mathematics

MDPI

Article

The Application of Residual Connection-Based State Normalization Method in GAIL

Yanning Ge [1,†], Tao Huang [2,†], Xiaoding Wang [1], Guolong Zheng [2,*] and Xu Yang [2,*]

[1] College of Computer and Cyber Security, Fujian Normal University, Fuzhou 350117, China; ged_mail@163.com (Y.G.); wangdin1982@fjnu.edu.cn (X.W.)
[2] College of Computer and Control Engineering, Minjiang University, Fuzhou 350108, China; huangtao@mju.edu.cn
[*] Correspondence: gzheng@mju.edu.cn (G.Z.); xu.yang@mju.edu.cn (X.Y.)
[†] These authors contributed equally to this work.

Abstract: In the domain of reinforcement learning (RL), deriving efficacious state representations and maintaining algorithmic stability are crucial for optimal agent performance. However, the inherent dynamism of state representations often complicates the normalization procedure. To overcome these challenges, we present an innovative RL framework that integrates state normalization techniques with residual connections and incorporates attention mechanisms into generative adversarial imitation learning (GAIL). This combination not only enhances the expressive capability of state representations, thereby improving the agent's accuracy in state recognition, but also significantly mitigates the common issues of gradient dissipation and explosion. Compared to traditional RL algorithms, GAIL combined with the residual connection-based state normalization method empowers the agent to markedly reduce the exploration duration such that feedback concerning rewards in the current state can be provided in real time. Empirical evaluations demonstrate the superior efficacy of this methodology across various RL environments.

Keywords: reinforcement learning; generative adversarial imitation learning (GAIL); residual connection; state normalization

MSC: 68T01

check for updates

Citation: Ge, Y.; Huang, T.; Wang, X.; Zheng, G.; Yang, X. The Application of Residual Connection-Based State Normalization Method in GAIL. *Mathematics* **2024**, *12*, 214. https://doi.org/10.3390/math12020214

Academic Editors: Hong Qu, Malu Zhang and Mingsheng Fu

Received: 31 October 2023
Revised: 15 December 2023
Accepted: 4 January 2024
Published: 9 January 2024

1. Introduction

Reinforcement learning (RL) [1], a significant branch of machine learning, emphasizes training agents to learn optimal strategies through environmental interactions. The primary goal of this approach is the maximization of long-term cumulative rewards. Recent advancements in RL methodologies have resulted in remarkable successes across various domains, such as gaming, robotic control, and autonomous driving [2–4]. These achievements underscore the versatility and efficacy of RL in solving complex, dynamic problems.

In RL, the term "agent" refers to a machine's ability to perceive and interact with its surroundings, distinguishing it from the "model" concept in supervised learning. An RL agent bases its decisions on sensory inputs and is motivated by rewards that affect the environmental state [5]. This approach contrasts with supervised learning, which relies on labeled data. Instead, RL uses a "reward function" as a guiding principle in its learning process. However, this method introduces distinct challenges. RL models are particularly sensitive to the configuration of the reward function, with minor changes potentially leading to significant variations in the resulting policies [6,7].

Imitation learning [8] has been implemented to address these RL challenges. This approach allows agents to emulate expert decision-making strategies through demonstration data. By adopting this method, the need for manual configuration of the reward function is circumvented, consequently reducing the agent's dependence on specific reward function configurations.

Imitation learning can be categorized into three main types:

- Behavior cloning (BC): This method boasts simplicity and rapid implementation, yet it necessitates extensive data support and is susceptible to cumulative errors [9].
- Inverse reinforcement learning (IRL): While it effectively tackles the problem of cumulative errors, IRL incurs significant computational demands and has constrained practical applicability [10].
- Generative adversarial imitation learning (GAIL): This method excels in learning policies that adapt to states beyond the scope of expert demonstrations, showcasing superior generalization abilities [11]. This feature sets it apart from behavior cloning, which is limited to mimicking strategies directly observable in expert demonstrations, resulting in narrower adaptability.

GAIL effectively reduces the agent's reliance on the reward function. However, its learning process can become unstable during exploration, potentially slowing down the learning rate, especially in situations with substantial variance in states or observations. The optimization algorithm within GAIL requires additional time to adapt to these diverse value ranges. As a result, this might lead to decreased learning efficiency, compromised training effectiveness, and a general downturn in performance.

Addressing these issues, our approach integrates state normalization with residual connections. In standard neural network training, challenges like gradient vanishing and exploding often present substantial obstacles. Implementing residual connections significantly alleviates these issues, thereby improving the speed and stability of the leaning process. Furthermore, state normalization helps the model achieve a deeper understanding of environmental states, consequently enhancing learning efficiency.

Furthermore, we enhance the model's representational capacity by incorporating attention mechanisms. In diverse reinforcement learning scenarios, the importance of states and actions varies across time steps. The inclusion of attention mechanisms allows our model to adaptively focus on relevant information from these varying periods. This feature enables the model to accurately understand and react to dynamic environmental changes, thus improving its performance in complex RL tasks.

Based on the aforementioned discussion, the main contributions of this paper can be summarized as follows:

- The introduction of a novel normalization method combined with residual connections, designed to effectively address the challenges of gradient vanishing and exploding in reinforcement learning models.
- The incorporation of attention mechanisms into the neural network to refine internal focus during the training process, enabling the model to selectively emphasize state feature variables that are critical for current decision making.

2. Related Work

Normalization techniques are crucial in training deep neural networks [12–14]. They significantly enhance training efficiency and performance by standardizing input features, especially when employing first-order optimization algorithms like stochastic gradient descent (SGD) [15,16]. The impact of normalization is particularly pronounced in this context. It not only accelerates the learning speed but also improves the overall stability and performance of the model. This allows networks to excel even in uniformly optimized environments. Among these techniques, batch normalization (BN) is essential in deep network training [17,18]. BN adjusts intermediate feature maps based on statistical data from small-batch samples. The success of BN has led to the development of other normalization methods, such as layer normalization [19,20] and spectral normalization [21,22], providing a diverse range of options for different learning environments.

In deep reinforcement learning, the utility of these normalization techniques is constrained, owing to the nature of online learning. A primary reason is that the input data frequently deviate from the principles independence and identical distribution. In the Section 4, we delve into the key factors that make BN inapplicable to reinforcement learning.

In addressing these issues, Bhatt A et al. [23] introduced a cross-normalization technique that combines transitions within and outside the policy. This approach not only optimizes stability but also reduces dependence on the target network. Additionally, Gogianu F et al. [24] demonstrated the efficacy of spectral normalization in controlling parameter updates in deep reinforcement learning, highlighting the necessity of focusing on neural network components and their learning dynamics. However, these methods overlook the pivotal role of the reward function in reinforcement learning training. To remedy this, our study proposes a dynamic normalization method that integrates residual connections within the GAIL framework. This innovation is aimed at addressing the previously neglected role of the reward function, with the goal of enhancing the model's generalization capabilities and training efficiency.

3. Preliminary

First, let us define a Markov decision process (MDP) [25], which is typically formulated as a sextuple, i.e., $\langle \mathcal{S}, \mathcal{A}, \mathcal{P}, r, \rho_0, \gamma \rangle$, where:

- $\mathcal{S}$ represents the set of all possible states;
- $\mathcal{A}$ is the set of all possible actions available to the agent (at each time step, the agent selects and executes a single action from this set);
- $\mathcal{P}(s_{t+1}|s_t, a_t)$ is the state transition function, which describes the probability of transitioning from state s_t to state s_{t+1} when action a_t is taken;
- $r(s, a)$ is the reward function, indicating the immediate reward that the agent receives given the current state and the chosen action;
- $\gamma \in [0, 1]$ is the discount factor, which is used to compute the present value of future rewards.

Next, we define the expert policy as π_E and the learner's policy as π. For policy π, the expected trajectory is expressed by the following formula:

$$\mathbb{E}_\pi[g(s,a)] \triangleq \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t g(s_t, a_t)\right] \tag{1}$$

Here, $s_0 \sim \rho_0$ denotes the initial state distribution, $a_t \sim \pi(a_t|s_t)$ represents the action chosen according to policy π, and g is a custom function that can be used to further describe state–action pairs [26].

Thus, the reinforcement learning process can be expressed as shown in Figure 1.

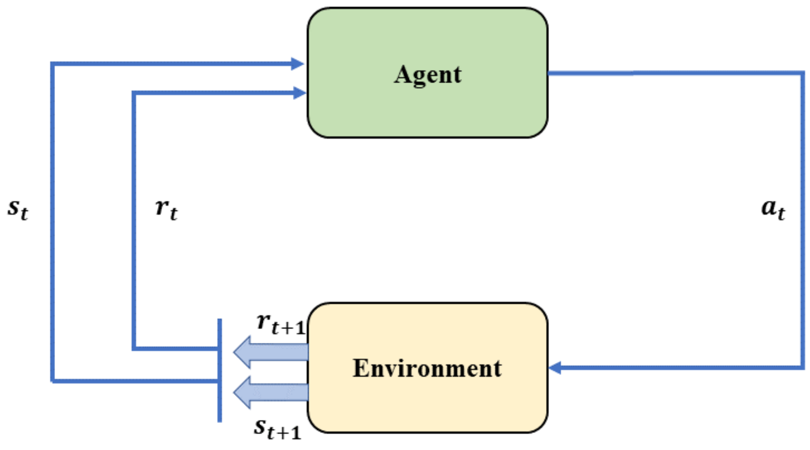

Figure 1. Reinforcement learning framework.

Residual connections: represent a substantial technological advancement in deep learning. Their fundamental principle is the introduction of a "shortcut path" or "skip connection" between neighboring layers in a network [27–29]. This architecture allows signals to skip one or more layers, transmitting directly to deeper layers. This method effectively addresses the problem of information decay, a frequent challenge in deep networks [30].

In a traditional feedforward network without residual connections, each layer is typically formulated as $y = H(x)$, where x represents the input, y is the output, and $H(x)$ denotes the operation performed by that layer. Conversely, in a residual network equipped with residual connections, the formulation of each layer is modified to $H(x) = F(x) + x$, where $F(x)$ signifies the residual mapping of the layer. This function is tasked with learning the necessary adjustments to the input (x) to yield results that more closely align with the anticipated output. The equation $F(x) = H(x) - x$ thus represents the residual or the difference between the input and the output. This structure allows the signal to bypass any number of layers, effectively preventing decay between layers.

The residual connection approach empowers the network to fully exploit the robust expressive capabilities inherent in deep abstraction while simultaneously addressing common challenges associated with training deep networks. Notably, it helps mitigate the issue of gradient vanishing. In the next section, we discuss how to combine residual connections with normalization methods.

GAIL: GAIL is a specialized form of imitation learning. Its fundamental concept revolves around aligning the learning policy with the expert policy as closely as possible in terms of the occupancy measure across all state–action pairs. To achieve this goal, the GAIL algorithm introduces a policy (π) and a discriminator (D). In this framework, the policy (π) plays a role similar to the generator in a generative adversarial network: for a specific state (s), it determines the corresponding action (a). On the other hand, the discriminator (D) takes the state–action pair (s, a) as input and outputs a value between 0 and 1, representing its judgment of whether the pair comes from an expert. To distinguish the data generated by the policy (π) from expert data, the task of the discriminator (D) is to make the output value for data from the expert as close to 0 as possible and the value for data from the policy (π) as close to 1 as possible.

Based on this objective, the loss function of D can be defined as:

$$L(\phi) = -E_{\rho_\pi}[\log D_\phi(s, a)] - E_{\rho_E}[\log(1 - D_\phi(s, a))] \tag{2}$$

Meanwhile, the goal of the policy (π) is to have its generated trajectories mistakenly recognized by D as expert trajectories. To achieve this, the output of D can be used as the reward function for the policy (π). Specifically, for sampled state–action pair (s, a), the reward ($r(s, a)$) can be set as $-\log D(s, a)$. Through continued adversarial training, the data distribution generated by the policy (π) gradually approaches the expert data distribution.

4. Methods

In RL, the state plays a pivotal role by offering the agent current insights about the environment, thereby facilitating informed decision making. The effectiveness and efficiency of a reinforcement learning algorithm are intrinsically linked to the quality of its state representation, as highlighted in [31]. A meticulously crafted state vector equips the agent with vital information regarding its surroundings, which is essential for executing suitable actions.

However, the original representation of states may encounter problems such as large differences in numerical ranges and masked feature variations. More critically, untreated state data can lead to instability in the learning process, such as gradient explosion or vanishing, which can affect learning performance [32]. To address these challenges, a strategy called "state normalization" is introduced. This is achieved by standardizing the state features to a fixed numerical range or giving them zero mean and unit variance. Such processing not only enhances the stability and convergence speed of learning but also improves the model's generalization capability.

Traditional single-normalization methods also have their limitations, such as normalizing only the output of individual layers, ignoring the mutual influence between layers in the network, or losing some key information in the normalization process. To overcome these issues, we consider incorporating "residual connections" into the normalization process. Residual connections allow information to flow across multiple layers of the network, thereby enhancing information flow and training stability. By combining the two, we can achieve efficient and stable network structures like a "residual normalization block", resulting in superior performance across various tasks.

When considering which normalization method to adopt, we attempted to incorporate batch normalization (BN) into the network to handle states. However, we found that the policy's performance deteriorated after applying BN. An analysis of the reasons is presented as follows:

- First, when the batch size is small, BN performs poorly. This is easily understood, as BN uses sample statistics within the batch to estimate the statistics of the entire dataset. With a small batch size, the estimation of statistics is inaccurate, leading to a decrease in training effectiveness. Even when we increased the batch size, the issue was not resolved.
- Secondly, by referring to relevant literature [17,33], we discovered that BN is challenging to apply effectively to deep reinforcement learning. Compared to supervised learning, in reinforcement learning, the agent needs to continuously interact with the environment to sample new data. These sampled data are unstable initially and gradually stabilize as training progresses. However, after policy improvement, the agent begins to explore new states, causing a change in the data distribution. At this point, the statistics computed by BN are no longer suitable for the new data, leading to policy degradation. In contrast, supervised learning can randomly sample from the training set, ensuring a stable data distribution and ensuring the effectiveness of BN.

In supervised learning:

1. Regardless of the network's performance, we always sample randomly from the training set, ensuring stable training data.
2. With stable data, BN tends to stabilize as well, computing fixed mean and variance.

However, in RL, the relationship between BN stability and data stability, as shown in Figure 2 is described as follows:

1. Training data are sampled by the agent interacting with the environment are initially unstable.
2. After some time, the data stabilize, and BN also stabilizes.
3. With policy improvement, the agent starts to explore new states, causing a change in the data distribution.
4. The statistics computed by BN are no longer suitable for the new data, leading to a decline in policy performance.
5. The decline in policy performance further to further unstable data, forming a vicious cycle.

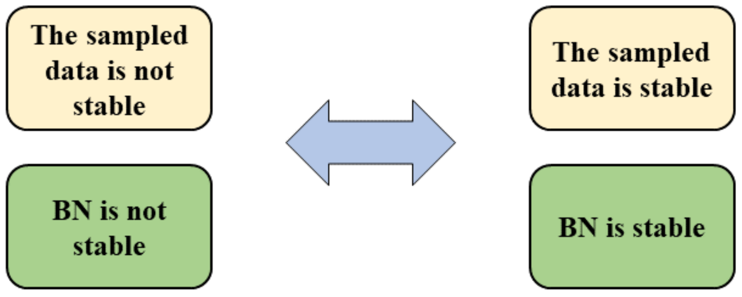

Figure 2. The logical relationship between BN stability and the adoption of data stability.

In summary, BN proves challenging to apply effectively in reinforcement learning training, necessitating other techniques to address the issue of data instability. Therefore, in this study, we did not directly employ BN but utilized a state normalization approach combined with residual connections.

First, let us explain how state normalization is performed. Here, we employ the "robust" Z-score standardization method [34]. Z-score standardization is a common data preprocessing technique used to adjust the scale of data such that their mean is 0 and their standard deviation is 1. Specifically, for each data point, we subtract the mean of the entire dataset, then divide it by the standard deviation of the dataset. However, in some cases, the original Z-score standardization may be affected by outliers. To address this issue, we use the "robust" Z-score standardization method, which utilizes the median of the data instead of the mean and the interquartile range (IQR, i.e., the difference between the upper quartile and the lower quartile) as a replacement for the standard deviation for normalization.

This method is more "robust" because the median and IQR are less affected by outliers. Therefore, if the data contain many outliers or the aim is to reduce the impact of outliers on the standardization results, using the "robust" Z-score method is recommended.

Step 1: We start by performing Z-score standardization on the states:

$$S_{\text{norm}} = \frac{S - \mu}{\sigma} \tag{3}$$

where S is the original state, and μ and σ are the mean and standard deviation of the state set, respectively. Since the "robust" option is chosen, $\mu = 0$ and $\sigma = 1$.

Step 2: We calculate the mean of the state set:

$$\text{mean} = \frac{\text{min\_range(state)} + \text{max\_range(state)}}{2} \tag{4}$$

where min_range(state) represents the minimum state information in the state set, and max_range(state) is the maximum.

Step 3: We divide the standardized states by the mean computed in Equation (4):

$$S_{\text{final}} = \frac{S_{\text{norm}}}{\text{mean}} \tag{5}$$

The completion of state normalization through these three steps prepares the model for further training. It is important to note, however, that this method may be less effective for simpler state spaces. For instance, in tasks like CartPole, where state variables exhibit minor fluctuations and lack significant outliers or complex distributions, this approach may not be as beneficial. Consequently, it tends to be more advantageous in scenarios involving data with complex distributions or outliers. The details are provided in Algorithm 1.

However, traditional normalization methods have some limitations. Notably, in high-dimensional or complex state spaces, they can distort the original structure and intrinsic relationships within the data.

In response to these challenges, we introduce a strategy that incorporates residual connections. The core principle of residual connections is to append the output from a preceding layer to the input of the subsequent layer. This approach enables the network to concentrate on learning the residuals rather than the full feature mappings [28], as shown in Figure 3.

Specifically, in state normalization, this method first normalizes the state to obtain a normalized form (state), then adds it to the original state, i.e., normalized (state) + state. This combined state is used as the input to the next layer, which is mathematically expressed as layer_input = normalized (state) + state.

This strategy brings the following benefits:

1. Due to the presence of residual connections, the original state information is preserved and not lost;

2. The network can learn differential information about the normalized state, enhancing training effectiveness;
3. Even when normalization performs poorly, the model can still rely on the original state information, thereby enhancing robustness;
4. Residual connections facilitate the flow of optimization gradients, avoiding the problems of gradient vanishing and explosion.

Algorithm 1 State Normalization

```
Require: $E$: Environment; $D$: Discriminator; $\pi$: Policy; $T_{\text{expert}}$: Expert trajectories; $N$: Number of training epochs; $R$: Replay Buffer; $C$: Replay Buffer capacity
 1: Initialize: $C = \{\}$, $R \leftarrow T_{\text{expert}}$
 2: Sample trajectories $T_{\text{policy}}$ using $\pi$ in $E$ and store in $R$, maintaining capacity $C$ by removing oldest data if necessary
 3: for epoch $= 1$ to $N$ do
 4:     $T_{\text{policy}} \leftarrow \pi(E)$,$R \leftarrow R \cup T_{\text{policy}}$
 5:     if $|R| > C$ then
 6:         Remove the oldest data from $R$
 7:     end if
 8:     for each $(s, a)$ in $R$ do
 9:         $D.train(s, a)$        //Train $D$ to differentiate between $T_{\text{expert}}$ and $T_{\text{policy}}$
10:     end for
11:     Train policy $\pi$ using rewards from $D$ and experiences from $R$,
                $\pi.train(D, (-logD(s, a)))$
        Where, $-logD(s, a)$ represents the returned reward
12:     Calculate state normalization statistics from $R$,$S = \{s_1, s_2, \ldots, s_n\} \leftarrow R$
13:     $\mu = \frac{1}{n}\sum_{i=1}^{n} s_i$;$\sigma^2 = \frac{1}{n}\sum_{i=1}^{n}(s_i - \mu)^2$; mean $= \frac{\min(S)+\max(S)}{2}$
14: end for
15: for each state $s$ in $R$ do
16:     $s_{\text{norm}} = \frac{s-\mu}{\sigma}$
17:     $s = \frac{s_{\text{norm}}}{\text{mean}}$
18: end for
```

The incorporation of residual connections notably enhances the network's representational and fitting capabilities. By integrating skip connections, the network becomes more adept at identifying features across various levels. Additionally, it gains the ability to bypass certain non-essential layers when needed [27]. This allows information to flow unimpeded, even in deep networks, reducing the problem of gradient vanishing and helping the network capture richer characteristics. Therefore, adopting residual connections not only preserves the original features of the state but also optimizes the training stability of the network, accelerating the convergence process.

To capture dependencies in long-range state sequences and empower the agent to make more effective decisions in complex environments, we introduced a self-attention mechanism in GAIL. The definition of this mechanism is as follows:

$$\text{Attention}(Q, K, V) = \text{Softmax}(\frac{QK^T}{\sqrt{d_k}})V \tag{6}$$

where Q, K, and V are the query, key, and value respectively, and d_k represents the dimensionality of the key.

In the context of RL, we consider the past states and actions as keys and values, respectively, and regard the current state as the query. Utilizing the self-attention mechanism enables the agent to evaluate the relevance of past states and actions for its present decision making, facilitating more informed choices.

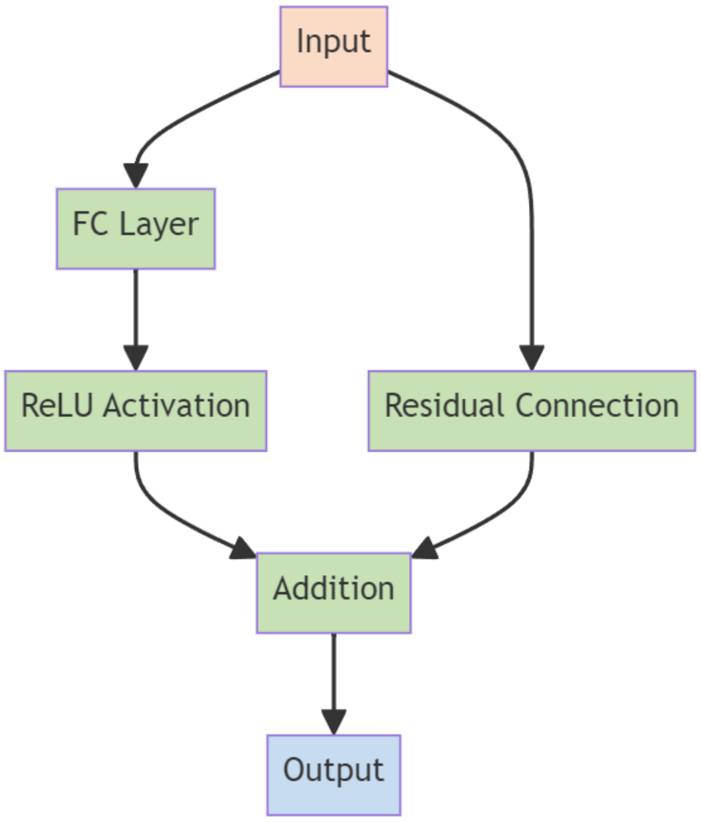

Figure 3. The structure of a residual module.

Figure 4 shows an example of a network architecture incorporating the self-attention mechanism. Here, s_E and a_E represent the expert's state and action, respectively, and D is the discriminator, whose goal is to distinguish whether the state–action pair comes from the agent or the expert. $D(s_t, a_t)$ is the discriminator's evaluation of the action (a_t) taken by the agent in state s_t. The workflow of GAIL is summarized as follows. The agent generates trajectories in the environment based on the policy (π), while the discriminator distinguishes the source of these trajectories. The agent updates its policy with the aim of misleading the discriminator into thinking that its behavior comes from the "expert". As training progresses, the agent's policy approaches that of the expert, while the discriminator strives to accurately differentiate, enhancing the effectiveness of imitation learning.

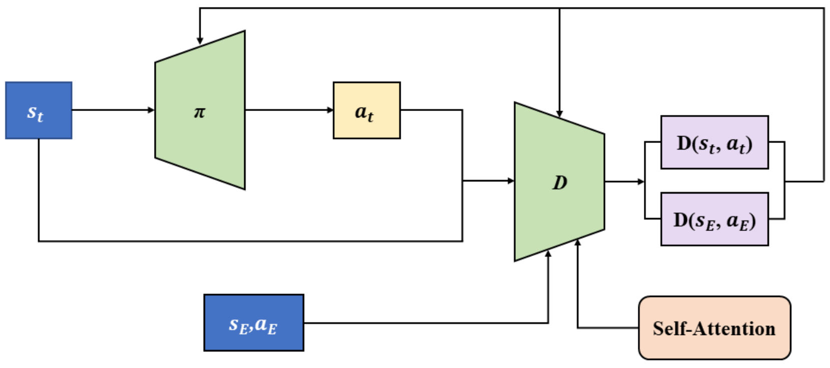

Figure 4. A neural network structure with self-attention.

5. Experiments

5.1. Experimental Description

In our experimental setup, task-specific reward mechanisms were established following the guidelines of OpenAI Gym [35]. We compiled multiple datasets from expert policies, each comprising around 50 trajectories of state–action pairs. During the model implementation phase, the GAIL policy interacted iteratively with the environment to collect new state–action pairs. These samples primarily informed the training of the discriminator. To boost its accuracy, a self-attention mechanism was integrated into the discriminator. This enhancement enabled precise evaluation of the differences between the generated policy and the actual expert policy. The output from the discriminator was then converted into a reward signal for the policy. This reward signal was utilized in training the policy through reinforcement learning algorithms.

In this experiment, we employed the proximal policy optimization (PPO) algorithm. Additionally, to ensure the stability of state–action pairs, we integrated a form of state normalization combined with residual connections into the algorithm. We evaluated the performance of the optimized GAIL model on three benchmark physical control tasks.

To ensure fairness in comparison, the hidden layers of the policy network in all experiments were uniformly set to have 128 units. At the start of each experiment, all networks underwent random initialization. For each task, we ensured that all three methods received an equal amount of environmental interaction training (Table 1).

In this experiment, the following hardware setup was used:

- CPU: 12th Gen Intel(R) Core(TM) i7-12650H; clock speed: 2300 Mhz; 10 cores and 16 logical processors.
- GPU: NVIDIA GeForce RTX 4050.

Table 1. Hyperparameter configuration.

Hyperparameter	Value	Description
Actor Learning Rate	1×10^{-3}	The learning rate within the actor network when generating expert trajectories.
Critic Learning Rate	1×10^{-2}	The learning rate within the critic network when generating expert trajectories.
Learning Rate	1×10^{-3}	The learning rate for the GAIL network.
Discount Factor (γ)	0.98	The discount factor for future rewards in the value function.
Lambda λ	0.95	The λ parameter for generalized advantage estimation (GAE).
Epsilon (ϵ)	0.2	The clipping parameter in proximal policy optimization (PPO).
Random Seed	0	The seed value for the random number generators to ensure reproducibility.

Our experiments commenced with preliminary tests on PyGame, selecting LunarLander-v2 as the test environment, as shown in Figure 5. In this game, the task is to maneuver a lander to ensure a safe landing on diverse terrain. The player's objective is to accurately guide the lander to a stable descent within a flat area marked by two flags. A successful landing in the designated zone rewards the player with high scores; conversely, if the lander crashes or strays off-screen, the game ends in failure, potentially resulting in low or negative scores.

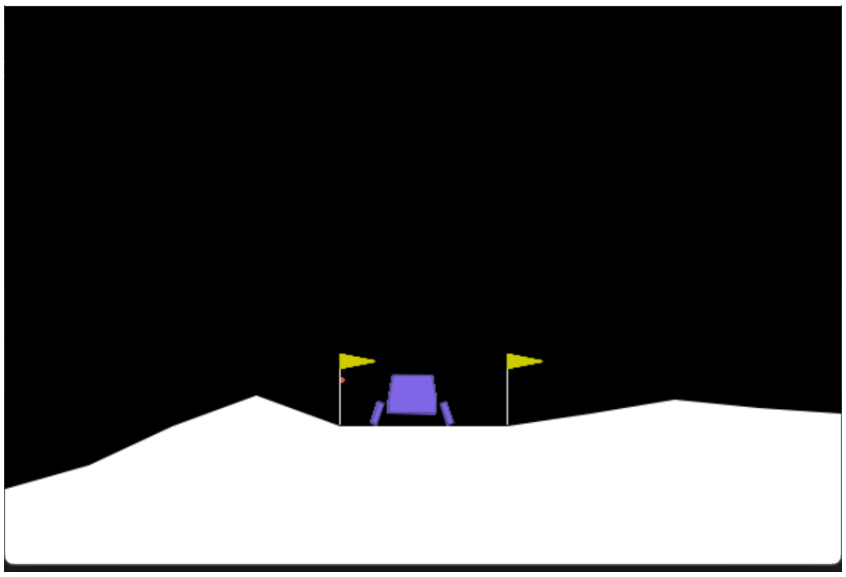

Figure 5. LunarLander game.

We compared the performance of the GAIL algorithm with that of our enhanced version, GAIL-norm, in this setting. The experimental results, as shown in Figure 6, revealed that our GAIL-norm method not only surpassed the traditional GAIL in terms of stability but also demonstrated superior performance from the early stages of the experiment.

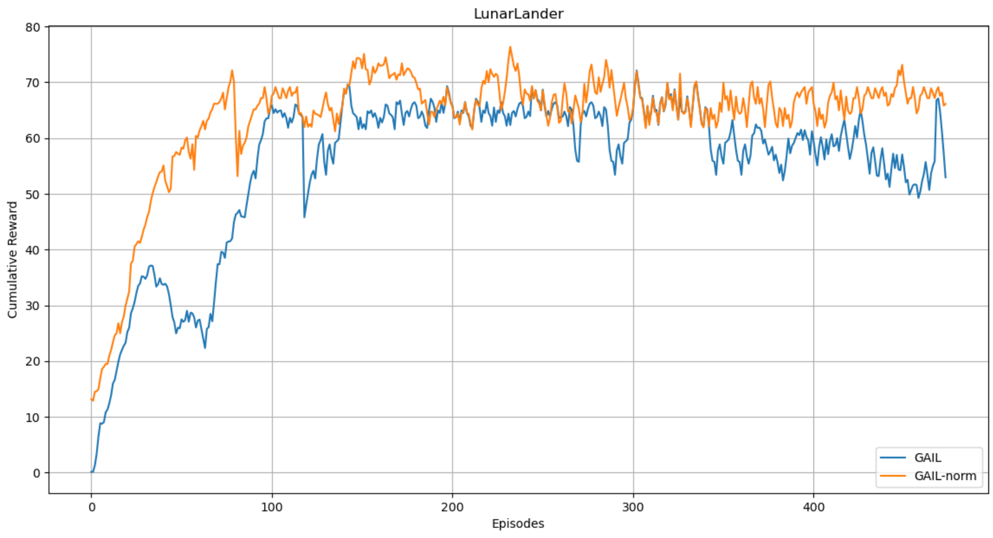

Figure 6. Performance comparison in LunarLander-v2.

We conducted two baseline tests on the improved GAIL:

1. Behavior cloning: The given dataset of state–action pairs was divided into 70% training data and 30% validation data. We trained the model using supervised learning, employing the Adam [36] optimizer. Throughout the training process, we continuously monitored the validation error until it no longer decreased.
2. GAIL: GAIL is a method that utilizes a GAN architecture [37]. Its objective is to minimize the Jensen–Shannon divergence between the distributions of expert and learner behavior.

We conducted simulations on the learned policy, with the obtained rewards presented on the vertical axis. In this simulation, we employed a data-scaling method, mapping the rewards of the expert policy to 1 and the rewards of the random policy to 0.

Figure 7 shows a performance comparison of our method with BC and GAIL on tasks such as Cartpole, HalfCheetah, and Humanoid.

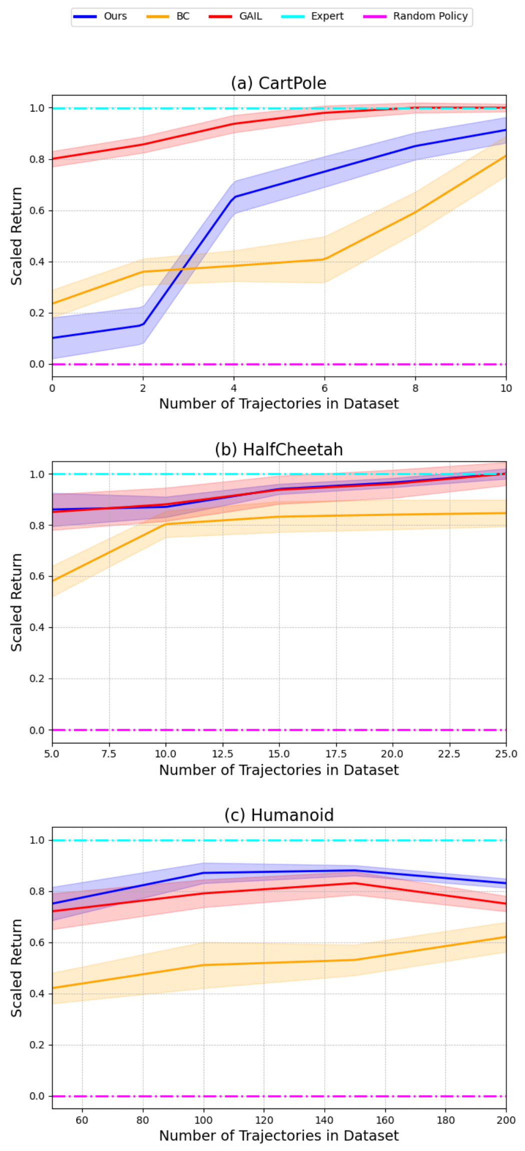

Figure 7. Performance comparison.

As mentioned earlier, the “robust” Z-score standardization method has advantages when dealing with data that contain numerous outliers. However, in environments like the CartPole task, where the data distribution is relatively simple and lacks a significant number of outliers, this method does not demonstrate the expected effectiveness.

Our method outperforms the other two algorithms in the remaining two tasks. Detailed experimental data clearly show that the performance of the behavior cloning (BC) method lags that of the other methods. This can primarily be attributed to the compounding error effect, which is a well-known issue with behavior cloning methods, as pointed out in the literature [9,38].

5.2. Ablation Experiment

To demonstrate the independent effects of combining the normalization method with residual connections and attention mechanisms on the algorithm’s performance, we conducted the following ablation experiment:

In the CartPole, HalfCheetah, and Humanoid control tasks, we compared the performance of GAIL models that only incorporate residual connections for normalization with GAIL models that only utilize attention mechanisms, as well as the performance of the complete model.

Due to the similarity of performance of the three models, displaying the standard deviation range would make the line graph appear too crowded. In order to observe the performance curves of the three models more clearly, we do not present the standard deviation in the graph of experimental results. The experimental results are shown in Figure 8.

In Figure 8, GAIL-1 represents the GAIL model that only incorporates attention mechanisms, while GAIL-2 represents the GAIL model that only utilizes residual connections for normalization.

The results are consistent with our expectations. In the CartPole task, GAIL-2's performance is noticeably lower compared to the other two models due to the use of the "robust" Z-score method. However, in the HalfCheetah and Humanoid tasks, the addition of either residual connections for normalization or attention mechanisms improved the performance of GAIL. This demonstrates that residual normalization and attention mechanisms, as two relatively independent modules, can effectively enhance the performance of the GAIL model in complex control tasks. The experimental results align with our hypothesis and validate the positive impact of residual normalization and attention mechanisms on the performance of GAIL.

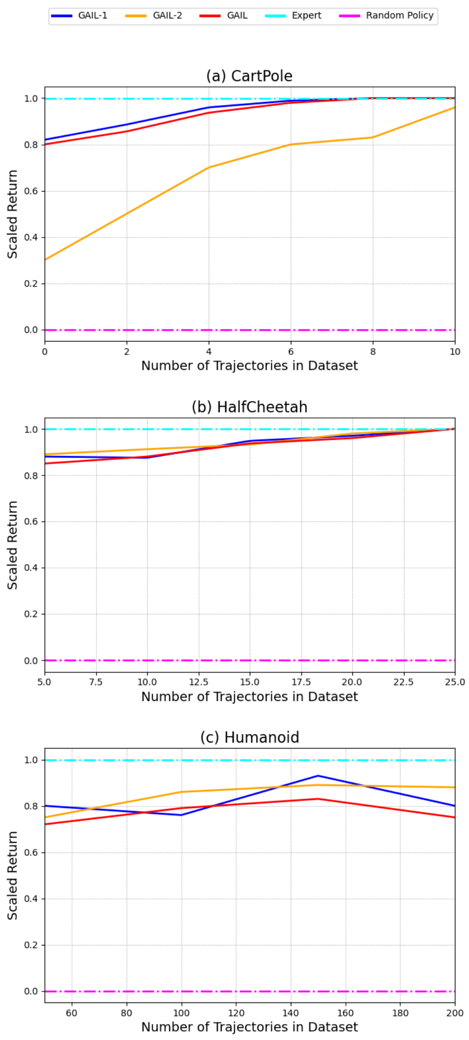

Figure 8. Experimental results.

6. Conclusions

This paper presents a new reinforcement learning framework designed to tackle the difficulties of state representation and algorithmic stability in complex, dynamic environments. We integrated a residually connected state normalization method with generative adversarial imitation learning (GAIL). This integration improves the utilization of state information and partially addresses the problems of gradient vanishing and exploding. Moreover, an attention mechanism is introduced to dynamically evaluate the significance of information across different time steps, enhancing decision-making accuracy. While these advancements boost algorithmic performance in complex scenarios, they demonstrate limited improvement in simpler, low-dimensional tasks and entail increased computational demands, necessitating high-performance computing resources. Future work will explore the integration of meta learning to increase this method's adaptability to various tasks and its capacity for rapid adjustment to new challenges.

Author Contributions: Conceptualization, Y.G. and T.H.; methodology, Y.G.; software, Y.G. and X.W.; experiment and validation, X.W. and X.Y.; writing—original draft preparation, Y.G. and G.Z.; writing—review and editing, G.Z.; supervision, X.Y. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Data Availability Statement: Data associated with this study are available from the corresponding author upon reasonable request.

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

References

1. Hussein, A.; Gaber, M.M.; Elyan, E.; Jayne, C. Imitation learning: A survey of learning methods. *ACM Comput. Surv. (CSUR)* **2017**, *50*, 1–35. [CrossRef]
2. Kendall, A.; Hawke, J.; Janz, D.; Mazur, P.; Reda, D.; Allen, J.M.; Lam, V.-D.; Bewley, A.; Shah, A. Learning to drive in a day. In Proceedings of the 2019 International Conference on Robotics and Automation (ICRA), Montreal, QC, Canada, 20–24 May 2019; pp. 8248–8254.
3. Silver, D.; Schrittwieser, J.; Simonyan, K.; Antonoglou, I.; Huang, A.; Guez, A.; Hubert, T.; Baker, L.; Lai, M.; Bolton, A.; et al. Mastering the game of go without human knowledge. *Nature* **2017**, *550*, 354–359. [CrossRef] [PubMed]
4. Andrychowicz, O.A.I.M.; Baker, B.; Chociej, M.; Jozefowicz, R.; McGrew, B.; Pachocki, J.; Petron, A.; Plappert, M.; Powell, G.; Ray, A.; et al. Learning dexterous in-hand manipulation. *Int. J. Robot. Res.* **2020**, *39*, 3–20. [CrossRef]
5. Kaelbling, L.P.; Littman, M.L.; Moore, A.W. Reinforcement learning: A survey. *J. Artif. Intell. Res.* **1996**, *4*, 237–285. [CrossRef]
6. Eschmann, J. Reward function design in reinforcement learning. In *Reinforcement Learning Algorithms: Analysis and Applications*; Springer: Cham, Switzerland, 2021; pp. 25–33.
7. Zhou, W.; Li, W. Programmatic reward design by example. *Proc. AAAI Conf. Artif. Intell.* **2022**, *36*, 9233–9241. [CrossRef]
8. Minsky, M. Steps toward artificial intelligence. *Proc. IRE* **1961**, *49*, 8–30. [CrossRef]
9. Ross, S.; Bagnell, D. Efficient reductions for imitation learning. In Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics, Sardinia, Italy, 13–15 May 2010; pp. 661–668.
10. Arora, S.; Doshi, P. A survey of inverse reinforcement learning: Challenges, methods and progress. *Artif. Intell.* **2021**, *297*, 103500. [CrossRef]
11. Ho, J.; Ermon, S. Generative adversarial imitation learning. *arXiv* **2016**, arXiv:1606.03476.
12. Lubana, E.S.; Dick, R.P.; Tanaka, H. Beyond batchnorm: Towards a unified understanding of normalization in deep learning. *Neural Inf. Process. Syst.* **2021**, *34*, 4778–4791.
13. Salimans, T.; Kingma, D.P. Weight normalization: A simple reparameterization to accelerate training of deep neural networks. *arXiv* **2016**, arXiv:1602.07868.
14. Wu, Y.; He, K. Group normalization. *Int. J. Comput. Vis.* **2018**, *128*, 742–755. [CrossRef]
15. Zinkevich, M.A.; Weimer, M.; Smola, A.; Li, L. Parallelized stochastic gradient descent. *Neural Inf. Process. Syst.* **2010**, *23*, 1–9.
16. Bottou, L. Large-scale machine learning with stochastic gradient descent. In Proceedings of the International Conference on Computational Statistics, Paris, France, 22–27 August 2010.
17. Ioffe, S.; Szegedy, C. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In Proceedings of the International Conference on Machine Learning, Lille, France, 6–11 July 2015; pp. 448–456.
18. Santurkar, S.; Tsipras, D.; Ilyas, A.; Madry, A. How does batch normalization help optimization? *arXiv* **2018**, arXiv:1805.11604.
19. Sun, J.; Cao, X.; Liang, H.; Huang, W.; Chen, Z.; Li, Z. New interpretations of normalization methods in deep learning. In Proceedings of the AAAI Conference on Artificial Intelligence, New York, NY, USA, 7–12 February 2020.

20. Zhang, B.; Sennrich, R. Root mean square layer normalization. *arXiv* **2019**, arXiv:1910.07467.
21. Miyato, T.; Kataoka, T.; Koyama, M.; Yoshida, Y. Spectral normalization for generative adversarial networks. *arXiv* **2018**, arXiv:1802.05957.
22. Lin, Z.; Sekar, V.; Fanti, G.C. Why spectral normalization stabilizes gans: Analysis and improvements. *Neural Inf. Process. Syst.* **2020**, *34*, 9625–9638.
23. Bhatt, A.; Argus, M.; Amiranashvili, A.; Brox, T. Crossnorm: Normalization for off-policy td reinforcement learning. *arXiv* **2019**, arXiv:1902.05605.
24. Gogianu, F.; Berariu, T.; Rosca, M.C.; Clopath, C.; Busoniu, L.; Pascanu, R. Spectral normalisation for deep reinforcement learning: An optimisation perspective. In Proceedings of the International Conference on Machine Learning, Virtual, 18–24 July 2021; pp. 3734–3744.
25. Puterman, M.L. *Markov Decision Processes: Discrete Stochastic Dynamic Programming*; John Wiley & Sons: Hoboken, NJ, USA, 2014.
26. Zhang, X.; Li, Y.; Zhang, Z.; Zhang, Z.-L. f-GAIL: Learning f-Divergence for Generative Adversarial Imitation Learning. *arXiv* **2020**. [CrossRef]
27. Bishop, C.M. *Neural Networks for Pattern Recognition*; Oxford University Press: Oxford, UK, 1995.
28. Ripley, B.D. *Pattern Recognition and Neural Networks*; Cambridge University Press: Cambridge, UK, 1996.
29. Venables, W.; Ripley, B. *Modern Applied Statistics with s-Plus*; Springer: New York, NY, USA, 1999.
30. He, K.; Zhang, X.; Ren, S.; Sun, J. Deep Residual Learning for Image Recognition. In Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR), Las Vegas, NV, USA, 27–30 June 2016. [CrossRef]
31. Lesort, T.; Díaz-Rodríguez, N.; Goudou, J.-F.; Filliat, D. State representation learning for control: An overview. *Neural Netw.* **2018**, *108*, 379–392. [CrossRef]
32. Charles, Z.; Papailiopoulos, D. Stability and Generalization of Learning Algorithms that Converge to Global Optima. *arXiv* **2017**. [CrossRef]
33. Lillicrap, T.P.; Hunt, J.J.; Pritzel, A.; Heess, N.; Erez, T.; Tassa, Y.; Silver, D.; Wierstra, D. Continuous control with deep reinforcement learning. *Computerence* **2015**. [CrossRef]
34. Kappal, S. Data normalization using median median absolute deviation MMAD based Z-score for robust predictions vs. min–max normalization. *Lond. J. Res. Sci. Nat. Form.* **2019**, *19*, 10.13140.
35. Brockman, G.; Cheung, V.; Pettersson, L.; Schneider, J.; Schulman, J.; Tang, J.; Zaremba, W. Openai gym. *arXiv* **2016**, arXiv:1606.01540.
36. Kingma, D.; Ba, J. Adam: A method for stochastic optimization. *arXiv* **2014**, arXiv:1412.6980.
37. Goodfellow, I.; Pouget-Abadie, J.; Mirza, M.; Xu, B.; Warde-Farley, D.; Ozair, S.; Courville, A.; Bengio, Y. Generative adversarial nets. *Adv. Neural Inf. Process. Syst.* **2014**, *27*, 2672–2680.
38. Ross, S.; Gordon, G.; Bagnell, D. A reduction of imitation learning and structured prediction to no-regret online learning. In Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics, Fort Lauderdale, FL, USA, 11–13 April 2011; pp. 627–635.