Investigating the Impact of Entropy Regularisation on an Attention Augmented Agent

Daniel Maguire
Student Number: 23222425
Supervisor: Prof. Conor Ryan
CSIS
University of Limerick

Submitted to the University of Limerick for the degree of
MSc. of Artificial Intelligence and Machine Learning 2024

Abstract

In order to effectively train Reinforcement Learning (RL) agents, it is imperative to find a balance between exploration and exploitation. Entropy regularisation can be used as a tool to help find that balance, as it encourages exploration and reduces the chance of settling on sub-optimal policies by increasing stochasticity in the policy distributions. The challenge with finding this balance using entropy regularisation is in identifying the optimal entropy configuration which maximises learning in RL agents. This research aims investigate how varying entropy regularisation coefficients can impact learning and stability across three Atari environments of varying complexities. The goal of this research is to identify optimal entropy coefficients across the three learning environments. The RL agent will be trained on SpaceInvaders, Seaquest and Breakout. The agent has been augmented to include an attention module, which can be used to visualise the agent's decisions using attention maps. The experiments in this research include adjusting and observing various entropy coefficients and decay strategies to assess their effects on learning outcomes. Decaying entropy is observed to be a more optimal strategy compared to static entropy across all environments. An assessment of the impact of the added complexity of the attention network on the agent's ability to learn is also conducted. The findings reveal that there is a correlation between the degradation of the agent's ability to learn and the complexity of the network.

Keywords: Reinforcement Learning, Entropy Regularisation, Exploration-Exploitation Balance, Attention, Attention Maps, Hyperparameter Tuning

Table of Contents

Table of Figures

Data Source and Generative AI

The data used in this research is a combination of three sources.

Data source 1: https://github.com/cjlovering/Towards-Interpretable-Reinforcement-Learning-Using-Attention-Augmented-Agents-Replication
This was used as a foundation for the attention augmented network.

Data source 2: https://github.com/rgilman33/simple-A2C-PPO
This was used as a foundation for the PPO implementation.

Data source 3: https://github.com/greydanus/visualize_atari
This source was used as a foundation for the attention map generation logic.

Generative AI was used as a dictation and note organising tool during this research. It was useful in taking my stream of consciousness on individual topics, and creating quick structured notes as I was working. I also used it as a way to discuss and break down complex topics encountered during the initial stages of research to help solidify my foundational understanding of these topics. This was helpful as these topics often required me to approach them from various perspectives, and having a conversational high-level discussion helped with getting new ideas and insights. The understanding I gained was then cross checked against the literature to confirm that the insights were accurate before diving into my own implementation and building upon this foundational knowledge.

1

Introduction

1.1 Motivation

The last ten years has brought forward some considerable advancements for the Reinforcement Learning (RL) field and has gained widespread attention especially after the introduction of AlphaGo (Silver *et al.*, 2017). There seems to be a lot of potential in RLs ability to handle complicated tasks like robotics and market prediction in the financial industry. One of the key challenges being tackled by RL researchers across the industry is finding the right balance between exploration and exploitation to optimise agent learning efficiency. While exploration is needed for an RL agent to learn new strategies for navigating its environment, exploitation on the other hand involves taking the strategies previously learned by the agent and using them to maximise the reward. RL researchers have highlighted the importance of balancing these two approaches, which has naturally led to various new strategies being developed, like the epsilon-greedy strategy that (Mnih *et al.*, 2013) popularised.

Entropy regularisation as a method has displayed potential in tackling the exploration exploitation dilemma. It is commonly employed in various state-of-the-art reinforcement learning practices today (Mnih *et al.*, 2016; Haarnoja *et al.*, 2018). The concept of entropy regularisation urges agents learning an environment to introduce a certain level of unpredictability in their decisions. This can be achieved by introducing randomness through the calculation of an entropy term in the loss function which is scaled based on a given entropy regularisation coefficient and may or may not decay over time. The idea behind this is

to help reduce premature convergence on suboptimal policies, allowing for continued exploration. There is a challenge in identifying the optimal entropy term that maximises learning efficiency i.e. how well the agent learns from its experiences, and also maintains stable learning. The entropy regularisation coefficient and its rate of decay in non-static scenarios can significantly impact the learning stability and performance of RL agents. Understanding how to fine-tune these parameters is necessary for taking advantage of entropy regularisation effectively.

Another significant advancement in field of RL is the integration with attention mechanisms by (Mott *et al.*, 2019). The integration of attention mechanisms is a reimplementation of attention from the Transformer architecture (Vaswani *et al.*, 2023). Attention gives RL agents the ability to focus on relevant parts of the input, with the aim of enhancing their decision making abilities. The literature indicates the use of attention in RL has the potential to improve both the interpretability and performance of RL models by focusing on the most important information in a given environment (Mott *et al.*, 2019). According to (Niv *et al.*, 2015), attention also enables an agent to handle high-dimensional inputs more efficiently. In general, attention implementations provide qualitative insights into the agents decision making process through the use of attention maps (Greydanus *et al.*, 2018), allowing us to see exactly what each attention head is focusing on.

The goal of this dissertation is to explore the impact of entropy regularisation on an attention augmented agent and observe how it learns when subjected to various entropy regularisation coefficients and rates of decay. It will aim to quantify and visually represent how varying entropy coefficients and decay strategies affect the performance of the attention augmented RL agents with the aid of attention maps. This research focuses on applying these techniques to agents attempting to learn three Atari environments of varying complexities: Space Invaders, Seaquest, and Breakout. The motivation behind this research is to attempt to demystify and aid in developing a systematic approach to hyperparameter tuning of entropy regularisation in RL agents, with a particular emphasis on attention-augmented models due to their increased interpretability. This research seeks to contribute to the development of more robust, interpretable, and efficient RL models. The findings can be used to guide future work in efficient hyperparameter tuning and the design of RL agents.

1.2 Research Questions (RQs)

How does entropy regularisation influence the performance of attention-augmented RL agents?

Is there a generalisable entropy term that will result in increased performance across all environments?

Does the implementation of an attention mechanism for enhanced interpretability impact the performance of RL models?

1.3 Overview of the Dissertation

This dissertation focuses on the use of entropy regularisation within attention augmented RL agents. The research aims to improve the exploration capabilities and overall performance of RL agents while also providing increased interpretability through qualitative measures. Below is an overview which provides a roadmap for the dissertation, summarising each chapter.

Chapter 2: Literature Review

In this chapter a comprehensive review of the existing literature in reinforcement learning, attention mechanisms, and entropy regularisation is conducted. It starts with an overview of foundational RL algorithms; Q-learning, Policy Gradient, and Actor-Critic methods, as well as their applications. It then explores the role of attention mechanisms, specifically when applied in RL. Here the focus is on the benefits of interpretability and the potential of improved performance. This chapter then investigates the current research on entropy regularisation, discussing its impact on exploration and policy optimisation.

Chapter 3: Specification, Design and Implementation

This chapter discusses the initial focus of the dissertation, and provides details on the design choices of the attention augmented RL agent. It discusses the challenges that were faced during the implementation, including issues with the Convolutional Neural Network (CNN) within the model, and the transition from the Advantage Actor Critic (A2C) algorithm to the

Proximal Policy Optimisation (PPO) implementation. It also covers the integration with the IMPALA (Importance Weighted Actor-Learner Architecture) CNN model which was subsequently used as the vision core of the agent, while also explaining the pivot in research focus towards examining the impact of entropy regularisation coefficients. The practical implementation of the designed RL agent is also discussed. It provides details into the architecture of the model, algorithm and hyperparameters, and details the development of the Attention Augmented IMPALA agent. The chapter also describes in further detail the process of transitioning from A2C to PPO to address the limitations of the former and enhance learning outcomes.

Chapter 4: Results

In this chapter the results are presented. It first looks at a comparison of RL models of various complexities to demonstrate an observed degradation in model performance which correlated with the increased complexity of the model and aligned with the literature. It then looks at the performance of the attention augmented agent under various entropy regularisation and environment configurations. It will also demonstrate the benefits of including visualised attention, as well as discuss what complexities it introduces.

Chapter 5: Conclusion

The concluding chapter summarises how the objectives of the dissertation were met, and reflects on the key findings which have been observed from the research. Recommendations are made for future work, and potential for further exploration and improvements to the methodologies used in the research are suggested.

2

Chapter 2: Literature Review

2.1 Reinforcement Learning (RL)

Reinforcement learning (RL) is a branch of the machine learning family where an agent learns to make decisions in an environment by performing actions to maximise the total reward through the process of trial and error. The agent interacts with the environment and receives feedback in the form of a reward signal. This information is then used to adjust its policy for selecting actions. There are several key algorithms which form the foundation of modern RL, each with unique methods for learning and optimisation. Many of them are adaptations and improvements on previous algorithms, designed to resolve performance or stability concerns with their predecessors. Q-learning is a model-free RL algorithm that seeks to learn the value of state-action pairs (C. J. C. H. Watkins, 1989). These state-action pairs, or Q-values, represent the iterative expected cumulative reward of taking a specific action in a given state while continuing to follow the current optimal policy. The Q-learning algorithm updates these values iteratively using the Bellman equation (Bellman, 1984). The introduction of deep Q-learning (DQN) improved on traditional Q-learning by enhancing it with a deep neural network to approximate Q-values, enabling the algorithm to handle high-dimensional sensory input data (Mnih *et al.*, 2013). Both Q-Learning and Deep Q-Learning are considered off-policy due to the fact that they directly look for an optimal action-value function completely independent of the policy being followed. Alternatively, policy gradient methods directly optimise the policy by computing gradients of the expected reward with respect to the policy parameters. These methods are considered on-policy because they learn from the actions that they are currently taking. The REINFORCE algorithm is an example of this and is an early

policy gradient method that uses Monte Carlo methods to estimate the gradient (Williams, 1992).

Actor-critic methods are a further innovation that provide a combination of value-based and policy gradient approaches by maintaining two separate networks: the actor, which drives the policy through decision making and the critic which evaluates the policy's potential actions by estimating a value function. These actor-critic methods harness the benefits of both value-based and policy gradient methods to provide continual feedback and decision making. This family of algorithms are considered on-policy, particularly due to the critic needing to learn and evaluate the policy currently being followed. An example using this actor-critic method is the Asynchronous Advantage Actor-Critic (A3C) algorithm. A3C uses multiple agents running in parallel to explore different parts of the state space, leading to more diverse experiences and improved learning (Mnih *et al.*, 2016). OpenAI's adapted version of the A3C algorithm, called A2C, differs primarily in that it updates synchronously, as opposed to the asynchronous nature of the A3C algorithm ('OpenAI Baselines: ACKTR & A2C', 2017). This resolved performance issues caused by workers operating on outdated policies, as all workers must now complete their trajectory before the policy is updated. Proximal Policy Optimization (PPO) further improves upon the A2C algorithm by employing a variety of optimisation techniques i.e. using a clipped surrogate objective function that limits the magnitude of policy updates and improving sample efficiency by training on the same sample set for multiple epochs (Schulman *et al.*, 2017).

2.2 Attention Mechanisms in RL

Attention mechanisms were first introduced as the core part of the transformer architecture by (Vaswani *et al.*, 2023) and are considered a revolutionary method of performing sequence modelling. The transformer improved on the previous state-of-the-art, LSTMs (Long Short Term Memory) (Hochreiter and Schmidhuber, 1997) by allowing parallel processing of input data as well as improved handling of long range dependencies (Vaswani *et al.*, 2023). Incorporating attention mechanisms into a RL neural network architecture allows for improved interpretability and the literature also suggests it can result in performance comparable to state-of-the-art models (Mott *et al.*, 2019). Attention mechanisms allow models to focus on relevant parts of the input when making decisions. When dealing with visual data

as input we can visualise what the agent is paying attention to through the use of saliency, or attention maps (Greydanus *et al.*, 2018). These maps provide insights into the decision making process of the model by highlighting the regions of the input that most influence the output.

An application of attention in RL is described by Mott et al in the DeepMind implementation. This research introduces a model that uses a soft top-down spatial attention mechanism applied to the visual output of a vision core. In this model, a recurrent vision core network processes the visual input. This produces the key and value tensors necessary for attention. The policy LSTM produces the query vector. Then by taking the inner product of the query vector and the keys tensor followed by a spatial softmax an attention map can be produced that focuses on task-relevant information (Mott *et al.*, 2019).

Another approach to using attention mechanisms in RL, which also builds upon Mott et al's work, focuses on dynamically pruning the number of attention heads used in multi head attention mechanisms. In normal circumstances, the number of attention heads is a hyperparameter which is set manually and remains statically configured throughout the training process. Duarte et al hypothesise that static parameterisation might not be optimal in certain cases due to the fact that the complexity of the task can potentially vary during the learning process. They propose a dynamic approach where the agent is configured to have a second policy which determines the appropriate number of attention heads at each timestep based on the contextual memory of the agent.

2.3 Entropy Regularisation in RL

Entropy regularisation within RL encourages exploration by preventing agents from prematurely converging on suboptimal policies. This is useful in environments with sparse rewards, where the agent receives infrequent feedback on its actions. Without the incorporation of entropy regularisation, agents may overestimate the value of certain states and actions, leading to suboptimal behaviour. By increasing the entropy of the policy, agents are incentivised to explore a broader range of actions, thereby improving the chances of discovering optimal strategies. (Ahmed *et al.*, 2019) demonstrated that entropy regularisation can make the optimisation landscape smoother, connecting local optima and allowing for

larger learning rates. This smoothing effect is necessary for avoiding convergence on local optima and improving the chance of finding more optimal policies in complex environments. Their experiments suggested that even with exact gradient information, policy optimisation is difficult due to the geometry of the loss function. By introducing entropy, they were able to show that the policy becomes more stochastic, which then smoothens the loss function and facilitates exploration. Ahmed et al. further highlighted that entropy encourages policies to keep exploring even after finding a suboptimal solution, increasing the likelihood of discovering better solutions during fine tuning. This adaptability is useful in environments where the conditions can change and the agent is forced to continuously learn and adjust its policy (Ahmed *et al.*, 2019). The robustness of an RL agent is also improved through entropy regularisation. Agents that explore a diverse range of states during training are better equipped to handle unusual or rare events. This robustness is a result of the agent's exposure to diverse scenarios and enables it to develop strategies that are effective across a range of conditions. As mentioned in the introduction, there are several popular strategies that have been developed over the past decade to balance exploration and exploitation aside from entropy regularisation (Liu, Gu and Liu, 2020), for example the aforementioned epsilon-greedy strategy (Mnih *et al.*, 2013), upper confidence bounds (Auer, 2002), boltzman exploration (Cesa-Bianchi *et al.*, 2017), Thompson sampling (Russo *et al.*, 2018), and for function approximation, noise-based exploration (Fortunato *et al.*, 2019).

2.5 State of the Art

Integrating attention mechanisms into RL has shown to improve both the performance and interpretability of RL models. This interpretability within RL models applied to environments of various complexities can be further increased by the integration of attention maps when applying attention mechanisms with visual inputs (Mott *et al.*, 2019). There have been various studies which have explored the application of attention mechanisms in RL. Fei et al. investigate optimising attention distributions using deep reinforcement learning and show that adjusting attention weights can improve the performance and explainability in sequence models. Their approach involves modifying attention weights to ensure that more informative parts of the input receive more focus. This method has been shown to produce more reasonable attention distributions and improve task performance across different attention

networks (Fei *et al.*, 2022). Applying attention mechanisms in multidimensional environments has also been proven beneficial. They show the importance of attention in environments with complex high dimensional data, where focusing on specific features is necessary for effective learning. Their models dynamically adjust focus based on the task at hand by leveraging attention mechanisms leading to improved performance (Niv *et al.*, 2015).

There is much to be explored in the intersection of entropy regularisation and attention augmented RL. A gap that has been identified in the research is that there has currently not been any exploration of the impact of entropy regularisation on a attention augmented RL agents. In the literature we can see that an entropy coefficient of 0.01 is often used (Mott *et al.*, 2019; Duarte *et al.*, 2024) but the reason is not given in these cases. This dissertation aims to address this gap by exploring the impact of applying various entropy regularisation coefficients to an RL agent, augmented with an attention mechanism and tested by attempting to train an agent to learn three Atari environments of various complexity. The motivation behind exploring the impact on an attention augmented agent is due to the increased interpretability the attention facilitates. We can draw more qualitative observations with the use of attention maps and potentially use these maps to aid in the debugging of the agent. The goal of this dissertation is to explore how adjusting the entropy coefficient affects learning in an attention augmented agent and to observe whether there is a generalisable entropy coefficient that can be applied in most, or all cases, that would result in an improved performance and/or stability in the agent. By addressing these questions this research hopes to contribute to the development of more robust, interpretable, and efficient RL models.

3

Chapter 3: Specification, Design and Implementation

3.1 Specification

3.1.1 Initial Research Focus

The initial focus of this dissertation was on the development of an attention augmented agent which was to be pruned dynamically using Grammatical Evolution (GE) (Ryan, Collins and Neill, 1998). The paper that this research was being modelled off was Duate et al's "Dynamically Choosing the Number of Heads in Multi-Head Attention" (Duarte *et al.*, 2024). The motivation of this research topic was due to the complexity of selecting the optimal hyperparameters for a given task. The idea was to replicate the Dynamic Heads experiment, without the addition of the second policy which they used to dynamically turn on or off attention heads based on the agent's perceived complexity of the current task. The goal was to prune unnecessary attention heads without affecting performance negatively. This was motivated by the understanding that "a large proportion of attention heads can be removed at test time without significantly impacting performance" (Michel, Levy and Neubig, 2019). However, significant gaps and inconsistencies were identified with the experiment implementation in Duarte et al's paper.

One of the first major inconsistencies identified was a discrepancy between the vision core output size (256) and the input size of the vision LSTM (64) within their model architecture. This mismatch was addressed early on by implementing an intermediate transformation layer to align the output with the LSTM's input requirements, as the logical solution to such an

issue. Despite this adjustment of adding a transformation layer, the agent continued to struggle with effective learning, indicating that there was a mistake the description of their vision core configuration.

Another key area where the authors failed to provide relevant information about their experiment implementation was the algorithm. They chose to use the Advantage Actor Critic (A2C) algorithm, but failed to provide any of the essential hyperparameters for the configuration of the algorithm, necessary for replicating the experiment. A2C operates by running multiple workers, all interacting with separate copies of an environment (Mnih *et al.*, 2016). Each worker interacts with the environment for a given number of timesteps. This is referred to as a trajectory. After all workers have completed their respective trajectory, the model then is trained on these aggregated learned experiences. This training occurs by processing the corresponding timesteps for each trajectory, based on a given batch size. All workers are then updated with the latest weights before executing the next trajectory. The number of workers, the trajectory length, and the batch size are A2C hyperparameters that can significantly affect learning. None of these hyperparameters were provided or discussed. These details will be explored further in Section 3.3.2.

Following the experiment implementation details resulted in a model that did not learn any of the environments effectively. The results of these experiments are further discussed and shown in Section 4.1**.**

3.1.2 Pivot in Research

The gaps in Duarte et al's experiment setup highlighted the need for an alternate approach to achieve stable learning. Building upon these observations, the research was pivoted towards investigating entropy regularisation as a means of improving the learning performance and stability, as well as exploring ways to maintain interpretability of the learning process. As entropy regularisation is a commonly used method for mitigating premature convergence by encouraging exploration, this was identified as a promising direction to take the research. The current experiment configuration was resulting in suboptimal learning, so alternative algorithms were investigated to see if learning could be achieved by repurposing what was currently implemented. The idea emerged to take the current A2C algorithm, which was deemed to be underperforming based on the poor implementation description, and improve upon it by upgrading it to the Proximal Policy Optimisation (PPO) algorithm which is less

sensitive to hyperparameters such as learning rate (Schulman *et al.*, 2017). This algorithm builds upon A2C and resolves many of the commonly occurring problems with the A2C algorithm. PPO is considered to be a more stable actor critic variant, which also contributed to the decision to adapt the experiment to facilitate PPO. Tests following the introduction of PPO to Duarte et al's implementation indicated that there was still no improvement in learning in any of the three Atari environments. Duarte et al's implementation was again scrutinised for inconsistencies, and it was concluded that this lack of improvement could be related to the convolution neural network (CNN) described, due to the aforementioned discrepancies between the input and output layers of the vision core.

As it was becoming more apparent that Duarte et al's CNN implementation had gaps which made further work on it a potential waste of time, it became clear that an updated CNN implementation would be needed. Additional research was then conducted to determine what models perform consistently well at RL tasks. From investigating the core models that are used by the Stable Baselines library ('Stable Baselines Documentation', 2023), the IMPALA CNN was chosen as it demonstrated consistent and stable learning across all Atari environments (Espeholt *et al.*, 2018). The IMPALA CNN was then adapted to be used instead of the Dynamic Heads configuration as the vision core CNN in the experiments in this research. This resulted in a learning model that could be used to execute the entropy experiments.

Focus was then recalibrated towards examining the impact of entropy regularisation coefficients on learning performance, as this was a topic commonly encountered during researching methods of resolving learning issues in RL agents. During the changes to Duarte et al's CNN implementation, both static and decaying entropy coefficients had been explored, which seemed like a promising approach to optimising learning efficacy. To further aid in assessing and confirming agent learning ability, the continued inclusion of a visualised attention mechanism also was highly appealing to provide qualitative insight.

This revised research aims to analyse the impact of various entropy regularisation terms across three Atari Gym environments: Seaquest, Breakout, and Space Invaders (Brockman *et al.*, 2016). This approach will provide insights into how these entropy regularisation coefficients impact learning across learning environments of varying complexities. Additionally, leveraging the attention-augmented agent allows for the generation of attention maps, which provides us a perspective into the model's decision-making process. This aspect

of the research aligns with the principles of Explainable AI (XAI) (Phillips *et al.*, 2021) with the goal to make the internal processes of the model more transparent and interpretable. This interpretable agent builds upon the functional aspects of the previous work and also explores an area of reinforcement learning and machine learning with significant theoretical and practical implications. This adjustment also allows for a more manageable and focused investigation within the available timeframe, the details of which will be explored further in Section 3.3.4.

3.1.3 Experiment Setup

The experiment setup involves testing the attention augmented agent's ability to learn three Gym Atari environments, Breakout, SpaceInvaders and Seaquest (Brockman *et al.*, 2016). These environments are chosen as they each vary in complexity. Complexity can be considered based on the action space of each of the environments, with Breakout having 4 possible actions, SpaceInvaders having 6 possible actions and Seaquest with 18 possible actions. The experiments involve testing various entropy regularisation coefficients which are applied to the loss calculation to encourage exploration. To facilitate interpretability, attention maps are also generated during training. These attention maps allow for qualitative insights to be drawn from the training process and allows for analysis of the importance of various input features, even after running for a relatively small number of timesteps, when compared to (Mott *et al.*, 2019), which ran their experiments for 1 billion timesteps.

Entropy coefficients of 0.1, 0.05, 0.01, and 0.001 are tested, as well as two entropy coefficients that decay logarithmically from 0.1-0.01 and from 0.05-0.01. An entropy coefficient of 0.01 is often used in the literature to encourage exploration so this will be considered the baseline (Mott *et al.*, 2019; Duarte *et al.*, 2024). The other static entropy terms are chosen based on the PPO hyperparameter sweep in (Eimer, Lindauer and Raileanu, 2023). PPO is the algorithm of choice and is configured with 72 workers, with each worker running through 256 timesteps per update. The optimiser used is Adam (Kingma and Ba, 2017) and the learning rate is set to 1e-5. The objective is to observe how these varying entropy coefficients influence learning and if there is a generalisable entropy coefficient that emerges as a reoccurring ideal choice in most, or all cases.

There are 4 attention heads configured for each agent. Each experiment is run for 10 million timesteps and the attention maps are generated for 400 consecutive frames every 3 million

timesteps. Each experiment is repeated three times for extended coverage. To provide an understanding of the scale of the experiments, the total number of steps run when totalling up all experiments is approximately 900 million frames.

3.1.4 Challenges

When adapting the Dynamic Heads paper's attention augmented model to use the IMPALA CNN, issues were faced which are believed to be related to the depth of the network. Deep networks in RL suffer from such problems as vanishing gradient (Hochreiter and Schmidhuber, 1997) and overfitting. The deep architecture of the network is necessary to facilitate the complexity of an attention network, but in turn has degraded the performance of the original IMPALA CNN. This increase in depth corelating to the degradation in performance occurs despite the network's enhanced capability to focus on relevant parts of the input through the use of attention. This shall be explored further in Section 4.1. This phenomenon of performance deteriorating as the network depth increases is a well-documented issue in the field of deep learning, and our observations align with findings in the literature, as is presented in (Sinha *et al.*, 2020). Here, the authors identify that increasing the number of layers in a neural network does not necessarily improve performance in RL tasks, and instead it often leads to a decrease in sample efficiency and stability due to the Data Processing Inequality (DPI) which states that information content tends to decrease through successive layers of non-linear transformations. This issue is further supported by (Schulman *et al.*, 2017) which demonstrates effective performance with relatively shallow networks. Adding to this, the paper "Learning to Walk with Deep Reinforcement Learning" also employs shallow network architectures to achieve significant advancements in robotic locomotion (Rudin *et al.*, 2022). These studies reveal a trend in the literature where shallow networks are preferred to avoid the pitfalls associated with deep architectures. Despite the degradation in learning performance, the attention augmented agent does continue to learn, albeit at a slower rate compared to the vanilla IMPALA CNN, and also provides significant qualitative benefits as it allows us to make use of attention maps. An investigation into how the networks performance degrades with increased depth will be conducted in Section 4.1, where the vanilla IMPALA CNN is compared with the IMPALA attention augmented model, as well as the intermediary stages, to clearly demonstrate the impact of the increased network depth.

3.2 Model Architecture

Attention Augmented IMPALA Agent

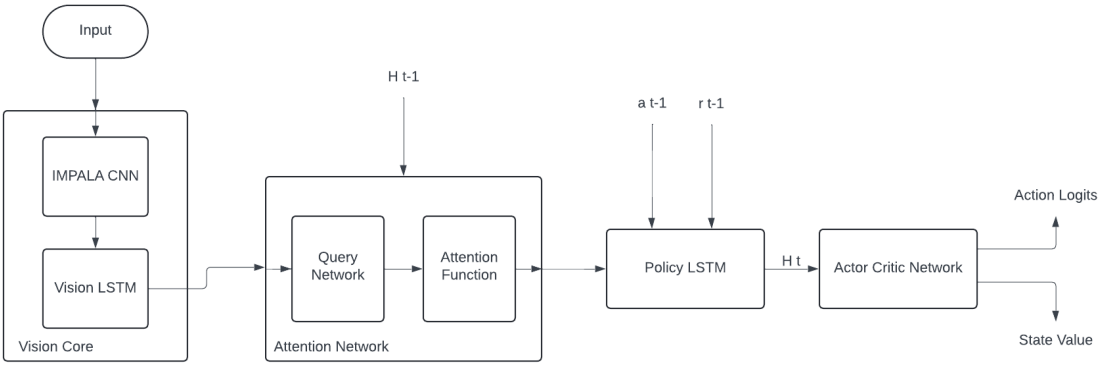

Figure 1. Attention augmented agent architecture

Below details the architecture of the proposed agent, which builds upon the work of (Mott *et al.*, 2019), combined with the IMPALA CNN and the Proximal Policy Optimisation to create a learning attention augmented agent. This agent is designed to handle sequential visual inputs and make informed decisions with the goal of learning three environments of varying complexities.

3.2.1 Vision Encoder

The vision encoder is necessary for extracting features from the visual input. It consists of two parts, a CNN to process and extract features and build a representation of the visual input, and a convolutional LSTM cell (ConvLSTMCell), which is used to maintain temporal information and context. The IMPALA CNN architecture was chosen as the convolutional neural network for the vision core due to its ability to efficiently process and extract features from image data as well as its performance on RL tasks (Espeholt *et al.*, 2018). The architecture of the model consists of three ImpalaBlock modules, which are convolutional layers combined with residual blocks. Block 1 takes a single channel input image and outputs 16 feature maps. This is passed to block 2 which increases the channels to 32. Block 3 then further processes the feature maps while keeping the channel count the same, at 32. Each ImpalaBlock convolutional layer first takes the number of in_channels and processes it to match the same number of out_channels. Batch normalisation is then applied (Ioffe and Szegedy, 2015). Max pooling with a kernel size of 3 and a stride of 2 is applied to reduce the spatial dimensions of

the feature maps. This is then followed by two residual block (ResBlock) modules which consist of a ReLU activation, a convolutional block which takes an input and outputs a shape of the same size, kernel size of 3, stride of 1 and padding of 1. Batch normalisation is then applied, followed by a ReLU, a second convolutional layer of the same configuration and one last layer of batch normalisation. The output is then concatenated with the initial input of the Resblock. The motivation behind the residual blocks comes from the ResNet paper (He *et al.*, 2015). Residual blocks help mitigate the gradient problem by allowing the input of the residual block to bypass the convolutional layers using a skip connection and are added to the output. This helps to improve backpropagation and allows for deeper networks to be trained. The residual blocks and batch normalisation layers help stabilise and accelerate training and the max pooling layers reduce the spatial dimensions making the network more computationally efficient. The motivation of using the IMPALA CNN is due to its proven effectiveness in high-dimensional visual tasks and its suitability for the diverse set of tasks presented in the Procgen Benchmark (Cobbe *et al.*, 2020).

After the CNN has extracted the features from the input, it outputs the extracted visual features to then be processed by the Convolutional LSTM cell (ConvLSTMCell) to capture temporal dependencies in the data. The ConvLSTMCell improves upon traditional LSTMs by adding convolutional operations within the LSTM's gate mechanisms, enabling the model to maintain spatial structure while learning temporal information. This makes the ConvLSTMCell suited for tasks where both spatial and temporal information are required. The ConvLSTMCell used in the vision encoder consists of 32 input channels, 32 hidden channels, and a kernel size of 3. The choice of 32 channels ensures that the temporal processing done by the LSTM is the same as the feature maps produced by the IMPALA CNN, maintaining uniform dimensionality throughout the network. The kernel size of 3 is used to provide a balance between capturing local spatial features and computational efficiency. Within the ConvLSTMCell a convolutional layer is used for each LSTM gate, as opposed to the standard matrix multiplication used in traditional LSTMs. The convolutional layer gates are input, forget, and output gates. These layers operate on both the input tensor and the hidden state to make sure that the spatial information is captured at every timestep. The input gate operation consists of a convolution of the input tensor with a learned filter followed by a convolution of the hidden state. The results of these convolutions are combined and passed through a sigmoid activation to limit how much of the new input is allowed into the cell state. Similar operations are performed by the forget gate to decide what information

from the previous cell state should be retained and respectively by the output gate to determine the new hidden state. The convolutional nature of these operations allows the ConvLSTMCell to maintain the spatial information from the input data throughout the temporal sequence, which is necessary for accurately modelling spatiotemporal sequences (Shi *et al.*, 2015). The hidden and cell states are updated at each timestep with the new states being passed on to the next timestep. This allows the network to learn how spatial patterns evolve over time.

3.2.2 Attention

After the ConvLSTMCell processes the now temporal feature maps, the next step in the model is to apply attention to focus on the relevant spatial regions of the feature maps. The attention mechanism in the model architecture is used to focus on specific parts of the input that are deemed necessary for navigating and completing the task and allows the model to focus on particular spatial locations and features. The attention network consists of a query network, a spatial basis, the application of a spatial softmax, and a function to apply the attention weights to the value tensor. The first steps in the attention mechanism involve taking the input tensor, which is the output of the ConvLSTMCell , and splitting it into the key tensor K and the value tensor V. These tensors are generated by splitting the input tensor along its channel dimension. The key tensor K contains the visual features which will be compared against a query tensor Q, which is generated from the previous output of the model. Figure 2 demonstrates the attention network to clearly show how the observation is split into key K and value V, whereas Q is calculated from the previous hidden state of the policy LSTM.

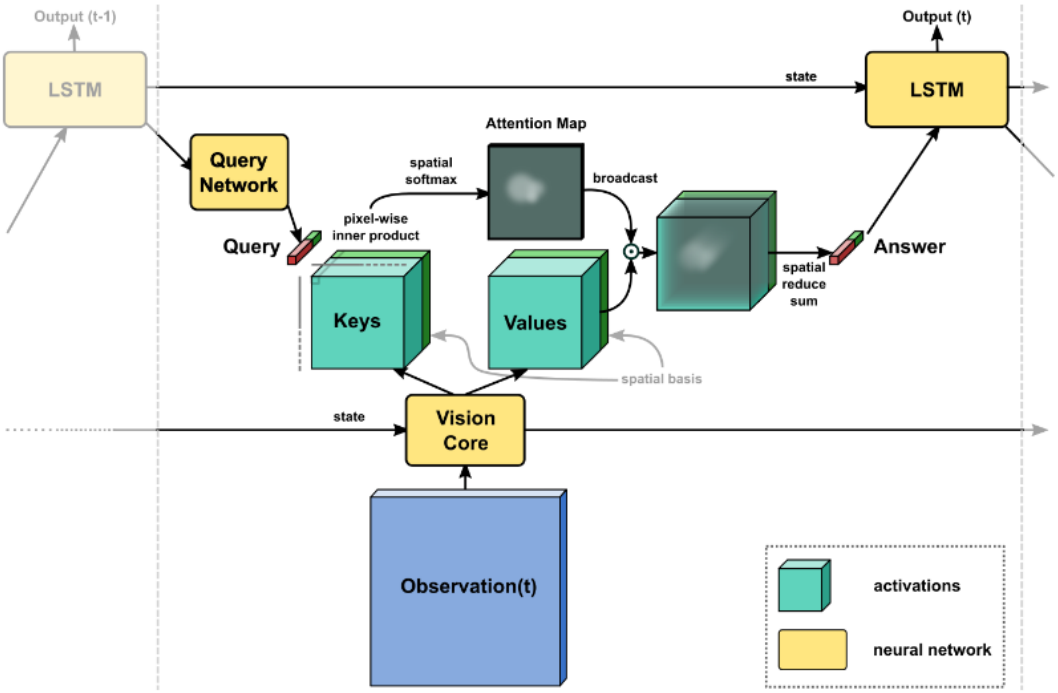

Figure 2. Overview of attention network. Adapted from *(Mott* et al., *2019)*

3.2.2.1 Query Network

The Query network is necessary for generating the query tensor Q. The Query Network is implemented as a fully connected network that processes the input vector and outputs a set of queries. The query network begins with an input vector which is the output from the ConvLSTMCell, with a dimension of 256. This input is then passed through a series of linear transformations. The first layer is a linear transformation that reduces the dimensions of the input vector from 256 to 128. The output is then passed through a normalisation layer to normalise it across the feature dimension to make sure that the activations have a mean of 0 and standard deviation of 1 to help prevent exploding or vanishing gradients. The output is then passed through a ReLU activation to add non linearity to the model, which allows the model to learn more complex patterns. A second transformation layer is then applied, maintaining the dimensions of 128. The output is then passed through a normalisation layer and a second ReLU. The final transformation layer then projects the 128 dimension feature vector into a 1280 dimensional space. This drastic increase in the dimensionality is designed to capture the patterns necessary for the attention to work correctly. This is then followed by a final normalisation layer and ReLU activation. The output is fed into a linear transformation layer that maps the outputted feature vector to target dimensions for the attention network.

The target dimensions are calculated by the sum of the key vector dimension and the spatial basis vector dimension, multiplied by the number of attention heads in the attention network. This transformation is designed to prepare the outputted data to be spread across the attention heads and allows each head to focus on a specific part of the input.

3.2.2.2 Spatial Basis

The spatial basis is necessary for adding spatial information to the extracted feature maps and enables the network to focus on specific areas of the input (Mott *et al.*, 2019). The spatial basis augments the input features with spatial encodings to allow the network to use the content and the spatial context of the input data. It is initialised with a channels parameter which determines the number of output channels after the spatial basis is applied. The spatial basis first starts by computing the positional encodings separately for height (p_h), and width (p_w). The height encoding is generated by building a matrix where each row corresponds to a position along the height of the tensor. This is then scaled by a factor proportional to the height of the tensor, and then normalised using π/h. This normalisation is necessary, as it ensures that the values of the positional encodings all exist within a range that is standardised and without it can result in the network struggling to learn meaningful patterns from the input data due to raw positional values being very high or very low. The same process is applied to the width encoding, with the width being normalised using π/w. Basis vectors U and V are defined for both height and width respectively. These are used to scale the positional encoding. The height positional encoding value is multiplied by the height basis vector, and the same is applied to the width. This results in a set of spatially varying values that encode positional information within them. These combined positional encodings are passed through a cosine function with the intent of allowing the model to understand relative positions more effectively. These transformed encodings are then combined using Einstein summation to compute the product that merges height and width information. The output is then reshaped to match the dimensions of the input tensor. The spatial basis tensor is expanded across the batch dimension so that each sample in the batch receives the encoding. Finally the calculated spatial basis is concatenated with the input tensor to augment the features with spatial information, thus providing the attention function with an input that includes both feature and positional data.

3.2.2.3 Spatial softmax

The spatial softmax function is then used to apply a softmax operation over the height and width dimensions to convert the values into probabilities which sum up to 1 across the spatial grid. This further helps the model to focus on the specific parts of the input that are most relevant. It does so by normalising the attention scores, of which indicate how much importance should be given to a particular part of the spatial grid. The input of the spatial softmax function is the output of the spatial basis and is a tensor consisting of batch size, height, width and features. Before applying the softmax, the spatial dimensions are flattened into a single dimension to transform the two dimensional spatial grid into a one dimensional vector for each batch and channel. This is required to ensure that the computed probabilities are computed across the whole grid of spatial locations, resulting in each location being treated as a separate entity. The softmax function is applied across the flattened spatial dimension and exponentiates each value to convert all values to positive as well as exaggerating the differences between them, and then normalises the resulting values by their sum to ensure they all add up to a total of 1. Once the softmax operation has completed, the tensor is reshaped back to its original shape of the input to ensure that the computed tensor is compatible with the rest of the model.

3.2.2.4 Applying Attention Weights to Value Tensor

The final part of the attention mechanism is the application of the attention weights A, which is the output of the spatial softmax, to the value tensor V. V contains the features that will be weighted by the attention scores. This is achieved by the height and width of A being flattened into a single dimension and the tensor is transposed to prepare it for matrix multiplication. The spatial dimensions of V are also flattened into a single dimension. Matrix multiplication is then applied between A and V to combine the attention scores with the features to weight them based on the importance derived from the attention function. This results in a new tensor where each individual feature has been scaled according to its attention score.

3.2.2.5 Answer Processor

Immediately following the application of the attention weights to the value tensor is the answer processer, whose role is to concatenate the attention weighted features, with the query tensor and the previous reward and action. This is an intermediary step and not necessarily part of the attention calculation. The motivation behind the answer processor is to take the results of the attention mechanism and combine it with other contextual data to enable the network to make more informed decisions. The query tensor is added to this tensor to allow the model to have context on what the model was asking for during the attention calculation. The previous reward and action are also included in this tensor as they provide context for the model on past decisions made. After concatenation, the resulting tensor is passed through a series of fully connected layers within the answer processor. Before the tensor is passed through the fully connected layers, it is first down sampled to a lower dimension through a linear transformation and the output is reduced to 512 units. A ReLU activation is then applied for non-linearity, followed by a second linear layer which further down samples the output to the size of the hidden state, 256. This is required to allow the output to be used in the next stage of the model.

3.2.3 Policy Core LSTM

The policy core is another necessary part of the model which integrates with the attention network to generate the query tensor Q for the next state. The policy core is designed to store temporal information about past states through its hidden state h, which changes over time as states are observed. The LSTM also contains a cell state c, which functions as long-term memory and is to be updated more conservatively than the hidden state, which makes it less impacted by rapid changes in the environment. The LSTM takes advantage of recurrent processing, where the model takes the previous hidden state and previous cell state, as well as the current input and uses this to update its current hidden state and cell state. Both updated hidden and cell states are then stored for use when model processes the next state. The hidden state tensor is also passed on to the next part of the model where it will be used in to calculate the query tensor Q for the attention network.

3.2.4 Actor and Critic Networks

The actor and critic network is the final part of the overall model and both actor and critic operate in parallel by taking in the same input and producing two separate results. The role of the actor is to determine which action to take in a given state. It can also be considered the policy, as it maps a state to an action. The actor outputs a probability distribution of the possible actions which then can be sampled to select the action to take. The actor is implemented as a fully connected linear layer which takes the output of the policy core LSTM which is a size of 256 and maps it to a vector of actions logits with each element of the tensor representing a possible action in the environment. The logits are then passed through a LogSoftmax activation to convert the logits into a log probability distribution over the actions. The critic network looks at the current state and provides an estimate of the expected future rewards, based on the current policy. This is very useful as it enables the advantage function of the model to evaluate how much better or worse an action taken by the actor is compared to the optimal action perceived by the critic following the current policy. The architecture of the critic network takes the output of the policy core LSTM as input and outputs a single value known as the state value. We can then use this value in our advantage calculation as a baseline to compare our current selected action against. The advantage function used will be discussed further in Section 3.3.3.

3.3 Implementation

3.3.1 Model Workflow

Previously we have delved in detail into all of the components of the model. As the model consists of a complex architecture, we will now examine on a higher level at how all of the individual components work together to form the greater network and how an individual state is processed through the network.

The model takes an input with the shape of (72, 84, 84, 1). The batch size of 72 is due to the model processing a frame from the 72 trajectories simultaneously. Each of these trajectories is interacting with a separate version of the environment simultaneously. Processing these trajectories in parallel, treating each of these trajectories as a separate instance allows for the model to learn from a diverse set of experiences and improves its ability to generalise. The

spatial dimensions are consisting of a height and width of 84 which have been reduced to save on computational resources when training the network. The input has also been converted to greyscale, hence the channel dimension of 1. This was also used to reduce the computational complexity, as the necessary information to learn the environment is still present within the frame.

The input is first passed through a set of convolutional layers in the form of ImpalaBlocks to extract features from the input state, as described in Section 3.2.1. After extracting the feature maps from the input state, the model applies a Convolutional LSTM to the feature maps to build up a temporal understanding of the sequence of frames. The output of the convolutional LSTM is then split in key and value tensors, represented as K and V respectively. The keys are used to learn the information that the model will query during the attention process, and the values contain the actual data that will be weighted based on the attention scores. Both K and V are augmented with spatial information using the spatial basis function, as described in Section 3.2.2.2. The model then generates the queries Q from the previous hidden state passed from the policy core LSTM, as described in section 3.2.3. These queries are matrix multiplied by the keys and then normalised using a spatial softmax function to compute the attention scores, as described in section 3.2.2.3. The attention scores are then applied to the value tensor V which results in a weighted representation of the input to highlight the most important learned features, as described in section 3.2.2.4. This is then combined with the query tensor Q, the previous rewards and the previous actions and this combined representation is passed through the answer processor which refines this information by passing it through a fully connected network to reduce its dimensions, as described in section 3.2.2.5. This output is then fed into the policy core LSTM, combining the previous hidden and cell state with the current input and which updates these internal states based on the new input, as described in Section 3.2.3. The output of the policy core LSTM is then used by the actor and critic networks, as described in Section 3.2.4. The actor uses this input to generate a log probability distribution over the possible actions from which the model can select an action to take. The critic takes the same input and generates an estimate of the value of the current state to guide learning by evaluating what the best possible action in the current state is based on the current policy.

3.3.2 Algorithm

3.3.2.1 From A2C to PPO to Address Suboptimal Policies

Initially the Advantage Actor-Critic (A2C) algorithm was implemented as the algorithm for the agent, as this was algorithm of choice used in (Duarte *et al.*, 2024), which was the experiment originally being replicated. The A2C algorithm combines both policy based and value based methods of reinforcement learning and is a synchronous variant of the Asynchronous Advantage Actor Critic (A3C) algorithm. Due to lack of specific configuration settings Duarte et al's implementation, such as number of trajectories per update, number of steps per update and batch size, as well as poor performance during training, a decision was made to upgrade the A2C algorithm to the Proximal Policy Optimisation (PPO) algorithm as mentioned in Sections 3.1.1 and 3.1.2. This decision to pivot to PPO was motivated by the need to address issues that arose from the Dynamic Heads A2C implementation as well as the fact that PPO essentially has been built upon A2C to address many of these limitations. Because of this, the conversion of A2C to PPO is straightforward, adding various optimisations to the overall algorithm to stabilise learning.

The main idea behind A2C revolves around the advantage function, which is derived from the Q value Q(s,a). The Q value represents the expected return of taking action a in state s, and can be broken down into the state value function V(s) and the advantage value A(s,a):

$$Q(s,a) = V(s) + A(s,a)$$

The advantage function can then be derived as

$$A(s,a) = Q(s,a) - V(s)$$

Given the Q value can be expressed as

$$Q(s,a) = r + \gamma V(s')$$

where r is the immediate reward and γ is the discount factor, the advantage function becomes:

$$A(s,a) = r + \gamma V(s') - V(s)$$

3.3.2.2 Limitations of A2C

Within the A2C algorithm, the actor is updated directly using gradients of the advantage function. These gradients are not constrained and thus, can result in extreme changes in the policy between updates. This leads to instability where the policy can start to diverge, especially in complex environments. A2C is also quite sensitive to the choice of learning rate and requires careful selection and thorough experimentation to find a learning rate which enables the policy to converge successfully. Due to the lack of specific A2C hyperparameter documentation in (Duarte *et al.*, 2024), a suitable configuration which resulted in effective learning using A2C was unable to be found.

3.3.2.3 Proximal Policy Optimisation (PPO)

PPO was chosen as a replacement for the A2C algorithm as it uses various techniques to mitigate the issues described in Section 3.3.2.2. PPO retains the core properties of the A2C algorithm i.e. using an actor critic framework with an advantage function to calculate how much better or worse an action was compared to the understanding of the current policy. It also typically also performs synchronous updates which involve running multiple environments in parallel and waiting until all environments complete their trajectories before updating the policy and value function. The enhancements present in PPO which stabilise learning are; the use of a clipped loss function to limit how much the policy can change after each update; the use of multiple training epochs on the same batch of experiences; and the use of a more sophisticated advantage function, referred to as Generalised Advantage Estimation (GAE) (Schulman *et al.*, 2017), which will be discussed in further detail in Section 3.3.3.

The most significant improvement in the PPO algorithm is the use of a clipped loss function to ensure that the updates are policy updates are stable and do not vary too much from the previous update. This clipping is applied to the calculation of both the policy and the critic loss calculation. To achieve this within the policy loss calculation, the log probabilities of the actions under the current policy are used to calculate the likelihood of the agent taking each action according to the current policy. The log probabilities of the actions under the old policy are also used as a baseline to measure how much the policy has changed between updates. The ratio between the new and old probabilities is calculated by taking the exponent of the difference between the log probabilities. The reason for calculating this ratio tells us how much more or less likely the current policy is to take the same actions compared to the

previous policy. An unconstrained update to the policy is then calculated along with a constrained update where the ratio is clamped between a range or 0.8 and 1.2 to prevent the ratio from moving too far from 1. The minimum of the unconstrained update and the constrained update is then selected to maintain a conservative policy update. The mean is then taken across all actions in the batch and this value is used to update the policy loss.

This same clipping is also applied to the critic loss calculation in a similar manner. The old state estimates, calculated before the last update are used as a baseline and are compared to the current value estimates to track how much the current estimates have changed after the update. A clipped state estimate is then computed by adding the difference between the current and previous value estimates. The difference is clamped between -0.2 and 0.2. Two potential losses are then calculated, the clipped loss, which is the squared difference between the returns and the clipped state estimates, as well as the unclipped loss, which is calculated using the squared difference between the returns and unclipped state estimates. The maximum of the unclipped and clipped loss is then taken as the more conservative update to ensure that the more significant critic loss is corrected by the update. The mean of the loss across the whole batch is then taken and multiplied by 0.5 to appropriately scale how much of the critic loss should contribute to the overall loss calculation.

PPO also improves sample efficiency by optimisation steps on the same shuffled batch of experiences for multiple epochs. In these experiments, three iterations are used on each batch. This reuse of data reduces the need to generate new experiences for effective learning to occur. Performing multiple passes on the same batch of data also helps to reduce variance as the agent has several chances to adjust to the new set of experiences. Sample efficiency is enhanced further by the use of mini batches. During each of the mentioned epochs, the batch of data is divided further in smaller batches and updated on each mini batch sequentially to smooth out the updates and make more optimal use of the available data. Mini batch updates tend to lead to more accurate gradient estimates compared to updating the model with the whole batch at once. A diagram of PPO illustrates this in Figure 3.

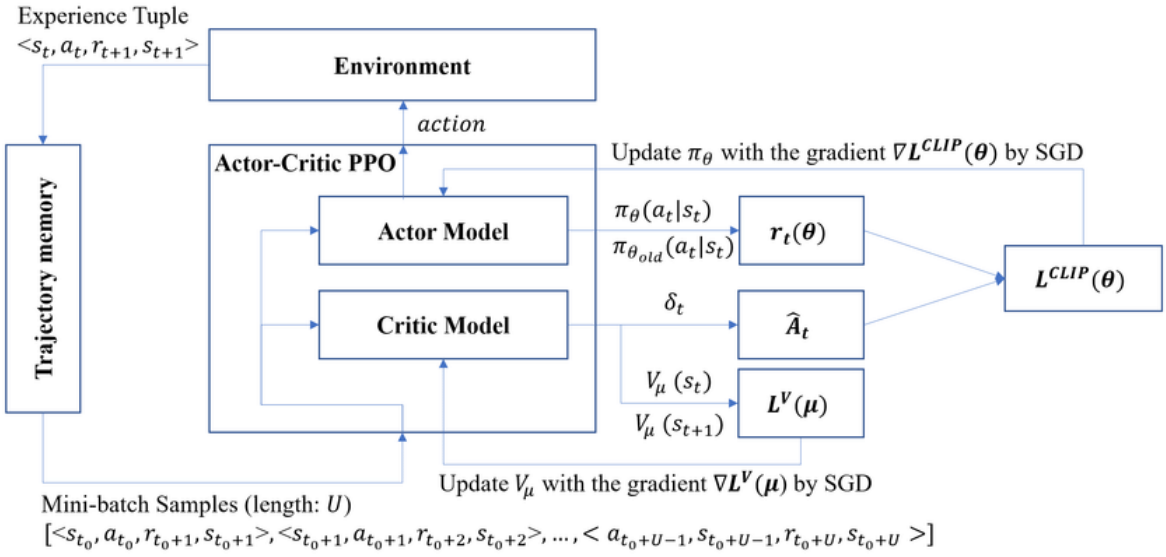

Figure 3. The actor-critic Proximal Policy Optimization (PPO) algorithm. Adapted from *(Lim* et al., *2020)* .

3.3.3 Generalised Advantage Estimation (GAE)

PPO also makes use of an improved advantage function, GAE, which is used to balance the bias variance trade-off in the advantage function estimation. If we are to use only the immediate rewards in the advantage calculation it can lead to high variance, and using long term rewards can introduce bias. GAE uses a hyperparameter, lambda λ, to control how much of the future rewards are factored in to the advantage calculation. Setting λ to 1 makes GAE function as the standard advantage function, and considers all future rewards equally, which chosen in (Duarte *et al.*, 2024), whereas a λ of 0 means that only the immediate reward and the next states value will be factored in to the advantage calculation. In this experimental setup a λ value of 0.95 was chosen to allow for some variance without treating all rewards as equal. γ is the discount factor gamma, which is used to compute the currently perceived value of the future reward. In the experiments a γ value of 0.999 is used to slightly discount future rewards to ensure a priority for immediate rewards. To calculate GAE, we iterate backwards through the rewards and calculate the advantage at each time step. The reason for iterating backwards through each time step is because the current advantage is dependent on the future advantages, which are known only after processing the later time steps.

GAE is calculated recursively as

$$\hat{A}_t = \delta_t + \gamma\lambda\hat{A}_{t+1}$$

γ is the discount factor, which is used as a hyperparameter to scale the current value of future rewards

λ determines how much of the future advantage estimates are factored in to the current advantage estimate

When λ = 0 it only considers immediate reward and the next state value, same as temporal difference error, shown below

When λ = 1 it factors all future rewards up to the end of the trajectory/episode, same as the standard advantage calculation

δ_t is the temporal difference error, calculated as

$$\delta_t = r_t + \gamma V(s_{t+1}) - V(s_t)$$

r_t is the reward received

$V(s_{t+1})$ is the estimated value of the next state

$V(s_t)$ is the estimated value of the current state

4

Chapter 4: Results

4.1 The Impact of Increasing the Complexity of an RL Model

4.1.1 Results Discussion on the Impact of Increasing the Complexity of an RL Model

In the experiments, five models were tested in each of the three Atari environments and results are shown in Figures 4, 5, 6, 7 and 8. Each of these figures compare five implemented models, consisting of: Duarte et al's implementation, which is the Dynamic Heads Paper Model; the standard IMPALA Model without any modifications to the architecture of the model, the IMPALA model with the Policy LSTM included; the IMPALA with Policy LSTM and Convolutional LSTM; and the final IMPALA Attention Model, which includes both the policy and convolutional LSTMs and the attention network. From examining Figures 4, 5, 6, 7 and 8, we can see some trends emerging in all three environments. The first noticeable observation is that the Dynamic Heads Paper Model implementation did not learn any of the environments. This is valuable to display as we can use this as a lower bound of what a model which is not learning the environment looks like. Another observation is that the IMPALA Model performs best in all three environments, which is not surprising, as this is a state-of-the-art model and is often used in RL tasks. This can be seen as our upper bound of what a model which is learning the environment looks like. The IMPALA with Policy LSTM model adds an extra layer of complexity to the unmodified IMPALA Model, and from examining performance across all three Atari environments, we can see that there is a slight drop in performance. The remaining models follow this trend of decreased performance correlating to model complexity, as we can see this occurring following the introduction of the policy and convolutional LSTMs to the IMPALA Model, and even more significantly across all

environments with the introduction of the attention network. This observed degradation of learning performance across all environments aligns to (Sinha *et al.*, 2020), where they identify that increasing the number of layers in a neural network generally does not improve performance in RL tasks; conversely it often leads to a decrease in performance as the depth of a network is increased. Each of the additions to the IMPALA architecture is increasing the depth of the network, and what results is a visible degradation of network performance and cumulative reward across all three of the Atari environments. Although there is a decrease in model performance, learning is still observed in all scenarios other than the Dynamic Heads Paper Model. Due to the higher performing models shown in Figure 8, it is difficult to see whether learning is occurring in the IMPALA Attention Model, so Figure 10 isolates the IMPALA Attention Model and the Dynamic Heads Paper Model to allow for a more granular comparison. Although learning is quite poor in the IMPALA Attention Model, the cumulative reward is still increasing over time, albeit very slowly. The only variation between the Dynamic Heads Paper Model compared to the IMPALA Attention Model is the use of the IMPALA CNN as part of the vision core instead of the CNN implementation described in (Duarte *et al.*, 2024). A reason for the improved performance when using the IMPALA CNN could be due to the fact that it uses residual blocks (He *et al.*, 2015) which facilitate networks to go deeper due to its use of concatenating the identity of the input to the output of each residual block. The key observation here is that although learning in the IMPALA Attention Model is hindered by increased complexity, it is still occurring. This model is used for all subsequent experiments.

4.1.2 Breakout – Average and Best Runs for Models of Various Complexities

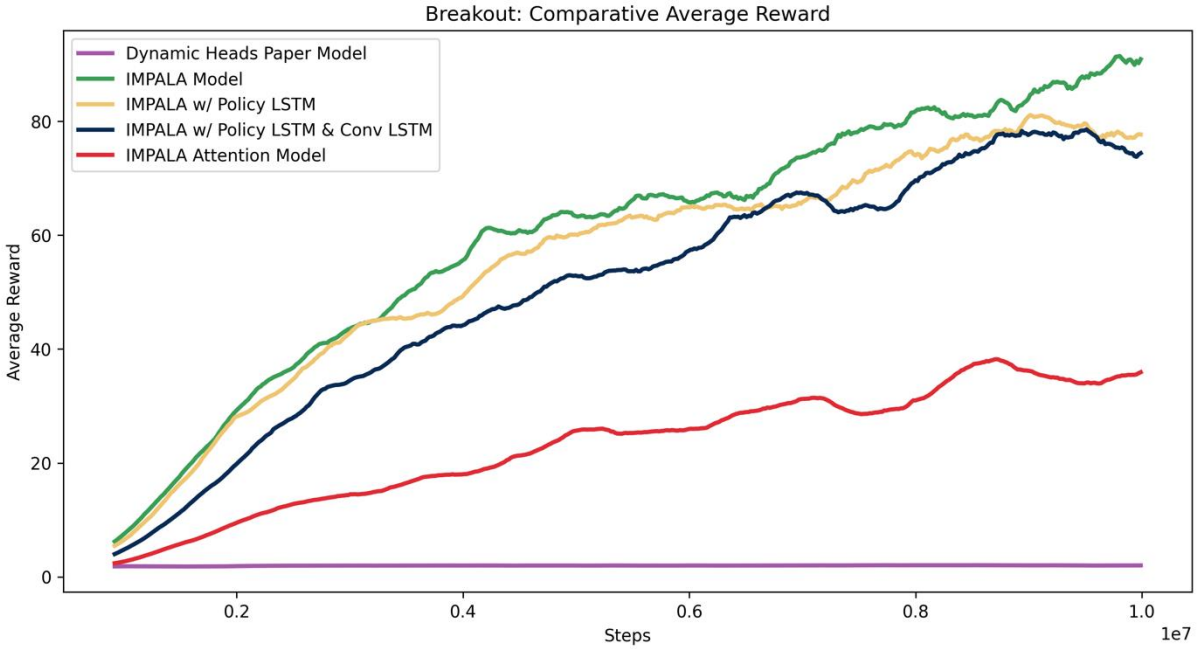

Figure 4. Breakout: comparative average reward of various models

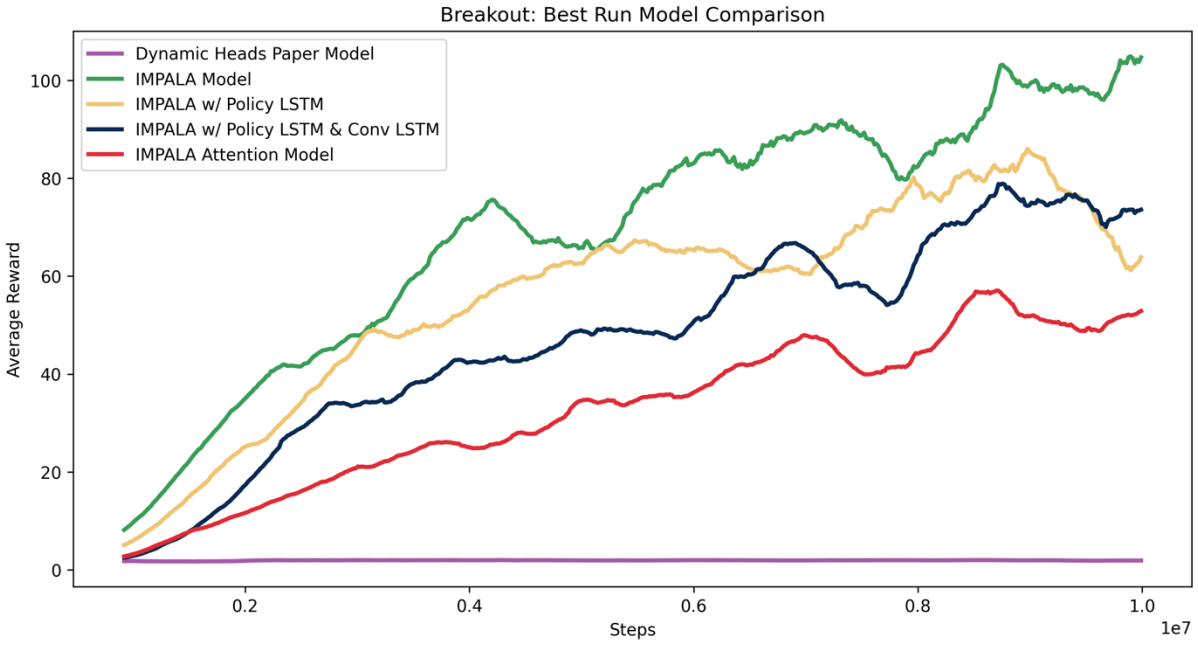

Figure 5. Breakout: comparative best run of various models

4.1.3 SpaceInvaders - Average and Best Runs for Models of Various Complexities

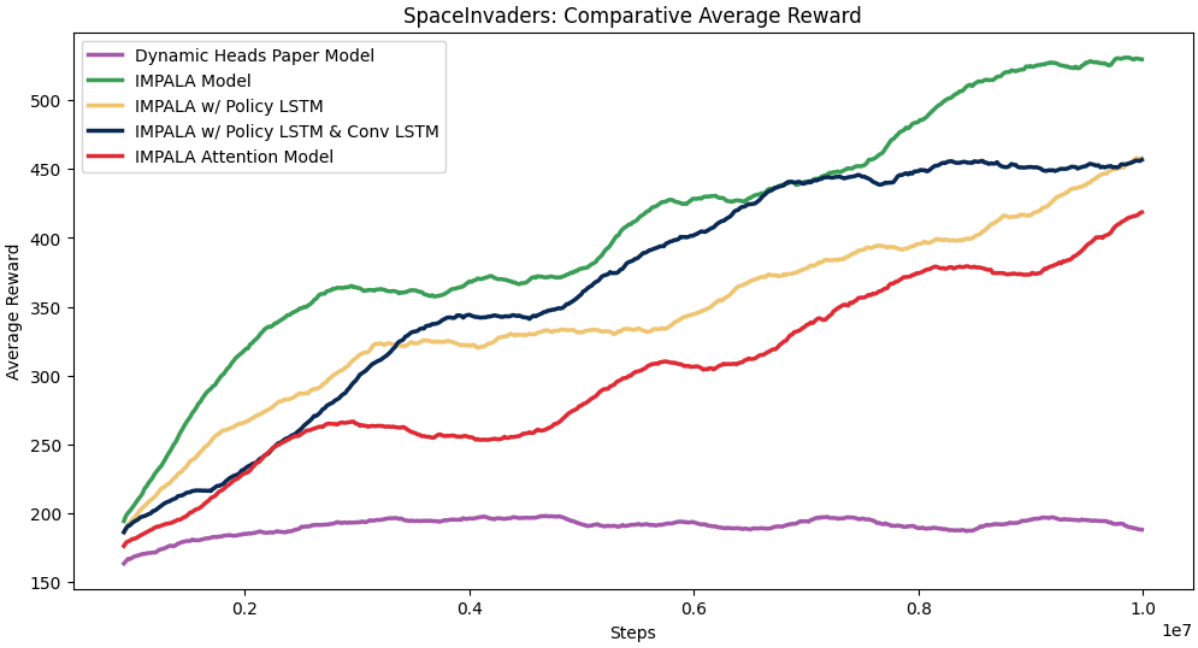

Figure 6. SpaceInvaders: comparative average reward of various models

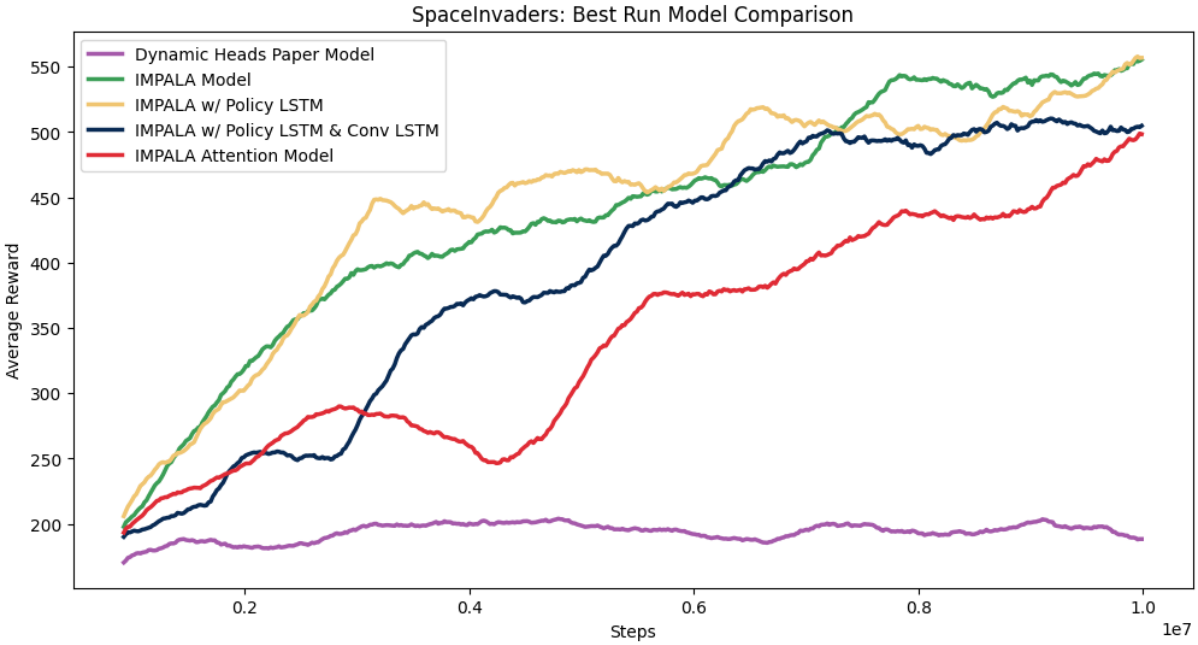

Figure 7. SpaceInvaders: comparative best run of various models

4.1.4 Seaquest - Average and Best Runs for Models of Various Complexities

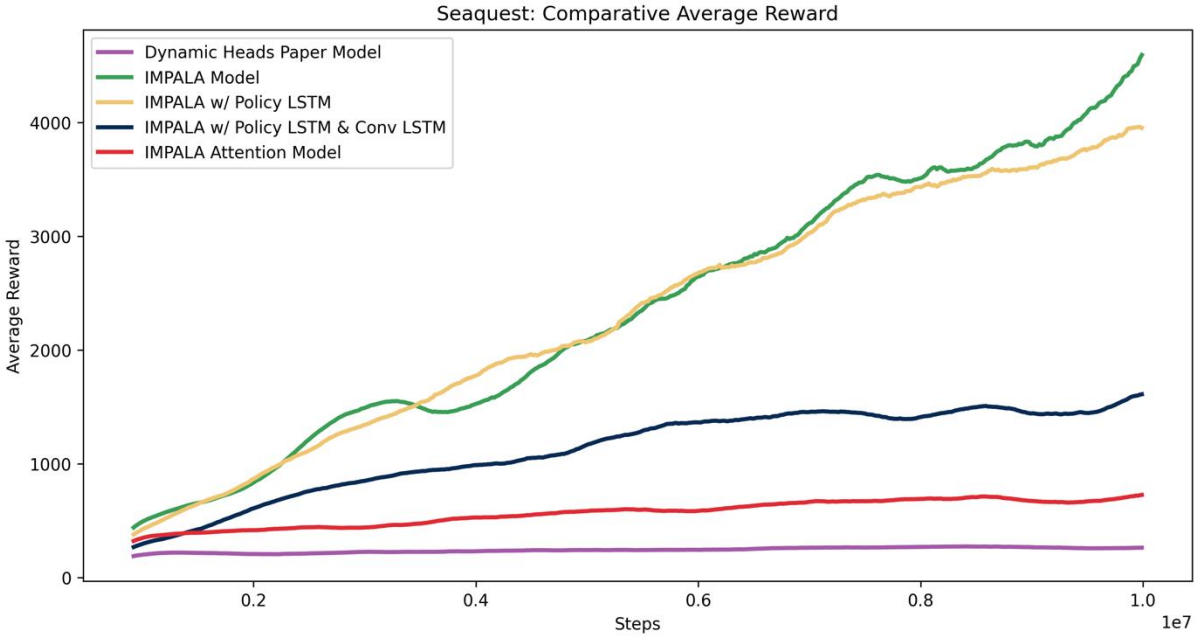

Figure 8. Seaquest: comparative average reward of various models

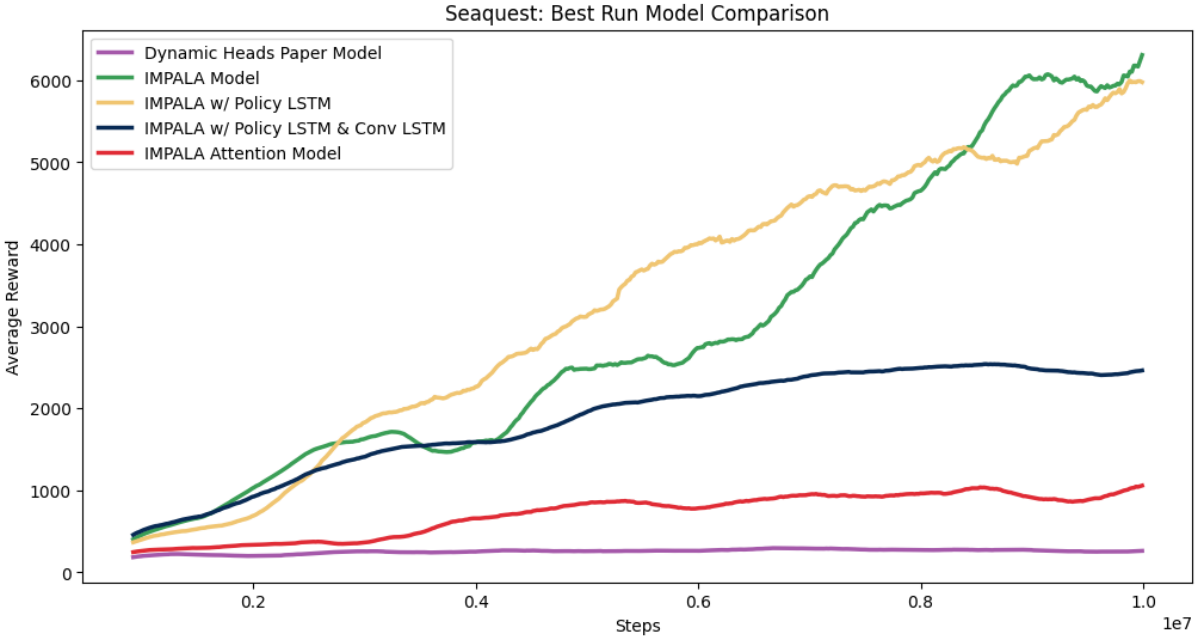

Figure 9. Seaquest: comparative best run of various models

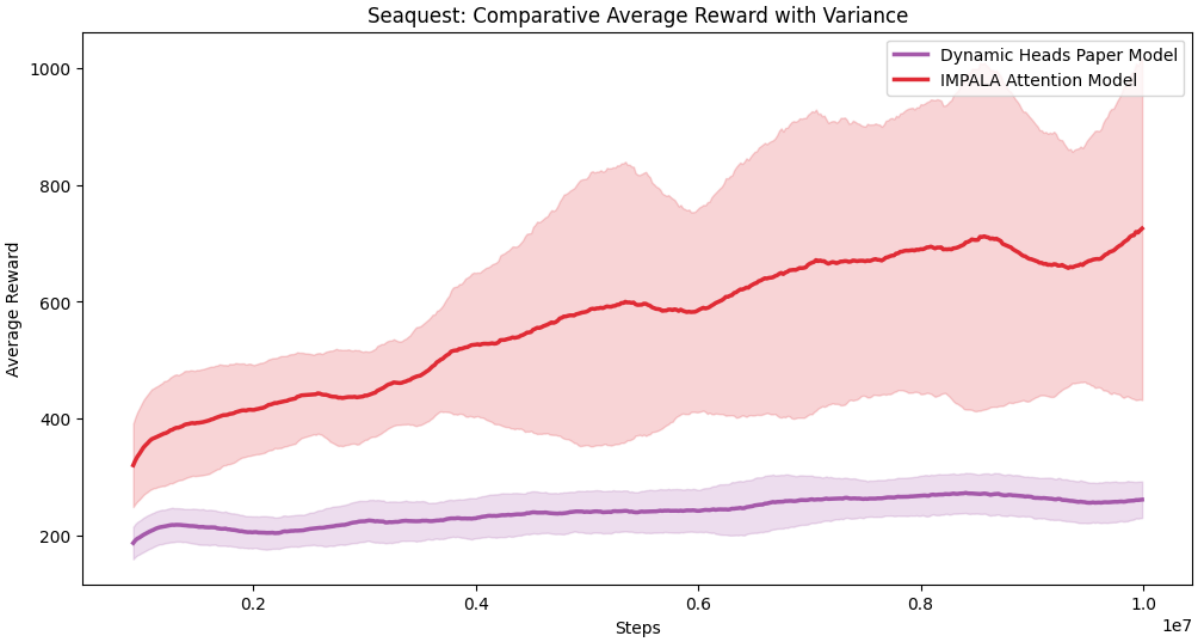

Figure 10. Seaquest: comparative average reward, examining Dynamic Heads Paper model and the IMPALA attention agent

4.2 The Impact of Varying Entropy Coefficients on an Attention Augmented Agent

4.2.1 Discussion on the Impact of Varying Entropy Coefficients on an Attention Augmented Agent

From examining Figures 11, 13 and 15 there are several observations that become apparent as a result of running an agent configured with various entropy coefficients three times and comparing the average cumulative rewards over time. The first observation is that generally the decaying entropy coefficients perform consistently better than the static entropy coefficients across all environments. In Breakout, the Decay 0.05 – 0.01 results in the best performance, and the Decay 0.1 – 0.01 also marginally outperforms the static 0.01 entropy coefficient by end of training. A similar observation can be drawn from the average rewards over time for the SpaceInvaders environment, shown in Figure 13, where the Decay 0.05 – 0.01 results in the highest cumulative reward by the end of training, with the Decay 0.1 – 0.01 following closely. In the average results for Seaquest, Figure 15, it can be clearly observed that the Decay 0.1 – 0.01 performs noticeably better than any of the other entropy coefficients. This indicates that the using an entropy decay in these experiments can improve stability which could be due to the more controlled decrease in entropy over the course of training, which can be observed in detail in Figures 20, 21 and 22.

Other than the notable performance of the decaying entropy coefficients, it is worth mentioning that the static 0.05 coefficient also showed promising results. In Figure 13 we can see that static 0.05 coefficient marginally outperforms the Decay 0.1 – 0.01 for SpaceInvaders. Furthermore, in Figure 15, we can see that Seaquest also had considerable performance for the static 0.05 coefficient, as it achieved higher average cumulative reward over time than the Decay 0.05 – 0.01. Interestingly, in Figure 11 for Breakout, an environment considered relatively less complex, the static 0.05 coefficient was one of the worst performing coefficients.

There is also a benefit to running RL agents multiple times due to their inherent stochasticity, which can result in different learning trajectories without the adjustment of any hyperparameters. Due to this constraint, it is necessary to run a model multiple times and observe the best performing agent to have a better understanding of the upper bounds of the

agent's capabilities given a set of hyperparameters. Observing the best iterative run of the models across the three environments in Figures 12, 14 and 16 reveal additional insights into each agent's performance. In the literature, an entropy coefficient of 0.01 is often used (Mott *et al.*, 2019; Duarte *et al.*, 2024). If we take this as a baseline, we can see that in all three environments, an entropy coefficient of 0.01 results in relatively decent performance, potentially justifying why this entropy coefficient value is commonly used. It can also be observed that in all three environments, for both the average and best performances, the Decay 0.05 – 0.01 consistently results in a higher cumulative reward than the static coefficient of 0.01 by the end of training, albeit marginal in all three cases.

Comparing the average runs and best runs for the various entropy coefficients can provide further insight. For example, while some of the best runs of the static entropy coefficients resulted in high cumulative reward, it can also be observed that the average performance of these entropy coefficients did not demonstrate the same results, especially when compared to the average of the decaying coefficients, i.e. 0.001 in Breakout, 0.01 in SpaceInvaders, and 0.01 in Seaquest. This indicates that the performance of the agent is less stable i.e. higher variance between runs, in the static coefficient agents. The individual agent runs configured with the various entropy coefficients are depicted for Breakout, SpaceInvaders and Seaquest in Figures 17, 18 and 19 respectively for further reference as to how the average and best runs were determined.

4.2.2 Breakout – Average and Best Run Comparison of Varying Entropy Coefficients

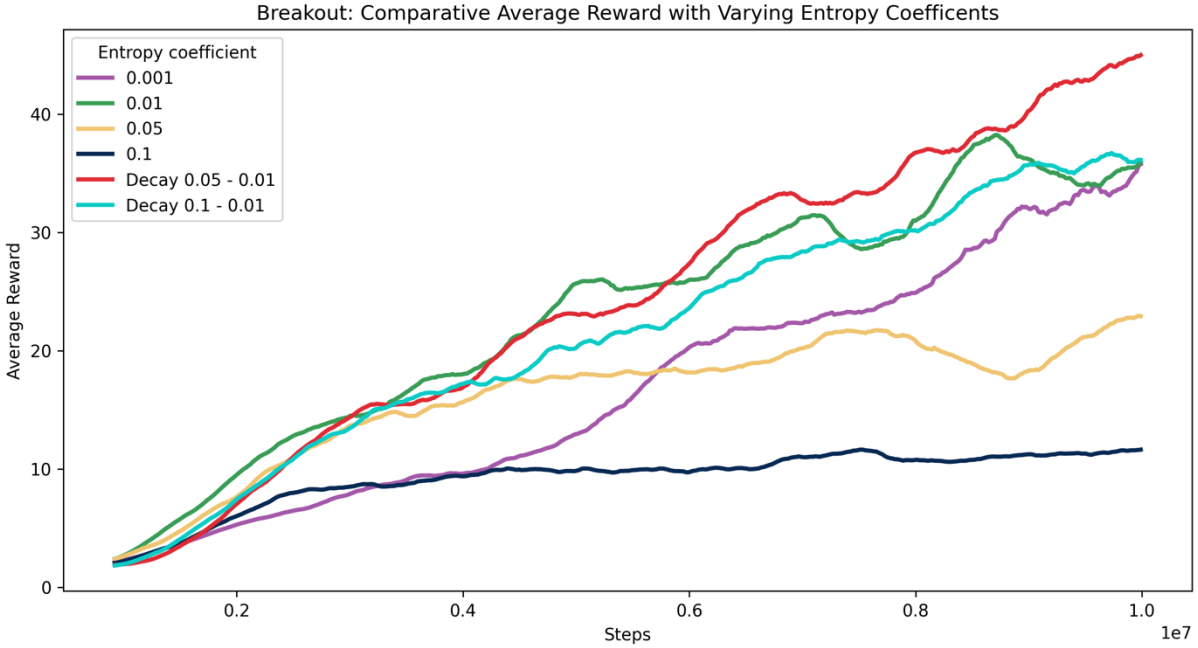

Figure 11. Breakout: Comparative average reward with varying entropy coefficients

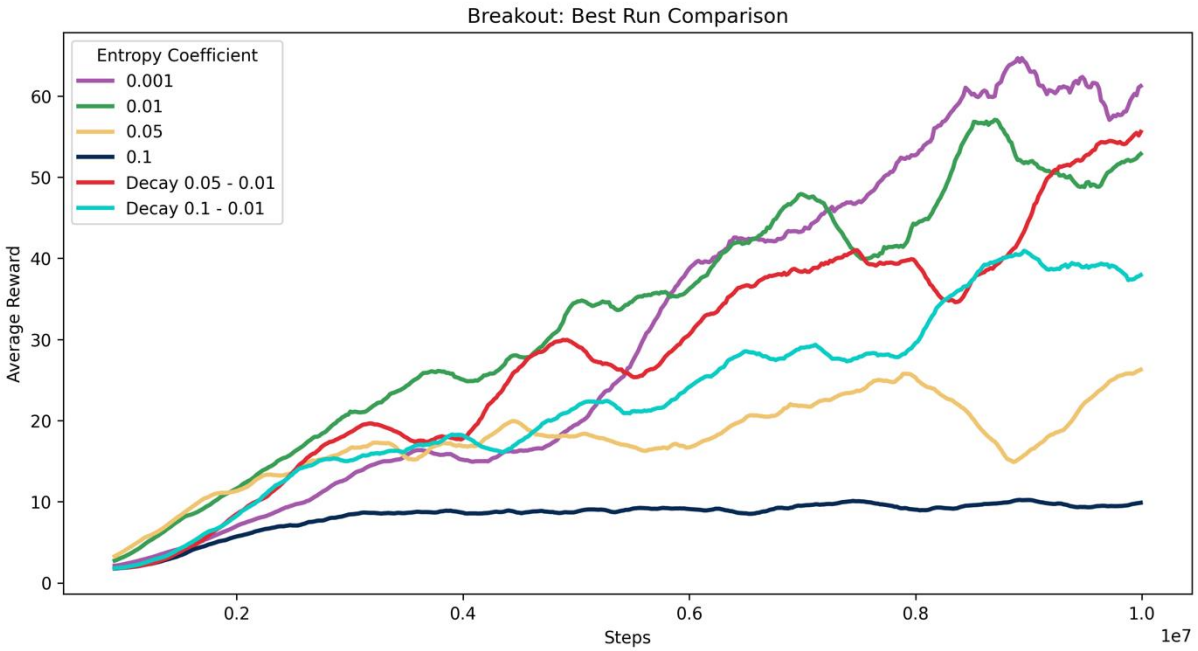

Figure 12. Breakout: Comparative best run with varying entropy coefficients

4.2.3 SpaceInvaders - Average and Best Run Comparison of Varying Entropy Coefficients

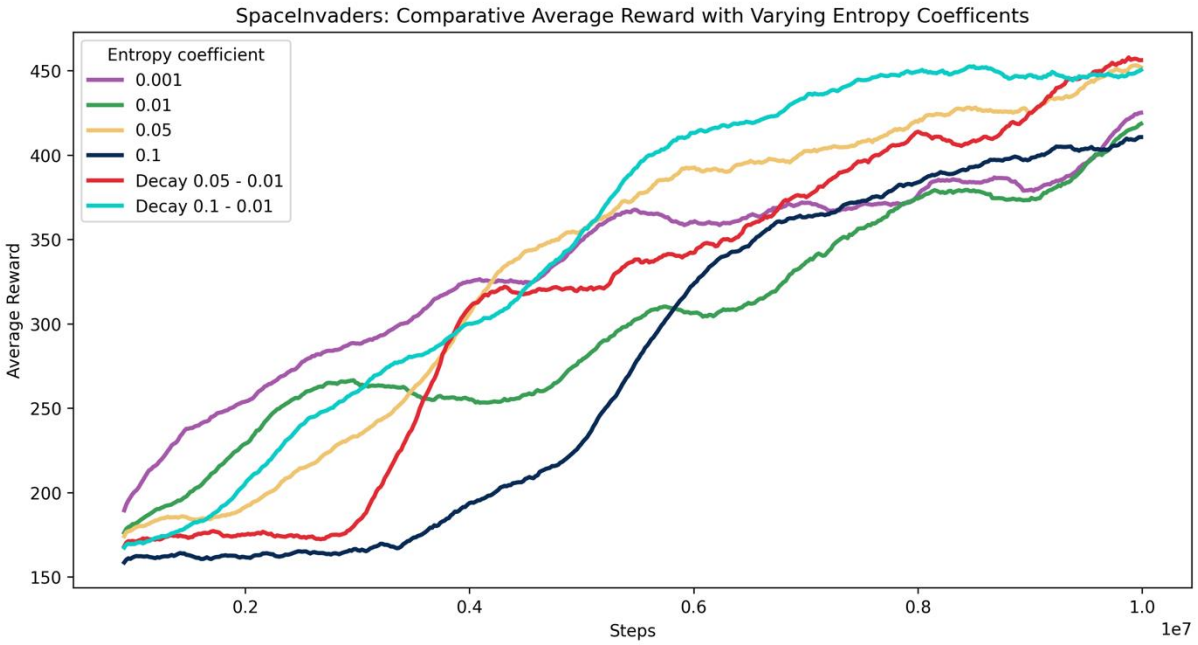

Figure 13. SpaceInvaders: Comparative average reward with varying entropy coefficients

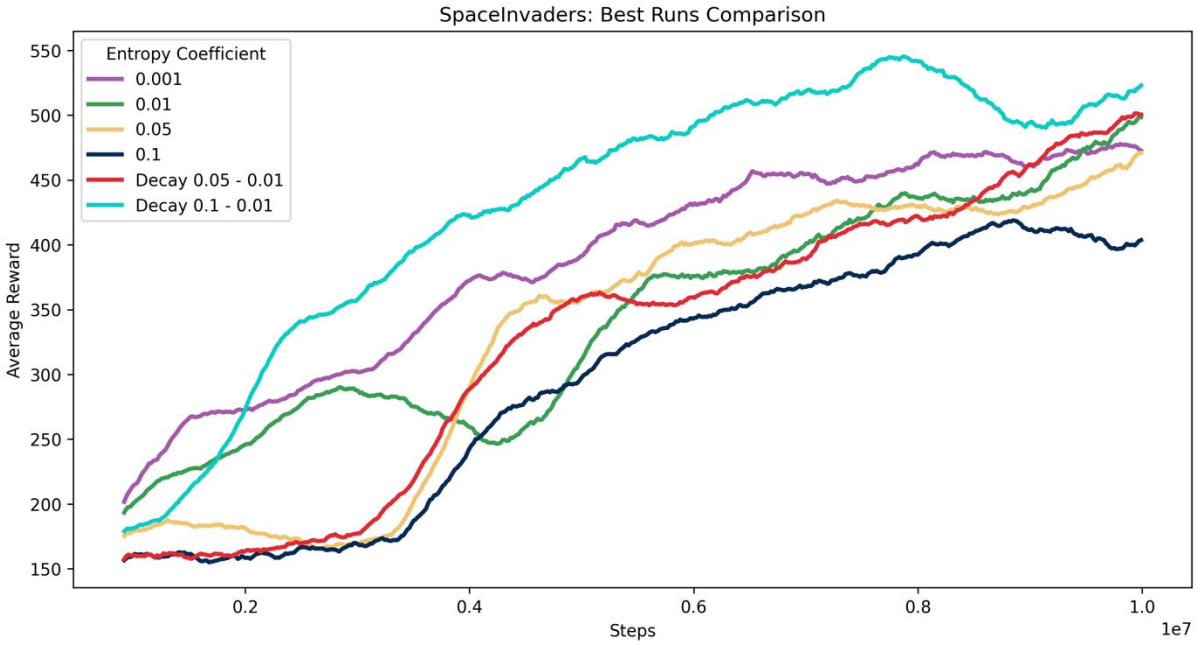

Figure 14. SpaceInvaders: Comparative best run with varying entropy coefficients

4.2.4 Seaquest - Average and Best Run Comparison of Varying Entropy Coefficients

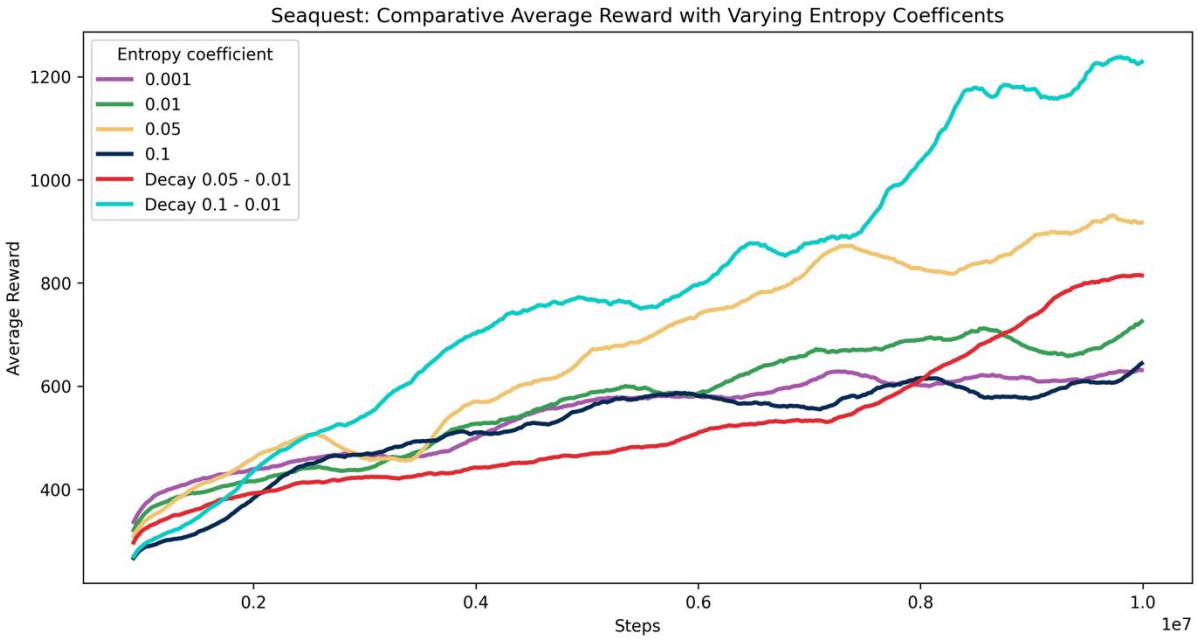

Figure 15. Seaquest: Comparative average reward with varying entropy coefficients

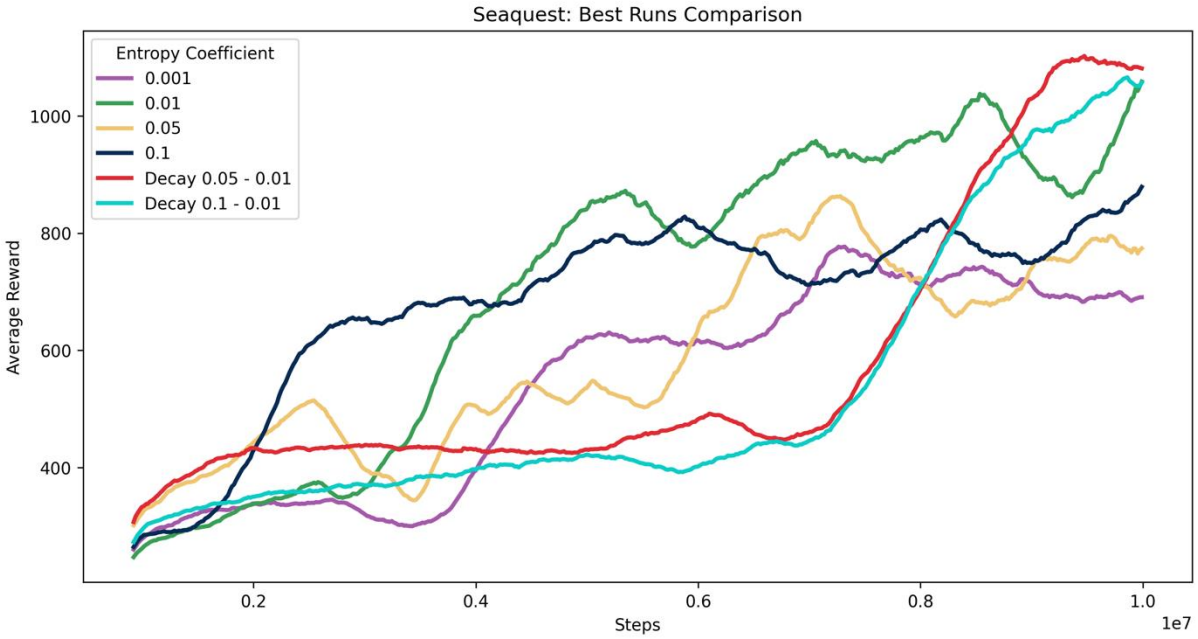

Figure 16. Seaquest: Comparative best run with varying entropy coefficients

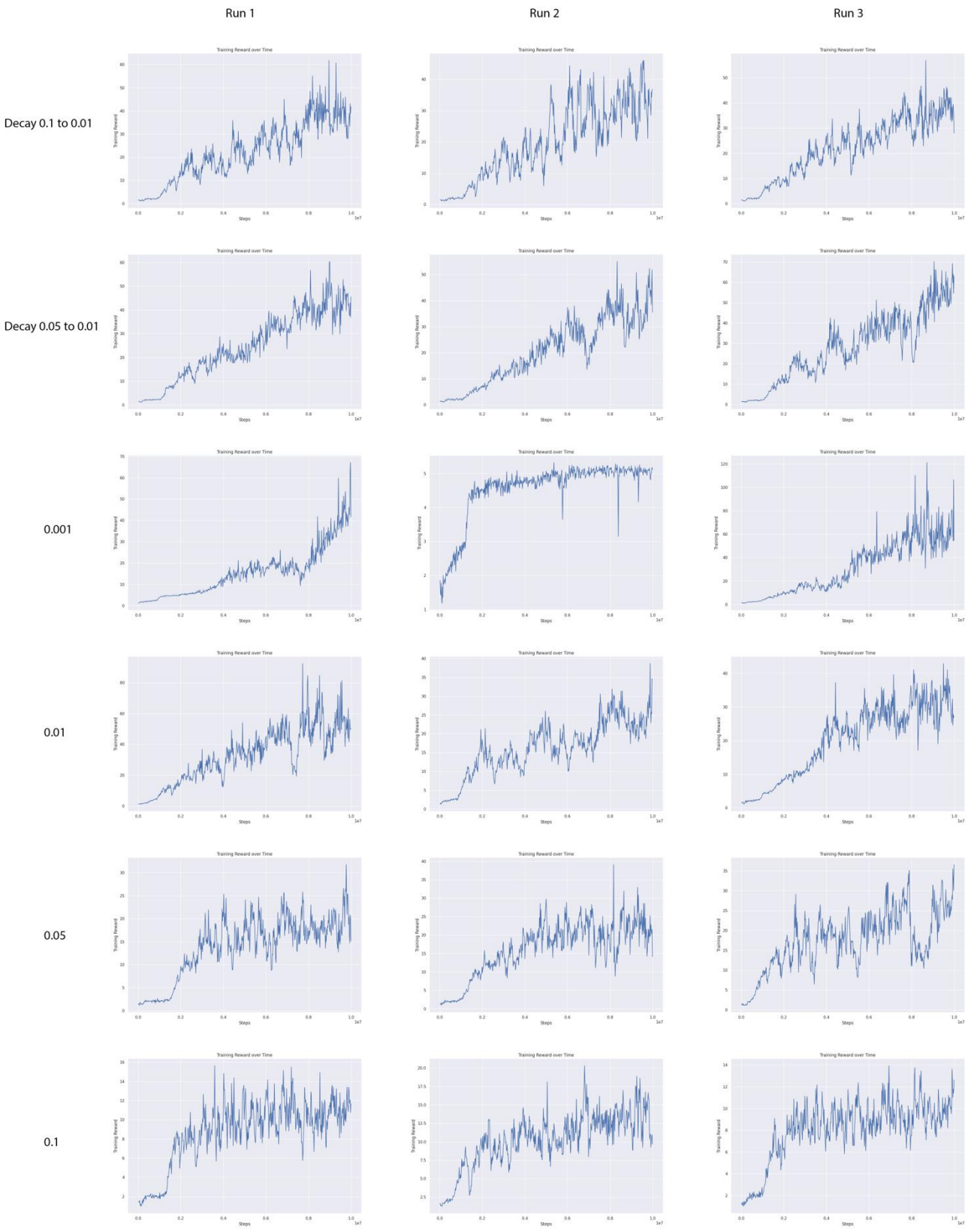

Figure 17. Breakout: Cumulative reward over time of each run for each entropy coefficient

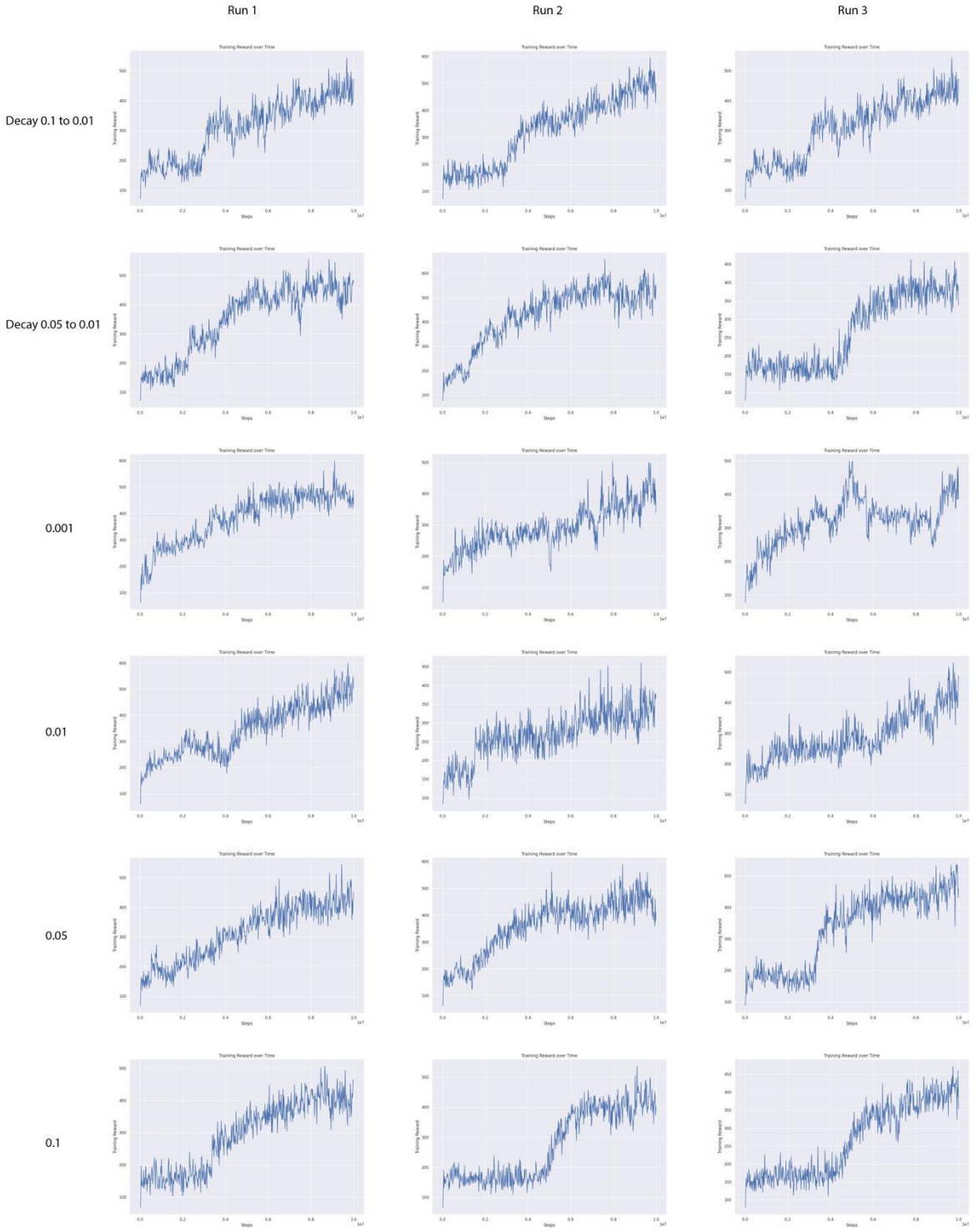

Figure 18. SpaceInvaders: Cumulative reward over time of each run for each entropy coefficient

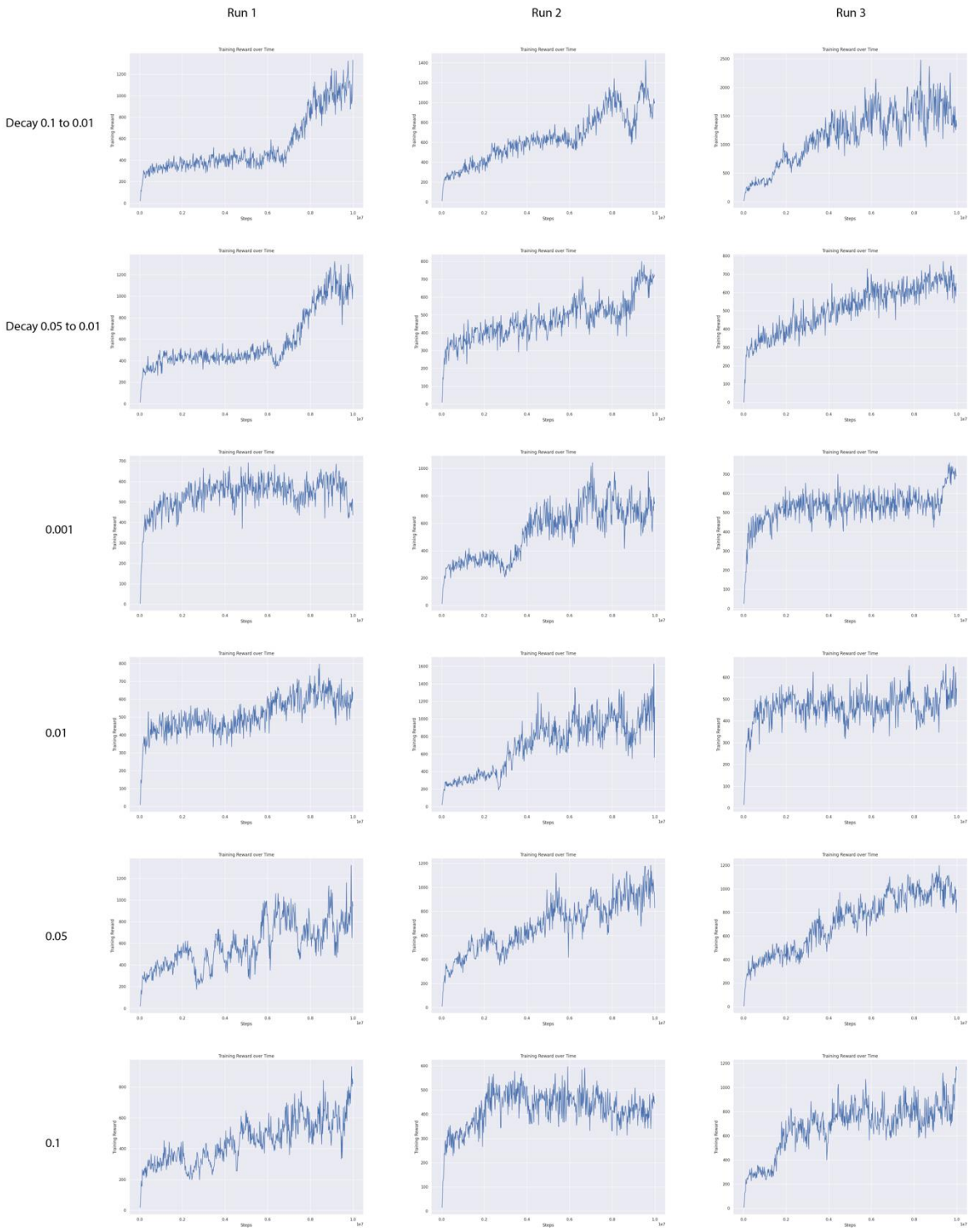

Figure 19. Seaquest: Cumulative reward over time of each run for each entropy coefficient

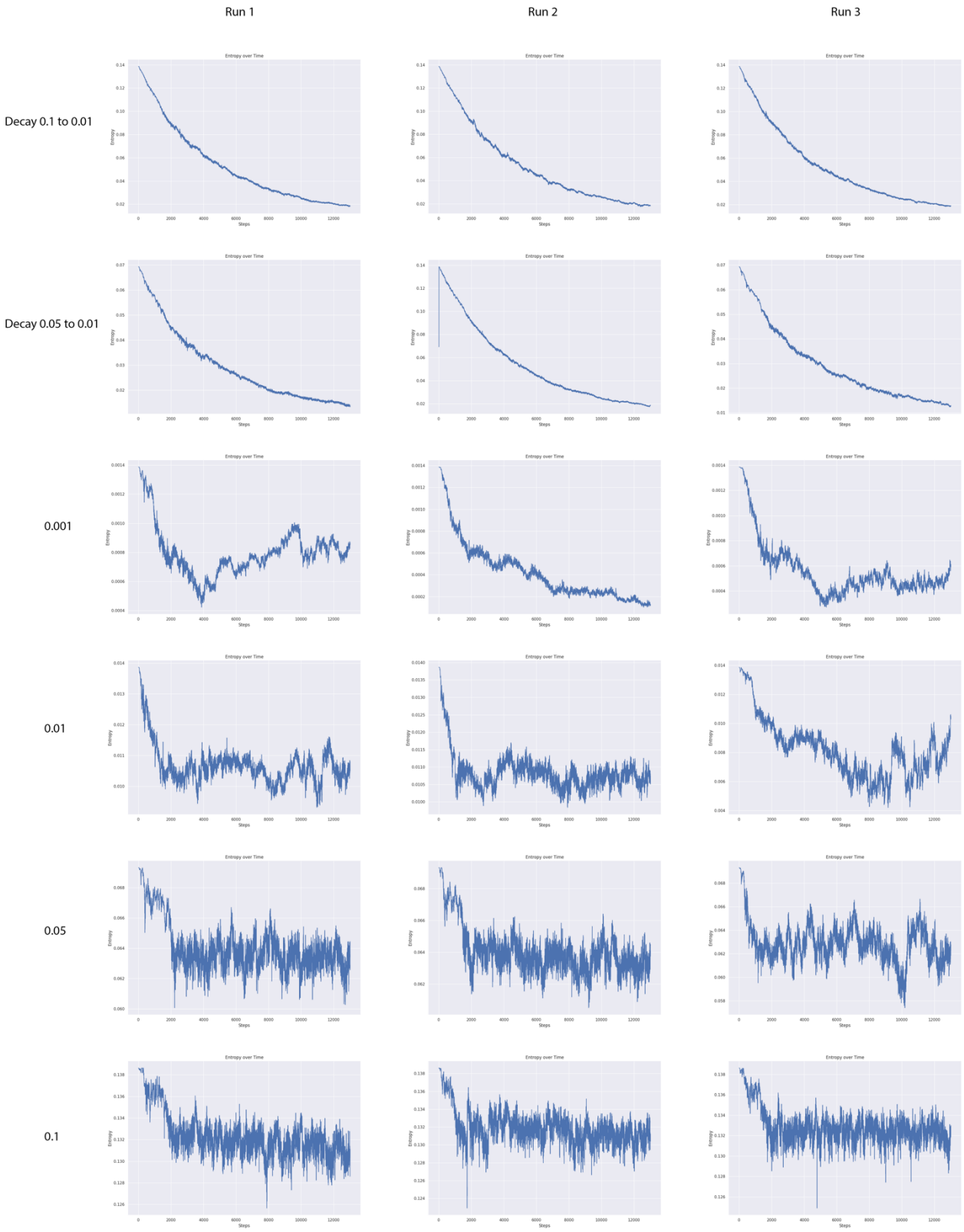

Figure 20. Breakout: Entropy of policy action distribution

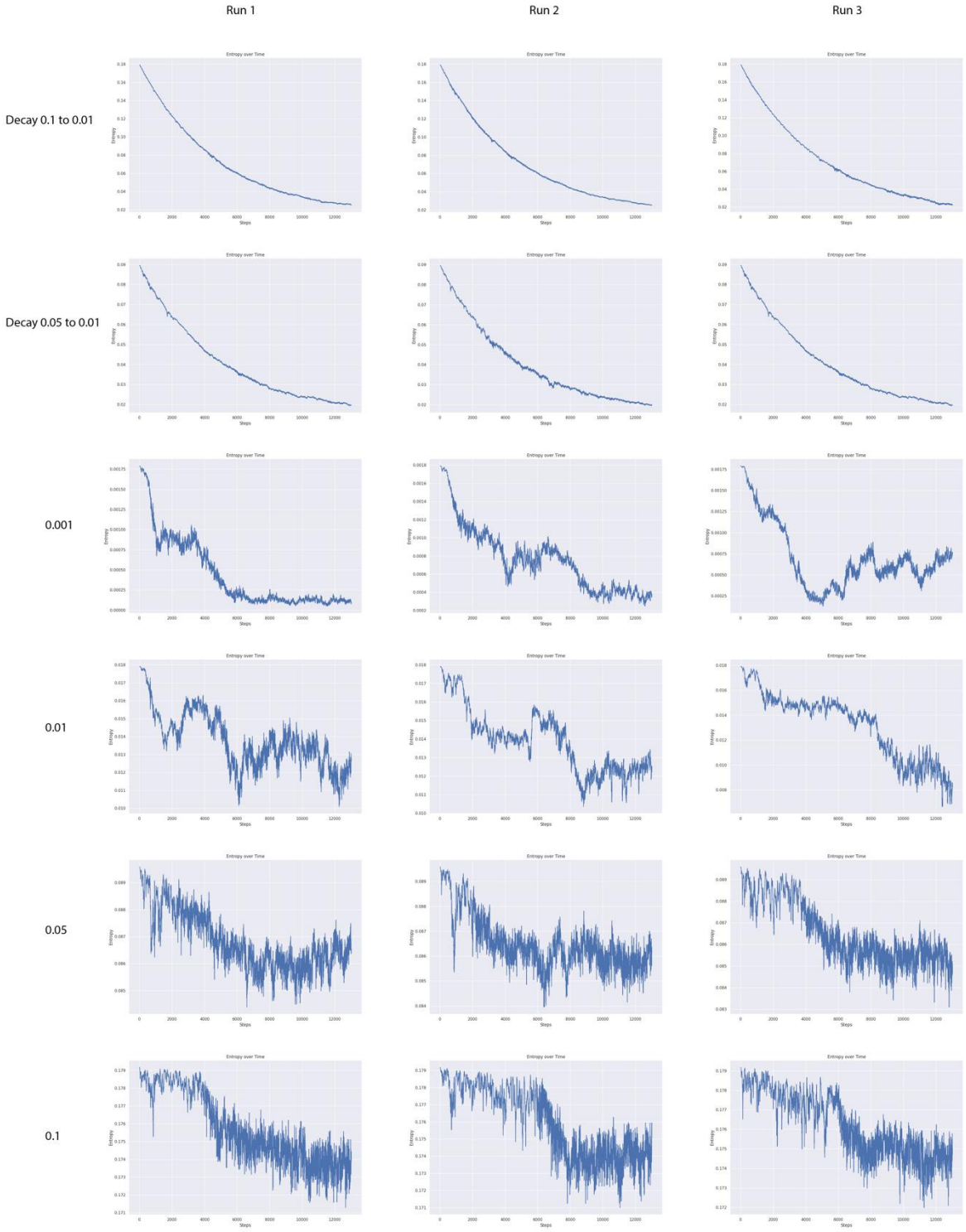

Figure 21. SpaceInvaders: Entropy of policy action distribution

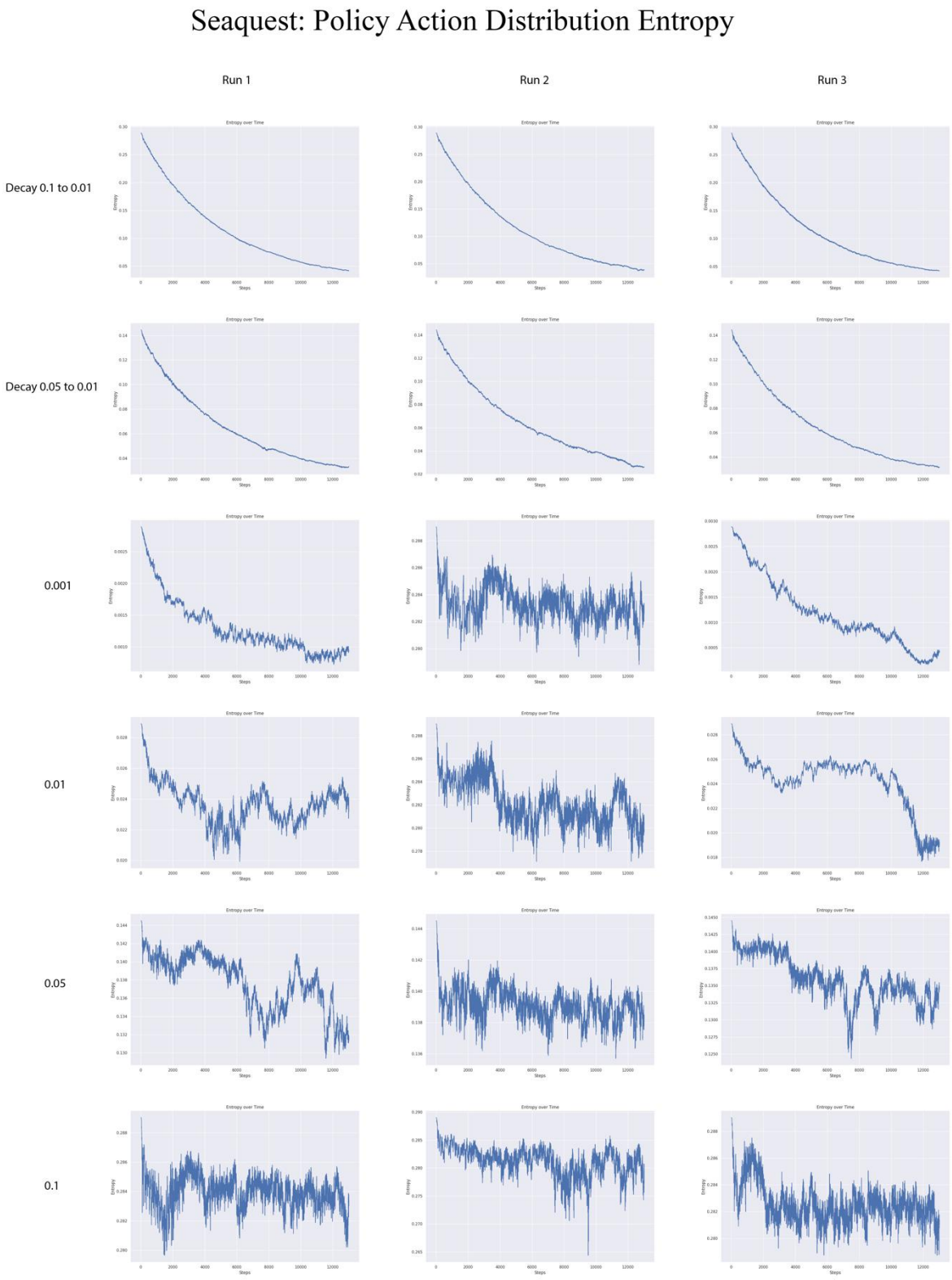

Figure 22. Seaquest: Entropy of policy action distribution

4.3 Attention Maps

4.3.1 Discussion on the Increased Interpretability Facilitated by Attention Maps

The use an attention network in the RL agent was heavily motivated by the notion of interpretability which can be achieved with the integration of attention map visualisation logic. Figures 26, 28 and 30 show a snapshot of the attention maps for the untrained IMPALA Attention Model in the Breakout, SpaceInvaders and Seaquest environments respectively. For comparison, Figures 27, 29 and 31 portray a snapshot of the attention maps for the same IMPALA attention agent after it has been trained on the three Atari environments for 10 million frames each. These snapshots provide qualitative information on what the agent is attending to, however a more appropriate way to view these maps is through video, which is an undeniably better format to draw qualitative observations from when working with spatiotemporal sequences of data. The following observations are derived from the video which can be viewed from the following link: https://t.ly/Ml42A

When examining the untrained SpaceInvaders agent, it is apparent that the agent is not focusing at all on its own position in the environment and instead is sporadically focusing on different sections of the enemy ship area. These points of focus seem to be random, as the areas it is focusing on do not have an impact on the immediate threat of getting destroyed. It is difficult to draw too many qualitative observations from the untrained agent other than to observe where the agent should perhaps be paying attention to, to achieve the goal of surviving in the environment and accumulating reward. As seen in the video and in Figure 28 below, attention head 4 is not active at all in this case and does not contribute. Alternatively, the trained SpaceInvaders model in the video is focusing on its own position in the environment and can be observed in attention head 1, 2 and 3, with head 4 seeming to attend to the area where it is considering navigating towards. We can also observe later in the video that head 1 and 2 seem to be attending to the immediate threat while 3 is focusing on other areas where the enemy space ships are present and approaching. Scans of the environment seem to be occurring sporadically in head 1 and head 4 as well, which may be a mechanism that the agent employs to scan for other regions of interest. There is some overlap in what each head is paying attention to – which can bring up the question: are 4 heads actually necessary in an environment of this complexity?

Examining the untrained Seaquest agent, it is apparent that the agent already is focusing on its own position in the environment, visible in attention head 2 and 3. This was surprising as this was something that was believed to be a learned behaviour, but the transition of pixel values could also be a contributor to the attention heads focus. Head 1 seems to be attending to scans of the environment, but again it is difficult to draw many qualitative observations from the untrained model. They are mainly used as a comparison for the trained models to see what has changed. After the model has been trained, we can observe that attention head 1 is attending to the agent's position in the environment almost exclusively. Attention heads 2 and 3 are focusing on almost identical regions at each timestep, and head 4 is not adding too much value as it is attending to regions of the environment which are not important.

Examining the untrained Breakout agent, the first clear observation is that attention head 4 is not active at all. It can also be observed that heads 1 and 3 are focusing on regions of the environment which are currently inaccessible and there is no value in attending to them. There are brief sequences of timesteps where we can observe head 2 and 3 focusing on the ball as well as its own position in the environment. After training it can be observed that much of the attention of heads 2, 3 and 4 are now focused on the position of the ball as well as sharing responsibility of attending to the agents position. We can also see that attention head 1 is initially focused on an area where it already has made a tunnel in the blocks, but as the environment switches to a different trajectory its focus changes to be on an unimportant region of the environment, the game score. (Mott *et al.*, 2019) observed that given enough time, the agent uses attention to set up "tripwires" to create an alert when an object of interest passes it, indicating to the agent that it needs to perform some action. While in this scenario attention head 1, and occasionally the other attention heads, seem to land their focus on the game score, which could not indicate to the agent that an action is needed, perhaps it could be connecting the score to reward, and may be following a similar path of deduction as the "tripwire" behaviour described. Mott et al trained their agent for 1 billion frames compared to these experiments of 10 million frames, so attention may become more focused or specialised when given enough time.

Another observation which is present generally across all environments is how active each attention head is over time. Generally, we can see from Figures 23, 24, and 25 that the activity of each attention head is varies greatly at the start of training, but as training progresses it generally can be observed that all heads become more or less equally active as their entropy

starts to stabilise. Some notable exceptions to this observation are for the static entropy coefficients of 0.1 in Seaquest, present in Figure 25, and 0.1 in SpaceInvaders, visible in Figure 24. This could be due to 0.1 being too high of an entropy coefficient, resulting in poor convergence.

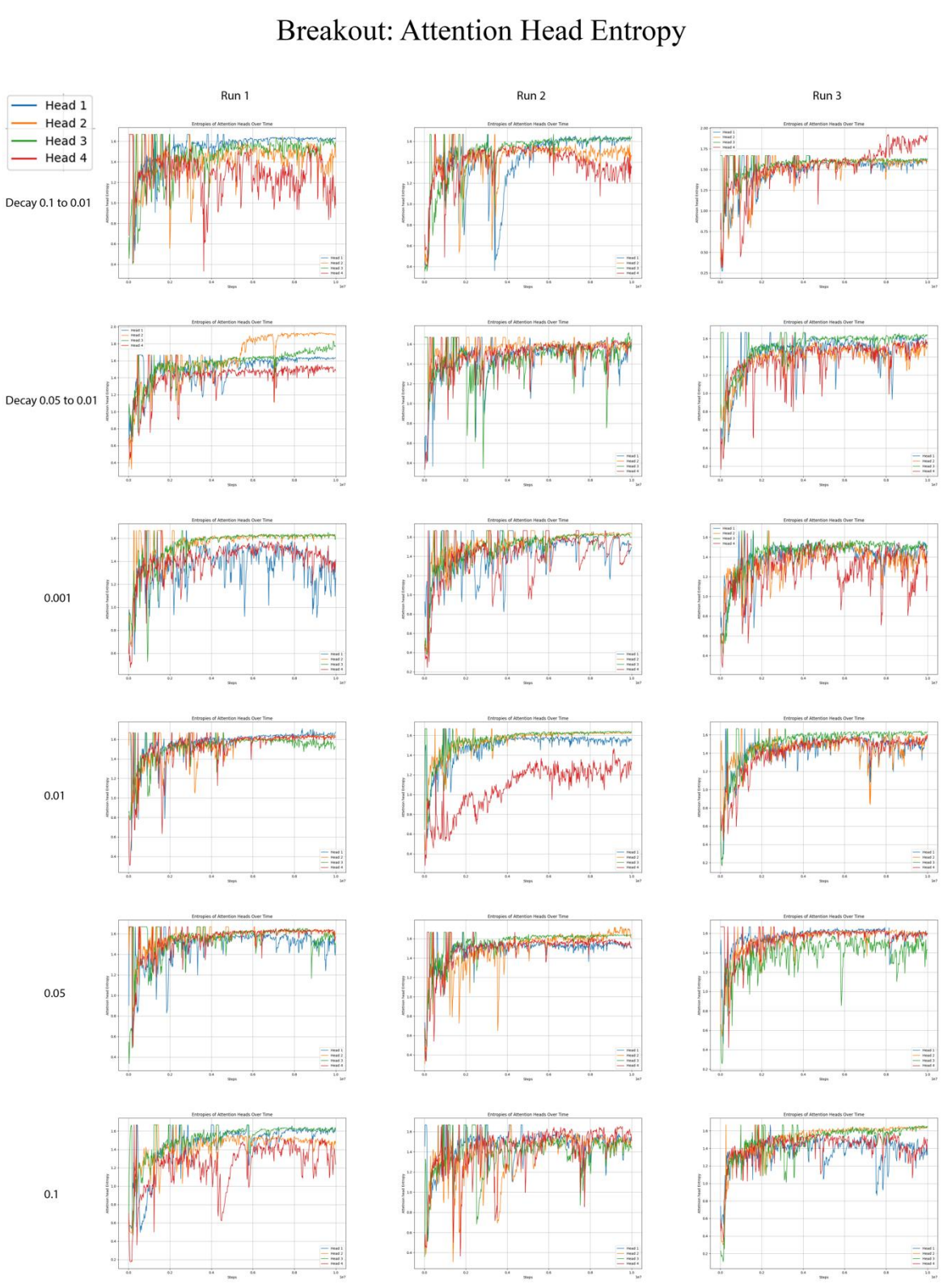

Figure 23. Breakout: Attention head entropy

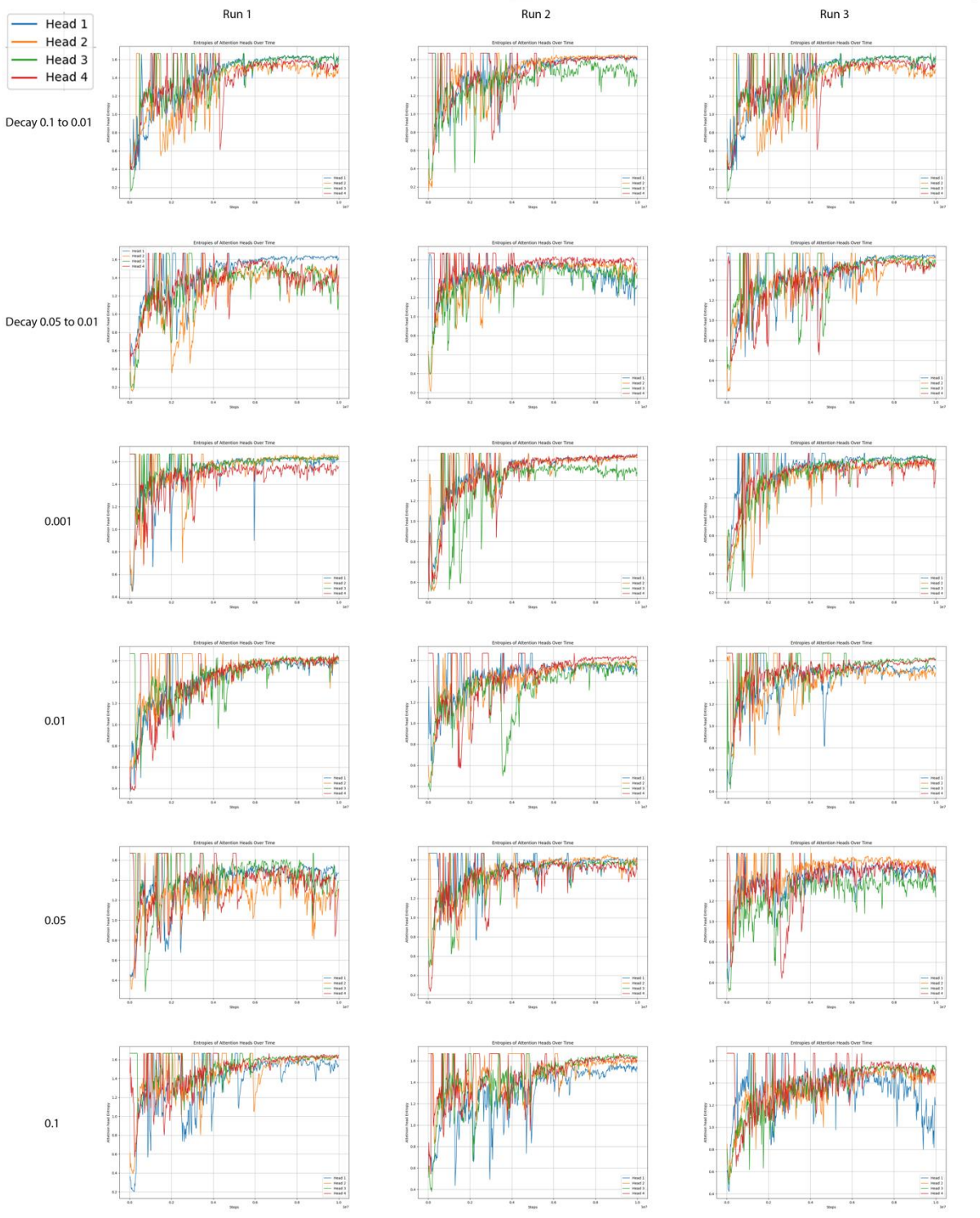

Figure 24. SpaceInvaders: Attention head entropy

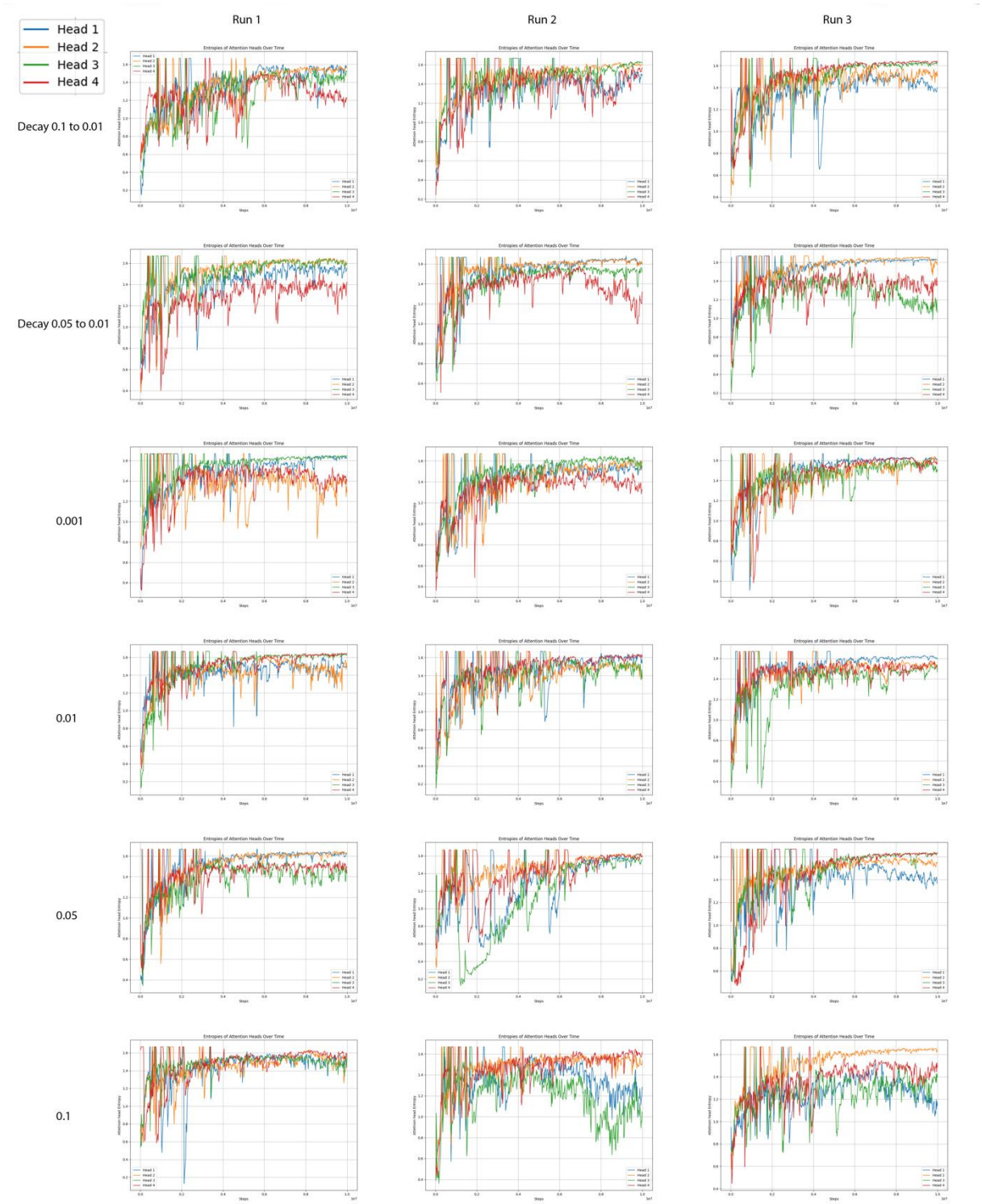

Figure 25. Seaquest: Attention head entropy

Link to video which makes better use of the attention map visualisations: https://t.ly/Ml42A

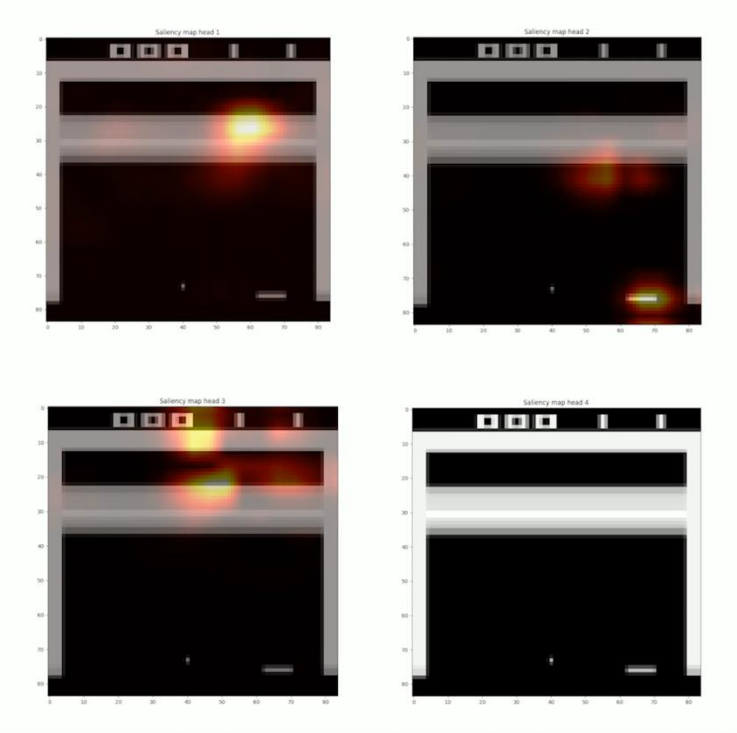

Figure 26. Breakout: Untrained Model Attention Maps

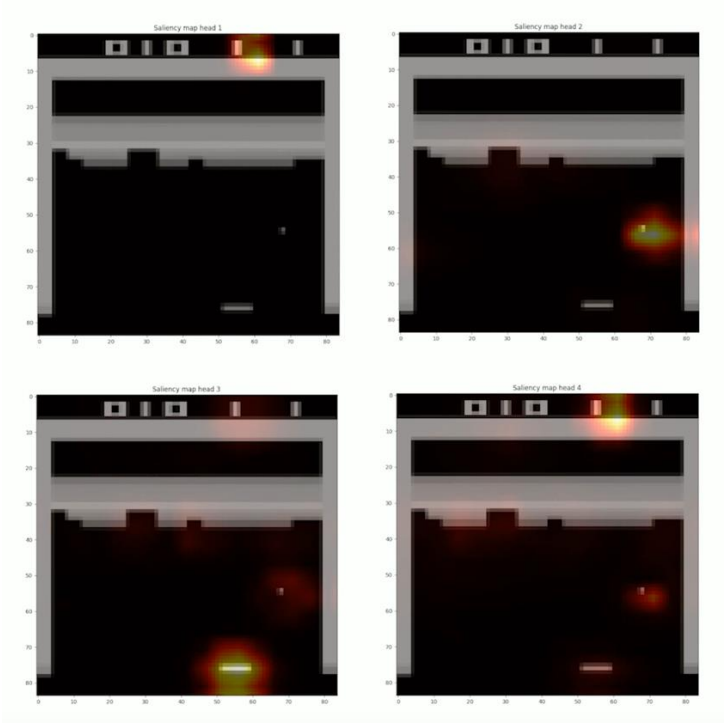

Figure 27. Breakout: Trained Model Attention Maps

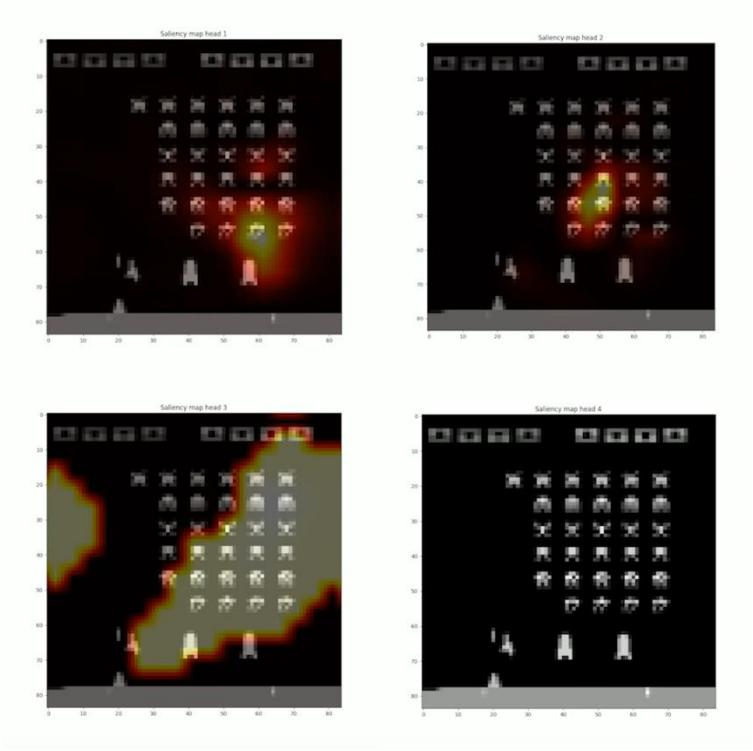

Figure 28. SpaceInvaders: Untrained Model Attention Maps

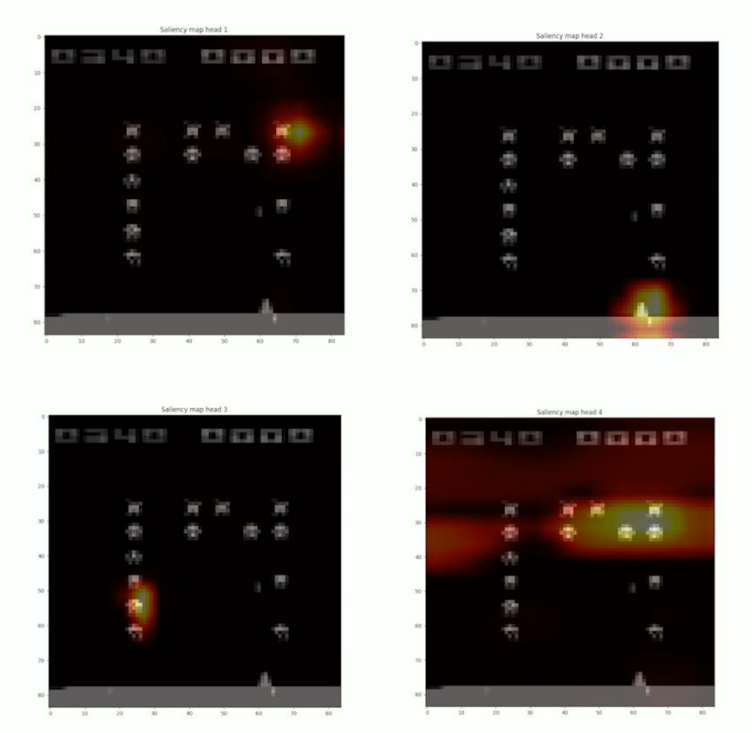

Figure 29. SpaceInvaders: Trained Model Attention Maps

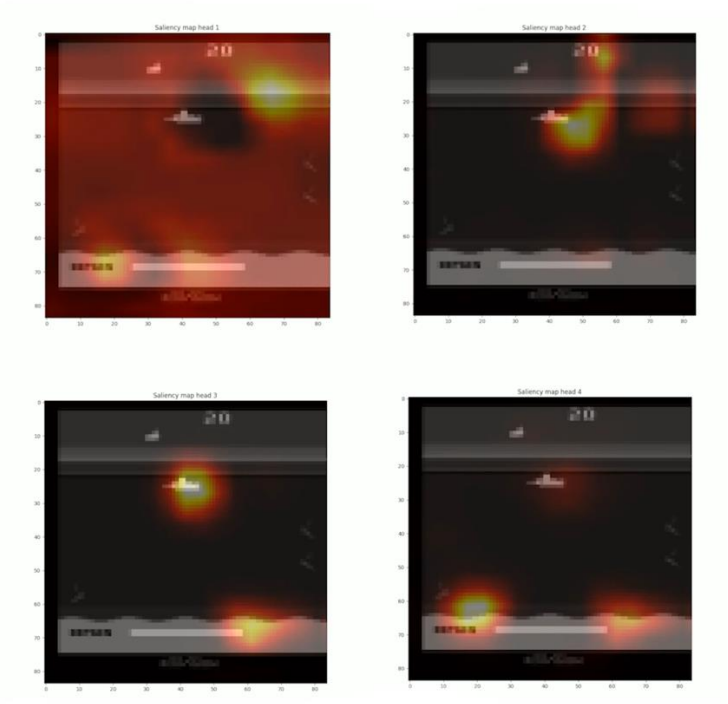

Figure 30. Seaquest: Untrained Model Attention Maps

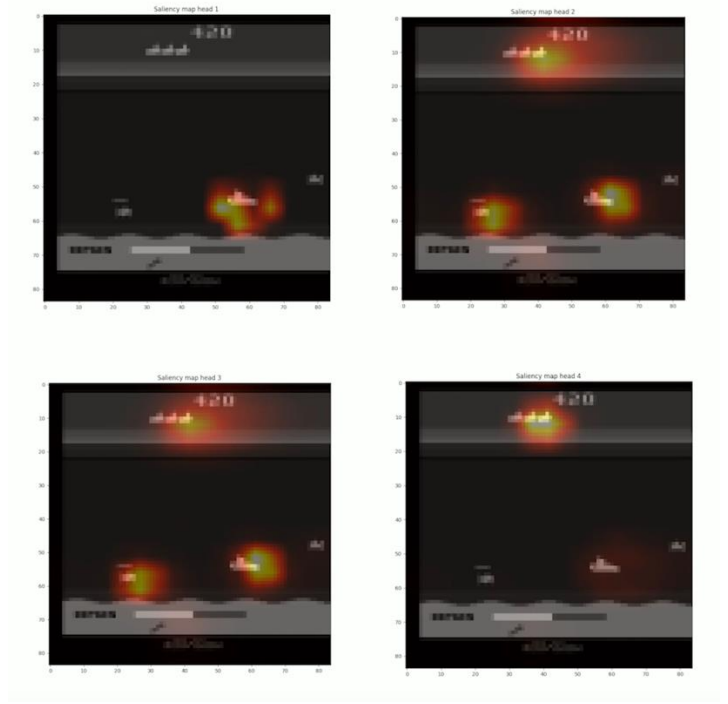

Figure 31. Seaquest: Trained Model Attention Maps

5

Chapter 5: Conclusion

In this dissertation the objective was to examine the intersection between attention mechanisms in RL and entropy regularisation, particularly focusing on how entropy affects the performance of an attention augmented agent attempting to learn three Atari environments of varying complexity. This research explored three areas: the influence of entropy regularisation on an attention augmented agent, the potential of a generalisable entropy coefficient that results in improved performance across all environments, and also the impact of the inclusion of an attention network within the RL model to determine how it affects performance and interpretability.

The experiments conducted show that entropy regularisation is an important mechanism to improve an agent's ability to explore. The results of the experiments indicate that the decaying entropy coefficients generally led to improved performance across all environments when compared to the static entropy coefficients. It was also observed that the overall stability of the agent was improved when tested over multiple runs, compared to the agents which were configured with static entropy coefficients. This research also investigated whether a single entropy coefficient could be applied across various environments to consistently improve performance. The findings suggest that, while the commonly used static entropy coefficient of 0.01 does result in decent performance, decaying entropy coefficients, specifically a decay from 0.05 – 0.01, often outperform the static coefficients. This data suggests that while a static entropy coefficient of 0.01 can be effective, more dynamic control of the entropy coefficient could be beneficial in producing more optimal agents.

The inclusion of the attention network within the RL model was shown to significantly increase the qualitative interpretability of the models' decisions by providing a means of

deciphering what the agent is attending to as it learns the environment. As one of the main criticisms of deep learning is their "black box" nature, through the use of attention maps to demystify the decisions being made by an agent, it enables both experts and non-experts to get a high-level insight into what the agent is learning. While by no means perfect, the use of attention maps may be considered a step in the right direction towards creating more interpretable and understandable models. This, in turn, can help build trust in deep learning, especially in areas where it can have significant consequences, i.e., healthcare. Unfortunately, with the increased interpretability also came degraded performance due to the increased complexity and depth of the network. This research showed that a trade-off emerged between interpretability and performance following the introduction of the attention mechanism. This trade-off is a problem for real world applications such as in healthcare, where interpretability is necessary but performance cannot be sacrificed just to have a more interpretable model. There are potential ways to resolve these performance issues, and the findings from the literature suggest that attention models can achieve competitive results compared with state-of-the-art models (Mott *et al.*, 2019), although this was not observed in these experiments.

For future work, the number of environments tested could be increased to see if a generalisable entropy coefficient, such as the decay from 0.05 – 0.01, continues to show promising results. Complexity of the action space and environment complexity could also be increased to examine whether decaying entropies continue to outperform static entropies when the dimensional complexity increases. The performance degradation of the attention augmented RL model caused by increased depth and complexity of the network is also something to be explored, potentially by incorporating techniques such as identity mapping from the D2RL architecture (Sinha *et al.*, 2020). Furthermore, increasing the number of repeated runs for an agent in a given environment would provide better aggregated performance information. Alternative architectures for the model could also be explored to see if there is a more optimal model configuration which facilitates increased performance.

References

Ahmed, Z., Roux, N.L., Norouzi, M. and Schuurmans, D. (2019) 'Understanding the impact of entropy on policy optimization'. arXiv. Available at: http://arxiv.org/abs/1811.11214.

Auer, P. (2002) 'Using Confidence Bounds for Exploitation-Exploration Trade-offs', *Journal of Machine Learning Research*, 3, pp. 397–422.

Bellman, R. (1984) *Dynamic programming*. Princeton, NJ: Princeton Univ. Pr.

Brockman, G., Cheung, V., Pettersson, L., Schneider, J., Schulman, J., Tang, J. and Zaremba, W. (2016) 'OpenAI Gym'. arXiv. Available at: http://arxiv.org/abs/1606.01540.

C. J. C. H. Watkins (1989) *Learning from Delayed Rewards*. Ph.D. Dissertation. Univ. of Cambridge.

Cesa-Bianchi, N., Gentile, C., Lugosi, G. and Neu, G. (2017) 'Boltzmann Exploration Done Right'. arXiv. Available at: http://arxiv.org/abs/1705.10257.

Cobbe, K., Hesse, C., Hilton, J. and Schulman, J. (2020) 'Leveraging Procedural Generation to Benchmark Reinforcement Learning'. arXiv. Available at: http://arxiv.org/abs/1912.01588.

Duarte, F., Lau, N., Pereira, A. and Reis, L. (2024) 'Dynamically Choosing the Number of Heads in Multi-Head Attention':, in *Proceedings of the 16th International Conference on Agents and Artificial Intelligence*. *16th International Conference on Agents and Artificial Intelligence*, Rome, Italy: SCITEPRESS - Science and Technology Publications, pp. 358–367. Available at: https://doi.org/10.5220/0012384500003636.

Eimer, T., Lindauer, M. and Raileanu, R. (2023) 'Hyperparameters in Reinforcement Learning and How To Tune Them'. arXiv. Available at: http://arxiv.org/abs/2306.01324.

Espeholt, L., Soyer, H., Munos, R., Simonyan, K., Mnih, V., Ward, T., Doron, Y., Firoiu, V., Harley, T., Dunning, I., Legg, S. and Kavukcuoglu, K. (2018) 'IMPALA: Scalable Distributed Deep-RL with Importance Weighted Actor-Learner Architectures'. arXiv. Available at: http://arxiv.org/abs/1802.01561.

Fei, H., Zhang, Y., Ren, Y. and Ji, D. (2022) 'Optimizing Attention for Sequence Modeling via Reinforcement Learning', *IEEE Transactions on Neural Networks and Learning Systems*, 33(8), pp. 3612–3621. Available at: https://doi.org/10.1109/TNNLS.2021.3053633.

Fortunato, M., Azar, M.G., Piot, B., Menick, J., Osband, I., Graves, A., Mnih, V., Munos, R., Hassabis, D., Pietquin, O., Blundell, C. and Legg, S. (2019) 'Noisy Networks for Exploration'. arXiv. Available at: http://arxiv.org/abs/1706.10295.

Greydanus, S., Koul, A., Dodge, J. and Fern, A. (2018) 'Visualizing and Understanding Atari Agents'. arXiv. Available at: http://arxiv.org/abs/1711.00138.

Haarnoja, T., Zhou, A., Abbeel, P. and Levine, S. (2018) 'Soft Actor-Critic: Off-Policy Maximum Entropy Deep Reinforcement Learning with a Stochastic Actor'. arXiv. Available at: http://arxiv.org/abs/1801.01290 (Accessed: 23 August 2024).

He, K., Zhang, X., Ren, S. and Sun, J. (2015) ‘Deep Residual Learning for Image Recognition’. arXiv. Available at: http://arxiv.org/abs/1512.03385.

Hochreiter, S. and Schmidhuber, J. (1997) ‘Long Short-Term Memory’, *Neural Computation*, 9(8), pp. 1735–1780. Available at: https://doi.org/10.1162/neco.1997.9.8.1735.

Ioffe, S. and Szegedy, C. (2015) ‘Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift’. arXiv. Available at: http://arxiv.org/abs/1502.03167.

Kingma, D.P. and Ba, J. (2017) ‘Adam: A Method for Stochastic Optimization’. arXiv. Available at: http://arxiv.org/abs/1412.6980.

Lim, H.-K., Kim, J.-B., Heo, J.-S. and Han, Y.-H. (2020) ‘Federated Reinforcement Learning for Training Control Policies on Multiple IoT Devices’, *Sensors*, 20(5), p. 1359. Available at: https://doi.org/10.3390/s20051359.

Liu, J., Gu, X. and Liu, S. (2020) ‘Policy Optimization Reinforcement Learning with Entropy Regularization’. arXiv. Available at: http://arxiv.org/abs/1912.01557.

Michel, P., Levy, O. and Neubig, G. (2019) ‘Are Sixteen Heads Really Better than One?’ arXiv. Available at: http://arxiv.org/abs/1905.10650.

Mnih, V., Badia, A.P., Mirza, M., Graves, A., Lillicrap, T.P., Harley, T., Silver, D. and Kavukcuoglu, K. (2016) ‘Asynchronous Methods for Deep Reinforcement Learning’. arXiv. Available at: http://arxiv.org/abs/1602.01783.

Mnih, V., Kavukcuoglu, K., Silver, D., Graves, A., Antonoglou, I., Wierstra, D. and Riedmiller, M. (2013) ‘Playing Atari with Deep Reinforcement Learning’. arXiv. Available at: http://arxiv.org/abs/1312.5602.

Mott, A., Zoran, D., Chrzanowski, M., Wierstra, D. and Rezende, D.J. (2019) ‘Towards Interpretable Reinforcement Learning Using Attention Augmented Agents’. arXiv. Available at: http://arxiv.org/abs/1906.02500.

Niv, Y., Daniel, R., Geana, A., Gershman, S.J., Leong, Y.C., Radulescu, A. and Wilson, R.C. (2015) ‘Reinforcement Learning in Multidimensional Environments Relies on Attention Mechanisms’, *The Journal of Neuroscience*, 35(21), pp. 8145–8157. Available at: https://doi.org/10.1523/JNEUROSCI.2978-14.2015.

‘OpenAI Baselines: ACKTR & A2C’ (2017) *OpenAI Baselines: ACKTR & A2C*. Available at: https://openai.com/index/openai-baselines-acktr-a2c/ (Accessed: 3 August 2024).

Phillips, P.J., Hahn, C.A., Fontana, P.C., Yates, A.N., Greene, K., Broniatowski, D.A. and Przybocki, M.A. (2021) *Four principles of explainable artificial intelligence*. NIST IR 8312. Gaithersburg, MD: National Institute of Standards and Technology (U.S.), p. NIST IR 8312. Available at: https://doi.org/10.6028/NIST.IR.8312.

Rudin, N., Hoeller, D., Reist, P. and Hutter, M. (2022) ‘Learning to Walk in Minutes Using Massively Parallel Deep Reinforcement Learning’. arXiv. Available at: http://arxiv.org/abs/2109.11978.

Russo, D.J., Van Roy, B., Kazerouni, A., Osband, I. and Wen, Z. (2018) ‘A Tutorial on Thompson Sampling’, *Foundations and Trends® in Machine Learning*, 11(1), pp. 1–96. Available at: https://doi.org/10.1561/2200000070.

Ryan, C., Collins, J. and Neill, M.O. (1998) ‘Grammatical evolution: Evolving programs for an arbitrary language’, in W. Banzhaf, R. Poli, M. Schoenauer, and T.C. Fogarty (eds) *Genetic Programming*. Berlin, Heidelberg: Springer Berlin Heidelberg (Lecture Notes in Computer Science), pp. 83–96. Available at: https://doi.org/10.1007/BFb0055930.

Schulman, J., Wolski, F., Dhariwal, P., Radford, A. and Klimov, O. (2017) ‘Proximal Policy Optimization Algorithms’. arXiv. Available at: http://arxiv.org/abs/1707.06347.

Shi, X., Chen, Z., Wang, H., Yeung, D.-Y., Wong, W. and Woo, W. (2015) ‘Convolutional LSTM Network: A Machine Learning Approach for Precipitation Nowcasting’. arXiv. Available at: http://arxiv.org/abs/1506.04214.

Silver, D., Schrittwieser, J., Simonyan, K., Antonoglou, I., Huang, A., Guez, A., Hubert, T., Baker, L., Lai, M., Bolton, A., Chen, Y., Lillicrap, T., Hui, F., Sifre, L., Van Den Driessche, G., Graepel, T. and Hassabis, D. (2017) ‘Mastering the game of Go without human knowledge’, *Nature*, 550(7676), pp. 354–359. Available at: https://doi.org/10.1038/nature24270.

Sinha, S., Bharadhwaj, H., Srinivas, A. and Garg, A. (2020) ‘D2RL: Deep Dense Architectures in Reinforcement Learning’. arXiv. Available at: http://arxiv.org/abs/2010.09163.

‘Stable Baselines Documentation’ (2023). Available at: https://stable-baselines.readthedocs.io/en/master/ (Accessed: 3 August 2024).

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, L. and Polosukhin, I. (2023) ‘Attention Is All You Need’. arXiv. Available at: http://arxiv.org/abs/1706.03762.

Williams, R.J. (2004) ‘Simple statistical gradient-following algorithms for connectionist reinforcement learning’, *Machine Learning*, 8, pp. 229–256.

Acknowledgments

Thank you to my supervisor Prof. Conor Ryan, Meghana Kshirsager and the BDS group for their support and guidance throughout this dissertation.
I also want to thank my partner, Kai, who has been incredibly supportive and encouraging throughout this challenging year.