GROUP 1
INVERTED PENDULUM SIMULATION USING REINFORCEMENT LEARNING

Yuanjun "Dastin" Huang and Po Hsiang Huang

Department of Electrical and Computer Engineering
University of California San Diego, La Jolla, CA 92092

ABSTRACT

Reinforcement Learning (RL) is widely used in intelligent system control problems. This type of system is capable of imposing an influence to the surrounding environment and perceiving the state of itself through a set of sensors. Oftentimes the system has a specific goal to achieve, such as changing the environment or staying in a specific status. Each time the system executes a specific action, its status as well as the environment will change according to a specific rule. In most real-world robot-environment-interactions, such a rule is based on a well-defined kinematic model, giving the problem a clear physical background. In this project, we will explore how reinforcement learning can be applied in such a scenario by focusing on a simple yet challenging cart pole problem. We will also compare different methods such as Q-Learning, Double Q-Learning, and Deep Q Network on the amount of iterations they take to solve the problem.

1. INTRODUCTION

The problem can be modeled as keeping a pole which is mounted to a cart using only one bearing with 1 degree of freedom straight-up as much as possible, while the cart can accelerate either forward or backward on a straight trajectory. Though having a concrete physical model, the problem can be further extended by adding additional constraints or new mechanical components, which helps approximate complex real-world physical systems.As a high-level abstraction of more general control problems, the cart pole problem provides a perfect experimental field for developing and testing RL algorithms because of its solid physical background and ultimate simplicity. We believe that this is a perfect problem to work on for the course project given its importance and abundance of relevant resources.

Our main objective is to develop an algorithm to keep an inverted pendulum from falling. Inverted pendulum problem, also known as the cartpole problem, is a system where a pole has its center of mass above its pivot point tied to a cart, which we want to keep from falling. It is naturally an unstable system which requires outside force to keep its balance. This is a widely-adopted baseline problem for many control algorithms. While this problem has a closed form solution, we want to take the approach of reinforcement learning to show how it can be applied to general problems of making a sequence of decisions. Our project will utilize OpenAI Gym, which is a toolkit developed specifically for reinforcement learning. We will run tests in the Cartpole-v0 and Cartpole-v1 environments, whose goal is to reach an average of 195 timesteps without falling with maximum steps set to 200, and an average of 475 timesteps without falling with maximum steps set to 500, respectively.

2. RELATED WORK

As a fundamental problem, the cart pole problem has been thoroughly studied by the research community with fruitful results in both closed-form control strategies and learning-based strategies. Fixed Point Controllers and Stabilization of the Cart-pole System and the Rotating Pendulum [1] discussed stabilization of cascaded nonlinear systems from a control theory perspective and applied the theorem to analyse cart pole problem. Comparison of Reinforcement Learning Algorithms Applied to the Cart-pole Problem[2] reviewed several popular RL algorithms and compared their performance with numerical simulations. Open AI Gym[3] was published in 2016, providing standardized environments for reinforcement learning tasks. Throughout the past few years, there have been many attempts to solve the cartpole problem with as few episodes as possible. These submissions are available on the OpenAI Gym Leaderboard, each with a unique approach and performance. From surveying this leaderboard, we found that the fastest solution would be implementing a closed-form solution, followed by genetic algorithms and deep reinforcement learning methods, both of which are excellent approaches to determining a sequence of decisions. We note that closed-form solutions are much less scalable as the problem gets exponentially complicated when more states are introduced in the environment. Balancing a CartPole System with Reinforcement Learning - A Tutorial[4] walks readers through a series of different Q-Learning methods to solve the cartpole problem, which we used as a starting point to understand and work on the problem.

Variations of Q Learning methods have received a lot of attention since OpenAI Gym provided us with a simple and

standardized test environment. There are many publications that we have found relating to our problem, and here are a few that impacted our implementation. From Deep Reinforcement Learning with Double Q-learning[5], we learned that Q learning can introduce bias which is what prompted us to also work on double Q learning. We saw how using neural network in conjunction with reinforcement learning allows the problem to be solved with relatively few episodes and higher adaptability [6]. From these existing researches, we also learned the impact of choosing different number of discrete states on the traditional Q learning method and how neural network eliminates that process by taking in either continuous kinematic state values or screenshots of the environment.

3. DATASET AND FEATURES

For our cartpole problem, we do not need external dataset. We are observing the current states from the environment and updating and performing actions accordingly. The observations from the environment include continuous values of the cart position, cart velocity, pole angle, and pole angular velocity. These observations are collected on the fly and corresponding actions to either push the cart to the left or to the right would be issued based on maximizing the rewards. This allows our cart to learn and make decisions without a large dataset or significant training time.

For our project, we are implementing Q Learning, Double Q Learning, and Deep Q Network to solve the inverse pendulum problem. These methods involve different feature extractions. For Q Learning and Double Q Learning, we need to construct Q Tables to capture the best actions for each observation. To do this, we first need to discretize the continuous observations into bins of states. Based on Neuronlike Adaptive Elements Learning Difficult Control Problems[6] , it is suggested to discretize the observations into 162 states. However, we found that the performance of our models was not as well as we expected. From one of the OpenAI Gym Leaderboard contributors, Trevor McInroe, we found that they actually discretize the states into a tuple of (1, 1, 6, 12) for each observation, giving little consideration to the cart position and velocity white assigning high importance to the pole angle and the highest importance to the pole angular velocity. We also tested out different values using grid search only to find that this combination yields results far better than others.

As for Deep Q Network, we do not need to discretize our observations from the environment. However, as part of our experiment, we tried extracting each frame instead of using the kinematic state values from the environment to compare the performance from different neural networks.

4. METHODS

Traditionally we have to iteratively solve the Bellman equation to find the best Q-value for each state. However, the dimension of state space increases significantly when the underlying physical model gets more complicated. For example, multiple inverted pendulums in a single scenario could potentially jeopardize the effectiveness of traditional approaches due to the number of states to keep track of. We believe that deep reinforcement learning would be capable of solving this problem by learning a function approximator to fit the Bellman equation. It takes away the need for discretizing continuous state values and this can lead to better quality results that can stay balanced for longer and are more robust to environmental changes.

4.1. Q Learning

Q, or the quality index, is the direct measurement of the future expected reward which we want to maximize in this problem. In the training process the of Q learning algorithm, we use the value of epsilon to control whether we want to explore or exploit our environment. Exploration performs a random action from the action space, and Exploitation performs the optimal action from our Q Table. This process allows the algorithm to learn and update the Q Table iteratively to keep the pole from falling.

Our problem follows a Markov chain where the cart performs an action, which triggers an new state and corresponding reward from the environment, which are then used to update the optimal action to take to keep the environment stable. The update rule for the Q table is also known as the Bellman equation [1] for the action value function, where we are updating each state and action's Q value by its corresponding old Q value plus the learning rate, alpha, times the sum of reward, discount factor gamma times the estimate of optimal future value, and negative old Q value. This update is performed iteratively until the pole falls from the cart, the cart moves too far away from the center, or the maximum number of steps is reached.

4.2. Double Q Learning

After successfully implementing Q Learning, we move to implement the Double Q Learning algorithm which involves using two Q Tables. The training process of Double Q Learning is fairly similar to that of the Q Learning algorithm. However, the optimal action chosen is based on the average values of the two Q tables instead of just one. Each Q table is

$$Q^{new}(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha(r_t + \gamma max_a Q(s_{t+1}, a) - Q(s_t, a_t))$$

Fig. 1. Bellman Equation for Q

updated alternately at each iteration. This algorithm was proposed by the published paper Deep Reinforcement Learning with Double Q-learning [5] in 2015 to counter bias introduced in a single Q Table as a result of overestimation of future return value. We implemented this along with Q Learning algorithm to compare with the performance of Deep Q Network.

4.3. Deep Q Network

While traditional approaches like Q Learning and Double Q Learning have already shown their effectiveness at solving the inverted pendulum problem, they suffer from some constraints. For example, both approaches require the states and control to be discreatized before running the learning algorithm, making it infeasible to directly deploy them on continuous-state control problem that is prevalent in real physical world. What's more, oftentimes engineers have to find a balance point between discretization granularity and control optimality, further increasing the workload. With the advent of deep learning, researchers have proposed to adopt deep learning techniques to calculate the Q function. This is based on the intuition that given a properly designed network architecture and sufficient unbiased training data, a deep neural network can be trained to approximate an arbitrary given function. Deep Q Learning is more suitable for problems with continuous state or control space, since it requires little effort to modify a deep learning model to take continuous input variables and generate continuous outputs.

As mentioned before, we can encode the system's state either as a tuple of of kinematic state values or as an image that can be easily rendered by OpenAI Gym toolkit. We design 2 different Deep Q Network architectures for each encoding scheme.

4.3.1. Fully Connected DQN For Kinematic Encoding

Under this setup, our goal is to build a Deep Q Network that mimics the Q Table described in the previous chapter. That being said, the network should take an input vector with length 4 and generate a corresponding output vector with length 2. Since the input is already a compact feature representation of the system's kinematic state, we don't need to design complex feature extraction structures any more. Following the convection of neural architecture design for low-dimension input and output, we adopt a 3-layer fully connected network (FCN) as our Deep Q Network, whose hidden layers' dimension details are shown in Fig.2. For activation function, we adopt sigmoid as the for the first 2 layers and ReLU for the output layer.

4.3.2. Convolution DQN For Image Encoding

Though the 4-tuple described is enough for a compact state representation, we're still interested in generating the control in a more "natural" way. Consider a human who is playing the cart-pole game, he/she generates the control based on the image representation of the kinematic system. While it is infeasible to implement traditional Q Learning algorithm under this setup due to the overwhelming dimensionality of the input, we can apply convolution layers in our Deep Q Network to effectively extract the features and thus reduce the data dimension. To encode the temporal difference information of our system, i.e., the cart's velocity and the pole's angular velocity, we subtract the rendered image of previous step from the current image, rather than feed the current image directly, to our Deep Q Network. We show the details of each network layer in Fig.3.

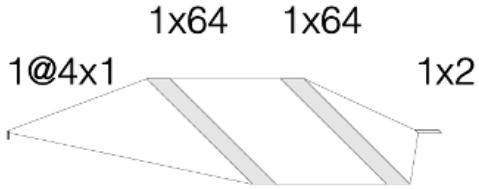

Fig. 2. Fully Connected Network Structure

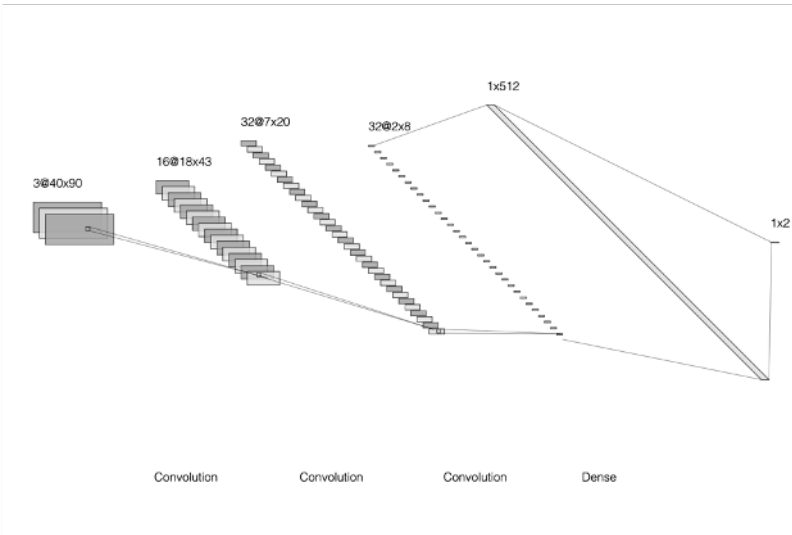

Fig. 3. Convolution Neural Network Structure

5. EXPERIMENTS AND RESULTS

We evaluate all 4 described approaches on OpenAI Gym CartPole-V1 environment under the same settings to ensure a fair comparison. CartPole-v1 terminates the simulation when 500 actions have been taken, regardless of the inverse pendulum system's state. It is also widely agreed-on metric that an algorithm is deemed to have successfully "solved" the inverse pendulum control problem once it can keep the system from falling for an average of 475 actions or more. We trained both

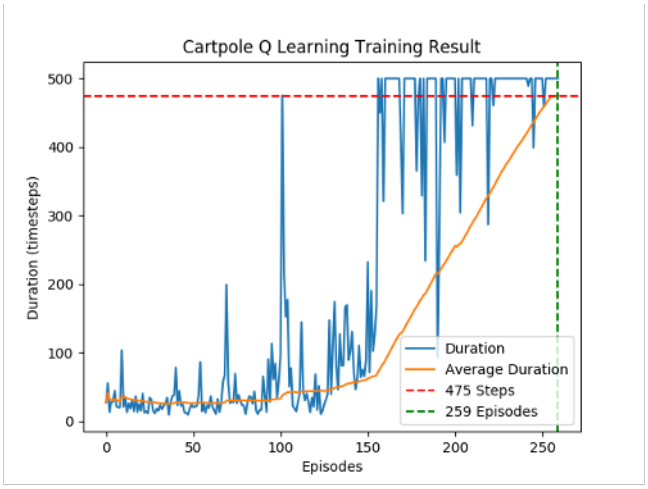

Fig. 4. Q Learning Performance w.r.t. Episode

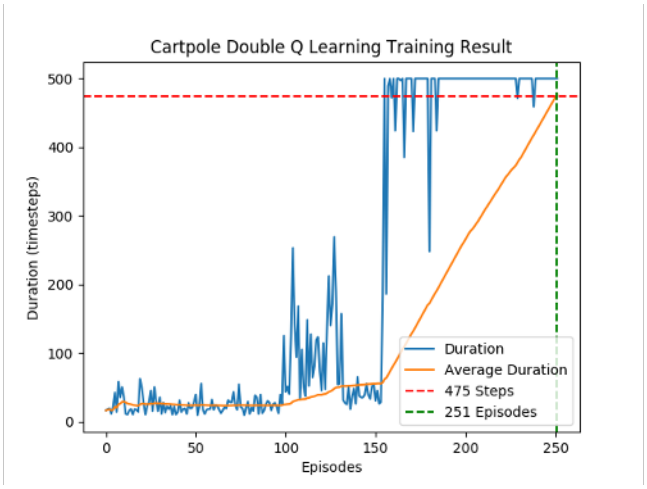

Fig. 5. Double Q Learning Performance w.r.t. Episode

Q Learning and Double Q Learning model until it meets the aforementioned criterion. For DQN models, however, since the weight initialization plays an important role in determining how fast model converges, the number of episodes needed to meet the criterion vary much from trial to trial so we simply train the model for a fixed number of episodes, i.e., 200 for FCN DQN and 400 for FCN DQN. The models' performance w.r.t. episodes run are shown in Fig.4 7.

We discover that FCN DQN requires less episodes to converge than traditional approaches. Furthermore, if the 500-action cap is to be removed (this can be done with some hacking on CartPole environment), the control strategy generated by FCN DQN can keep the inverse pendulum straight-up longer before falling compared to the strategy acquired using traditional approaches.

However, the success of FCN DQN doesn't mean that DQNs will always outperform traditional approaches. To our dismay, CNN DQN fails to generate a satisfying control strategy no matter how many episodes we train. We believe this failure is due to the fact that CNN DQN is actually trying to solve a different problem than the other 3 models since the input is different. The image input has a dimension that is larger than that of 4-tuple kinematic state by several magnitudes, making the problem much difficult to solve by nature.

6. CONCLUSION

We studied the inverse pendulum problem and implement 4 different RL approaches to solve it. The FCN-based DQN outperforms traditional RL in the sense that it takes less episodes to generate a better control strategy. However, CNN-based DQN fails solve the problem because the state space for image input is too huge.

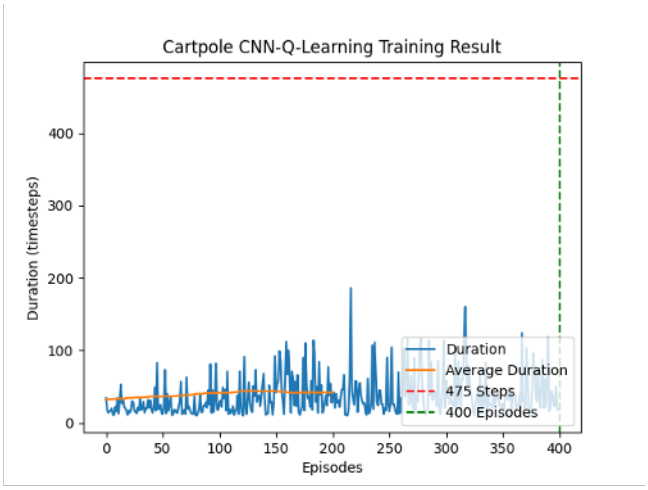

Fig. 6. CNN DQN Performance w.r.t. Episode

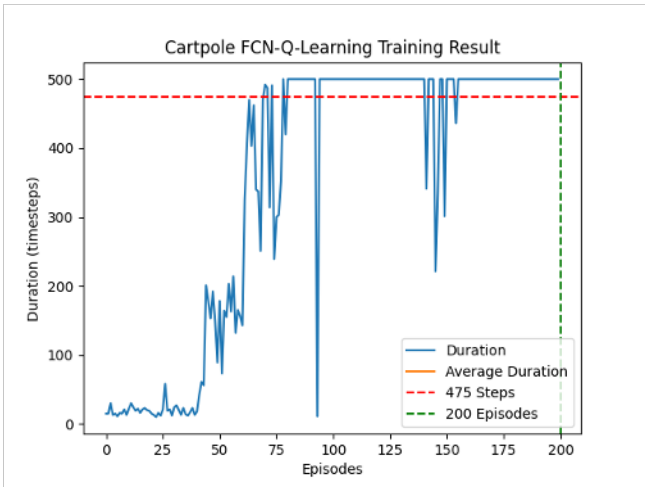

Fig. 7. FCN DQN Performance w.r.t. Episode

7. REFERENCES

[1] R. Olfati-Saber. Fixed point controllers and stabilization of the cart-pole system and the rotating pendulum. In *Proceedings of the 38th IEEE Conference on Decision and Control (Cat. No.99CH36304)*, volume 2, pages 1174–1181 vol.2, 1999.

[2] Savinay Nagendra, Nikhil Podila, Rashmi Ugarakhod, and Koshy George. Comparison of reinforcement learning algorithms applied to the cart-pole problem. In *2017 International Conference on Advances in Computing, Communications and Informatics (ICACCI)*, pages 26–32, 2017.

[3] Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. Openai gym. *CoRR*, abs/1606.01540, 2016.

[4] Swagat Kumar. Balancing a cartpole system with reinforcement learning - A tutorial. *CoRR*, abs/2006.04938, 2020.

[5] Hado van Hasselt, Arthur Guez, and David Silver. Deep reinforcement learning with double q-learning. *CoRR*, abs/1509.06461, 2015.

[6] Andrew G. Barto, Richard S. Sutton, and Charles W. Anderson. Neuronlike adaptive elements that can solve difficult learning control problems. *IEEE Transactions on Systems, Man, and Cybernetics*, SMC-13(5):834–846, 1983.

8. CONTRIBUTIONS

Yuanjun ”Dastin” Huang mainly works on the Deep Q Learning model design and training. Po Hsiang Huang mainly focuses on implementation of traditional Q-Learning algorithms that serve as a baseline for our DQN. Both team members contribute to the video presentation, final report and management of GitHub repository.

9. REPLY TO REVIEWS

9.1. Group 26

The talk explicitly introduces the background of Inverted Pendulum problem and OpenAI Gym and details of the feature extraction and model architecture. The comparison of implementation Q-learning, Double Q-learning and Deep Q learning for the cart-pole problem looks reasonable and thoughtful.

Thank you!

How will the difference in two pipelines influence the result? Expect the fact that the first one starting from rendered image will be more difficult to implement

We are asking this question ourselves as well. We wanted to see how the different neural network architecture could influence the results. Indeed, there was a lot more preprocessing for training with CNN and the training time was longer as well. Because of the difference in training input, the test results we observed were wildly different as well. CNN tend to keep the cartpole more centered while FCN sometimes have the cartpole move out of the frame despite the pole being balanced. However, FCN appears to reach the target timestep within fewer episodes of training. This goes to show that both architectures have their pros and cons and it would take more experimenting to determine whether there is an architecture better suited for this problem

What was the reason for choosing the architecture of your current failed CNN model? Since it has relatively large input parameter, could you please try to change the feature map number in some layer?

We build the CNN architecture on top of a reference model provided by PyTorch RL tutorial. Changing the feature map number will probably be helpful, but we didn't have enough time to finish the work by project submission deadline.

There is Double Deep Q-learning on your to do list. Since double Double Q-learning can avoid overestimate in Q-learning, it is actually a good and worthwhile point to keep trying with Double Deep Q-learning.

Yes we agree. We were not able to implement the Double Deep Q-Learning within permitted time, but this is definitely something we would like to explore beyond this class.

What is the reason for choosing current variable combination, such as learning rate, discount rate, and initial conditions?

We referenced different literatures and implementations from Open AI Gym leaderboard and got a general sense of where the range of these values lie. We then performed grid search for each parameter and the parameters we are using performed the best in terms of accuracy and episodes needed for convergence.

The visualization for each model looks great. But could you visualize your result and the learning process somehow for all the model in one plot? It will be more intuitional and easier to do the comparison.

We tried plotting as suggested but the plot was rather messy and hard to read. Additionally, because the termination criteria for Deep Q Networks are different, we decided it was best to keep the plots separate.

9.2. Group 29

Using OpenAI Gym, the authors provide clear background information and solid literature review. The authors implement interesting machine learning techniques - Q Learning, Double Q learning, and Deep Q Learning, which make use of Markov chains. They looked to papers and leaderboards for inspiration, and identified methods for feature extraction to improve upon initial scores that were lower than they expected.

Thank you for the encouragement!

I wonder why the training result is so lossy?

The training result may appear to be lossy for two reasons. 1. Because the maximum number of timesteps is limited according to the problem definition so that a stable system does not run on forever, the duration of our training result is in fact cut off a at a threshold of 500. 2. Because there is alternating exploration and exploitation phases, where the exploration setting makes the cartpole move in random direction. This often times creates scenarios where the algorithms have not seen before and therefore falls prematurely. However, this is all part of the training process so it is actually a good thing that the algorithms are seen all the different possible combinations.

Does the network performance generalize to test cases with different parameters (mass, starting angle, etc) or does it need to be retrained for each situation?

Our methods would have to be retrained in the case of different physical environment as that would drastically affect the outcome (new state) of each action. However, these reinforcement learning methods do not take long to train so it should not be a big issue to retrain under different physical environment. For a more complicated environment, the training time could increase significantly, in which case we can consider using transfer learning on different sets of physical environment to reduce training time.

How does your result compare to the other models on the leaderboard? I may have missed this but didn't find it in your presentation.

Our Q-Learning and Double Q-Learning results are on par with the performance of the top 30 models on the leaderboard, and our Deep Q Networks are on par with the performance of the top 20 models on the leaderboard. However, we want to put a disclaimer that this is based purely on the minimum number of episodes required to reach the target number of timesteps. Because each model can be drastically different from each other in terms of parameter settings and architec-

tures, it is possible that the comparison could be drastically different in the case, say for example, we leave the testing environment on for indefinite amount of time.

Nice work with you DQN-FNN! Overall I think this was a good project and very well presented with clean slides and thoughtful design.

Thank you for the encouragement!

9.3. Group 31

Done well:

The presentation did a thorough description on the 'Inverted Pendulum problem' and why they want to investigate the four different methods of solving the problem. The detailed implementation details, results analysis and interesting code demo clearly demonstrate the comparison between performances of traditional Q-learning model, double Q-learning model, and more recently CNN based and FCN based models.

Thank you for the encouragement!

Some improvements / Unclearness:

The CNN based solution (that takes a H*W*3 input image as input) performs poorly as shown in the results section, and the given analysis for it is that the data is too complex (too many features). We would suggest trying to use a grayscale input image that transform the H*W*3 into a H*W*1 shaped input and in this way reduces the input complexity to 1⁄3 as before while still keeping the essential information (locations, angles, etc.). Also input image preprocessing like resizing each dimension proportionally into a smaller input image or cropping the useful content of the image might be worth trying.

Thank you for these suggestions! We will definitely consider these in future iterations!

At the 'Results Summary' slide, it is not clear about the difference between Cartpole-v0 and Cartpole-v1 environments setup, are they only differ by the number of Maximum time steps?

Yes, it the only differences are the maximum time steps and the target average timestep. We made sure to include this in the introduction section of our report.

A minor suggestion about the 'Results' slide, the legends are not very properly placed (Especially for the top right figure, the legend covers some portion of performance curves).

We will move the legend to the top left to avoid overlapping with the training result curve. Thank you for pointing this out!