Integrating Deep Learning and Explainability in Game Development

# 1st Dr. Jeya Mala

*School of Computer Science and Engineering Vellore Institute of Technology Chennai* Chennai, India

https://orcid.org/0000-0002-2100-8218

# 3rd Mhanjhusriee Baskar

*School of Computer Science and Engineering Vellore Institute of Technology Chennai* Chennai, India

[mhanjhu15@gmail.com](mailto:mhanjhu15@gmail.com)

# 2nd Nathan Flex

*School of Computer Science and Engineering Vellore Institute of Technology Chennai* Chennai, India

[nathanflex03@gmail.com](mailto:nathanflex03@gmail.com)

# 4th Prachi Acharya

*School of Computer Science and Engineering Vellore Institute of Technology Chennai* Chennai, India [prachiacharya1111@gmail.com](mailto:prachiacharya1111@gmail.com)

***Abstract*—In recent years, deep learning (DL) has emerged as a powerful tool for solving complex decision-making prob- lems by enabling agents to learn from their interactions with the environment. This paper presents the development of two games using DL techniques to train agents and explores the integration of explainability models, specifically SHAP (Shap- ley Additive Explanations), EigenCAM (Eigen Class Activation Map), Saliency Maps, and counterfactual explanations, to in- terpret and understand the decision-making processes of the trained agents. By employing these explainability techniques, the study not only demonstrates the effectiveness of DL in game development but also provides valuable insights into the agents’ behavior, highlighting key features that influence their actions. Experimental results show that explainability models can identify the most critical factors guiding the agents’ decisions, leading to improvements in model transparency, performance, and user trust in AI-driven game systems.**

***Index Terms*—Reinforcement learning, Explainable AI, SHAP, Counterfactual explanations, Game development, Model inter- pretability,Deep learning, Pygame, EigenCAM.**

1. Introduction

The rapid advancement of artificial intelligence (AI) has revolutionised various fields, with deep learning (DL) playing a pivotal role in enhancing gameplay and decision-making within the gaming industry. DL has been employed to develop autonomous agents that can learn from and adapt to dynamic game environments, leading to breakthroughs in both classic and modern video games. These agents, capable of mastering complex strategies through trial and error, have significantly elevated the overall gaming experience by offering more challenging and interactive opponents. However, despite DL’s success in producing high-performing agents, its decision- making process remains largely opaque, operating as a black- box model. This lack of transparency presents challenges for developers and users in understanding the reasoning behind an agent’s actions, which is particularly important in dynamic gaming environments where quick decisions impact fairness, strategy refinement, and overall user trust. As the complexity

of AI systems grows, so does the need for interpretability. Explainable AI (XAI) has emerged as a critical field, aimed at providing clearer insights into how AI models, particularly those powered by DL, arrive at their decisions. This is es- pecially vital in gaming, where understanding the rationale behind in-game actions can significantly improve debugging, model refinement, and player engagement. Techniques such as SHAP (Shapley Additive Explanations) and counterfactual explanations have gained prominence for their ability to shed light on individual feature contributions, offering developers the tools to demystify the actions of reinforcement learning agents. The objective of this research is to integrate explain- ability into DL-driven game development by applying these XAI techniques to two custom-built games: the Fruit Catcher and the Shooter. Through a detailed analysis of the decision- making processes in these games, the study aims to create a framework that not only produces efficient game agents but also ensures that their actions are transparent and interpretable. By providing clear explanations of agent behaviour, this re- search contributes to more trustworthy and user-friendly AI systems in gaming. Additionally, it underscores the importance of explainability for real-time debugging and targeted model improvements, facilitating broader adoption of DL technolo- gies in more complex, real-world gaming scenarios.

1. Literature Review

In recent research on explainable reinforcement learning (XRL), several innovative methods have emerged to enhance model interpretability. Linear Model U-Trees (LMUTs), a method that integrates linear models within U-Trees, allowing reinforcement learning policies to remain interpretable without sacrificing performance, is introduced in [3]. This approach is effective in various RL environments and holds potential for scaling to more complex domains.

The authors of [7] introduce contrastive explanations, a novel method that explains why certain actions are preferred by

highlighting the expected outcomes of alternative choices. This approach proves particularly useful in helping users under- stand agent behavior in real-time environments. Additionally, Shapley counterfactual credits, a method for attributing credit to individual agents in multi-agent RL, is introduced in [10]. By combining Shapley values with counterfactual reasoning, this method provides a more interpretable way to assess each agent’s contribution, improving transparency in complex systems.

Various attention mechanisms in reinforcement learning, which allow agents to focus on key features or states during decision-making, offering a more transparent understanding of the agent’s decision process, are explored in [8]. Lastly, the general concept of model interpretability, arguing for a more nuanced understanding of the trade-offs between accuracy, complexity, and transparency, is critiqued in [5]. These diverse approaches contribute to a deeper understanding of how to bal- ance interpretability with model complexity in reinforcement learning, with implications for both theoretical development and practical applications.

1. Summary of findings

In this research, we evaluated the performance and inter- pretability of Deep Q-Network (DQN) models trained for a Fruit Catcher game across three different epochs (1, 10, 100) using SHAP and counterfactual explanations. SHAP consistently demonstrated high precision similarity but showed decreasing decision stability with increased training, while counterfactuals revealed variations in sparsity, proximity, and validity, highlighting areas for model improvement. The study also compared unambiguity and inference times, finding SHAP to be stable and computationally efficient, while counterfactu- als required more processing time as model complexity grew. Additionally, in a Shooter Game, cosine similarity compar- isons between Heat map and EigenCAM techniques showed that Heat Maps provided more consistent explanations. These insights underscore the potential of combining explainability techniques like SHAP and counterfactuals to refine reinforce- ment learning models, enhance decision-making, and improve model transparency for real-world applications.

1. Methodology

This paper aims to integrate explainability of AI models with Deep Neural Networks through games. A corresponding dataset of game states has been collected from each game, on which Deep NN models have been trained. These models have then been analyzed via XAI methods in order to understand their functioning. The results have then been analyzed by calculating select evaluation parameters which determine the trustworthiness of the explanation.

As in Fig.1, two models have been selected to perform XAI analysis on.

1. Deep Q-Learning Network:

DQN combines Q-learning, a value-based method, with deep neural networks to approximate the Q-values for each possible action the agent can take in a given state.

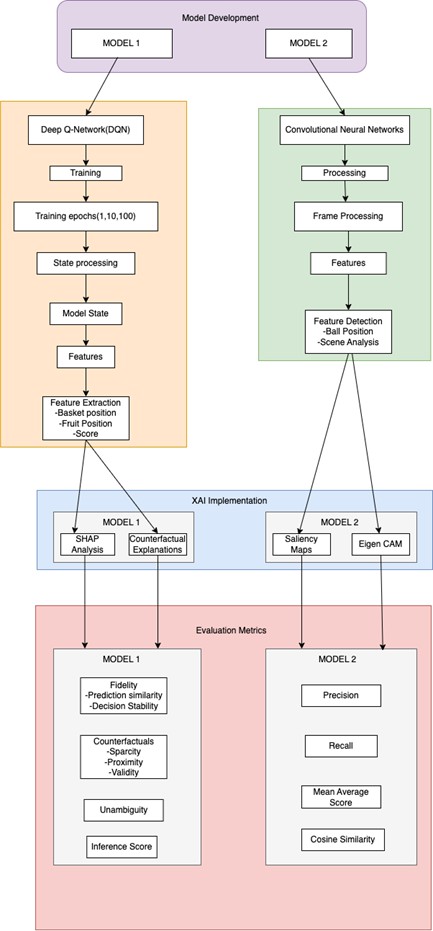


Fig. 1. Architecture Diagram

The agent utilizes a neural network to approximate the Q-values (Q(s,a;*θ*)). The agent was initialized with a specific state shape (stateshape), action size (|A|), and hyper parameters like learning rate decay (*α*) and discount factor gamma (*γ*).

*Q*(*s, a*) ← *Q*(*s, a*) + *α*(*r* + *γa′maxQ*(*s′, a′*) − *Q*(*s, a*))

1. YOLOv8 CNN Model:

It uses a single convolutional neural network (CNN) to predict both bounding boxes and class probabilities

from an input image. The architecture incorporates a backbone (for feature extraction), a neck (for feature aggregation), and a head (for prediction). YOLOv8 improves efficiency with a CSP (Cross Stage Partial) network backbone and integrates anchor-free detection, which allows for better generalization. The model is evaluated based on three parameters:

*Tp*

most to the model’s final decision. For the shooter game, EigenCAM provides an intuitive way to visualize how the CNN layers are processing the image, showing what elements in the image the model considers important to predicting the position of the target. This is particularly useful for understanding the deeper layers of the CNN and ensuring that the model has learned the correct features for targeting the object.

*Precision* =

*Tp*

+ *Fp*

Finally, to evaluate these explanations, we used the follow- ing parameters:

*Tp*

*Recall* =

*Tp* + *Fn*

1 Σ

*k*

1. Fidelity:

The ability of the model to generate true explanations for model predictions. To compare fidelity for our

*mAP* =

*n*

∗*APi*

*i*

models, the features that were marked important were masked and the input was then passed through the

In order to understand the predictions given by the models, we selected the following XAI methods:

* 1. SHAP Analysis:

The SHAP (Shapley Additive Explanations) method was integrated to explain the decisions made by the RL agent. SHAP values (*ϕi*) were calculated to visualize and interpret the impact of different features on the agent’s actions, providing insights into the model’s decision-making process. The generated SHAP heat maps highlighted the significance of various features, helping to understand the underlying factors influencing the agent’s behavior. These values where computed using the below formula that computes the weighted average contribution of feature across all possible subsets of features, quantifying how much contributes to the difference in the model’s output

* 1. Counterfactual Explanations:

Counterfactual explanations were used to identify the key positions that the model should have considered for making different decisions. The goal was to find a ”counterfactual” state *scf* where the agent’s decision *acf* would differ from the action a taken in the current state s.

* 1. Saliency Maps:

Used to determine which pixels of the input image are most critical for the model when predicting the coordinates of the target. By computing the gradient of the output with respect to the input image, the saliency map highlights regions that most influence the model’s prediction. This allows developers to verify whether the model is correctly focusing on the target object and ignoring irrelevant parts of the scene, such as the background or other objects.

* 1. EigenCAM:

A variation of Class Activation Maps (CAM) that uses principal component analysis (PCA) to generate a heat map indicating which regions of an image contribute the

model again to observe any difference in the model’s output.

1. Unambiguity:

The clarity or precision of the explanations provided by the XAI model to avoid confusion caused by multiple interpretations. To calculate unambiguity, similar states for each game were fed to each model, and the re- spective explanations were compared through similarity measures.

1. Experimentations
2. *Game 1: Fruit Catcher*

The Fruit Catcher game is a dynamic and interactive game where a player controls a basket, which moves horizontally at the bottom of the screen. The objective is to catch falling fruits to score points. Fig. 2 contains a screenshot of a sample game state

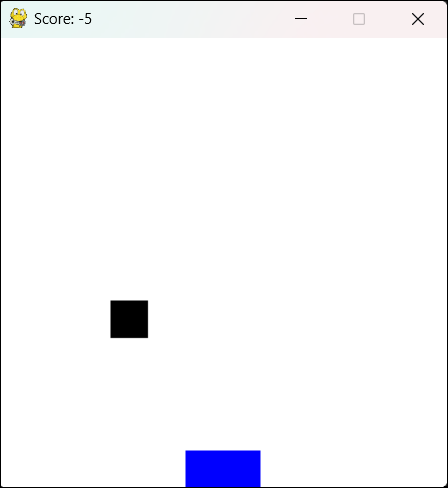


Fig. 2. The game.

The model processes the game screen as input and then extracts spatial features, which are crucial for identifying the position of fruits and the basket. These features are then

passed through fully connected layers that map them to Q- values representing possible actions like moving the basket left or right. The agent learns to maximize rewards (catching fruits) by minimizing negative feedback (missing fruits) over time through trial and error. This combination of RL and deep learning allows the agent to improve its performance autonomously.

1. *Game 2: Shooter Game*

The Shooter Game mimics the mechanics of a first-person shooter (FPS) where the objective is to shoot your target quickly and as accurately as possible. The player is placed in an open 3D virtual space with a red ball appearing at random positions mid-air. Fig. 3 contains a screenshot of the game’s environment.

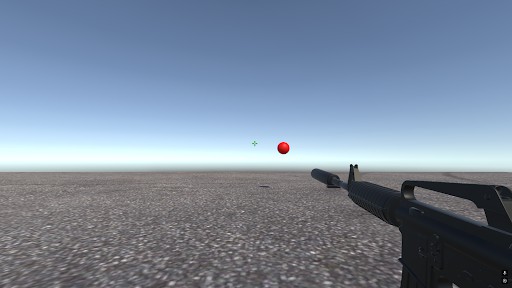


Fig. 3. The game.

The deep learning model, deployed in a Unity environment, has been trained using Python and a custom image dataset. The dataset consists of scenes where the red ball appears in various positions in mid-air. A convolutional neural network (CNN) was utilized for this task, which processes the images frame by frame and outputs the predicted coordinates of the ball. The agent then moves the mouse cursor to these coordinates and clicks on the target.

1. Results
   1. *Fruit Catcher Game*

The DQN model has been trained for the Fruit Catcher Game under three configurations:

* + - 1 epoch
    - 10 epochs
    - 100 epochs

*1) (i) 1 Epoch Model:* At the 1-epoch stage, the model demonstrated incorrect decision-making for the given state. When the fruit appeared on the right side of the screen, the model mistakenly moved the basket to the left, contrary to the optimal action of moving it to the right (Fig. 4).

To investigate this error, we analysed the SHAP values associated with the decision. The analysis showed that the basket’s position at coordinates (11,4) and (11,3) had a positive influence on the model’s decision, leading it to incorrectly



Fig. 4. Model’s decision at 1 epoch, moving the basket to the left despite the fruit being on the right.

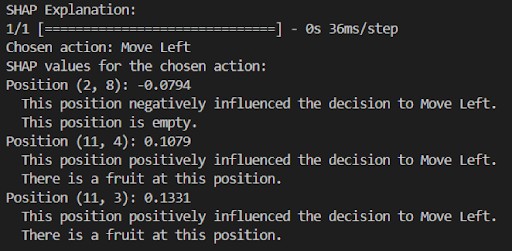


Fig. 5. SHAP analysis indicating the positive influence of basket positions at (11,4) and (11,3) on the decision.

assume a fruit was present at these positions (Fig. 5). This behaviour indicates that the model had not yet learned to accurately associate the position of the fruit with the correct movement of the basket, highlighting the need for further training to improve its decision-making capabilities.

To address this, we also applied counterfactual analysis by perturbing the state and identifying the closest state that resulted in the correct decision. Fig. 6 illustrates the perturbed state, while the third graph of Fig. 6 shows the weight adjustments required in the original state to achieve the correct outcome. We also provided a textual summary of which positional values should be modified to guide future training efforts. This analysis is key to improving the model’s decisions by focusing on specific feature adjustments.

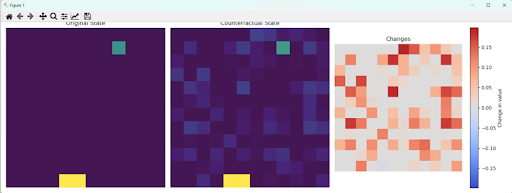


Fig. 6. Counterfactual analysis showing the perturbed state and necessary weight adjustments.

1. *10 Epochs:* With further training, we observed that the model’s performance improved gradually. The fruit’s position started to positively influence the decision-making process, leading to more accurate actions in some states (Fig. 7).



Fig. 7. Improved decision-making with some accurate actions at 10 epochs.

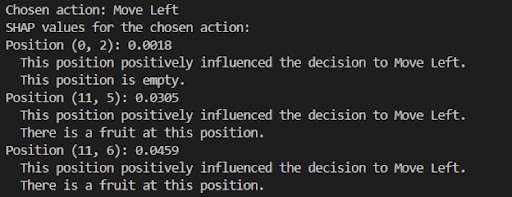


Fig. 8. Example of an inconsistent decision made by the model at 10 epochs.

However, the model was still inconsistent and continued to make occasional errors, such as the one illustrated in Fig. 7.



Fig. 9. Example of an inconsistent decision made by the model at 10 epochs.

To further understand the model, counterfactual analysis was applied, highlighting which aspects of the state needed more focus to consistently achieve the correct decision (Fig. 8).

1. *100 Epochs:* After training for 100 epochs, there was a significant improvement in the model’s decision-making. The position of the fruit became a dominant factor, greatly influ- encing the model’s decisions across most states (Fig. 9 and 10). This shift marked a substantial improvement in the model’s ability to accurately interpret game states. By analyzing these results, we can further enhance model performance through

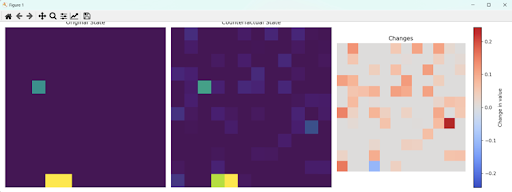


Fig. 10. Counterfactual analysis showing the state adjustments needed for correct decision-making.

an in-depth understanding of how various features affect its behavior and decisions.

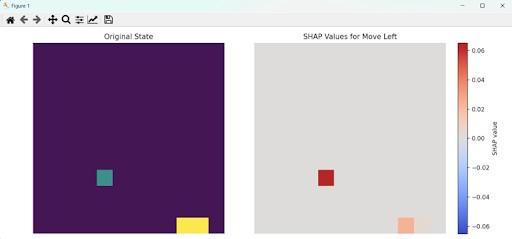


Fig. 11. Significant improvement in decision-making at 100 epochs, with the fruit’s position greatly influencing the model’s actions.

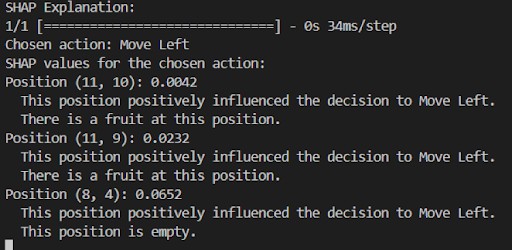


Fig. 12. Further analysis showing the dominant influence of the fruit’s position on the model’s decisions at 100 epochs.

* 1. *Shooter Game Model Explanations:*

1. *Saliency Maps:* In Figure 13, a heatmap is used to illustrate the regions of the image that the model considers important for identifying the target class, which in this case is the ball. The highlighted portion in orange corresponds to the areas where the model focused its attention when making its classification. This visualisation provides insight into the model’s interpretability by showing how it associates different regions of the image with the presence of the target object.

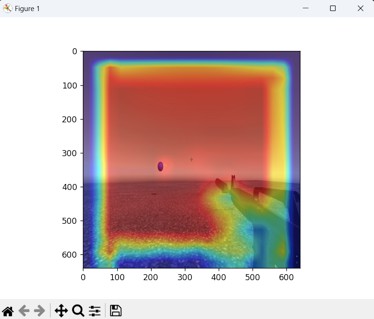
 

Fig. 13. Heat map showing the regions of the image considered important for identifying the target class (ball).

Fig. 15. EigenCAM visualisations for different layers: Final convolutional layer, Final concatenation layer.

1. *EigenCAM:* Figures 14, and 15 present EigenCAM visualisations that illustrate the model’s focus during the object detection task. They specifically show two different layers of the model: the final convolutional layer (third last layer) and the final concatenation layer (second last layer), respectively. In these visualisations, areas marked in red indicate the pixels the model deems most important for detecting the object. This method helps us understand which parts of the image are contributing to the model’s decision-making process, thus providing a better grasp of the model’s inner workings.

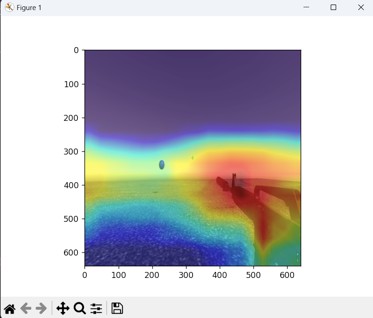


Fig. 14. EigenCAM visualisations for different layers: Final convolutional layer, Final concatenation layer.

1. Evaluation of XAI Models
2. *Fruit Catcher Game*

In Table I, for SHAP, Precision Similarity remains con- sistently high across all epochs, with values around 0.9966 to 0.9972, indicating that the model’s feature importance explanations are highly similar and precise as training pro- gresses. However, Decision Stability decreases significantly from 0.9610 at 1 epoch to 0.3920 at 100 epochs, suggesting that the model’s decisions become less stable and consistent as the complexity of the model increases over time.

TABLE I

Fidelity Analysis

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Models** | **SHAP** | | **Counterfactual** | | |
| ***Precision***  ***Similarity*** | ***Decision***  ***Stability*** | ***Sparsity*** | ***Proximity*** | **Validity** |
| 1-Epoch | 0.9966 | 0.9610 | 0.4160 | 0.0630 | 1.0000 |
| 10-Epoch | 0.9972 | 0.7420 | 0.3993 | 0.4383 | 0.6000 |
| 100-Epoch | 0.9971 | 0.3920 | 0.3778 | 0.4178 | 0.8000 |

In contrast, for counterfactual explanations, variations are observed across the metrics:

* + Sparsity slightly decreases from 0.4160 at 1 epoch to 0.3778 at 100 epochs, indicating a slight increase in the complexity of counterfactual explanations (more features being perturbed to generate valid explanations).
  + Proximity increases notably between 1 and 10 epochs, from 0.0630 to 0.4383, showing an improvement in gen- erating counterfactuals that are closer to the original state. However, it slightly decreases at 100 epochs (0.4178), implying some loss of proximity as the model continues to evolve.
  + Validity fluctuates, starting at a perfect score of 1.0000 at 1 epoch, decreasing to 0.6000 at 10 epochs, and then rising again to 0.8000 at 100 epochs. This suggests that while the counterfactuals are initially valid, the model struggles with validity in mid-training, before regaining accuracy later on.

TABLE II

Unambiguity Analysis

|  |  |  |
| --- | --- | --- |
| **Epochs** | **SHAP Consistency** | **Counterfactual Consistency** |
| 1 Epoch | 1.0000 | 0.6040 |
| 10 Epochs | 1.0000 | 0.5913 |
| 100 Epochs | 1.0000 | 0.5055 |

In Table II, for SHAP, consistency remains perfect across all epochs, maintaining a value of 1.0000 from 1 epoch to 100 epochs, indicating that SHAP consistently provides clear and unambiguous explanations, regardless of the model’s complexity. Counterfactual consistency shows a decline as

the number of training epochs increases, indicating that the complexity of the model’s decision boundaries increases the ambiguity in the explanations.

TABLE III

Inference Time

|  |  |  |
| --- | --- | --- |
| **Epochs** | **SHAP Time (s)** | **Counterfactual Time (s)** |
| 1 Epoch | 0.0630 | 0.5534 |
| 10 Epochs | 0.0759 | 0.7309 |
| 100 Epochs | 0.0594 | 0.8068 |

In Table III, for SHAP, the inference time remains relatively stable across epochs, ranging from 0.0630 seconds at 1 epoch to 0.0594 seconds at 100 epochs. This slight fluctuation indicates that SHAP’s computational efficiency is not significantly impacted by the increased complexity of the model as training progresses.

In contrast, the inference time for counterfactual explanations increases steadily as the model is trained for more epochs. The time rises from 0.5534 seconds at 1 epoch to 0.8068 seconds at 100 epochs. This trend suggests that generating counterfactual explanations becomes more computationally intensive as the model evolves, likely due to the growing complexity of the decision boundaries and feature interactions over longer training durations.

Overall, while SHAP maintains relatively consistent performance, counterfactual explanations require more time as the model’s complexity increases, highlighting the trade-off between explanation depth and computational cost.

1. *Shooter Game*

TABLE IV

Unambiguity Analysis

|  |  |
| --- | --- |
| **Model** | **Cosine Similarity** |
| Heatmap | 1 |
| EigenCAM | 0.831 |

In Table IV, the heatmap method shows a perfect cosine similarity score of 1, indicating that the explanations produced are highly consistent and closely aligned with the model’s feature importance. This suggests that the heatmap provides reliable and accurate insights into how the model interprets input data. The EigenCAM method, with a cosine similarity score of 0.831, still offers strong interpretability but is slightly less consistent compared to the heatmap. While EigenCAM highlights relevant features well, the lower score indicates that its explanations may not fully align with the model’s internal decision-making as precisely as the heatmap. In summary, both methods provide valuable explanations, but the heatmap is more aligned with the model’s understanding of the input, offering higher interpretability.

1. Implications for Future Research

The current study opens several avenues for future research, particularly in enhancing the interpretability and performance of reinforcement learning models. One promising direction is the use of counterfactual explanations to refine the Deep Q- Network (DQN) architecture. Counterfactual explanations can be utilised to identify specific scenarios where the model’s decisions deviate from the optimal policy, providing insights into why certain actions were taken. By understanding these deviations, researchers can pinpoint areas of the network that require adjustment, leading to more precise weight assignment during the training process.

Improving weight assignment based on counterfactual insights could lead to a more robust learning framework, where the model can better generalise across diverse states and actions. Additionally, integrating counterfactual explanations can facil- itate more targeted fine-tuning of the network’s hyperparam- eters, such as learning rates and exploration strategies, poten- tially accelerating the convergence of the training process.

Future work could also explore combining counterfactual ex- planations with other explainability techniques, such as SHAP or Grad-CAM, to gain a more comprehensive understanding of the model’s decision-making processes across different layers of the network. This multi-faceted approach to model inter- pretability would not only improve the accuracy and reliability of the DQN but also make reinforcement learning models more transparent and accessible for real-world applications.

1. Conclusions

In this study, we explored the application of Explain- able AI (XAI) techniques to improve the interpretability and performance of deep learning (DL) models in two distinct games: the Fruit Catcher and the Shooter. Through the use of SHAP, EigenCAM, and counterfactual explanations, we gained insights into the model’s decision-making processes, allowing us to identify key features affecting performance and make targeted improvements. Specifically, the SHAP analysis revealed that the initial Fruit Catcher model prioritised the basket’s position over the fruit’s, which was corrected by retraining the model with a balanced focus on both features by training it for more epochs. Additionally, the counter- factuals offered valuable insights into the specific changes needed to correct the model’s decisions when it made an incorrect prediction. These findings underscore the critical role of explainability in DL, as it allows developers to diagnose and rectify performance issues that may not be apparent from traditional evaluation metrics alone. The integration of explainability into DL systems not only helps improve model accuracy but also provides transparency, which is essential for real-world applications, especially in environments where safety and fairness are paramount.

References

1. Erika Puiutta, Eric MSP Veith, *Explainable Reinforcement Learning: A Survey*, OFFIS – Institute for Information Technology, Escherweg 2, 26121 Oldenburg, Germany.
2. Julius Adebayo, Michael Muelly, Ilaria Liccardi, Been Kim, *Debugging Tests for Model Explanations*, Massachusetts Institute of Technology, Google Inc, 2020.
3. Guiliang Liu, Oliver Schulte, Wang Zhu, Qingcan Li, *Toward In- terpretable Deep Reinforcement Learning with Linear Model U-Trees*, arXiv:1807.05887v1 [cs.LG], 16 July 2018.
4. Andrew Silva, Taylor Killian, Ivan Rodriguez Jimenez, *Optimization Methods for Interpretable Differentiable Decision Trees in Reinforcement Learning*, Office of Naval Research under grant N00014-19-1-2076 and by MIT Lincoln Laboratory.
5. Zachary C. Lipton, *The mythos of model interpretability*, Queue, 16(3):31–57, 2018.
6. David A. McAllester, Satinder P. Singh Sutton, Richard S., Yishay Mansour, *Policy gradient methods for reinforcement learning with function approximation*, in NIPS, pages 1057–1063.
7. Jasper van der Waa, Jurriaan van Diggelen, Karel van den Bosch, Mark Neerincx, *Contrastive explanations for reinforcement learning in terms of expected consequences*, Proceedings of the IJCAI/ECAI 2018 Workshop on Explainable AI, 2018.
8. Alex Mott, Daniel Zoran, Mike Chrzanowski, Daan Wierstra, Danilo Jimenez Rezende, *Towards interpretable reinforcement learning using attention augmented agents*, in NeurIPS, 2019.
9. Peng Jin, Jiaxu Tian, Dapeng Zhi, Xuejun Wen, Min Zhang, *TRAINIFY: A CEGAR-Driven Training and Verification Framework for Safe Deep Reinforcement Learning*, DOI, https://doi.org/10.1007/978-3-031-13185-1- 10.
10. J. Li, K. Kuang, B. Wang, F. Liu, L. Chen, F. Wu, J. Xiao, *Shapley counterfactual credits for multi-agent reinforcement learning*, SIGKDD, 2021.
11. Renan A. Rojas-Gomez, Raymond A. Yeh, Minh N. Do, Anh Nguyen, *Inverting Adversarially Robust Networks for Image Synthesis*, 16th Asian Conference on Computer Vision (ACCV 2022).
12. Shusen Liu, Bhavya Kailkhura, Donald Loveland, Yong Han, *Gen- erative Counterfactual Introspection for Explainable Deep Learning*, arXiv:1907.03077, 2019.
13. Jon Vadillo, Roberto Santana, Jose A. Lozano, *When and how to fool explainable models (and humans) with adversarial examples*, arXiv:2107.01943.
14. Meike Nauta, Ron van Bree, Christin Seifert, *Neural prototype trees for interpretable fine-grained image recognition*, arXiv:2012.02046, 2020.
15. Jonathan Dinu, Jeffrey Bigham, J. Zico Kolter, *Challenging common interpretability assumptions in feature attribute explanations*, NeurIPS 2020 ML-Retrospectives, Surveys Meta-Analyses Workshop, 2020.