electronics

MDPI

Article

Task-Based Visual Attention for Continually Improving the Performance of Autonomous Game Agents

Eren Ulu [1,2,*], Tolga Capin [2], Bora Çelikkale [3] and Ufuk Celikcan [1]

[1] Department of Computer Engineering, Hacettepe University, 06800 Ankara, Türkiye; celikcan@cs.hacettepe.edu.tr
[2] Department of Computer Engineering, TED University, 06420 Ankara, Türkiye; tolga.capin@tedu.edu.tr
[3] Department of Software Engineering, Cankaya University, 06790 Ankara, Türkiye; boracelikkale@cankaya.edu.tr
* Correspondence: eren.ulu@tedu.edu.tr

Abstract: Deep Reinforcement Learning (DRL) has been effectively performed in various complex environments, such as playing video games. In many game environments, DeepMind's baseline Deep Q-Network (DQN) game agents performed at a level comparable to that of humans. However, these DRL models require many experience samples to learn and lack the adaptability to changes in the environment and handling complexity. In this study, we propose Attention-Augmented Deep Q-Network (AADQN) by incorporating a combined top-down and bottom-up attention mechanism into the DQN game agent to highlight task-relevant features of input. Our AADQN model uses a particle-filter -based top-down attention that dynamically teaches an agent how to play a game by focusing on the most task-related information. In the evaluation of our agent's performance across eight games in the Atari 2600 domain, which vary in complexity, we demonstrate that our model surpasses the baseline DQN agent. Notably, our model can achieve greater flexibility and higher scores at a reduced number of time steps.Across eight game environments, AADQN achieved an average relative improvement of 134.93%. Pong and Breakout games both experienced improvements of 9.32% and 56.06%, respectively. Meanwhile, SpaceInvaders and Seaquest, which are more intricate games, demonstrated even higher percentage improvements, with 130.84% and 149.95%, respectively. This study reveals that AADQN is productive for complex environments and produces slightly better results in elementary contexts.

Keywords: deep reinforcement learning; deep Q-learning; layer-wise relevance propagation; particle filter; bottom-up and top-down visual attention; saliency map; convolutional neural network

check for updates

Citation: Ulu, E.; Capin, T.; Çelikkale, B.; Celikcan, U. Task-Based Visual Attention for Continually Improving the Performance of Autonomous Game Agents. *Electronics* **2023**, *12*, 4405. https://doi.org/10.3390/electronics12214405

Academic Editor: Dah-Jye Lee

Received: 3 September 2023
Revised: 17 October 2023
Accepted: 20 October 2023
Published: 25 October 2023

Copyright: © 2023 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (https://creativecommons.org/licenses/by/4.0/).

1. Introduction

Reinforcement Learning (RL) has achieved great success in solving several tasks, such as Atari games in the arcade learning environment (ALE) [1], where the sequence of environmental observations serves as the basis for determining decisions [2]. RL algorithms process environmental data to learn a policy that chooses the best action to maximize cumulative reward [3]. During RL, the agent interacts with its environment to arrive at different states by performing actions that cause the agent to obtain positive or negative rewards.

Nevertheless, the limited adaptability of RL approaches poses challenges when dealing with complex tasks. Consequently, developing methods that enable the application of RL to complex environments is a significant research problem [3]. The goal is to enhance the capabilities of RL algorithms to effectively handle intricate tasks, allowing for more robust and efficient learning in complex scenarios.

The combination of deep neural networks (DNNs) and Q-learning [4] led to the deep Q-network (DQN) algorithm [2,5], which has been used in various works to develop models for complex tasks. These agents demonstrate remarkable success, surpassing human-level performance and outperforming baseline benchmarks [6].

However, the DQN algorithm can suffer inefficiency and inflexibility, which can limit its performance [6,7]. It is vulnerable in complex environments regarding data efficiency, as there is an infinite number of possible experiences in such environments, and there is a need to process many states, requiring high computational power [8]. The DQN algorithm has received criticism for its need for more flexibility, specifically when adapting to changes in the environment or incorporating new tasks. In such cases, the algorithm typically requires a complete restart of the learning process, which can be time-consuming and inefficient [7]. It also has certain limitations in terms of generalization compared to regular neural networks [9].

Researchers have developed a range of extensions to the original DQN algorithm to address these issues, such as double DQN [10]. Moreover, several works have proposed the use of visual attention mechanism [11–14], which allows the network to focus on specific regions of an input image rather than processing the entire image at once [11].

The attention mechanism can be implemented in various ways, such as using convolutional neural networks (CNNs) to extract image features, recurrent neural networks (RNN), and visual question answering (VQA) models that process the textual question input [15–17]. The VQA attention mechanism serves the purpose of selectively directing attention toward image regions that are crucial for answering a question. The attention mechanism effectively prioritizes the attended regions by assigning importance scores or weights to different image regions based on their relevance to the question. As a result, these regions are granted a higher degree of significance in the overall analysis [15–20].

In this work, we address the aforementioned limitations of the baseline DQN agent and propose Attention-Augmented Deep Q-Network (AADQN) by extending DQN with a dual attention mechanism to highlight task-relevant features of input. The dual attention mechanism unifies bottom-up and top-down visual attention mechanisms within the AADQN model, as illustrated in Figure 1. This way, AADQN allows the agent to efficiently concentrate on the most relevant parts of the input image. The top-down attention (TDA) incorporates particle filters, enabling dynamic identification of task-related features. The integration of particle filters provides a regularization effect, effectively mitigating overfitting and enhancing generalization. The bottom-up attention consists of two parts: a preliminary bottom-up attention (PBA) module and a bottom-up attention refinement (BAR) module. The PBA extracts the feature importances, which TDA subsequently utilizes to initialize the particle filters. BAR then refines attention by considering the saliency value of the features.

AADQN differs from the previous work using visual attention with DQN [11–14] by refining top-down focus through bottom-up attention. The collaboration of both attention mechanisms enables more accurate decision-making and improves the model's robustness to variations, reducing its sensitivity to noise. It is also fundamentally different from the VQA-based RL approaches, which aim to align visual and textual information to make decisions and are typically applied to tasks where understanding and reasoning about visual and textual data are essential, such as answering questions about the content of an image [15–20]. Our dual attention mechanism, enhanced by a particle filter, can be applied to various tasks without relying on textual information, enabling AADQN to adaptively focus on different regions of the input.

Overall, the contributions of our work can be summarized as follows:

- AADQN introduces a novel particle-filter -enhanced dual attention approach to DQN, integrating both bottom-up and top-down attention mechanisms. This unified approach handles the complexity of the environment by extracting essential task-related features, thereby enhancing efficiency and improving overall performance.
- AADQN enhances the flexibility performance of DQN by freezing the unimportant or irrelevant units of CNN's inner layers during the decision-making process. These units, identified by low relevance scores assigned by AADQN, undergo transfer learning to optimize the model's robustness and flexibility.

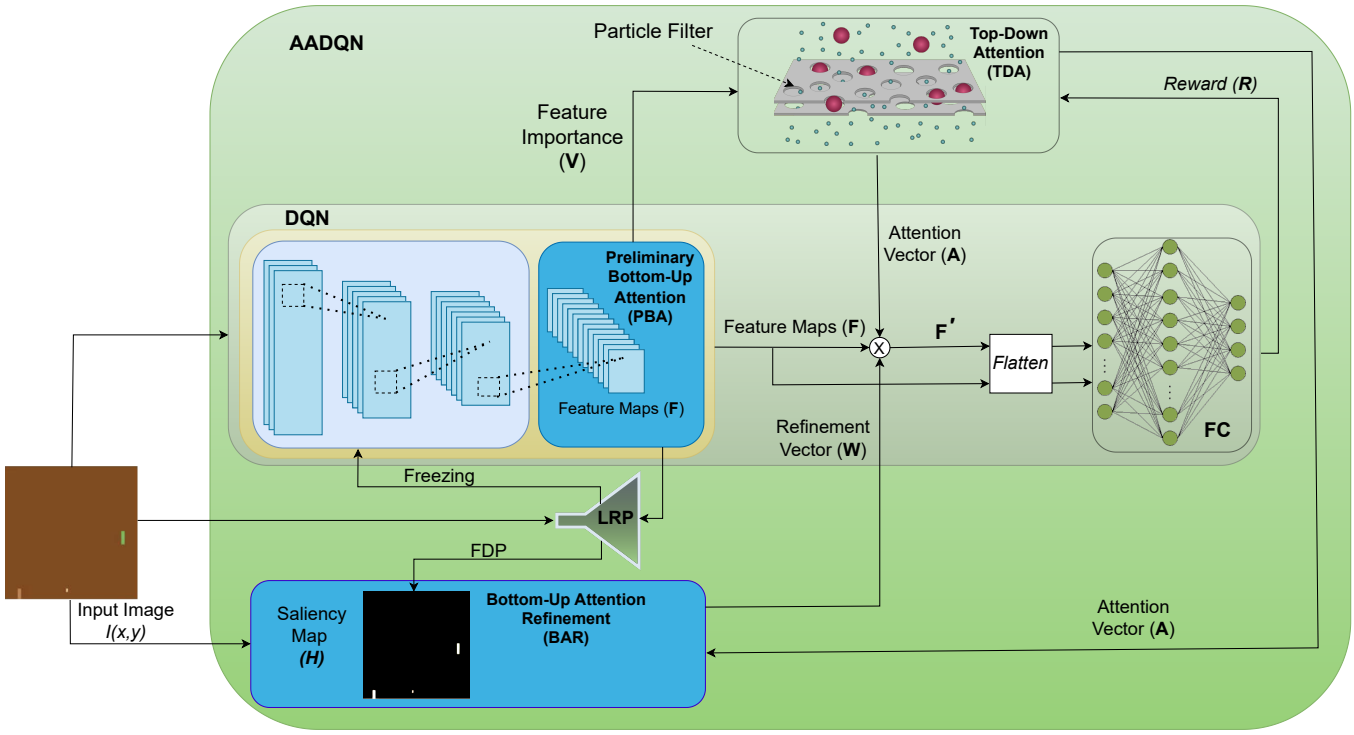


Figure 1. Our proposed Attention-Augmented Deep Q-Network extends DQN with a particle filter-enhanced dual attention mechanism as follows: The top-down attention (TDA) mechanism employs particle filters to compute the attention vector. The bottom-up attention mechanism comprises two components: the preliminary bottom-up attention (PBA) and the bottom-up attention refinement (BAR). PBA determines the feature importances for initializing the particle states. BAR enhances focus by considering the saliency map of the input. Layer-wise relevance propagation (LRP) determines the feature decomposed pixels of the input and freezes the unimportant neurons on CNN.

The remainder of this paper is organized as follows: Section 2 gives an overview of the related work. Then, Section 3 describes the background of Q-learning and baseline DQN. The proposed algorithm and algorithmic efficiency analysis are presented in Sections 4 and 5.4, respectively. These are followed by the experiment results and performance comparison reported in Section 5. Finally, Section 6 concludes the paper.

2. Related Work

Reinforcement learning algorithms, such as DQN [2], have shown great success in learning to play Atari 2600 games from raw pixel inputs. While DQN [2] can learn to play games effectively, it can suffer from instability and inefficiency in learning [6,7].

To address these shortcomings, several modifications have been proposed to the original DQN algorithm [6,8,10,21,22]. The prioritized experience replay method, proposed by Schaul et al. [21], prioritizes experience replay based on the importance of the sample, so that it replays critical transitions more frequently, and therefore learns more efficiently. A dueling neural network architecture, which is introduced by Wang et al. [10], separates the state value function and the action function, resulting in more stable learning and better performance than the state of the art on the Atari 2600 domain. Several works have presented a distributed architecture for deep reinforcement learning [8,22,23]. Such architectures distribute the learning process across multiple parallel agents, which enables more efficient exploration of the state space and faster convergence [8]. Another group of work has aimed to learn additional auxiliary functions with denser training rewards to improve the sample efficiency problem [24]. Some research has combined multiple techniques, such as prioritized experience replay, dueling networks, and distributional RL, to achieve performance enhancements over DQN [6].

A number of studies have used attentional mechanisms to improve the performance of their models [12,13,25–27]. Some of these use bottom-up attention, allowing the agent to selectively attend to different parts of the input, regardless of the agent's task. Others have applied attention to DRL problems in the Atari game domain [11,12,28]. Additionally, several others have explored attention by incorporating a saliency map [29] as an extra layer [30] that modifies the weights of CNN features.

Studies that use the basic bottom-up saliency map [29] as an attention mechanism in RL have used many hand-crafted features as inputs [31]. Yet, these models show an inability to attend to multiple input information with sufficient importance simultaneously. Top-down attention mechanisms can also be used to improve the performance of DQNs by allowing the agent to selectively attend to relevant parts of the input based on its current tasks [32].

Most of the previous DRL studies that use attention mechanisms are generally based on back-propagation learning [13,27], which is actually not ideal to be used by DRL [25,30] as it can lead to inflexibility [33]. Few other works have proposed to learn attention without back-propagation [25,30]. Yuezhang et al. infer that attention from optical flow only applies to issues involving visual movement [30]. The Mousavi attention mechanism uses a bottom-up approach, which is directed by the inherent characteristics of the input information regardless of the reward [25]. We incorporate a unified bottom-up and top-down visual attention mechanism into the DQN model to improve the game agent's training stability and efficiency.

3. Background on Q-Learning and DQN

Basic Q-learning learns a function that updates an action-value (Q) table for different states and actions, enabling an agent to learn the optimal policy for taking actions in an environment by maximizing expected cumulative rewards based on the current, as represented by the variable s in (Equation (1)) [34]:

$$Q(s,a) = R(s,a) + \gamma maxQ(s',a') \tag{1}$$

In the above; the Q-value for a state s when taking action a is calculated as the sum of the immediate reward $R(s,a)$ and the highest Q-value achievable from the subsequent state s'. The variable γ, referred to as the discount factor, governs the extent to which future rewards contribute to this calculation.

The agent performs the sequences of actions to maximize the total expected reward. The Q-value is formalized as follows [34]:

$$Q(s,a) \leftarrow Q(s,a) + \alpha\left[R(s,a) + \gamma maxQ(s',a') - Q(s,a)\right] \tag{2}$$

where α is the learning rate, and $\left[R(s,a) + \gamma maxQ(s',a') - Q(s,a)\right]$ is called the temporal difference error.

To address the limitations of basic Q-learning, such as slow convergence and the need for manual feature engineering, Mnih et al. proposed a deep Q-network (DQN) [2] to utilize deep neural networks to approximate the learning function, allowing for more complex and efficient learning directly from raw sensory inputs. The DQN architecture (Figure 2) uses a CNN to process the input image representing the game state and produces Q-values for all available actions. The CNN's convolutional layers extract important features, generating a feature map. This feature map is then flattened and fed into a fully connected (FC) network, which computes the Q-values for the actions in the current state.

In traditional supervised neural networks, learning the target is always fixed at each step, and only estimates are updated. In contrast, there is no specific target in reinforcement learning: the target itself is estimated, and therefore, it changes during learning. However, changing the target during the training process leads to unstable learning. To resolve this issue, DQN uses a dual neural network architecture, referred to as the prediction and target network, in which there are two Q-networks with identical structures but different parameters. Learning is primarily accomplished through the update of the prediction network at each step to minimize the loss between its present Q-values and the target

Q-values. The target network is a copy of the prediction network that is updated less frequently, usually by copying the weights of the prediction network every n step. This way, the target network serves to maintain stability and to prevent the prediction network from overfitting to the current data by keeping the target values fixed for a window of n time steps.

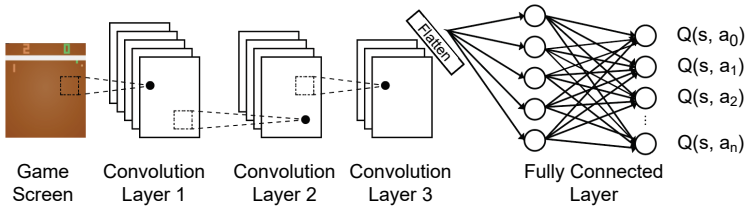


Figure 2. DQN architecture.

4. Attention-Augmented DQN Algorithm (AADQN)

Our proposed AADQN approach enhances the DQN architecture by incorporating bottom-up and top-down attention mechanisms. This integration empowers the game agent to selectively attend to task-relevant features while disregarding irrelevant features, thereby reducing the complexity of the task at hand.

As illustrated in Figure 1, the CNN part of the model takes the present state of the game as input I and extracts the set of feature maps $\boldsymbol{\mathcal{F}} = \{\boldsymbol{F_j} : 1 \leq j \leq D\}$. Using $\boldsymbol{\mathcal{F}}$, PBA defines the feature importance vector $\boldsymbol{V} = [V_1, V_2, \ldots, V_D]$. The TDA mechanism then generates particle vectors $\boldsymbol{\omega_i} : 1 \leq i \leq m$, each with D dimensions representing the D feature maps, and converts them to the attention vector $\boldsymbol{A} = [A_1, A_2, \ldots, A_D]$. Next, the BAR mechanism refines attention vector $(\boldsymbol{A})$ by considering the saliency map $\boldsymbol{H}$.

After calculating the CNN mid-layer relevance scores by LRP rules, AADQN freezes the irrelevant units on the mid-layers. Freezing the irrelevant neurons on the mid-layers improves the model's robustness and flexibility in complex environments.

The overall flow of AADQN is given in Algorithm 1. In the following, we break down this process and describe the AADQN modules in detail.

4.1. Preliminary Bottom-Up Attention (PBA)

PBA determines the feature importance vector $\boldsymbol{V}$ of the set of feature maps $\boldsymbol{\mathcal{F}}$, which is then utilized by the TDA mechanism to initialize the particles' states.

To generate feature importance $\boldsymbol{V}$, we first compute the normalized mean activation value $\hat{F}_j$ of each feature map $\boldsymbol{F_j}$ in $\boldsymbol{\mathcal{F}}$, where $1 \leq j \leq D$. The mean activation value of a feature map gives a measure of the activity within a CNN, s.t., it serves to assess the level of activity in a feature map.

To select the most informative features, we calculate the entropy of each feature map $\boldsymbol{F_j}$ using the Shannon entropy formula [35] given by

$$E_j = -\sum_{i=1}^{B} b_i \log_2 b_i \tag{3}$$

where B is the number of feature value ranges, and b_i is the probability of observing a feature value in the ith range [35].

The entropy E_j is then also normalized:

$$\hat{E}_j = \frac{E_j}{\sum_{i=1}^{D} E_i} \tag{4}$$

Algorithm 1: Attention-Augmented DQN (AADQN)

Initialize DQN network;
Initialize particles: $\boldsymbol{\omega_i} : 1 \leq i \leq m$;
while *stopping criterion not met* **do**
  get input frame: $I(x, y)$;
  extract CNN feature maps;
  calculate features' importance value: $V_j = \frac{1}{2}(\hat{F}_j + \hat{E}_j)/\chi_j^2$;
  calculate the prior probability of each particle: $p_j = V_j / \max(V_1, V_2, \ldots, V_D)$;
  update all particle states based on prior probability;
  particle state normalization: $\hat{\omega}_i^j = \frac{\omega_i^j}{\sum_{j=1}^{D} \omega_i^j}$;
  get reward R_t from DQN;
  calculate state value of particles: $Q(s_t, \hat{\omega}_i)$;
  calculate likelihood probability of particles: $P(R_t|\omega_i) \propto \exp(-(\epsilon_i - \min(Q(s_t, \hat{\omega}_i))))$;
  likelihood normalization: $p(\omega_i) = \frac{P(R_t|\omega_i)}{\sum_{i=1}^{m} P(R_t|\omega_i)}$;
  attention update: $\boldsymbol{A_j} = \frac{1}{m}\sum_{i=1}^{m} \omega_i^j / \sum_{j=1}^{D} \frac{1}{m} \sum_{i=1}^{m} \omega_i^j$;
  saliency map ($\boldsymbol{H}$) extracted from input and normalized in $[0, 1]$: ;
  calculate relevance score of neurons: $\sum_i \phi_l^{i \leftarrow j} = \phi_{l+1}^j$;
  extract features FDP information: $\mathbf{P_j} = I(x, y) \quad s.t. \quad \phi_j(x, y) > \bar{\phi}_j$;
  calculate feature saliency value: $H_j = \frac{1}{|P_j|} \sum_{x,y \in \mathbf{P_j}} H(x, y)$;
  extract the refinement vector: W_j;
  refine feature maps: $F^{'} = F \odot (W \odot A)^{\Uparrow}$;
  if $A_j > \theta$ **then**
    freeze irrelevant neurons corresponding to the j-th feature map;
  end
end

Features with uniform distribution are not desired in CNNs, as they do not provide the network with the ability to learn complex patterns and features in the input data. To this end, the chi-square (χ^2) Test [36] given by

$$\chi_j^2 = \sum_{i=1}^{K^2} \frac{(O_i - \mu_i)^2}{\mu_i} \tag{5}$$

is used to improve the feature selection, where the χ^2 metric is used to assess the uniformity of the activation values. Here, O_i represents the activation values of the feature maps, while μ_i corresponds to the activation values that are uniformly distributed. A higher value of χ^2 indicates the lower importance of the feature in determining the output.

The evaluation of feature importance typically relies on activation values [37,38] or, in certain cases, the entropy of feature maps [39,40]. We use both metrics to achieve

better performance across diverse environments and enhance the assessment of feature importance, as in

$$V_j = \frac{\frac{1}{2}(\hat{F}_j + \hat{E}_j)}{\chi_j^2} \tag{6}$$

V_j makes up the feature importance vector $\boldsymbol{V}$, which is used in TDA to initialize the particle states.

4.2. Top-Down Attention (TDA)

The TDA mechanism helps the agent focus on task-specific information by assigning weights to feature maps through the attention vector $\boldsymbol{A} = [A_1, A_2, \ldots, A_D]$, where A_j represents the attention weight of the feature map $\boldsymbol{F_j}$. This mechanism uses a particle filter that generates a set of particles $\boldsymbol{\Omega} = \{\boldsymbol{\omega_i} : 1 \leq i \leq m\}$ to approximate the posterior distribution over the attention vector for the current task. Each particle is a D-dimensional binary vector $\boldsymbol{\omega}$ that estimates the probability distribution of the feature maps with respect to importance. In a particle vector $\boldsymbol{\omega_i}$, the jth element ω_i^j corresponds to the feature map $\boldsymbol{F_j}$ and indicates whether that feature is useful ('1') or not ('0') for the task at hand. This binary representation provides a selection of essential features [41–44].

The algorithm iteratively updates the particles using the reward feedback. Then, re-samples the particles (Figure 3) and generates a new sample set. Finally, the distribution of the feature maps is estimated.

Combining gradient descent and particle filter provides the following advantages:

- Following the local gradient within the particle filter allows for finding minima (including the global minimum) with fewer particles [45].
- Enhancing the gradient descent method with multiple hypotheses helps to avoid getting stuck in local minima [46].

Similar to DQN [2], we apply $D = 64$ feature maps in its final CNN layer. To estimate the distribution of these feature maps, the number of particles required depends on the distribution complexity. Increasing the number of particles generally enhances the accuracy of distribution estimation. However, there is no universally optimal number of particles that applies to all scenarios. It is common to employ a sufficiently large number of particles to ensure reliable estimates [43,44]. In our case, $m = 250$ particles are utilized to process the set of 64 feature maps.

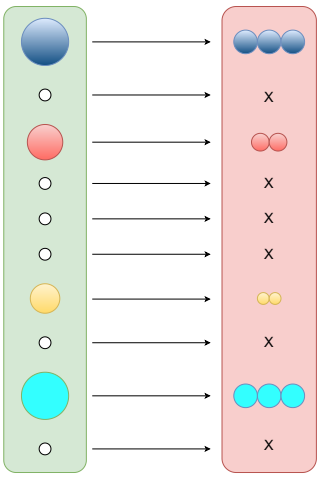


Figure 3. The size of each particle corresponds to its weight. In the re-sampling process, some particles are selected multiple times, while others are not chosen, as indicated by the 'x' symbol. Re-sampling removes particles with very low probabilities (white particles) and replaces them with particles that have higher probabilities. A distinct color is assigned to each particle and its corresponding resampled particles.

In order to generate the particle set $\boldsymbol{\Omega}$ that TDA uses to compute the attention vector $\boldsymbol{A}$, we first normalize the importance value V_j of each feature $\boldsymbol{F_j}$ by the maximum value across all feature maps and convert to the probability. This ensures that the most active feature will have a probability of 1 as follows:

$$p(V_j) = \frac{V_j}{\max(\boldsymbol{V})} \tag{7}$$

Then, the binary ω_i^j are generated randomly using a Bernoulli distribution, i.e., $p(\omega_i^j)$ has a probability of $p(V_j)$ for $\omega_i^j \leftarrow 1$ and a probability of $1 - p(V_j)$ for $\omega_i^j \leftarrow 0$. In this way, TDA initializes the particles by focusing on the most informative features with regard to the feature importance V_j, so that the features with higher V_j are more likely to be attended to than those with lower V_j.

To calculate the particles' R_t likelihood, i.e., the immediate reward that is output from DQN's given state, first, the initialized particle state ω_i is normalized as follows:

$$\hat{\omega}_i^j = \frac{\omega_i^j}{\sum_{j=1}^{D} \omega_i^j} \tag{8}$$

After updating the state of each particle as described above, we calculate the error between the predicted particle state Q-value $(Q(s_t, \hat{\omega}_i))$ and the R_t, which is returned from DQN as follows:

$$\epsilon_i = (R_t - Q(s_t, \hat{\omega}_i))^2 \tag{9}$$

where $\hat{\omega}_i$ is the normalized particle.

In the TDA process, rewards are determined based on normalized particles. A good particle state has a stronger predictive ability $Q(s_t, \hat{\omega}_i)$ for the target reward R_t. The main objective in this context is to identify the particle state that achieves the highest accuracy among a given set of particles, with the aim of minimizing the error in (Equation (9)). By minimizing the squared error, the algorithm seeks to find the particle state that closely aligns with the desired target reward, which is used in the next time step.

Particles are updated based on the likelihood of the immediate reward R_t, which is proportional to the following error value, s.t.,

$$P(R_t|\omega_i) = exp(-(\epsilon_i - \min(\epsilon_1, \epsilon_2, \ldots, \epsilon_m))) \tag{10}$$

Once the likelihoods are calculated, $p(\omega_i)$ are found as follows:

$$p(\omega_i) = \frac{P(R_t|\omega_i)}{\sum_{i=1}^{m} P(R_t|\omega_i)} \tag{11}$$

Using $p(\omega_i)$, the particles are then re-sampled with replacement, and the posterior distribution is updated for the next step.

Finally, the attention vector $\mathbf{A}$ is reset with the normalized mean of the particle states as in

$$A_j = \frac{\bar{\omega}^j}{\sum_{j=1}^{D} \bar{\omega}^j} \tag{12}$$

where $\bar{\omega}^j$ is given by

$$\bar{\omega}^j = \frac{1}{m} \sum_{i=1}^{m} \omega_i^j \tag{13}$$

4.3. Bottom-Up Attention Refinement (BAR)

BAR enhances the focus by considering the saliency values of the essential features. It aims to improve agent performance by increasing the attention weights of the essential features

($A_j >= \theta$) (Equation (16)) with lower saliency values H_j. While bottom-up methods can be useful in identifying salient features, they are not solely effective in identifying task-related features since they can miss much crucial task-related information due to noise or complexity. As a result, our approach improves the learning ability of the agent while considering this attention refinement, allowing for improved capture of task-related information.

Traditional saliency prediction methods rely on low-level features like color, contrast, and texture [47,48], but they struggle to capture the full range of factors influencing visual saliency maps [49]. BAR utilizes the saliency attentive model (SAM) by Cornia et al. [49], which uses a convolutional long short-term memory to enhance saliency predictions iteratively.

We quantify the relevance score of FDP with respect to specific features within the CNN using LRP [50]. These relevance scores are used to select the specific feature map-related pixels. These pixels are determined by whose relevance scores are greater than the average value of the corresponding pixels' relevance scores.

BAR obtains the saliency value $H(x, y)$ corresponding to a specific pixel (x, y) from the saliency map ($\boldsymbol{H}$) that is generated by the SAM model [49].

To calculate the specific feature saliency value H_j according to the pixel information using the saliency map ($\boldsymbol{H}$), we first normalize the saliency map within the range of $[0, 1]$. Then, BAR calculates the saliency value of a specific feature map denoted as j by considering only pixels with average relevance scores higher than the average relevance of the corresponding feature map.

$$\mathbf{P_j} = I(x, y) \quad s.t. \quad \phi_j(x, y) > \bar{\phi}_j \tag{14}$$

Here, $\mathbf{P}$ corresponds to the pixels that have a higher relevance score than the average relevance value ($\bar{\phi}$) of that specific feature map:

$$H_j = \frac{1}{|P_j|} \sum_{x,y \in \mathbf{P_j}} H(x, y) \tag{15}$$

The refinement vector $\boldsymbol{W}$ is obtained by considering the feature saliency value H_j using

$$W_j = \begin{cases} 1 + e^{-\alpha H_j} & \text{if } A_j >= \theta \\ 1 & \text{if } A_j < \theta \end{cases} \tag{16}$$

where θ threshold is defined as the average value of the attention vector $\boldsymbol{A}$.

The attention vector $\boldsymbol{A}$ is refined by multiplying element-wise with the refinement vector $\boldsymbol{W}$ as $(W \odot A)$. Then, each refined element is replicated to align with the dimensions of a feature map, ensuring that the same attentional value is applied uniformly across all spatial locations in the feature map $(W \odot A)^{\Uparrow}$. Finally, this process re-weights the feature maps, amplifying feature maps with attentional weights as (Equation (17)):

$$\boldsymbol{F}^{'} = \boldsymbol{F} \odot (\boldsymbol{W} \odot \boldsymbol{A})^{\Uparrow} \tag{17}$$

Here $\odot$ corresponds to the Hadamard product, and $\Uparrow$ shows the upscaling the refined attention vector by replication.

4.4. Layer-Wise Relevance Propagation (LRP)-Based Transfer Learning

AADQN employs a transfer learning scheme to enhance the agent's flexibility by reducing features and improving adaptation to noise and complexity. This scheme is based on LRP by Saraee et al. [50].

LRP propagates the output of the network backward through its CNN layers, assigning relevance scores to each neuron in each layer based on its contribution to the output, as illustrated in Figure 4, according to the LRP rule as in [51], which satisfies

$$\sum_i \phi_l^{i \leftarrow j} = \phi_{l+1}^j \tag{18}$$

where ϕ_l^i is the relevance of neuron i at layer l, and ϕ_{l+1}^j is the relevance of neuron j at the next layer $l+1$. According to the LRP method, the relevance value of the feature map is redistributed in the lower layers, and $\phi_l^{i \leftarrow j}$ is defined as the share of ϕ_{l+1}^j to neuron i in the lower layer l. Back-propagation continues until relevance scores are extracted for all neurons, including the input layer.

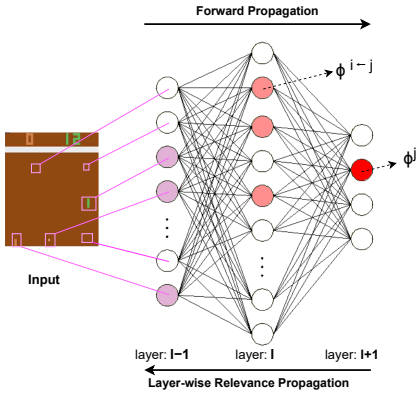


Figure 4. The backward process of the LRP in the propagation of features' importance relevance value across the middle layers. ϕ^j is the relevance value of neuron (j) at the layer $(l+1)$. According to the LRP method, the relevance value of the feature map is redistributed in the lower layers, and $\phi^{i \leftarrow j}$ is defined as the share of ϕ^j to neuron i in the lower layer l [50,51].

As described in Section 4.3 above, the relevance scores for the input layer are then used in BAR to highlight the FDP information in the input data that leads to the output of the feature maps.

When computing the final layer feature maps of the CNN, multiple internal CNN layers are employed. The vast number of parameters within the mid-layers of the network enables the capture of intermediate features at various levels of abstraction [52]. By harnessing these intermediate features, especially through dual attention mechanisms, the models' generalization capabilities can be enhanced [15]. To avoid unrecoverable information loss [53], AADQN freezes the unimportant neurons of the inner layers [53,54], which improves the model computing efficiency. Moreover, the model becomes most robust to noise and can improve flexibility in complex environments.

5. Experiment Results

In our work, we focus on DRL game agent's inefficiency and inflexibility problems. In this context, we investigate the importance of applying a visual attention mechanism to the performance of the game agent. To evaluate the performance of our model, we address the following questions:

- How does AADQN's game-play average score compare with DQN?
- How is AADQN's learning stability in comparison to DQN?

To compare the two algorithms, we have implemented our approach on the Atari 2600 environment in the OpenAI Gym environment [1]. From Atari 2600, we selected eight games (Pong, Wizard of Wor, SpaceInvaders, Breakout, Asterix, Seaquest, Beam Rider, Qbert) (Figure 5) with varying complexities and difficulty levels.

All experiments were run on a workstation with an Intel i7-8700 CPU and an Nvidia GTX-1080Ti GPU. The source code for the AADQN algorithm is available at the paper's GitHub page, https://github.com/celikcan-cglab/AADQN, accessed on 20 October 2023.

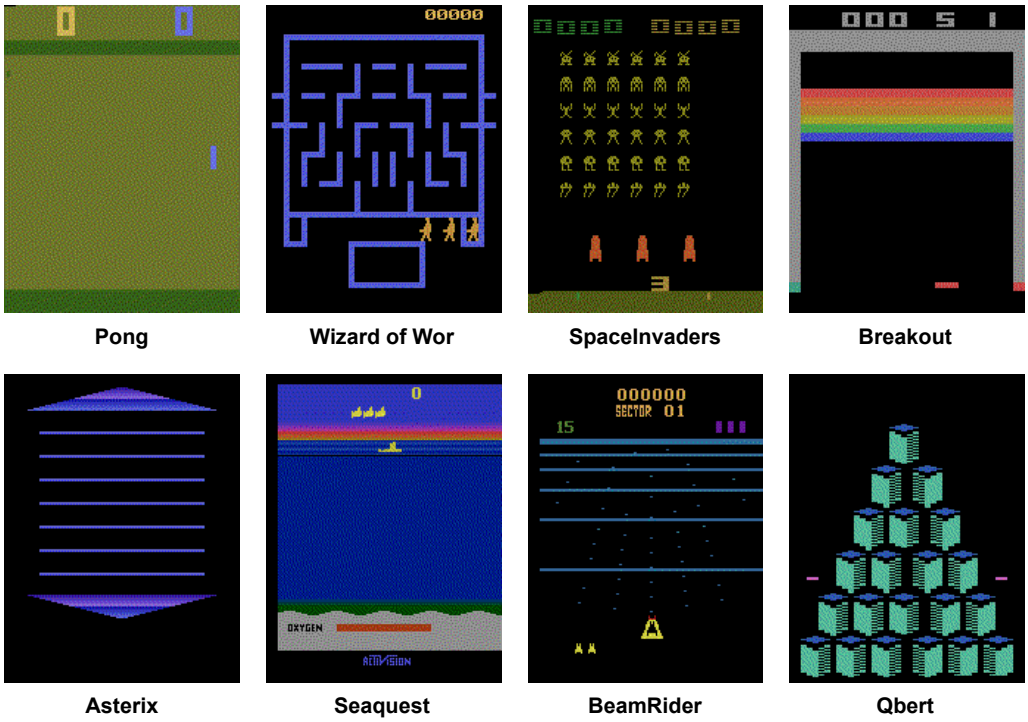


Figure 5. Pong, Wizard of Wor, SpaceInvaders, Breakout, Asterix, Seaquest, Beam Rider, Qbert.

5.1. Comparison with Baseline DQN

In this section, we provide a comparative analysis with respect to DQN in terms of game average score and time step. The time-step analysis includes the number of steps required to achieve a certain average score performance threshold. Table 1 provides the comparison of training the AADQN and DQN benchmark agents in average scores achieved on the eight tasks at the same final time spent for each task, and Figure 6 illustrates the progress of the two agents in terms of achieved scores until the final time step.

In two relatively simpler games, Pong and Breakout, we have observed an increasing average score for both algorithms with increasing time steps. Furthermore, AADQN outperformed the DQN algorithm throughout the entire process. Until 25×10^5 time steps, both algorithms performed similarly, but after this point, AADQN started to outperform DQN. This is likely due to the fact that in simple environments, learning is easier for both networks, and the game's high-level task is easier to learn due to the simplicity of these game environments.

In the other more complex games, we have observed a better game score performance of AADQN after a certain point of game experience. The difference between AADQN and DQN grows with increasing time steps of learning. For example, in the Wizard of Wor game environment, DQN shows a better performance than our approach until 30×10^6 time steps. This is because particles in the AADQN algorithm in the initial steps were not yet different from random states. However, after (30×10^6) steps, AADQN has a better performance than the DQN method throughout the entire experiment. This clearly shows that when the particles are close to the desired state, which represents a target configuration of the features map, AADQN reaches the optimal policy earlier than DQN. We have observed similar behavior in other complex game environments (SpaceInvaders, Seaquest, Beamrider, and Qbert).

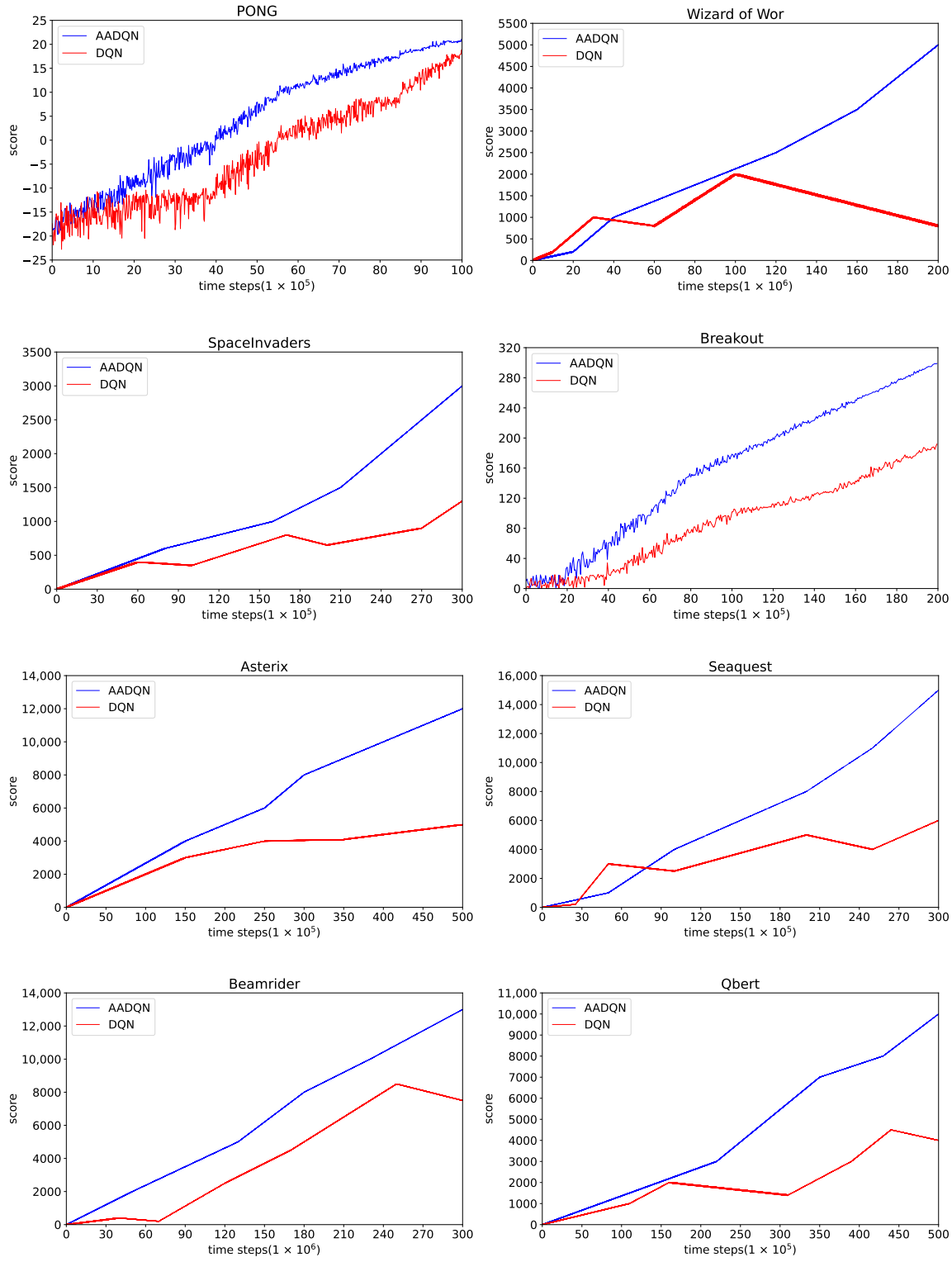


Figure 6. AADQN and DQN data efficiency comparison on eight Atari games. The x-axis shows the total number of training time steps. The y-axis shows the average score.

Table 1. Average score and improvement comparison.

Game	DQN Avg. Score	AADQN Avg. Score	Improvement by AADQN (%)	Time Step
Pong	18.88	20.64	9.32%	1×10^7
Wizard of Wor	791.51	5004.19	532.23%	2×10^8
SpaceInvaders	1299.33	2999.47	130.84%	3×10^7
Breakout	192.13	299.85	56.06%	2×10^6
Asterix	5021.27	11,999.97	138.98%	5×10^7
Seaquest	6001.04	15,000.1	149.95%	3×10^7
Beamrider	7503.97	12,999.49	73.23%	3×10^8
Qbert	3997.46	9999.47	150.14%	5×10^7
average	3103.19	7290.39	134.93%	8×10^7

The decrease in score performance of the baseline DQN method during learning is also noteworthy in these complex environments, whereas the AADQN score has an increasing trend throughout the experiment. For example, in the Wizard of Wor game, at different learning steps (e.g., 30 and 100×10^6), DQN's score actually decreased. These complex environments, including Wizard of Wor, can be non-stationary, meaning the optimal policy may change over time. DQN assumes a stationary environment, and if this assumption is violated, the algorithm may struggle to adapt, leading to a decrease in performance. Moreover, DQN faces a challenge in exploration-exploitation trade-offs. In some stages of training, the agent may prioritize exploration and try different actions to learn about the environment. This exploration can lead to a decrease in the average score. If the exploration strategy is not well-tuned, the agent may not explore enough to discover optimal policies.

Overall, these findings provide empirical evidence supporting the claim of faster learning in AADQN with respect to DQN.

5.2. Comparison with Other Algorithms

In this section, we extend the performance comparison of AADQN to other state-of-the-art algorithms in addition to DQN. The results are given in Table 2, where we provide the final state of each analyzed algorithm for the eight games under consideration at the same specific time step to ensure an equitable comparison.

The results reveal that there are specific environments where AADQN falls slightly behind the other state-of-the-art approaches. For instance, the asynchronous actor-learner architecture of A3C [22], which utilizes parallel agents, achieves a higher average score in the Wizard of Wor game. A3C is recognized for its ability to efficiently explore diverse actions and policies in intricate environments, making it valuable in the "Wizard of Wor" game. Similarly, the dueling network architecture (Dueling DQN) by Wang et al. [10], which separates value and advantage function estimation for Q-value computation, exhibits better performance in the Qbert game. This separation allows Dueling DQN to excel in situations where distinguishing between the value and advantage of different actions is critical, such as in the Qbert game. Nonetheless, in other games, AADQN outperforms these methods. Overall, we can conclude that AADQN exhibits relatively better performance in complex game environments.

Table 2. Average scores comparison across eight games.

Game	Random Play [2] Avg. Score	A3C [22] Avg. Score	Dueling DQN [10] Avg. Score	DQN Avg. Score	AADQN Avg. Score	Time Step
Pong	−20.7	−15	14.5	18.88	**20.64**	1×10^7
Wizard of wor	563.5	**6000**	5666	791.51	5004.19	2×10^8
SpaceInvaders	148	480	864	1299.33	**2999.47**	3×10^7
Breakout	1.7	30.0	164.5	192.13	**299.85**	2×10^7
Asterix	210	2224	2857	5021.27	**11,999.97**	5×10^7
Seaquest	68.4	850	9000	6001.04	**150,001.1**	3×10^7
BeamRiders	363.9	7600	**15,112**	7503.97	12,999.49	3×10^8
Qbert	163.9	8668	**15,714**	3997.46	9999.47	5×10^7

5.3. Comparison of Learning Stability

Figure 7 demonstrates the fluctuation rate in the average game scores of the agent during the start-up and convergence phases for the eight games under consideration. For the sake of a clear comparison between DQN and AADQN, the two plots are overlaid in each sub-figure so that they are shown with the same scale but different ranges (e.g., in the Asterix game, the AADQN scores are in the range [1550–1610], whereas DQN is the range [1150–1210]).

It is seen that the AADQN and DQN methods have different start-up and convergence phase fluctuation behaviors. Significant fluctuations in the start-up phase for both algorithms indicate instability or inconsistent learning. In the convergence phase, AADQN's average score stabilizes and mitigates the fluctuation compared to the DQN algorithm. This point indicates that the agent has converged to a relatively optimal policy and is consistently performing well. As shown in Figure 6, in the Pong game, AADQN becomes stabilized sooner than the baseline DQN. Similar stability behavior in the start-up and convergence phases are observed in all games, regardless of game complexity.

5.4. Algorithmic Efficiency

The time required for training DQN can vary significantly depending on the hardware. So, in this section, we explain the computational efficiency of AADQN in terms of total time steps in order to provide a fair comparison with the baseline. Our framework is based on DQN together with particle-filter-based top-down and saliency-based bottom-up attention. So, the computation overhead on top of DQN comes from these attention mechanisms. The particles can be sampled in parallel, which adds a single particle computation to the system. However, the attention mechanism decreases the total time steps required to obtain the same average score with a focus on significant feature maps. Similarly, the saliency maps can be extracted parallel to the DQN forward pass. LRP relevance scores can be calculated parallel to the DQN backward pass, which requires no additional time steps. In Figure 6, the time-step comparisons of DQN and AADQN are given.

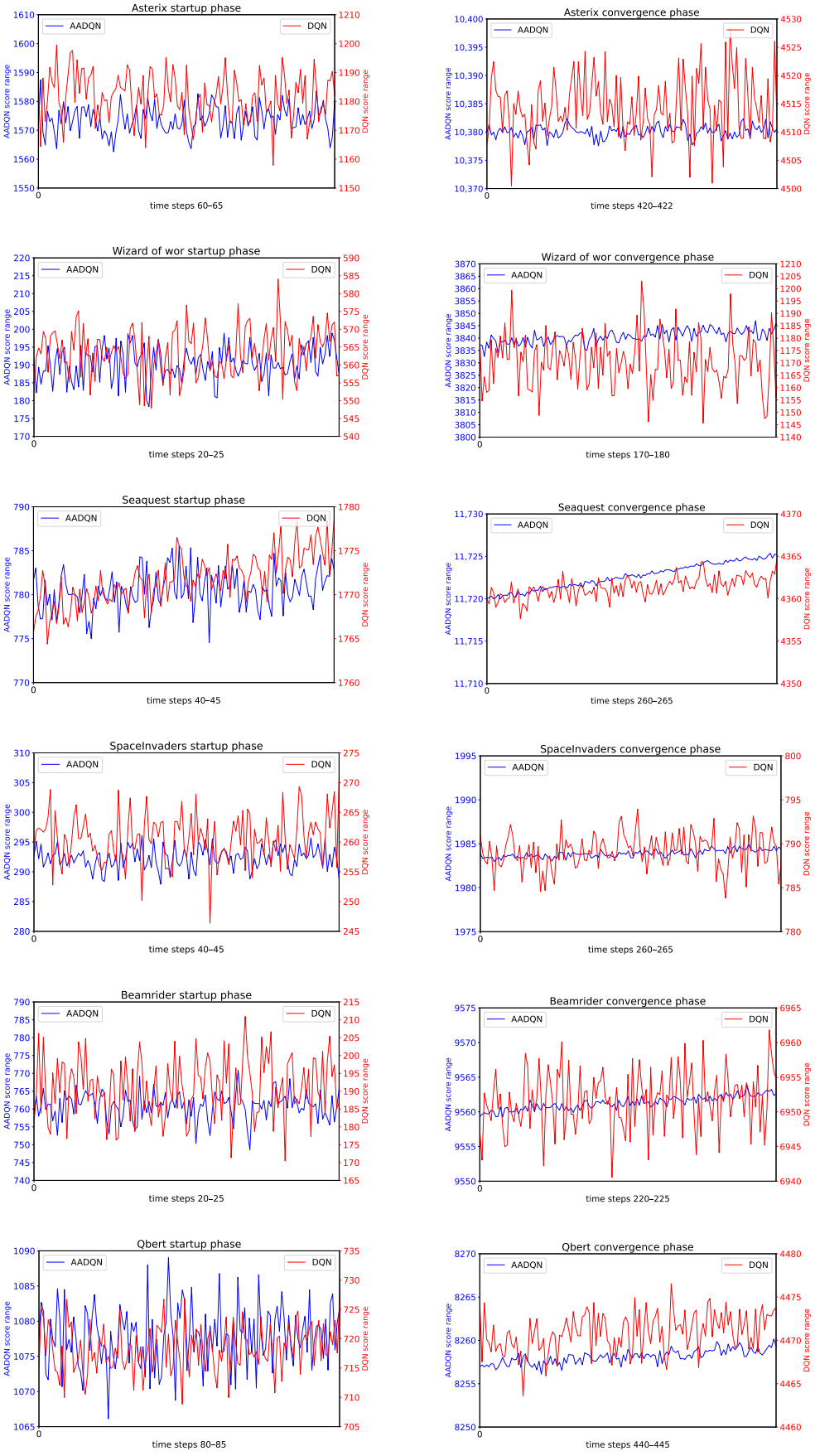


Figure 7. Comparison of the start-up (**left**) and convergence (**right**) phases between AADQN and DQN.

6. Conclusions

Motivated by the lack of efficiency and flexibility of standard DQN, this study proposed a new attention-augmented DQN model (AADQN), which integrates bottom-up and top-down attention mechanisms, surpasses DQN performance in shorter steps, and demonstrates adaptability in intricate environments.

While the bottom-up method identifies basic lower-level features, it may miss task-related information due to noise or complexity by itself. The particle-filter-based top-down attention mechanism addressed this limitation and enhanced DQN's performance. Additionally, by utilizing LRP, unimportant neurons on CNN were identified and frozen, leading to an improvement in DQN robustness.

Leveraging a combination of particle filter and gradient descent impacts the determination of gradient direction in a way that causes time-step basis acceleration during the training process and improves the average score by about 134.93% across the eight game environments. The particle-filter-based attention reduces fluctuation and provides more reliable convergence, together with alleviating the necessity for extensive training data that are typically required by baseline DQN.

When comparing with other studies, it is worth noting that in certain complex games like 'Wizard of Wor' and 'Qbert', AADQN was outperformed by A3C and Dueling DQN algorithms, respectively. However, if we exclude these two games from our analysis, we can generally observe that AADQN performs relatively well in complex game environments.

In future research, it would be valuable to conduct more extensive comparisons with other methods, such as Gorila and Double DQN, and provide a more comprehensive analysis involving other game environments in varying complexity. Furthermore, future research in this field could explore the impact of clustering the extracted feature map of a CNN to reduce the dimensionality of the attention vector. This clustering technique may have the potential to enable game agents to attend to multiple areas simultaneously, thereby enhancing their ability to focus on multiple regions of interest at once. Furthermore, inhibitory/negative signals of LRP can also be taken into account, which might further enhance its contribution by refining its functionality.

Author Contributions: Methodology, E.U. and U.C.; Software, E.U., B.Ç. and U.C.; Validation, E.U. and B.Ç; Writing—original draft, E.U., T.C. and U.C.; Writing—review & editing, T.C., B.Ç. and U.C.; Supervision, T.C. and U.C. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Data Availability Statement: The source of all datasets and models used in this study is given in the manuscript.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Bellemare, M.G.; Naddaf, Y.; Veness, J.; Bowling, M. The arcade learning environment: An evaluation platform for general agents. *J. Artif. Intell. Res.* **2013**, *47*, 253–279. [CrossRef]
2. Mnih, V.; Kavukcuoglu, K.; Silver, D.; Rusu, A.A.; Veness, J.; Bellemare, M.G.; Graves, A.; Riedmiller, M.; Fidjeland, A.K.; Ostrovski, G.; et al. Human-level control through deep reinforcement learning. *Nature* **2015**, *518*, 529–533. [CrossRef] [PubMed]
3. Sutton, R.S.; Barto, A.G. *Reinforcement Learning: An Introduction*; MIT Press: Cambridge, MA, USA, 2018.
4. Watkins, C.J.; Dayan, P. Q-learning. *Mach. Learn.* **1992**, *8*, 279–292. [CrossRef]
5. Mnih, V.; Kavukcuoglu, K.; Silver, D.; Graves, A.; Antonoglou, I.; Wierstra, D.; Riedmiller, M. Playing atari with deep reinforcement learning. *arXiv* **2013**, arXiv:1312.5602.
6. Hessel, M.; Modayil, J.; Van Hasselt, H.; Schaul, T.; Ostrovski, G.; Dabney, W.; Horgan, D.; Piot, B.; Azar, M.; Silver, D. Rainbow: Combining improvements in deep reinforcement learning. In Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, New Orleans, LA, USA, 2–7 February 2018.
7. Dabney, W.; Rowland, M.; Bellemare, M.; Munos, R. Distributional reinforcement learning with quantile regression. In Proceedings of the AAAI Conference on Artificial Intelligence, New Orleans, LA, USA, 2–7 February 2018; Volume 32.

8. Nair, A.; Srinivasan, P.; Blackwell, S.; Alcicek, C.; Fearon, R.; De Maria, A.; Panneershelvam, V.; Suleyman, M.; Beattie, C.; Petersen, S.; et al. Massively parallel methods for deep reinforcement learning. *arXiv* **2015**, arXiv:1507.04296.
9. Haarnoja, T.; Tang, H.; Abbeel, P.; Levine, S. Reinforcement learning with deep energy-based policies. In Proceedings of the International Conference on Machine Learning, PMLR, Sydney, Australia, 6–11 August 2017; pp. 1352–1361.
10. Wang, Z.; Schaul, T.; Hessel, M.; Hasselt, H.; Lanctot, M.; Freitas, N. Dueling network architectures for deep reinforcement learning. In Proceedings of the International Conference on Machine Learning, PMLR, New York, NY, USA, 19–24 June 2016; pp. 1995–2003.
11. Mnih, V.; Heess, N.; Graves, A.; Kavukcuoglu, K. Recurrent models of visual attention. In Proceedings of the Advances in Neural Information Processing Systems, (NIPS 2014), Montreal, QC, Canada, 8–13 December 2014; Volume 27.
12. Sorokin, I.; Seleznev, A.; Pavlov, M.; Fedorov, A.; Ignateva, A. Deep attention recurrent Q-network. *arXiv* **2015**, arXiv:1512.01693.
13. Manchin, A.; Abbasnejad, E.; Hengel, A.v.d. Reinforcement learning with attention that works: A self-supervised approach. In Proceedings of the International Conference on Neural Information Processing, Sydney, Australia, 12–15 December 2019; Springer: Cham, Switzerland, 2019; pp. 223–230.
14. Bramlage, L.; Cortese, A. Generalized attention-weighted reinforcement learning. *Neural Netw.* **2022**, *145*, 10–21. [CrossRef]
15. Lu, S.; Ding, Y.; Liu, M.; Yin, Z.; Yin, L.; Zheng, W. Multiscale feature extraction and fusion of image and text in VQA. *Int. J. Comput. Intell. Syst.* **2023**, *16*, 54. [CrossRef]
16. Lu, S.; Liu, M.; Yin, L.; Yin, Z.; Liu, X.; Zheng, W. The multi-modal fusion in visual question answering: A review of attention mechanisms. *PeerJ Comput. Sci.* **2023**, *9*, e1400. [CrossRef]
17. Zeng, G.; Zhang, Y.; Zhou, Y.; Yang, X.; Jiang, N.; Zhao, G.; Wang, W.; Yin, X.C. Beyond OCR+ VQA: Towards end-to-end reading and reasoning for robust and accurate textvqa. *Pattern Recognit.* **2023**, *138*, 109337. [CrossRef]
18. Ma, Z.; Zheng, W.; Chen, X.; Yin, L. Joint embedding VQA model based on dynamic word vector. *PeerJ Comput. Sci.* **2021**, *7*, e353. [CrossRef] [PubMed]
19. Zhang, X.; Zhang, F.; Xu, C. Reducing Vision-Answer biases for Multiple-choice VQA. *IEEE Trans. Image Process.* **2023**, *32*, 4621–4634. [CrossRef] [PubMed]
20. Zheng, W.; Yin, L.; Chen, X.; Ma, Z.; Liu, S.; Yang, B. Knowledge base graph embedding module design for Visual question answering model. *Pattern Recognit.* **2021**, *120*, 108153. [CrossRef]
21. Schaul, T.; Quan, J.; Antonoglou, I.; Silver, D. Prioritized experience replay. *arXiv* **2015**, arXiv:1511.05952.
22. Mnih, V.; Badia, A.P.; Mirza, M.; Graves, A.; Lillicrap, T.; Harley, T.; Silver, D.; Kavukcuoglu, K. Asynchronous methods for deep reinforcement learning. In Proceedings of the International Conference on Machine Learning, PMLR, New York, NY, USA, 19–24 June 2016; pp. 1928–1937.
23. Bellemare, M.G.; Dabney, W.; Munos, R. A distributional perspective on reinforcement learning. In Proceedings of the International Conference on Machine Learning, PMLR, Sydney, Australia, 6–11 August 2017; pp. 449–458.
24. Maulana, M.R.; Lee, W.S. Ensemble and auxiliary tasks for data-efficient deep reinforcement learning. In Proceedings of the Joint European Conference on Machine Learning and Knowledge Discovery in Databases, Bilbao, Spain, 13–17 September 2021; Springer: Cham, Switzerland, 2021; pp. 122–138.
25. Mousavi, S.; Schukat, M.; Howley, E.; Borji, A.; Mozayani, N. Learning to predict where to look in interactive environments using deep recurrent q-learning. *arXiv* **2016**, arXiv:1612.05753.
26. Zhang, R.; Liu, Z.; Zhang, L.; Whritner, J.A.; Muller, K.S.; Hayhoe, M.M.; Ballard, D.H. Agil: Learning attention from human for visuomotor tasks. In Proceedings of the European Conference on Computer Vision (ECCV), Munich, Germany, 8–14 September 2018; pp. 663–679.
27. Mott, A.; Zoran, D.; Chrzanowski, M.; Wierstra, D.; Jimenez Rezende, D. Towards interpretable reinforcement learning using attention augmented agents. In Proceedings of the Advances in Neural Information Processing Systems, (NeurIPS 2019), Vancouver, BC, Canada, 8–14 December 2019; Volume 32.
28. Hausknecht, M.; Stone, P. Deep recurrent q-learning for partially observable mdps. In Proceedings of the 2015 AAAI Fall Symposium Series, Arlington, VA, USA, 12–14 November 2015.
29. Itti, L.; Koch, C.; Niebur, E. A model of saliency-based visual attention for rapid scene analysis. *IEEE Trans. Pattern Anal. Mach. Intell.* **1998**, *20*, 1254–1259. [CrossRef]
30. Yuezhang, L.; Zhang, R.; Ballard, D.H. An initial attempt of combining visual selective attention with deep reinforcement learning. *arXiv* **2018**, arXiv:1811.04407.
31. Greydanus, S.; Koul, A.; Dodge, J.; Fern, A. Visualizing and understanding atari agents. In Proceedings of the International Conference on Machine Learning, PMLR, Stockholm, Sweden, 10–15 July 2018; pp. 1792–1801.
32. Mirowski, P.; Pascanu, R.; Viola, F.; Soyer, H.; Ballard, A.J.; Banino, A.; Denil, M.; Goroshin, R.; Sifre, L.; Kavukcuoglu, K.; et al. Learning to navigate in complex environments. *arXiv* **2016**, arXiv:1611.03673.
33. Silver, D.; Hasselt, H.; Hessel, M.; Schaul, T.; Guez, A.; Harley, T.; Dulac-Arnold, G.; Reichert, D.; Rabinowitz, N.; Barreto, A.; et al. The predictron: End-to-end learning and planning. In Proceedings of the International Conference on Machine Learning, PMLR, Sydney, Australia, 6–11 August 2017; pp. 3191–3199.
34. Li, Y. Deep reinforcement learning: An overview. *arXiv* **2017**, arXiv:1701.07274.
35. Wang, J.; Jiang, T.; Cui, Z.; Cao, Z. Filter pruning with a feature map entropy importance criterion for convolution neural networks compressing. *Neurocomputing* **2021**, *461*, 41–54. [CrossRef]

36. McHugh, M.L. The chi-square test of independence. *Biochem. Medica* **2013**, *23*, 143–149. [CrossRef] [PubMed]
37. Li, H.; Kadav, A.; Durdanovic, I.; Samet, H.; Graf, H.P. Pruning filters for efficient convnets. *arXiv* **2016**, arXiv:1608.08710.
38. Lin, M.; Ji, R.; Wang, Y.; Zhang, Y.; Zhang, B.; Tian, Y.; Shao, L. Hrank: Filter pruning using high-rank feature map. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, Seattle, WA, USA, 13–19 June 2020; pp. 1529–1538.
39. Hur, C.; Kang, S. Entropy-based pruning method for convolutional neural networks. *J. Supercomput.* **2019**, *75*, 2950–2963. [CrossRef]
40. Soltani, M.; Wu, S.; Ding, J.; Ravier, R.; Tarokh, V. On the information of feature maps and pruning of deep neural networks. In Proceedings of the 2020 25th International Conference on Pattern Recognition (ICPR), Milan, Italy, 10–15 January 2021; pp. 6988–6995.
41. Kalman, R.E. A new approach to linear filtering and prediction problems. *J. Basic Eng. Mar.* **1960**, *82*, 35–45. [CrossRef]
42. Kalman, R.E.; Bucy, R.S. New results in linear filtering and prediction theory. *J. Basic Eng. Mar.* **1961**, *83*, 95–108. [CrossRef]
43. Elfring, J.; Torta, E.; van de Molengraft, R. Particle filters: A hands-on tutorial. *Sensors* **2021**, *21*, 438. [CrossRef] [PubMed]
44. Arulampalam, M.S.; Maskell, S.; Gordon, N.; Clapp, T. A tutorial on particle filters for online nonlinear/non-Gaussian Bayesian tracking. *IEEE Trans. Signal Process.* **2002**, *50*, 174–188. [CrossRef]
45. Kamsing, P.; Torteeka, P.; Yooyen, S. An enhanced learning algorithm with a particle filter-based gradient descent optimizer method. *Neural Comput. Appl.* **2020**, *32*, 12789–12800. [CrossRef]
46. Grest, D.; Krüger, V. Gradient-enhanced particle filter for vision-based motion capture. In Proceedings of the Workshop on Human Motion, Rio de Janeiro, Brazil, 20 October 2007; Springer: Berlin/Heidelberg, Germany, 2007; pp. 28–41.
47. Goferman, S.; Zelnik-Manor, L.; Tal, A. Context-aware saliency detection. *IEEE Trans. Pattern Anal. Mach. Intell.* **2011**, *34*, 1915–1926. [CrossRef]
48. Zhang, J.; Sclaroff, S. Saliency detection: A boolean map approach. In Proceedings of the IEEE International Conference on Computer Vision, Sydney, Australia, 2–8 December 2013; pp. 153–160.
49. Cornia, M.; Baraldi, L.; Serra, G.; Cucchiara, R. Predicting human eye fixations via an lstm-based saliency attentive model. *IEEE Trans. Image Process.* **2018**, *27*, 5142–5154. [CrossRef]
50. Bach, S.; Binder, A.; Montavon, G.; Klauschen, F.; Müller, K.R.; Samek, W. On pixel-wise explanations for non-linear classifier decisions by layer-wise relevance propagation. *PLoS ONE* **2015**, *10*, e0130140. [CrossRef]
51. Yeom, S.K.; Seegerer, P.; Lapuschkin, S.; Binder, A.; Wiedemann, S.; Müller, K.R.; Samek, W. Pruning by explaining: A novel criterion for deep neural network pruning. *Pattern Recognit.* **2021**, *115*, 107899. [CrossRef]
52. Saraee, E.; Jalal, M.; Betke, M. Visual complexity analysis using deep intermediate-layer features. *Comput. Vis. Image Underst.* **2020**, *195*, 102949. [CrossRef]
53. He, Y.; Dong, X.; Kang, G.; Fu, Y.; Yan, C.; Yang, Y. Asymptotic soft filter pruning for deep convolutional neural networks. *IEEE Trans. Cybern.* **2019**, *50*, 3594–3604. [CrossRef]
54. Livne, D.; Cohen, K. Pops: Policy pruning and shrinking for deep reinforcement learning. *IEEE J. Sel. Top. Signal Process.* **2020**, *14*, 789–801. [CrossRef]

Disclaimer/Publisher's Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.