FocalNet - Prioritized Experience Replay - Simplified Transformer DQN

Shilong Wu, Yinglong Li, Yiran Shao

https://github.com/B01803717/CS2951F-Final-Project

Abstract

Among the pioneering works in deep reinforcement learning is Deep Q-Network, which demonstrated significant success in mastering image-based tasks like humans. DQN encounters various challenges and limitations in aspects including feature extraction and training inefficiency. In this work, we explore and incorporate a series of techniques to address these challenges, including using Transformer and Focal Modulation blocks for enhanced network capability and implications to reduce computational efforts, and refined Experience Replay algorithms for a smoother and more stable training process. Our experiments demonstrate how these modifications each and together bring visible improvements.

1 Introduction

Just as how humans learn complex tasks, reinforcement learning agents typically attempt to solve an entire task with random actions and refine their strategy through rewards. However, unlike humans who progressively refine their skills, traditional reinforcement-learning methods may struggle with inefficient learning, often due to the vast and unpredictable nature of image-based tasks and the inability to effectively represent and learn from these complex environments.

In recent years, deep reinforcement learning has gained popularity by establishing powerful frameworks for training agents to achieve human-level performance in a wide range of complex image-based tasks, by combining principles of deep learning with reinforcement learning. Among the fundamental works in this field is Deep Q-Networks (DQN), which demonstrated remarkable success in mastering a variety of Atari 2600 games through training directly on raw pixel inputs taken from the game screen. Despite its achievements, DQN faces numerous challenges, including feature extraction capabilities, training inefficiency and instability, etc., due to various limitations in the vanilla design of its network structure and training process.

In our work, a combination of improvements from different aspects is explored, including modifying the image processing network to enhance feature extraction and value estimation capabilities, employing improved Experience Replay techniques to smooth and strengthen model training and performance, and also improving and simplifying network modules to reduce training time and increase efficiency. Choices of incorporating specific techniques are made with consideration of the needs of our game environment and limitations on computation resources. Building upon the foundation laid by DQN and its extensions, this project aims to improve the performance and stability of DQN, trained and tested specifically in the environment of the Atari 2600 game: Pong.

2 Background / Related Work

2.1 Deep Q-Network (DQN)

One of the seminal algorithms in the domain of deep reinforcement learning is Deep Q-Network (DQN), introduced by Mnih et al[6]. A DQN, employs a convolutional neural network (CNN) to approximate a state-value function in a Q-Learning framework. It enables the agents to learn directly from high-dimensional visual inputs, such as raw pixels from the game screen, without the need for specifically handcrafted features for the model. The model showed groundbreaking capabilities to achieve human-level performance in playing several Atari 2600 games, such as Pong, Beam Rider, Breakout, Enduro, etc. Nevertheless, while DQN served as a pioneering model in the field of deep reinforcement learning, it also has its limitations. Challenges such as sample inefficiency and complexity, training instability leading to premature convergence and suboptimal performance, limited generalization capability in unseen or partially observable environments, and high sensitivity to hyperparameters tuning are underscoring the need to refine and extend its capabilities. In response, numerous extensions and improvements on DQN have been proposed in literature, such as implementing double Q-learning[5], improving the experience replay process[9, 1, 8], creating dueling architectures[12], etc., each aimed at addressing specific challenges and improving the performance and stability of DQN.

2.2 Random Experience Replay

The impressive performance of DQN in Atari games is directly associated with the help of Random Experience Replay, which refers to storing the previous episode steps in memory, where random samples are repeatedly drawn during the training process. The concept of Experience Replay helps to break the correlations and temporal patterns between sequential experiences, and allows the agent to learn from a diverse set of experiences. Random replay is an important part of this technique as it prevents the agent from focusing only on recent experiences, thereby preventing it from getting stuck in local minima of a limited range of situations and encouraging exploration of different actions and states. As a result, DQN with the random experience replay technique has demonstrated improved training efficiency, robustness, stability, and performance. Specifically, at each time step t the agent's experience $e_t = (s_t, a_t, r_t, s_{t+1})$ is stored in a dataset $D_t = \{e_1, e_2, ..., e_t\}$, and Q-learning updates are performed on uniformly drawn random mini-batches of experiences from this dataset.

However, limitations of random replay exist, such as additional memory requirements in environments with large state and action spaces or long trajectories, the sampling efficiency when exploring the diverse state-action space, the non-stationarity which may introduce biases during training.

2.3 Prioritized Experience Replay (PER)

Experience replay allows online RL agents to remember and reuse past experiences. The typical approach is to uniformly sample from the replay memory. However, some experiences are more valuable for learning than others. Schaul et al[9] proposes the Prioritized Experience Replay (PER) to replay important transitions more frequently. In contrast to the uniform sampling used in the original DQN, PER assigns different sampling probabilities to each transition in the replay buffer based on its significance, aiming to replay more important transitions more frequently. The key idea behind PER is that an RL agent can learn more effectively from some transitions than others. The importance of a transition is measured by its TD error δ, which indicates how surprising or unexpected the transition is. Transitions with high absolute TD errors are considered more informative and are sampled with higher probability.

However, the prioritized sampling introduces a bias in the learning process, as the distribution of sampled transitions no longer matches the true underlying distribution of experiences. This bias occurs because the agent focuses more on high-priority transitions and less on low-priority ones, potentially leading to overfitting and suboptimal learning. To address this issue, importance sampling correction is employed in PER. Importance sampling correction assigns weights to the sampled transitions based on their priorities, effectively compensating for the bias introduced by prioritized sampling. By incorporating importance sampling weights, the agent can compensate for the bias introduced by prioritized sampling. The weights are used to scale the TD errors during the Q-value update step, ensuring that the contribution of each sampled transition is proportional to its true probability of occurrence. This correction helps to stabilize the learning process and prevents the agent from overfitting to high-priority transitions.

When the priority buffer is empty, the priority of the first frame is initialized to 1. For subsequent frames, the priority P_i is calculated as:

$$P_i = \max_{k=1}^{K} P_k \tag{1}$$

where K is the number of elements in the buffer and P_k represents the priority of the k-th element.

Once the buffer size exceeds the batch size, we update the importance sampling weight parameter β' at each frame t using the following equation:

$$\beta' = \beta + \frac{\text{frame} \cdot (1 - \beta)}{N} \tag{2}$$

By gradually increasing β' over time, we ensure that the importance sampling correction is applied more strongly as training progresses. The importance sampling weight w_i for a transition i is calculated based on β as follows:

$$w_i = (N \cdot P(i))^{-\beta} \tag{3}$$

where N is the total number of transitions in the replay buffer, and $P(i)$ is the probability of sampling transition i based on its priority.

The sampling probability $P(i)$ is obtained by normalizing the priorities:

$$P(i) = \frac{P_i}{\sum_{k=1}^{N} P_k} \tag{4}$$

During the learning process, the agent samples a batch of transitions from the PER buffer. The loss for each transition i in the batch is calculated using the temporal-difference (TD) error and the importance sampling weight w_i. The Q-value update is performed as follows:

$$Q(s_i, a_i) = Q(s_i, a_i) + \text{learning\_rate} \times w_i \times \text{TD\_error} \tag{5}$$

The purpose of using the importance sampling weights in the loss function is to adjust the gradients during the optimization process. By scaling the squared TD errors with the weights, the gradients are modified to give more importance to underrepresented transitions and less importance to overrepresented ones. This helps to correct the bias introduced by prioritized sampling during the learning process.

2.4 Simplified Transformer

Attention mechanisms have gained prominence for their ability to selectively focus on desired information within the input, enabling more effective feature extraction and learning. Studies[10, 3] have explored the usage of attention mechanisms in improving DQN performance. Transformers[11], originally proposed for natural language processing tasks, by using self-attention mechanisms, have demonstrated remarkable success in capturing long-range dependencies and contextual information from sequential data and excel at identifying relevant patterns across temporal and spatial dimensions. Transformers have been used to either help extract more meaningful representations from complex visual inputs[4], or facilitate effective exploration and exploitation strategies and improve decision-making and policy learning efficiency in sequential decision-making domains by capturing relationships extended timeframes[2].

Powerful as Transformer blocks can be in various tasks and the improvements they can bring to our DQN model, they are far from simple, needing to combine attention mechanisms and MLP blocks with skip connections and different network layers. Models can thus be fragile and training speed could be heavily impacted. In a recent study proposed by He et al[7], signal propagation theory and empirical observations are combined to explore to what extent modifications can be made to the standard Transformer block to simplify it while preserving its performance. It is shown through research and experiments that many components such as projection, skip connections, sub-blocks, normalization layers, etc. can be removed without the cost of training speed. The result is a 15% faster training throughput and using 15% fewer parameters, while matching the per-update training speed and performance of standard transformers in experiments.

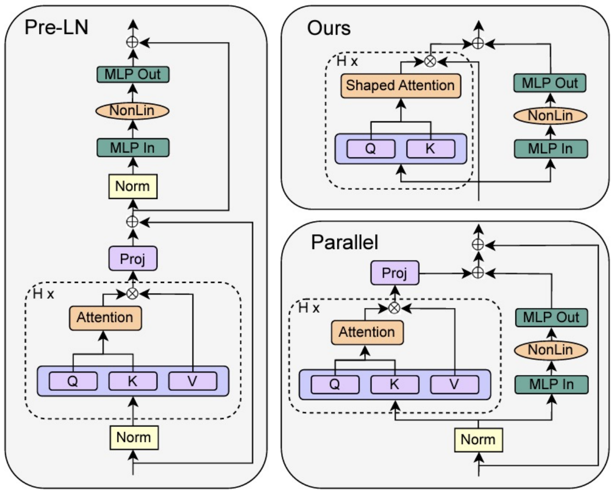


Figure 1: Figure from the Simplifying Transformer Blocks Paper. Left: The standard Pre-LN block. Top Right: The proposed most simplified block. Bottom Right: The parallel block (Wang and Komatsuzaki, 2021).

2.5 FocalNet

Inspired by human visual perception, Focal Modulation Network (FocalNet)[13] has emerged by dynamically modulating its focus on specific regions of interest. In contrast with self-attention (SA) and its first interaction last aggregation process, the Focal Modulation performs first aggregating last interaction. It changes computing token-to-token attention scores that has quadratic complexity to a lightweight operation between a token and its context, resulting in much more efficient and effective performance for modeling token interactions.

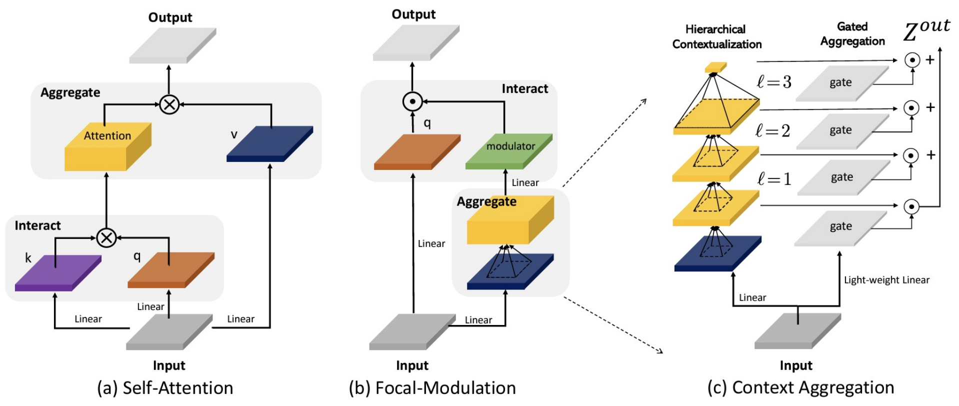


Figure 2: Figure from the FocalNet Paper. Left: Comparing SA (a) and Focal Modulation (b) side by side. Right: Detailed illustration of context aggregation in Focal Modulation (c).

Focal modulation is essentially composed of three parts: (i) focal contextualization to extract contexts from various ranges at different granularity levels, implemented using a stack of depth-wise convolutional layers; (ii) gated aggregation to selectively aggregate and condense context features at different granularity levels through element-wise multiplications; (iii) element-wise affine transformations to merge the aggregated features into the query vector. Essentially, the focal modulation can be instantiated to:

$$y_i = q(x_i) \odot m(i, \mathbf{X}) \tag{6}$$

where q(·) is a query projection function, and m(·) is a context aggregation function whose output is called the *modulator*.

The total complexity for a feature map is $O(HW(3C^2 + C(2L+3) + C\Sigma_l(k^l)^2))$, while a a vanilla self-attention in ViTs take $O((HW)^2C + HW(3C^2))$.

The Focal Modulation technique has demonstrated its superiority through direct improvement upon self-attention mechanisms, with FocalNet outperforming previous SOTA Transformer-backbone models in computer-vision tasks such as object detection and segmentation with much smaller model size and training data size. It came to our attention as Atari games often involve visually cluttered and complex environments, where distinguishing between relevant game elements such as the ball, paddle, obstacles, etc. and irrelevant background details can be crucial for superior performance. By incorporating the Focal Modulation in a DQN, the agent could learn to dynamically modulate its attention on the game screen, focusing on relevant regions in various granularities while filtering out distractions, leading to better performance in gameplay.

3 Model and Methodology

For our experimental setup, we select the $PongNoFrameskip-v4$ environment from the Atari game as our testbed for evaluating the performance of our reinforcement learning agent.

We employ a Prioritized Experience Replay (PER) buffer to sample a batch of image frames based on their importance. The PER buffer assigns higher sampling probabilities to transitions with larger temporal-difference (TD) errors, enabling the agent to prioritize learning from the most informative experiences. This selective sampling strategy enhances the efficiency and effectiveness of the learning process.

The sampled image frames undergo initial processing through a CNN. The CNN serves as a feature extractor, capturing the spatial and temporal information present in the frames. To further enhance the understanding and interpretation of the image frames, we incorporate a Focal Modulation block (including preprocessing and postprocessing between feature maps and patch embeddings) into our model. This allows the agent to effectively interpret the spatial relationships and contextual information presented in the frames, resulting in a rich and informative representation.

The processed features from the Focal Modulation block are then fed into a Simplified Transformer Block. In our approach, we adapt the Transformer to learn and understand the temporal dependencies and patterns within the sequences. It enables the agent to contextualize the information across multiple frames and make informed decisions based on the learned representations.

Finally, the learned representations from the simplified transformer are passed through a fully connected layer to estimate the Q-values for each action.

The below figure is the training result that we got, where the x-axis represent the number of episodes, and the y-axis represent the rewards. Our model achieves reward 0 in episode 253, and reward 20 in episode 303. The Max reward averaging over 10 episodes is 20.6, and the Max reward averaging over 10 episode is 19.7.

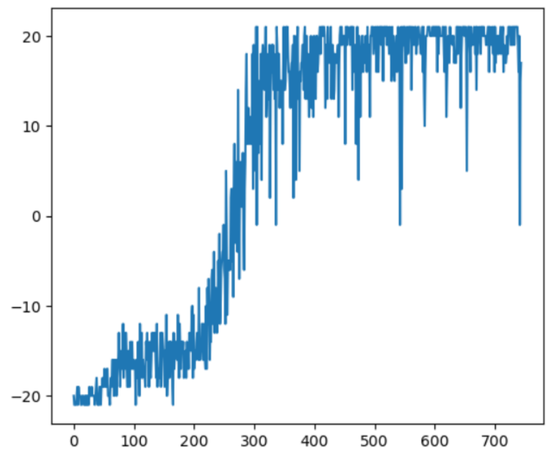


Figure 3: The training result of our proposed model, where the x-axis represent the number of episodes, and the y-axis represent the rewards.

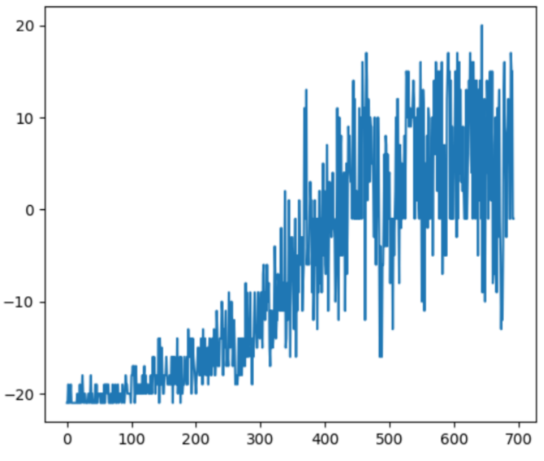
4 Experiments

4.1 Experiment: Random Replay Buffer vs Prioritized Replay Buffer

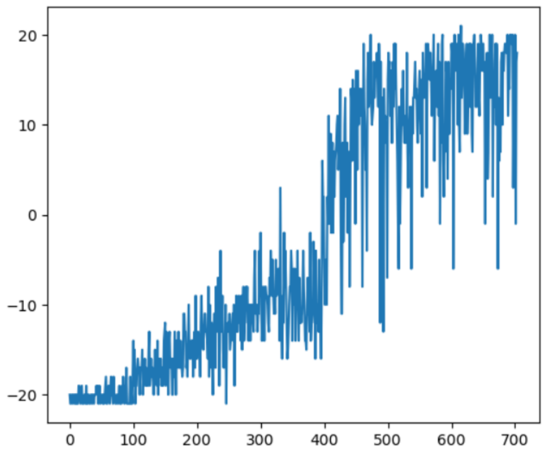
We conduct experiments with two distinct model configurations. The first model (a) incorporates a standard Transformer, a Random Replay Buffer, while excluding the FocalNet. The second model (b) utilizes a standard Transformer architecture and a Prioritized Replay Buffer, and similarly excluding the FocalNet.

Figure 4 presents the experimental results, highlighting the differences in performance between the two models. The model using the Random Replay Buffer achieves a reward of 0 at episode 338 and a reward of 20 at episode 660. The maximum reward averaged over 10 episodes 11.4, while the maximum reward averaged over 40 episodes stands at 7.325. In contrast, the model using the Prioritized Replay Buffer achieves a reward of 0 at episode 331 and a reward of 20 at episode 615. The maximum reward averaged over 10 episodes 17.6, while the maximum reward averaged over 40 episodes stands at 15.55.

Based on these results, we can conclude that although the model with Prioritized Replay Buffer and Random Replay Buffer reached rewards of 0 and 20 with similar episode counts, the model incorporating the Prioritized Replay Buffer has a lot higher reward averaging over 10 and 20 episodes. This observation suggests that the Prioritized Replay Buffer enables the model to consistently achieve high scores during the testing phase. By prioritizing the replay of transitions with higher temporal-difference errors, the agent can effectively learn from the most informative experiences, leading to more stable and reliable performance. In contrast, the model using the Random Replay Buffer exhibits a more sporadic performance pattern. Although it occasionally attains high scores, the average rewards over 10 and 20 episodes are notably lower compared to the Prioritized Replay Buffer model. This implies that while the Random Replay Buffer model may achieve high scores in certain instances, it struggles to maintain consistent performance across multiple episodes.



(a) With Random Replay Buffer



(b) With Prioritized Replay Buffer

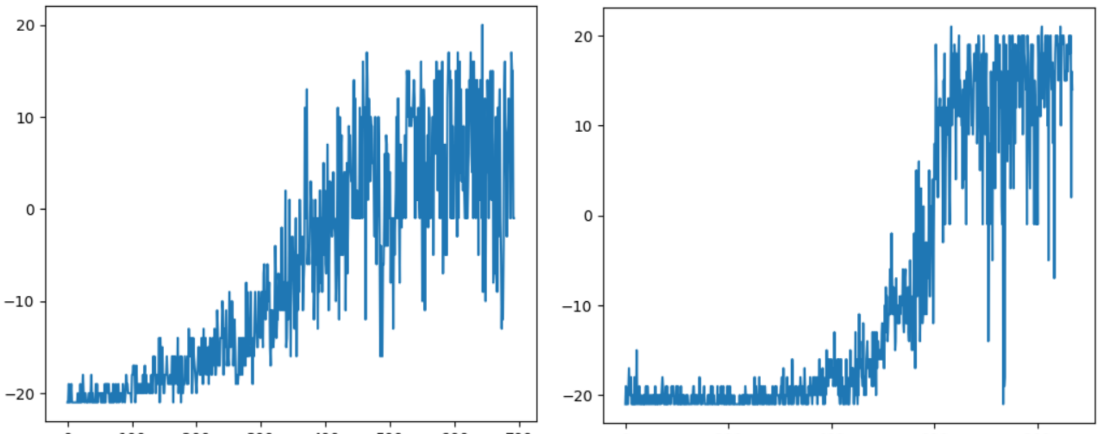
Figure 4: Comparing the training result of two different models, where the x-axis represent the number of episodes, and the y-axis represent the rewards. Left(a): Normal Transformer DQN with Random Replay Buffer. Right(b): Normal Transformer DQN with Prioritized Replay Buffer.

4.2 Experiment: No FocalNet vs FocalNet

We conduct experiments with two distinct model configurations. The first model incorporates a standard Transformer architecture without FocalNet and a Random Replay Buffer. The second model utilizes a standard Transformer with FocalNet and a Random Replay Buffer.

Figure 5 presents the experimental results, highlighting the differences in performance between the two models. The model without FocalNet achieves a reward of 0 at episode 338 and a reward of 20 at episode 660. The maximum reward averaged over 10 episodes 11.4, while the maximum reward averaged over 40 episodes stands at 7.325. In contrast, the model with FocalNet reaches a reward of 0 at episode 565 and a reward of 20 at episode 633. The maximum reward averaged over 10 episodes for this model is 18.5, while the maximum reward averaged over 40 episodes stands at 16.725.

Based on these results, we can conclude that the model using FocalNet exhibits a slower initial learning phase, taking more episodes to reach a reward of 0 compared to the model without FocalNet. This slower convergence can be attributed to the increased complexity introduced by the FocalNet architecture. The focal self-attention mechanism employed by FocalNet requires additional computational resources and training time to effectively capture and process the local and global dependencies within the input images. However, once the FocalNet model reaches a reward of 20, it demonstrates superior stability and consistently produces higher rewards compared to the model without FocalNet. This suggests that the integration of FocalNet allows the agent to effectively interpret and utilize the rich spatial and contextual information present in the input frames. By capturing both local and global dependencies, FocalNet facilitates the learning of more informative and discriminative features, enabling the agent to make better-informed decisions.



(a) Without FocalNet (b) With FocalNet

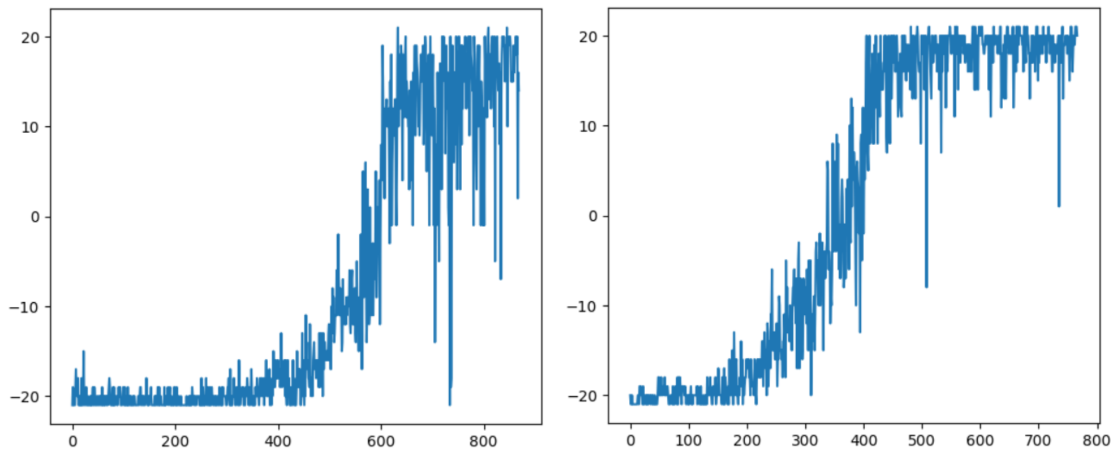
Figure 5: Comparing the training result of two different models, where the x-axis represent the number of episodes, and the y-axis represent the rewards. Left(a): Normal Transformer DQN with Random Replay Buffer. Right(b): Normal Transformer DQN with Random Replay Buffer and FocalNet.

4.3 Experiment: Standard Transformer vs Simplified Transformer

We conduct experiments with two distinct model configurations. The first model incorporates a standard Transformer architecture with FocalNet and a Random Replay Buffer. The second model utilizes a Simplified Transformer with FocalNet and a Random Replay Buffer.

Figure 5 presents the experimental results, highlighting the differences in performance between the two models. The model with Standard Transformer achieves a reward of 0 at episode 565 and a reward of 20 at episode 633. The maximum reward averaged over 10 episodes 18.5, while the maximum reward averaged over 40 episodes stands at 16.725. In contrast, the model with Simplified Transformer reaches a reward of 0 at episode 338 and a reward of 20 at episode 405. The maximum reward averaged over 10 episodes for this model is 20.0, while the maximum reward averaged over 40 episodes stands at 19.125. Most importantly, the time it takes to train the model is a lot less. For the model with Standard Transformer, it took 9 hours 50 minutes to train, while the model with Simplified Transformer only took 7 hours 52 minutes to train.

Based on these results, we can conclude that the model with Simplified Transformer can achieve a little better result with significant less time. The observed performance gains can be attributed to the reduced complexity of the Simplified Transformer architecture. By eliminating redundant or less informative components, such as certain projection layers, skip connections, and normalization layers, the Simplified Transformer reduces the number of parameters that need to be trained. This reduction in model complexity allows for faster training and optimization, as the agent can focus on learning the most relevant and meaningful representations from the input data.



(a) With Standard Transformer (b) With Simplified Transformer

Figure 6: Comparing the training result of two different models, where the x-axis represent the number of episodes, and the y-axis represent the rewards. Left(a): Normal Transformer DQN with Random Replay Buffer and FocalNet. Right(b): Simplified Transformer DQN with Random Replay Buffer and FocalNet.

5 Conclusion

In this study, we investigated the impact of incorporating advanced architectural components and training strategies in reinforcement learning models for image-based tasks. Specifically, we examined the effects of integrating FocalNet, Simplified Transformer, and Prioritized Experience Replay (PER) on the performance and learning efficiency of the agents.

Through a series of experiments, we compared models with different combinations of these components to assess their individual and combined contributions. The results demonstrate that each of these architectural choices and training strategies brings unique benefits to the reinforcement learning process.

The combination of FocalNet, Simplified Transformer, and PER in a single reinforcement learning model yielded the most impressive results. By leveraging the strengths of each component, the integrated model achieved faster learning, superior stability, and consistently high performance across multiple episodes. The FocalNet architecture enhanced the agent's visual understanding, the Simplified Transformer streamlined the learning process, and PER optimized the utilization of the agent's experience.

In conclusion, this study highlights the significant benefits of incorporating advanced architectural components and training strategies in reinforcement learning models for image-based tasks. The integration of FocalNet, Simplified Transformer, and Prioritized Experience Replay proves to be a powerful combination for enhancing the performance, learning efficiency, and stability of reinforcement learning agents.

In future work, we would hope to explore more enhanced Experience Replay techniques and implement other approaches to further exploit the use of Experience memories and sequential actions. We also hope to conduct more experiments and verify the performance of these techniques on more complex environments including other image-based tasks.

References

[1] Marcin Andrychowicz et al. *Hindsight Experience Replay.* 2018. arXiv: `1707.01495 [cs.LG]`.

[2] Lili Chen et al. *Decision Transformer: Reinforcement Learning via Sequence Modeling.* 2021. arXiv: `2106.01345 [cs.LG]`.

[3] Jinyoung Choi, Beom-Jin Lee, and Byoung-Tak Zhang. *Multi-focus Attention Network for Efficient Deep Reinforcement Learning.* 2017. arXiv: `1712.04603 [cs.LG]`.

[4] Nicklas Hansen, Hao Su, and Xiaolong Wang. *Stabilizing Deep Q-Learning with ConvNets and Vision Transformers under Data Augmentation.* 2021. arXiv: `2107.00644 [cs.LG]`.

[5] Hado van Hasselt, Arthur Guez, and David Silver. *Deep Reinforcement Learning with Double Q-learning.* 2015. arXiv: `1509.06461 [cs.LG]`.

[6] Matthew Hausknecht and Peter Stone. *Deep Recurrent Q-Learning for Partially Observable MDPs.* 2017. arXiv: `1507.06527 [cs.LG]`.

[7] Bobby He and Thomas Hofmann. *Simplifying Transformer Blocks.* 2023. arXiv: `2311.01906 [cs.LG]`.

[8] Steven Kapturowski et al. "Recurrent experience replay in distributed reinforcement learning". In: *International conference on learning representations.* 2018.

[9] Tom Schaul et al. *Prioritized Experience Replay.* 2016. arXiv: `1511.05952 [cs.LG]`.

[10] Ivan Sorokin et al. *Deep Attention Recurrent Q-Network.* 2015. arXiv: `1512.01693 [cs.LG]`.

[11] Ashish Vaswani et al. *Attention Is All You Need.* 2023. arXiv: `1706.03762 [cs.CL]`.

[12] Ziyu Wang et al. "Dueling network architectures for deep reinforcement learning". In: *International conference on machine learning.* PMLR. 2016, pp. 1995–2003.

[13] Jianwei Yang et al. *Focal Modulation Networks.* 2022. arXiv: `2203.11926 [cs.CV]`.