

Explainable Deep Reinforcement Learning for UAV Autonomous Navigation

Lei He, Student Member, IEEE, Aouf Nabil, Member, IEEE, Bifeng Song

Abstract—Modern deep reinforcement learning plays an important role to solve a wide range of complex decision-making tasks. However, due to the use of deep neural networks, the trained models are lacking transparency which causes distrust from their user and hard to be used in the critical field such as self-driving car and unmanned aerial vehicles. In this paper, an explainable deep reinforcement learning method is proposed to deal with the multirotor obstacle avoidance and navigation problem. Both visual and textual explanation is provided to make the trained agent more transparency and comprehensible for humans. Our model can provide real-time decision explanation for non-expert users. Also, some global explanation results are provided for experts to diagnose the learned policy. Our method is validated in the simulation environment. The simulation result shows our proposed method can get useful explanations to increase the user’s trust to the network and also improve the network performance.

Index Terms—Explainable, Deep reinforcement learning, UAV obstacle avoidance.

I. INTRODUCTION

UNMANNED Aerial Vehicles (UAVs) have been widely used in many application, such as good delivery, emergency surveying and mapping. Autonomous navigation in the large unknown complex environment is an essential capability for these UAVs to operate more intelligent and safety.

In general, there are two main solutions for UAV obstacle avoidance. The first solution relies on the state estimator using VIO or SLAM, then generate safety trajectories using optimization method [1], [2]. It’s a cascade process include mapping, localization planning and control. This kind of method can generate nearly optimal trajectories for some optimization objectives such as safety and smoothness, they require lots of computation and memory to store the map and run the optimization algorithms every step. In addition, these techniques also suffer from high drift and noise, impacting the quality of both localization and the map used for planning.

Another solution is using a reactive control method, which can generate control command from the perception information directly [3], [4]. This method is efficient, however, it is always non-optimal because of lacking global information.

UAV navigation is a sequential decision-making problem. Some researchers modelled this problem as a Markov decision process (MDP) and solved using reinforcement learning (RL) methods. For example, Ross et al [5] build and Imitation learning (IL)-based controller using a small set of human demonstrations and achieved a good performance in natural forest environments. Imanberdiyev et al [6] developed a high-level control method for autonomous navigation of UAVs using a novel model-based reinforcement learning method, TEXPLORE. He et al [7] combine bio-inspired monocular vision perception method with a deep reinforcement learning (DRL) reactive local planner to address the UAV navigation problem. They also proposed learning from demonstration method to speed up the training process [8]. Wang et al [9] formulated the navigation problem as a partially observable Markov decision process (POMDP) and solved by a novel online DRL algorithm. He also invested the sparse reward situation using a learn with help (LwH) method [10]. Comparing to the optimization-based method, the RL method can get the end-to-end policy which can process raw sensor data directly such as images. There is no need to do the optimization every time, which is computation efficiency. Also, once the training converges, the optimal policy will be obtained at every state.

Although DRL method can get excellent performance, an enormous problem is that deep learning methods turn out to be uninterpretable black boxes, which create serious challenges to the Artificial Intelligence (AI) system based on neural network [11]. This problem falls with the so-called explainable AI (XAI) filed. Arrieta et al gives a review of XAI [12].

Comparing to the burst of XAI research in supervised learning, explainability for RL is hardly explored [13]. Juozapaitis et al [14] explain the RL agent using reward decomposition. This approach decomposes reward into sums of semantically meaningful reward types so that actions can be compared in terms of trade-offs among the types. Reward deposition is also used in strategic tasks such as StarCraft II [15]. Jung Hoon Lee [16] proposed a method to derive a secondary comprehensible agent from NN-based RL agent, the decision makings are based on simple rules. Beyret et al [17] proposed an explainable RL for robotic manipulation. They presented a hierarchical DRL system include both low-level agent handling actions and high-level agent learning the dynamics and the environment. The high-level agent is used to interpret for the human operator. Madumal et al [18] use causal models to derive causal explanations of the behaviour of model-free reinforcement learning agent. A structural causal model is learned during the reinforcement learning phase. The explanations of behaviour are generated based on the counterfactual analysis of the causal model. They also introduced a distal explanation model that can analyse counterfactual and opportunity chains using decision trees and causal models [19].

Explainability is critical and essential for DRL-based UAV navigation system. On the one hand, it’s useful for non-expert
user to know the reason why the controller turn right rather than turn left when it facing an obstacle. On the other hand, it also benefits the network and controller designer to know the decision making progress and do some adjustment to improve network performance.

This work proposed an explainable deep reinforcement learning method for UAV navigation and obstacle avoidance in the complex unknown environment. First, a navigation policy is trained using DRL method in a high-fidelity simulation environment. Then, the trained network is explained using a post-hoc explanation method based on feature attribution. Comparing to the transparent model methods, post-hoc methods can provide explanations of an RL policy after its training, which keeps the model performance. Both real-time visual and textual explanation is provided for non-expert users to make them understand the trained model. Moreover, some trajectory explanations can also be used for experts to analyze and improve the network.

Our main contributions can be summarised as follows:
- An autonomous navigation policy for UAV learned using DRL method.
- A novel CNN attention visualization method based on fair feature attribution.
- Real-time textual model decision explanation for non-expert users.

II. PRELIMINARIES

A. MDP and DRL

In this work, the navigation and obstacle avoidance problem is formulated with MDP. An MDP is defined by a tuple \(<S,A,R,P,\gamma>\), where \(S\) is the set of the states, \(A\) is consists of a set of actions \(s\), a set of actions \(a\), a reward function \(R(s,a)\), a transition function \(P(s'|s,a)\), and a discount factor \(\gamma \in (0,1)\). In each state \(s\in S\), the agent takes an action \(a\in A\). After executing the action \(a\) in the environment, the agent receives a reward \(R(s,a)\) and reaches a new state \(s'\), determined from the probability distribution \(P(s'|s,a)\). The goal of DRL is to find a policy \(\pi\) mapping states to actions that maximizes the expected discounted total reward over the agent’s lifetime. This concept is formalized by the action value function: \(Q^\pi(s,a) = E^\pi \sum_{t=0}^{T} \gamma^t R(s_t, a_t)\), where \(E^\pi\) is the expectation over the distribution of the admissible trajectories \((s_0, a_0, s_1, a_1, \ldots)\) obtained the policy \(\pi\) starting from \(s_0 = s\) and \(a_0 = a\).

B. Reinforcement Learning for UAV Navigation

Here, we treat the UAV navigation problem as a sequential decision process and formulated it as an MDP. Suppose the UAV takes off from a departure position in a 3-D environment, which is denoted as \((x_0, y_0, z_0)\) in the Earth-fixed coordinate frame, and targets at flying to a destination that is denoted as \((x_d, y_d, z_d)\). The observation or the state at time \(t\) consists of both raw depth image and UAV state features: \(o_t = [o_{\text{deep}}, o_{\text{state}}]\). The state feature consists of relative position to goal and current velocity: \(o_{\text{state}} = [d_{xy}^t, d_z^t, \xi^t, v_{xy}^t, v_z^t, \phi^t]\), where \(d_{xy}^t\) and \(d_z^t\) denote the distance between the UAV's current position and the destination position in x-y plane and z axis, \(\xi^t\) is the relative angle between UAV current first-perspective direction to the destination position, \(v_{xy}^t\) and \(v_z^t\) are the UAV current speed and \(\phi^t\) is the steering angular speed. Action \(a = [v_{cmd}, v_z^{cmd}, \phi_{cmd}]\) generated from the policy network \(\pi(s)\) consists of 2 linear velocity and 1 angular velocity. These actions are passed to the low-level controller as velocity setpoint command to achieve the navigation. The network architecture is shown in Fig. 1.

C. Feature Attribution

Formally, suppose we have a function \(F : R^n \rightarrow [0,1]\) that represents a deep neural network and an input \(x = (x_1, \ldots, x_n) \in R^n\). An attribution of the prediction at input \(x\) relative to a baseline input \(x'\) is a vector \(A_F(x, x') = (a_1, \ldots, a_n) \in R^n\) where \(a_i\) is the contribution of \(x_i\) to the prediction \(F(x)\). There are two different types of feature attribution algorithms: Shapley-value-based algorithm and gradient-based algorithm. There is a fundamental difference between these two algorithm types.

Shapley value is a classic method to distribute the total gains of a collaborative game to a coalition of cooperating
players. It is a fair way to attribute the total gain to the players based on their contributions. For ML models, we formulate a game for the prediction at each instance. We consider the total gains to be the prediction value for that instance, and the players to be the model features of that instance. The collaborative game is all of the model features cooperating to form a prediction value. A Shapley-value-based explanation method tries to approximate Shapley values of a given prediction by examining the effect of removing a feature under all possible combinations of presence or absence of the other features. Shapley values are the only additive feature attribution method that satisfies the desirable properties of local accuracy, missingness, and consistency. However, exact Shapley value computation is exponential in the number of features.

A gradient-based explanation method tries to explain a given prediction by using the gradient of the output with respect to the input features. However, the problem with gradients is that they break sensitivity, a property that all attribution methods should satisfy. For example, consider a one variable, one ReLU network, \( f(x) = 1 - \text{ReLU}(1 - x) \). Suppose the baseline is \( x = 0 \) and the input is \( x = 2 \). The output changes from 0 to 1, but the gradient is zero at \( x = 1 \) because \( f \) becomes flat after \( x = 1 \), so the gradient method gives attribution of 0 to \( x \). This phenomenon has been reported in [20]. To address this problem, Sundararajan et al. [21] proposed Integrated Gradients (IG) algorithm. However, this algorithm requires computing the gradients of the model output on a few different inputs (typically 50) between current feature value and baseline value.

D. SHAP and DeepSHAP

SHAP (SHapley Additive exPlanations), proposed by Lundberg and Lee [22], can assigns each feature an importance value for a particular prediction. For a simple linear regression problem, the predictions can be written as:

\[
y_{i} = b_{0} + b_{1}x_{1i} + \cdots + b_{d}x_{di}
\]  

(1)

where \( y_{i} \) is the \( i \)-th predicted response, \( x_{1i}, \ldots, x_{di} \) are the features of current observation, and \( b_{0}, \ldots, b_{d} \) are the estimated regression coefficients. If the features are independent, the contribution of the \( k \)-th feature to the predicted response \( \hat{y}_{i} \) can be unambiguously expressed as \( b_{k}x_{ki} \) for \( k = 1, \ldots, d \).

SHAP is a generalization of this concept to more complex neural network models. We define the following:

- \( F \) is the entire set of features, and \( S \) denotes a subset.
- \( S \cup i \) is the union of the subset \( S \) and feature \( i \).
- \( E[f(X) | X_{s} = x_{s}] \) is the conditional expectation of model \( f() \) when a subset \( S \) of features are fixed at the local point \( x_{s} \).

Then, the SHAP value is defined to measure the contribution of the \( i \)-th feature as

\[
\Phi_{i} = \sum_{S \subseteq F \setminus \{i\}} \frac{|S|!(|F| - |S| - 1)!}{|F|!} [f_{S \cup \{i\}}(x_{S \cup \{i\}}) - f_{S}(x_{S})]
\]  

(2)

SHAP values are proved to satisfy good properties such as fairness and consistency on attributing importance scores to each feature. But the calculation of SHAP values is computationally expensive. In our case, we use Deep SHAP, which is a model-specific method to improve computational performance through a connection between Shapley values and DeepLIFT [23].

DeepSHAP [24] is a framework for layer-wise propagation of Shapley values that builds upon DeepLIFT [23]. If we define including an input as setting it to its actual value instead of its reference value, DeepLIFT can be thought of as a fast approximation method of the Shapley values. If our model is fully linear, we can get exact SHAP values by summing the attributions along all possible paths between input \( x_{i} \) and the model’s output \( y \). However, in our network, for example fully connected network, there are non-linear activation function applied after the linear part, such as ReLU, tanh or sigmoid operations. To deal with the non-linear part, DeepLIFT provided the Rescale rule and the RevealCancel rule. Passing back nonlinear attributions linearly is an approximation, but there are two main benefits: 1) fast computation using only one backward pass and 2) a guarantee of local accuracy.

III. PROPOSED METHOD

In this section, we introduce our model explanation method. The trained policy network consists of CNN perception part and FC control part. A novel visual explanation method is proposed to localize the CNN attention position. In addition, a textual explanation method based on the feature attribution is also provided for real-time action explanation.

A. Visual explanation combines both CAM and SHAP values

Understanding the insights of CNN has always been a pain point, though CNN can get excellent predictive performance. In our problem, CNN is used to extract the visual feature from the raw depth image. CNN visualization can provide a better explanation for the RL policy.

In [25], a deconvolutional network (Deconvnet) approach was proposed to visualize activated pattern in each hidden
unit. This method can visualize features individually but is limited as it is hard to summarize all hidden patterns into one pattern. Simonyan et al. [26] visualize partial derivatives of predicted class scores w.r.t. pixel intensities, while Guided Backpropagation [27] makes modifications to raw gradients that result in qualitative improvements. This method can provide fine-grained visualizations.

In [28], the authors proposed Class Activation Map (CAM) using global average pooling (GAP) layer to summarize the activation of the last CNN layer. However, it is only applicable to a particular CNN architecture where global average pooled convolutional feature maps are fed directly into softmax. Grad-CAM provides a new way of combining feature maps using the gradient signal that does not require any modification in the network architecture [29]. It can be used to off-the-shelf CNN architecture. Grad-CAM uses the gradient information flowing into the last convolutional layer of CNN to assign importance values to each neuron for a particular decision of interest. Both CAM and Grad-CAM is mainly used for the classification problem.

To visualize the CNN perception part of our network, a method combined both CAM and SHAP values is proposed. Because our problem is a regression problem, we call this method SHAP-RAM (SHAP value-based regression activation map). Similar to CAM method, global average pooling (GAP) layer is used to summarize the visual feature in our CNN perception network. The output of the GAP layer is defined as the CNN feature. Different from CAM and Grad-CAM, in our method, the SHAP value of CNN feature is used to determine the importance of the CNN feature which generated from the corresponding activation map. A coarse localization map highlighting the important regions in the image is generated by a weighted sum of the last CNN activation map, where SHAP value is the weight.

Comparing to CAM, our method can be used in any network architecture with GAP layers. Comparing to Grad-CAM, SHAP value is used as weights of the forward activation maps rather than gradients, which can provide a fairer attribution of the activation maps.

B. Real-time textual explanation for DRL based UAV navigation

Our model has 3 continuous action outputs, horizontal velocity $v_x$, vertical velocity $v_z$ and steering angular velocity $\phi$. To get the textual explanation, each action is divided into 3 parts based on the reference action, as shown in Fig. 3. If the action is similar to the reference action, we think that this action is to maintain current action. If the output action either bigger or smaller than the reference action, a specific text is used to describe the action, such as 'slow down' or 'speed up'. The final textual output of the action is the combination of these three textual descriptions, for example, the action can be described as 'slow down, maintain the altitude and turn right'.

Finally, both visual and textual explanation is used to explain the network policy output. Because of the fast computing speed, a real-time explanation can be achieved for every action.

Fig. 3. Action description. Each action is divided into 3 parts. While the prediction fall into the central part, we say it is maintain the current state. Otherwise, there will be a textual description of each action. The final description will be the combination of these three individual descriptions.

IV. MODEL TRAINING

A. Training Environment and Setting

The navigation network is trained from scratch in AirSim [30] simulator built on Unreal Engine, which provides high fidelity depth image and a low-level controller to stabilize the UAV. A customized environment is created using the Unreal Engine which is shown in Fig. 4. The size of the environment is square with 200 meters on each side. Some stones were randomly placed as obstacles. At the beginning of each episode, the quadrotor takes off from the centre of the environment. The goal is set randomly on the circle with a radius of 70 meters and centred on the take-off point. The episode terminated when the quadrotor reaches the goal position with an accept radius of 2 meters or crashed on the obstacles. The controller is running at 10Hz to generate velocity command to the low-level controller provided by AirSim.

An off-policy model-free reinforcement learning algorithm, Twin Delayed DDPG (TD3) [31], is used for model training. As the successor of the DDPG method, TD3 addresses the overestimate problem issue of Q-value in DDPG by introducing three critical tricks: clipped double Q-Learning, delayed policy update and target policy smoothing [32]. This DRL algorithm is widely used for continuous control problem. The hyperparameters of TD3 are summarized in Table I in Appendix.

B. Reward Function Design

The reward function is critical for DRL problem. In general, the reward function for navigation is simple, we can only reward for reach the goal as soon as possible and punish collision. However, because the state space is very huge in the
navigation task, it’s better to introduce some continuous reward signal to guide the exploration and speed up the training process. After a lot of testing, a hand-designed reward function is utilized which consists of a continuous goal approaching reward and some penalty terms:

\[
r(s_t) = \begin{cases} 10, & \text{if success} \\ R_{\text{goal}} - P_{\text{state}}, & \text{otherwise} \end{cases} \tag{3}
\]

where \( R_{\text{goal}} = d(s_{t-1}) - d(s_t) \) is the goal approaching reward and \( d(s_t) \) is the Euclidean distance from current position to goal position at time \( t \). \( P_{\text{state}} \) is the penalty term at current step:

\[
P_{\text{state}} = \omega_1 \cdot C_{\text{obs}} - \omega_2 \cdot C_{\text{act}} - \omega_3 \cdot C_{\text{pos}} \tag{4}
\]

where \( C_{\text{obs}}, C_{\text{act}} \) and \( C_{\text{pos}} \) are penalty terms for obstacle, action, and position error.

\[
C_{\text{obs}} = \frac{d_{\text{safe}} - d_{\text{obs}}(s_t)}{d_{\text{safe}} - d_{\text{safe min}}} \tag{5}
\]

is the penalty term to prevent the quadrotor from getting close to the obstacle. In equation 5, \( d_{\text{safe}} \) and \( d_{\text{safe min}} \) is the safety distance and minimum distance allowed to the obstacles. \( d_{\text{obs}}(s_t) \) is the minimum distance to the obstacle at time \( t \). In our training process, \( d_{\text{safe}} = 5 \) and \( d_{\text{safe min}} = 1 \), which means we give punishment if the quadrotor gets close to the obstacle in 5 meters. When the minimum distance to the obstacle is less than 1 meter, it is considered crashed and this episode terminates. To stabilize the training process, the continuous reward part is constrained to -1 to 1.

C. Training Result

The policy network is trained for 100k time steps (around 1000 episodes). To speed up the training process, the Airsim simulation clock speed is set to 10. The total training process took about 7 hours on an Intel i7-8700 processor and an Nvidia GeForce GTX1060 GPU. The episode reward and success rate are plotted in Fig. 5. From the training result, the policy gets over 80% success rate which means the network can guide the UAV to the goal position without collision with any obstacles.

V. MODEL EXPLANATION

After training, we can get a policy with good performance. In order to keep the performance, we do the post-hoc real-time explanation based on the trained policy. DeepSHAP method is used to get feature importance and our explanation will be generated based on these SHAP values.

A. Defining the Reference

Feature attribution method generates the contribution of each feature based on a reference input or baseline input. The choice of the reference input is critical for obtaining insightful results [23]. In practice, choosing a good reference would rely on domain-specific knowledge. For instance, in object recognition networks, it is the black image.

In our case, we choose the depth image without any obstacles as the reference input image. For state feature input, we set \( a_{\text{ref}} = [d_{xy} = 70, d_z = 0, \xi = 0, v_{xy} = 0, v_z = 0, \phi = 0] \) which means the UAV just take off from the start point and has no velocity. The reference image is shown in Fig. 6. Based on this reference input, we can get reference action from the policy network: \( v_{xy}^{\text{cmd}} = 3.71 m/s, v_z^{\text{cmd}} = -0.03 m/s, \phi^{\text{cmd}} = 4.15^\circ/s \).

B. Trajectory Analysis

We choose one of the trajectories from the evaluation process to get some inside information of the policy. Fig. 7 shows the depth image at different time steps. Fig. 8 and Fig. 9 shows the control command and state features over the trajectory. From \( d_{xy} \) in Fig. 9, we can see that the UAV always fly towards the goal position and the distance to goal is reducing over the trajectory. Finally, at \( t = 160 \), UAV reached the goal position.

C. Action explanation

Action explanation can be generated for every time step. Here, 3 specific time steps are choosing to demonstrate our visual and textual explanation for actions. As shown in Fig. 10 at \( t = 0 \), the action is slow down, keep altitude and turn right. The explanation shows both slow down and turn right are caused by the angular error to goal. This makes sense because the direction at \( t = 0 \) doesn’t match the goal position, so the UAV need turn right. At \( t = 53 \), the action is slow down, climb and turn right. The explanation shows this is caused by the CNN feature. From the heatmap generated using SHAP-RAI, we can see the CNN detected left edge of the stone.
which is the obstacle. At $t = 89$, the action is slow down, climb and turn left. This is also caused by the CNN feature.

To find out the meaning of the CNN features, we also plotted the last CNN layer activation map at both $t = 53$ and $t = 89$ as shown in Fig. [11]. From this activation map, we can see $t = 53$, CNN feature 8 is the left and right edges of the obstacle which contributes most to the slow down action. CNN feature 7 is the obstacle and some ground which contributes to the climb. CNN feature 4 shows the right side edge of the obstacle with some free space background, which leads to the turn right action.

### D. Model analysis

After the action explanation, we can summarize all the feature attribution over the 20 trajectories, 2858 time steps in total. Fig. [12] shows the SHAP summary plot that orders the features based on their importance to the different action.

We can see that the CNN feature contributes most to action $a_1 : v_{xy}^{cmd}$ and $a_2 : v_{z}^{cmd}$. Except the CNN features, the current horizontal velocity $v_{xy}$ and distance to goal $d_{xy}$ are the most importance features contribute to $a_1 : v_{xy}^{cmd}$, $d_{xy}$, $v_{xy}$ and $v_{z}$ contributes more to $a_2 : v_{z}^{cmd}$ the vertical velocity command. The angle error $\xi$ is the most important feature to $a_3 : \phi_{cmd}$.

With the feature value and its SHAP value, we can invest the relationship between the feature intensity and its importance measurement as shown in Fig. [13]. From the plot, we can find that there is some relationship between the feature value and the SHAP value. For example, the angle error shows a positive correlation to its SHAP value. However, the angular speed shows a negative correlation.

### VI. Conclusion

In this paper, the UAV autonomous navigation problem is solved with the DRL technique. Different from other works, this paper mainly focused on the model explainability rather than treat the trained model as a black box. Based on the feature attribute, both visual and textual explanation are generated to open the black box. To get a better visual explanation of the CNN perception part, a new saliency map generation method proposed combining both CAM and SHAP values. Our method can provide real-time action textual explanation for non-expert users which is important for the application of DRL based model in the real world.

Because this paper mainly focused on the explanation part, the trained model is not perfect. There still some explanations don’t make sense. In the future, the model will be fine-trained and improved based on the explanation. Finally, the trained model and explanation method will be verified on a UAV platform in the real complex outdoor environment.

### Appendix

#### A. Hyperparameters of TD3

The hyperparameters are shown in Table

### REFERENCES

[1] S. Liu, M. Watterson, K. Moha, K. Sun, S. Bhattacharya, C. J. Taylor, and V. Kumar, “Planning dynamically feasible trajectories for quadrotors using safe flight corridors in 3-d complex environments,” IEEE Robotics and Automation Letters, vol. 2, no. 3, pp. 1688–1695, 2017.
TABLE I

| Hyperparameter       | Value  |
|----------------------|--------|
| mini-batch size      | 128    |
| replay buffer size   | 50000  |
| discount factor      | 0.99   |
| learning rate        | 0.0003 |
| random exploration steps | 2000  |
| square deviation of exploration noise | 0.3   |

Fig. 10. Action explanation at 3 different time steps.

Fig. 11. Last CNN layer activation map.

[2] B. Zhou, F. Gao, L. Wang, C. Liu, and S. Shen, “Robust and efficient quadrotor trajectory generation for fast autonomous flight,” *IEEE Robotics and Automation Letters*, vol. 4, no. 4, pp. 3529–3536, 2019.

[3] S. Paschal and J. Rose, “Fast, lightweight autonomy through an unknown cluttered environment: Distribution statement: Aapproved for public release; distribution unlimited,” in *2017 IEEE Aerospace Conference*. IEEE, 2017, pp. 1–8.

[4] H. D. Escohbar-Alvarez, N. Johnson, T. Hebble, K. Klingebiel, S. A. Quintero, J. Regenstein, and N. A. Browning, “R-advance: Rapid adaptive prediction for vision-based autonomous navigation, control, and evasion,” *Journal of Field Robotics*, vol. 35, no. 1, pp. 91–100, 2018.

[5] S. Ross, N. Melik-Barkhudarov, K. S. Shankar, A. Wendel, D. Dey, J. A. Bagnell, and M. Hebert, “Learning monocular reactive uav control in cluttered natural environments,” in *2013 IEEE international conference on robotics and automation*. IEEE, 2013, pp. 1765–1772.

[6] N. Imanberdiyev, C. Fu, E. Kayacan, and I.-M. Chen, “Autonomous navigation of uav by using real-time model-based reinforcement learning,” in *2016 14th International Conference on Control, Automation, Robotics and Vision (ICARCV)*. IEEE, 2016, pp. 1–6.

[7] L. He, N. Aouf, J. F. Whidborne, and B. Song, “Integrated moment-based lqmg and deep reinforcement learning for uav obstacle avoidance,” pp. 7491–7497, 2020.

[8] ———, “Deep reinforcement learning based local planner for uav obstacle avoidance using demonstration data,” *arXiv preprint arXiv:2008.02521*, 2020.

[9] C. Wang, J. Wang, Y. Shen, and X. Zhang, “Autonomous navigation of uavs in large-scale complex environments: A deep reinforcement learning approach,” *IEEE Transactions on Vehicular Technology*, vol. 68, no. 3, pp. 2124–2136, 2019.

[10] C. Wang, J. Wang, J. Wang, and X. Zhang, “Deep reinforcement learning-based autonomous uav navigation with sparse rewards,” *IEEE Internet of Things Journal*, 2020.

[11] R. Groebel, A. Chander, K. Holzinger, F. Lecue, Z. Akata, S. Stumpf, P. Kieseberg, and A. Holzinger, “Explainable ai: the new 42?” in *International cross-domain conference for machine learning and knowledge extraction*. Springer, 2018, pp. 295–303.

[12] A. B. Arrieta, N. Díaz-Rodriguez, J. Del Ser, A. Bennetot, S. Tabik,
Fig. 12. Feature analysis over the trajectory.

A. Barbado, S. García, S. Gil-López, D. Molina, R. Benjamins et al., “Explainable artificial intelligence (xai): Concepts, taxonomies, opportunities and challenges toward responsible ai,” Information Fusion, vol. 58, pp. 82–115, 2020.

[13] A. Heuillet, F. Couthouis, and N. D. Rodríguez, “Explainability in deep reinforcement learning,” arXiv preprint arXiv:2008.06693, 2020.

[14] Z. Juozapaitis, A. Koul, A. Fern, M. Érwig, and F. Doshi-Velez, “Explainable reinforcement learning via reward decomposition.” in IJCAI/ECAI Workshop on Explainable Artificial Intelligence, 2019.

[15] R. Pocius, L. Neal, and A. Fern, “Strategic tasks for explainable reinforcement learning,” in Proceedings of the AAAI Conference on Artificial Intelligence, vol. 33, 2019, pp. 10 007–10 008.

[16] J. H. Lee, “Complementary reinforcement learning towards explainable agents,” arXiv preprint arXiv:1901.00188, 2019.

[17] B. Beyret, A. Shafti, and A. A. Faisal, “Dot-to-dot: Explainable hierarchical reinforcement learning for robotic manipulation,” arXiv preprint arXiv:1904.06703, 2019.

[18] P. Madumal, T. Miller, L. Sonenberg, and F. Vetere, “Explainable reinforcement learning through a causal lens,” arXiv preprint arXiv:1905.10958, 2019.

[19] ——, “Distal explanations for explainable reinforcement learning agents,” arXiv preprint arXiv:2001.10284, 2020.

[20] A. Shrikumar, P. Greenside, A. Shcherbina, and A. Kundaje, “Not just a black box: Learning important features through propagating activation differences,” arXiv preprint arXiv:1605.01713, 2016.

[21] M. Sundararajan, A. Taly, and Q. Yan, “Axiomatic attribution for deep networks,” arXiv preprint arXiv:1703.01365, 2017.

[22] S. M. Lundberg and S.-I. Lee, “A unified approach to interpreting model predictions,” in Advances in neural information processing systems, 2017, pp. 4765–4774.

[23] A. Shrikumar, P. Greenside, and A. Kundaje, “Learning important features through propagating activation differences,” arXiv preprint arXiv:1704.02685, 2017.

[24] H. Chen, S. Lundberg, and S.-I. Lee, “Explaining models by propagating shapley values of local components,” arXiv preprint arXiv:1911.11888, 2019.

[25] M. D. Zeiler and R. Fergus, “Visualizing and understanding convolutional networks,” in European conference on computer vision. Springer, 2014, pp. 818–833.

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

[27] J. T. Springenberg, A. Dosovitskiy, T. Brox, and M. Riedmiller, “Striving for simplicity: The all convolutional net,” arXiv preprint arXiv:1412.6806, 2014.

[28] B. Zhou, A. Khosla, A. Lapedriza, A. Oliva, and A. Torralba, “Learning deep features for discriminative localization,” in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), June 2016.

[29] R. R. Selvaraju, M. Cogswell, A. Das, R. Vedantam, D. Parikh, and 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.

[30] S. Shah, D. Dey, C. Lovett, and A. Kapoor, “Airsim: High-fidelity visual and physical simulation for autonomous vehicles,” in Field and Service Robotics, 2017. [Online]. Available: https://arxiv.org/abs/1705.09565

[31] S. Fujimoto, H. Van Hoof, and D. Meger, “Addressing function approximation error in actor-critic methods,” arXiv preprint arXiv:1802.09477, 2018.

[32] J. Achiam, “Spinning Up in Deep Reinforcement Learning,” 2018.