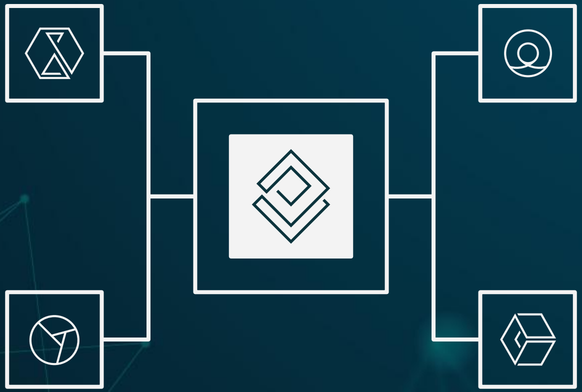
Hyperparameter tuning of the PPO algorithm for OpenAI's CarRacing

Vojtěch Sýkora

Introduction

Deep Learning & Neural Networks

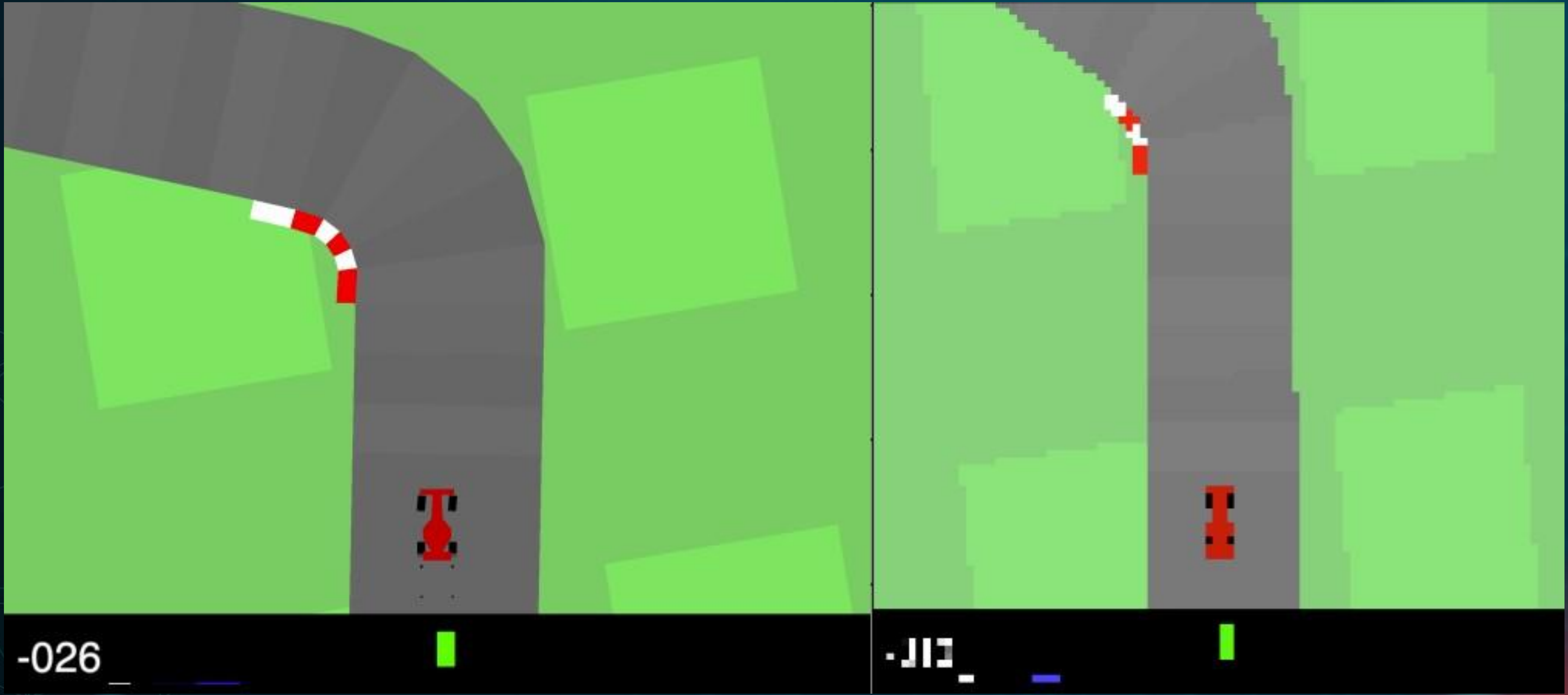
Proximal Policy Optimization



Reinforcement Learning

Autonomous Cars

Car Racing Environment



Left: Render for humans

Right: 96x96 render for agents

Car Racing Environment

- **Action Space**
  - Continuous - There are 3 actions: steering (-1 is full left, +1 is full right), throttle, and breaking
  - Real world physics
- **Observation Space**
  - Image 96x96x3 RGB
- **Reward**
  - -0.1 for every frame
  - +1000/N for every track tile visited. N is the total number of tiles visited in the track.
  - Aims = stay on track & go as fast as possible
  - 900 score is a solved environment

Proximal Policy Optimization Algorithm (PPO)

- **Stable Baseline**
- **Usable for discrete & continuous action spaces**
- **Minimizes loss → maximizes reward**
- **Policy Gradient method**
- **Proximal**
  - Stay close to previous policy
    - Stability
    - Avoid overfitting
    - Improve performance

Policy Gradient Methods

- **Learn Online** (difference from DQN)
- **Do not store past experiences in a replay buffer**
  - Learn directly after each episode
  - Once a memory is used it is discarded
- **PG methods = 1 gradient update per data sample**
  - **PPO = multiple epochs of updates from same data sample**
-

Algorithm 1 PPO, Actor-Critic Style

---

for iteration$=1,2,\ldots$ **do**
  for actor$=1,2,\ldots,N$ **do**
    Run policy $\pi_{\theta_{\text{old}}}$ in environment for T timesteps
    Compute advantage estimates $\hat{A}_1,\ldots,\hat{A}_T$
  end for
  Optimize surrogate L wrt θ, with K epochs and minibatch size $M \leq NT$
  $\theta_{\text{old}} \leftarrow \theta$
end for

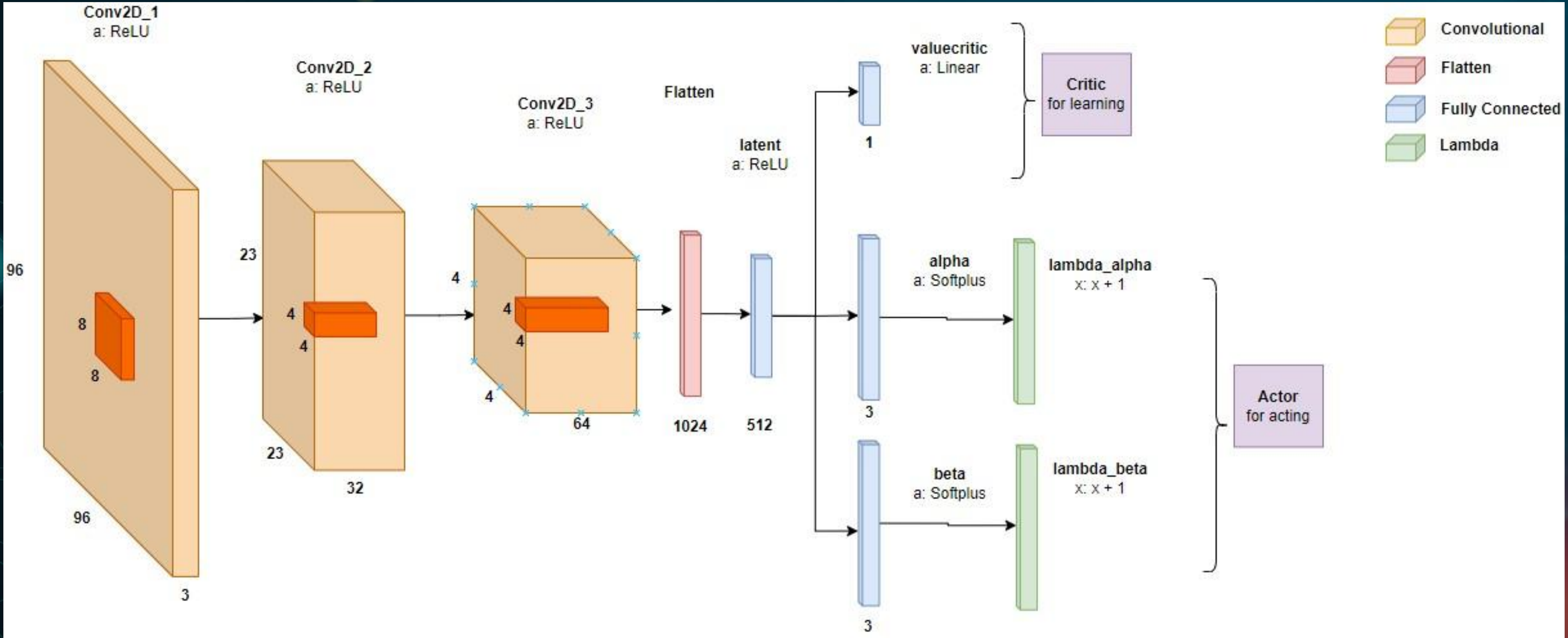
1. **Collect experiences**
2. **Run Gradient Descent on policy network**

Deep Neural Network Structure

- Convolutional Neural Network for image processing
- Actor
  - Estimate actions (using Beta distribution)
- Critic
  - Estimate value of current state

$$f(x, \alpha, \beta) = \frac{x^{\alpha-1}(1-x)^{\beta-1}}{B(\alpha, \beta)}$$

Deep Neural Network Structure



Hyperparameter Tuning

Initial

Our initial hyperparameters were

$$\begin{aligned} \text{horizon} &= 128 \\ \text{mini-batch size} &= 256 \\ \text{epochs per episode} &= 3 \\ \text{gamma} &= 0.99 \\ \text{clipping range} &= 0.2 \\ \text{gae lambda} &= 0.95 \\ \text{value function coefficient} &= 1 \\ \text{entropy coefficient} &= 0.01 \\ \text{learning rate} &= 2.5e{-4} \end{aligned}$$

Experience Collection

Horizon

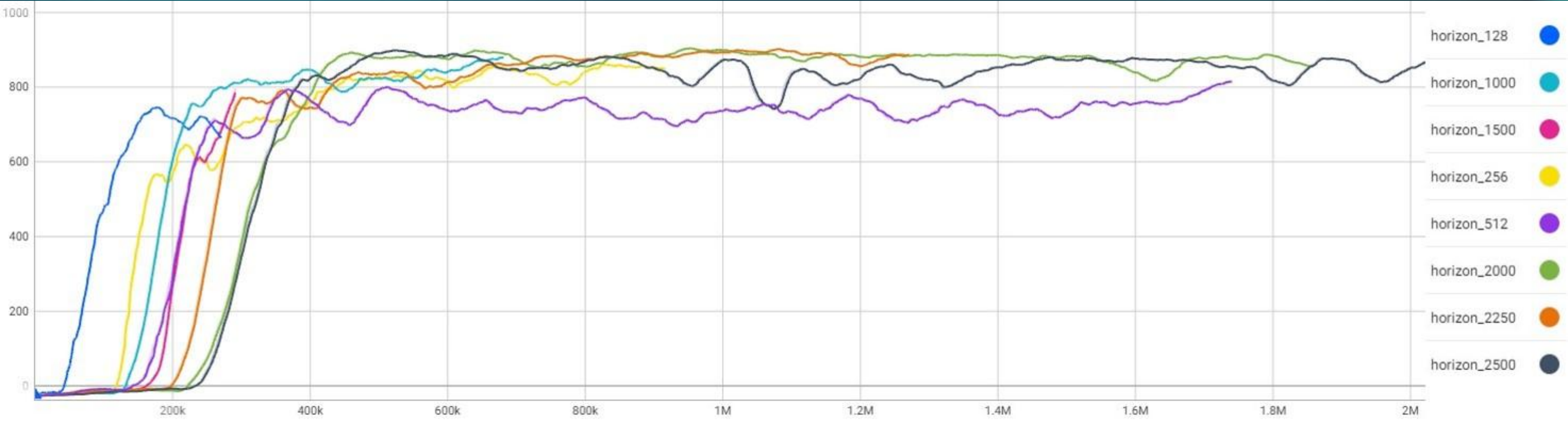
Mini-batch size

Epochs

Horizon (2250)

= The number of steps in each episode

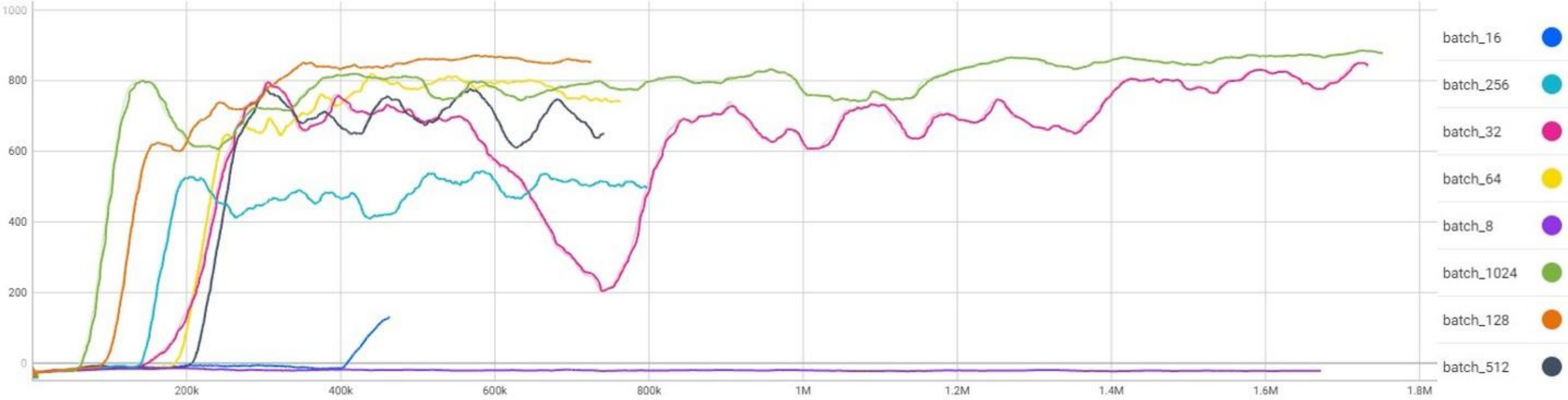
- **Low horizon**
  - Car explores only start of track
  - Learns track in smaller sections
- **High horizon**
  - Car explores turns before it knows how to drive
  - Longer initial training time



Mini-batch size (1024)

= **We optimize using gradient descent using a single batch of experiences at one time.**

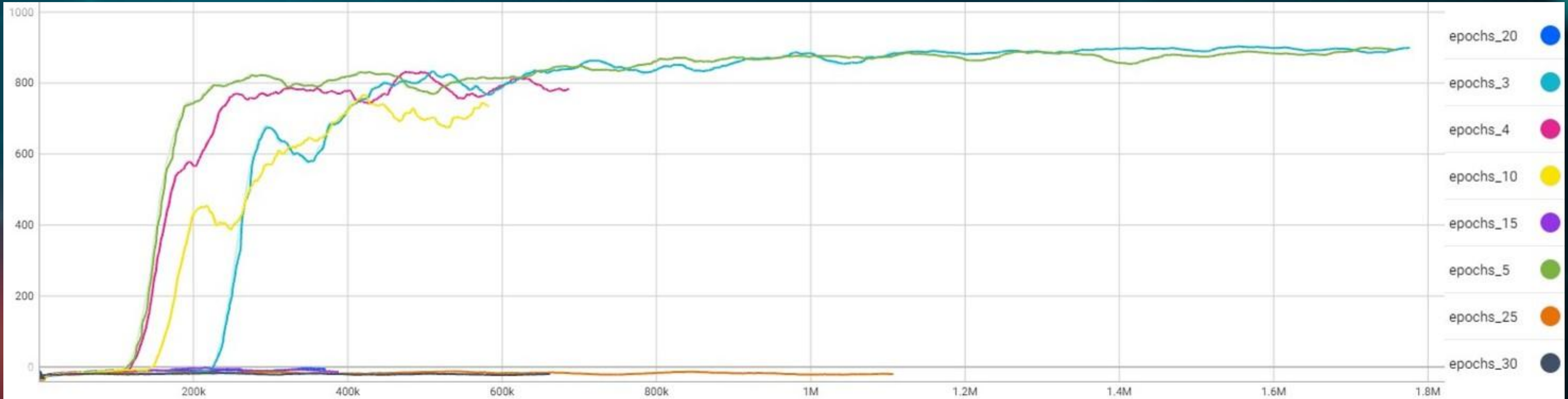
- **Small**
  - Noisy = regularizing effect, lowers generalization error
  - Fits into memory



Epochs (3)

= One cycle through the full training dataset

- **Lot of epochs**
  - overfitting



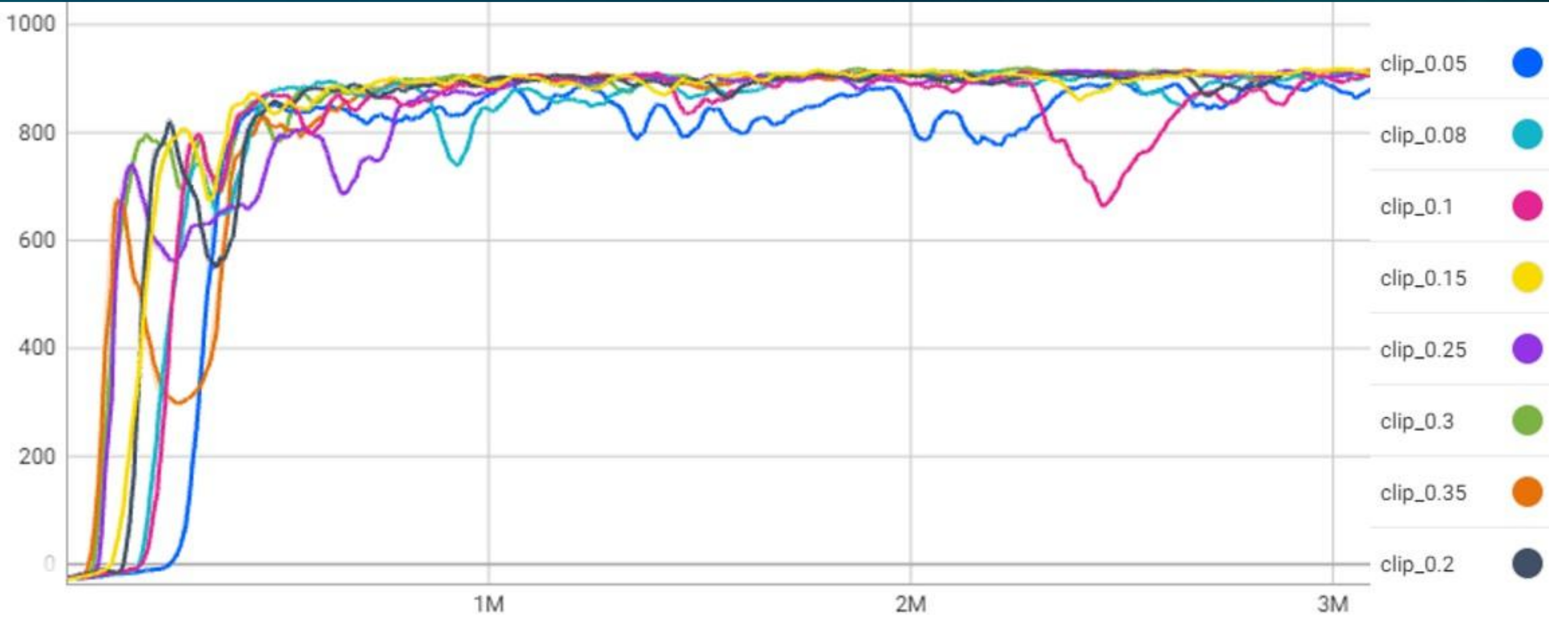
Policy Updating

Clipping Range

Gamma

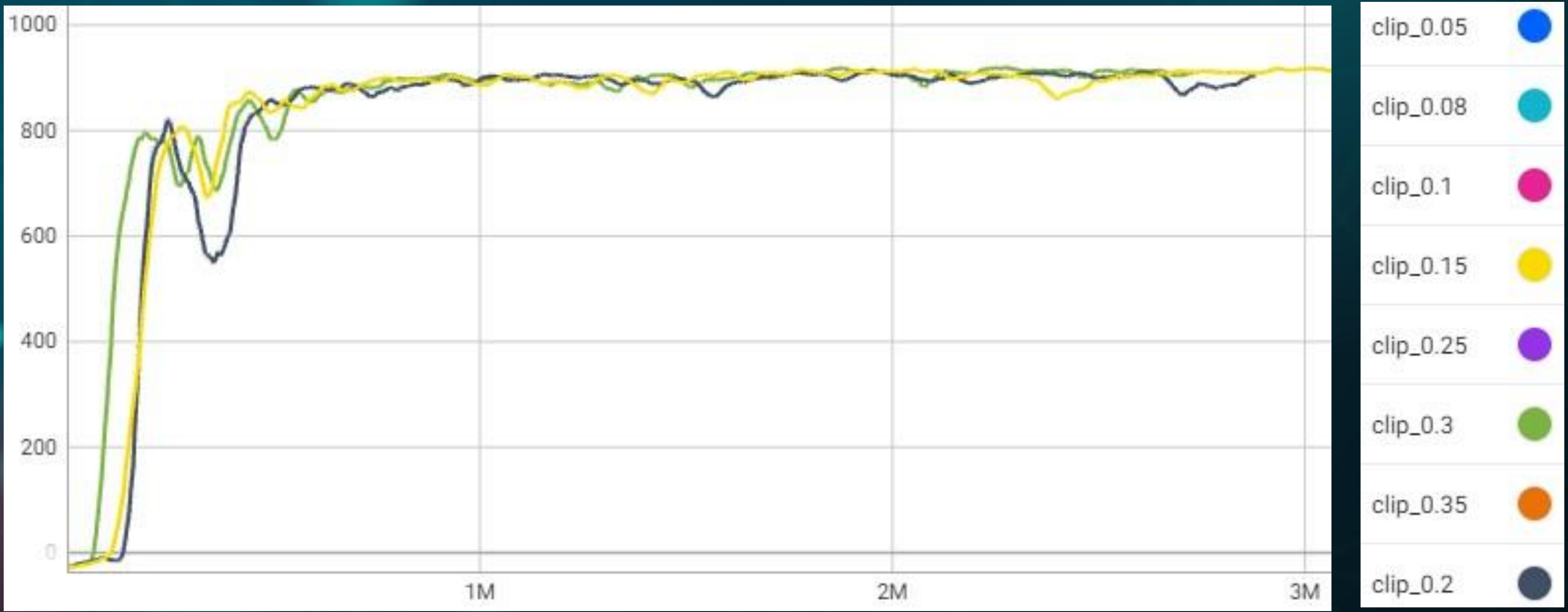
GAE Lambda

Clipping range



Clipping range (0.15)

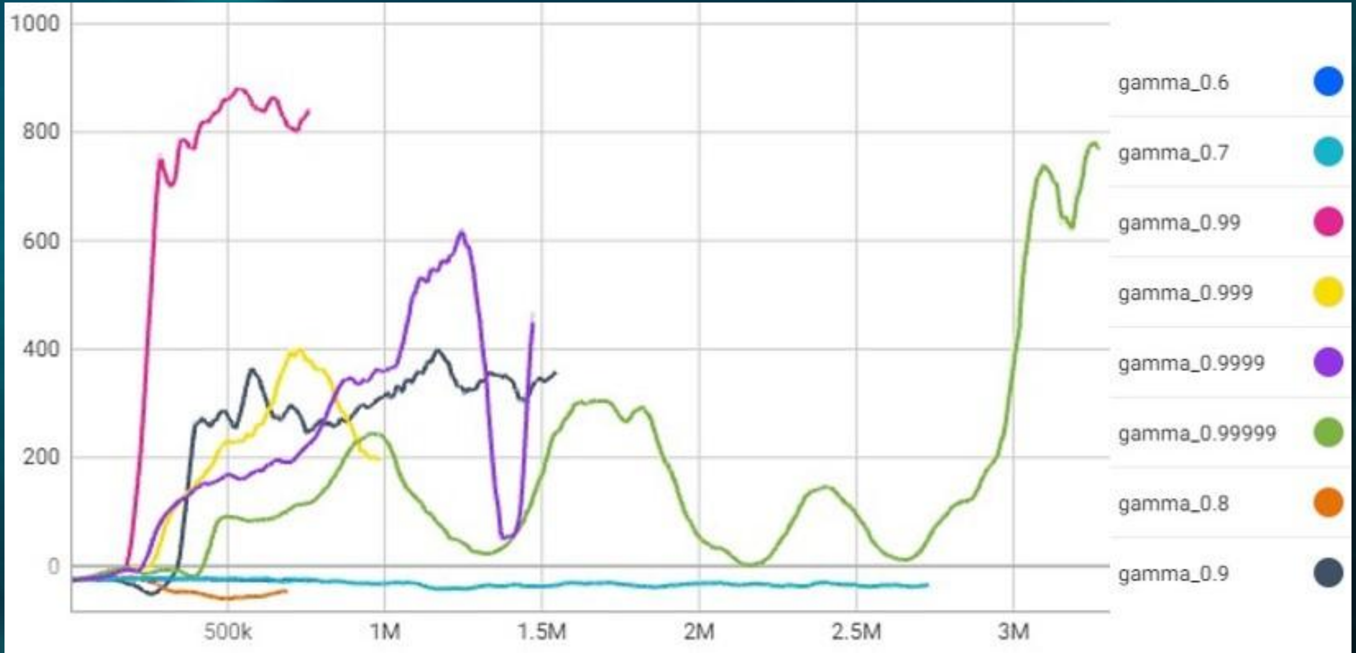
the higher the clipping range, the larger the policy update can be done, which could result in a drastic change in the policy.
To keep the policy stable, a smaller number is often used



Gamma (0.99)

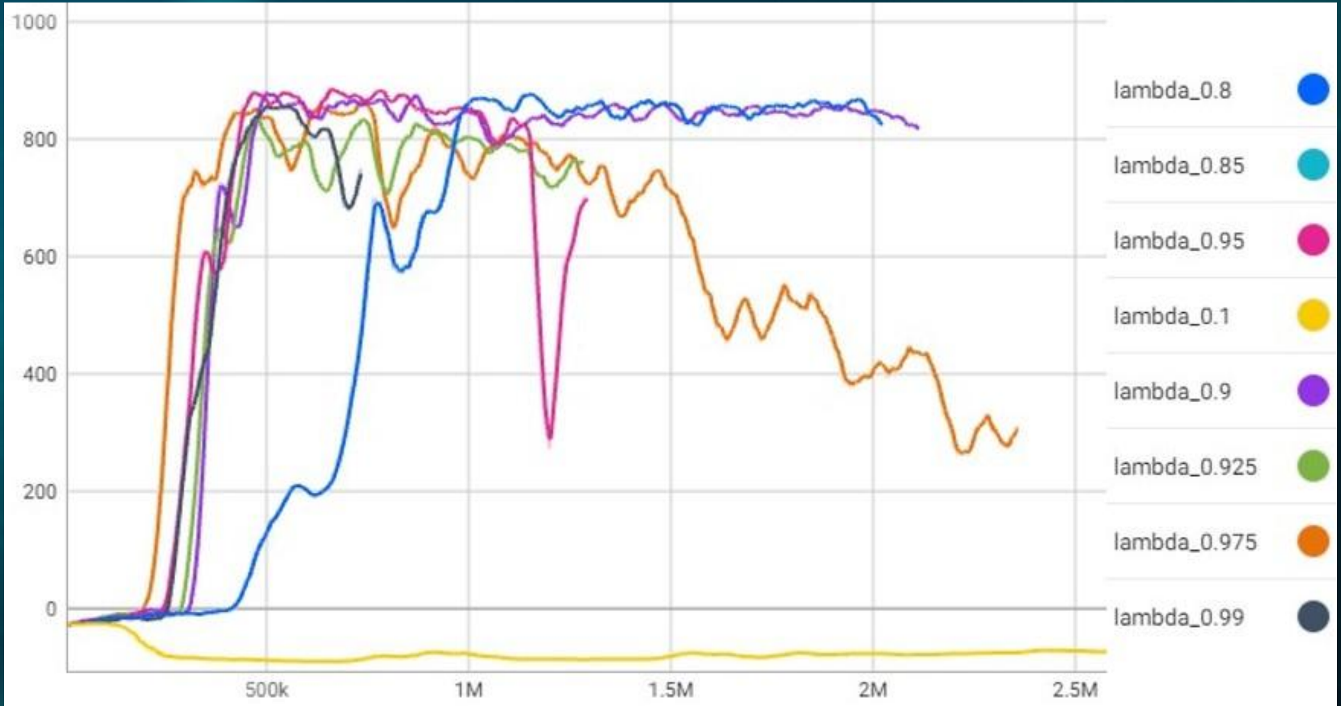
Our agent prefers rewards that it will receive now rather than the same reward further down the line

If we have gamma=0.9, the reward in 6 steps is half as important as the immediate reward, whereas, with gamma=0.99, the reward in 60 steps is half as important as the immediate reward.



GAE Lambda (0.9)

If you want to have a smoother training curve corresponding to training being more stable, choose a λ close to zero. A number close to zero means high bias and low variance, while a number close to 1 means the opposite.



Loss function coefficients

C1

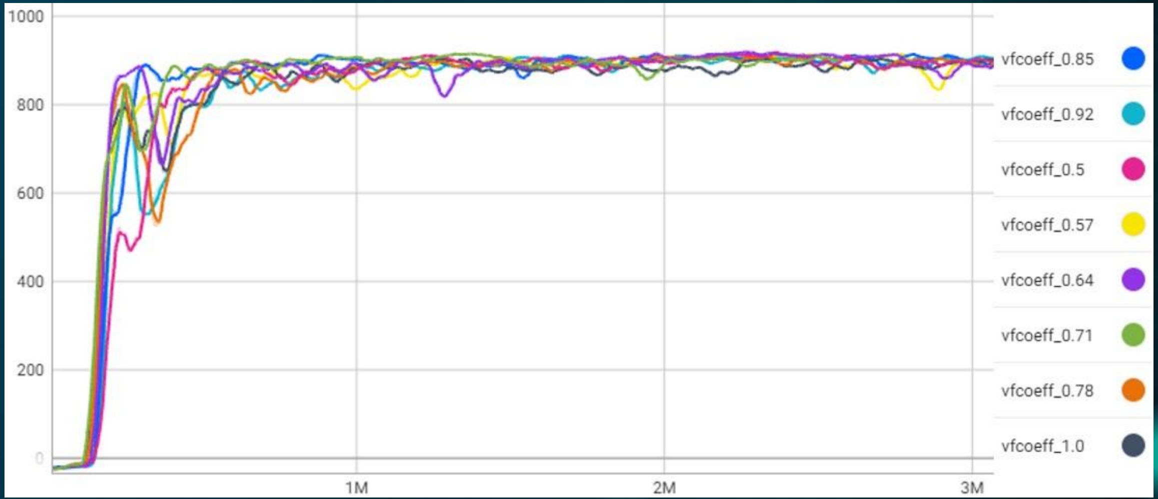
Value Function Coefficient

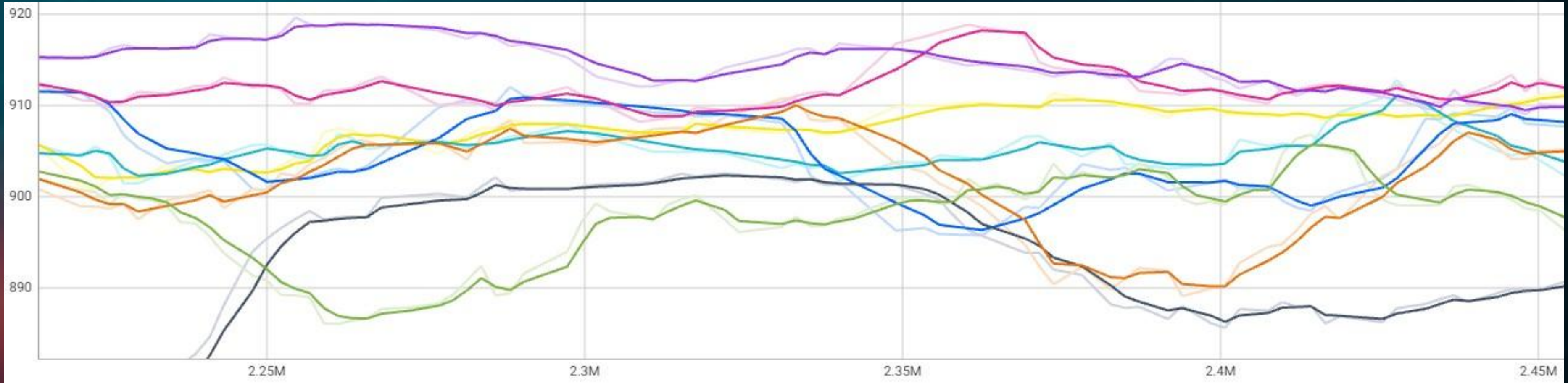
C2

Entropy Coefficient

Value Function Coefficient (0.64)

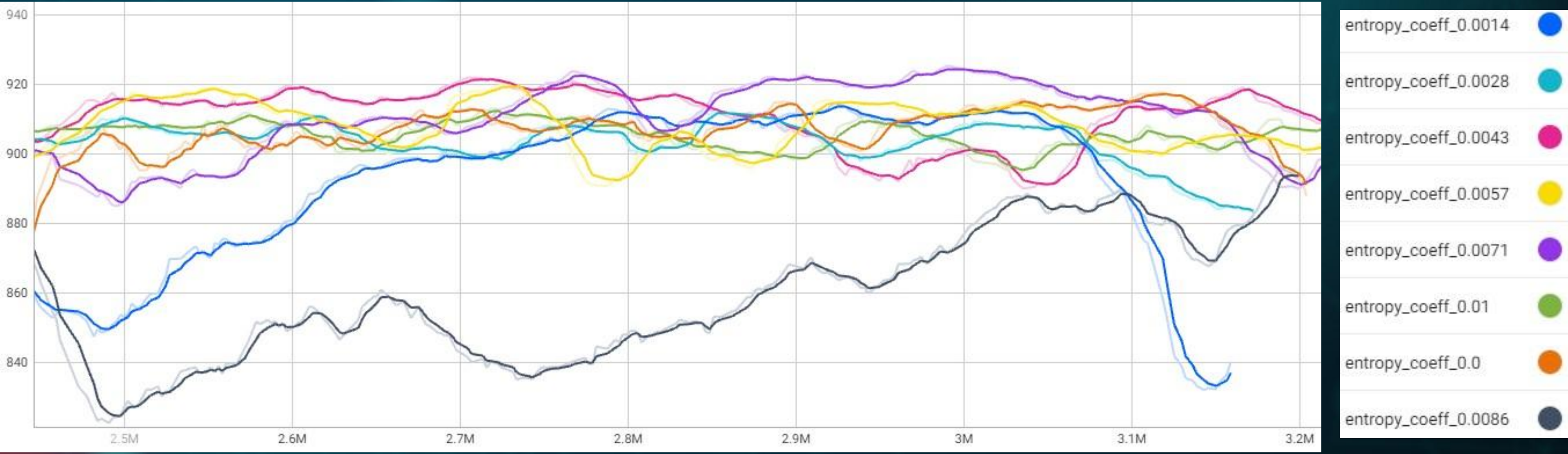
It decides how influential should our prediction, of the value of a state, be.





Entropy Coefficient (0.0071)

helps prevent premature dominance of one action probability over the policy which could prevent exploration. A policy has minimum entropy when a single action has an overly dominant probability.



General

Optimizer learning rate

Terminating Condition

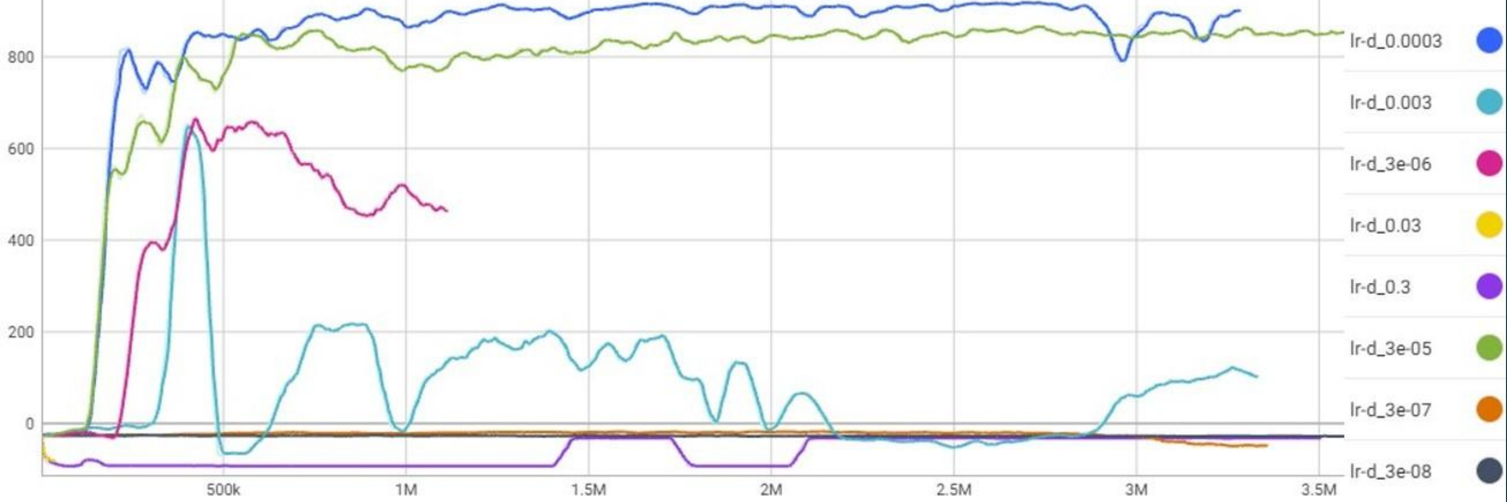
Optimizer Learning Rate (0.0003)

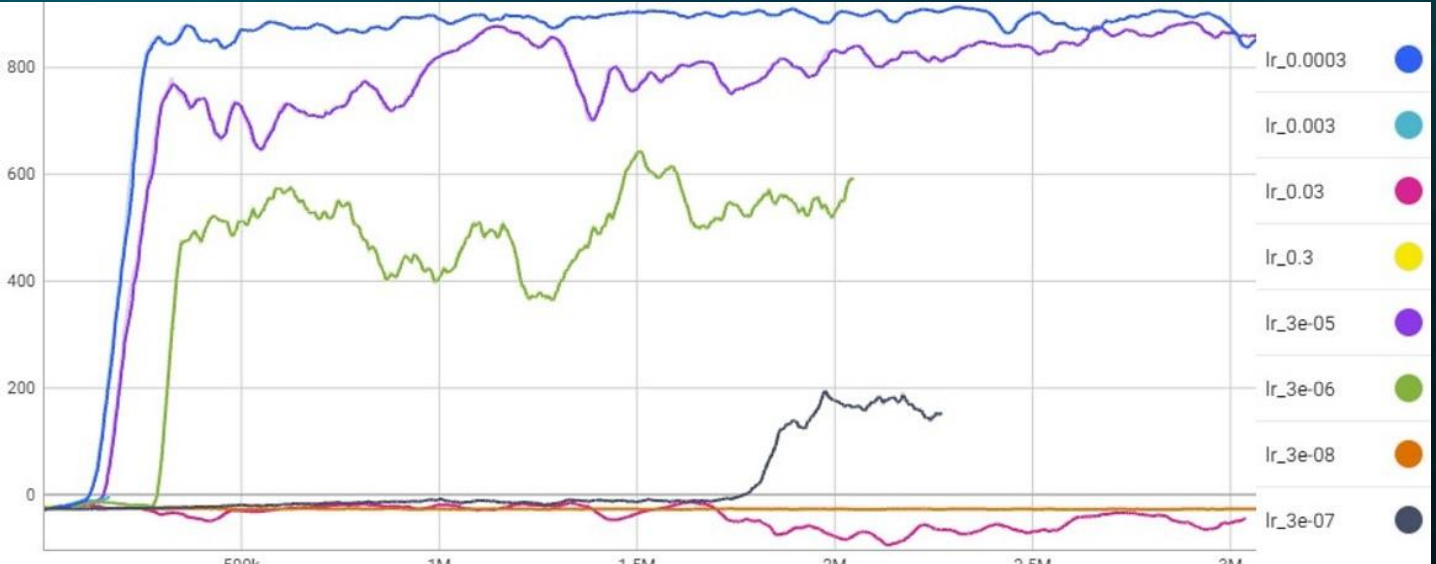
= how large of an impact should the optimizer have during a single update.

- **For our experiment we chose the Adam optimizer**
- **Discounted VS constant learning rate**
- **Discounted changes after each episode**
    - a discounted learning rate we multiplied the initial learning rate by a decreasing number $\left(1 - \frac{\text{current episode number}}{\text{final episode number}}\right)$ which fell linearly from 1 to 0

beginning of training = useful to explore and be able to escape some local minima.

Somewhat good agent = much less desirable to change the policy significantly in a single update.


800
600
400
200
0
500k
1M
1.5M
2M
2.5M
3M
3.5M
lr-d_0.0003
lr-d_0.003
lr-d_3e-06
lr-d_0.03
lr-d_0.3
lr-d_3e-05
lr-d_3e-07
lr-d_3e-08


800
600
400
200
0
lr_0.0003
lr_0.003
lr_0.03
lr_0.3
lr_3e-05
lr_3e-06
lr_3e-08
lr_3e-07

Terminating Condition

- **Environment solving score of 900**
- **We wanted to explore hyperparameters = run as long as possible**
    - **Placeholder 4000 episodes**
        - Because of hardware used

Conclusion

- Deep Reinforcement Learning, Proximal Policy Optimization
- Car Racing - Real life physics, continuous
- 10 Hyperparameters
  - Different impacts on score
  - Explainable occurrences on training graphs
- Environment solved (gained over 900 score)
- Further projects
  - Autonomous driving in more challenging environments
  - Modified CarRacing-v2
    - Wind, obstacles ...

Bibliography

[AAB+15] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dandelion Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng, *TensorFlow: Large-scale machine learning on heterogeneous systems*, 2015, Software available from tensorflow.org.

[BCP+16] Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba, *Openai gym*, arXiv preprint arXiv:1606.01540 (2016).

[KB14] Diederik P. Kingma and Jimmy Ba, *Adam: A method for stochastic optimization*, 2014.

[NMK22] Andrew Ng, Younes Bensouda Mourri, and Kian Katanforoosh, 2022.

[RN10] Stuart J. Russell and Peter Norvig, *Artificial intelligence: a modern approach*, 3 ed., Pearson, 2010.

[SLM+15] John Schulman, Sergey Levine, Philipp Moritz, Michael I. Jordan, and Pieter Abbeel, *Trust region policy optimization*, 2015.

[SWD+17] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov, *Proximal policy optimization algorithms*, CoRR **abs/1707.06347** (2017).

[16] Şenol Çelik and Mehmet Korkmaz, *Beta distribution and inferences about the beta functions*, Asian Journal of Science and Technolog **7** (2016), 2960–2970.