Ablation Study: Sample Efficient Actor-Critic with Experience Replay

109550034 Nai-Chieh Huang
310554006 Wen-Tao Chu
310554001 Yu-Wei Yang
310605010 Zion Sung

Outline

- Introduction
- Implementation Details
- Ablation Results
- Conclusion

Outline

- **Introduction**
- Implementation Details
- Ablation Results
- Conclusion

Sample Efficient Actor-Critic

- On-policy methods are computationally expensive (data inefficient)
- Experience replay techniques tackle the problem
- Deep Q-learning
- *Actor-Critic with Experience Replay?*
- **ACER!**

Actor-Critic with Experience Replay

- Retrace [Munos, 2017]
- Truncation
- Trust Region Method [Schulman, 2015]

Discrete ACER vs Continuous ACER

- Discrete ACER
  a. Multi-Step Estimation of the state-action value function
  b. Importance Weight Truncation with Bias Correction
  c. Efficient Trust Region Policy Optimization
- Continuous ACER
  a. Stochastic Dueling Network (SDNs) for policy evaluation
  b. Trust Region Updating for continuous setting

Discrete ACER:
Multi-Step Estimation of the state-action value function

- Here we use Retrace $Q^{\text{ret}}(x_t, a_t)$ to estimate $Q^{\pi}(x_t, a_t)$
- Using multi-step $Q^{\text{ret}}(x_t, a_t)$, it can significantly reduce bias in the estimation of the policy gradient
- In the learning of critic, it can speed up the learning process since Retrace is return-based

R. Munos, T. Stepleton, A. Harutyunyan, and M. G. Bellemare. Safe and efficient off-policy reinforcement learning.

Discrete ACER:
Importance Weight Truncation with Bias Correction

- Truncate the importance weights via the following gradient term (sampling version) :

sampling from replay buffer

$$
\begin{aligned}
\hat{g}_t^{\text{acer}} &= \boxed{\bar{\rho}_t}^{\leq c} \nabla_\theta \log \pi_\theta(a_t|x_t)[Q^{\text{ret}}\boxed{(x_t, a_t)} - V_{\theta_v}(x_t)] \\
&+ \underset{\boxed{a\sim\pi}}{\mathbb{E}}\left(\underbrace{\boxed{\left[\frac{\rho_t(a) - c}{\rho_t(a)}\right]_+}}_{\leq 1} \nabla_\theta \log \pi_\theta(a|x_t)[Q_{\theta_v}(x_t, a) - V_{\theta_v}(x_t)]\right)
\end{aligned}
$$

sampling from current policy

Discrete ACER:
Efficient Trust Region Policy Optimization

- New TRPO with ***average policy network*** for the **ACER** 's PG

the network ϕ_θ: $\pi(\cdot|x) = f(\cdot|\phi_\theta(x))$

$$\begin{aligned}\widehat{g}_t^{\text{acer}} &= \bar{\rho}_t \nabla_{\phi_\theta(x_t)} \log \boxed{f(a_t|\phi_\theta(x))} [Q^{\text{ret}}(x_t, a_t) - V_{\theta_v}(x_t)] \\ &\quad + \underset{a\sim\pi}{\mathbb{E}} \left(\left[\frac{\rho_t(a) - c}{\rho_t(a)}\right]_+ \nabla_{\phi_\theta(x_t)} \log \boxed{f(a_t|\phi_\theta(x))} [Q_{\theta_v}(x_t, a) - V_{\theta_v}(x_t)] \right)\end{aligned}$$

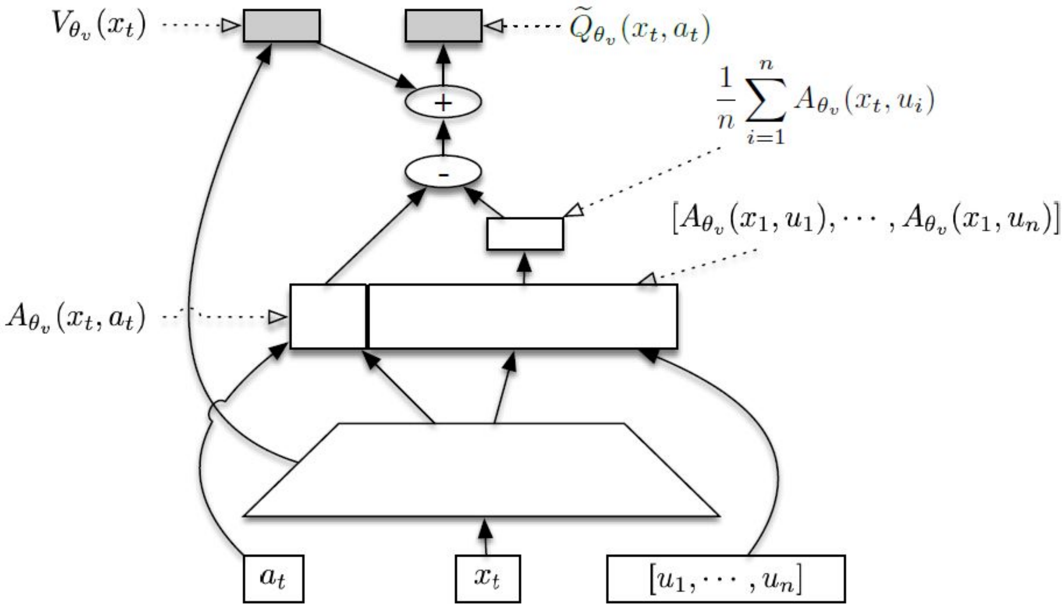
Update parameter : $\theta_a \leftarrow \alpha\theta_a + (1-\alpha)\theta$

Discrete ACER: Efficient Trust Region Policy Optimization (Cont.)

這裡看要不要放 optimization problem solved by KKT condition (我覺得多了)

Continuous ACER: Stochastic Dueling Network (SDNs) for policy evaluation

- We use SDNs to estimate both V^π and Q^π off-policy :



Continuous ACER:
Trust Region Updating for continuous setting

- It is similar to discrete version with the exception *f* and *SDNs*
- We sample $a'_t \sim \pi_\theta(\cdot|x_t)$ to obtain the following MC approximation :

the same as Retrace but the truncated importance ratio equal to 1

$$\begin{aligned}\hat{g}_t^{\text{acer}} \quad = \quad & \bar{\rho}_t \nabla_{\phi_\theta(x_t)} \log f(a_t|\phi_\theta(x_t)) \boxed{(Q^{\text{opc}}(x_t, a_t)} - V_{\theta_v}(x_t)) \\ & + \left[\frac{\rho_t(a'_t) - c}{\rho_t(a'_t)}\right]_+ (\widetilde{Q}_{\theta_v}(x_t, a'_t) - V_{\theta_v}(x_t)) \nabla_{\phi_\theta(x_t)} \log f(a'_t|\phi_\theta(x_t))\end{aligned}$$

Outline

- Introduction
- **Implementation Details**
- Ablation Results
- Conclusion

Implementation Details

We use **stable-baselines** to do our ablation study

- Advantage
  - General
  - Easy to use
    - Only need few lines of code
  - The name is clear
- Disadvantage
  - Environments is hard to build and complex
    - We spend lots of time to fix the environment problem…

```
import ...
env = gym.make('CartPole-v1')
model = ACER(MlpPolicy, env, verbose=1)
model.learn(total_timesteps=25000)
```

Outline

- Introduction
- Implementation Details
- **Ablation Results**
- Conclusion

Ablation Results

Environment:

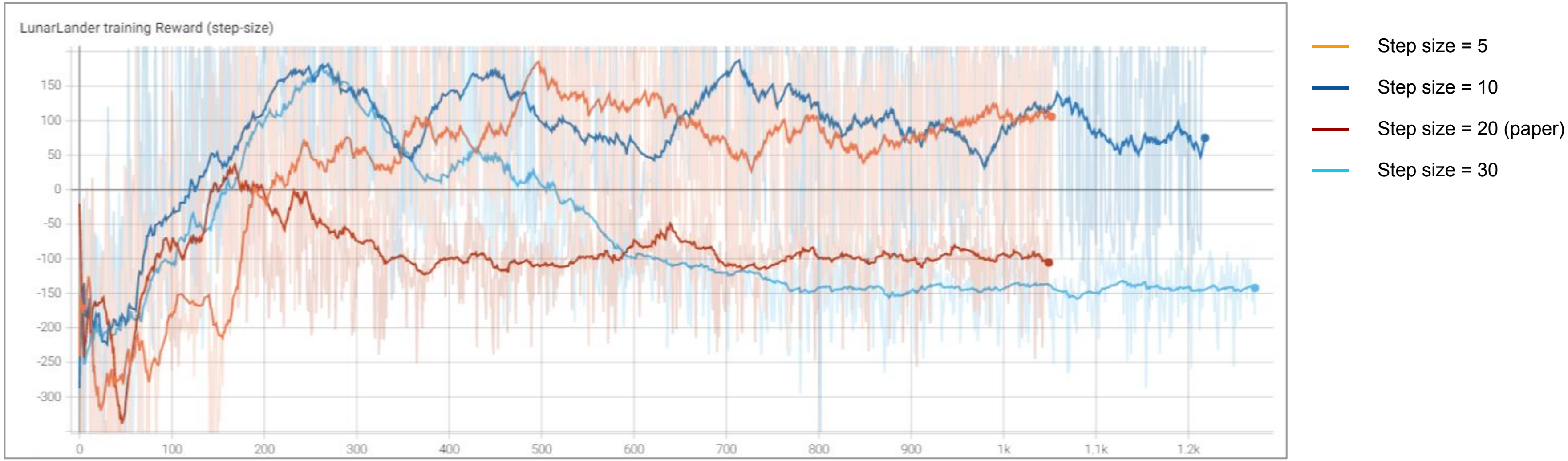
- LunarLander-v2
- CartPole-v1

Modify three parameters:

- Step size
- Importance weight truncate
- Policy trust region

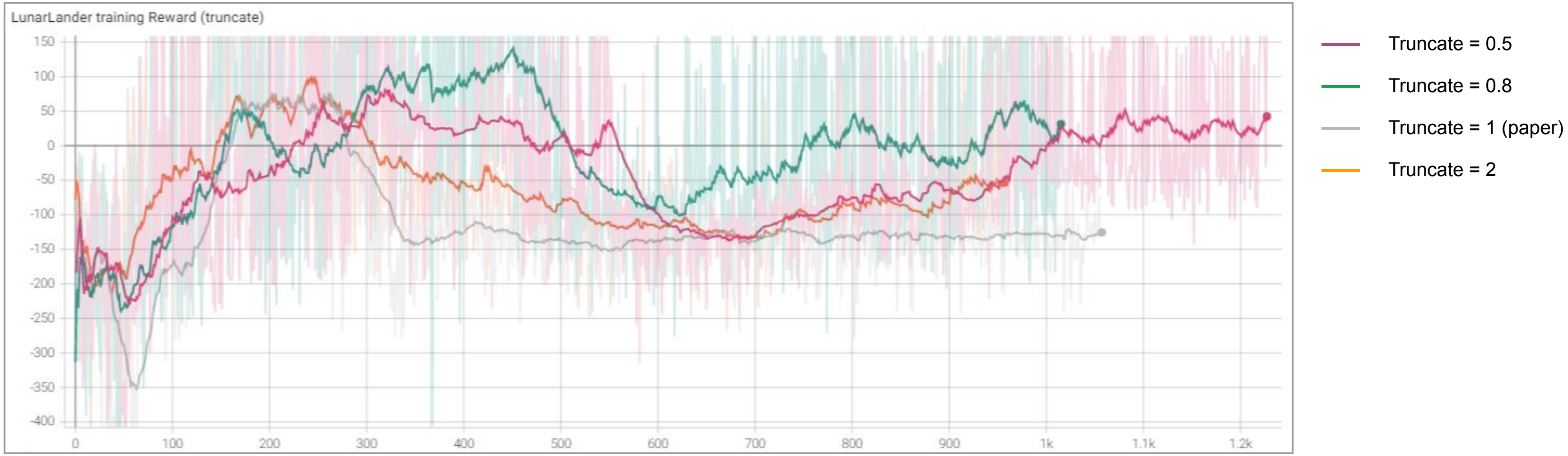
LunarLander-v2

Step size :



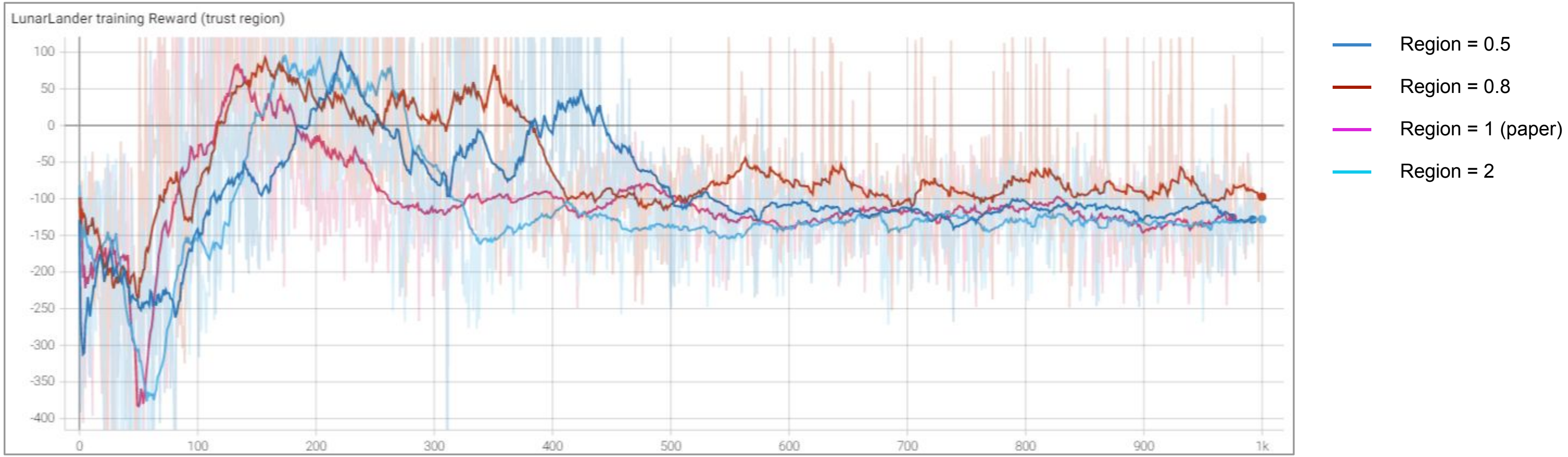
LunarLander-v2

Importance weight truncate :



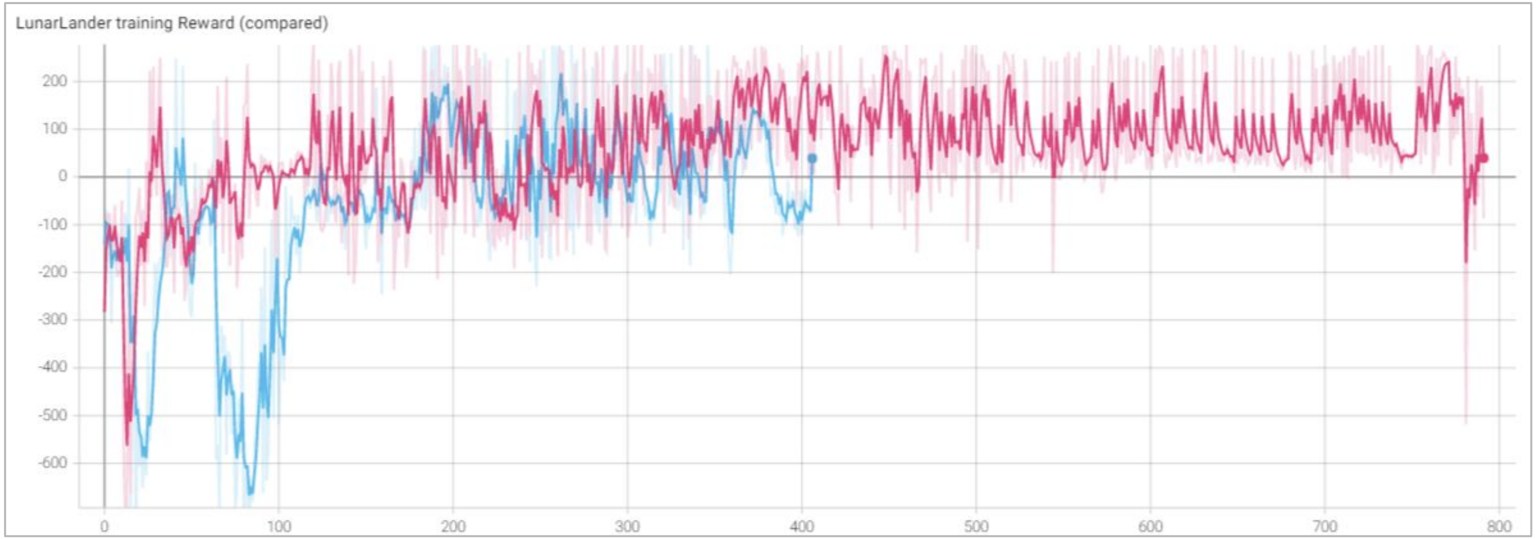
LunarLander-v2

Policy trust region :



LunarLander-v2

Policy trust region :

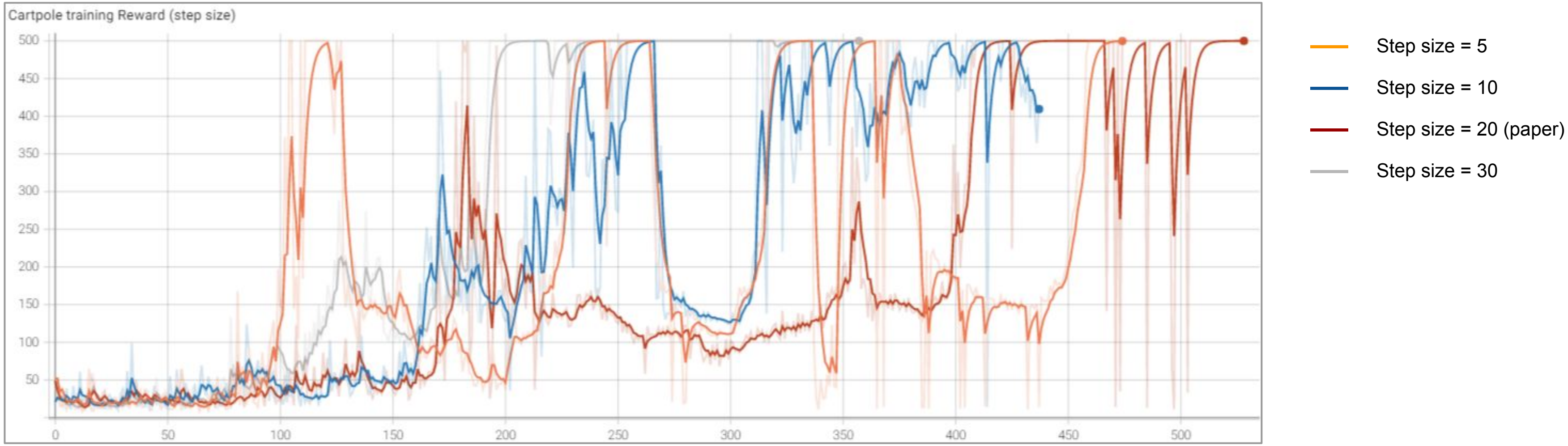


Step size=5 truncate=0.8 trust region=0.8

Step size=5 truncate=2 trust region=0.8

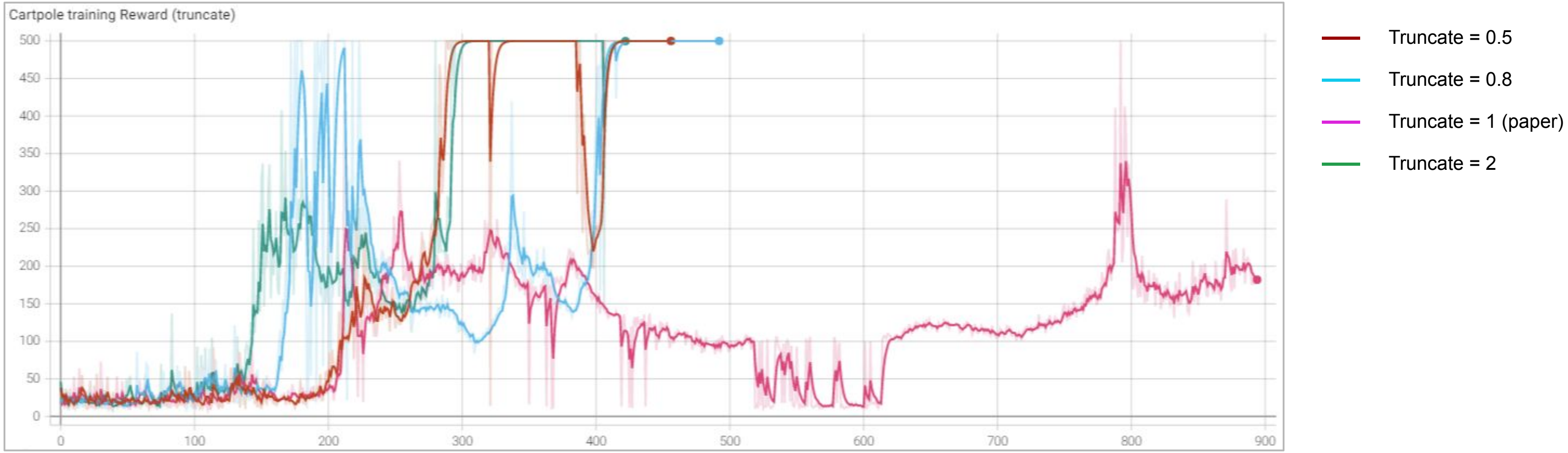
CartPole-v1

Step size :



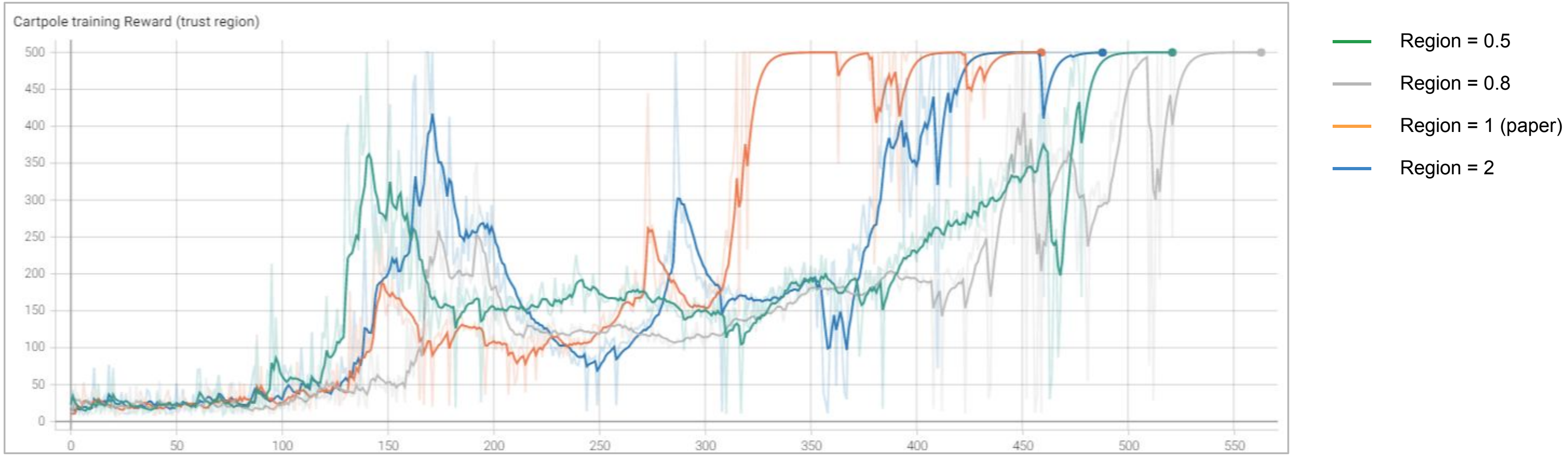
CartPole-v1

Importance weight truncate :



CartPole-v1

Policy trust region :



Outline

- Introduction
- Implementation Details
- Ablation Results
- **Conclusion**

Conclusion

- Novel ideas of ACER
- Technical Implementation
- ACER is sensitive to step sizes
- Better choices of hyperparameters for short-episode training

Thank you for listening!