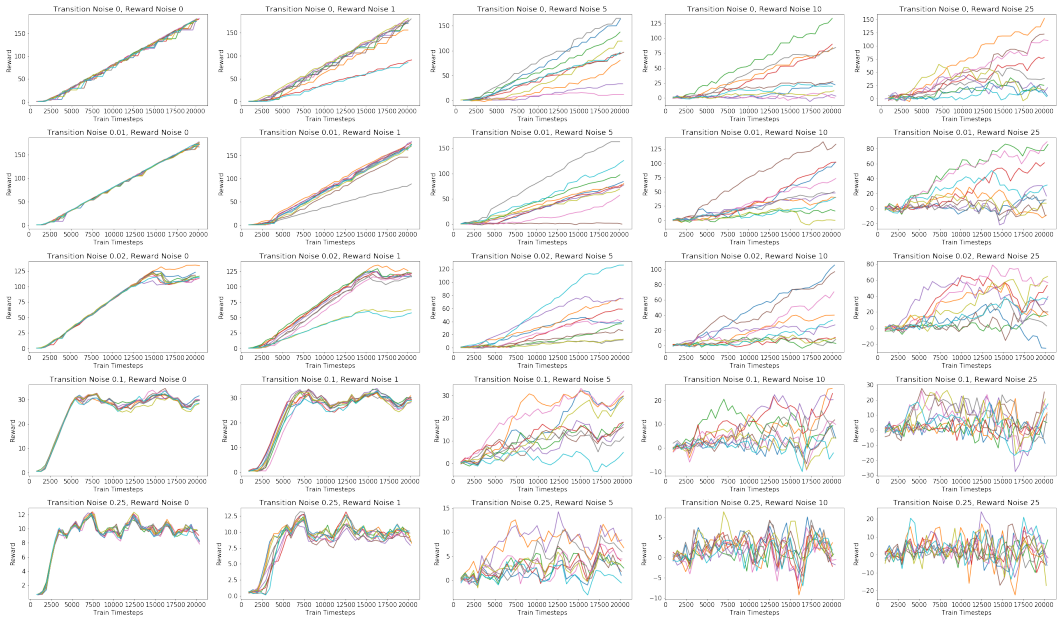



Figure 4: Training Learning Curves for DQN **when varying transition noise and reward noise**. Please note the different colorbar scales.

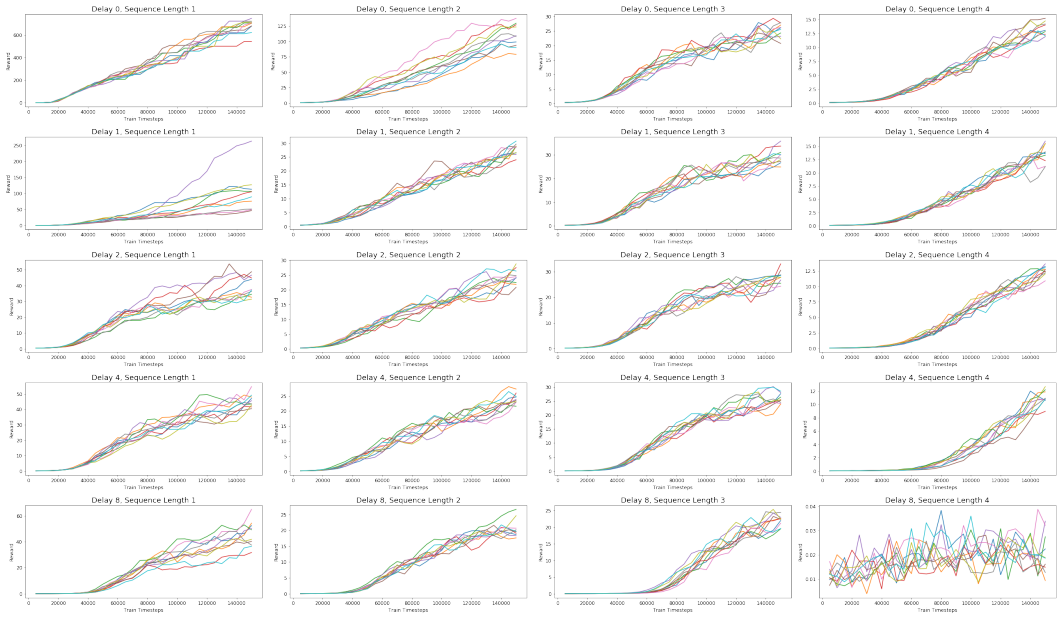



Figure 5: Training Learning Curves for A3C **when varying delay and specific sequences**. Please note the different Y-axis scales.

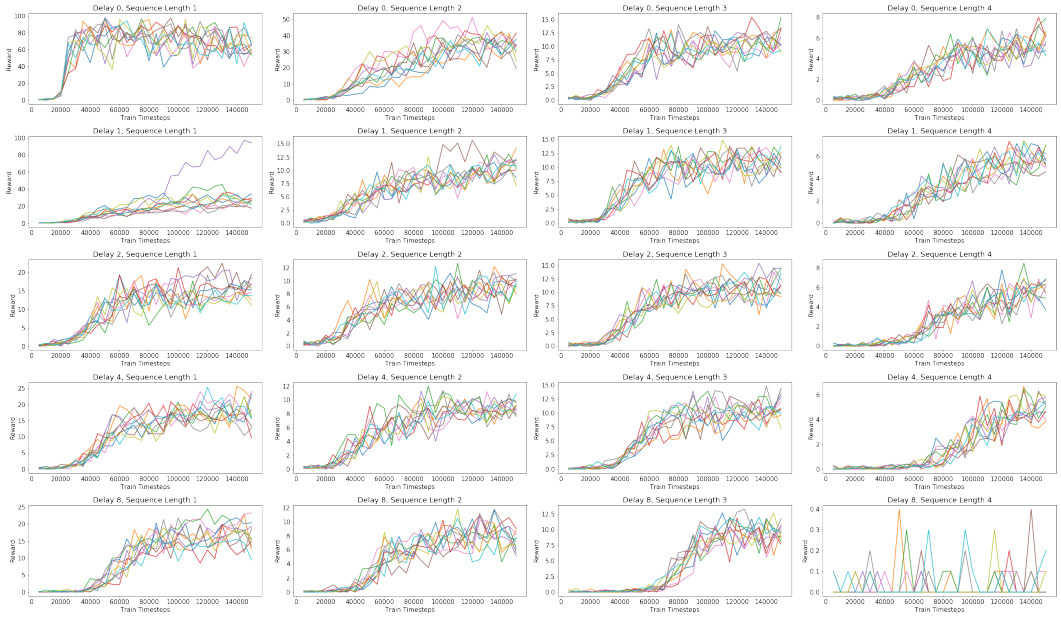



Figure 6: Evaluation Learning Curves for A3C **when varying delay and specific sequences**. Please note the different Y-axis scales.

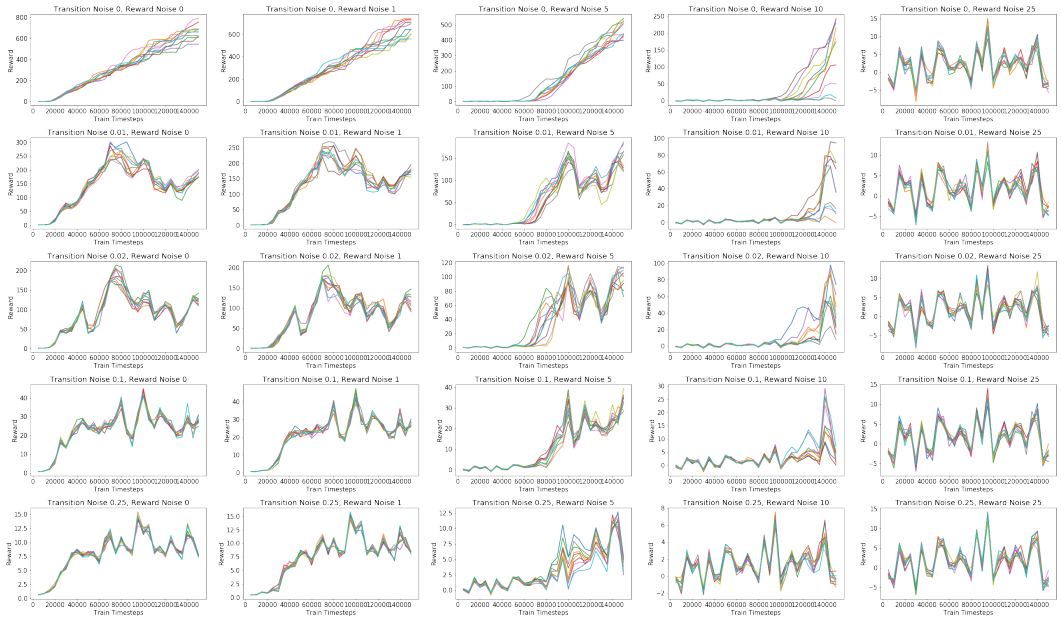



Figure 7: Training Learning Curves for A3C **when varying noises**. Please note the different Y-axis scales.

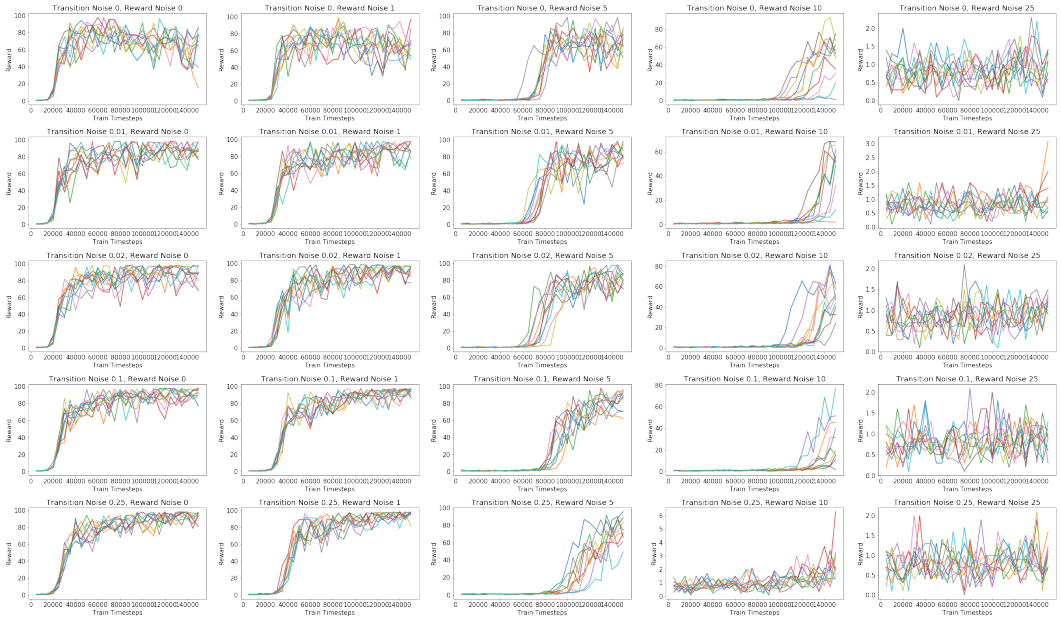



Figure 8: Evaluation Learning Curves for A3C **when varying noises**. Please note the different Y-axis scales.

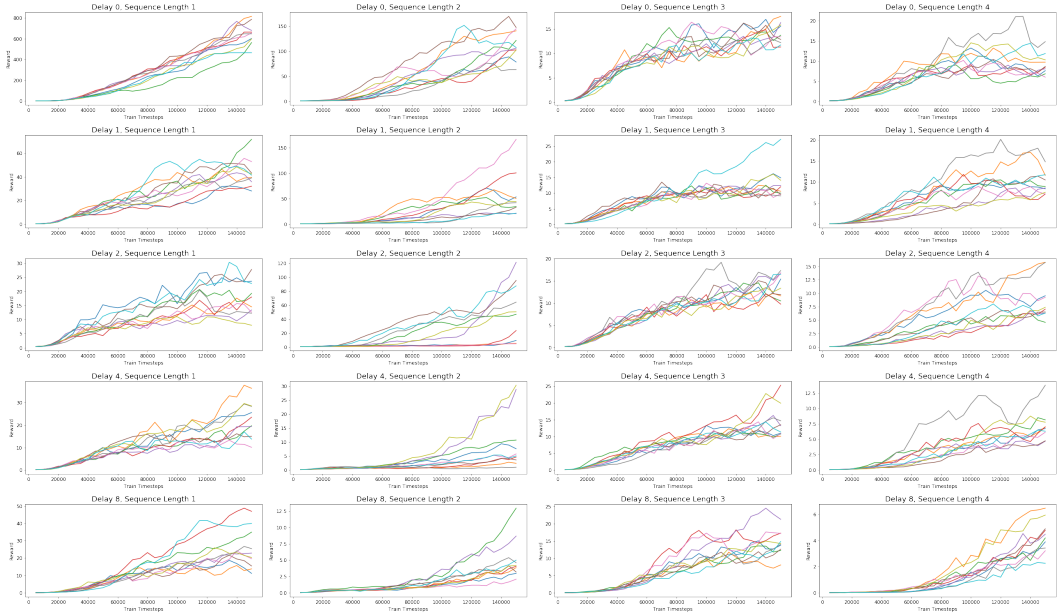



Figure 9: Training Learning Curves for A3C with LSTM **when varying delay and specific sequences**. Please note the different Y-axis scales.

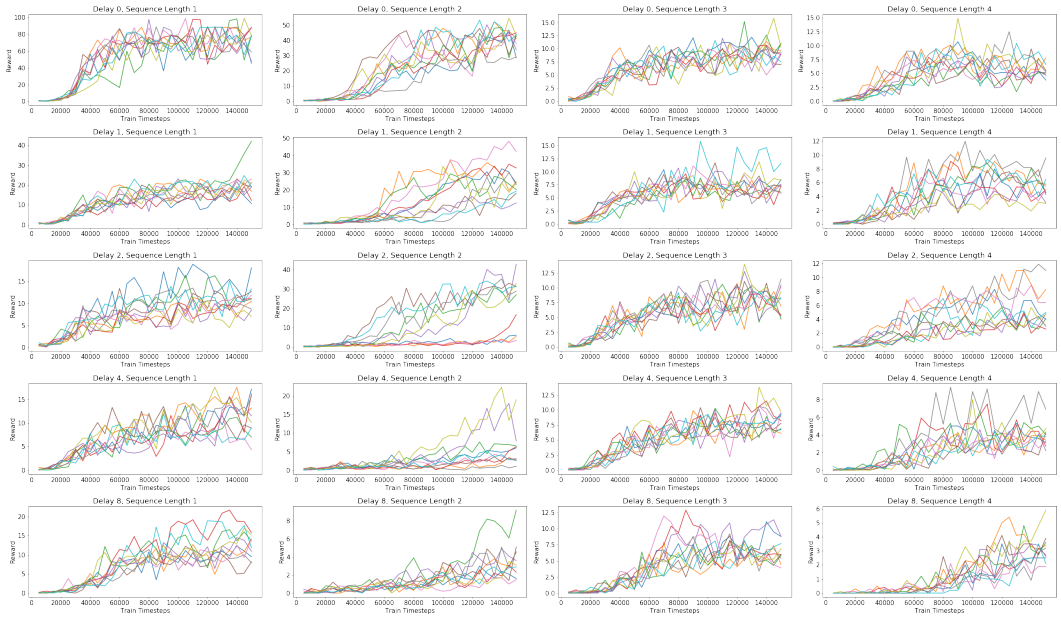



Figure 10: Evaluation Learning Curves for A3C with LSTM **when varying delay and specific sequences**. Please note the different Y-axis scales.

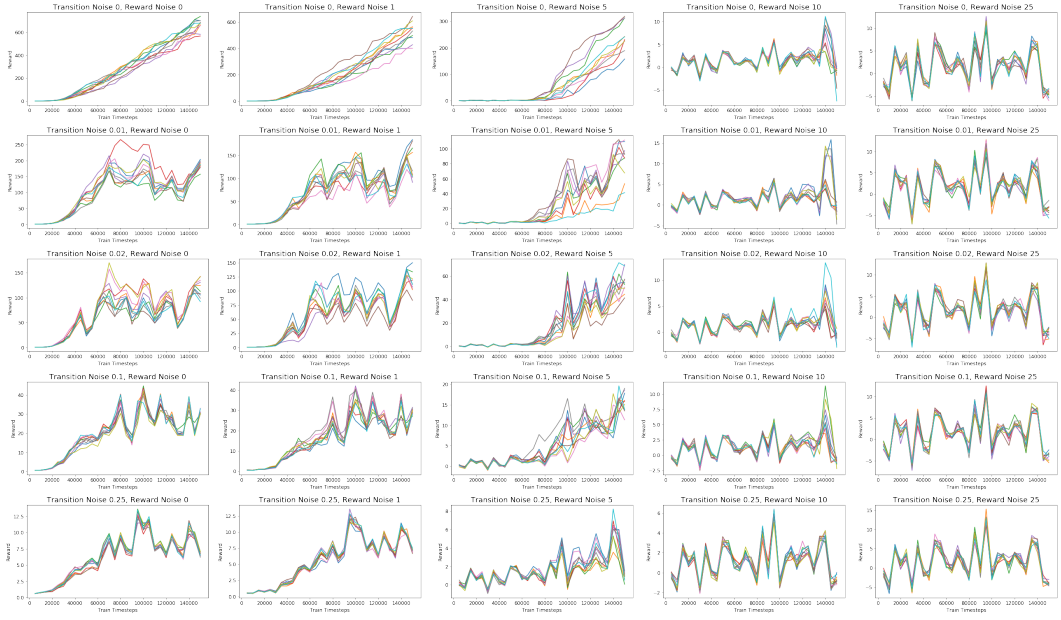



Figure 11: Training Learning Curves for A3C with LSTM **when varying noises**. Please note the different Y-axis scales.

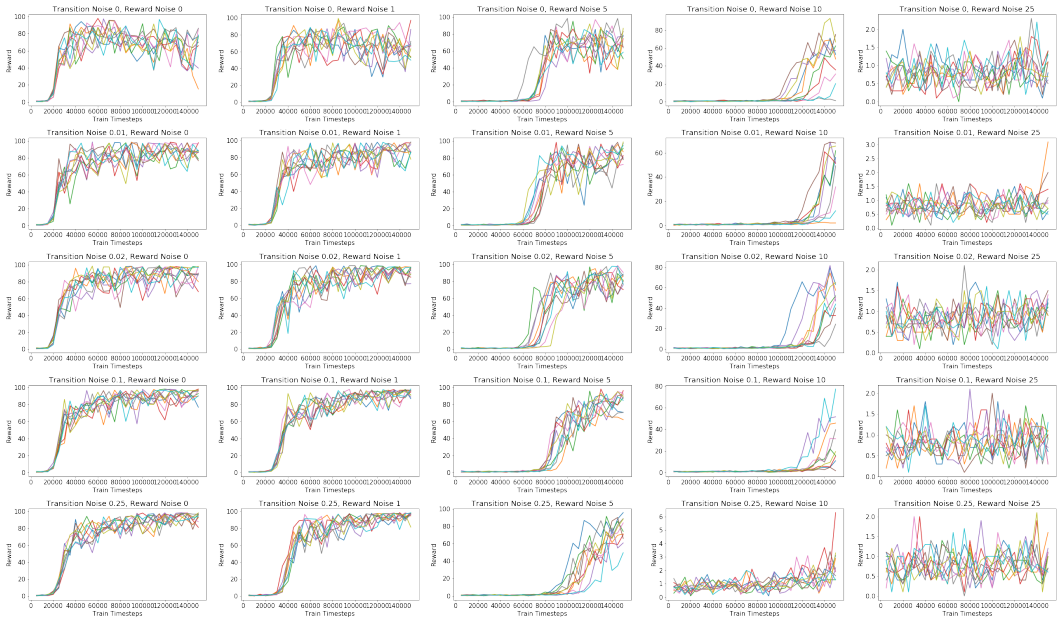



Figure 12: Evaluation Learning Curves for A3C with LSTM **when varying noises**. Please note the different Y-axis scales.

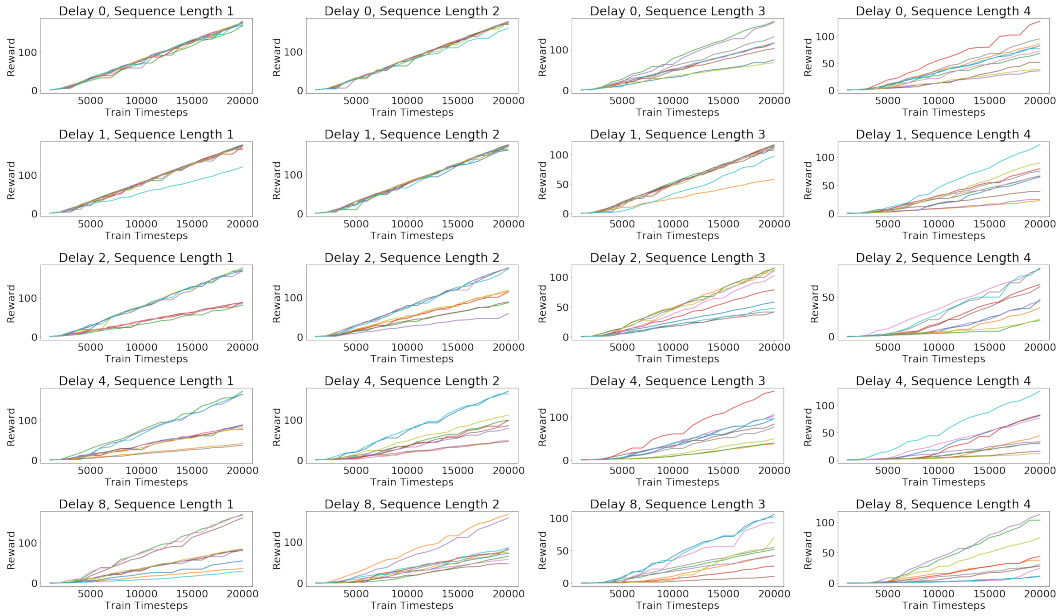



Figure 13: Training Learning Curves for Rainbow **when varying delay and specific sequences**. Please note the different Y-axis scales.

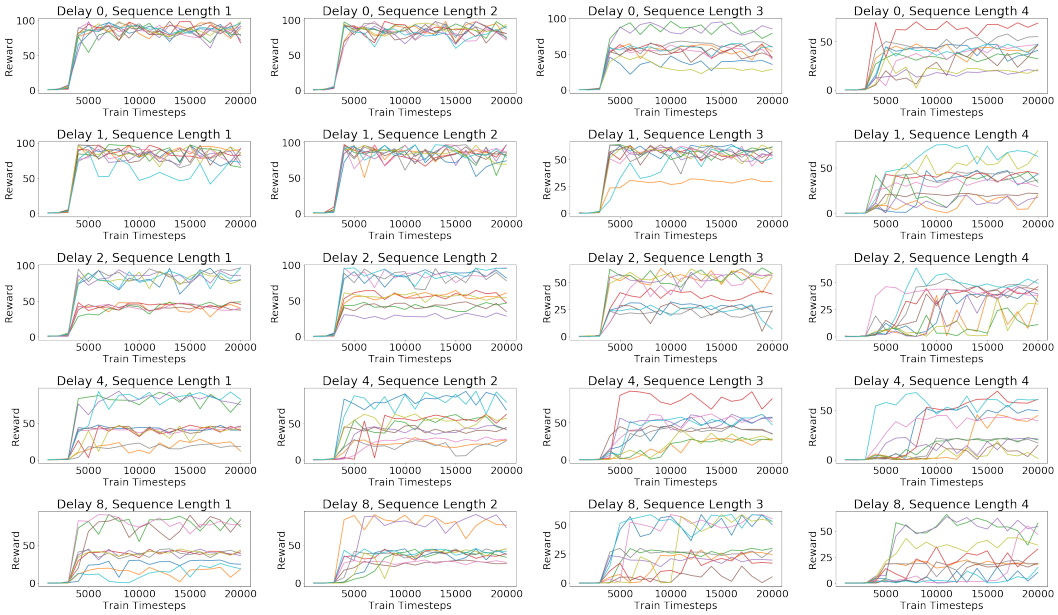



Figure 14: Evaluation Learning Curves for Rainbow **when varying delay and specific sequences**. Please note the different Y-axis scales.

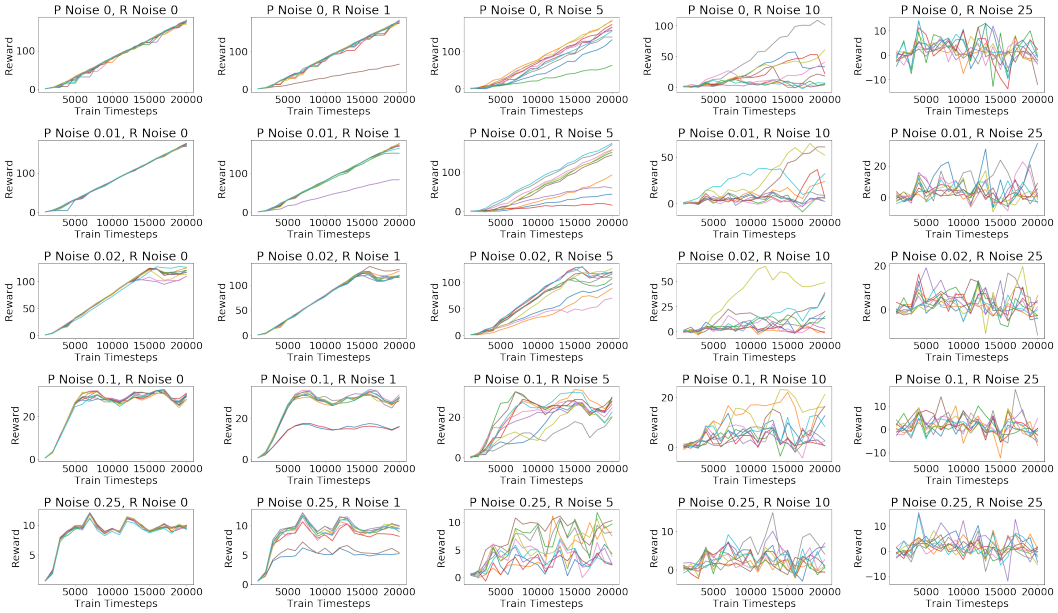



Figure 15: Training Learning Curves for Rainbow **when varying noises**. Please note the different Y-axis scales.

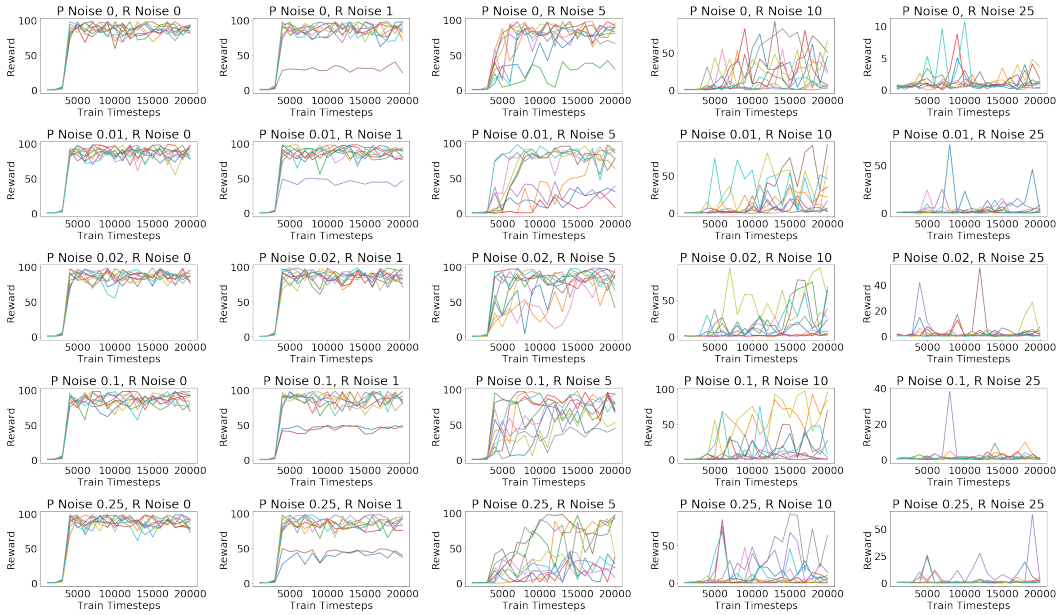
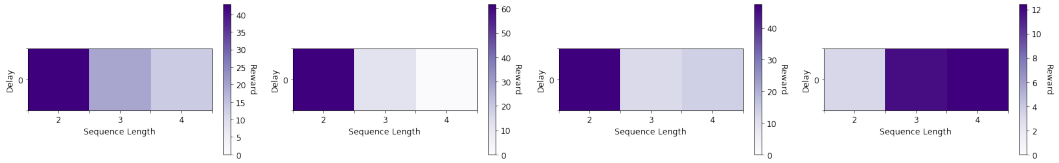



Figure 16: Evaluation Learning Curves for Rainbow **when varying noises**. Please note the different Y-axis scales.

# C Plots for Making Rewards Denser




(a) DQN (b) Rainbow (c) A3C (d) A3C + LSTM

Figure 17: Mean episodic evaluation rollout reward at the end of training for the different algorithms **when making reward for specific sequences denser**. Please note the different colorbar scales.

# D More on Conclusion and Future Work

Another key meta-feature, or rather meta-feature related to specific sequences, is manifolds. Let's say every specific sequence that was rewarded had a length $n$ and the reward density was, say, $0.1$. If we had no prior knowledge of the environment and did completely random exploration, we would need O($n^{|S|}$) sequences to obtain a reward signal which is exponential in the state space size (if we ignore the action space). We need to have some prior knowledge about the manifolds that exist in the environment to explore in a directed manner. This we intend to infuse through having a model of the environment. We will initially create simple manifolds for different tasks, such as rewarding following a circle in a 3-D continuous space and test directed exploration strategies to see how these choose to explore.

The states and actions contained in a specific sequence could just be a single *compound* state and *compound* action if we discretised time in a suitable manner. This brings us to the idea of learning at multiple timescales. HRL algorithms with formulations like the options framework (Sutton et al., 1999), could try to identify these specific sequences at the higher level and then carry out "atomic" actions at the lower level.

We also hope to benchmark other algorithms like PPO[1] (Schulman et al., 2017), Rudder (Arjona-Medina et al., 2018), MCTS (Silver et al., 2016) and table-based algorithms and to show theoretical results match with practice on toy benchmarks.

We also aim to promote reproducibility in RL as in (Henderson et al., 2017) and hope our benchmark helps with that goal.

We need different RL algorithms for different environments. Aside from some basic heuristics such as applying DDPG (Lillicrap et al., 2015) to continuous environments and DQN to discrete environments, it is not very clear when to use which RL algorithms. We hope this will be a first step to being able to identify from the environment what sort of algorithm to use and to help build adaptive algorithms which adapt to the environment at hand. Additionally, aside from being a great benchmark for RL algorithms, it is also a great didactic tool for teaching how RL algorithms work in different environments.

# References


Arjona-Medina, J. A., Gillhofer, M., Widrich, M., Unterthiner, T., Brandstetter, J., and Hochreiter, S. (2018). Rudder: Return decomposition for delayed rewards.

Henderson, P., Islam, R., Bachman, P., Pineau, J., Precup, D., and Meger, D. (2017). Deep reinforcement learning that matters.

Lillicrap, T. P., Hunt, J. J., Pritzel, A., Heess, N., Erez, T., Tassa, Y., Silver, D., and Wierstra, D. (2015). Continuous control with deep reinforcement learning.

Schulman, J., Wolski, F., Dhariwal, P., Radford, A., and Klimov, O. (2017). Proximal policy optimization algorithms.

Silver, D., Huang, A., Maddison, C. J., Guez, A., Sifre, L., Van Den Driessche, G., Schrittwieser, J., Antonoglou, I., Panneershelvam, V., Lanctot, M., et al. (2016). Mastering the game of go with deep neural networks and tree search. *nature*, 529(7587):484.

Sutton, R. S., Precup, D., and Singh, S. (1999). Between mdps and semi-mdps: A framework for temporal abstraction in reinforcement learning. *Artificial intelligence*, 112(1-2):181–211.


[1] We tried PPO but could not get it to learn