Using Pre-Training to Speed Up Deep Reinforcement Learning

George-Kirollos Saad
Supervisor: Prof. Michael Guerzhoy

April 2023

B.A.Sc. Thesis

Division of Engineering Science
UNIVERSITY OF TORONTO

ABSTRACT

This study investigates the effect of neuron activation entropy thresholding and gradient-weighted entropy thresholding on sample efficiency in deep reinforcement learning (DRL) networks using advantage actor-critic (A2C) algorithms with Tanh and ReLU activation functions in CartPole environments. DRL combines neural networks and reinforcement learning to optimize intelligent agents for decision-making. However, DRL typically requires large amounts of iterations and training many agents, which can be computationally expensive.

The research explores the potential benefits of neuron activation entropy thresholding, gradient-weighted entropy thresholding, and guided-backpropagation techniques in improving sample efficiency. Experiments were conducted on both Tanh and ReLU activation functions to build an optimized intelligent agent and evaluate the sample efficiency boost.

The results indicate that neuron activation entropy thresholding and gradient-weighted entropy thresholding can potentially improve sample efficiency in A2C networks with ReLU activations, particularly in the CartPole-v0 environment. Although findings for Tanh activations were inconclusive, the experiments involving ReLU activation functions demonstrated promising results, with significant boosts in sample efficiency on both the training and test sets. Guided-backpropagation on ReLU models further reinforced this observation, indicating even more significant improvements in sample efficiency.

This study provides valuable insights into the potential benefits of neuron activation entropy thresholding and gradient-weighted entropy thresholding for improving sample efficiency in DRL models, leading to faster convergence in real-world applications. Future research could explore other thresholding techniques, visualization methods, and environments to gain a deeper understanding of factors influencing reinforcement learning model performance.

Acknowledgements

I would like to thank Prof. Michael Guerzhoy for his support and advice throughout this year while working on this thesis project. I would also like to thank my family and friends for their support.

Contents

List of Figures

List of Tables

1 Introduction

Deep reinforcement learning (DRL) applies the benefits of neural networks to reinforcement learning (RL), with the goal of training agents to make action decisions in order to maximize rewards [12]. A key feature of deep learning is its ability to extract high-level features [8]. However, in order to do this, it typically requires large amounts of training data which sparse reward RL problems don't provide. Mnih et al.'s work, which will be discussed in more details in the Literature Review, has shown that deep learning can still be effective in these RL environments, through a successful implementation on seven Atari 2600 games. In order to build upon this DRL research, this paper aims to experiment on the OpenAI Gym CartPole [2] environment, due to its dense reward system.

The CartPole environment is composed of a vertical pole attached to a cart that moves horizontally along a track [2]. There are 2 possible actions: pushing the cart to the left (0) or to the right (1). The goal is to continually push the cart in either direction (with fixed force) to keep the pole balanced. The episode terminates if the pole exceeds an angle of $12°$ in either direction or if the cart moves off the edge of the display (at ± 2.4 units). The episode is successful if it does not terminate after 200 steps for the v0 environment or 500 steps for the v1 environment. This is visualized in Figure 1:

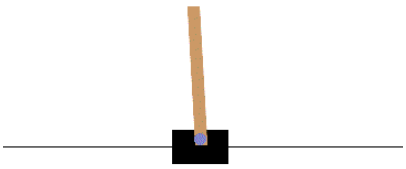

Figure 1: Visualization of the CartPole Envrionment [2].

Running through environment simulations is a time-consuming part of the training process, therefore exploring ways to reduce the number of simulations that need to be run to successfully train the DRL model would be beneficial. This can be measured by calculating the sample efficiency (SE):

$$SE = \frac{\text{\# of successes}}{\text{\# of total iterations}} \tag{1}$$

We are particularly interested in determining the sample efficiency boost relative to a baseline which is measured as:

$$SE_{boost} = (\frac{SE_{thresh}}{SE_{base}} - 1) \times 100 \tag{2}$$

where SE_{thresh} is the SE when thresholding is applied and SE_{base} is the SE at the baseline we are comparing too, which varies depending on the experiment.

This research aims to investigate methods to train a deep reinforcement learning network to identify important features, in order to optimally select actions using a policy in a general reinforcement learning environment. My hypothesis is that guided back-propagation (GBP) can help identify the relevant neurons that correspond to important features. Therefore, the primary objectives are to:

1. Build an optimized intelligent agent that uses GBP alongside gradient-weighted neuron activation entropy thresholding.
2. Evaluate the SE of this agent and achieve an improvement in SE relative to previous thesis work by Michael Ruan (on the CartPole environment) [12].
3. Evaluate the SE of the agent on other environments to ensure the solution applies more generally.

We aim to use neuron activation entropy thresholding alongside GBP to help identify important neurons in the network. This thesis will build off of Michael's work by applying GBP instead of the existing back-propagation step. Then, the gradient with respect to each neuron (after GBP) will be used to weight the neuron activation entropies before thresholding. A further step of exploration, if the GBP method is found to be successful, would be to combine this with action entropy thresholding to try to maximize the SE gain.

2 Literature Review

In order to better understand the current literature with respect to solving RL problems, it is important to understand the existing algorithms and what their advantages and shortcomings are, which this section aims to outline. Furthermore, understanding the role of entropy and the insight it can provide when applied to neural networks can help identify potential solutions to speed up deep reinforcement learning. Existing work in the field, particularly previous thesis work by Michael Ruan and Daniel Pinheiro Leal on this topic, can provide background on what has previously been attempted and either succeeded or failed.

2.1 Reinforcement Learning Algorithms

Given the unavailability of the reward function and state-transition probabilities, the Cartpole problem can be approached using model-free learning algorithms [9]. This can be further divided into on-policy and off-policy algorithms. In on-policy algorithms, the agent uses the same policy to generate action value estimates and select actions. In off-policy algorithms, one policy is used to generate the action value estimates and another to select the actions. The following sections will focus on exploring a range of such algorithms.

2.1.1 Q-Learning

Q-Learning is an off-policy temporal difference (TD) learning algorithm [8]. Of specific interest here is the deep Q-learning network (DQN). It involves creating a deep neural network (the Q-function) and using experience replay to generate the action-value estimates. This involves storing the experiences at each time step and using that to update the network's parameters. The equation for updating the Q-function is as follows [13]:

$$Q(s,a) = Q(s,a) + \alpha[R + \gamma * \max_{a_{max}} Q(s', a_{max}) - Q(s,a)] \tag{3}$$

where Q is the Q-function, s and a are the current state and action, s' is next state, a_{max} is the action that maximizes the Q-function, α is the learning rate and γ is the discount factor.

An advantage of using experience replay is that the behavior of the agent is based on an average of the previous states, which limits divergence and oscillation. An ϵ-greedy approach is used for action selection, which means that there is a 1-ϵ probability of using a greedy selection and an ϵ probability of a random action being selected in order to allow for exploration of new actions.

2.1.2 SARSA

State-Action-Reward-State-Action (SARSA) is an on-policy TD control algorithm [13]. The Q-function update rule for SARSA can be summarized by the following equation:

$$Q(s,a) = Q(s,a) + \alpha[R + \gamma * Q(s', a') - Q(s,a)] \tag{4}$$

where Q is the Q-function, s and a are the current state and action, s' and a' are the next state and action according to the policy, α is the learning rate and γ is the discount factor.

The main distinction between SARSA and Q-learning is that in SARSA, the next action used in the Q-function update is determined by the policy, whereas in Q-learning, the next action is based on whichever action maximizes the Q-function.

2.1.3 Actor-Critic

Actor-critic algorithms are off-policy algorithms. The critic follows a temporal difference (TD) learning algorithm to estimate the values of states and actions by projecting the value function onto the span of the feature subspace [4]. Then, the actor uses the critic's result to update its policy. The advantage is calculated using the Q-function and is used to estimate the advantage of taking a particular action in a given state [7]. The advantage function can be expressed as follows:

$$A(s,a) = Q(s,a) - V(s) \tag{5}$$

where Q is the Q-function and V is the value function.

Figure 2 is from the book "Deep Learning and the Game of Go" [10] and it demonstrates that the role of the advantage function is to reinforce actions that are estimated to be low value but turn out to be useful for successful completion. In the opposite case, if the agent was performing high estimated value actions and suddenly took a turn and lost the game, those actions would have a low advantage, reinforcing that the moves that lead to the downward turn disadvantaged the agent.

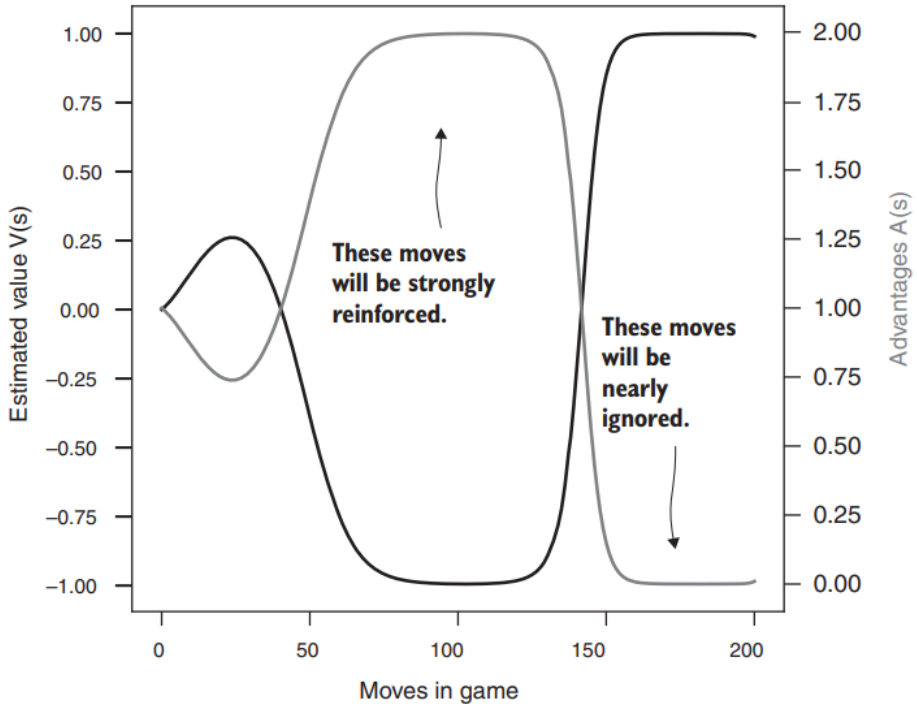

Figure 2: Advantage and Estimate Value for Hypothetical Game [10].

2.1.4 Comparison

The “Asynchronous Methods for Deep Reinforcement Learning” paper compared 4 asynchronous algorithms: one-step Q-learning, one-step SARSA, n-step Q-learning, and advantage actor-critic [7]. The Asynchronous Advantage Actor Critic (A3C) was notably successful on continuous motor control problems, such as the CartPole problem [7], which was found to outperform the other 3 algorithms explored. Given the computational cost and resources required to implement A3C, A2C was implemented as part of this paper since it follows the same idea but had a more reasonable computational cost.

2.2 Entropy

Entropy is used in the field of information theory to define information gain. The formula used is as follows [5]:

$$H(X) = -\sum_{k=1}^{K} P(x_k) * log_2(P(x_k)) \tag{6}$$

where K is the number of data points and $P(x_k)$ is the proportion or probability of data point x_k.

Determining how entropy can be used and methods to calculate it in the context of deep reinforcement learning is the focus of this section.

2.2.1 Action Entropy

One use of entropy in DRL is to promote exploration in policies. In a paper by Mnih et al. [7], they added the policy's entropy as a regularization term to the objective function, weighted by a factor β. This allowed for more exploration before converging on a policy while using the A3C algorithm. The additional entropy regularization term is as follows:

$$\beta \nabla_{\theta'} H(\pi(s_t; \theta')) \tag{7}$$

where θ' represents the policy parameters, π is the policy, s_t is the state, and H is the entropy.

Another paper that explored the use of entropy to promote exploration in DRL is the "Entropy-Aware Model Initialization for Effective Exploration in Deep Reinforcement Learning" paper by Jang and Kim [3]. Their findings are that low initial entropy leads to more failure in learning since it inhibits exploration. They devised an entropy-aware model initialization strategy to counteract this, which involves repeatedly generating models and measuring initial entropy until a certain threshold is exceeded. This method allowed for faster and higher performance RL.

2.2.2 Neuron Activation Entropy

In a paper by Mitra and Franco [6], they explored how to evaluate unsupervised data adaptation in deep neural networks for speech recognition. Since much of the data in this space is unseen, unsupervised adaption is used to help improve the model at processing unseen data. This resulted in data variability which was propagated as distortions from layer to layer in the network. They found that a good measure for this distortion is the neuron activation entropy over a time window. More specifically, the average neuron entropy for each neuron was calculated and the mean of the top 70^{th} percentile of these values was used as a normalized and ranked summary entropy measure (NRSE). The conclusion was that a higher NRSE score was correlated with a worse unsupervised data adaption hypothesis. Therefore, this paper demonstrates the potential for the use of neuron activation entropy as a measure of a how well a model will perform based on a hypothesis.

2.2.3 Histogram Estimator

One approach to calculating entropy involves creating a histogram with K bins that span the range of the data. The relative frequency for each bin can be used as the probability values. Wallis notes in his paper [14] that a correction should be made to the calculation of entropy when the bin widths are not constant. This would likely be the case in the use case here, so the suggested entropy equation that should be used is as follows:

$$H(X) = -\sum_{k=1}^{K} P(x_k) * log_2(P(x_k)) + log_2(w) \tag{8}$$

where w is the width of the bins and all other variables are as defined in Equation 6.

2.3 Guided Back-Propagation

Guided back-propagation (GBP) works by setting all negative gradients of the ReLU activations to 0 during the back-propagation step [11]. It is typically used as a debugging tool for neural networks, due to its ability to visually emphasize significant features in the input of a network (such as images). Rosynski, Kirchner, and Valdenegro-Toro found that, although guided back-propagation doesn't visualize learned models well, it can visualize deep reinforcement learning policies well, since it can strongly emphasize the key features for the agent's action selections. Therefore, since the neurons in the network can model latent features, guided back-propagation may be able identify the useful neurons before applying thresholding.

2.4 Previous Thesis Work

Some previous work in this area was done by Daniel Pinheiro Leal (EngSci 2T0 + PEY) and involved using a deep Q-learning network (DQN) with state representation learning (SRL) and achieved a 22% increase in SE.

This work was further built upon by Michael Ruan (EngSci 2T1 + PEY) by performing action entropy thresholding with SRL, resulting in a 19.78% increase in SE over Daniel's results. Both of these experiments were done on OpenAI Gym's CartPole-v0 environment. Michael also explored performing neuron activation entropy thresholding, however this resulted in negligible improvement in SE from his experimentation. To attempt to improve this result, the gradients with respect to the neurons were used as weights for the neuron activation entropies before thresholding, however this also did not yield any improvement.

In order to evaluate the entropy of the activations, Michael fit the Tanh activation output to a 0-mean Gaussian curve and used that as the probability density function. This makes an assumption about

the distribution of the activation values as being Gaussian which is inaccurate, so we will fix this by following the histogram estimator method.

2.5 Conclusion

From the exploration of various reinforcement learning algorithms, we find that the A2C algorithm is most appropriate and performs the best on dense reward environments like CartPole [7]. We also note that entropy can be useful in promoting exploration in action selection, allowing for more accurate and faster model convergence [7] [3]. We also see that neuron activation entropy was used in deep neural networks (but not in DRL specifically) to measure model performance based on a hypothesis [6]. We find a technique to measure entropy by fitting the data to a histogram and explore important considerations that must be made with respect to bin width [14]. We also saw how GBP was used as a visualization technique in DRL and how the insight it provides to identify key features can potentially be applied to neuron activation entropy thresholding [11]. Lastly, this thesis aims to build on top of the exploration that Michael has done in his thesis, which used the A2C algorithm and applied action and neuron activation entropy thresholding with gradient weighting [12].

3 Experimentation Setup and Methodology

This section outlines the experimental setup, encompassing the model architectures employed, the approach to calculating neuron activation entropy and gradient weighting, as well as the methodology for entropy thresholding. These details are vital in providing a comprehensive understanding of the environment in which the models were investigated and how the experiments were conducted.

3.1 Environment & Architecture

Experimentation was performed on both the CartPole-v0 and CartPole-v1 environments, using the A2C algorithm from the Stable Baselines3 package [1]. The model architecture used was a 2-layer, 64x64 network with a learning rate of 2.50×10^{-3}. There was mainly 2 variations used in these experiments, one with a ReLU activation function after the first hidden layer and the other with a Tanh activation. ReLU was used in later experiments in order to allow for experimentation with guided backpropagation, since it requires an activation function with non-negative outputs. The network architecture for the Tanh-based model is provided in Figure 3 and the ReLU-based model in Figure 4. In order to both observe and evaluate the results from these experiments, unless otherwise specified, a training set consisting of 500 runs with different seeds (0-499) and a testing set consistent of an independent 500 runs with different seeds (500-999) was used.

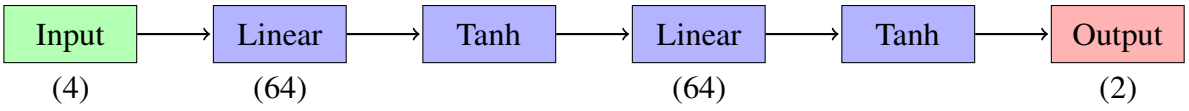

Figure 3: A2C Policy Network Architecture with Tanh Activations

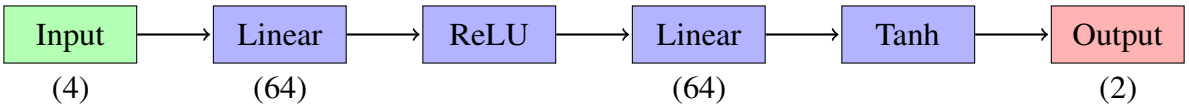

Figure 4: A2C Policy Network Architecture with ReLU & Tanh Activations

From Michael's work on action entropy [12], he found that there was no conclusive correlation between architecture and entropy thresholding, and thus there was architecture invariance. He also found that higher SE boost was correlated with higher learning rates by performing a sweep of 9 learning rates. So, the experiments we performed here were based on the same architecture and hyperparameters used by Michael, in order to be able to compare the results.

3.2 Entropy Calculation

The neuron activation entropy for all experiments (unless otherwise specified), was calculated using the activation values for all 128 neurons in the hidden layers of the network over a 50-step window. First, a histogram of these activation values is computed over 7 bins and then the entropy is calculated based on the frequencies of the histogram bins. The entropy is a measure of the randomness or uncertainty in the probability distribution.

The entropy is then calculated using the following formula, based on Equation 6:

$$H(X) = -\sum_{i=1}^{7} P(x_i) \cdot \log(P(x_i)) \tag{9}$$

where $H(X)$ is the entropy, $P(x_i)$ is the frequency of the i-th bin, and the sum is over all the bins. The algorithm for neuron activation entropy calculation is provided in Algorithm 1.

3.3 Gradient Weighting

Experiments were also performed by weighting the activation values by the gradient of the neuron. So, for a neuron with activation a_i, the gradient-weighed activation value would be $grad(a_i) \times a_i$. Then. the entropy is calculated using these values using the same approach as subsection 3.2. The algorithm for gradient-weighted neuron activation entropy calculation is provided in Algorithm 2.

3.4 Thresholding

In experiments that involved thresholding based on entropy or gradient-weighted entropy values, threshold increments of 0.01 were used starting at a value close to but less than the lowest entropy value for that experiment and stopping at a value close to but greater than the largest entropy value for that experiment. At every threshold value, sample efficiency values were calculated and compared to a baseline sample efficiency where no thresholding was applied. The thresholding process involves running a single 50-step iteration of every model and discarding any models for which the entropy is below the threshold. The rest of the models run until completion.

4 Neuron Activation Entropy Thresholding Results

Experiments that explored the effect on neuron activation entropy thresholding were performed on both the models with Tanh activations and ReLU activations. Experiments on the Tanh model were performed on both the CartPole-v0 and CartPole-v1 environments, while experiments on the ReLU model were done only on the CartPole-v0 environment. Table 1 below summarizes the results of the experiments.

Experiment	Optimal Threshold	Training SE Boost (%)	Testing SE Boost (%)
Tanh, CartPole-v0	1.35	19.07	4.64
ReLU, CartPole-v0	1.05	27.89	-5.01
Tanh, CartPole-v1	1.20	5.25	-1.95

Table 1: Summary of Neuron Activation Entropy Thresholding Results

4.1 Tanh Activation, CartPole-v0

For the model architecture with Tanh activations, as portrayed in Figure 3, a sizeable sample efficiency boost was identify when thresholding was applied, as seen in Table 1. Although smaller on the test set, there was still a boost which indicates that there may be a correlation between thresholding based on entropy and achieving improved sample efficiency.

In Figure 5 and Figure 6, it is evident that there is a spike in sample efficiency at specific threshold values. Interestingly, the threshold values at the peaks for both the training and test sets are quite close, indicating a potential correlation between the threshold and sample efficiency.

In an effort to gain deeper insights into the relationship between entropy and the number of iterations required for a model to reach success, a classifier was developed to predict whether a model would converge in more or fewer steps than the median based on the activation entropy values of its

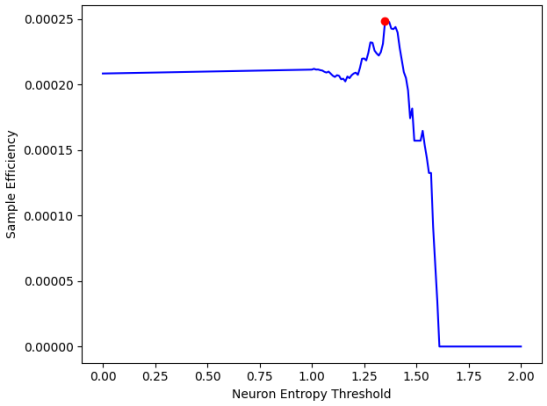

Figure 5: Threshold vs. Sample Efficiency on Training Data (Tanh Activation, CartPole-v0)

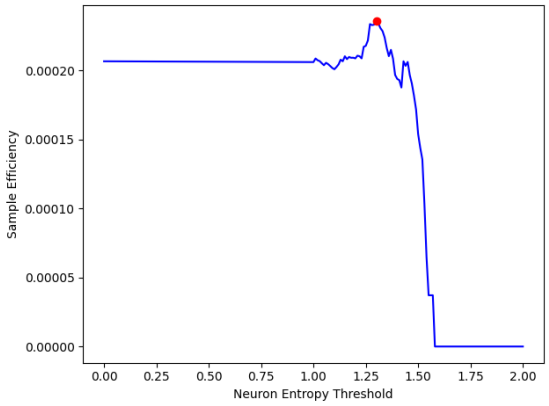

Figure 6: Threshold vs. Sample Efficiency on Test Data (Tanh Activation, CartPole-v0)

neurons. Three different experimental configurations were employed, focusing on the activation entropies of: (1) all 128 neurons, (2) only the 64 neurons in the first hidden layer, and (3) only the 64 neurons in the second hidden layer. In all cases, the classifier consistently predicted the same class for the test data, regardless of the input provided. This rendered the classifier ineffective in generating useful insights into the relationship between entropy and convergence efficiency.

The lack of meaningful output from the classifier could be attributed to the presence of high noise in the data, which might necessitate additional trials to obtain a clearer understanding of the relationship. Alternatively, it is possible that the relationship between entropy and the efficiency of reaching success is non-linear in nature, further complicating the interpretation of the results obtained. Thus, further investigation is required to develop a more comprehensive understanding of the association between neuron activation entropy and convergence efficiency within this context.

Analyzing the entropy values versus the number of iterations scatter plot on the training data in Figure 7 does not reveal a clear relationship. Most entropy values are situated between 0.8 and 1.5; however, no apparent relationship between entropy and the steps until reward (number of iterations) is discernible.

By considering a successful experiment as one that converges in no more than 3000 iterations, we observe that the overall success rate on the training data is 50.4%. However, when focusing solely on models with an entropy value of at least 1.3 (the threshold identified earlier), the success rate increases to 58.8%. This observation supports the notion that applying a threshold based on entropy can enhance efficiency. This is visualized in Figure 8, where the successful models are in red and the vertical line denotes the entropy threshold.

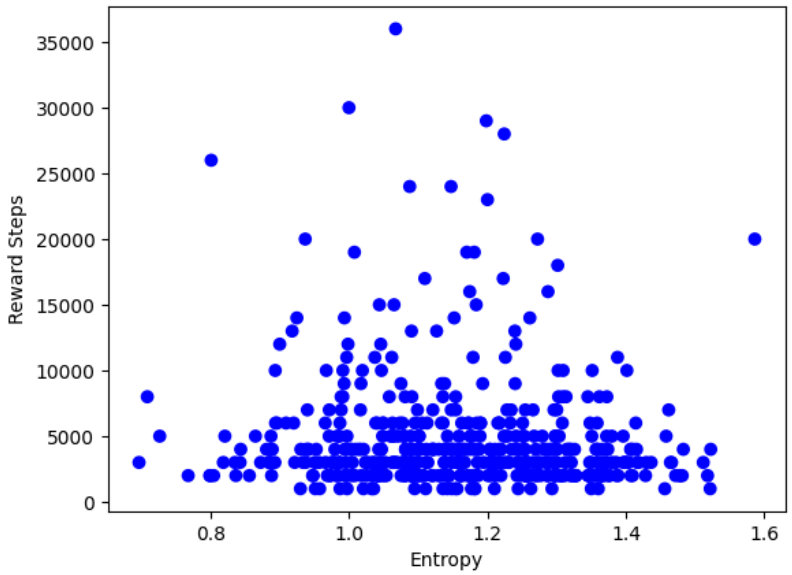

Figure 7: Entropy vs. Number of Iterations on Training Data (Tanh Activation, CartPole-v0)

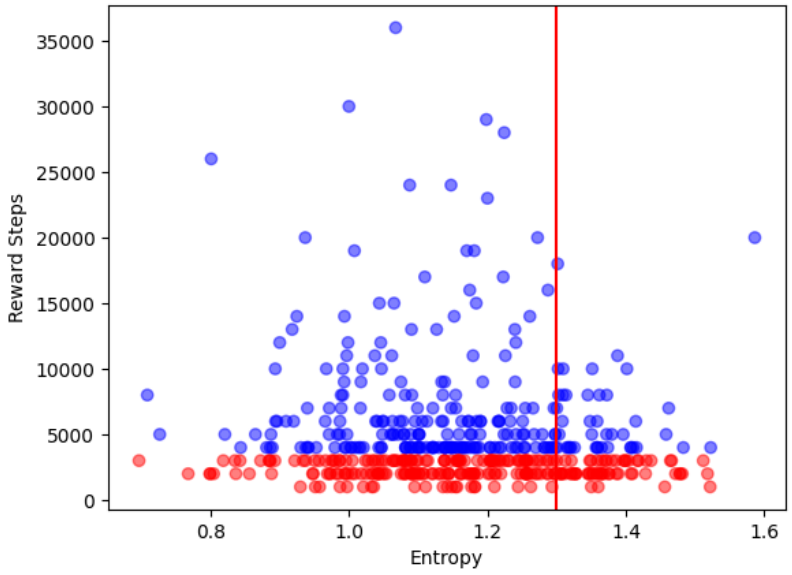

Figure 8: Scatter Plot with Success and Entropy Threshold (Tanh Activation, CartPole-v0)

In order to evaluate the potential for achieving more consistent results through the implementation of additional experiments, a comprehensive set of 10,000 models was executed, with each model employing a unique seed value ranging from 0 to 9,999. The primary objective of this extensive experimentation was to assess the presence of a significant correlation between entropy and the

efficiency of reinforcement learning training, specifically within the context of using Tanh activations in the CartPole-v0 environment.

Following the completion of the 10,000 models, the results indicated that the optimal threshold value stood at 1.1, accompanied by an SE boost of a mere 0.81%. Consequently, these findings suggested a lack of substantial correlation between entropy and reinforcement learning training efficiency in the aforementioned environment. A visual representation of these results can be found in Figure 9, which effectively captures the outcomes of this extensive research exercise.

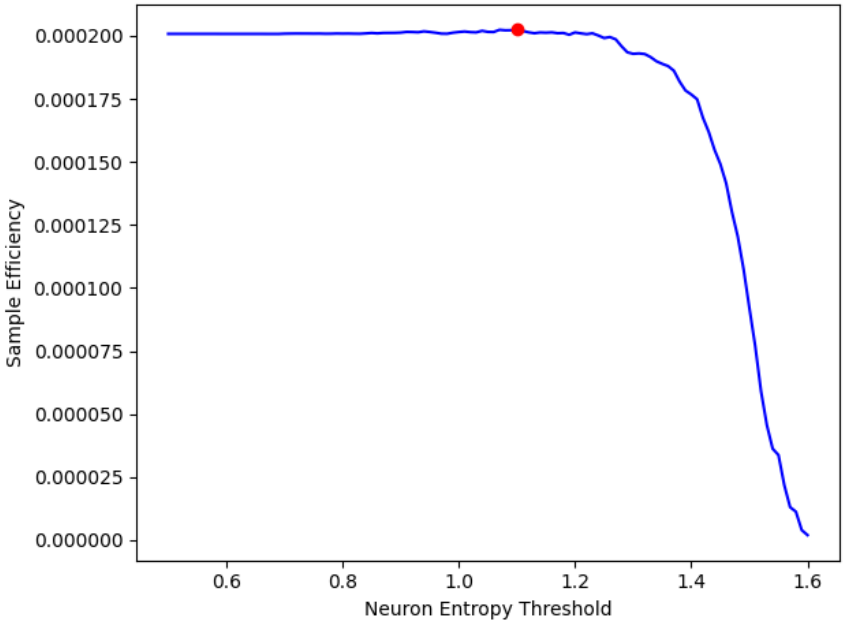

Figure 9: Entropy vs. Number of Iterations on 10,000 Models (Tanh Activation, CartPole-v0)

4.2 ReLU Activation, CartPole-v0

In the experiments involving the model architecture with ReLU activations, as illustrated in Figure 4, the focus was exclusively on the CartPole-v0 environment. The outcomes of these trials can be found in Table 1. Upon examining the table, a notable sample efficiency boost in the training set emerges when thresholding is implemented. However, a negative boost is observed in the test set, which suggests that improved performance in the training set may not directly correspond to enhanced performance in the test set.

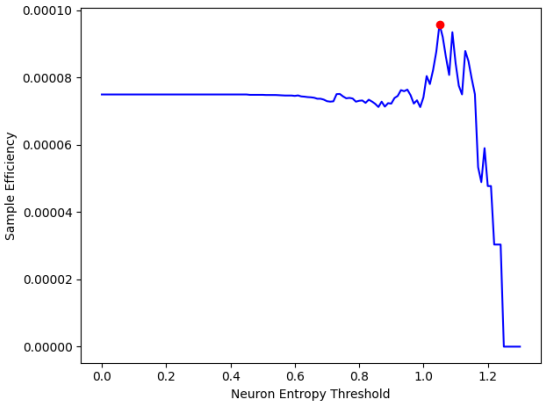

Figure 10: Threshold vs. Sample Efficiency on Training Data (ReLU Activation, CartPole-v0)

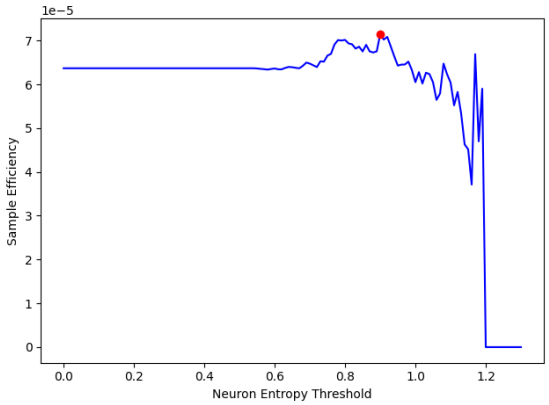

Figure 11: Threshold vs. Sample Efficiency on Test Data (ReLU Activation, CartPole-v0)

In Figure 10 and Figure 11, a notable difference can be observed between the performance on the training set and the performance on the test set. Both the training and test sets display a spike in sample efficiency at specific threshold values, although the peak values differ. This observation may imply that the relationship between thresholding based on entropy and achieving improved sample efficiency is more intricate than initially anticipated.

Like the Tanh model, the classifier was unable to provide any useful predictions when using the ReLU model's entropy values. This could be attributed to high noise in the data or the non-linear nature of the relationship between entropy and the efficiency of reaching success.

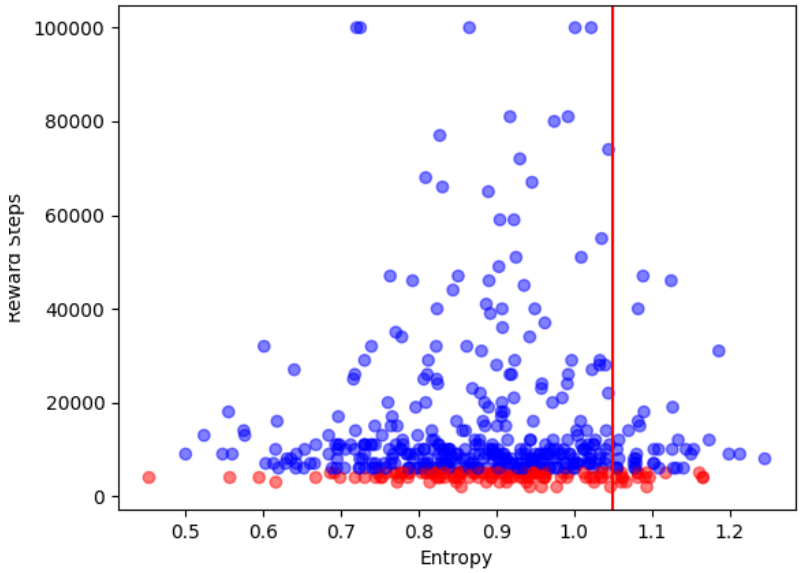

Figure 12: Entropy vs. Number of Iterations on Training Data (ReLU Activation, CartPole-v0)

Figure 12 presents the scatter plot of entropy values versus the number of iterations on the training data for the ReLU model. In this case, the relationship between entropy and steps until reward (number of iterations) is still ambiguous. The majority of entropy values lie between 0.6 and 1.2, but no evident correlation between the two variables can be discerned. By considering a successful experiment as one that converges in no more than 5000 iterations, we observe that the overall success rate on the training data is 23.8%. However, when focusing solely on models with an entropy value of at least 1.05 (the threshold identified earlier), the success rate increases to 30.4%. Successful models are visualized in red. This observation supports the notion that applying a threshold based on entropy can enhance efficiency.

4.3 Tanh Activation, CartPole-v1

For the model architecture with Tanh activations, the experiments were also conducted in the CartPole-v1 environment. The results are displayed in Table 1. From the table, it can be observed that there is a slight sample efficiency boost on the training set when thresholding is applied. However, the test set displayed a negative boost, which may indicate that the performance on the training set does not necessarily translate to a better performance on the test set.

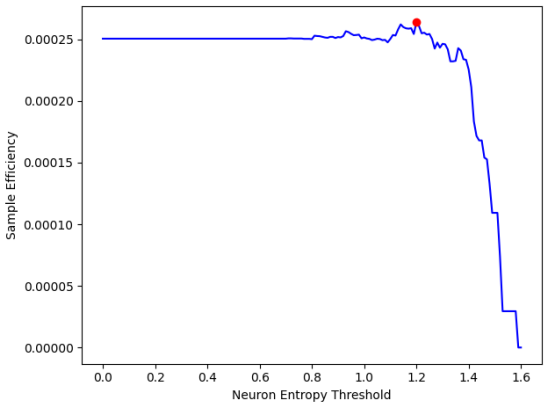

Figure 13: Threshold vs. Sample Efficiency on Training Data (Tanh Activation, CartPole-v1)

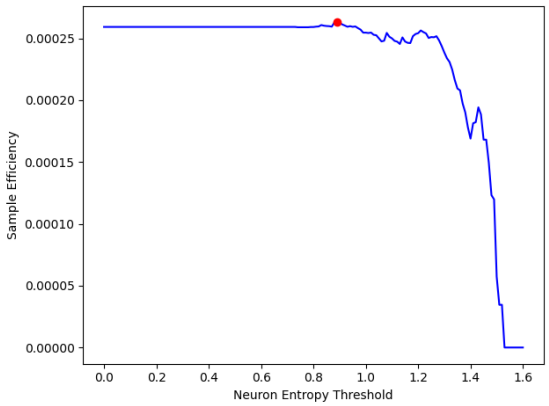

Figure 14: Threshold vs. Sample Efficiency on Test Data (Tanh Activation, CartPole-v1)

From Figure 13 and Figure 14, it can be observed that there is a difference between the performance on the training set and the performance on the test set. A spike in the sample efficiency can be seen at specific threshold values in both the training and test sets, although, similar to the ReLU model, the peak values differ. This may suggest that the relationship between thresholding based on entropy and achieving improved sample efficiency is more complex in the CartPole-v1 environment as well.

In the case of the Tanh model trained on the CartPole-v1 environment, the classifier was also unable to provide any useful predictions. Similar to the previous cases, this could be due to high noise in

the data or the non-linear nature of the relationship between entropy and the efficiency of reaching success.

The scatter plot for this analysis can be found in Figure 15, where the entropy values are plotted against the number of iterations on the training data. In this figure, the correlation between entropy and the number of steps until reward (iterations) remains unclear. The majority of entropy values are observed to lie within the range of 0.8 and 1.5, but there is no apparent correlation with the number of iterations.

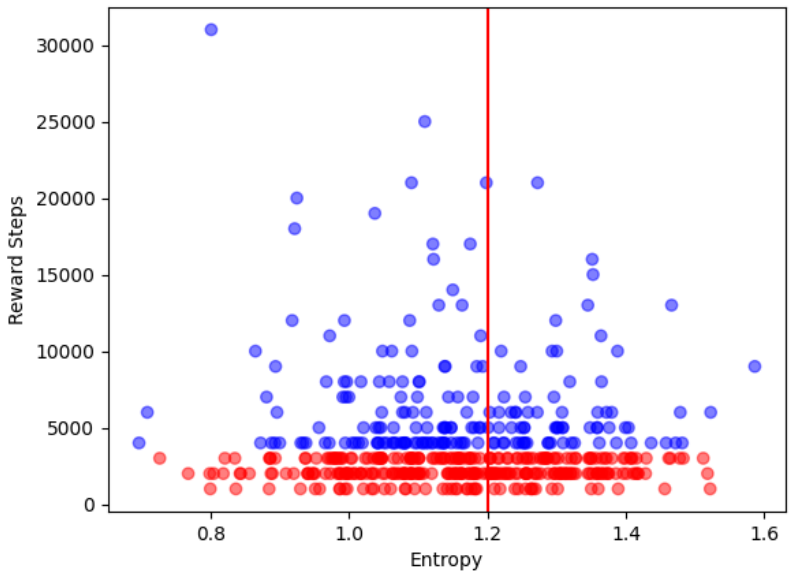

Figure 15: Entropy vs. Number of Iterations on Training Data (Tanh Activation, CartPole-v1)

To evaluate the performance of the model, a success criterion is established, defining a successful model (marked in red) as one that converges within a maximum of 3000 steps. Using this criterion, the overall success rate is calculated to be 62.0%. By applying an optimal threshold of 1.2 for entropy values, the success rate is found to be 64.2%, which is only slightly higher than the overall rate. This finding indicates the need for further research and experimentation to better understand the relationship between neuron activation entropy and sample efficiency in models utilizing Tanh activations, particularly in the context of the CartPole-v1 environment.

5 Gradient-Weighted Entropy Thresholding Results

Experiments exploring the effect of gradient-weighted entropy thresholding were performed on both the models with Tanh activations and ReLU activations, using the CartPole-v0 environment. In

this approach, the neuron activation entropies were multiplied by the gradient magnitudes to create a gradient-weighted entropy, which was then thresholded. Table 2 below summarizes the results of the experiments.

Experiment	**Optimal Threshold**	**Training SE Boost (%)**	**Testing SE Boost (%)**
Tanh, CartPole-v0	1.09	19.53	-7.24
ReLU, CartPole-v0	0.88	19.32	77.48

Table 2: Summary of Gradient-Weighted Entropy Thresholding Results

5.1 Tanh Activation, CartPole-v0

The influence of applying gradient-weighted entropy thresholding on the model architecture with Tanh activations is illustrated in Table 2. The results demonstrate an enhancement in sample efficiency on the training set. However, at the same threshold value, the sample efficiency boost is negative on the test set. Consequently, the findings are inconclusive regarding a potential correlation between gradient-weighted entropy thresholding and improved sample efficiency. This observation is depicted in Figure 16 and Figure 17.

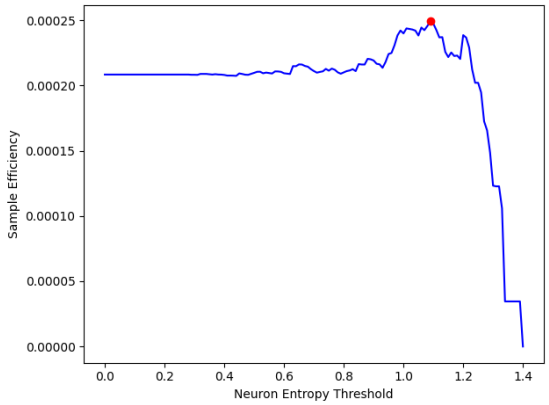

Figure 16: Gradient-Weighted Threshold vs. Sample Efficiency on Training Data (Tanh Activation, CartPole-v0)

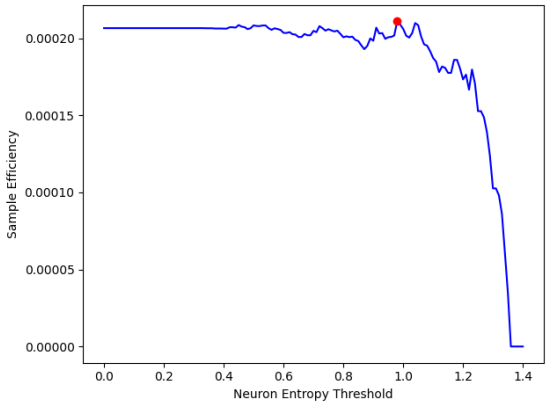

Figure 17: Gradient-Weighted Threshold vs. Sample Efficiency on Test Data (Tanh Activation, CartPole-v0)

In the 10,000 model experiment, the optimal threshold value was 0.84 with an SE boost of only 2.19%. So, experimenting over more models did not yield a clear, significant boost in SE. These results are shown in Figure 18.

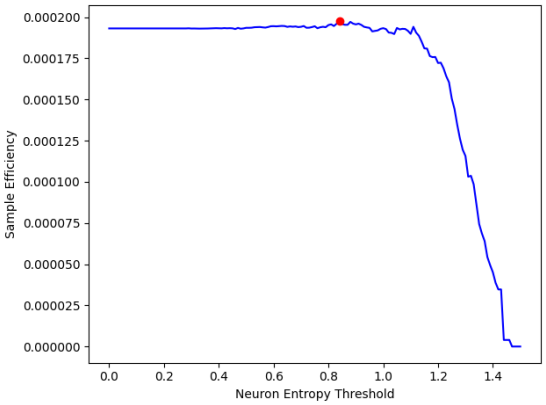

Figure 18: Gradient-Weighted Entropy vs. Number of Iterations on 10,000 Models (Tanh Activation, CartPole-v0)

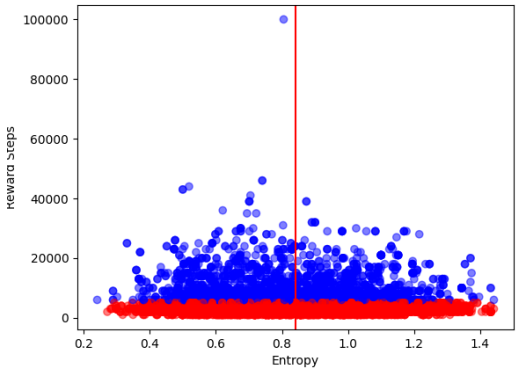

Figure 19: Scatter Plot with Success and Grad Entropy Threshold (Tanh Activation, CartPole-v0)

Considering a model successful if it converges in 5000 iterations or fewer and using an entropy threshold of 0.84, we obtain an overall success rate of 72.9% and a success rate of 73.1% for models with an entropy greater than 0.84. This is illustrated in Figure 19, where the red points correspond to successful models. Consequently, the difference in success rates is not significant, indicating no apparent relationship between the gradient-weighted entropy threshold and faster model convergence for Tanh models in the CartPole-v0 environment.

5.2 ReLU Activation, CartPole-v0

The impact of applying gradient-weighted entropy thresholding on the model architecture with ReLU activations can be observed in Table 2. The results demonstrate a significant increase in sample efficiency for the training set and an even larger boost for the test set, with both improvements occurring at approximately the same threshold values. This can be visualized in Figure 20 and Figure 21.

We define a model as successful if it converges within 5000 iterations or fewer. By utilizing the optimal entropy threshold of 0.88, which was identified earlier, we achieve an overall success rate of 20.8%. Moreover, for models with an entropy greater than 0.88, we observe a success rate of 23.8%. This is depicted in Figure 22, where the red points represent successful models. Despite these findings, the difference in success rates is not significant enough to draw a definitive conclusion regarding the relationship between the gradient-weighted entropy threshold and faster model convergence for ReLU models in the CartPole-v0 environment. Further investigation with additional data might help in clarifying this relationship.

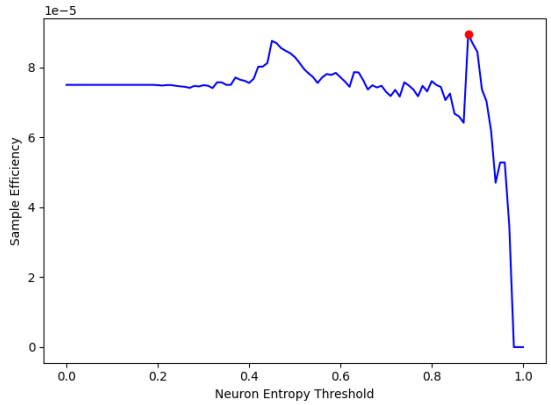

Figure 20: Gradient-Weighted Threshold vs. Sample Efficiency on Training Data (ReLU Activation, CartPole-v0)

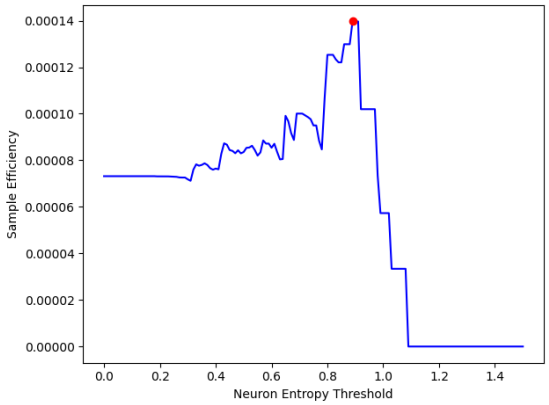

Figure 21: Gradient-Weighted Threshold vs. Sample Efficiency on Test Data (ReLU Activation, CartPole-v0)

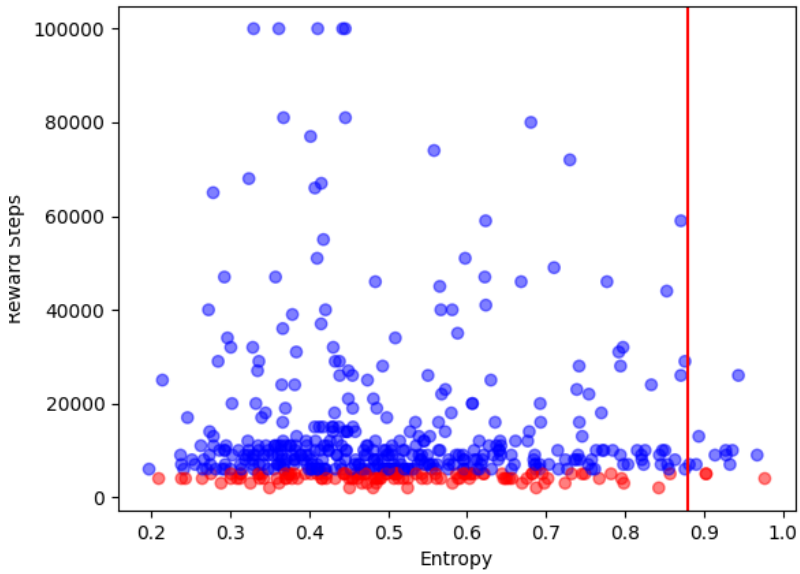

Figure 22: Scatter Plot with Success and Gradient-Weighted Entropy Threshold (ReLU Activation, CartPole-v0)

6 Guided-Backpropagation Results

Experiments exploring the effect of guided-backpropagation were performed on the model with ReLU activations in the CartPole-v0 environment. Guided-backpropagation is a technique used to visualize the importance of each neuron in the model by selectively backpropagating gradients through the network. In this context, the goal is to investigate if guided-backpropagation can provide useful insights into the relationship between neuron activation entropy and sample efficiency.

6.1 ReLU Activation, CartPole-v0

For the model architecture with ReLU activations, the experiments were conducted using guided-backpropagation in the CartPole-v0 environment. On the training set, the optimal threshold was found to be 0.58 with an SE boost of 71.38%. On the test set, the same threshold showed a 16.53% SE boost.

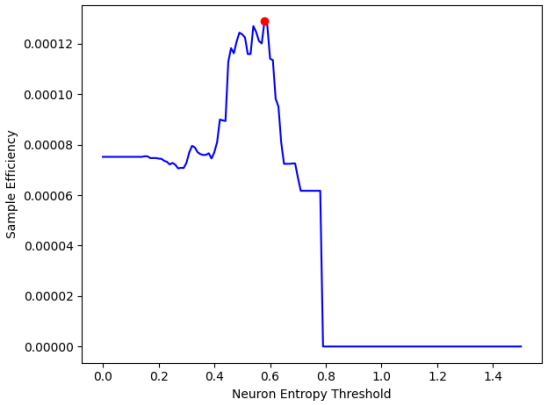

Figure 23: Guided-Backpropagation Threshold vs. Sample Efficiency on Training Data (ReLU Activation, CartPole-v0)

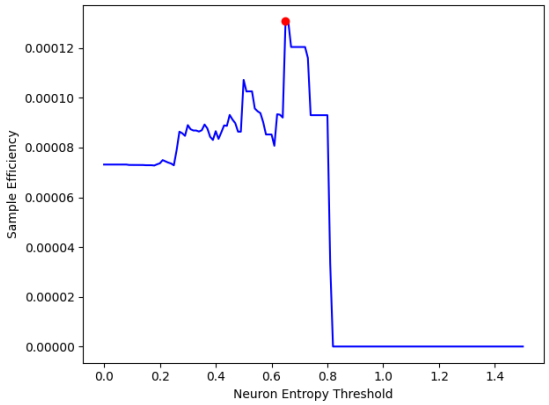

Figure 24: Guided-Backpropagation Threshold vs. Sample Efficiency on Test Data (ReLU Activation, CartPole-v0)

In Figure 23 and Figure 24, the relationship between the guided-backpropagation threshold and sample efficiency is displayed for both the training and test sets. The results are promising since they show significant boosts when thresholding is applied.

The classifier was also unable to provide any useful predictions when using the guided-backpropagation values. This could be attributed to high noise in the data or the non-linear nature of the relationship between guided-backpropagation gradient-weighted entropy values and the efficiency of reaching success.

By considering a successful experiment as one that converges in no more than 5000 iterations, we observe that the overall success rate on the training data is 20.8%. However, when focusing solely on models with an entropy value of at least 0.58 (the threshold identified earlier), the success rate increases to 33.3%. Figure 25 shows the scatter plot of guided-backpropagation values versus the number of iterations with this threshold indicated and the successful models colored in red. This observation supports the notion that applying a threshold based on entropy can enhance efficiency.

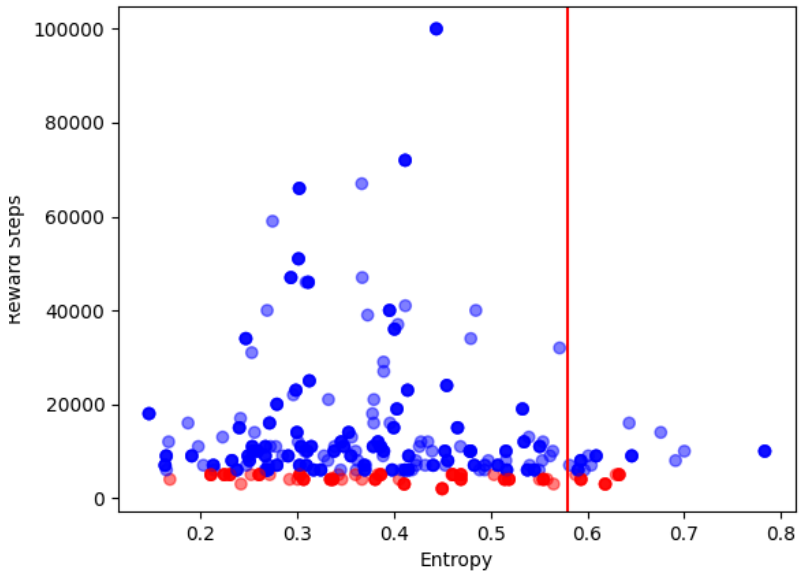

Figure 25: Scatter Plot with Success and Guided-Backpropagation Gradient-Weighted Entropy Threshold (ReLU Activation, CartPole-v0)

7 Discussion

The experiments conducted in this study aimed to investigate the relationship between neuron activation entropy and sample efficiency in A2C networks, with a focus on architectures employing Tanh and ReLU activation functions in the CartPole-v0 and CartPole-v1 environments. In addition to exploring the direct impact of neuron activation entropy, the experiments also sought to analyze the effect of gradient-weighted entropy thresholding and the implementation of guided-backpropagation on model performance.

The results from the experiments involving Tanh activation functions in the CartPole-v0 environment indicated a potential correlation between thresholding based on neuron activation entropy and improved sample efficiency, although further investigation is required to confirm this relationship, since even running an experiment on more data (10,000 models) did not show a strong relationship. Experiments conducted on ReLU activation functions in the same environment yielded slightly more promising results, with significant boosts in sample efficiency when thresholding was applied. Moreover, the implementation of guided-backpropagation on ReLU models displayed even more promising results and points to the potential utility of guided-backpropagation in improving the efficiency of reinforcement learning models. However, since the experiments on ReLU activations only used 1,000 models (500 training and 500 test), additional, larger experiments should be used to confirm this.

When considering the experiments employing gradient-weighted entropy thresholding, the findings for Tanh activations in the CartPole-v0 environment were inconclusive, with an increase in sample efficiency on the training set on par with the non-gradient-weighted entropy results and a negative boost on the test set. However, the results for ReLU activations in the same environment demonstrated a significant improvement in sample efficiency for both the training and test sets when applying gradient-weighted entropy thresholding. This improvement suggests that gradient-weighted entropy thresholding could be a useful technique for enhancing the performance of a model while reducing the number of iterations required for training. The results on the ReLU experiments with guided-backpropagation applied also showed significant SE boosts on both the training and test sets.

8 Conclusion

The results obtained from this study indicate that neuron activation entropy thresholding and gradient-weighted entropy thresholding can potentially improve sample efficiency in A2C networks with ReLU activations, particularly in the CartPole-v0 environment. Although the findings for Tanh activations were inconclusive, the experiments involving ReLU activation functions demonstrated promising results, with significant boosts in sample efficiency on both the training and test sets. This observation was further reinforced by the implementation of guided-backpropagation on ReLU models, which showed even more significant improvements in sample efficiency.

Due to the inconclusive findings for Tanh activations, further research with a larger dataset and a more comprehensive range of environments would be necessary to better understand the relationship between neuron activation entropy and sample efficiency in these cases. Additionally, future studies could explore the impact of other thresholding techniques and visualization methods, such as integrated gradients and saliency maps, in order to gain a deeper understanding of the factors influencing the performance of reinforcement learning models.

In conclusion, this study provides valuable insights into the potential benefits of neuron activation entropy thresholding and gradient-weighted entropy thresholding for improving sample efficiency in A2C networks with ReLU activations. By employing these thresholding techniques in conjunction with guided-backpropagation, it may be possible to enhance the performance and efficiency of reinforcement learning models, leading to faster convergence in real-world applications.

References

[1] DLR-RM. *Stable-Baselines3 Docs - Reliable Reinforcement Learning Implementations*. URL: https://stable-baselines3.readthedocs.io/en/master/.

[2] Farama Foundation. *Cart Pole*. URL: https://www.gymlibrary.dev/environments/classic_control/cart_pole/.

[3] Sooyoung Jang and Hyung-Il Kim. "Entropy-Aware Model Initialization for Effective Exploration in Deep Reinforcement Learning". In: *CoRR* abs/2108.10533 (2021). arXiv: 2108.10533. URL: https://arxiv.org/abs/2108.10533.

[4] Vijay Konda and Vijaymohan Gao. "Actor-critic algorithms". In: (Jan. 2000).

[5] Tom M. Mitchell. *Machine learning, International Edition*. McGraw-Hill Series in Computer Science. McGraw-Hill, 1997. ISBN: 978-0-07-042807-2. URL: https://www.worldcat.org/oclc/61321007.

[6] Vikramjit Mitra and Horacio Franco. "Leveraging Deep Neural Network Activation Entropy to cope with Unseen Data in Speech Recognition". In: *CoRR* abs/1708.09516 (2017). arXiv: 1708.09516. URL: http://arxiv.org/abs/1708.09516.

[7] Volodymyr Mnih et al. "Asynchronous Methods for Deep Reinforcement Learning". In: *CoRR* abs/1602.01783 (2016). arXiv: 1602.01783. URL: http://arxiv.org/abs/1602.01783.

[8] Volodymyr Mnih et al. "Playing Atari with Deep Reinforcement Learning". In: *CoRR* abs/1312.5602 (2013). arXiv: 1312.5602. URL: http://arxiv.org/abs/1312.5602.

[9] Savinay Nagendra et al. "Comparison of reinforcement learning algorithms applied to the cart-pole problem". In: *2017 International Conference on Advances in Computing, Communications and Informatics (ICACCI)*. 2017, pp. 26–32. DOI: 10.1109/ICACCI.2017.8125811.

[10] Max Pumperla and Kevin Ferguson. "Chapter 12. Reinforcement learning with actor-critic methods". In: *Deep Learning and the Game of Go*. Manning Publications Co., 2019, 247–261.

[11] Matthias Rosynski, Frank Kirchner, and Matias Valdenegro-Toro. "Are Gradient-based Saliency Maps Useful in Deep Reinforcement Learning?" In: *CoRR* abs/2012.01281 (2020). arXiv: 2012.01281. URL: https://arxiv.org/abs/2012.01281.

[12] Michael Ruan. "Improving Sample Efficiency of Deep Reinforcement Learning using Action and Neuron Activation Entropy Thresholding". Bachelor's Thesis. 2020.

[13] Richard S. Sutton and Andrew G. Barto. "Reinforcement Learning: An Introduction". In: *IEEE Transactions on Neural Networks* 16 (2005), pp. 285–286.

[14] Kenneth F Wallis. *A note on the calculation of entropy from histograms*. 2006.

A Algorithms

A.1 Neuron Entropy Calculation Algorithm

Algorithm 1 Neuron Activation Entropy - calculation of entropy for a single model

function NEURONENTROPY($activation$)

$activations\_histogram \leftarrow histogram(activation, bins = 7)$

$activation\_freqs \leftarrow get\_frequencies(activations\_histogram)$

$H \leftarrow -\sum_{i=1}^{7} activation\_freqs[i] * log(activation\_freqs[i])$

return H

end function

A.2 Gradient-Weighted Neuron Entropy Calculation Algorithm

Algorithm 2 Gradient-Weighted Neuron Activation Entropy - calculation of gradient-weighted entropy for a single model

function NEURONGRADENTROPY($activation, gradient$)

$activation \leftarrow activation * gradient$

$activations\_histogram \leftarrow histogram(activation, bins = 7)$

$activation\_freqs \leftarrow get\_frequencies(activations\_histogram)$

$H \leftarrow -\sum_{i=1}^{7} activation\_freqs[i] * log(activation\_freqs[i])$

return H

end function

B Code

B.1 GitHub

All code used in this work is included at the GitHub repository here: gkysaad/reinforcement-learning-thesis