Acta Astronautica 224 (2024) 37–47

Contents lists available at ScienceDirect

Acta Astronautica

journal homepage: www.elsevier.com/locate/actaastro

Research Paper

Efficient adversarial attacks detection for deep reinforcement learning-based autonomous planetary landing GNC

Ziwei Wang [*], Nabil Aouf

Department of Engineering, City, University of London, Northampton Square, London, EC1V 0HB, United Kingdom

ARTICLE INFO

Keywords:
Adversarial Attack
Adversarial Attack Detection
Deep Reinforcement Learning
Explainable Artificial Intelligence
Navigation, Guidance and Control

ABSTRACT

Given the constraints of remote communication and the unpredictability of the environment, autonomous planetary landing mechanisms are expected to achieve the high criteria of autonomy and provide optimal trajectory in future space exploration missions. As the results, applying Deep Reinforcement Learning (DRL) techniques into autonomous landing has produced encouraging findings. Due to the black-box nature of deep learning algorithms, one of the main concerns regarding the robustness of DRL is its vulnerability to adversarial attacks. This constraint prevents the transfer of DRL-based autonomous landing schemes from simulation to real-world applications. In this article, we explore how the DRL-based autonomous landing will be impacted by adversarial attacks and how to protect the system effectively and efficiently. To achieve this, a Long Short Term Memory (LSTM) based adversarial attack detector is been proposed. The proposed method adopts the explainability measurement of the target DRL scheme and flag the detection of adversarial attacks when acting. The proposed method is built and tested on 3D digital terrain model of Candidate Landing Site for 2020 Mission in Jezero Crater to simulate the landing scenario on the Mars. The experimental results demonstrate the proposed methodology can effectively detect adversarial attacks when acting on DRL agent with a high confidence in detection accuracy.

1. Introduction

Autonomous planetary landings are critical events in space exploration, where spacecraft must safely descend onto the surface of celestial bodies such as Mars, the Moon, or asteroids. Traditional landing systems rely on pre-programmed trajectories and sensor-based feedback to guide the descent. However, these methods can be limited by uncertainties in the environment, unpredictable terrain, and unforeseen obstacles [1,2].

Artificial Intelligence (AI) has made remarkable progress in tackling intricate challenges over the last decade. Whilst growing the number of studies are being conducted on integrating such approaches into space exploration missions [3], recent studies have shown encouraging outcomes when applying Deep Reinforcement Learning (DRL) techniques to achieve autonomous planetary landings [4–6]. In DRL-based autonomous planetary landing, the problem is formulated as a Markov sequential decision-making problem, where an agent (lander) continuously interacts with the environment and learns to make decisions based on sensory input, such as camera images and radar measurements. Then, the agent receives rewards or penalties based on its actions. Over the time, the agent refines its policy to achieve safe and optimal landing trajectories.

Today, visual camera is one of the preferred sensors in many autonomous applications because of its excellent performance and affordability in comparison to other types of sensors, such as radars and LiDARs [7]. Recent research also tries to bring visual camera as primary sensor into various space exploration missions, such as spacecraft relative pose estimation [8] and autonomous planetary landing [9]. However, one of the main concerns regarding the robustness of Deep Learning (DL) techniques is its vulnerability to adversarial attacks [10]. Adversarial attacks aim to make small perturbations to the input images that are imperceptible to human vision and can significantly affect the decision made by the model [11]. The danger of adversarial attacks for DRL-based autonomous planetary landing guidance and control is that it can compromise the safety and performance of the lander and cause the failure of landing on the ground. For example, an attacker could temporarily perturb the images that the DRL receives from its camera, such that the lander would misclassify the terrain and resulting in the lander colliding with the terrain. Therefore, it is critical to develop effective and efficient methods to safeguard the DRL-based autonomous landing systems from adversarial attacks, and to ensure the robustness and resilience of the autonomous landing system in future space exploration missions.

[*] Corresponding author.
E-mail addresses: ziwei.wang.3@city.ac.uk (Z. Wang), nabil.aouf@city.ac.uk (N. Aouf).

https://doi.org/10.1016/j.actaastro.2024.07.052
Received 24 May 2024; Received in revised form 2 July 2024; Accepted 28 July 2024
Available online 2 August 2024

Owing to the black-box nature of the algorithms, DRL models typically exhibit a lack of transparency. This results in a challenge for people to comprehend the decision-making processes of these model. To address this challenge, various eXplainable AI (XAI) techniques are proposed to make the decision-making processes of AI models comprehensible [12–14]. By applying XAI methods to DRL, users can understand the basis on which decisions are made by the DRL models in response to relevant observations. The mechanisms employed by XAI methods elevate the potential for their utilisation in detecting adversarial attacks on DRL models.

This work endeavours to showcase an innovative exploration of the susceptibility of DRL-based autonomous planetary landing guidance and control systems against digital adversarial attacks targeting camera input images. Furthermore, it proposes a novel methodology for the detection of such adversarial attacks upon their occurrence. To this end, this paper makes the following contributions:

- Firstly, a 3D simulator for vision-based planetary landing is introduced. The simulator is not only rendering the camera view of candidate landing site, but also involves physical interactions between the lander and the environment to accurately simulate the planetary landing scenario.
- Secondly, we design and train a vision-based DRL scheme for Mars landing guidance and control, which is subsequently formulated as the target DRL system against adversarial attacks.
- Then, the Fast Gradient Sign Method (FGSM) [15] is utilised to generate invisible perturbations in the input images, introducing a range of FGSM attack configurations to illustrate the effects of digital adversarial attacks on the DRL scheme.
- Subsequently, an LSTM-based detection mechanism is proposed, leveraging the explainable SHapley Additive exPlanation (SHAP) values [12] from the DRL-based landing guidance and control system to identify adversarial attacks affecting the input images.

The paper is organised as follows: Section 2 provides an overview of current DRL-based autonomous planetary landing algorithms and discusses existing methods for detecting adversarial attacks. Section 3 outlines the proposed design of the vision-based DRL Mars landing guidance and control scheme, the adoption of FGSM attacks on the pose estimator, and the design of the LSTM-based adversarial attack detection scheme. Section 4 presents the test experiment results of the proposed adversarial attack detector. Finally, Section 5 concludes the paper and discusses avenues for future work.

2. Background and related works

This section introduces a review of the current approaches of DRL-based autonomous Planetary landing approaches, XAI and adversarial attacks for DRL models.

2.1. DRL-based autonomous planetary landing

Gaudet et al. [5] proposed a six Degree of Freedom (DOF) Mars landing scheme by Proximal Policy Optimization (PPO) [16]. PPO is a type of on-policy reinforcement learning algorithm that the agent learns from the same policy it uses to interact with the environment. Typically, PPO employs a clipping mechanism within the objective function to constrain the policy update to a specified range, as shown in Eq. (1).

$$L^{CLIP}(\theta) = \hat{E}_t\left[\min(r_t(\theta)\hat{A}_t, \text{clip}(r_t(\theta), 1-\varepsilon, 1+\varepsilon)\hat{A}_t)\right] \tag{1}$$

This precaution prevents substantial alterations that could otherwise result in performance deterioration. This approach enhances PPO's stability and efficiency in handling large-scale optimisation problems, as well as its robustness to hyperparameter settings. This DRL-based landing scheme proposed by Gaudet et al. [5] aims to provide direct commands to each thruster, enabling the lander to successfully reach the target location with the desired velocity and minimal fuel consumption. To achieve this, the DRL scheme takes the estimated lander's states as its observation, including the error between expected velocity and current velocity, lander's attitude and altitude, the angular velocity, and estimated time to land. This work demonstrated that DRL is capable of providing guidance and control to enable the lander to perform a soft landing at a desired pose and velocity, even with a large various distance capability.

Xu et al. [17] demonstrated DRL-based 6 DOF landing control framework and compared the performance of three off-policy DRLs, including Deep Deterministic Policy Gradient (DDPG) [18], Twin Delayed DDPG (TD3) [19] and Soft Actor Critic (SAC) [20]. DDPG employs a deterministic policy to select actions and a critic network to estimate the Q-value. As an off-policy DRL algorithm, DDPG reuses past experiences sampled from previous interactions with the environment to enhance sample efficiency. TD3 is an extension of DDPG which aims to reduce the overestimation. TD3 introduces twin Q-networks to reduce the overestimation of Q-values. It employs a delayed policy update method to stabilise training process and proposes target policy smoothing to reduce variance. Unlike DDPG and TD3, SAC employs a stochastic policy and incorporates an entropy regularisation mechanism. SAC aims to maximise both the expected return and the entropy of the policy, thereby encouraging the agent to engage in more exploration during the early training phase. SAC has demonstrated substantial performance improvements in various DRL tasks, especially in complex environments with high-dimensional observation and action spaces. The framework introduced by Xu et al. [17] defined the DRL's action space as the thrust outputs of the engines, while the observation space encompasses the lander's velocity deviation, relative position, and attitude, as well as the sine and cosine values of the attitude angles. The comparative analysis by Xu et al. [17] under identical simulation conditions suggests that SAC outperforms DDPG and TD3 in tracking the trajectory with the reference velocity during soft landing scenarios.

In addition to utilising the lander's numerical status (position, attitude, velocity, etc.) as the DRL's observation, recent studies have also investigated enhancing the observation space with information from real sensors.

Scorsoglio et al. [9] combined the images taken by the onboard camera with altimeter data (position, velocity, estimated time to land) of the lander as observations for the DRL policy, considering a 3 DOF lunar soft landing scenario. To obtain an accurate image view of the landing site (terrain), they built a landing simulator in Blender to render the images obtained from the onboard camera. Their experimental results show that combining optical and altimeter data enables autonomous landing on planetary bodies. Besides the image and altimeter data, Ciabatti et al. [6] considered incorporating LiDAR input into the DRL's observations. In their DRL framework, the policy takes a combination of RGB image views of the terrain, LiDAR point cloud data indicating the distance between the lander and the terrain, and altimeter data of the lander's relative position and attitude. They also proposed a transfer learning approach in which the DDPG is initially trained on a simulated lunar landing site and then transferred to other planetary landing sites.

2.2. Explainability in deep learning

XAI is a technique that offers insights into the decision-making processes of DL models. The objective of XAI is to establish a transparent framework in which AI decisions are comprehensible and trustworthy to humans. It seeks to demystify the “black box” nature of DL models by delivering explanations that are intelligible to human experts, thus promoting accountability and trust in AI systems.

Lundberg and Lee introduced SHAP values as a method for interpreting machine learning models, that drawing on the concept of Shapley values from game theory [12]. SHAP values assign an importance score to each feature for a given prediction, thereby highlighting each feature's importance. Through the analysis of SHAP values, humans can understand how predictions of complex machine learning models are made by considering SHAP values of each input feature.

Contrastive Gradient-based (CG) saliency map is introduced by Simonyan et al. [21], which serves as a visual explanation tools for DL models. This method generates a heatmap where the magnitude of the model's gradients signifies the importance of input variables. The heatmap accentuates regions in the input image that, if altered, would affect the output class. Users can consult the heatmap to discern the features most critical to the model's prediction.

Similar to CG saliency map, Class Activation Mapping (CAM) [22] creates visual explanation maps by identifying the spatial locations in the input image that are most influential to a particular prediction. CAM proves especially useful for image classification tasks using Convolutional Neural Networks (CNNs). Building upon this, Gradient-weighted Class Activation Mapping (Grad-CAM) by Selvaraju et al. [23] offers visual explanations for a broader spectrum of CNN-based methods. Grad-CAM employs the gradients of any target concept channelling into the final convolutional layer to generate a localisation map, spotlighting the pivotal areas in the input image for predicting the concept. These XAI techniques elucidate the workings of CNNs, enabling a clearer understanding of their predictive behaviours.

2.3. *Adversarial attacks on DRL and detection*

Ilahi et al. [24] provide a comprehensive review of recent approaches to adversarial attacks on DRL models, as well as the current methodologies for defending these attacks. Lin et al. [25] proposed a frame prediction module to detect adversarial attacks and provide action suggestions to maintain DRL's performance. This method detects adversarial examples by comparing the action distributions from both the current observed frame and a predicted frame. If an adversarial attack is detected, the agent will make actions based on the predicted frame rather than the adversarial frame. The experimental results demonstrate that the method can achieve an accuracy of 60%–100%, depending on the technique used to generate adversarial examples.

Havens et al. [26] examine a scenario in which a policy learning process is subjected to adversarial attacks at predetermined intervals. They propose a Meta Learned Advantage Hierarchy (MLAH) framework wherein the agent simultaneously learns both nominal and adversarial sub-policies, enabling it to detect adversarial attacks effectively.

Xiang et al. [27] proposed an adversarial attack detection method for an automatic pathfinding task, which considers five factors: energy point gravitation, key point gravitation, path gravitation, included angle, and the placid point. By analysing these factors, they were able to calculate the probabilities of an adversarial attack on the robot. Their experimental results show that the proposed adversarial attack detector achieves a detection precision of approximately 70% on their target task.

Previous work from our group, Hickling et al. [28] introduced both a CNN-based and an LSTM-based adversarial attack detector for DRL-guided Uncrewed Aerial Vehicles (UAVs). Simulation results indicate that the LSTM-based detector achieves a 90% detection accuracy within the DRL model. Furthermore, it outperforms the CNN-based detector in terms of both accuracy and speed, fulfilling the real-time requirements essential for DRL-based UAV guidance.

Based on the review of recent approaches in this section, the adoption of DL-based techniques for future space exploration offers significant advantages to researchers and engineers. However, the vulnerability of such DL schemes can be a bottleneck when deploying these techniques into real applications. To the best of our knowledge, there is currently no literature examining the impact of adversarial attacks on DRL-based planetary landing GNC, nor how to detect these adversarial attacks in this scenario. This work proposes, for the first time, an adversarial attack detection scheme to address the detection of adversarial attacks in DRL-based planetary landing GNC.

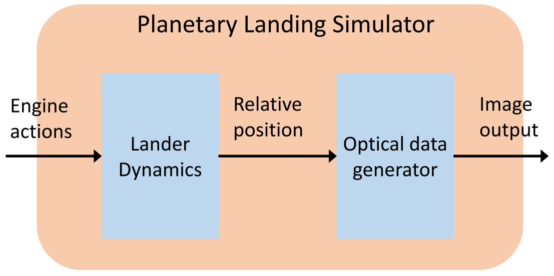

Fig. 1. Dataflow of the planetary landing simulator.

3. Methodology

In this section, we introduce a newly designed planetary landing simulator, developed to generate reliable vision data and enable physical interactions between the DRL actions and the environment in a predetermined planetary landing scenario. A monocular vision-based DRL system has been trained to provide guidance and control, facilitating a soft landing at the targeted position and velocity. Following this, FGSM attacks are employed on the optical input data to produce an adversarial image, which serves to assess the impact on the DRL system. SHAP values are utilised to create XAI signatures for both the adversarial and normal input images. Lastly, we propose and train an LSTM-based adversarial attacks detector that learns to discern normal and adversarial SHAP values, effectively detecting adversarial attacks on the vision-based DRL system.

Our objective is to ultimately develop an adversarial attack detection scheme for the DRL-based planetary landing GNC system. This system employs the SHAP values explainability mechanism from the DRL agent to interactively detect and flag potential adversarial attacks while the DRL agent is in operation.

3.1. *Planetary landing simulator*

The planetary landing simulator consists of two main components: a optical data generator and a 3 DOF lander dynamics. The 3 DOF controller takes the engine actions command as its inputs and outputs the relative position of the lander, while the optical data generator takes the relative position and outputs the relevant vision view. Fig. 1 demonstrates the top-level design of the planetary landing simulator.

3.1.1. *Optical data generator*

To generate reliable vision data for the planetary landing scenario, a 3D model was constructed using Blender [29], which is an open-source suite specifically designed for 3D modelling, animation, and rendering. Both a HiRISE Digital Terrain Model (DTM) and textures of the Candidate Landing Site for the 2020 Mission in Jezero Crater on Mars were imported to simulate a realistic ground model [30]. The candidate landing site is depicted with dimensions of approximately 14 km × 7 km in the x and y direction. Furthermore, we enhanced the surface texture with additional orange colouration to improve the realism of the candidate landing site's appearance. Fig. 2 illustrates the rendered texture and view of the 3D terrain.

The lander's onboard camera is oriented towards the ground. This camera is configured to have a focal length of 35 mm, a sensor size of 36 mm × 36 mm. This camera configuration is determined by the size limitations of the selected DTM model and landing range. The camera's field of view (FOV) is designed to fit within the dimensions of the DTM model, allowing the DRL agent to sufficiently explore the terrain without the FOV extending beyond the terrain model. Fig. 3(a) illustrates an example of the relative camera's FOV at 1200 m above the landing site. The camera output is configured for 24 × 24 RGB images. Despite the low resolution of our configuration, the DRL still

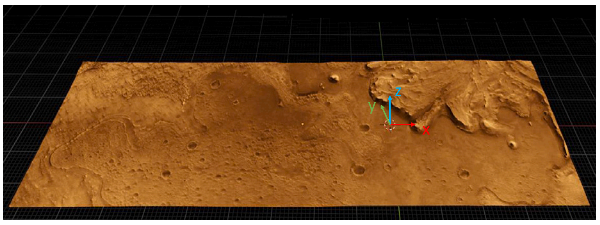

Fig. 2. Rendered view of the candidate landing site for the 2020 Mission in Jezero Crater. The origin is the expected landing position with a relative position of (0,0,0). The red and green arrows indicate the x and y directions and z-axis is shown in blue and aligned upward to the ground. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

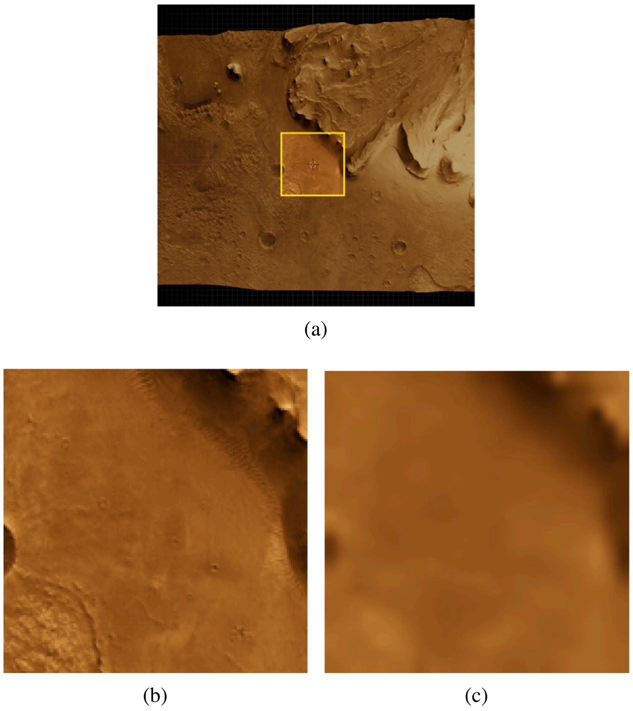

Fig. 3. Example of rendered camera view at 1200 m above the origin. (a) The yellow square shows the camera's FOV under our current configuration. (b) Image resolution of 1024 × 1024 pixels. (c) Image resolution of 24 × 24 pixels. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

demonstrates good performance in test experiments. Additionally, both the DRL training process and the Blender rendering process benefit from the low-resolution output, as it contains less learnable parameters and requires fewer computational resources. Therefore, in this case, we decided to configure the output resolution as 24 × 24. Figs. 3(b) and 3(c) illustrate the rendered camera output from Blender.

3.1.2. Lander dynamics

In this paper, the physical motion of the 3 DOF Mars landing guidance and control system is considered, while the spacecraft's attitude, its orientation relative to the flight path, are not considered. Instead of the relative position of the lander being controlled by the individual action of each thrust, we formulate the relative position of the lander that is controlled by the forces of the thrusts in x, y and z directions. The simulator takes into consideration mass variations although assuming a constant mass distribution, i.e. the centre-of-gravity of the lander is fixed with respect to the geometric frame. The relative position of the lander, controlled by the thrusts, is then given by Eqs. (2)–(6).

$$P_t = P_{t-1} + v_t \times \delta t \tag{2}$$

$$v_t = v_{t-1} + \delta v \tag{3}$$

$$\delta v = \delta a \times \delta t \tag{4}$$

$$\delta a = \frac{F}{m} \tag{5}$$

$$F = F_{Thr} + F_{aero} \tag{6}$$

where the P_t and P_{t-1} indicate the relative position of the lander at time t and $t-1$. v_t and v_{t-1} denote the velocity at time t and $t-1$, respectively. δv represents the change in velocity, δa represents the acceleration at time δt and m represents the lander's mass. F, F_{Thr} and F_{aero} indicate the resultant force, thrusts' forces and aerodynamic drag forces, respectively. The aerodynamic drag forces, F_{aero}, are calculated based on the atmospheric properties retrieved directly from the Mars Climate Database (MCD) v.5.2 [31]. Typically, the atmospheric properties in our simulator involve the atmosphere density, speed of sound, wind velocity along x direction, wind velocity along y direction and wind velocity along z direction. The aerodynamic drag forces are then computed by Eq. (7).

$$F_{aero} = \frac{1}{2}\rho V^2 C_D(Ma)S \tag{7}$$

where ρ is the atmosphere density, C_D is the lander drag coefficient as a function of the Mach number (Ma), S is the lander reference area, and V is the lander velocity relative to the fluid velocity expressed in the landing site frame.

In our implementation, we assume that the engines can provide thrust forces along the x, y and z directions. In the x and y directions, the thrust forces can be positive or negative, within the range of the maximum thrust's force and minimum thrust's force. In the z direction, the thrust force only operates downward to the ground, where the thrust force should be between 0 and F_{MAX}. In this case, the fuel consumption is computed by integrating the mass flow rate, as shown in Eq. (8).

$$\delta m = \frac{\|F\|}{g_0 I_{sp}} \tag{8}$$

where δm indicates the fuel consumption. $\|F\|$ is the normalised thrust forces. g_0 and I_{sp} donate the standard gravity and specific impulse of the engine, respectively.

3.2. Vision-based soft landing DRL

The soft landing controller can be treated as a Markov Decision Problem, where the agent chooses an action $a_t \in \mathcal{A} \subset \mathbb{R}$, corroding to the input observation state, $s_t \in \mathcal{S}$, at the time step t. Then, the agent receives a reward, r_t and the new observation state, s_{t+1}.

To address the continuous action space of the planetary landing scenario, we employ the SAC framework, which is proposed by Haarnoja et al. [20]. The SAC is an off-policy actor–critic model that utilises a stochastic policy. It is designed to maximise the entropy of the policy, thus promoting exploration. The temperature parameter α determines the relative importance of the entropy term versus the reward in the objective function, balancing the trade-off between exploration (entropy) and exploitation (reward). A higher value of α emphasises exploration, encouraging the policy to assign probabilities to a variety of actions, while a lower value of α emphasises the reward, fostering a more deterministic policy that focuses on actions with higher expected returns. In this case, the objective function of the SAC framework can be seen as:

$$J(\pi) = \mathbb{E}_{(s_t,a_t)\sim\rho_\pi}\left[\sum_t \gamma^t (r(s_t,a_t) + \alpha\mathcal{H}(\pi(\cdot|s_t)))\right] \tag{9}$$

where $J(\pi)$ is the objective function, ρ_π denotes the state–action distribution according to policy π, $r(s_t,a_t)$ is the reward function, γ is the discount factor, α is the temperature parameter, and $\mathcal{H}$ represents the entropy of the policy.

Additionally, the SAC utilises the 'clipped double-Q trick' [19] to mitigate overestimation bias in value approximation. With high entropy in the policy, the aim is to explicitly encourage exploration. This approach motivates the policy to assign equal probabilities to actions with identical or nearly identical Q-values. It also ensures that the policy does not collapse into a pattern of repeatedly selecting a specific action, which could exploit any inconsistencies in the approximated Q function. Hence, the Q-network in SAC will be represented as:

$$Q(s_t, a_t) = r(s_t, a_t) + \gamma \mathbb{E}_{s_{t+1} \sim p}[\min_{i=1,2} Q_{\theta_i}(s_{t+1}, a_{t+1}) - \alpha \log \pi(a_{t+1}|s_{t+1})] \quad (10)$$

where Q_{θ_i} are the critic networks parameterised by θ_i and p represents the transition probability.

The parameters of the policy π and the Q-networks are updated through gradient ascent and descent, respectively. The policy parameters are updated to maximise the expected return plus entropy, while the Q-network parameters are updated to minimise the Bellman residual. The updates can be expressed as:

$$\nabla_\phi J(\pi_\phi) = \mathbb{E}_{s_t \sim \rho_\pi, a_t \sim \pi_\phi}[\nabla_\phi \log \pi_\phi(a_t|s_t)(Q_\theta(s_t, a_t) - \alpha \log \pi_\phi(a_t|s_t))] \quad (11)$$

$$\nabla_\theta J(Q_\theta) = \mathbb{E}_{(s_t,a_t)\sim \mathcal{D}}[\nabla_\theta(Q_\theta(s_t, a_t) - (r(s_t, a_t) + \gamma \mathbb{E}_{s_{t+1}\sim p}[\min_{i=1,2} Q_{\theta_i}(s_{t+1}, a_{t+1}) - \alpha \log \pi(a_{t+1}|s_{t+1})]))^2] \quad (12)$$

where $\mathcal{D}$ is the replay buffer from which mini-batches are sampled.

3.2.1. DRL model

Recent investigations involving the integration of optical data into DRL observations still require additional state information from the environment, such as LiDAR [6] and the altimeter data [6,9]. In our approach, the DRL framework solely relies on vision data for its observations, which can potentially reduce the power consumption and computational needs of the required sensors.

At each time step, the proposed DRL takes the image at current time step, I_t, and image at last time step, I_{t-1} as its observation state, as shown in Eq. (13).

$$obs_t = [I_t, I_{t-1}] \quad (13)$$

The feature extractor consists a CNN with three convolutional layers. Each convolutional layer employs 3 × 3 convolution kernels with a stride of 1, and padding is applied. This is followed by a LeakyReLU activation function with a negative slope of 0.01. The outputs of the last convolutional layer for I_t and I_{t-1} are concatenated and then fed into three fully-connected (FC) layers, which output a 1-D tensor with a size of 16. The output tensor serves to further process the features for the subsequent decision-making steps of the SAC model. Fig. 4 demonstrates the overall design of the CNN-based feature extractor.

Then, the SAC model takes the output from the feature extractor and feeds it into its critic and actor networks. In this case, the SAC model employs different architectures for the critic and actor networks. Typically, the actor network is marginally deeper than the critic network. Our rationale for adopting a deeper actor is that it may facilitate better exploration–exploitation trade-offs, given that a deeper network introduces more nonlinearity. Specifically, in our approach where the image serves as the sole input, we employed for a deeper actor to manage the rich features of image-based input, thereby it could potentially achieve an improved exploration–exploitation trade-off. Fig. 5 illustrates the integrated SAC agent of feature extractor, the critic network and the actor network.

Due to the texture quality of the DTM model in the proposed planetary landing simulator, the images captured by the camera may become blurred as the lander descends closer to the ground. This could potentially challenge the feature extractor's ability to handle varying qualities of input images. Additionally, from a practical perspective, the camera may not consistently maintain image quality over a long descent range. The details of the captured images can change as the lander descends closer to the ground. For instance, the perspective from which the camera views the landscape could change as the lander descends. This can reveal new details that were not visible from a higher altitude. Employing different DRL agents to continuously provide guidance over the trajectory could also enhance system performance. In this context, we propose to employ three separate SAC agents to manage the landing process. Whilst other altitude thresholds could also be considered, in our current solution, we have chosen 400 m and 30 m as the thresholds for different agents to act, based on our observations of the rendered outputs. Fig. 6 illustrates the integration of the three SAC agents with the planetary landing simulator.

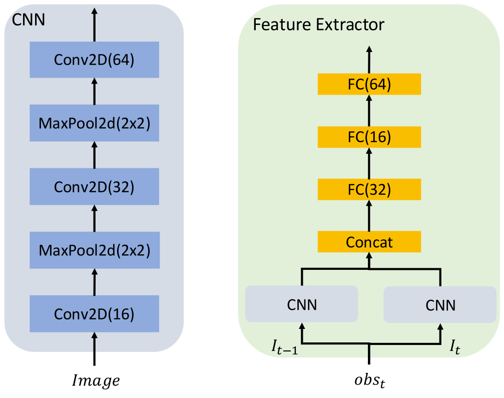

Fig. 4. Feature extractor for handling image input.

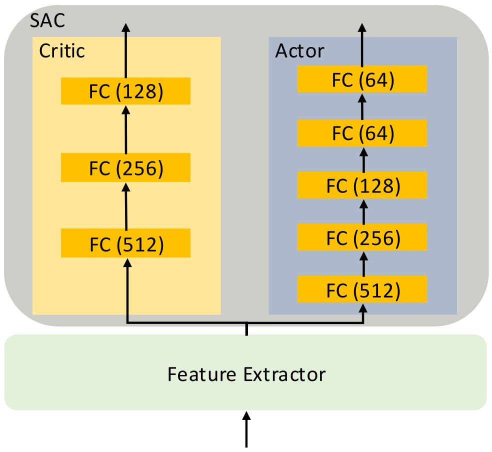

Fig. 5. Integrated SAC model of the feature extractor, the critic network and the actor network.

3.2.2. Reward function

To achieve a soft landing, first we trained the agent to maintain a steady descent velocity. Then, a relative reward will be given based on the difference between the lander's velocity and the expected velocity. Fuel composition is also considered to allow the agent to output accurate engine control commands while minimising fuel consumption over the trajectory. The overall reward function for training the proposed SAC agents can be formulated as:

$$r = \alpha r_{velo} + \beta r_{fuel} + r_{lost} + r_{bonus} + r_{success} \quad (14)$$

where r_{velo} is the reward for maintaining the desired velocity and r_{fuel} is reward based on the fuel consumption. r_{lost} represents a penalty

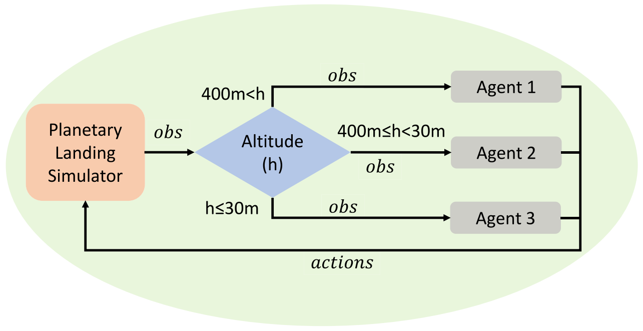

Fig. 6. Integrated three SAC agents with the planetary landing simulator. Each agent operates within its designated altitude range, ensuring effective handling of the varying image quality during the descent.

applied if the agent's observation moves out of the simulated area. $r_{success}$ donates the bonus to be given if the agent reaches the target position with expected soft landing velocity and r_{bonus} is the extra bonus given if lander's current velocity is within certain range of expect velocity. α and β are two weights for relevant reward item.

To calculate the velocity reward, we follow the methodology provided by Gaudet et al. [5], where the lander is expected to follow an estimated velocity at each time step. In this case, the r_{velo} will be calculated by:

$$r_{velo} = \|v - v_{target}\| \tag{15}$$

$$v_{target} = -v_0(\frac{\hat{p}}{\|\hat{p}\|})(1 - exp(-\frac{t_{go}}{\tau})) \tag{16}$$

$$t_{go} = \frac{\|\hat{p}\|}{\|\hat{v}\|} \tag{17}$$

$$\hat{p} = \begin{cases} p - [0,0,15], & p_z > 15 \\ [0,0,p_z], & otherwise \end{cases} \tag{18}$$

$$\hat{v} = \begin{cases} v - [0,0,-2], & p_z > 15 \\ v - [0,0,-1], & otherwise \end{cases} \tag{19}$$

$$\tau = \begin{cases} 20, & p_z > 15 \\ 100, & otherwise \end{cases} \tag{20}$$

where the velocity reward is calculated by normalise the difference between lander's current velocity, v, and the target velocity, v_{target}. v_0 donates the lander's initial velocity, p indicates the lander's current position and τ is a hyperparameter used to make the lander achieve the soft landing when close to the ground. When the lander's current velocity is within a certain magnitude, r_{bonus} will be given by following conditions:

$$r_{bonus} = r_x + r_y + r_z \tag{21}$$

$$r_x = \begin{cases} 0.05, & |v_x - v_{target_x}| < |0.2v_{target_x}| \\ 0, & otherwise \end{cases} \tag{22}$$

$$r_y = \begin{cases} 0.05, & |v_y - v_{target_y}| < |0.2v_{target_y}| \\ 0, & otherwise \end{cases} \tag{23}$$

$$r_z = \begin{cases} 0.05, & |v_z - v_{target_z}| < |0.2v_{target_z}| \\ 0, & otherwise \end{cases} \tag{24}$$

Other parameters in the reward function, Eq. (14), are described in Table 1.

Table 1
Reward function parameters.

Parameter	Value	Condition
α	−0.1	$p_z > 30$
	−0.5	$p_z \leq 30$
β	−0.05	
r_{fuel}	Eq. (8)	
r_{lost}	−100	Pure black pixel in image ≥ 20%
	0	Pure black pixel in image < 20%
$r_{success}$	40	$\|\|p\|\| < 5$ and $p_z \leq 0$ and $\|\|v\|\| < 2$
	0	otherwise

3.3. Adversarial attacks

In this paper, the adversarial examples are produced by utilising FGSM attacks. [15]. The objective of FGSM attacks is to introduce minimal distortions to the original input images, thereby maximising the loss function of the neural network. The adversarial generation process for an input image x through an FGSM attack is calculated by Eq. (25).

$$x' = x + \epsilon \times sign(\nabla_x L(\theta, x, y)) \tag{25}$$

where ϵ represents the magnitude of the perturbation, which quantifies the intensity of the attack. L denotes the loss associated with the input x and the corresponding target output y. The term $\nabla_x L$ computes the gradient of the loss with respect to the input image x, given the true label y, while θ symbolises the parameters of the trained model. In this case, the efficacy of the FGSM attack can be modulated by adjusting the value of ϵ.

In practical applications, the perturbation parameter ϵ must be sufficiently small to render the alterations on the input image imperceptible to the human eye, yet substantial enough to markedly affect the decision-making of the DRL model. The value of ϵ is constrained within the interval $[0, 1]$. Specifically, a value of 0 indicates that the adversarial image remains identical to the original input image, devoid of any perturbation. Conversely, a value of 1 signifies that the adversarial image will exhibit perturbations that are significantly distorted to human vision. Fig. 7 illustrates an example of applying FGSM attacks to onboard camera view of the proposed vision-based soft landing DRL at position of $[-1200, 100, 1200]$ m with $\epsilon = 5/255$.

3.4. Explanability and adversarial attacks detector

3.4.1. Explanability via DeepSHAP

The nature of DL models often leaves users with mere predictive outcomes, devoid of insights into the underlying reasons for their accuracy or errors. To bridge this gap, XAI techniques have been developed to interpret the decision-making processes of neural networks models. These techniques provide interpretative insights when there is a shift in the model's predictions. In this research, we introduce an innovative adversarial attacks detection method that leverages XAI techniques. This method utilises the variation in SHAP values of input images as an indicator to detect the occurrence of adversarial attacks.

Building upon the foundational work of DeepLIFT, as presented by Shrikumar et al. [32], user approximates Shapley values for deep neural networks. This approximation, known as DeepSHAP, is facilitated by linearising the network's non-linear components. The process involves employing a reference input distribution, which allows for a linear approximation to calculate the expected model's value.

Nevertheless, the direct computation of SHAP values for image-based inputs is computationally expensive due to the very high pixel count in the input images. Employing DeepSHAP necessitates the derivation of Shapley values for each individual pixel across all output neurons. To circumvent this computational cost, our approach calculates SHAP values for the output from the feature extractor, rather than

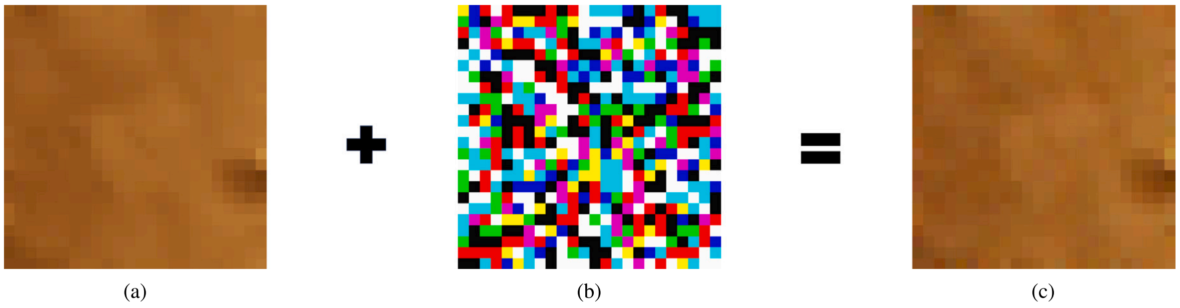

Fig. 7. An example of applying FGSM attacks to the input image. (a) The original input image. (b) Perturbation patch with $\epsilon = 5/255$. (c) Resultant adversarial image.

the input image itself. As previously illustrated in Fig. 5, our envisioned vision-based DRL model incorporates a feature extractor that reduces the feature maps from the preceding convolutional layer to a 16-feature output. Consequently, SHAP values are computed solely for these 16 feature outputs. This saving in the computation makes the generation of SHAP values for the vision-based DRL could potentially meet the critical implementation time constraints.

3.4.2. *Adversarial attacks detector*

To efficiently identify adversarial attacks on the vision-based DRL landing scheme, we propose an LSTM-based detection mechanism. This detector is designed to evaluate the SHAP values emanating from the feature extractor's outputs, indicating whether an adversarial attack is occurred in the vision inputs. The LSTM, a variant of Recurrent Neural Networks (RNNs) renowned for its performance in sequential data analysis, such as speech recognition, is leveraged here due to its inherent capacity to mitigate the long-term dependency challenges that plague traditional RNNs [33]. Its architecture facilitates the retention and conveyance of information across sequences, ensuring that valuable insights from earlier time steps are not disregarded.

In the proposed method, SHAP values are computed for each respective output neuron within the DRL framework. This contrasts with adversarial attacks on a conventional classification CNN, which typically alter only the final output label. In the context of DRL, an adversarial incursion over certain time steps may influence all output neurons, potentially leading to erroneous decisions that could result in the lander crashing on terrains or mislanding. It is reasonable to postulate that a degree of interdependency exists among these output neurons. Consequently, the construction of an LSTM-based detector for adversarial attacks is posited to yield high levels of detection accuracy due to its ability to capture these dependencies.

Fig. 8 delineates the architecture of the proposed detector for adversarial attacks. This detector processes the SHAP values derived from the feature extractor of the vision-based DRL scheme, which yields a total of 16 output features. Given that the DRL system in question operates within a 3-D action space, the SHAP values assume a matrix form of $(3, 16)$. To input the SHAP values into the proposed adversarial detector, SHAP values are structured as sequential data with a length of 3. The output of the detector is a Boolean value, either $True$ or $False$, signifying the presence or absence of adversarial attacks.

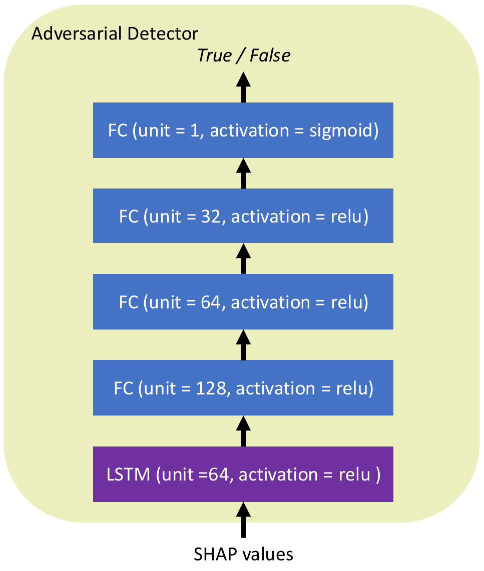
Adversarial Detector
True / False
FC (unit = 1, activation = sigmoid)
FC (unit = 32, activation = relu)
FC (unit = 64, activation = relu)
FC (unit = 128, activation = relu)
LSTM (unit =64, activation = relu)
SHAP values

Fig. 8. Proposed adversarial attack detector. The LSTM layer has 64 units and ReLu is applied as the activation function for the LSTM layer. FC layers in the format of $(units, activation)$. The output layer of the adversarial detector, which is also formed from the FC layer and outputs a Boolean to detect adversarial attacks.

Table 2
Planetary landing simulator configuration.

Parameter	Value
Initial position	[−1200, 100, 1200] ± [100, 10, 100] m
Initial velocity	[11, −1.5, −11] ± [1, 0.5, 1] m/s
Initial lander mass	300 kg
Max thrust force	[1500, 1500, 1500] N
Min thrust force	[−1500, −1500, 0] N
Frame rate	1 frame/s if altitude $>$ 30 m 2 frame/s if altitude $\leq$ 30 m

4. Experimental results

To assess the proposed methodology, three sets of experiments have been conducted. The first experiment involved training the vision-based DRL models using the proposed planetary landing simulator. The second experiment entailed generating SHAP values for the trained DRL model under both normal and adversarial examples. Subsequently, these SHAP values were utilised to train and validate the proposed adversarial attack detector. Finally, the vision-based DRL model and the adversarial attack detector were integrated to evaluate the detector's performance within the operational loop

4.1. *Performance of the vision-based DRL*

Table 2 illustrates the configuration settings of the planetary lander simulator, which is designed to emulate the soft landing process on Mars.

As discussed in previous section, three DRLs operate in a sequential manner. The first DRL agent ($Agent$ 1) is active when the altitude exceeds 400 m. The second DRL agent ($Agent$ 2) takes over at altitudes ranging from 400 m down to 30 m. Finally, the third DRL agent ($Agent$ 3) assumes control at altitudes below 30 m, as described in Fig. 6. In this scenario, the three DRL agents are trained in a sequential process. Initially, $Agent$ 1 is trained. Subsequently, $Agent$ 2 undergoes training using the states provided by $Agent$ 1. Finally, $Agent$ 3 is trained based on the initial states produced by $Agent$ 2.

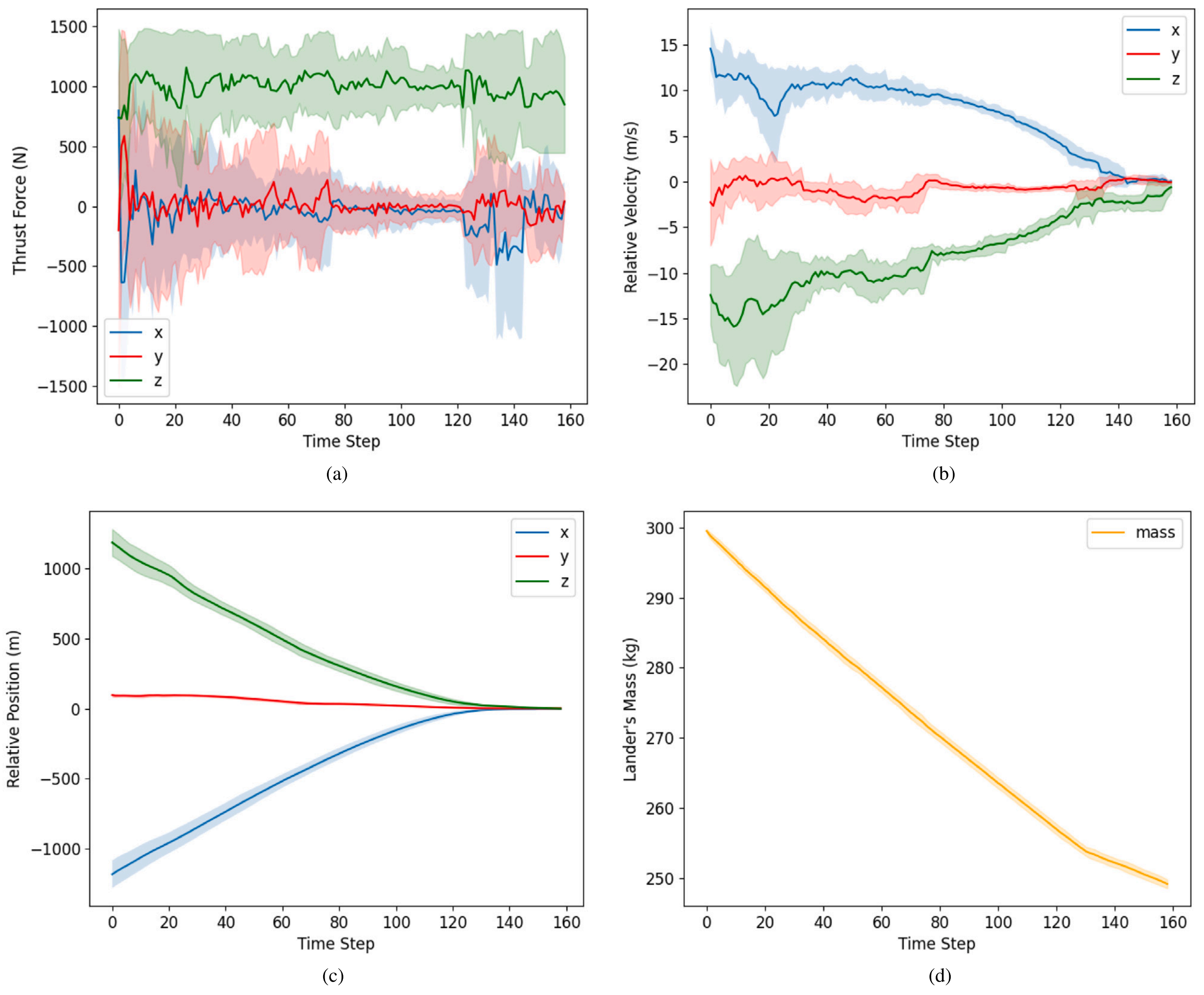

Fig. 9. Testing results for the proposed vision-based Soft Landing DRL over 100 episodes. (a) The thrust forces made by the agents against landing time step. (b) The relevant velocity against landing time step. (c) The relevant position against landing time step. (d) The mass of the lander against landing time step.

Table 3
DRL training hyperparameters.

Parameter	Value
Batch size	2048
Learning rate	1×10^{-4}
Optimizer	Adam
Action noise	0.1
Discount factor	0.99
Soft update coefficient	0.005
Replay buffer size	1,000,000

In this experiment, all three agents are trained with the Adam optimizer using a fixed learning rate of 1×10^{-4}. The batch size is set to 2048. Additionally, to encourage the agents to have more exploration in the environment, Gaussian noise is applied to the DRL's actions with a mean of 0 and a standard deviation of 0.1. Details of training hyperparameters are listed in Table 3. Then, the DRL models are tested with 300 random episodes initialised based on the configurations in Table 2 and achieves a of 100% in successful landing conditions that is defined in Table 1. Fig. 9 demonstrates a landing example controlled by the proposed vision-based DRL with 100 test episodes.

4.2. Adversarial attacks detector

As mentioned in Section 3, the SHAP values are computed at the output of the feature extractor in the proposed DRL scheme. The feature extractor contains 16 output neurons, therefore, 16 values are calculated for each output neuron of the DRLs, resulting 3×16 output SHAP values.

In this study, SHAP values are derived by the DeepSHAP algorithm [12]. This algorithm determines SHAP values for given inputs through an integration process over a set of background samples. It approximates SHAP values by aggregating the discrepancies between the expected output of the DL model on these background samples and the actual output of the current model. For this purpose, 1000 states are randomly chosen from the operational environment to calculate the downsampled features at the feature extraction stage. These images constitute the background dataset for the DeepSHAP explainer. In the development of the adversarial attack detector, we utilise 30,000 SHAP value sets for normal instances and another 30,000 for adversarial instances. Adversarial samples are generated by attacking the proposed DRLs at a random time step with random ϵ values: 20/255, 15/255, 10/255, 5/255, and 1/255. Following this, 6000 perturbed samples are randomly selected for each epsilon value to compute the respective SHAP values. Thus the dataset consists of a total of 60,000 examples.

To train the proposed adversarial attack detector, the dataset is divided into training and testing sets using an 0.8 train–test split,

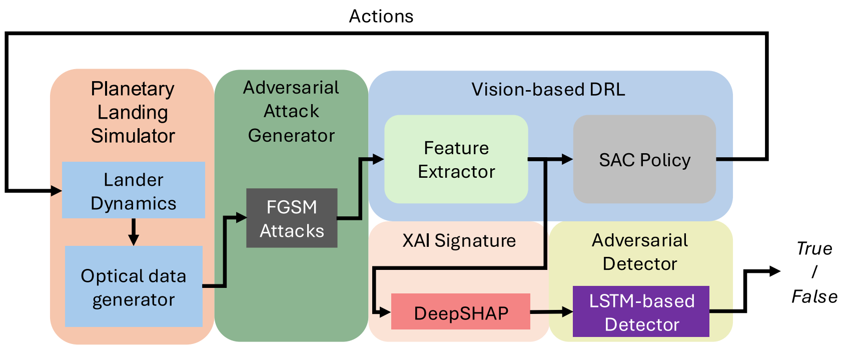

Fig. 10. Overview of integrated experiment. Images are generated by the Planetary Landing Simulator, then pass through the Adversarial Attack Generator and FGSM-based attacks are applied to perturb the images. Then, the vision-based DRL output the relative actions to the Planetary Landing Simulator to get next state of the lander. Subsequently, SHAP values are generated on the output of the feature extractor and pass through the Adversarial Detector to output $True/False$ statement indicating whether adversarial attacks are involved.

Table 4
LSTM training hyperparameters.

Parameter	Value
Batch size	256
Initial learning rate	1×10^{-3}
Optimizer	Adadelta
Learning rate decay	Exponential decay
Validation split	0.2
Early termination	200
Maximum epochs	10,000

Table 5
LSTM training and testing accuracy.

	Dataset size	Accuracy
Training	48,000	96.89%
Testing	12,000	97.16%

yielding 48,000 samples for training and 12,000 for testing. The adversarial attack detector is trained over 10,000 epochs using the Stochastic Gradient Descent (SGD) method coupled with the Adadelta optimizer. The training process is set with batch size of 256 and starting learning rate of 1e–3 with Exponential Learning Rate Decay applied. To enhance training efficiency, early termination is employed if the validation loss does not decrease over. Details of training hyperparameters are listed in Table 4.

Upon completion of training, evaluating the trained adversarial attack detector over the whole dataset, it achieved a training accuracy of 96.89% and an accuracy of 97.16% on the test set, which have been reported in Table 5. The experimental result indicates that the proposed adversarial attack detector reliably identifies adversarial attacks targeting the vision-based DRL during the planetary landing phase with considerable precision.

4.3. Integrated DRLs and adversarial attacks detector

The final experiment is designed to assess the performance of the proposed adversarial attack detector during the operation of the DRLs, as well as to evaluate the impact of adversarial attacks on the vision-based DRL landing scheme. Fig. 10 illustrates the on-the-loop test that integrates the planetary landing simulator, the vision-based DRLs, and the adversarial attack detector.

This experiment utilises the FGSM attacks with ϵ values chosen from 20/255, 15/255, 10/255, 5/255, and 1/255. Each ϵ value undergoes testing across 30 episodes. In this scenario, the FGSM attacks are initiated at a random time step during the episode and continue to perturb the image for the subsequent 30 time steps or until the lander makes contact with the ground. The detection accuracy is calculated as following:

Table 6
Experimental results of how the DRL will be effected by the adversarial attack and the accuracy of adversarial attack detector for various ϵ values.

ϵ	Successful landing	Detection accuracy
20/255	0%	99.64%
15/255	3.33%	99.76%
10/255	20%	99.39%
5/255	40%	98.94%
1/255	100%	84.0%
Average	48.67%	96.35%

$$Accuracy = \frac{Correct\ Detection}{No.\ of\ Inputs} \times 100\% \tag{26}$$

where the $Correct\ Detection$ is characterised by the condition where input frames subjected to an adversarial attack are identified as $True$ and those without an adversarial attack are identified as $False$.

Table 6 presents the experimental results of the detection accuracy of the adversarial attack detector while the DRL is in operation for different ϵ values in FGSM attack. The impact of adversarial attacks on the vision-based DRL has been assessed, as indicated by the 'Successful Landing' metric in Table 6. A 'Successful Landing' occurs when the lander touches down and the normalised position error is within 5 metres of the target landing site, and the normalised velocity is below 2 m/s, as detailed in Table 1.

From the test results, the proposed adversarial attack detector successfully identifies incoming FGSM attacks with an average accuracy of 96.35% using test ϵ values. As the ϵ value decreases, indicating fewer perturbations to the input images, there is a slight decline in detection accuracy. However, this corresponds with an increase in the successful landing rate. The detection accuracy for adversarial attacks experiences a more pronounced decrease when $\epsilon = 1/255$, attributable to the minimal perturbation applied to the input image. Despite this, the lander achieves the successful landing criteria in all test episodes under these conditions. This outcome implies that the feature extractor within the proposed DRL is capable of producing highly accurate features, even with minor adversarial perturbations. Consequently, the lander is still able to arrive at the target location with the desired velocity. For higher perturbations, i.e. $\epsilon \geq 5/255$, the adversarial attack detector demonstrates a high level of confidence in identifying incoming FGSM attacks. However, strong perturbations to the input images can lead to poor performance in the current vision-based DRL guidance scheme.

5. Conclusion

This study initiates an investigation into the effects of adversarial attacks on a vision-based DRL framework for guidance and control in planetary landing. A planetary landing simulator is developed to generate optical data along with corresponding aerodynamic parameters for the target landing scenario. The paper introduces a DRL scheme that employs the SAC policy, relying solely on visual data for observation. The research then delves into the vulnerability of the vision-based DRL to FGSM attacks. Following this, an adversarial attack detector is introduced, utilising SHAP value-based explanations to pinpoint adversarial manipulations in input images. A series of experiments are conducted to assess the efficacy of the vision-based DRL in landing guidance and control, the influence of adversarial attacks on DRL performance during the landing phase, and the effectiveness of the newly proposed adversarial attack detector. The experimental results show that the proposed adversarial attack detector performs robustly in detecting adversarial attacks, achieving an average of 96.35% detection rate while DRL is operating on the simulation environment.

Based on the findings from this paper, there are some research avenues that can be further explored to improve the robustness of employing the DRL-based approach in future space exploration missions. On the one hand, the experimental results indicate that the vision-based DRL guidance scheme lacks the robustness to ensure a satisfactory success rate for landing under adversarial attacks. Training DRLs against adversarial attacks on the dynamics of the environment could be a future strategy to improve the robustness of DRL-based techniques in future space exploration missions.

On the other hand, while the proposed LSTM-based detector demonstrates high accuracy in identifying adversarial attacks, our current work has not thoroughly examined the actual recourse following the detection of these attacks. The integration of the DRL-based guidance scheme with the adversarial attack detector to develop adversarial defence mechanisms presents a promising avenue for future work. Specifically, upon detection of an adversarial attack, exploring methods to rectify the actions of DRL systems could improve the overall performance of the planetary landing guidance scheme.

CRediT authorship contribution statement

Ziwei Wang: Writing – review & editing, Writing – original draft, Validation, Software, Resources, Methodology, Investigation, Formal analysis, Data curation. **Nabil Aouf:** Writing – review & editing, Supervision, Project administration, Methodology, Investigation, Funding acquisition, Formal analysis, Conceptualization.

Declaration of competing interest

The authors declare the following financial interests/personal relationships which may be considered as potential competing interests: Nabil Aouf reports financial support was provided by European Space Agency. If there are other authors, they declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgements

This work is funded by the European Space Agency (ESA), Contract Number: ESA AO/2-1856/22/NL/GLC/ov. We would like to thank Jose Pizarro and Christophe Honvault from ESA for their support and advice.

References

[1] E. Taheri, N.I. Li, I. Kolmanovsky, Co-state initialization for the minimum-time low-thrust trajectory optimization, Adv. Space Res. 59 (9) (2017) 2360–2373.

[2] H. Yang, H. Baoyin, Fuel-optimal control for soft landing on an irregular asteroid, IEEE Trans. Aerosp. Electron. Syst. 51 (3) (2015) 1688–1697.

[3] A.E. Johnson, J.F. Montgomery, Overview of terrain relative navigation approaches for precise lunar landing, in: 2008 IEEE Aerospace Conference, 2008, pp. 1–10, http://dx.doi.org/10.1109/AERO.2008.4526302.

[4] L. Cheng, Z. Wang, F. Jiang, Real-time control for fuel-optimal moon landing based on an interactive deep reinforcement learning algorithm, Astrodynamics 3 (2019) 375–386.

[5] B. Gaudet, R. Linares, R. Furfaro, Deep reinforcement learning for six degree-of-freedom planetary landing, Adv. Space Res. 65 (7) (2020) 1723–1741.

[6] G. Ciabatti, S. Daftry, R. Capobianco, Autonomous planetary landing via deep reinforcement learning and transfer learning, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2021, pp. 2031–2038.

[7] Y. Li, J. Moreau, J. Ibanez-Guzman, Emergent visual sensors for autonomous vehicles, IEEE Trans. Intell. Transp. Syst. 24 (5) (2023) 4716–4737.

[8] D. Rondao, N. Aouf, M.A. Richardson, Chinet: Deep recurrent convolutional learning for multimodal spacecraft pose estimation, IEEE Trans. Aerosp. Electron. Syst. (2022).

[9] A. Scorsoglio, R. Furfaro, R. Linares, B. Gaudet, Image-based deep reinforcement learning for autonomous lunar landing, in: AIAA Scitech 2020 Forum, 2020, p. 1910.

[10] Y. Lin, H. Zhao, X. Ma, Y. Tu, M. Wang, Adversarial attacks in modulation recognition with convolutional neural networks, IEEE Trans. Reliab. 70 (1) (2020) 389–401.

[11] J. Grabinski, J. Keuper, M. Keuper, Aliasing and adversarial robust generalization of CNNs, Mach. Learn. 111 (11) (2022) 3925–3951.

[12] S.M. Lundberg, S.-I. Lee, A unified approach to interpreting model predictions, Adv. Neural Inf. Process. Syst. 30 (2017).

[13] P.E. Pope, S. Kolouri, M. Rostami, C.E. Martin, H. Hoffmann, Explainability methods for graph convolutional neural networks, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2019, pp. 10772–10781.

[14] M. Nazari, A. Kluge, I. Apostolova, S. Klutmann, S. Kimiaei, M. Schroeder, R. Buchert, Explainable AI to improve acceptance of convolutional neural networks for automatic classification of dopamine transporter SPECT in the diagnosis of clinically uncertain parkinsonian syndromes, Eur. J. Nucl. Med. Mol. Imaging (2022) 1–11.

[15] I.J. Goodfellow, J. Shlens, C. Szegedy, Explaining and harnessing adversarial examples, 2014, arXiv preprint arXiv:1412.6572.

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

[17] X. Xu, Y. Chen, C. Bai, Deep reinforcement learning-based accurate control of planetary soft landing, Sensors 21 (23) (2021) 8161.

[18] T.P. Lillicrap, J.J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, D. Wierstra, Continuous control with deep reinforcement learning, 2015, arXiv preprint arXiv:1509.02971.

[19] S. Fujimoto, H. Hoof, D. Meger, Addressing function approximation error in actor-critic methods, in: International Conference on Machine Learning, PMLR, 2018, pp. 1587–1596.

[20] T. Haarnoja, A. Zhou, P. Abbeel, S. Levine, Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor, in: International Conference on Machine Learning, PMLR, 2018, pp. 1861–1870.

[21] K. Simonyan, A. Vedaldi, A. Zisserman, Deep inside convolutional networks: Visualising image classification models and saliency maps, 2013, arXiv preprint arXiv:1312.6034.

[22] B. Zhou, A. Khosla, A. Lapedriza, A. Oliva, A. Torralba, Learning deep features for discriminative localization, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2016, pp. 2921–2929.

[23] R.R. Selvaraju, M. Cogswell, A. Das, R. Vedantam, D. Parikh, D. Batra, Grad-cam: Visual explanations from deep networks via gradient-based localization, in: Proceedings of the IEEE International Conference on Computer Vision, 2017, pp. 618–626.

[24] I. Ilahi, M. Usama, J. Qadir, M.U. Janjua, A. Al-Fuqaha, D.T. Hoang, D. Niyato, Challenges and countermeasures for adversarial attacks on deep reinforcement learning, IEEE Trans. Artif. Intell. 3 (2) (2021) 90–109.

[25] Y.-C. Lin, M.-Y. Liu, M. Sun, J.-B. Huang, Detecting adversarial attacks on neural network policies with visual foresight, 2017, arXiv preprint arXiv:1710.00814.

[26] A. Havens, Z. Jiang, S. Sarkar, Online robust policy learning in the presence of unknown adversaries, Adv. Neural Inf. Process. Syst. 31 (2018).

[27] Y. Xiang, W. Niu, J. Liu, T. Chen, Z. Han, A PCA-based model to predict adversarial examples on Q-learning of path finding, in: 2018 IEEE Third International Conference on Data Science in Cyberspace, DSC, IEEE, 2018, pp. 773–780.

[28] T. Hickling, N. Aouf, P. Spencer, Robust adversarial attacks detection based on explainable deep reinforcement learning for uav guidance and planning, IEEE Trans. Intell. Veh. (2023).

[29] Home of the blender project - free and open 3d creation software, 2024, https://www.blender.org/. (Accessed May 2024).

[30] S.S. Sutton, M. Chojnacki, A.S. McEwen, R.L. Kirk, C.M. Dundas, E.I. Schaefer, S.J. Conway, S. Diniega, G. Portyankina, M.E. Landis, et al., Revealing active mars with HiRISE digital terrain models, Remote Sens. 14 (10) (2022) 2403.

[31] E. Millour, F. Forget, A. Spiga, T. Navarro, J.-B. Madeleine, L. Montabone, A. Pottier, F. Lefevre, F. Montmessin, J.-Y. Chaufray, et al., The mars climate database (MCD version 5.2), in: European Planetary Science Congress, Vol. 10, 2015, pp. 2015–2438.

[32] A. Shrikumar, P. Greenside, A. Kundaje, Learning important features through propagating activation differences, in: International Conference on Machine Learning, PMLR, 2017, pp. 3145–3153.

[33] Y. Yu, X. Si, C. Hu, J. Zhang, A review of recurrent neural networks: LSTM cells and network architectures, Neural Comput. 31 (7) (2019) 1235–1270.