Supplementary Material for "Outcome-Based Semifactuals for Reinforcement Learning"

Yanzhe Bekkemoen YANZHE.BEKKEMOEN@NTNU.NO
Helge Langseth HELGE.LANGSETH@NTNU.NO
Norwegian University of Science and Technology

Editors: Hung-yi Lee and Tongliang Liu

Appendix A. Desiderata for Semifactual Explanations

In this appendix, we discuss the desiderata of semifactual explanations outlined in Aryal and Keane (2024) and Gajcin et al. (2024), which partially overlap. We first review the desiderata outlined in Aryal and Keane (2024) point-by-point:

Distance The distance between the semifactual and query state should be as far from each other as possible without crossing the decision boundary. Our method can achieve a higher distance by increasing the δ value. What is an acceptable δ value will depend on the environment, the policy, and what the user desires to see.

Plausible The semifactual state should be within-distribution and reachable. We ensure this by using simulation from the query state and not a generative approach.

Sparsity The number of features changed should be small. Small changes are not necessarily meaningful and challenging to measure and enforce for image data, where the features are pixels. In addition, small changes in pixels, as discussed in the paper, can lead to large changes to a state's semantics. Instead, we ensure sparsity by simulating from the query state.

Robustness Small changes to the query state should result in a similar semifactual state. Since a generative approach is not involved, the semifactual state found should not change much unless we are far from a critical state.

Confusability A user should understand the semifactual state to have the same outcome as the query state and not be confused with a counterfactual state. We inform the user that our semifactual state is on the border between positive and negative outcomes.

In contrast to Aryal and Keane (2024), Gajcin et al. (2024) propose another set of desiderata that are similar but tailored to reinforcement learning:

Validity Ensure the outcome stays the same in the query and semifactual state. The validity property is the definition of a semifactual. We ensure validity by using the threshold value δ.

Temporal Distance The distance in time should not be too large. Since we simulate from the query state, our outcome-based semifactual (OSF) state will stay close to the query state.

Stochastic Uncertainty The semifactual explanation should be informative and give insight regarding the decision boundary. We ensure the informativeness of our explanation by pushing the OSF as close to the decision boundary as possible.

Fidelity Ensure that the path taken to reach the semifactual state will likely be taken by the policy. We do not want to fulfill this desideratum since we wish to remove the actions that do not affect the outcome. Our method differs from Gajcin et al. (2024)'s method in what we try to convey and achieve for this property.

Exceptionality The explanation should be surprising and unexpected. This property will depend on how well the user understands the policy's decision boundary. We believe those states closest to the decision boundary will be most informative and potentially unexpected to the user.

Appendix B. Human Evaluation

We have not done user studies in our work. However, we need to do user studies to better understand the explanations and how they compare to other explainable reinforcement learning methods. In the future, we would like to use a between-subjects design where user groups test different conditions. In the experiments, we can first teach the users using explanations and then ask users to predict the policy's actions in unseen states. We can also ask the users which agent out of several agents they would choose to make choices on their behalf, based on explanations. Furthermore, the users will be asked how useful they find the explanations using the Likert scale.

Appendix C. Pseudocode and Extension

Algorithm 1 shows the pseudocode of our method, detailed in Section 4 of the paper.

In the paper, we focus on the deterministic case. To extend the method to stochastic environments, we can take inspiration from Amir and Amir (2018) and return a set of OSF states by simulating several times from the same query state. We can let a user set the number of OSF states to show for a query state by introducing a new hyperparameter T. Another way is to dynamically choose T based on how similar the OSF states we obtain are. We leave it to future work to focus on a better and more thought-through method for stochastic environments.

Appendix D. Qualitative Comparison in MsPacman

Fig. 1 shows explanations produced by our method in MsPacman. In Fig. 1(a), the agent prioritizes the power pill as the agent switches action immediately. From this explanation, we know that the agent prioritizes power pills highly as the action switching happens immediately. Fig. 1(b) shows that in some situations in MsPacman, the agent does not have

Algorithm 1 Find OSF state s'_{t+n}

Input: s_t: query state, δ: stopping criterion, Q: state-action value function, T: state transition probability function, and IG: importance gap estimation function.
Output: s_x: outcome-based semifactual state.

procedure FIND_OSF(s_t, δ, Q, T, IG)
  $z \leftarrow 0$; $i \leftarrow 0$; $s'_t \leftarrow s_t$
  while $z < \delta$ **do**
    $a_{t+i} \leftarrow \arg\max_a Q(s_{t+i}, a)$ ▷ Get action for trajectory τ
    $s_{t+i+1} \sim T(\cdot \mid s_{t+i}, a_{t+i})$ ▷ Get next state for trajectory τ
    $s'_{t+i+1} \sim T(\cdot \mid s'_{t+i}, a_t)$ ▷ Get next state for trajectory τ'
    $z \leftarrow IG(s_{t+i+1}, s'_{t+i+1} \mid a_t)$ ▷ Compute importance gap
    $i \leftarrow i + 1$
  end while
  $s'_{t+n} \leftarrow s'_{t+i-1}$
  return s'_{t+n}
end procedure

a plan when it wanders around since staying still and doing nothing for approximately 200 timesteps does not reduce the return. The behavior might indicate the agent is suboptimal.

Compared to Huber et al. (2023)'s explanation, which we show in Fig. 1 of the paper, our explanations have high plausibility as they are not generated with a generative model. Furthermore, the distance between the query and semifactual states is close since the semifactual state starts from the query state.

Appendix E. Ablation Study with the Hyperparameter δ

We show the results for additional δ values in Fig. 2. The return falls in many environments as we increase the δ value. The increase in the δ value signals that we allow the agent to take actions further away from what the agent believes is the optimal action. Hence, falling return makes sense with increasing δ values. The falling in return with increasing δ value does not apply for Assault and MsPacman shown in Figs. 2(a) and 2(c). The rising in return in these two environments might indicate that these two agents are suboptimal. To find an appropriate δ value, we can make a plot like those in Fig. 2. Afterward, select a δ value based on an acceptable loss of return using the plot.

We plot the same experiment results for action switching in Fig. 3, where we observe less action switching as δ increases. The increasing trend is to be expected as we allow the agent to care less about the difference in return.

Appendix F. Computational Complexity

This section shows 1) how long it takes to generate an OSF explanation for a single state, and 2) how running two environments affects the computational cost. Since episode lengths differ between episodes and environments, we compute the time to take a single step in the environments. Table 1 shows that it takes less than 100 ms to generate an OSF explanation

for a single state, where most are close to 10 ms. Table 2 shows that the overhead of stepping in two environments is approximately double that of a single environment ($\delta = 0.00$), but that still remains negligible in all our experiments. As our method aims to explain and not improve a policy, the additional cost shown in Table 2 will likely not hinder the adoption of our method. The improved policy performance is only a side effect of our method.

Table 1: The number of seconds it takes to generate an OSF explanation for a single state. The column headers δ refer to the threshold value in our algorithm. The uncertainty represents a 95% confidence interval.

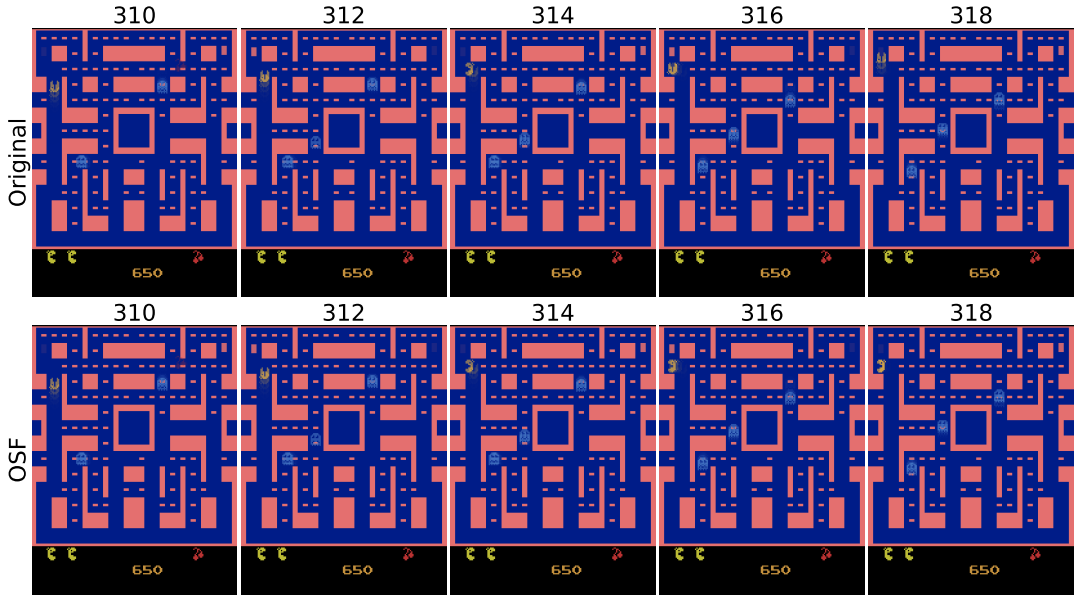
	Hyperparameter δ				
Environment	0.01	0.02	0.03	0.04	0.05
Assault	0.010 ± 0.001	0.011 ± 0.001	0.013 ± 0.001	0.012 ± 0.001	0.014 ± 0.002
Breakout	0.012 ± 0.001	0.014 ± 0.002	0.019 ± 0.003	0.018 ± 0.002	0.023 ± 0.003
MsPacman	0.015 ± 0.002	0.018 ± 0.003	0.021 ± 0.004	0.024 ± 0.005	0.031 ± 0.008
Pong	0.021 ± 0.006	0.045 ± 0.017	0.062 ± 0.025	0.057 ± 0.022	0.054 ± 0.022
Seaquest	0.012 ± 0.002	0.015 ± 0.003	0.013 ± 0.002	0.016 ± 0.003	0.019 ± 0.004
SpaceInvaders	0.015 ± 0.002	0.017 ± 0.002	0.021 ± 0.004	0.022 ± 0.003	0.023 ± 0.003

Table 2: The number of seconds to take a single step in the environment. The column headers δ refer to the threshold value in our algorithm.

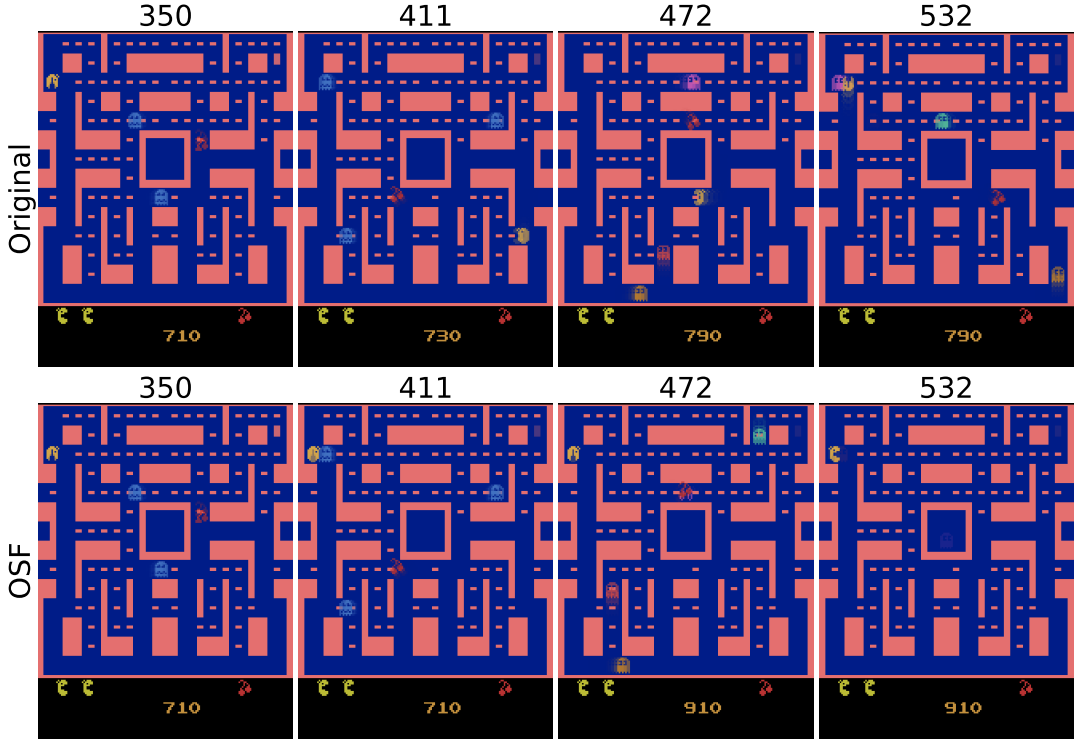
	Hyperparameter δ					
Environment	0.00	0.01	0.02	0.03	0.04	0.05
Assault	0.0014	0.0040	0.0039	0.0039	0.0039	0.0038
Breakout	0.0015	0.0041	0.0040	0.0039	0.0039	0.0039
MsPacman	0.0014	0.0038	0.0038	0.0037	0.0037	0.0036
Pong	0.0014	0.0035	0.0033	0.0032	0.0033	0.0033
Seaquest	0.0015	0.0039	0.0038	0.0040	0.0038	0.0037
SpaceInvaders	0.0015	0.0039	0.0038	0.0037	0.0038	0.0037

Appendix G. Python Libraries

We used the following Python libraries to implement our method and carry out the experiments: PyTorch (Ansel et al., 2024), NumPy (Harris et al., 2020), Matplotlib (Hunter, 2007), Gymnasium (Towers et al., 2024), Stable Baselines3 (Raffin et al., 2021), RL Baselines3 Zoo (Raffin, 2019), Tyro (Yi, 2025), tqdm (da Costa-Luis et al., 2024), OpenCV (Bradski, 2000), and Joblib (joblib developers, 2025).

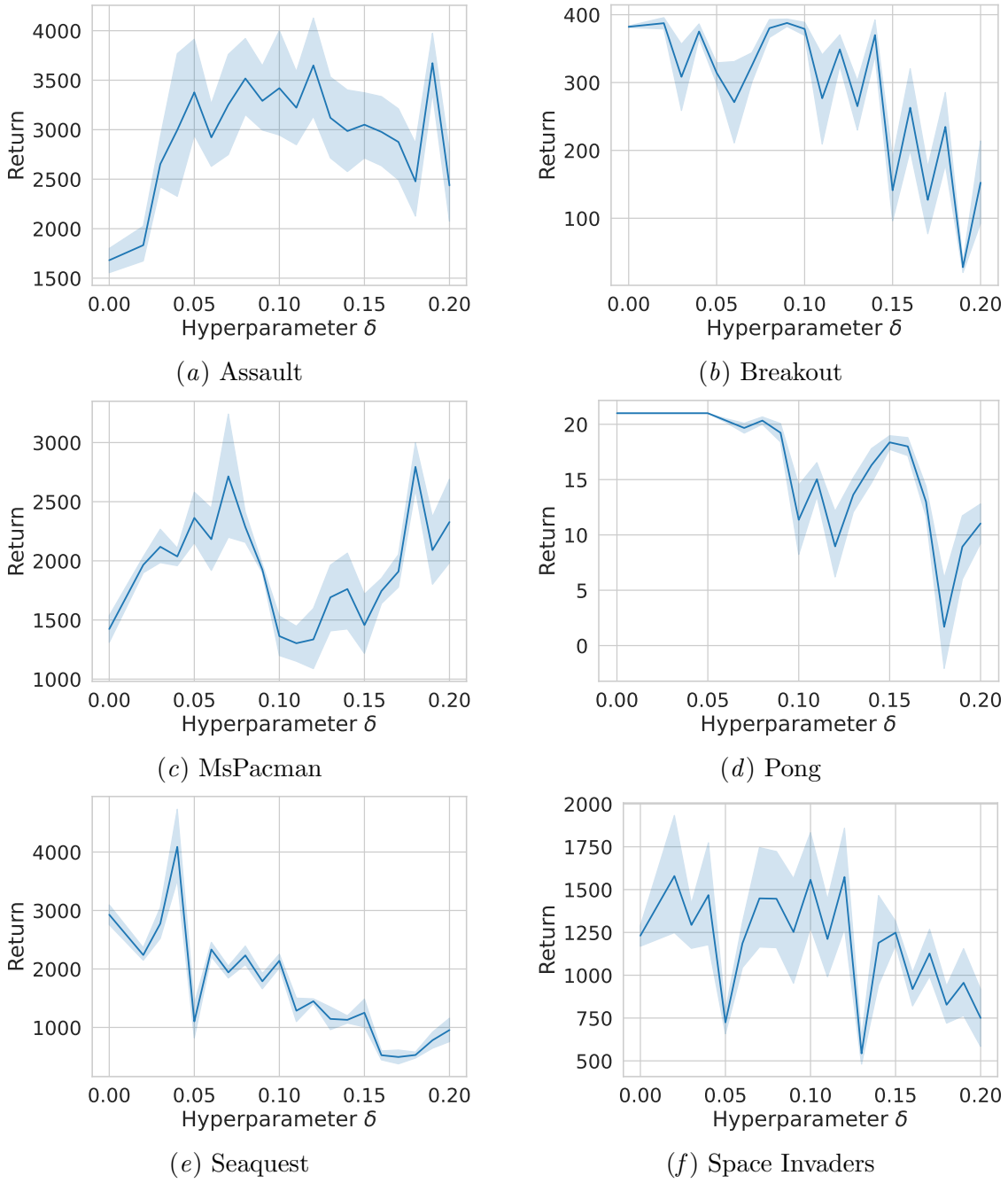


(*a*) The agent taking the power pill (Original) versus not taking it (OSF).



(*b*) The agent moving around in the environment (Original) versus doing nothing (OSF).

Figure 1: OSF explanations for the MsPacman environment. The hyperparameter δ is set to 0.2.



(a) Assault

(b) Breakout

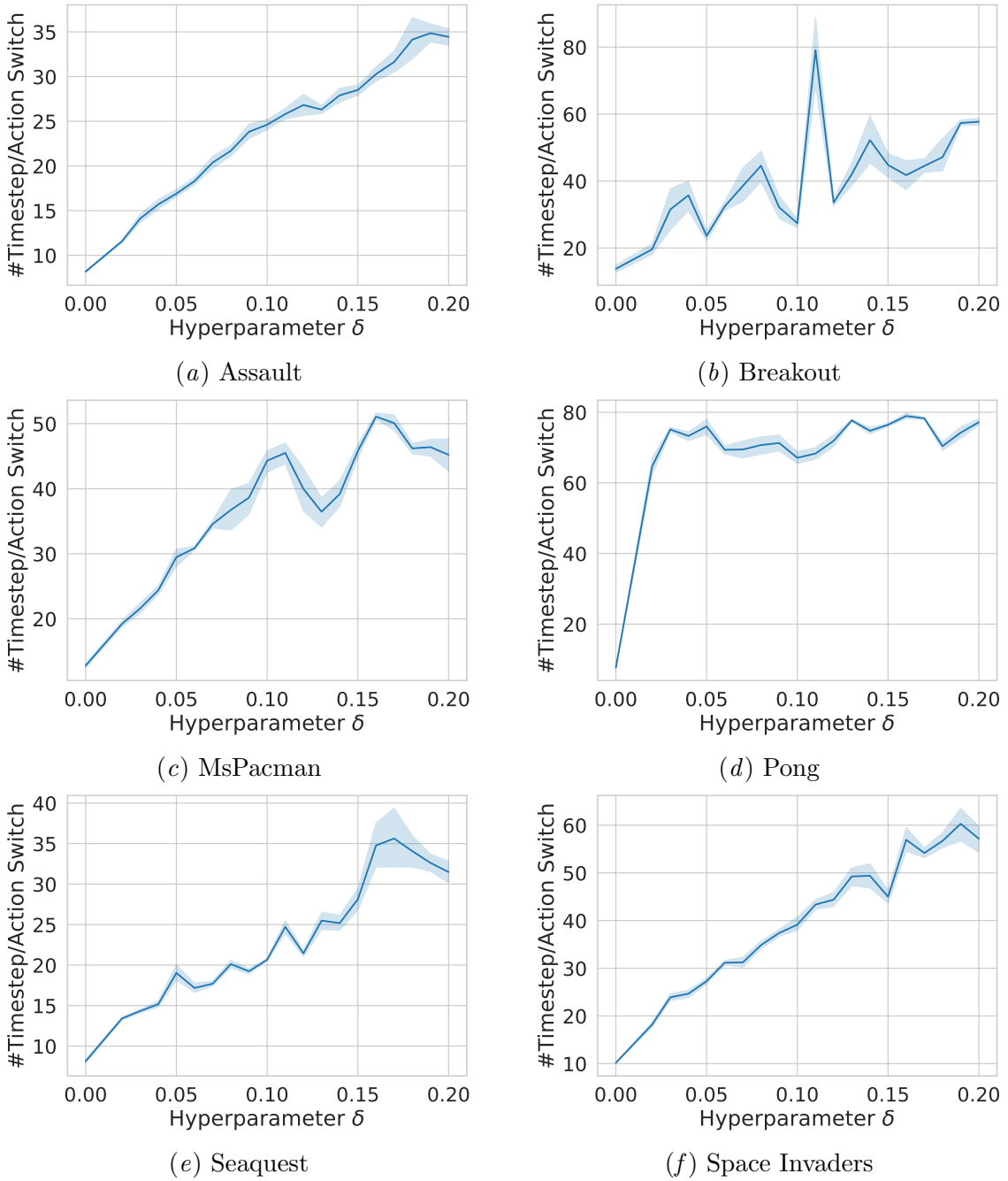
(c) MsPacman

(d) Pong

(e) Seaquest

(f) Space Invaders

Figure 2: The return as a function of the hyperparameter δ, averaged over 30 episodes. The shaded regions represent a 95% confidence interval.



(a) Assault (b) Breakout

(c) MsPacman (d) Pong

(e) Seaquest (f) Space Invaders

Figure 3: The number of timesteps, on average, the policy executes the same action consecutively before switching to a different action as a function of the hyperparameter δ, averaged over 30 episodes. The shaded regions represent a 95% confidence interval.

References

D. Amir and O. Amir. HIGHLIGHTS: Summarizing Agent Behavior to People. In *Proc. of AAMAS*, 2018. URL http://dl.acm.org/citation.cfm?id=3237869.

J. Ansel, E. Z. Yang, H. He, N. Gimelshein, A. Jain, M. Voznesensky, B. Bao, P. Bell, D. Berard, E. Burovski, G. Chauhan, A. Chourdia, W. Constable, A. Desmaison, Z. DeVito, E. Ellison, W. Feng, J. Gong, M. Gschwind, B. Hirsh, S. Huang, K. Kalambarkar, L. Kirsch, M. Lazos, M. Lezcano, Y. Liang, J. Liang, Y. Lu, C. K. Luk, B. Maher, Y. Pan, C. Puhrsch, M. Reso, M. Saroufim, M. Y. Siraichi, H. Suk, S. Zhang, M. Suo, P. Tillet, X. Zhao, E. Wang, K. Zhou, R. Zou, X. Wang, A. Mathews, W. Wen, G. Chanan, P. Wu, and S. Chintala. PyTorch 2: Faster Machine Learning Through Dynamic Python Bytecode Transformation and Graph Compilation. In *Proc. of ASPLOS*, 2024. doi: 10.1145/3620665.3640366.

S. Aryal and M. T. Keane. Even-Ifs from If-Onlys: Are the Best Semi-factual Explanations Found Using Counterfactuals as Guides? In *Proc. of ICCBR*, 2024. doi: 10.1007/978-3-031-63646-2_3.

G. Bradski. The OpenCV Library. *Dr. Dobb's Journal of Software Tools*, 2000.

Casper da Costa-Luis, Stephen Karl Larroque, Kyle Altendorf, Hadrien Mary, richardsheridan, Mikhail Korobov, Noam Yorav-Raphael, Ivan Ivanov, Marcel Bargull, Nishant Rodrigues, Shawn, Mikhail Dektyarev, Michaλ Górny, mjstevens777, Matthew D. Pagel, Martin Zugnoni, JC, CrazyPython, Charles Newey, Antony Lee, pgajdos, Todd, Staffan Malmgren, redbug312, Orivej Desh, Nikolay Nechaev, Mike Boyle, Max Nordlund, MapleCCC, and Jack McCracken. tqdm: A fast, Extensible Progress Bar for Python and CLI, 2024. URL https://doi.org/10.5281/zenodo.14231923.

J. Gajcin, J. Jeromela, and I. Dusparic. Semifactual Explanations for Reinforcement Learning. In *Proc. of HAI*, 2024. doi: 10.48550/arXiv.2409.05435.

Charles R. Harris, K. Jarrod Millman, Stéfan van der Walt, Ralf Gommers, Pauli Virtanen, David Cournapeau, Eric Wieser, Julian Taylor, Sebastian Berg, Nathaniel J. Smith, Robert Kern, Matti Picus, Stephan Hoyer, Marten H. van Kerkwijk, Matthew Brett, Allan Haldane, Jaime Fernández del Río, Mark Wiebe, Pearu Peterson, Pierre Gérard-Marchant, Kevin Sheppard, Tyler Reddy, Warren Weckesser, Hameer Abbasi, Christoph Gohlke, and Travis E. Oliphant. Array programming with NumPy. *Nat.*, 2020. doi: 10.1038/S41586-020-2649-2.

T. Huber, M. Demmler, S. Mertes, M. L. Olson, and E. André. GANterfactual-RL: Understanding Reinforcement Learning Agents' Strategies through Visual Counterfactual Explanations. In *Proc. of AAMAS*, 2023. doi: 10.48550/ARXIV.2302.12689.

John D. Hunter. Matplotlib: A 2D Graphics Environment. *Comput. Sci. Eng.*, 2007. doi: 10.1109/MCSE.2007.55.

The joblib developers. joblib, August 2025. URL https://doi.org/10.5281/zenodo.16964648.

A. Raffin. RL Baselines3 Zoo: A Training Framework for Stable Baselines3 Reinforcement Learning Agents, 2019. URL https://github.com/DLR-RM/rl-baselines3-zoo.

A. Raffin, A. Hill, A. Gleave, A. Kanervisto, M. Ernestus, and N. Dormann. Stable-Baselines3: Reliable Reinforcement Learning Implementations. *J. Mach. Learn. Res.*, 2021. URL http://jmlr.org/papers/v22/20-1364.html.

M. Towers, A. Kwiatkowski, J. Terry, J. U. Balis, G. De Cola, T. Deleu, M. Goulão, A. Kallinteris, M. Krimmel, A. KG, et al. Gymnasium: A Standard Interface for Reinforcement Learning Environments. *CoRR*, 2024. doi: 10.48550/arXiv.2407.17032.

Brent Yi. brentyi/tyro, September 2025. URL https://github.com/brentyi/tyro. original-date: 2021-10-05T08:54:08Z.