Self-Attention for Visual Reinforcement Learning

Zachary Fernandes
Rensselaer Polytechnic Institute
fernaz@rpi.edu

Ethan Joseph
Cornell Tech
eaj73@cornell.edu

Dean Vogel
Rensselaer Polytechnic Institute
vogeld2@rpi.edu

Mei Si
Rensselaer Polytechnic Institute
sim@rpi.edu

***Abstract*—Reinforcement learning has been extensively applied to playing video games that involve visual observations, using convolutional neural networks (CNNs) to process the image input. Recent research has demonstrated that attention mechanisms in neural networks can be highly effective in various tasks, although their potential for visual reinforcement learning has not been fully explored.**

This study explores three approaches to incorporating attention into reinforcement learning. The first approach is to add self-attention to the later dense layers of the CNN. The second approach involves using a convolutional self-attention module within the convolutional layers of the network. The third approach incorporates self-attention with a large kernel convolution within the convolutional layers.

To assess the efficacy of the proposed approaches, we conducted a comparative analysis of their performance against a baseline CNN feature extractor for a Proximal Policy Optimization (PPO) agent across multiple image-based Atari environments. Our findings suggest that attention mechanisms can be advantageous in scenarios where the PPO agent needs to adapt to minor environmental fluctuations. However, in games with fixed setups, baseline PPO tends to outperform the variations with the attention mechanism.

***Index Terms*—reinforcement learning, attention, computer vision**

I. Introduction

Deep reinforcement learning (RL) has emerged as a popular approach for Game AI and has exhibited remarkable proficiency in several complex games, such as Dota 2 [1], StarCraft II [2], Stratego [3], Gran Turismo [4], and Chess [5]. As numerous games utilize visual inputs for the RL agent, it is essential to comprehend and enhance the visual processing aspects of RL algorithms to augment their performance.

Deep reinforcement learning algorithms commonly employ visual processing networks that are composed of stacked convolutional layers, frequently adapted from the ”Nature CNN” architecture initially proposed in the Deep Q-Network (DQN) work [6]. Subsequently, the convolutional layers are followed by dense layers, which enable the agent to acquire its policy or state values.

While initially developed for natural language processing tasks, attention mechanisms in neural networks have demonstrated significant efficacy across various visual tasks [7]. In particular, self-attention is a process that enables a model to zero in on particular portions of the input data. Self-attention associates keys, values, and queries with each input sequence element (e.g., words in a sentence and pixels in an image) [8]. Keys, values, and queries determine which elements in the sequence are most relevant to the current element. Each query is compared to all keys to determine its weights during the attention computation. The query's attention output is calculated by weighting and summing the values. Self-attention has been shown to improve performance in natural language processing, image recognition, and other applications. Self-attention is useful for natural language processing and computer vision because it captures long-range connections and relationships between input sequence elements.

Self-attention has been recently explored in the context of reinforcement learning to improve the visual processing components of RL algorithms [9]–[11]. However, the results from these studies are not consistent. Some have reported significant performance improvements by incorporating attention mechanisms into RL algorithms [10]. However, other studies have reported mixed or negative results when using attention mechanisms in RL [9]. Therefore, the effectiveness of attention mechanisms in RL algorithms remains an open research question, and further investigation is needed to determine the conditions under which they are most effective. In a recent work – Decision Transformer – incorporates a transformer for visual RL. However, it is not for vision processing but for modeling the Markov Decision Process (MDP) in a purely sequential manner [12].

This study systematically evaluates self-attention mechanisms' efficacy in vision-based reinforcement learning tasks. To achieve this goal, we use a CNN-based Proximal Policy Optimization (PPO) [13] agent as our baseline and test three variants of the agent with attention on a subset of the Atari games. We chose PPO as our baseline instead of DQN due to its faster training speed. PPO was also used as the baseline in a previous study [10], which demonstrated improvements by incorporating self-attention in reinforcement learning.

The first variant we evaluate is the incorporation of convolutional self-attention in reinforcement learning [10], [11]. Following the naming convention in [10], we call this condition

Self Attention Network (SAN). The second variant we explore is Large Kernel Attention (LKA) [14], which combines spatial attention with channel attention and has achieved near-state-of-the-art performance on computer vision benchmarks. To the best of our knowledge, LKA has not been previously applied in the context of reinforcement learning. Finally, we investigate the effects of adding attention to the dense layers of the PPO instead of the convolutional layers. As the PPO algorithm uses separate policy and value heads, we evaluate adding attention to both the shared and independent dense layers (see Fig. 2 for details). We refer to these conditions as Dense Self-Attention (DSA).

Our experiments show that incorporating self-attention mechanisms into visual RL games can be advantageous, but not always. We evaluated seven Atari game environments, including MsPacman, Breakout, Demon Attack, Asterix, Freeway, Frostbite, and Krull, performing $40M$ training steps for each environment. This number of steps is recommended by [10] as a reasonable amount for finishing training an Atari game. When training the agents using a fixed initial game setup, i.e., using the same random seed, in most cases, PPO outperformed other variants with attention. Even when PPO did not perform better, it often scored similarly to the highest. However, we observed that variations with attention had an advantage when we varied the games' initial states during the training process, suggesting that self-attention may help RL agents adapt to variations in the games better. Additionally, we noticed that adding attention to dense layers generally hurt agents' performances, except when the attention layer was the last layer before the network output.

This work contributes in four main ways:

- We validated the use of self-attention networks (SAN) for reinforcement learning, as proposed by [10], with some differences observed in results.
- We explored the use of Large Kernel Attention (LKA) [14] for RL.
- We investigated the impact of adding self-attention to dense layers of PPO.
- We compared the training results of RL algorithms under different seed settings, with fixed or varying initial states during training.

The findings provide insights into the optimal application of self-attention in RL and offer directions for future research.

II. Related Work

Attention is a concept borrowed from cognitive science that refers to the phenomenon that biological systems focus on the most relevant information. It has been widely applied in natural language processing [15] and computer vision [7]. In computer vision, the attention mechanism allows the neural networks to focus on the most relevant part of the image and discard other information. Based on where attention is applied, these works can be categorized into four types:

- Channel attention: attention is applied to visual channels.
- Spatial attention: attention is applied to spatial regions.
- Temporal attention: attention is applied to temporal information. This is typically used when processing videos.
- Branch attention: attention is applied to the branches of the neural network. This is used in combination with a multi-branch structure.

The above methods can be combined; this allows the neural network to pay attention to multiple aspects of the calculation, such as channel & spatial attention.

There have been several attempts to combine visual attention with reinforcement learning (RL) techniques. In one study, [11] added visual attention to RL algorithms for playing Catch, a toy game, and Atari games. They added an attention layer at the end of the convolutional layers, which can be considered as spatial attention. Their results were mixed, with the RL algorithms with attention performing better in some conditions, particularly when the visual signals were noisy, but comparable to models without attention in other cases.

Another study by [10] employed convolutional self-attention (SAN) for RL, which also used spatial attention. They added an attention layer between the first and second convolutional layers and tested their method on Atari games. Their results demonstrated the advantage of incorporating attention in about 60% of the game environments they tested.

Recently, some researchers have focused on modeling RL as a sequence prediction system and using attention-based methods such as transformers for this purpose [12], [16], [17]. However, these problem classes have different formulations, and this work focuses on RL as a learned reward signal rather than a sequence prediction task.

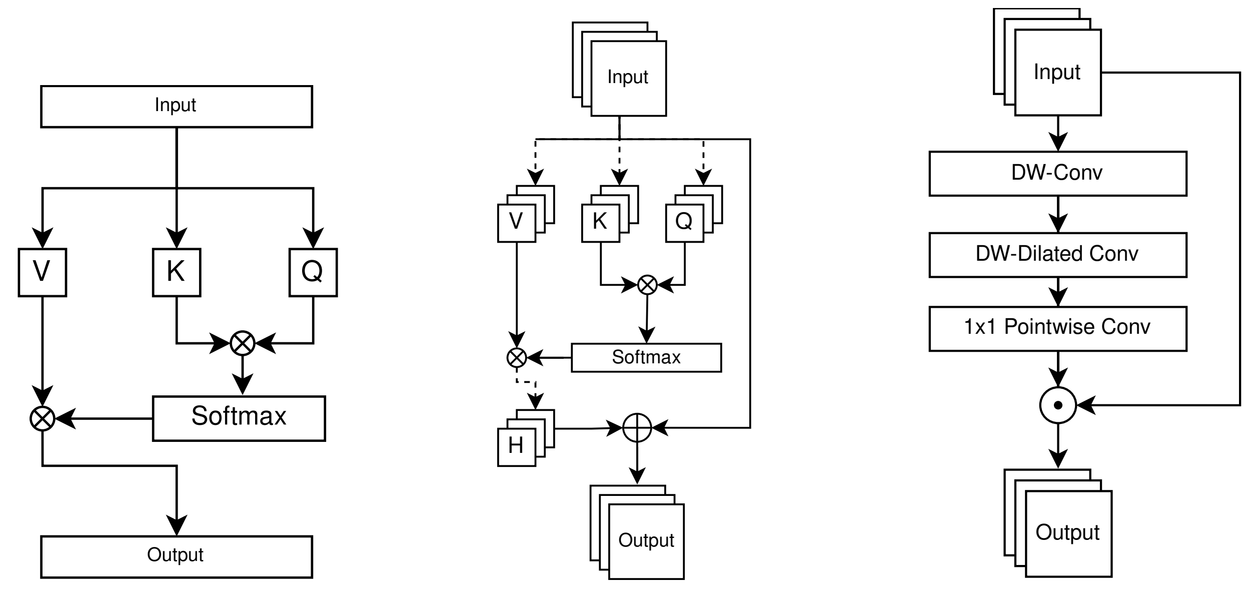
III. Methodology

We aim to explore the potential of attention mechanisms for improving the performance of reinforcement learning agents in environments with visual observation spaces. While there have been attempts at incorporating attention mechanisms into RL, attention does not seem to significantly boost the performance of RL models, at least not to a level comparable with attention's performance gains in visual tasks alone [7].

A. Attention Mechanisms

Many attention mechanisms have been introduced in recent years, ranging from the original attention implementation [18], to the multi-head self-attention that enables transformer architectures [8], to a convolutional self-attention initially designed for generative adversarial networks [19], to novel large-kernel attention designed for visual tasks [14]. Despite a plethora of research utilizing attention in other tasks, there is no consensus on which attention mechanism works best for reinforcement learning and a limited number of explorations. In this work, we compare the following attention mechanisms on RL: Dense Self-Attention (DSA), Convolutional Self-Attention (SAN), and Large Kernel Attention (LKA).

1) Dense Self-Attention: What we refer to by Dense Self-Attention (DSA) is the scaled dot-product attention proposed by [8]. This attention mechanism was initially designed for language-related tasks and works on 1D sequences of data.



(a) Dense Self-Attention; $\otimes$ denotes the dot-product.

(b) Convolutional Self-Attention; dashed arrows represent a 1x1 convolution operation, $\otimes$ denotes matrix-multiplication, and $\oplus$ denotes element-wise addition.

(c) Large Kernel Attention; consisting of a depth-wise convolution (DW-Conv), depth-wise dilation convolution (DW-Dilated Conv), and a pointwise convolution; $\odot$ denotes element-wise multiplication.

Fig. 1: Structure of different attention mechanisms

However, it has also seen success in visual tasks when images are transformed into linear sequences [20]. To compute dense self-attention, we transform the input into Key, Query, and Value vectors using dense fully connected layers, compute the dot product between the Key and Query vectors, apply a softmax function on the result, and finally compute the dot product between the result and the Value vector to receive our output. More formally, the features from the previous layer $\boldsymbol{x}$ are first transformed into 3 vectors $\boldsymbol{K}, \boldsymbol{Q}, \boldsymbol{V}$, where $\boldsymbol{K}(x) = \boldsymbol{W_k}x$, $\boldsymbol{Q}(x) = \boldsymbol{W_q}x$, and $\boldsymbol{V}(x) = \boldsymbol{W_v}x$. $\boldsymbol{W_k} \in \mathbb{R}^{d\times n}, \boldsymbol{W_q} \in \mathbb{R}^{d\times n}, \boldsymbol{W_v} \in \mathbb{R}^{d\times n}$ are learned weight matrices implemented as dense fully connected layers, d is a hyperparameter (embedding dimension), and n is the size of the input.

The output of the dense self-attention module $\boldsymbol{y}$ is given by

$$\boldsymbol{y} = \text{softmax}(\frac{\boldsymbol{Q}\boldsymbol{K}^T}{\sqrt{d}})\boldsymbol{V} \tag{1}$$

See Fig. 1a for structure and comparison.

2) Convolutional Self-Attention: Convolutional self-attention (SAN) is designed specifically to capture long-range dependencies and model relationships between widely separated spatial regions in images. Our implementation of SAN was first introduced by [19] for generative adversarial networks. The image features from the previous hidden layer are again transformed into Key, Query, and Value feature spaces, this time using 1x1 convolution operations, decreasing the number of filters eightfold. We compute the matrix-product between Key and Query features and softmax it, then compute the matrix-product between the result and the Value feature space. We transform this output into feature space H using another 1x1 convolution operation, and finally perform element-wise addition with the input to yield the output. More formally, the image features from the previous layer with height and width of size N and C channels ($x \in \mathbb{R}^{C\times N}$) are first transformed into three feature spaces $\boldsymbol{K}, \boldsymbol{Q}, \boldsymbol{V}$, where $\boldsymbol{K}(x) = \boldsymbol{W_k}x$, $\boldsymbol{Q}(x) = \boldsymbol{W_q}x$, $\boldsymbol{V}(x) = \boldsymbol{W_v}x$, and $\boldsymbol{H}(x) = \boldsymbol{W_h}x$. The attention $\boldsymbol{a} = (\boldsymbol{a_1}, \boldsymbol{a_2}, ..., \boldsymbol{a_j}, ..., \boldsymbol{a_N}) \in \mathbb{R}^{C\times N}$ is given by

$$\boldsymbol{a_j} = \boldsymbol{H}\left(\sum_{i=1}^{N} \text{softmax}(\boldsymbol{Q}(\boldsymbol{x_j})\,\boldsymbol{K}(\boldsymbol{x_i})^T)\boldsymbol{V}(\boldsymbol{x_i})\right), \tag{2}$$

In the previous equation, $\boldsymbol{W_k} \in \mathbb{R}^{C/8\times C}, \boldsymbol{W_q} \in \mathbb{R}^{C/8\times C}, \boldsymbol{W_v} \in \mathbb{R}^{C/8\times C}, \boldsymbol{W_h} \in \mathbb{R}^{C\times C/8}$ are learned weight matrices implemented as 1x1 convolutions. Finally, the output y of the convolutional self-attention module is given by

$$\boldsymbol{y_i} = \gamma \boldsymbol{a_i} + \boldsymbol{x_i} \tag{3}$$

where γ is a learnable scalar and it is initialized as 0. See Fig. 1b for a diagram and comparison.

3) Large Kernel Attention: Large kernel attention (LKA) [14] is designed to model adaptability and long-range dependence, while being able to make use of local contextual information due to its large-kernel convolution operation. Additionally, it is designed to capture adaptability in the channel dimension, which is important for image-based reinforcement learning as greyscale frames are often stacked in the channel dimension (frame-stacking) in order to model time [21]. The large-kernel convolution operation can be decomposed into an attention mechanism via three parts: a depth-wise convolution, a depth-wise dilation convolution, and a pointwise convolution. The LKA module consists of an element-wise multiplication between the input and the results of a large-kernel convolution on that input. See Fig. 1c for structure and comparison.

B. Proximal Policy Optimization

Proximal Policy Optimization (PPO) [13] is an Actor-Critic algorithm [22]. These algorithms learn both policy (actor) and value (critic) functions via deep neural networks. In PPO, the actor generates trajectories, from which the critic estimates the advantage via Generalized Advantage Estimation (GAE) [23]

$$A_{\pi_\theta}(s_t, a_t) = \sum_{i=0}^{\infty} (\gamma\lambda)^i (r_i + \gamma V_\phi(s_{i+1}) - V_\phi(s_i)) \tag{4}$$

where π_θ is the parameterized policy, V_ϕ is the parameterized critic, γ is the discount factor, and λ is the advantage estimation parameter. The policy objective maximizes

$$\begin{aligned} J(\theta) \approx \frac{1}{T}\sum_{t=0}^{T-1} min\bigg(&\frac{\pi_\theta(s_t|a_t)}{\pi_{\theta_i}(s_t|a_t)} A(s_t, a_t), \\ &\text{clamp}\left(A(s_t, a_t), 1-\epsilon, 1+\epsilon\right)\bigg) \end{aligned} \tag{5}$$

The actor then updates policy parameters θ for $\pi_\theta(a|s)$ in the direction of the gradient by some learning rate $\alpha : \theta = \theta_{i-1} + \alpha\nabla J(\theta_{i-1})$. This repeats until the policy parameters reach a maximum or the gradient becomes insignificant.

Due to the vectorized nature of PPO's interactions and the efficiency of the above clipped surrogate objective, it is one of the most common and successful model-free RL algorithms.

C. CNN Feature Extraction with Attention

Many implementations of actor-critic methods in environments with image-based observations utilize a shared CNN feature extractor base. This network takes a frame (or more commonly, a stack of frames) f_t and transforms it into a flattened vector of features l_t that is fed to separate networks to yield the policy and value components of the actor-critic framework (see Fig. 2 for a diagram of the feature extraction architecture used in our experiments). This method of decoupling feature extraction from policy learning by using a separate network for creating an internal representation of important features has the advantage of reducing the search space for policy and value networks [24].

Since the feature extraction network is integral for reducing state space, it is important that the network learns to effectively extract important features from the input images. Since self-attention has proven effective in modeling feature dependency and importance, it seems likely that there are advantages to be had in adapting self-attention for RL.[1]

[1]Value-based methods that employ a CNN (e.g., DQN) can likely utilize attention in a similar way but were not evaluated in this work.

D. Early vs. Late Attention

We also posit that the position of an attention module affects what features the network is able to attend to. An attention mechanism placed early in the CNN will be able to attend to individual pixels and regions within an image and allow later layers to focus on features within those regions, whereas a late attention mechanism will be able to attend to fully extracted features and effectively select which ones are most important for propagation to the policy and value networks. Further, placing attention layers late in the network and separately for the policy and value network has the potential to enable the learning of different attention maps for each network.

IV. Experiments

Our experiment setup is inspired by and kept consistent with [10]. This enables us to compare our results with the original SAN work. We trained multiple agents with varying attention mechanisms using PPO implemented with the Stable Baselines3 (SB3) library [25] for reproducibility. SB3 is a set of reliable implementations of reinforcement learning algorithms in PyTorch.

A. Agent Architecture Variations

See Fig. 2 for a full architecture diagram. Below we specify which attention mechanisms are used and where they are used for each variation.

1) Baseline PPO (PPO): Our baseline model is the default model used by SB3 for all actor-critic policies. It takes a 4-channel frame-stacked input and consists of 3 convolutional layers with kernels of sizes 8, 4, and 3 and output channels of 32, 64, and 64 respectively. The convolutional layers are followed by two fully connected layers with 512 and 64 output features respectively (no attention modules). The final output features are fed to separate policy and value networks consisting of a single fully connected layer that maps to the action space for the policy network and 1 value feature for the value network.

2) Convolutional Self-Attention (SAN): We use the same architecture as PPO and add a convolutional self-attention layer between the 1st and 2nd convolutional layers in the network (Convolutional Self-Attention in Module $A1$).

3) Late Convolutional Self-Attention (SAN_L): We use the same architecture as PPO and add a convolutional self-attention layer after the last convolutional layer in the network (Convolutional Self-Attention in Module $A2$).

4) Large Kernel Attention (LKA): We use the same architecture as PPO and add a large kernel attention layer between the 1st and 2nd convolutional layers in the network (Large Kernel Attention in Module $A1$).

5) Late Large Kernel Attention (LKA_L): We use the same architecture as PPO and add a large kernel attention layer after the last convolutional layer in the network (Large Kernel Attention in Module $A2$).

6) Shared Dense Self-Attention (DSA_{SH}): We use the same architecture as PPO and add a dense self-attention layer after the final fully connected layer in the network (Dense Self-Attention in Module $A3$).

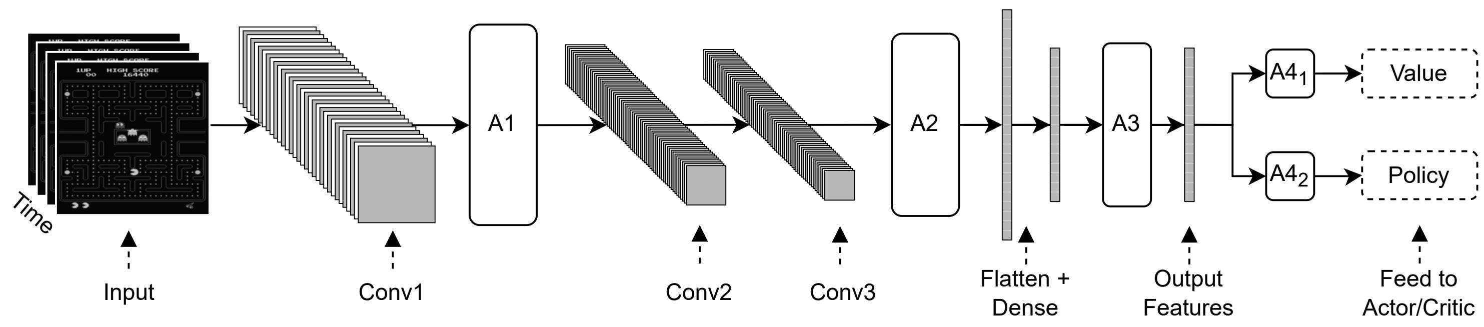


Fig. 2: Diagram for all model architectures. Modules marked with 'A' are placeholders for attention mechanisms (See Section IV-A for details).

7) Separated Dense Self-Attention (DSA_{SP}): We use the same architecture as PPO, and add a dense self-attention layer separately to the policy and value networks. (Dense Self-Attention in Modules $A4_1$ and $A4_2$).

8) Separated Shallow Dense Self-Attention (DSA_{SP_S}): We adopt the same architecture as DSA_{SP}, with the only difference being that the dense self-attention layer is directly connected to the output of the value or policy network, whereas in DSA_{SP}, there is an additional 64-nodes dense layer following the self-attention layer. This variation was created accidentally while we were creating the DSA_{SP} model. Instead of creating the intended model, we created DSA_{SP_S} by mistake, but we observed that it performed well in many games. Therefore, we decided to keep this variation and performed evaluations with it.

B. Environments and Observation Space

We trained and evaluated each model on 7 different Atari Gym environments: MsPacman-v4, Breakout-v4, DemonAttack-v4, Asterix-v4, Freeway-v4, Frostbite-v4, and Krull-v4.

In previous works on attention in reinforcement learning, game environments such as Asterix, Demon Attack, and Ms. Pacman were used to evaluate a model's ability to process and reason with spatial and temporal information from various areas of the frame. These games typically require a player to keep track of multiple moving objects and their spatial relation to the player and other objects, making them potentially suitable for testing the effectiveness of attention mechanisms. Similarly, in [10], attention-based models have shown performance gains in Freeway, Frostbite, and Krull compared to the default PPO model. Although Breakout may not be as well-suited to attention mechanisms, it is a popular benchmark in visual RL research due to its minimal action space, which requires an agent to efficiently predict the ball's motion.

To maintain consistency with [10], we preprocessed the frames of visual data by converting them to greyscale, resizing them to 84x84 pixels, and stacking every 4 consecutive frames to create the final 84x84x4 observation space.

C. Evaluation Methodology and Hyperparameters

We conducted two sets of experiments: one with fixed seeds for setting up the game environments throughout all training episodes, which is a common approach in related works such as [10], and another with varying seeds that change every $10M$ steps.

For experiments with fixed seeds, following [10], we trained each model on the 7 Atari environments for $40M$ timesteps, and evaluated every $25K$ steps. We utilized SB3's standard implementation of Proximal Policy Optimization (PPO), with hyperparameters tuned according to tests done with RL Baselines3 Zoo [26]: Learning Rate was set to lin_$2.5e^{-4}$, Discount Factor was set to 0.99, Adam was used as the optimizer, vf_coef (Value Function Coefficient) was set to 0.5, ent_coef (Entropy Coefficient) was set to 0.01, and clip_range (Clipping Range) was set to decay linearly from 0.1. We compared learning curves between agents during training, and the highest score achieved during evaluation after smoothing the learning curves.

The training was performed on a cluster with V100 GPUs. We trained each game on a single GPU, and it took approximately 24 hours to finish training on a single game environment.

We used RL Baselines3 Zoo's [26] evaluation plotting script to plot the results, but we modified it to apply a 1-D Gaussian filter with a standard deviation of 3 to smooth the evaluation rewards. This filter was applied to both the plots and the tables to mitigate any potential outliers.

The results are presented in Fig. 3, Fig. 4, Table I and Table II.

V. Results and Discussion

A. Experiments Using the Same Random Seed

Fig. 3 and Table I present the results from the experiments that used the same random seed throughout training. While some variations of the attention-based models outperformed plain PPO in a few conditions, the differences were not significant. Among all the games, PPO performed the worst in Frostbite, with LKA achieving the highest score of 9043, while PPO scored only 8881. Various other attention-based

Environments	PPO	SAN	SAN_L	LKA	LKA_L	DSA_{SH}	DSA_{SP}	DSA_{SP_S}
MsPacman-v4	**8325**	4943	4441	3425	2873	1009	1821	4639
Breakout-v4	41	37	38	27	39	19	18	**46**
DemonAttack-v4	**49903**	19191	14821	10275	11767	2937	1508	35281
Asterix-v4	**7517**	3817	3186	4001	4450	1011	2118	5108
Freeway-v4	**32**	23	23	23	24	23	22	**32**
Frostbite-v4	8881	**9093**	8961	9043	8825	896	3132	8970
Krull-v4	7372	**7532**	7522	6948	6957	2379	7156	7321

TABLE I: Highest score achieved for each agent during evaluations using the same random seed (improvements upon PPO underlined, highest scores per environment bolded).

Environments	PPO	SAN	SAN_L	LKA	LKA_L	DSA_{SH}	DSA_{SP}	DSA_{SP_S}
MsPacman-v4	**7449**	4429	5394*	4781*	4042*	2217*	2696*	4270
Breakout-v4	40	37	36	32*	42*	27*	30*	**46**
DemonAttack-v4	30055	31616*	22811*	**38824***	29570*	3182*	2776*	15823
Asterix-v4	5273	3492	5303*	5738*	3683	2351*	1588	**6151***
Freeway-v4	**32**	**32***	**32***	**32***	22	22	23*	31
Frostbite-v4	**9167***	9144*	315	9112*	8983*	2111*	6538*	8894
Krull-v4	7131	6666	7235	7394*	6922	**8186***	6650	7134

TABLE II: Highest score achieved for each agent during evaluations using varying random seed (improvements upon PPO underlined, highest scores per environment bolded, star indicates the score is higher than the corresponding one in Table I).

models also performed better than PPO. These results are consistent with those reported in [10].

In [10], SAN and its variations showed a clear advantage over the default PPO algorithm. However, we did not observe this advantage in our study, and we suspect that it may be due to differences in initialization parameters of the game environment or hyperparameters of the training process. It is known that PPO is highly dependent on hyperparameters [27]–[29]. The fact that the scores received by PPO and other agents in our experiments differ from those in [10] further supports this theory. For example, PPO only received a score of 2503 in Frostbite in [10].

While the difference in results between our study and [10] is not surprising, it weakens the general point made in the latter that adding attention to RL models is generally a good idea. In fact, our experiments showed that adding a single dense self-attention after the final layer of CNN (DSA_{SH}), or separately for policy and value networks (DSA_{SP}), always reduced performance. A study by [30] suggested that attention in RL may cause overfitting and lead to a drop in performance when the environment changes, which could be a reason why attention did not improve RL agents' performance in general in our study. Furthermore, adding attention to later layers is more likely to cause overfitting.

We did observe an exception when the attention layer is placed right next to the output nodes (DSA_{SP_S}); the model performed reasonably well compared to other variations. In every environment, DSA_{SH} greatly underperforms, which we hypothesize is because the policy and value networks might need to attend to separate features in the input (e.g. the policy network pays more attention to the position of the player model while the value network attends to the game score display). In contrast, DSA_{SP_S} worked much better. It not only outperforms DSA_{SH} in all the games, but also achieves comparable or better performance than the baseline PPO and other variations with attention. In contrast, DSA_{SP} performed much worse than DSA_{SP_S}. We suspect this is because the additional dense layer after attention causes more data to be needed to train the attention layer.

Comparing the results across different versions of the SAN and LKA models, we can observe two indications. Firstly, there is no clear difference between adding attention in earlier or later layers, which is consistent with the findings from [10]. In their study, having attention at both early and late layers did not perform significantly better or worse than only having attention at the earlier layer. Secondly, there is no clear advantage between LKA-based models and SAN-based models.

B. Experiments Using Varying Random Seed

The results of the experiments are presented in both Fig. 4 and Table II. In these experiments, we observed a different trend than in the previous experiments where the default PPO agent was more often outperformed by agents with attention. In particular, SAN and LKA based models performed the same as or better than the default PPO in DemonAttrack, Asterix, Freeway, and Krull. Additionally, DSA_{SP_S} outperformed PPO in Breakout, Asterix, and Krull, and DSA_{SH} outperformed PPO in Krull. These results suggest that self-attention may be able to help PPO adapt to environments with variations.

In these experiments, the seeds were randomly changed every $10M$ steps, which determined the initial game state. As the changes are random, a change in the seed may or may not cause the game environment to look significantly different from the previous one. In Fig. 4, we can clearly see a dip in performance for many models at around $30M$ steps, including PPO, in MsPacman, but most of the models were able to recover soon after. Similarly, there was a major dip in performance for DSA_{SH} in Asterix at $20M$ steps. There was also a brief peak in performance for DSA_{SH} in Krull around $10M$ steps followed by a dip in performance. These patterns were not present in Fig. 3.

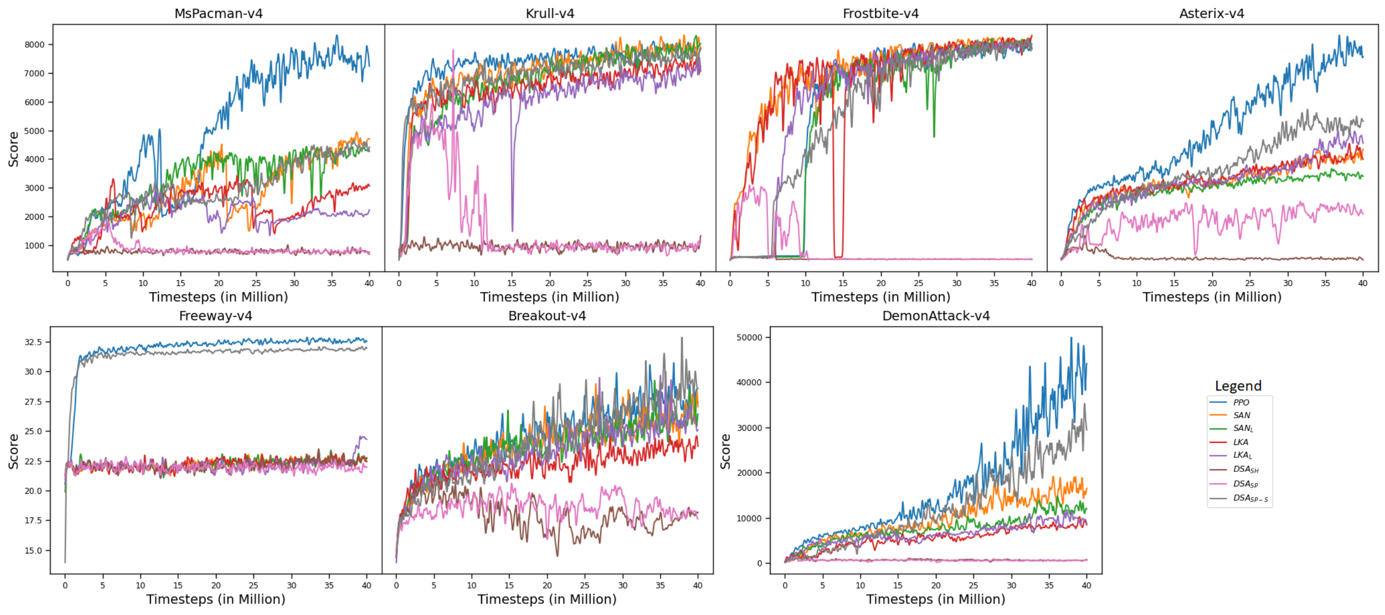


Fig. 3: Evaluation Results Using the Same Random Seed.

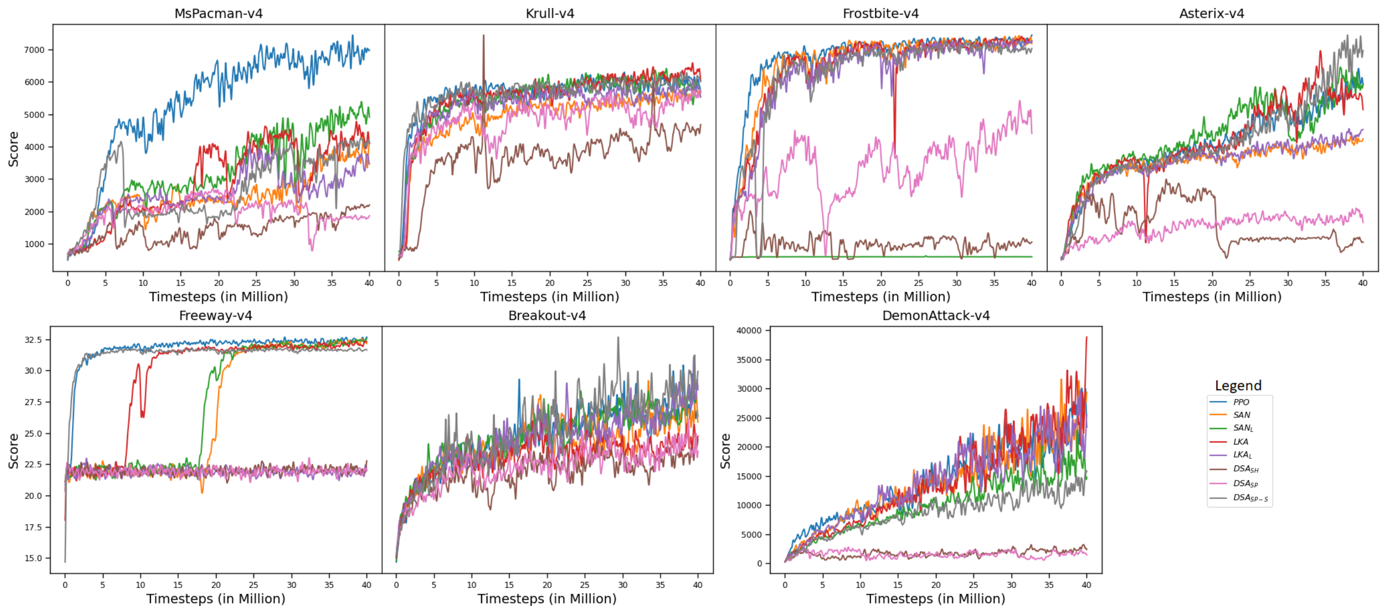


Fig. 4: Evaluation Results Using Varying Random Seed.

The experimental findings revealed that the default PPO model exhibited worse performance when trained with varying seeds across most games, compared to training with a consistent seed. In contrast, attention-based models demonstrated improved performance when the seed changed during training. In Table II, the star symbol denotes instances where the algorithm outperformed its counterpart with constant seeds. Notably, PPO only exhibited better performance in Frostbite, while LKA demonstrated better results in all games. SAN_L, LKA_L, DSA_{SH}, and DSA_{SP} all performed better in over half of the games under varying seed conditions. Intuitively, learning an effective policy becomes more challenging when the game's seeds change during training, as it changes the game's initial state. Our results suggest that attention-based models may be able to learn a more general policy that can handle variations in the game.

VI. Conclusion and Future Work

Our objective is to examine the impact of various self-attention implementations in visual reinforcement learning and understand how attention can enhance agent performance. Our study reveals that incorporating attention can lead to performance improvements in PPO, particularly in game environments where starting states vary. However, it is worth noting that no single attention implementation consistently outperformed the default PPO model. These findings are consistent with previous research trends, as highlighted in studies such as [10], [11].

Three future directions have been identified for this research. Firstly, the current implementation separates dense self-attention (DSA) from visual attention (SAN or LKA). In future work, we aim to explore the combination of these two attention types to assess their collective impact. Additionally, we plan to investigate the potential benefits of attention in conjunction with other reinforcement learning (RL) optimization algorithms, such as DDPG or A2C.

Secondly, the transformer model, which utilizes self-attention, has been successful in processing sequential data, particularly natural language text. Vision transformers (ViT) have also demonstrated promising results on image classification benchmarks and have gained attention in the computer vision research community. We opted to evaluate the LKA model instead of transformer models in our experiment; it is worth noting that the LKA model has outperformed ViT in several computer vision tasks. However, since our LKA based models did not achieve exceptional performance, we also consider exploring the use of ViT models in future work.

Finally, [19] mentions that there may be an interaction effect between visual attention and layer normalization. We are interested in evaluating this effect systematically as well.

References

[1] C. Berner, G. Brockman, B. Chan, V. Cheung, P. Debiak, C. Dennison, D. Farhi, Q. Fischer, S. Hashme, C. Hesse *et al.*, "Dota 2 with large scale deep reinforcement learning," *arXiv preprint arXiv:1912.06680*, 2019.

[2] O. Vinyals, I. Babuschkin, W. M. Czarnecki, M. Mathieu, A. Dudzik, J. Chung, D. H. Choi, R. Powell, T. Ewalds, P. Georgiev *et al.*, "Grandmaster level in starcraft ii using multi-agent reinforcement learning," *Nature*, vol. 575, no. 7782, pp. 350–354, 2019.

[3] J. Perolat, B. de Vylder, D. Hennes, E. Tarassov, F. Strub, V. de Boer, P. Muller, J. T. Connor, N. Burch, T. Anthony *et al.*, "Mastering the game of stratego with model-free multiagent reinforcement learning," *Science*, vol. 378, no. 6623, pp. 990–996, 2022.

[4] P. R. Wurman, S. Barrett, K. Kawamoto, J. MacGlashan, K. Subramanian, T. J. Walsh, R. Capobianco, A. Devlic, F. Eckert, F. Fuchs *et al.*, "Outracing champion gran turismo drivers with deep reinforcement learning," *Nature*, vol. 602, no. 7896, pp. 223–228, 2022.

[5] D. Silver, T. Hubert, J. Schrittwieser, I. Antonoglou, M. Lai, A. Guez, M. Lanctot, L. Sifre, D. Kumaran, T. Graepel *et al.*, "A general reinforcement learning algorithm that masters chess, shogi, and go through self-play," *Science*, vol. 362, no. 6419, pp. 1140–1144, 2018.

[6] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski *et al.*, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, pp. 529–533, 2015.

[7] M.-H. Guo, T.-X. Xu, J.-J. Liu, Z.-N. Liu, P.-T. Jiang, T.-J. Mu, S.-H. Zhang, R. R. Martin, M.-M. Cheng, and S.-M. Hu, "Attention mechanisms in computer vision: A survey," *Computational Visual Media*, pp. 1–38, 2022.

[8] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," *Advances in Neural Information Processing Systems*, vol. 30, 2017.

[9] I. Sorokin, A. Seleznev, M. Pavlov, A. Fedorov, and A. Ignateva, "Deep attention recurrent q-network," *arXiv preprint arXiv:1512.01693*, 2015.

[10] A. Manchin, E. Abbasnejad, and A. van den Hengel, "Reinforcement learning with attention that works: A self-supervised approach," in *Neural Information Processing*, T. Gedeon, K. W. Wong, and M. Lee, Eds. Cham: Springer International Publishing, 2019, pp. 223–230.

[11] L. Yuezhang, R. Zhang, and D. H. Ballard, "An initial attempt of combining visual selective attention with deep reinforcement learning," *arXiv preprint arXiv:1811.04407*, 2018.

[12] L. Chen, K. Lu, A. Rajeswaran, K. Lee, A. Grover, M. Laskin, P. Abbeel, A. Srinivas, and I. Mordatch, "Decision transformer: Reinforcement learning via sequence modeling," *Advances in Neural Information Processing Systems*, vol. 34, pp. 15 084–15 097, 2021.

[13] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," *arXiv preprint arXiv:1707.06347*, 2017.

[14] M.-H. Guo, C.-Z. Lu, Z.-N. Liu, M.-M. Cheng, and S.-M. Hu, "Visual attention network," *arXiv preprint arXiv:2202.09741*, 2022.

[15] A. Galassi, M. Lippi, and P. Torroni, "Attention in natural language processing," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 32, no. 10, pp. 4291–4308, 2020.

[16] H. Furuta, Y. Matsuo, and S. S. Gu, "Generalized decision transformer for offline hindsight information matching," *arXiv preprint arXiv:2111.10364*, 2021.

[17] Q. Zheng, A. Zhang, and A. Grover, "Online decision transformer," in *International Conference on Machine Learning*. PMLR, 2022, pp. 27 042–27 059.

[18] D. Bahdanau, K. Cho, and Y. Bengio, "Neural machine translation by jointly learning to align and translate," *CoRR*, vol. abs/1409.0473, 2015.

[19] H. Zhang, I. Goodfellow, D. Metaxas, and A. Odena, "Self-attention generative adversarial networks," in *International Conference on Machine Learning*. PMLR, 2019, pp. 7354–7363.

[20] A. Kolesnikov, A. Dosovitskiy, D. Weissenborn, G. Heigold, J. Uszkoreit, L. Beyer, M. Minderer, M. Dehghani, N. Houlsby, S. Gelly, T. Unterthiner, and X. Zhai, "An image is worth 16x16 words: Transformers for image recognition at scale," 2021.

[21] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. Riedmiller, "Playing atari with deep reinforcement learning," *arXiv preprint arXiv:1312.5602*, 2013.

[22] V. Konda and J. Tsitsiklis, "Actor-critic algorithms," in *Advances in Neural Information Processing Systems*, S. Solla, T. Leen, and K. Müller, Eds., vol. 12. MIT Press, 1999.

[23] J. Schulman, P. Moritz, S. Levine, M. Jordan, and P. Abbeel, "High-dimensional continuous control using generalized advantage estimation," *arXiv preprint arXiv:1506.02438*, 2015.

[24] A. Raffin, A. Hill, R. Traoré, T. Lesort, N. Díaz-Rodríguez, and D. Filliat, "Decoupling feature extraction from policy learning: assessing benefits of state representation learning in goal based robotics," *arXiv preprint arXiv:1901.08651*, 2019.

[25] A. Raffin, A. Hill, A. Gleave, A. Kanervisto, M. Ernestus, and N. Dormann, "Stable-baselines3: Reliable reinforcement learning implementations," *Journal of Machine Learning Research*, vol. 22, no. 268, pp. 1–8, 2021. [Online]. Available: http://jmlr.org/papers/v22/20-1364.html

[26] A. Raffin, "Rl baselines3 zoo," https://github.com/DLR-RM/rl-baselines3-zoo, 2020.

[27] L. Engstrom, A. Ilyas, S. Santurkar, D. Tsipras, F. Janoos, L. Rudolph, and A. Madry, "Implementation matters in deep rl: A case study on ppo and trpo," in *International Conference on Learning Representations*, 2019.

[28] M. Andrychowicz, A. Raichuk, P. Stańczyk, M. Orsini, S. Girgin, R. Marinier, L. Hussenot, M. Geist, O. Pietquin, M. Michalski *et al.*, "What matters in on-policy reinforcement learning? a large-scale empirical study," *arXiv preprint arXiv:2006.05990*, 2020.

[29] S. Huang, R. F. J. Dossa, A. Raffin, A. Kanervisto, and W. Wang, "The 37 implementation details of proximal policy optimization," in *ICLR Blog Track*, 2022, https://iclr-blog-track.github.io/2022/03/25/ppo-implementation-details/. [Online]. Available: https://iclr-blog-track.github.io/2022/03/25/ppo-implementation-details/

[30] Y. Tang and D. Ha. (2020) Using selective attention in reinforcement learning agents. [Online]. Available: https://ai.googleblog.com/2020/06/using-selective-attention-in.html