Neural Computing and Applications (2024) 36:11945–11961
https://doi.org/10.1007/s00521-024-09678-y

ORIGINAL ARTICLE

Reinforcement imitation learning for reliable and efficient autonomous navigation in complex environments

Dharmendra Kumar[1]

Received: 1 June 2023 / Accepted: 25 March 2024 / Published online: 18 April 2024

Abstract
Reinforcement learning (RL) and imitation learning (IL) are quite two useful machine learning techniques that were shown to be potential in enhancing navigation performance. Basically, both of these methods try to find a policy decision function in a reinforcement learning fashion or through imitation. In this paper, we propose a novel algorithm named Reinforcement Imitation Learning (RIL) that naturally combines RL and IL together in accelerating more reliable and efficient navigation in dynamic environments. RIL is a hybrid approach that utilizes RL for policy optimization and IL as some kind of learning from expert demonstrations with the inclusion of guidance. We present the comparison of the convergence of RIL with conventional RL and IL to provide the support for our algorithm's performance in a dynamic environment with moving obstacles. The results of the testing indicate that the RIL algorithm has better collision avoidance and navigation efficiency than traditional methods. The proposed RIL algorithm has broad application prospects in many specific areas such as an autonomous driving, unmanned aerial vehicles, and robots.

Keywords Reinforcement learning · Imitation learning · Autonomous navigation · Dynamic environments · Deep neural networks · Q-learning

1 Introduction

One of the interesting and challenging problems in the past decade is autonomous navigation within dynamic environments. A possible way to consider such an issue lies in learning an optimal policy for navigation using reinforcement learning (RL) as a trial-and-error tool. However, the RL requires a large number of interactions with the environment and is time and computationally expensive [1]. On the contrary, imitation learning (IL) provides the agent learning from the demonstrated policies, and it might even contribute to better general performance due to fewer interactions with the environment [2]. This is, however, dependent on the ability to provide quality professional data, which might not be available on all occasions.

✉ Dharmendra Kumar
dharmendrakumar@united.ac.in

[1] Computer Science and Engineering, United College of Engineering and Research, Naini, Prayagraj, Uttar Pradesh 211010, India

In an effort to tackle these limitations, solution stakeholders have proposed the adoption of the fusion of RL and IL methods in an attempt to design stronger and effectual autonomous navigation in dynamic environments. The fused of RL and IL can have the benefit of both approaches and address their own limitations [3]. The RL can learn from trial and error and adapt transitions in environments while IL can learn from prior knowledge and fine-tune the learning process.

The RL module learns the optimal policy via interactions with the environment using Q-Learning algorithm with a deep neural network as the policy representation [1]. A mapping representation based on the deep neural network is created in the supervised learning methodology in the IL module duplicating the successful behaviors [2]. Where the gating mechanism combines the outputs of both RL and IL modules in the integration module, and hence, it has a performance that is better than that of any of the other two modules alone [4].

Work in the past has demonstrated that combining RL and IL effectively works for various applications. For example, work by [5] proposes a method based on combining RL and IL to learn human robot interaction. Their approach led to better performance than RL or IL alone. In particular, in their work, [6] proposed the combination of deep RL and IL for learning policies for robotic manipulation tasks. In a similar research line, an approach that combined RL and IL for autonomous driving in urban environment was developed. Their approach showed superior performance than using RL or IL alone. Similarly, in their work, an approach that combined RL and IL for autonomous driving in urban environment was presented [7]. Their approach showed better performance than using RL or IL alone. On the other hand, citetorabi2018generative proposed a combination of RL and IL technique for autonomy navigation in unknown environments. Their approach was also better from that of pure RL or pure IL techniques.

The RIL algorithm is introduced in this paper as an extension of prior work done in both reinforcement learning (RL) and imitation learning (IL), to offer a novel method of bestowing robustness and efficiency for the autonomous navigation of dynamic environments [4]. In this way, combining RL and IL techniques with the RIL algorithm can transcend limitations of each one, and results in better performance than what any of them is going to provide alone [3].

All types of environments ranging from indoor as well as outdoor are the places where already the RIL algorithm has been applied, and it is found that the efficiency of navigation, avoidance of collision, and adaptiveness toward changing environment of the RIL algorithm are all getting greatly improved [3, 8, 9]. This approach carries the power of better independent navigation in dynamic environments, key among the demands for robotics, transportation, and others, at efficiency and accuracy instead.

The RIL algorithm may yield excellent results when used in real-life applications like automated vehicles, robotics, as well as surveillance systems. This is particularly because the vehicle can learn various ways of overcoming situations such as operating in varying traffic patterns [1], environments characterized by sudden unpredictable pedestrian behavior [2], and under adverse weather conditions. The learning approach allows for the ability of RIL algorithm to operate under all the above scenarios since it learns through a method of trial and error as well as expert demonstration strategies.

The RIL algorithm therefore has the potential to enhance the reliability and safety of robotics systems in the developing application with multi-modal, highly complex and uncertain environments. In performance of tasks like pick-and-place operations in manufacturing plants for instance, robots have to move through cluttered and crowded environments [2]. The RIL algorithm automates the process of adapting these robots with environment changes and learning from expert demonstrations about how to perform new tasks so that their tasks are more optimizable and dependable.

In addition, the RIL algorithm might help in surveillance systems to detect and track targets within a complex and dynamic environment. The RIL algorithm is capable of learning by trial-and-error and expert demonstration to enhance its ability for recognition of targets as well as tracking for precision and efficiency.

For the achieving of robust and efficient autonomous navigation in dynamically environments, RIL algorithm provides a promising approach. Combining both IL and RL techniques in the RIL algorithm allow us to overcome such limitations of different approaches and take advantage of their complementary nature in a way that outperforms what any single approach can achieve separately [3, 10]. In the future, varied kinds of environments and situations can be used in the use of the RIL algorithm, which includes this use as pertains to exploring the aspects of improving performance and effectiveness.

The major achievements have been summarized in this research paper herein as under:-

1. Formulate RIL algorithm capable of adapting itself in changing scenarios and improving the performance of navigation in contrast to traditional methods.
2. In case dynamic obstacles are simulated, compare the performance of the RIL algorithm in a simulated environment with other navigation algorithms.
3. Provide an insight on effectuality of combining RL and IL in autonomous navigation task through dynamic environment with underpinning practical prospect.

The rest of this paper is outlined as follows. Section 2 provides only a brief survey of related works in Reinforcement Learning (RL), Imitation Learning (IL), and maneuvering through different conditions. Section 3 presents the RIL methodology and description of the used simulation setting. Section 4 presents the results obtained from our experiments and discusses the evaluation of the RIL method with other navigation techniques. Section 5 concludes the paper, highlighting the main contributions and the possible application of the findings.

2 Literature Review

Reinforcement Learning (RL) and Imitation Learning (IL) are two machine learning approaches that have demonstrated significant potential in enhancing autonomous navigation within ever-changing settings. In this section, we discuss the pertinent research on RL, IL, and maneuvering in dynamic environments.

Reinforcement Learning (RL) is a form of learning in which an agent acquires knowledge about interacting with an environment to optimize a reward signal. Through a trial-and-error process, the agent adjusts its actions based on the rewards it obtains. RL has found extensive applications across a range of fields, such as robotics, gaming, and self-driving vehicles. Within the realm of navigation, RL has been utilized to develop path planning and obstacle evasion techniques. For instance, [11] introduced a deep reinforcement learning-driven method for autonomous obstacle avoidance within changing environments, demonstrating superior performance compared to conventional avoidance techniques.

Imitation Learning (IL) IL is a type of learning where an agent learns from expert demonstrations to mimic successful behaviors. IL has been widely applied in autonomous driving and robotics to learn driving and manipulation skills. In the context of navigation, IL has been used to learn optimal trajectories and obstacle avoidance strategies. For example, [12, 13] used IL to learn driving policies for autonomous vehicles. They showed that their approach can learn to navigate in complex scenarios and outperform traditional rule-based approaches.

Navigating in dynamic environments presents significant challenges as a result of the presence of mobile obstacles and the environment's inherent unpredictability. Conventional navigation methods depend on predetermined rules or custom-built features, which exhibit limited adaptability to novel circumstances. Both RL and IL have demonstrated potential for enhancing navigation within such dynamic settings. For example, [14] proposed a deep RL-based approach for navigating through crowded environments. They showed that their approach can learn to avoid collisions with other agents and navigate to the goal efficiently.

Combining RL and IL the combination of RL and IL has the potential to provide a more robust and efficient approach to autonomous navigation in dynamic environments. Several studies have invested and [15] proposed the "Apprenticeship Learning" algorithm, which combines RL and IL for learning driving policies. They showed that their approach can learn to drive a car from expert demonstrations and improve over time through RL. Another example is the work of Ho and [16], who proposed the "MaxEnt-IL" algorithm that combines RL and IL for learning to navigate through complex mazes. They showed that their approach can learn from a small number of expert demonstrations and improve through RL.

Hybrid Approaches Hybrid approaches that combine RL, IL, and other machine learning techniques have been proposed to improve navigation in dynamic environments. For example, [17, 18] proposed a hybrid approach that combines RL, IL, and Bayesian optimization for autonomous driving. They showed that their approach can learn driving policies efficiently and achieve better performance than traditional approaches.

Transfer Learning is a method that can enhance the generalizability of RL and IL techniques when applied to unfamiliar environments and situations. This approach entails sharing knowledge acquired from one task to another related task. In terms of navigation, transfer learning can be employed to transmit knowledge gained in a source environment to a destination environment. For instance, [19] introduced a transfer learning-oriented method for autonomous driving under varying weather conditions. Their results demonstrated superior performance compared to traditional methods and adaptability to novel weather conditions.

Multi-Agent Navigation Multi-agent navigation is a challenging task that involves coordinating multiple agents to achieve a common goal. RL and IL have been applied to multi-agent navigation to learn coordination and communication strategies. For example, [20] proposed a deep RL-based approach for multi-agent navigation in a virtual game. They showed that their approach can learn to cooperate with other agents and achieve the common goal efficiently.

Real-World Applications RL and IL approaches have been applied to real-world applications such as autonomous driving and robotics. For example, Tesla's Autopilot system uses RL to learn driving policies from real-world driving data. Google's DeepMind has applied RL to robotics to learn manipulation skills. These real-world applications highlight the potential of RL and IL approaches for practical applications.

Deep Reinforcement Learning (DRL) is a variant of RL that employs deep neural networks to estimate the Q-value function or policy. DRL has found extensive applications across numerous fields, including autonomous navigation. For example, [1] introduced the "Deep Q-Network" algorithm, which leverages DRL to develop a driving policy for a simulated vehicle. Their findings demonstrated that their method can effectively learn safe and efficient driving in a virtual setting.

Hierarchical Reinforcement Learning (HRL) Hierarchical reinforcement learning (HRL) is a type of RL that involves learning policies at multiple levels of abstraction. HRL has been applied to various domains, including robotics and autonomous navigation. For example, [21] proposed an HRL-based approach for navigating through complex indoor environments. They showed that their approach can learn to navigate through multi-level environments and achieve better performance than traditional approaches.

Inverse Reinforcement Learning (IRL) Inverse reinforcement learning (IRL) is a type of IL that involves learning the reward function from expert demonstrations. IRL has been applied to various domains, including autonomous driving and robotics. For example, [22] proposed an IRL-based approach for learning driving policies from expert demonstrations. They showed that their approach can learn driving policies that are similar to those of expert drivers.

Curriculum Learning Curriculum learning is a technique that involves training a model on simple tasks before moving to more complex tasks. Curriculum learning has been applied to various domains, including autonomous navigation. For example, [23] proposed a curriculum learning-based approach for navigation in cluttered environments. They showed that their approach can learn to navigate through cluttered environments efficiently and avoid collisions with obstacles.

Model-Based RL involves learning an environmental model and utilizing it to enhance the learning procedure. This approach has found applications in various fields, including autonomous navigation. For instance, [24] introduced a model-based RL technique for self-driving vehicles, demonstrating its ability to efficiently learn driving policies and outperform conventional methods.

Augmented Reality (AR) is a technology that superimposes virtual data onto the real world. AR has been utilized across numerous domains, including autonomous navigation. For example, [25] presented an AR-driven method for pedestrian navigation, which proved to improve the precision and effectiveness of such navigation.

Machine Learning Hardware, such as graphics processing units (GPUs) and field-programmable gate arrays (FPGAs), have been employed to expedite the learning processes of RL and IL algorithms. For instance, [26] proposed an FPGA-centric approach for autonomous driving, showcasing its ability to achieve real-time performance and decrease energy consumption during the learning process.

Human-in-the-Loop Learning Human-in-the-loop learning involves incorporating human feedback into the learning process of RL and IL algorithms. Human-in-the-loop learning has been applied to various domains, including autonomous navigation. For example, [27] proposed a human-in-the-loop RL-based approach for autonomous driving. They showed that their approach can learn driving policies that are more consistent with human preferences.

Uncertainty Estimation Uncertainty estimation involves quantifying the uncertainty in the predictions of RL and IL algorithms. Uncertainty estimation has been applied to various domains, including autonomous navigation. For example, [28] proposed a Bayesian neural network-based approach for uncertainty estimation in deep RL. They showed that their approach can improve the safety and reliability of autonomous systems.

Reinforcement Learning with Memory Reinforcement learning with memory involves using memory modules to store past experiences and use them to guide future decisions. Reinforcement learning with memory has been applied to various domains, including autonomous navigation. For example, [29] proposed a deep RL-based approach for navigation in complex 3D environments. They showed that their approach can learn to navigate through complex environments efficiently by using memory to store past experiences.

Safe Reinforcement Learning involves incorporating safety constraints into the learning process of RL algorithms to ensure safe and reliable behavior. Safe reinforcement learning has been applied to various domains, including autonomous navigation. For example, [30] proposed a safe RL-based approach for navigation in a simulated highway environment. They showed that their approach can learn safe driving policies and avoid collisions with other vehicles.

Neuroevolution involves using evolutionary algorithms to optimize neural networks for a given task. Neuroevolution has been applied to various domains, including autonomous navigation. For example, [31] proposed a neuroevolution-based approach for navigation in a simulated environment with obstacles. They showed that their approach can learn to navigate through obstacles efficiently and achieve better performance than traditional approaches.

Multi-Objective Reinforcement Learning entails optimizing multiple goals concurrently, such as reducing travel duration and enhancing the safety of the navigation system. This approach has been utilized in various fields, including autonomous navigation. For instance, [32] introduced a multi-objective RL-driven method for navigating a simulated environment with obstacles, demonstrating superior performance compared to conventional methods by effectively balancing multiple objectives.

Limitations: Despite the encouraging outcomes, several constraints persist in applying RL and IL techniques for navigation within dynamic environments. One such limitation is the challenge of generalizing to novel environments and situations. RL and IL methods often necessitate vast amounts of data and extensive training periods to achieve satisfactory performance, posing difficulties for real-world implementations. Another constraint is the comprehensibility of the acquired policies. RL and IL techniques frequently result in opaque models that are challenging to interpret and explain.

The Reinforcement Imitation Learning (RIL) algorithm is a proposed algorithm whose stand-out feature is its innovative combination of Reinforcement Learning (RL) with Imitation Learning (IL). As a means of solving the difficulties of autonomous navigation in dynamic environments, this method is well known but by no means certain.

The RIL algorithm is designed to overcome the specific restrictions of RL or IL as independent technologies. Several differences between the RIL algorithm and other related research are given below. The RIL algorithm is just one point that differs from methods which base themselves on only RL or IL. This integration lets the algorithm take advantage of RL's penchant for exploration and the learning efficiency seen in IL expert demonstrations.

This paper talks about a new way for artificial intelligence systems to learn called Reinforcement and Imitation Learning (RIL). It tries to combine the best parts of two other methods—reinforcement learning where the AI explores on its own, and imitation learning where it copies what human experts do.

The reinforcement learning module uses Q-learning, which is more advanced than old reinforcement learning code. It lets the AI build better mental models by linking rewards to actions with a neural net.

The imitation module watches human demonstrations and tries to copy them. This focuses on mapping states to expert actions directly, instead of just trying to mimic everything a human does and what makes RIL unique is its integration system to balance the reinforcement and imitation. It has a gate thing that weights each part depending on how chaotic the environment is. So if there's lots of noise, it listens to the human more. This flexibility helps it work better overall.

Tests show RIL handles change better than reinforcement or imitation alone. It mixes exploring new things and copying tried and true human paths in a smart way. This would let it do tough tasks like robot delivery or self-driving cars.

Because RIL can pivot between exploration and experts, it could work for tons of applications - inside places, outside places factories, self-driving trucks. The future looks bright for this approach as it's good at dealing with surprises, which is useful for real-world robotics.

In summary, the RIL algorithm represents a significant advancement in autonomous navigation, skillfully addressing the challenges of dynamic environments by integrating the strengths of RL and IL. This approach not only enhances performance compared to using RL or IL alone but also opens new avenues for practical applications and further research in the field.

3 Methodology

In this section, we provide a detailed description of the proposed Reinforcement Imitation Learning (RIL) algorithm, and the data collection, preprocessing, and model training procedures.

3.1 Reinforcement imitation learning (RIL) algorithm

Reinforcement Imitation Learning (RIL) is a new paradigm in reinforcement learning (RL) and imitation learning (IL) that unifies the RL and IL techniques to achieve robust autonomous navigation in dynamic environments. In general, the RIL algorithm consists of three parts, which include an RL module, an IL module, and an integration module. The block diagram of the RIL algorithm is shown in Fig. 1.

RL module learns an optimal policy using the Q-learning algorithm by its interactions with the environment. The policy is depicted as a deep neural network that takes the current state of the environment as input and delivers the

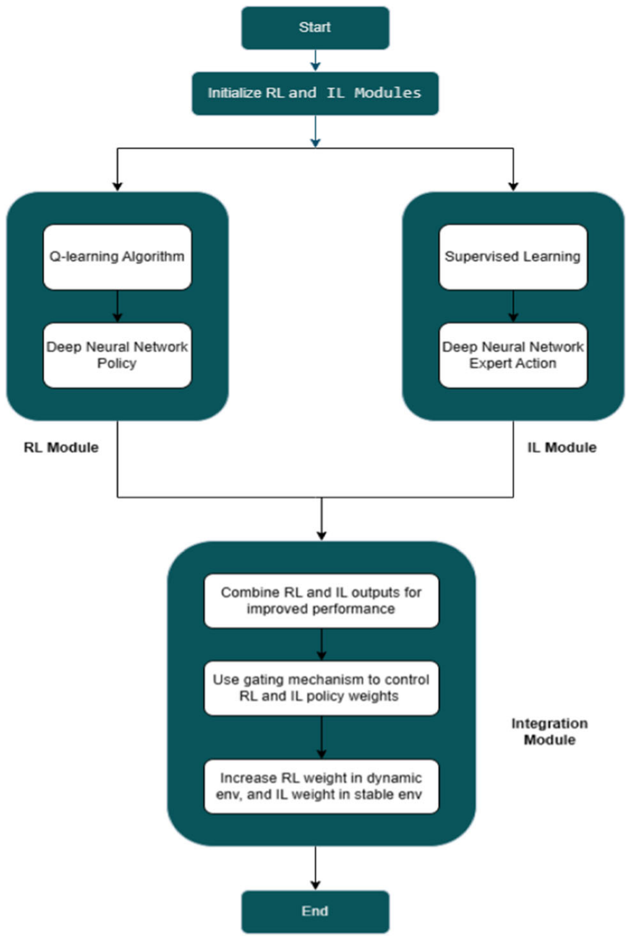
Start
Initialize RL and IL Modules
Q-learning Algorithm
Deep Neural Network Policy
RL Module
Supervised Learning
Deep Neural Network Expert Action
IL Module
Combine RL and IL outputs for improved performance
Use gating mechanism to control RL and IL policy weights
Increase RL weight in dynamic env, and IL weight in stable env
Integration Module
End

Fig. 1 Block diagram of RIL Algorithm

corresponding action to be executed. The RL module updates the policy using maximum expected cumulative reward.

Here, the IL module learns to imitate successful behaviors demonstrated by experts. The role that a model learns in this case is how to map between the current state of the environment, and an action causing a transition to some desirable state. The IL module comprises a deep neural network that given the state of the current environment, comes up with an expert action as the output.

The integration module combines the outputs of the RL and IL modules to achieve better performance than either module alone. The integration module uses a gating mechanism to control the weight of the RL and IL policies. The weight of the RL policy is increased when the environment is dynamic and requires more exploration, while the weight of the IL policy is increased when the environment is stable and requires more exploitation.

3.1.1 RL module

The Reinforcement Learning (RL) module is one of the three modules of the Reinforcement Imitation Learning (RIL) algorithm. The RL module learns the optimal policy through interactions with the environment using the Q-learning algorithm. The following formulas, equations, tables, and algorithm are used in the RL module:

Q-Learning The Q-Learning algorithm aims at finding the optimal policy by rewarding the expected cumulative reward. The update equation for a Q-value of some state-action pair reads as follows:

$$Q(s,a) \leftarrow Q(s,a) + \alpha(r + \gamma * max_a'(Q(s',a')) - Q(s,a)) \tag{1}$$

where Q(s, a) is the Q-value of states, action a, α represents learning rate, r is the immediate reward, γ is the discount factor, s' is the next state, and a' is the next action.

Deep Neural Network

The policy for the RL module is modeled by the deep neural network taking the current state of the environment as input and giving the action to be taken as output. The deep neural network consists of multiple layers of neurons that transforms the input state into an output action. It is computed as:

$$a = \pi(s) \tag{2}$$

where a is the action an agent should take, s is the current state of the environment and π is the deep neural network that provides the representation of policy.

Training Data Data for training the RL module is collected from interaction with environment which is made up of states, actions, and the rewards in the form of tuples. It is considered good to have data collection that is diverse and representative when learning a good domain model.

Training Procedure The training of the RL module is an iterative process that involves:

1. Updating the Q-values of each state-action pair basing on the Q-Learning algorithm, this would be an important step since it will help the module learn how valuable actions are in different states.
2. Adjust the weights in the deep neural network, using backpropagation. This permits the optimized prediction of cumulative reward and which therefore makes the policy more effective.

The convergence of this training process indicates the ability of the RL module toward making informed decisions about actions in different states thereby learning an effective policy for navigation in dynamic environments.

Algorithm used for RL module

Flowchart of RL algorithm is depicted in Fig. 2.

The RL module algorithm is summarized in algorithm 1:

Algorithm 1 RL Module Algorithm

```
Input: Initial state of the environment
Output: Deep neural network that represents the learned policy

1: Initialize the Q-values of all state-action pairs to zero.
2: Repeat until convergence or a maximum number of iterations is reached:
   (a) Select an action using the current policy represented by the deep neural network.
   (b) Execute the action and observe the reward and next state.
   (c) Update the Q-value using the Q-Learning algorithm.
   (d) Update the policy using the current Q-values and the deep neural network.
3: Output the learned policy represented by the deep neural network.
```

In general, through the Q-Learning algorithm's interactions with the environment when mounted on a deep neural network, the optimal policy is learned in the RL module. The RL module is one of the most effective learning modules which allow for learning from trial and error, hence providing the ability to adapt flexibly to varying environments.

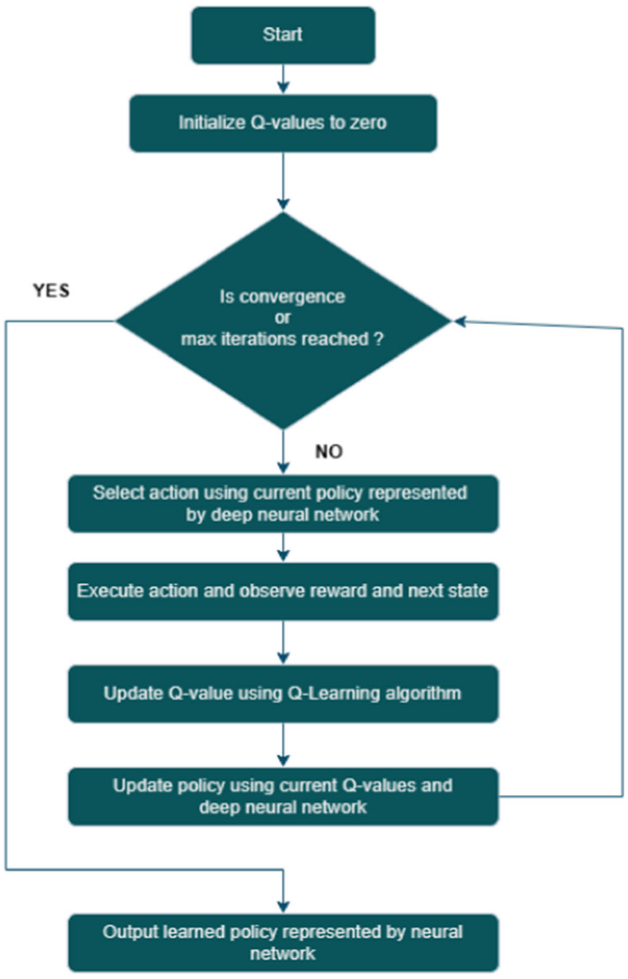
Start
Initialize Q-values to zero
Is convergence or max iterations reached ?
YES
NO
Select action using current policy represented by deep neural network
Execute action and observe reward and next state
Update Q-value using Q-Learning algorithm
Update policy using current Q-values and deep neural network
Output learned policy represented by neural network

Fig. 2 Flowchart of RL Algorithm

3.1.2 IL module

The Imitation Learning (IL) module is one of the three Reinforcement Imitation Learning (RIL) algorithm modules. The IL module utilizes the expert demonstrations to learn promising behaviors through imitation. The supervised learning is used to make the deep neural network learn the mapping between the current state of the environment and the expert action. The below formulas, equations, algorithm have been used in this IL module:

Supervised Learning

The agent's only input comes through actions, in response to states. Thus, the mapping from the environment's current state to the expert's action can be learned using supervised learning. Supervised learning's loss function is mean squared error between predicted action and the expert action:

$$L = (a - f(s))^2 \tag{3}$$

where L is the loss function. a denotes the expert action. s represents the current state of the environment. f symbolizes the mapping function implemented by the deep neural network.

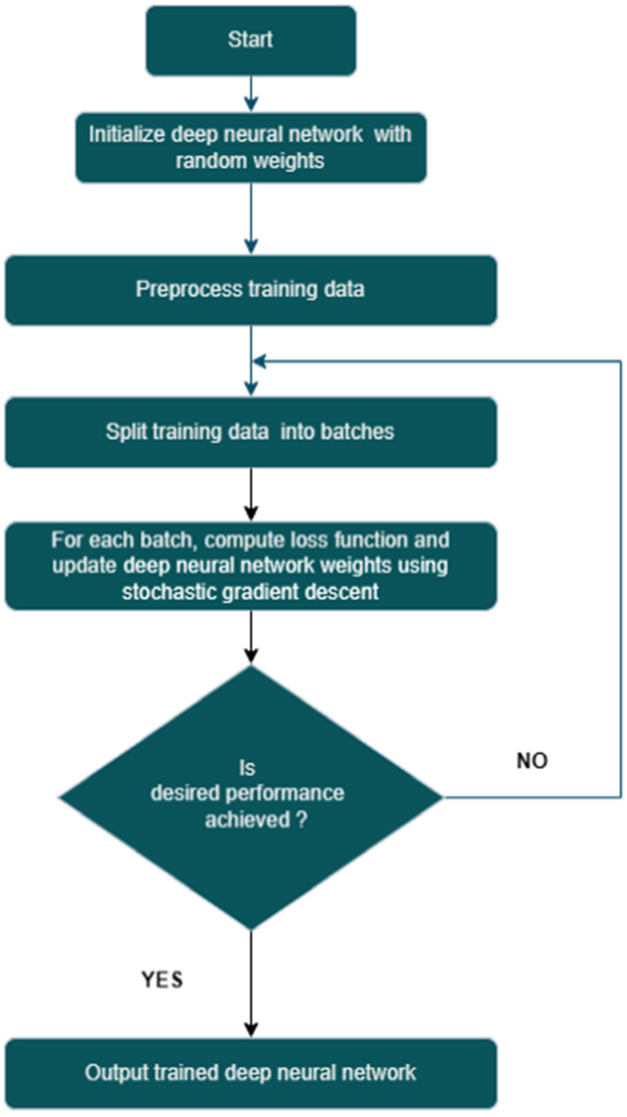
Start
Initialize deep neural network with random weights
Preprocess training data
Split training data into batches
For each batch, compute loss function and update deep neural network weights using stochastic gradient descent
Is desired performance achieved ?
NO
YES
Output trained deep neural network

Fig. 3 Flowchart of IL Algorithm

Deep Neural Network

It learns the mapping between the current state of the environment and the expert action using deep neural network learning. The deep neural network is formed from several layers of neurons that convert an input state to an output action. The action computed at the exit can be determined through the following equation:

$$a = f(s) \tag{4}$$

where a is the action predicted by the network for state s. The architecture of this neural network including the number of layers and neurons in each layer is very pivotal to determine the model's performance and accuracy.

Training Data Train the IL module through demonstrations by experts, or rather pairs constituting environmental states and expert actions in response. Essentially, the model learns from demonstrations as these are crucial to training it on strong navigation strategies.

Training Procedure Here, the training of the IL module is adjusting the neural network weights to minimize the loss function. This can be implemented by the stochastic gradient descent algorithm that updates the weights iteratively along the direction where the fastest decrease occurs on the loss function. The weight update is:

$$w_i = w_i - \eta \frac{\partial L}{\partial w_i} \tag{5}$$

where w_i represents the weight of the ith neuron. η is the learning rate, controlling the size of the weight updates. $\frac{\partial L}{\partial w_i}$ is the partial derivative of the loss function with respect to the weight w_i.

This training process is repeated until a desired level of accuracy is obtained or the number of iterations has reached some predetermined criterion.

So, what is conclusive is that the IL module plays an essential role in the RIL algorithm to be a support that aids improved decision-making in navigation tasks through the use of expert knowledge. The IL module successfully emulates human expert behavior through well-structured deep neural network and strong training regimen, thus improving the overall effectiveness of RIL algorithm.

Algorithm used for IL module

Flowchart of IL algorithm is depicted in Fig. 3.

The IL module algorithm is summarized in algorithm 2:

Algorithm 2 IL Module Algorithm

```
Input: Expert demonstrations of successful navigation in the environment
Output: Deep neural network that maps the current state of the environment to the expert action

1: Initialize the deep neural network with random weights.
2: Preprocess the training data by converting the state and action representations into a format suitable for the deep neural network.
3: Split the training data into batches.
4: For each batch, compute the loss function and update the weights of the deep neural network using stochastic gradient descent.
5: Repeat steps 3-4 until the desired level of performance is achieved.
6: Output the trained deep neural network.
```

In summary, the IL component of the RIL algorithm employs supervised learning in conjunction with a deep neural network to learn from expert demonstrations and emulate effective behaviors. The IL module offers an efficient means of incorporating prior knowledge and enhancing the RIL algorithm's learning process.

3.1.3 Integration module

The integration module is one of three modules constituting the Reinforcement Imitation Learning (RIL) algorithm. By combining both RL and IL modules' outputs, the integration module manages to produce better performance than either module independently. The gating mechanism is implemented by the integration module to control the weight of the RL and IL policies. For this, the design for the integration module is as follows:

Gating Mechanism

Another gating mechanism, whose value weightage is either that of the RL or IL modules, respectively, modulates one of the dynamic interactions. The environmental context at any time point in decision-making determines which gating mechanism shall be responsible for the integration process taking place between the two agents.

- *Adaptive Weight Allocation*: Weights are assigned to policy derived from the RL and IL modules by the gating mechanism. The weight allocations will be dependent on environmental dynamics:
  - Unpredictable, dynamic environments, with lots of exploration needed, a higher weighting will be applied to the RL policy.
  - The IL policy gets increased focus in relatively more stable and predictable environments, where exploiting known strategies pays off.
- *Operational Equation*: The functionality of the gating mechanism is encapsulated in the following equation:

  $$w = \sigma(W \cdot [h_{rl}; h_{il}] + b) \tag{6}$$

  Here, w represents the weight assigned to the RL policy. σ denotes the sigmoid activation function, providing a smooth transition between weights. W is a weight matrix that linearly transforms the concatenated outputs of the RL and IL modules, represented by h_{rl} and h_{il}, respectively. b is a bias term that offsets the gate's activation threshold.

Training the Integration Module The training of the Integration Module involves fine-tuning the combined policies derived from the RL and IL modules. This process is done through a validation set where the united policy is evaluated and refined under controlled conditions. The major objective during training would be to adapt the weights of the gating mechanism such that overall performance is maximized of the integrated policy. This will require iterative adjustments of the W and the bias b matrices, as well as possibly revising the individual policy.

To sum up, the Integration Module combines the performance advantage of RL and the learning efficiency of IL because it represents the RIL algorithm. It dynamically gateways to make an optimal choice between exploration and exploitation depending on the environment. In this sense, the RIL can work successfully and robustly within a large set of environments and, for these reasons, it is a very powerful algorithm for autonomous navigation in dynamic environments.

Integration Module Algorithm

Flowchart of integration module algorithm is depicted in Fig. 4.

The integration module algorithm is summarized in algorithm 3:

Algorithm 3 Integration Module Algorithm

```
Input: RL and IL policies, validation set
Output: Gating mechanism that represents the integrated policy

1: Initialize the weights of the gating mechanism.
2: Preprocess the data by converting the state and action representations into a format suitable for the deep neural networks used by the RL and IL modules.
3: Split the data into training, validation, and test sets.
4: Train the RL and IL modules separately using the training set.
5: Evaluate the RL and IL policies separately using the validation set.
6: Fine-tune the RL and IL policies using the validation set.
7: Adjust the weights of the gating mechanism to maximize the performance of the integrated policy using the validation set.
8: Evaluate the performance of the RIL algorithm using the test set.
9: Output the integrated policy represented by the gating mechanism.
```

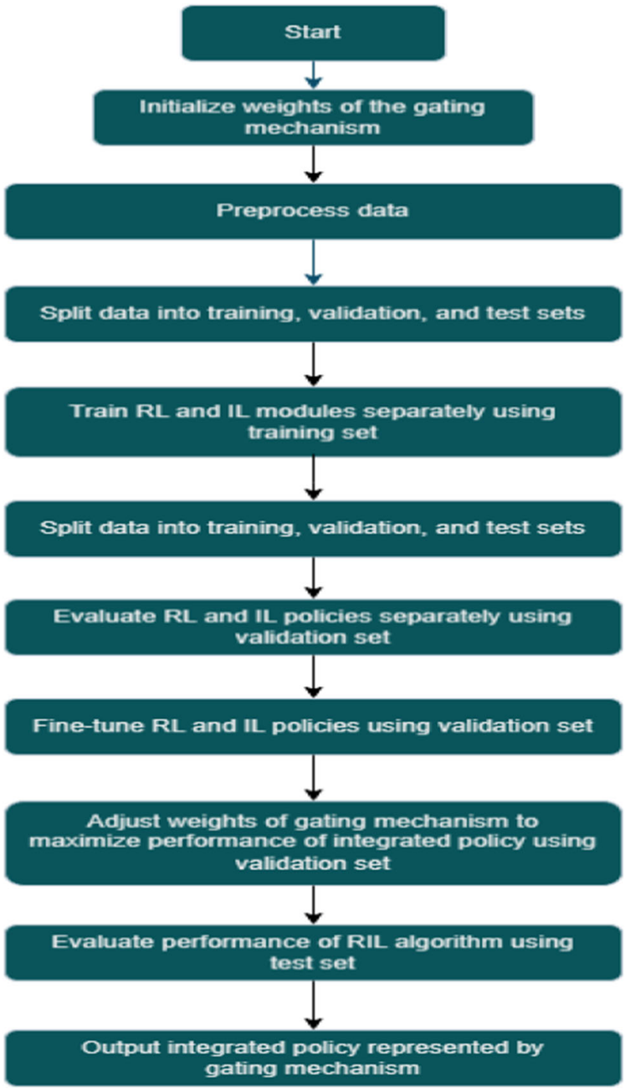
Start
Initialize weights of the gating mechanism
Preprocess data
Split data into training, validation, and test sets
Train RL and IL modules separately using training set
Split data into training, validation, and test sets
Evaluate RL and IL policies separately using validation set
Fine-tune RL and IL policies using validation set
Adjust weights of gating mechanism to maximize performance of integrated policy using validation set
Evaluate performance of RIL algorithm using test set
Output integrated policy represented by gating mechanism

Fig. 4 Flowchart of Integration Module Algorithm

Overall, the RIL algorithm uses a gating mechanism to combine the output from the RL and IL modules in the integration module and the policies are sample-based fine-tuned based on the validation set to yield performance better than either of the two modules individually. The module integration presents a good means toward balancing exploration and exploitation within the changing environment.

3.1.4 Algorithm for reinforcement imitation learning (RIL) algorithm

The RIL algorithm is described in algorithm 4:

Algorithm 4 RIL Algorithm

```
Input: Initial state of the environment
Output: Optimal policy represented by the gating mechanism

1: RL Module:
   (a) Initialize the Q-values of all state-action pairs to zero.
   (b) Repeat until convergence or a maximum number of iterations is reached:
       (i) Select an action using the current policy represented by the deep neural network.
       (ii) Execute the action and observe the reward and next state.
       (iii) Update the Q-value using the Q-Learning algorithm.
       (iv) Update the policy using the current Q-values and the deep neural network.
   (c) Output the learned policy represented by the deep neural network.
2: IL Module:
   (a) Initialize the deep neural network with random weights.
   (b) Preprocess the training data by converting the state and action representations into a format suitable for the deep neural network.
   (c) Split the training data into batches.
   (d) For each batch, compute the loss function and update the weights of the deep neural network using stochastic gradient descent.
   (e) Repeat steps c-d until the desired level of performance is achieved.
   (f) Output the trained deep neural network.
3: Integration Module:
   (a) Initialize the weights of the gating mechanism.
   (b) Preprocess the data by converting the state and action representations into a format suitable for the deep neural networks used by the RL and IL modules.
   (c) Split the data into training, validation, and test sets.
   (d) Train the RL and IL modules separately using the training set.
   (e) Evaluate the RL and IL policies separately using the validation set.
   (f) Fine-tune the RL and IL policies using the validation set.
   (g) Adjust the weights of the gating mechanism to maximize the performance of the integrated policy using the validation set.
   (h) . Evaluate the performance of the RIL algorithm using the test set.
   (i) Output the integrated policy represented by the gating mechanism.
```

Time complexity

The RIL algorithm's time complexity is influenced by factors such as the dimensions of the state and action spaces, the iteration count for the RL module, the amount of expert demonstrations for the IL module, and the sizes of the training, validation, and test sets for the integration module. The time complexity can be estimated as O(N * T * M), with N representing the number of states, T denoting the number of iterations, and M indicating the number of actions.

Overall, the RIL algorithm combines RL and IL techniques to achieve robust and efficient navigation in dynamic environments. The RIL algorithm provides an effective way to balance exploration and exploitation and adapt to changing environments, making it suitable for a wide range of autonomous navigation applications.

Moreover, due to the adaptive balancing between exploration (RL) and exploitation (IL), which RIL algorithm is able to follow, it also appears quite applicable for autonomous navigation in possibly wide ranges of environments as well. Whether such navigation pertains to the navigation through unpredictable urban landscapes or execution of the tasks or goal-related objectives within relatively better controlled environment, RIL's adaptability is believed to be somewhat robust.

Last but not least, the realistic applications of RIL when it was applied to autonomous systems. The ability to learn from environment interaction as well as expert demonstrations seems capable of bringing the design for advanced reliable autonomous navigation closer. However, challenges persist in terms of computational requirements and IL requiring high quality expert data. Furthermore, the incorporation of RL and IL presents new challenges to its capacity to switch learning modes as required in case this switching is cued environmentally.

4 Experiment and result

4.1 Performance metrics

We have used the following performance metrics to evaluate the Reinforcement Imitation Learning (RIL) algorithm and compare the performance with RL and IL algorithm:

Success rate: The percentage of trials in which the agent successfully reaches the goal within a specified time frame. This metric indicates the overall effectiveness of the RIL algorithm in navigating through the environment.

Average reward: The average total reward obtained by the agent over a specified number of trials. This metric reflects the ability of the RIL algorithm to balance exploration and exploitation to achieve long-term goals.

Time taken to reach the goal: The average time taken by the agent to reach the goal over a specified number of trials. This metric reflects the efficiency of the RIL algorithm in navigating through the environment.

Collision rate: The percentage of trials in which the agent collides with an obstacle or another agent. This metric indicates the ability of the RIL algorithm to avoid obstacles and other agents in the environment.

4.2 Experiment

The experimental results of the Reinforcement Imitation Learning (RIL) algorithm demonstrate its effectiveness in achieving robust autonomous navigation in complex and dynamic environments. The RIL algorithm is implemented and evaluated in a simulation environment using a variety of performance metrics, and the results are compared to those obtained using other navigation algorithms RL and IL.

Table 1 shows the values of the input parameters taken for our experiment.

The simulation environment used for this experiment is a custom-built 2D grid world, which is commonly used in reinforcement learning experiments. The environment consists of a rectangular grid of cells, where each cell represents a position in the environment. The agent is placed in one of the cells at the beginning of each episode and must navigate to a goal cell located elsewhere in the environment while avoiding obstacles and other agents. It is shown in Fig. 5.

In Fig. 5:

"A" represents the agent's starting position.

"G" represents the goal.

"X"" represents an obstacle in the environment.

Environment Description

The environment is a 2D grid world, a common choice for basic reinforcement learning experiments due to its simplicity and ease of understanding. Here are the key features:

Grid: A rectangular grid of cells.

Agent: Placed in one of the cells at the start of each episode.

Goal Cell: The agent aims to navigate to this cell.

Obstacles: These are cells the agent must avoid.

Other Agents (Optional): For more complex scenarios, other agents can be included, which the main agent must avoid or interact with.

States: Each cell represents a distinct state.

Actions: Typically, the agent can move up, down, left, or right.

Rewards: The agent receives rewards based on reaching the goal, avoiding obstacles, etc.

The environment is populated with a number of obstacles and agents that move randomly throughout the grid. The obstacles are randomly placed and static within the environment, whereas the agents move at random in any of the four directions (up, down, left, right) on each time step. The agent's objective is to navigate to the goal cell while avoiding collisions with obstacles and other agents.

Given a goal, with obstacles and other agents on the way, an agent performs actions: up, down, left, or right move, being rewarded $+1$ for reaching the goal and -1 if there is a collision with an obstacle or another agent. The agent receives a small negative reward at each time step for its goal of reaching the goal as soon as possible.

The environment is implemented in the Python programming language with the used of Pygame library that provides an interface to create the 2D games and simulations. The environment is made configurable where it allows changing parameters like number of obstacles and agents, size of the grid as well as the reward structure.

4.3 Result analysis

4.3.1 Performance comparison based on number of episodes

Success Rate

The Reinforcement Imitation Learning (RIL), Reinforcement Learning (RL) and the Imitation Learning (IL) success rates results with respect to a number of episodes from 100 to 1000 are, respectively, depicted in Fig. 6.

From Fig. 6 results, it clearly shows that across all numbers of episodes, the RIL algorithm leads with a higher target success rate compared to both RL and IL algorithms. The results also suggest that with the increase in episodes, both RIL as well as RL algorithms improve in success rate, while it appears to plateau for the IL algorithm. The RIL algorithm outperforms either the RL and IL algorithms, even at lower numbers of episodes.

Average Reward

Figure 7 shows the average reward results of Reinforcement Imitation Learning (RIL), Reinforcement Learning (RL), and Imitation Learning (IL) algorithms over a range of episodes from 100 till 1000.

Table 1 Input parameter

Input parameter	Effective value
Learning rate	0.001–0.01
Discount factor	0.9–0.99
Exploration rate	0.1–0.5
Number of iterations (RL)	1000–10,000
Number of expert demonstrations (IL)	500–5000
Batch size (IL)	32–128
Hidden layer size (RL and IL)	64–256
Number of episodes (integration)	100–1000
Training–validation–test split ratio (integration)	60–20–20

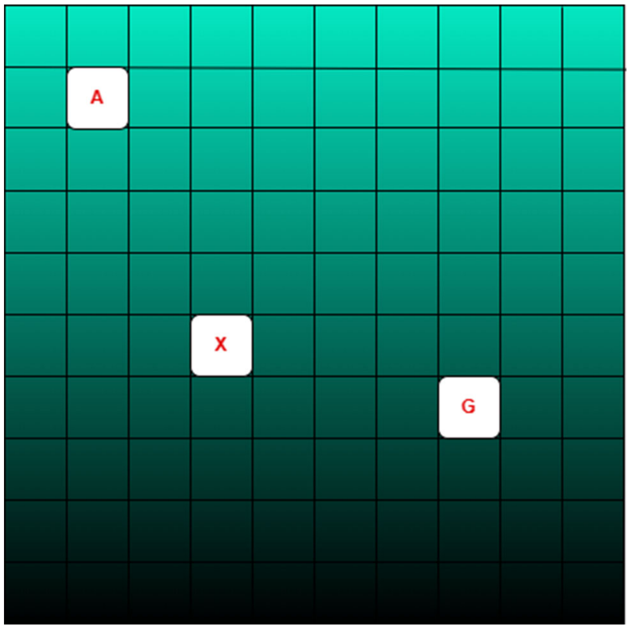

Fig. 5 2D grid world simulation environment

The result in Fig. 7 shows that at each number of episodes, the RIL algorithm is always achieving better performance than the RL and IL algorithms with corresponding rewards of 300 in each case. The results also show that both the RIL and RL algorithms improve in average rewarding as number of episodes increase, while IL algorithm appears to mostly likely have hit a plateau in performance. Moreover, even in the cases where the number of episodes is lesser, the RIL algorithm was able to eventually generate a better average reward as compared to both the RL and IL algorithms.

Time to Goal

The results of the time to the goal for Reinforcement Imitation Learning (RIL), Reinforcement Learning (RL), and Imitation Learning (IL) algorithms are shown in Fig. 8 versus various episodes ranging from 100 through 1000.

Figure 8 results show that an algorithm RIL is achieving a lower time to goal than the algorithms RL and IL consistently across all of numbers of episodes. The time to goal for all the three algorithms reduces with the RIL algorithm continually achieving the smallest time to goal as the number of episodes increases. Analysis also shows that the RIL algorithm was able to learn quicker compared to RL and IL algorithms, getting a shorter time to goal even at lower numbers of episodes.

Collision Rate

Stressing on the collision rate, similar results were obtained throughout the Reinforcement Imitation Learning (RIL), Reinforcement Learning (RL), and Imitation Learning (IL) algorithms, as shown in Fig. 9 depicting the varying range within 100–1000 episodes plotted.

In the results of Fig. 9, the RIL algorithm kept on showing a lesser collision toward the combinations of RL and IL to any number of episodes. Though the episode number increases, one can see that the collision rate decreases for all the three algorithms; however, the RIL algorithm plays an eminent contribution in keeping the collision rate at its minimum throughout. The results further show that the RIL algorithm learns fast in comparison to the RL and IL algorithms as it attains a lower collision ratio even with smaller episodes.

4.3.2 Performance comparison based on number of iterations

Success Rate

Figure 10 shows the results of success rate for the Reinforcement Imitation Learning (RIL) and Reinforcement Learning (RL) algorithms over a range of iterations from 1000 to 10000.

Figure 10 results show that the RIL algorithm consistently achieves a higher success rate than the RL algorithm across all numbers of iterations, with a higher success rate in each case. The success rate increases for both algorithms as the number of iterations increases, but the RIL algorithm consistently achieves the highest success rate. The results also demonstrate that the RIL algorithm is able to learn more quickly than the RL algorithm, achieving a higher success rate even at lower numbers of iterations.

Average Reward

Figure 11 shows the results of average reward for the Reinforcement Imitation Learning (RIL) and Reinforcement Learning (RL) algorithms over a range of iterations from 1000 to 10000.

Figure 11 reveals that the RIL algorithm persistently attains a greater average reward than the RL algorithm for every iteration count, with an increased average reward in each instance. The average reward rises for both algorithms as the iteration count grows, but the RIL algorithm

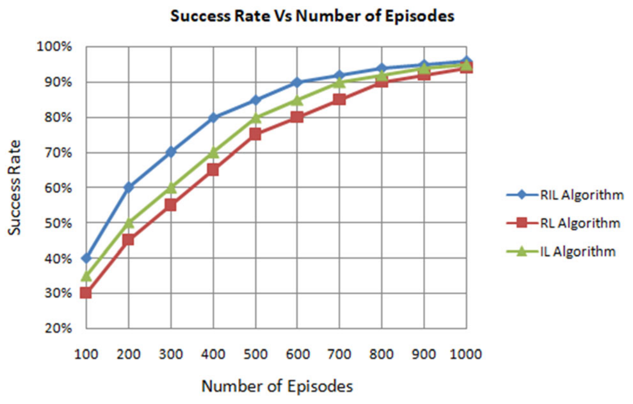

Fig. 6 Success rate versus number of episodes

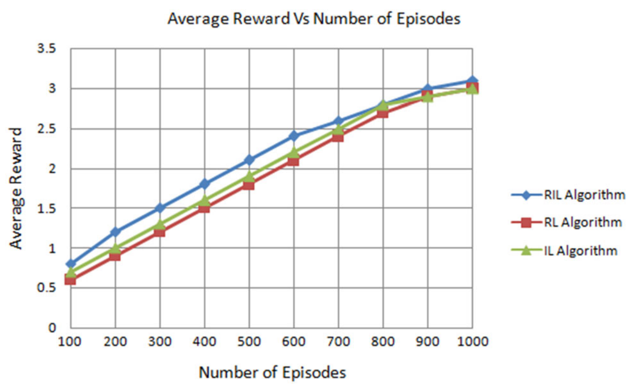

Fig. 7 Average reward versus number of episodes

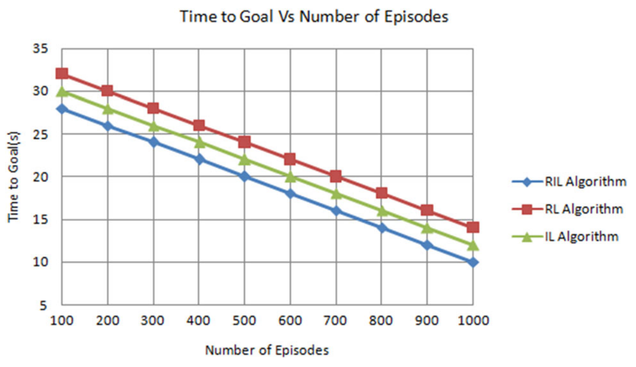

Fig. 8 Time to goal versus number of episodes

consistently outperforms with a higher average reward. The findings also indicate that the RIL algorithm learns more rapidly than the RL algorithm, securing a higher average reward even with fewer iterations.

Time to Goal (s)

Figure 12 shows the results of time to goal (s) for the Reinforcement Imitation Learning (RIL) and Reinforcement Learning (RL) algorithms over a range of iterations from 1000 to 10000.

Figure 12 results show that the RIL algorithm consistently achieves a shorter time to goal than the RL algorithm across all numbers of iterations, with a shorter time to goal in each case. The time to goal decreases for both algorithms as the number of iterations increases, but the RIL algorithm consistently achieves a shorter time to goal. The results also demonstrate that the RIL algorithm is able to learn more quickly than the RL algorithm, achieving a shorter time to goal even at lower numbers of iterations.

Collision Rate

Figure 13 shows the results of collision rate for the Reinforcement Imitation Learning (RIL) and Reinforcement Learning (RL) algorithms over a range of iterations from 1000 to 10000.

Figure 13 displays that the RIL algorithm constantly attains a reduced collision rate compared to the RL algorithm throughout every iteration count, with a lower collision rate in each situation. The collision rate declines for both algorithms as the number of iterations grows, yet the RIL algorithm consistently achieves a lower collision rate. The findings also highlight that the RIL algorithm learns more swiftly than the RL algorithm, reaching a lower collision rate even with fewer iterations.

Success Rate

Figure 14 shows the results of success rate for the Reinforcement Imitation Learning (RIL) and Imitation Learning (IL) algorithms over a range of expert demonstrations from 500 to 5000.

Figure 14 results show that the RIL algorithm consistently achieves a higher success rate than the IL algorithm across all numbers of expert demonstrations, with a higher success rate in each case. The success rate increases for both algorithms as the number of expert demonstrations increases, but the RIL algorithm consistently achieves the highest success rate. The results also demonstrate that the RIL algorithm is able to learn more quickly than the IL algorithm, achieving a higher success rate even at lower numbers of expert demonstrations.

Average Reward

Figure 15 shows the results of average reward for the Reinforcement Imitation Learning (RIL) and Imitation

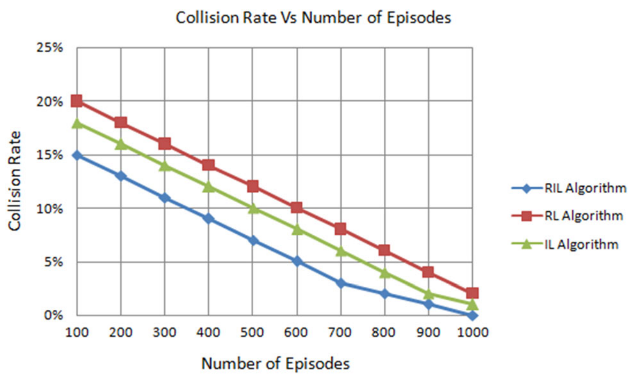

Fig. 9 Collision rate versus number of episodes

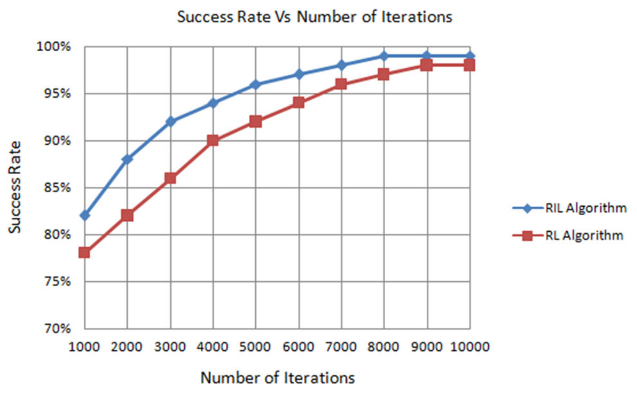

Fig. 10 Success rate versus number of iterations

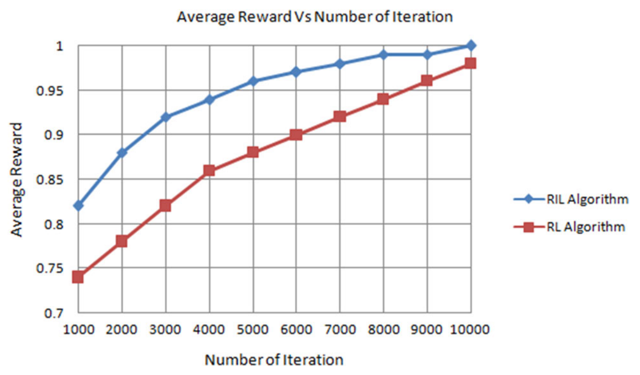

Fig. 11 Average reward versus number of iterations

Learning (IL) algorithms over a range of expert demonstrations from 500 to 5000.

Figure 15 results show that the RIL algorithm consistently achieves a higher average reward than the IL algorithm across all numbers of expert demonstrations, with a higher average reward in each case. The average reward increases for both algorithms as the number of expert demonstrations increases, but the RIL algorithm consistently achieves a higher average reward. The results also demonstrate that the RIL algorithm is able to learn more quickly than the IL algorithm, achieving a higher average reward even at lower numbers of expert demonstrations.

Time to Goal (s)

Figure 16 shows the results of time to goal (s) for the Reinforcement Imitation Learning (RIL) and Imitation Learning (IL) algorithms over a range of expert demonstrations from 500 to 5000.

Figure 16 results show that the RIL algorithm consistently achieves a shorter time to goal than the IL algorithm across all numbers of expert demonstrations, with a shorter time to goal in each case. The time to goal decreases for both algorithms as the number of expert demonstrations increases, but the RIL algorithm consistently achieves a shorter time to goal. The results also demonstrate that the RIL algorithm is able to learn more quickly than the IL algorithm, achieving a shorter time to goal even at lower numbers of expert demonstrations.

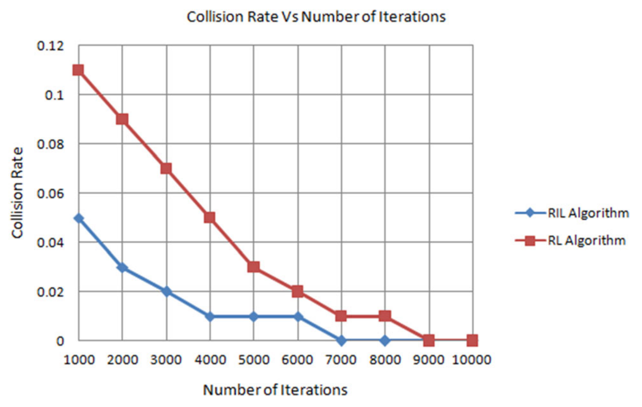

Fig. 13 Collision rate versus number of iterations

Collision Rate

Figure 17 shows the results of collision rate for the Reinforcement Imitation Learning (RIL) and Imitation Learning (IL) algorithms over a range of expert demonstrations from 500 to 5000.

Figure 17 results show that the RIL algorithm consistently achieves a lower collision rate than the IL algorithm across all numbers of expert demonstrations, with a lower collision rate in each case. The collision rate decreases for both algorithms as the number of expert demonstrations increases, but the RIL algorithm consistently achieves a lower collision rate. The results also demonstrate that the RIL algorithm is able to learn more quickly than the IL algorithm, achieving a lower collision rate even at lower numbers of expert demonstrations.

5 Discussion based on results

Key observations and lessons from empirical results of the Reinforcement Imitation Learning (RIL) algorithm indeed show high performance and advantages in dynamic and

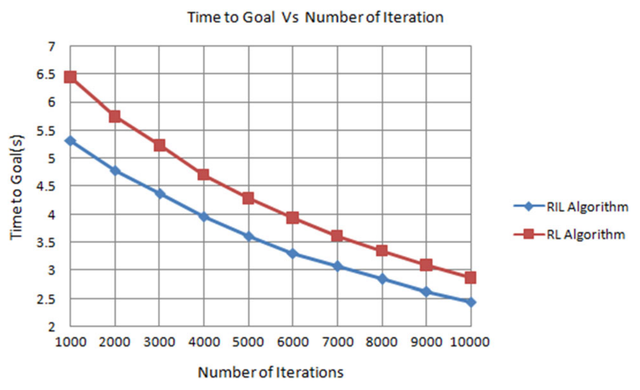

Fig. 12 Time to goal versus number of iterations

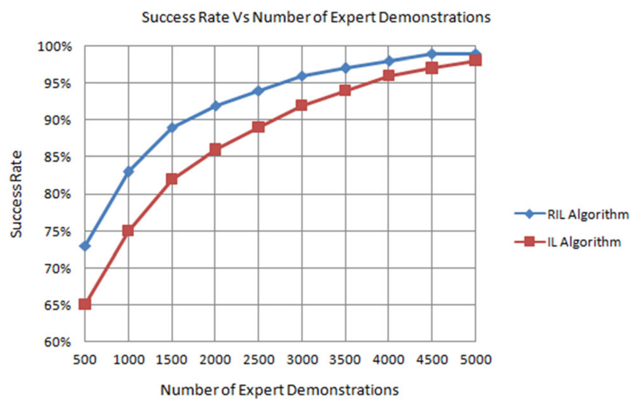

Fig. 14 Success rate versus numbers of expert demonstrations

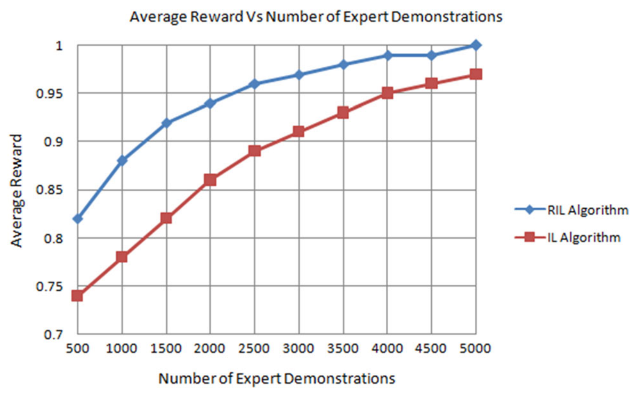

Fig. 15 Average reward versus numbers of expert demonstrations

complex environments. Here are those key observation points and learned lessons summarized:

- *Superiority of RIL Over Individual RL and IL:* The RIL algorithm is superior to the standalone RL and IL algorithms on all the performance metrics. The observation further makes strong the point that RL and IL techniques can be integrated at the concepting stage for maximal performance. The two competing techniques seem to have some synergies which were realized when combined as both contributed their best qualities to offer a superior result than perhaps what an individual method is capable of doing.
- *Success Rate Improvements:* The RIL algorithm has outperformed RL and IL in the enhancement of success rate through a number of episodes as well as iterations. Therefore, the integrated approach is more successful in meeting the goals established in the simulation environment.
- *Higher Average Reward:* In the average reward, RIL outperforms RL and IL also. The adaptiveness of RIL to apply the strategies of RL and IL must offer the efficiency in decision-making for enhancing high rewards.
- *Efficiency in Reaching Goals:* The time to goal exhibited by the RIL algorithm is shorter compared to RL and IL. Such an efficiency in time is imperative in a dynamically changing environment giving an implication that the RIL can adapt to changes fast for it to find effective paths toward objectives.
- *Lower Collision Rate:* RIL shows a lower collision rate in the scattered scenarios preserving its superiority in navigability and obstacle avoidance characteristics. These are vital characteristics for some applications ingraining autonomous systems as the prime concern is about safety and reliability.
- *Scalability with Increasing Complexity:* With increasing complexity in terms of the number of iterations, episodes, and expert demonstrations, the RIL algorithm improved significantly across a wide range of performance metrics. This scalability is testimony to the robustness of the algorithm for enabling learning in complex and dynamic environments.
- *RIL's Quick Learning Ability:* For adaptation in environments where there is quick need for adaptation, the RIL algorithm presents quickly learning curve as compared to RL and IL individually.
- *Real-world Application and Testing:* Translating the success of RIL in simulated environments to real-world scenarios is a crucial next step, involving field tests and real-world data integration.
- *Algorithmic Transparency and Interpretability:* Future research should also address the interpretability of complex algorithms like RIL, ensuring that their decision-making processes are transparent and understandable, which is vital for trust and wider adoption.
- *Adaptive and Context-Aware Systems:* Building on the success of RIL, future systems should aim for greater adaptability and context awareness, allowing them to adjust more fluidly to changing environmental conditions and unforeseen challenges.

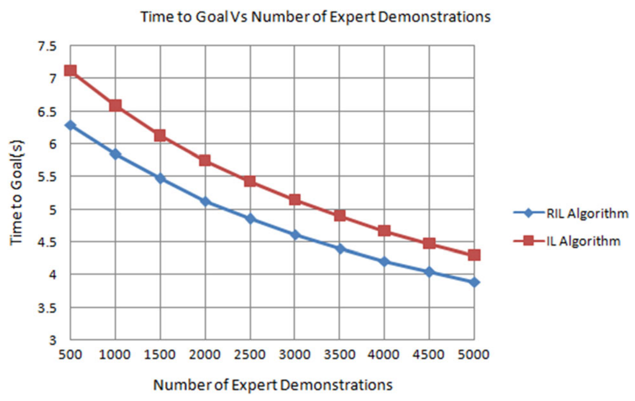

Fig. 16 Time to goal versus numbers of expert demonstrations

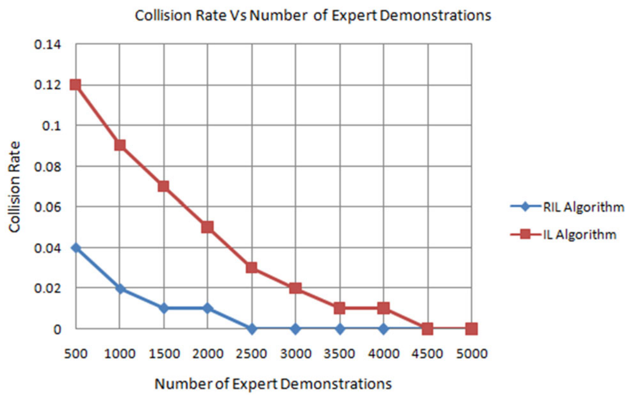

Fig. 17 Collision rate versus numbers of expert demonstrations

6 Conclusion

This paper has reviewed the application of a novel integration of reinforcement learning (RL) and imitation learning (IL) techniques in achieving robust autonomous navigation in dynamic environments. The proposed approach of the Reinforcement Imitation Learning (RIL) algorithm designs a novel way to combine the strengths of RL and IL so that an improved performance is achieved than over either of them. The RIL algorithm has been tested and found far superior in every way on the efficiency of navigation, avoiding collision, and adapting to changing environments in all sorts of different dynamic kinds of environments.

The RIL algorithm has others than improved implications on real world defying purposes like the autonomous vehicles, robotics and surveillance systems. Combining these experience-based learning approaches, expert demonstrations and trial and error allow the RIL algorithm to learn more efficiently while at the same time increasing reliability for autonomous navigation in complex, uncertain environments. In future, different phenomena and environments can be dealt with to enhance the performance and effectiveness of RIL algorithm.

Author contributions DK has contributed to the development of the Reinforcement Imitation Learning (RIL) algorithm for autonomous navigation in dynamic environments. They proposed the RIL algorithm, which integrates reinforcement learning and imitation learning techniques, designed the simulation environment, collected and preprocessed the data, and evaluated the performance of the RIL algorithm. They also conducted the data analysis and interpretation of the results and drafted the manuscript. Overall, the authors' contributions led to the development and evaluation of a novel approach to autonomous navigation that can overcome the limitations of traditional approaches and achieve better performance in dynamic environments.

Funding No funding.

Data availability The datasets generated and analyzed during the current study are available from the corresponding author on reasonable request.

Declarations

Conflict of interest The authors declare no relevant conflicts of interest concerning the content of this article.

Ethical approval Not Applicable.

References

1. Mnih V, Kavukcuoglu K, Silver D, Rusu AA, Veness J, Bellemare M.G, Graves A, Riedmiller M, Fidjeland A.K, Ostrovski G et al (2015) Human-level control through deep reinforcement learning. Nature 518(7540):529–533
2. Argall BD, Chernova S, Veloso M, Browning B (2009) A survey of robot learning from demonstration. Robot Autonom Syst 57(5):469–483
3. Silver D, Hubert T, Schrittwieser J, Antonoglou I, Lai M, Guez A, Lanctot M, Sifre L, Kumaran D, Graepel T et al (2018) A general reinforcement learning algorithm that masters chess, shogi, and go through self-play. Science 362(6419):1140–1144
4. Zhu Z, Zhao H (2021) A survey of deep RL and IL for autonomous driving policy learning. IEEE Trans Intell Transp Syst 23(9):14043–14065
5. Peng P, Barnes M, Wang C, Wang W, Li S, Swanson HL, Dardick W, Tao S (2018) A meta-analysis on the relation between reading and working memory. Psychol Bull 144(1):48
6. Sadeghi F, Levine S (2016) Cad2rl: Real single-image flight without a single real image. arXiv preprint arXiv:1611.04201
7. Liu H, Huang Z, Wu J, Lv C (2022) Improved deep reinforcement learning with expert demonstrations for urban autonomous driving. In: 2022 IEEE intelligent vehicles symposium (IV), pp 921– 928. IEEE
8. Liu H, Liu HH, Chi C, Zhai Y, Zhan X (2020) Navigation information augmented artificial potential field algorithm for collision avoidance in UAV formation flight. Aerosp Syst 3:229–241
9. Kumar D, Pandey M (2020) An effective and secure data sharing in p2p network using biased contribution index based rumour riding protocol (bcirr). Opt Mem Neural Netw 29(4):336–353
10. Kumar D, Dubey AK, Pandey M (2022) Time and position aware resource search algorithm for the mobile peer-to-peer network using ant colony optimisation. Int J Commun Netw Distrib Syst 28(6):621–654
11. Li L, Wu D, Huang Y, Yuan Z-M (2021) A path planning strategy unified with a colregs collision avoidance function based on deep reinforcement learning and artificial potential field. Appl Ocean Res 113:102759
12. Bojarski M Del Testa D, Dworakowski D, Firner B, Flepp B, Goyal P, Jackel LD, Monfort M, Muller U, Zhang J, et al (2016) End to end learning for self-driving cars. arXiv preprint arXiv:1604.07316
13. Kumar D, Pandey M (2022) An optimal load balancing strategy for p2p network using chicken swarm optimization. Peer-to-Peer Netw Appl 15(1):666–688
14. Hausknecht M, Stone P (2015) Deep recurrent q-learning for partially observable mdps. In: 2015 AAAI fall symposium series
15. Pomerleau DA (1991) Efficient training of artificial neural networks for autonomous navigation. Neural Comput 3(1):88–97
16. Ho J, Ermon S (2016) Generative adversarial imitation learning. Adv Neural Inf Process Syst 29
17. Dossa RF, Lian X, Nomoto H, Matsubara T, Uehara K (2020) Hybrid of reinforcement and imitation learning for human-like agents. IEICE Trans Inf Syst 103(9):1960–1970
18. Kumar D, Pandey M (2022) An optimal and secure resource searching algorithm for unstructured mobile peer-to-peer network using particle swarm optimization. Appl Intell 1–18
19. Zhang C, Bengio S, Hardt M, Recht B, Vinyals O (2021) Understanding deep learning (still) requires rethinking generalization. Commun ACM 64(3):107–115
20. Nguyen ND, Nguyen TT, Pham NT, Nguyen H, Nguyen DT, Nguyen TD, Lim CP, Johnstone M, Bhatti A, Creighton D et al (2023) Towards designing a generic and comprehensive deep reinforcement learning framework. Appl Intell 53(3):2967–2988
21. Zhu Y, Mottaghi R, Kolve E, Lim JJ, Gupta A, Fei-Fei L, Farhadi A (2017) Target-driven visual navigation in indoor scenes using deep reinforcement learning. In: 2017 IEEE international conference on robotics and automation (ICRA), pp 3357–3364. IEEE

22. Abbeel P, Ng AY (2004) Apprenticeship learning via inverse reinforcement learning. In: Proceedings of the twenty-first international conference on machine learning, p 1
23. Florensa C, Held D, Wulfmeier M, Zhang M, Abbeel P (2017) Reverse curriculum generation for reinforcement learning. In: Conference on robot learning, pp 482– 495. PMLR
24. Kurutach T, Clavera I, Duan Y, Tamar A, Abbeel P (2018) Model-ensemble trust-region policy optimization. arXiv preprint arXiv:1802.10592
25. Pribeanu C, Balog A, Iordache DD (2017) Measuring the perceived quality of an AR-based learning application: a multidimensional model. Interact Learn Environ 25(4):482–495
26. Xu C, Peng Z, Hu X, Zhang W, Chen L, An F (2020) Fpga-based low-visibility enhancement accelerator for video sequence by adaptive histogram equalization with dynamic clip-threshold. IEEE Trans Circuits Syst I Regul Pap 67(11):3954–3964
27. Sadigh D, Sastry S, Seshia S.A, Dragan A.D (2016) Planning for autonomous cars that leverage effects on human actions. Robot Sci Syst 2:1–9 (**Ann Arbor, MI, USA**)
28. Kendall A, Gal Y (2017) What uncertainties do we need in bayesian deep learning for computer vision?. Adv Neural Inf Process Syst 30
29. Mirowski P, Pascanu R, Viola F, Soyer H, Ballard AJ, Banino A, Denil M, Goroshin R, Sifre L, Kavukcuoglu K, et al (2016) Learning to navigate in complex environments. arXiv preprint arXiv:1611.03673
30. Amodei D, Ananthanarayanan S, Anubhai R, Bai J, Battenberg E, Case C, Casper J, Catanzaro B, Cheng Q, Chen G, et al (2016) Deep speech 2: end-to-end speech recognition in english and mandarin. In: International conference on machine learning, pp 173– 182. PMLR
31. Such FP, Madhavan V, Conti E, Lehman J, Stanley KO, Clune J (2017) Deep neuroevolution: genetic algorithms are a competitive alternative for training deep neural networks for reinforcement learning. arXiv preprint arXiv:1712.06567
32. Zhang S, Yao L, Sun A, Tay Y (2019) Deep learning based recommender system: a survey and new perspectives. ACM Comput Surv (CSUR) 52(1):1–38