Course Project for Cpt-S 434: Improving Actor Critic Model Efficacy With Mish Activation Function for Reinforcement Learning

Christian Rouhana

Abstract

Reinforcement Learning(RL) is a method of training a neural network or machine learning agent to perform various tasks. While other training methods rely on provided datasets, a reinforcement learning agent must learn from trial and error. This can lead to a very high training time before the agent reaches convergence. This can be made worse by multi-dimensional or noisy data. Because of this, model efficacy has become a more and more important topic in reinforcement learning research. Through this work, I investigate the creation of a tuned actor-critic(AC) hybrid model using the recently developed Mish [5] activation function. It will be compared with a similar algorithm utilizing ReLU. The data collected from this will be used to draw conclusions about the Mish activation function's effect on model efficacy

1 Introduction

Reinforcement Learning has become a popular method of solving control problems. In the most general sense, an RL agent does decisions and policy updates based on various behaviors and responses. Deep Reinforcement Learning (DRL) combines RL with the use of neural networks to deal with the problem of environments with high dimensions of information.

The Actor-Critic Method [3] is an approach to reinforcement learning in which two algorithms, actor and critic, where the critic applies temporal difference learning(TD learning) under a linear approximation architecture, and the actor updates its information through a gradient direction computed from the critic response. Deep Actor-Critic Algorithms have seen great success in recent years. Arulkumaran et al. [1] provide an introduction to DRL algorithms, including deep actor-critic algorithms. As DRL algorithms are used to solve more complex problems, model efficacy has become a more pressing issue. The algorithms have to process a large amount of data, and decrease the amount of passes, or "episodes", a model requires for convergence is essential for minimizing resource usage.

OpenAI Gym [4] is a reinforcement learning environment developed for the purpose of testing and training new algorithms without having to create an environment from scratch. It features many of the famous control problems in AI, 3D control problems, and even Atari games. The 'CartPole-v1' problem is used in this paper's experiment. This model is based off the problem described by Barto, Sutton, and Anderson [2].

The structure of this paper is focused on the application of a new developed application function, 'Mish' [5], in reducing one model's convergence time. In the next section, I elaborate on the background of DL and RL methods pertaining to this project. In section 3, I explain my model and related functions in the OpenAI Gym Cartpole environment. In section 4, I will provide commentary on my experiment and results, then explore potential future work on this subject in section 5.

2 Background

2.1 Reinforcement Learning

In the realm of control problems, models used to be trained with imitation learning. Imitation learning required human input to be fed to the model. This had many limitations. For one, human input could not cover the scope of all situations the model could encounter. Also, it is much more time consuming to develop. Reinforcement learning on the other hand, requires no human input. The model will run the problem repeatedly, adjusting its behavior according to a reward function and any other information it can derive from the environment's state. This is typically based off some sort of discounted episodic Markov decision process (MDP) [9]. The collected reward is discounted by a value between 0 and 1. It has been combined with convolutional neural networks (CNN) to help an agent deal with more complex environments. Recurrent Neural Networks (RNN) have also been used in this application, specifically for natural language processing.

2.2 OpenAI Gym

OpenAI Gym [4] was developed for the purpose of testing reinforcement learning algorithms. These environments come with various sets of tools for training an algorithm. They have predefined reward functions, observations are already organized into various data structures, and it is all returned through a 'step()' function. It also organizes trials into episodes, allowing for easy control loops on the agent side. The paper by Brockman et al.[4] explains this in more detail.

2.3 Actor-Critic Algorithms

Early methods of reinforcement learning were based off single value functions. These would reach convergence, but would result in a high amount of variability and slow training time. Various methods were investigated to help prevent the issue. In this section, I will focus on actor-critic algorithms. Rather than having a single value method, an actor generates an action to carry out, and a critic generates a desired value that the actor will compare and use to adjust its next action. The critic value is usually calculated through a form of temporal difference learning. There are various methods of actor-critic learning. Two of the most popular methods are advantage actor-critic and asynchronous actor

critic. The paper by Bhatnagar et al. [3] provides an in-depth background of actor-critic learning. For this paper, a hybrid actor-critic algorithm is used for its ease of implementation under time constraints. While many actor-critic methods update the actor and critic function separately, a hybrid actor-critic method updates both together. The model will be explained more in section 3.

2.4 Mish Activation Function

Mish is a self-regularized, non-monotonic activation function developed to improve upon the shortcomings of found in other activation functions like ReLU, Swish, and Leaky ReLU. Mish is mathematically defined as [5]

$$f(x) = xtanh(softplus(x))$$

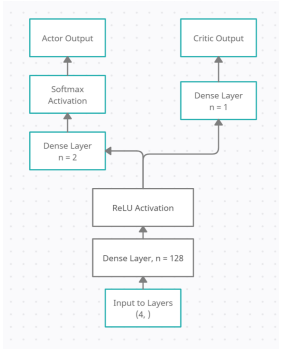
The output of Mish is unbounded above and bounded below, providing a balance of not having too small of gradients and retaining strong regularization effects. A more in-depth explanation of Mish and its advantages can be observed in Diganta Misra's paper [5]. Mish has already been used in new research. In a paper by Haomin Qiu et. al [7], a new Dueling Deep Q Network (Dueling DQN) applies the Mish activation function. In the OpenAI Gym Cartpole game [4], the model achieves a 35.76% accuracy improvement over a traditional dueling DQN.

3 Hybrid Actor-Critic Model with Mish

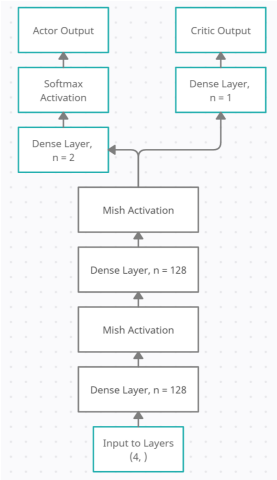
A hybrid Actor-Critic model was chosen for this project because of its simple implementation and modular nature. The model is a modified version of the implementation defined in the Keras documentation [6]. The model is outlined in Figure 1. The Keras implementation utilized a hidden value of $n = 128$ and a learning rate of $\lambda = 0.01$ for their implementation. For the Mish implementation, a hyperparameter tuning script was run and it was found that hidden value $n = 128$ and $\lambda = 0.003$ performed the best. Tested parameters included hidden values $n = 64, 128, 256, 512, 1024$ and learning rates $\lambda = 0.001, 0.003, 0.01, 0.1$. Values such as average reward step size and discount were not adjusted due to time constraints. Average model reward is updated through the equation [3]

$$\hat{J}_{t+1} = (1 - \xi_t)\hat{J}_t + \xi_t r_{t+1}$$

Where $\hat{J}_{t+1}$ is the running reward, ξ_t is the reward step size, $\hat{J}_t$ is the episode reward, and r_{t+1} is the episode reward. A step size of $\xi_t = 0.05$ is retained from the Keras implementation. Adam was held constant as the optimization function as well. Stochastic Gradient Descent was tested but showed poor initial results and was not pursued further. Both models use Huber loss. MeanSquaredError $loss = (y_{true} - y_{pred})^2$[8] was tested as well. Huber was retained due to its handling of outliers that serves reinforcement learning very well[6]. The actor and critic losses are calculated separately but are summed before the application of gradients and optimization function.



(a) Keras Model



(b) Mish Implementation

Figure 1: Diagrams of Each Model's Neural Network

In regard to the environment, 'CartPole-v1' in OpenAI Gym was selected. It is a continuous environment in which any step observation made by the model is a Box(4) of 4 values

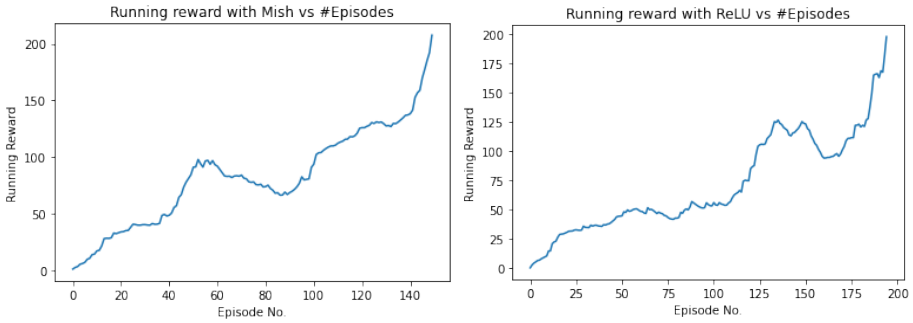
- Cart Position $-4.8 <= pos <= 4.8$
- Cart Velocity $-\infty <= vel <= \infty$
- Pole Angle $-0.418rad <= \theta <= 0.418rad$
- Pole Angular Velocity $-\infty <= vel_\theta <= \infty$

The agent can only do two actions, push the cart left (0) or right (1). The reward function is simply, $r+ = 1$ for every step taken where the pole does not fall. Episode termination occurs when the pole angle is above 12 degrees, the cart reaches the end of the screen, or episode length is greater than 200. the OpenAI Gym leaderboard says for CartPole, a model has solved the problem when it achieves an average score greater than or equal to 195 for 100 consecutive trials.

4 Experiment Design and Results

First, the Mish model from Figure 1 was trained and tested once to demonstrate its ability to solve the CartPole problem. the isolated trial reached convergence in 150 episodes on seed 42. Also, given a max possible score of 250 to save training time, it achieved an average score of 250 episodes over 100 trials (given random seed of 43). This was repeated with an equivalent ReLU model. Hyperparameter tuning ended up identifying the same parameters of hidden

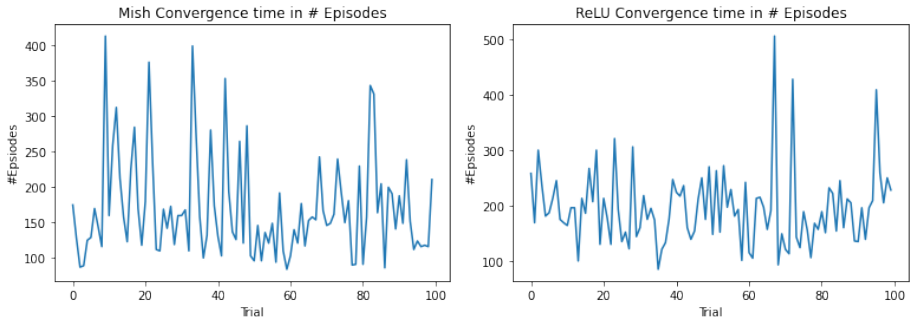
value $n = 128$ and learning rate $\lambda = 0.003$ for the ReLU model. The model was trained on seed 42 in 195 episodes and tested on a random seed 60. It achieved an average score of 229.61 over 100 trials. Figure 2 displays the convergence line of each of these trials. To adequately compare the two, each model is trained



(a) Mish Model (b) Relu Model

Figure 2: Individual Trial of Each Network

from scratch 100 times on the same seed. Both tests were ran on the same hardware, but number of episodes was used to retain consistency. Runtime's main factor in this environment is the number of episodes, as the actor must attempt to balance the pole for the full duration regardless. Seen in Figure 4, both



(a) Mish Model (b) Relu Model

Figure 3: Convergence Time(Episodes) per Trial

data sets demonstrate a strong right skew. Yet the chart for ReLU displays a a higher median than the Mish Model. The charts in Figure 3 show that the Mish model seems less consistent, but generally stays in a lower convergence count than the ReLU model. Some simple statistics in Figure 5 support these claims. The model utilizing Mish scored a 7.1% lower mean convergence rate, a 1.19% better peak convergence time, and a 10.12% better worst convergence time. Yet it had a slightly worse standard deviation at 71.99 Episodes vs the

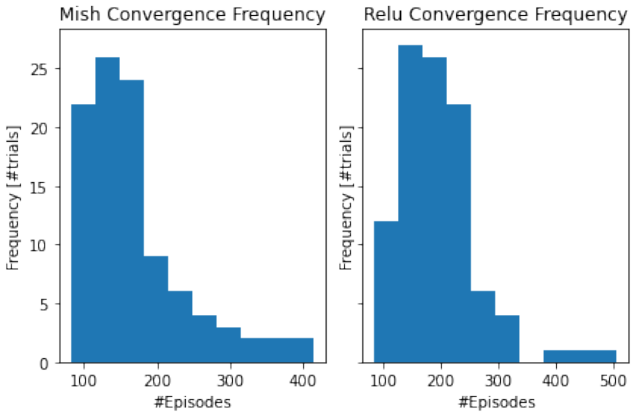


Figure 4: Convergence Time(Episodes) Frequency(No. of Trials)

Function	Avg Convergence	Fastest Convergence	Slowest Convergence	Standard Deviation
Mish	168.82 Eps	83 Eps	413 Eps	71.99 Eps
ReLU	194.63 Eps	85 Eps	506 Eps	67.65 Eps

Figure 5: Statistics for 100 Trial Test

ReLU model's 67.7 Episodes. Across the board however, Mish outperformed ReLU in terms of speed for this model. Mish may be slightly less reliable than ReLU with the current standard deviation, but the data supports that Mish is performing more efficiently per episode overall.

5 Discussion & Future Work

The data presented in section 4 gives clear evidence that Mish has great potential to improve model efficacy when used over ReLU in a reinforcement learning environment.Future work must be carried out to solidify these results. The Actor-Critic method was used for its simplicity of implementation due to time constraints. This experiment would be far more conclusive if it was tested with more current models such as asynchronous actor-critic, advantage actor-critic, dueling DQN, etc. Also, if these tests were run in multiple environments vs multiple variants of ReLU and other popular activation functions, this would be a truly conclusive experiment.

References

[1] Kai Arulkman et al. "A Brief Survey of Deep Reinforcement Learning". In: *IEEE Signal Processing Magazine* (). URL: https://arxiv.org/pdf/1708.05866v1.pdf.

[2] A.G. Barto, R.S. Sutton, and C.W. Anderson. "Neuronlike adaptive elements that can solve difficult learning control problems". In: *IEEE Transactions on Systems, Man, and Cybernetics* SMC-13 (), pp. 834–846. URL: https://ieeexplore.ieee.org/document/6313077.

[3] Shalabh Bhatnagar et al. "Natural Actor-Critic Algorithms". In: *Automatica, Elsevier* (2009). URL: https://hal.inria.fr/hal-00840470/document.

[4] Greg Brockman et al. *OpenAI Gym*. URL: https://arxiv.org/abs/1606.01540.

[5] Diganta Misra. "Mish: A Self Regularized Non-Monotonic Activation Function". In: *British Machine Vision Virtual Conference* (2020). URL: https://www.bmvc2020-conference.com/assets/papers/0928.pdf.

[6] Apoorv Nandan. *Actor Critic Method*. Keras. URL: https://keras.io/examples/rl/actor_critic_cartpole/.

[7] Haomin Qiu and Feng Liu. "A State Representation Dueling Network for Deep Reinforcement Learning". In: *2020 IEEE 32nd International Conference on Tools with Artificial Intelligence (ICTAI)*. 2020, pp. 669–674. DOI: 10.1109/ICTAI50040.2020.00107.

[8] *Regression Losses*. Keras. URL: https://keras.io/api/losses/regression_losses/.

[9] Kun Shao, Zhentao Tang, and Yuanheng Zhu. *A Survey of Deep Reinforcement Learning in Video Games*. URL: https://arxiv.org/abs/1912.10944.