Outcome-Guided Counterfactuals from a Jointly Trained Generative Latent Space

Eric Yeh[1][0000−0002−6470−8118], Pedro Sequeira[1], Jesse Hostetler[1], and Melinda Gervasio[1]

SRI International {eric.yeh, pedro.sequeira, jesse.hostetler, melinda.gervasio}@sri.com

Abstract. We present a novel generative method for producing higher-quality counterfactual examples for decision processes using a latent space that jointly encodes observations and associated behavioral outcome variables, such as classification decisions, actions taken, and estimated values. Our approach trains a variational autoencoder over behavior traces to both reconstruct observations and predict the outcome variables from the latent encoding. The resulting joint observation and outcome latent allows for unconditioned sampling of both observations and outcome variables from this space. This grants us the ability to generate counterfactuals using multiple methods, such as gradient-driven updates to move towards desired outcomes, interpolations against relevant cases drawn from a memory of examples, and combinations of these two. This also permits us to sample counterfactuals where constraints can be placed over some outcome variables, while others are allowed to vary. This flexibility also permits us to directly address the plausibility of generated counterfactuals by using gradient-driven updates to raise the data-likelihood of generated examples. We use this method to analyze the behavior of reinforcement learning (RL) agents against several outcome variables that characterize agent behavior. From experiments in three different RL environments, we show that these methods produce counterfactuals that score higher on standard counterfactual quality measures of proximity to the query and plausibility in contrast to observation-only gradient updates and case-based baselines. We also empirically demonstrate that counterfactuals sampled from a jointly trained space are of higher quality than those from the common practice of using latents from reconstruction-only autoencoders. We conclude with an analysis of counterfactuals produced over the joint latent using combinations of latent and case-based approaches for an agent trained to play a complex real-time strategy game, and discuss future directions of investigation for this approach.

Introduction

Consider a scenario where a human user needs to decide whether a self-driving vehicle is assessing on-scene risks correctly. Feature importance methods can highlight portions of the input deemed to govern system decisions, such as the road span in front of the vehicle. While invaluable for development, they are often inadequate for conveying an actionable understanding of agent behavior to users [37]. On the other hand, *counterfactuals*—i.e., contrastive, example-based explanations that *change* specific aspects

of the environment to arrive at a different outcome—are well-aligned with how humans develop an actionable understanding of autonomous systems [21]. Several explainable AI studies have also shown them to be effective for conveying explanations about AI systems [4]. A counterfactual adding jaywalkers to the scene increases the agent's perceived risk, assuring the user that the agent recognizes the danger to the pedestrians and is aligned with human expectations.

We present a novel generative model for generating counterfactuals that focuses on "black-box" analyses of observational and behavioral data of machine learning systems. For this work, our focus is on reinforcement learning (RL) agents, although the method itself is broadly applicable beyond RL. Using a corpus of performed trajectories and corresponding outcome variables, we train a variational autoencoder [17], modified to jointly reconstruct the agent's observations and predict outcome variables that characterize its behavior. This joint observation and outcome latent permits unconditioned sampling that reflects correlations between observations and outcomes. This flexibility allows integration of different counterfactual generation methods, such as using interpolations in latent space towards a case-based example while applying gradient-driven updates to increase data-likelihood.

Ensuring plausible counterfactuals is a key issue: Outside of case-based approaches that draw counterfactuals from a database of existing instances, interpolated or synthesized counterfactuals run the risk of being implausible, which may reduce their credibility. We show that gradient adjustments in the joint latent to increase the data-likelihood of counterfactuals improves plausibility and reduces the number of concrete anomalies generated. We summarize our contributions as follows:

- A novel approach leveraging the use of a latent space jointly trained over observations and outcomes in the generation of counterfactuals, and a demonstration of its importance for improving counterfactual quality.
- A flexible framework for traversal over the latent space with different types of constraints.
- A novel gradient-driven approach using an approximation of the data-likelihood gradient that improves plausibility and decreases the number of anomalous counterfactuals.

Related Work

Automated counterfactual generation has been explored by numerous communities. One line of work has focused on case-based approaches, selecting the Nearest Unlike Neighbor (NUN), the instance that is closest to the query based on a proximity measure such as edit distance, whose outcome variable of interest has changed, e.g., has a different label associated with it [15]. In cases where a sufficiently close NUN cannot be identified, transformation templates from good counterfactual pairs in the data are used as a template for interpolation. Another approach uses feature-relevance methods like SHAP [20] to tailor a feature edit schedule for converting the query into a counterfactual [39]. As is common in counterfactual works, a proxy function is trained to predict the outcome variable given the instance and used to ensure that the synthesized

counterfactual changes the outcome. Most of this work focuses on tabular data, where data is organized into discrete features (e.g., columns) and copying values has less risk of creating an implausible instance than with less structured data, such as images.

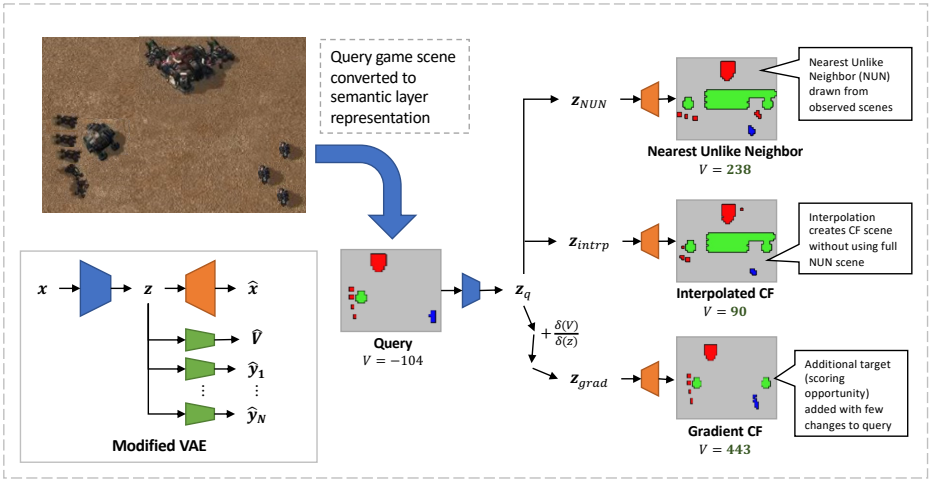

Fig. 1. A variational autoencoder is trained to both reconstruct the input and predict several outcome variables such as the agent's value function (left). As an example, given a *low-value* query instance in the StarCraft II domain encoded as spatial feature layers using PySC2 (agent units are shown in are blue; enemies, red, and assets that can be captured, green), the goal is to obtain a *higher-valued* counterfactual. The query is encoded to its latent (z_q) and counterfactuals (CF) are obtained by three methods (right): identifying the Nearest Unlike Neighbor from data (NUN, top) and then doing a partial latent interpolation to the NUN (middle), or using the gradient information to generate the example (bottom). In these actual examples produced by the methods, adding an additional target (green circle) for the agent raises the value estimate. The interpolation and gradient methods produce alternatives with fewer defending forces and placement of obstacles than the NUN.

Another body of work stems from the adversarial AI community, where input perturbation and sampling-based methods are used to generate counterfactuals [35]. In particular, a Generative Adversarial Network (GAN) was used to generate counterfactual scenes for Atari game scenes [26]. However, sampling for suitable solutions can be time-consuming, whereas gradient- or example-directed traversals can be more efficient.

Cyclic consistency approaches have also been employed to generate counterfactuals by latent manipulation [16]. However, this approach focuses on a single class transforms, and is currently not amenable to counterfactuals over different combinations of outcome variables.

Gradient descent in the feature space towards the desired outcome has been used for tabular data [23] and, in particular, in the adversarial machine learning community for generating counterfactuals over image data [22, 40]. However, these approaches run

the risk of generating adversarial counterfactuals, such as shifting a minimal set of pixels, that may be imperceptible to human users and have low utility for understanding the model. Diffusion based processes have also been explored for counterfactual generation, albeit this line of investigation has focused on pixel-level reconstructions [12], compared with our joint approach which learns a latent that encodes observations and outcomes.

Our approach is motivated by the Plug-and-Play approach, which uses gradient-derived signals from discriminators to iteratively shift the latent of generative models to produce outputs with the desired characteristics; and Plug-and-Play language models (PPLM), which add gradient adjustments to increase the likelihood of generated instances [25, 5]. Iterative gradient-based adjustment of a latent space was also explored in XGEMS [13], albeit in that case, the latent was not trained to include outcome information, and reconstructions were not adjusted for plausibility. A similar approach was taken for attribute-based perturbation in the latent space of a conditional variational autoencoder (C-VAE) for counterfactual generation [40]. This approach trained embeddings for observations and outcomes separately, concatenating them as inputs to the decoder. This requires that the full set of outcomes be known for a query and outcome, whereas the joint latent requires only the query observation and allows for adjustments that affect a subset of outcomes, leaving others to vary freely. In addition, previous approaches have neither explicitly measured nor attempted to ensure plausibility. As multiple studies have shown, generative models are not guaranteed to produce samples that are plausible from the in-distribution set, and taking measures to avoid anomalous examples is required [24]. The closest these have approached is through use of computer-vision measures of image quality, such as Fréchet Inception Divergence [8] which uses statistics over feature activations in a network to act as a form of perceptual distance. However, it is used primarily to identify unrealistic artifacts such as blurry images. In addition, these approaches also mostly ignore the issue of proximity, and focus only on whether valid counterfactuals can be produced.

Preliminaries

We now describe preliminaries for this work, starting with background on the generative model, followed by how counterfactual queries are formulated. We then describe our metrics for counterfactual effectiveness and how they are implemented in our experiments.

Variational Autoencoders

Variational autoencoders (VAEs) are probabilistic generative models that encode a high-dimensional input $\mathbf{x}$ into a lower-dimensional latent representation $\mathbf{z}$ from which the original input can be approximately reconstructed [17]. The encoder module of a VAE maps the input to its latent representation, $\mathbf{z} = enc(\mathbf{x})$, and the decoder reconstructs the input, $\mathbf{x} \approx dec(\mathbf{z})$. The latent encoding is regularized by penalizing the KL divergence from a prior distribution $q(\mathbf{z})$ (typically a standard Gaussian) to the conditional

distribution $q(\mathbf{z}|\mathbf{x})$ induced by the encoder. The VAE loss is given by

$$\mathcal{L}(\mathbf{x}) = \mathbb{E}_{q(\mathbf{z}|\mathbf{x})} \log p(\mathbf{x}|\mathbf{z}) - D_{KL}(q(\mathbf{z})||p(\mathbf{z}|\mathbf{x})),$$

where the decoder likelihood $p(\mathbf{x}|\mathbf{z})$ is typically implicit from a reconstruction loss, such as the MSE $||\mathbf{x} - dec(\mathbf{z})||^2$. It can be shown [17] that the VAE loss is a lower bound on the data likelihood, $\mathcal{L}(x) \leq \log p(\mathbf{x})$. Because the encoder distribution $q(\mathbf{z}|\mathbf{x})$ approximates the known prior distribution $q(\mathbf{x})$, samples from the input space can be generated by drawing $\mathbf{z} \sim q(\mathbf{z})$ and passing the result through the decoder.

Counterfactual Generation with VAEs

Let M denote the model whose behavior we wish to explore using counterfactual analysis. Given a query input $\mathbf{x}_q$, we want to generate a counterfactual input $\mathbf{x}_c$ that is "related" to $\mathbf{x}$ but for which M would behave differently. We quantify the behavior of M with a vector of outcome variables $\mathbf{y} = (y^{(i)})$, $i \in \{1, \ldots, N\}$. When M is a reinforcement learning agent, for example, $\mathbf{y}$ might include the value achieved by the agent, secondary performance measures such as time to reach the goal, and/or categorical measures like whether the agent violated certain constraints.

Our approach to counterfactual generation is based on perturbing the latent representation $\mathbf{z}_q$ of the query input to create a counterfactual latent representation $\mathbf{z}_c$, then decoding $\mathbf{z}_c$ to obtain a counterfactual input $\mathbf{x}_c$, exploiting the ability of VAEs to learn a latent representation space with meaningful axes of variation [9, 18]. We extend the basic VAE model to reconstruct both the input $\mathbf{x}$ and the outcome variables $\mathbf{y}$ from the latent representation $\mathbf{z}$, using a separate predictor for each outcome variable, $y^{(i)} = \sigma_i(\mathbf{z})$ (Figure 1, bottom left). Our intent is to cause the latent representation to encode information about the relationship between the input and the outcome variables, so that traversing the latent space will produce inputs that result in different outcomes. This also provides a trained predictor that can indicate when an example meets the counterfactual outcome criteria. This use of a trained proxy to determine if the outcome is met is a common practice in counterfactual generation from observational data [15].

We say that a counterfactual is *valid* if it achieves a desired change in the outcome variables, and define a validity predicate $\kappa_{i,s,\epsilon}$ that indicates whether the ith outcome variable was changed appropriately, given by

$$\kappa_{i,s,\epsilon} = \begin{cases} \mathbb{I}(y_c^{(i)} \neq y_q^{(i)}) & \text{if } y^{(i)} \text{ is categorical} \\ \mathbb{I}(y_c^{(i)} - y_q^{(i)} \geq s\epsilon) & \text{if } y^{(i)} \text{ is numeric} \end{cases},$$

where $s \in \{-1, 1\}$ is the desired sign of the difference between numeric variables, and ϵ is the desired size of the difference. For brevity, we shorten the validity predicate to κ_i for the rest of this paper. While our experiments consider only a single criterion (a single i), our approach can be extended easily to multiple criteria.

Counterfactual Quality Measures

For this work, we evaluate counterfactual generation methods along the following measures:

- **Proximity**: How different a generated counterfactual is from its query.
- **Plausibility**: Whether the counterfactual is something one would expect to observe in the domain of interest.
- **Validity**: Whether the counterfactual satisfies the counterfactual criterion κ_i.

Keeping the counterfactual similar to the original instance is important for understanding the relation between the features and the outcome variables and *proximity* is commonly measured through a variety of feature-level edit distance metrics, with counterfactuals having a *sparser* set of differences from the query being better. *Plausibility* is particularly important for systems that synthesize counterfactuals, as anomalous or implausible examples may be discounted by users [15, 21]. Is is often measured by how likely the counterfactual is to be drawn from the actual data, Finally, *validity* is necessary since automatically generated counterfactuals may not actually meet the counterfactual criterion κ_i.

We measure the inverse of proximity of a counterfactual to its query via an *observational difference* score. For categorical features, this is computed as a feature edit distance, summed over the number of label changes to convert between observations. For numeric features, we use the absolute score difference, normalized to 0-1. Formally, for an index of all features, the score $i \in I$, $\text{odiff}(x^1, x^2)$ is computed as follows:

$$\text{odiff}(\mathbf{y}_1, \mathbf{y}_2) = \sum_i \begin{cases} \mathbb{I}(y_1^{(i)} \neq y_2^{(i)}) \text{ if } i \text{ is categorical} \\ \\ \frac{|y_1^{(i)} - y_2^{(i)}|}{W^{(i)}} \qquad \text{if } i \text{ is numeric} \end{cases},$$

where $W^{(i)}$ is the interval width that normalizes the value.

Finally, there is no guarantee that a given method can produce a *valid* counterfactual, one that satisfies the criterion κ_i. We thus grade each counterfactual generation method by the fraction of queries for which it was able to produce a valid counterfactual, that is $\frac{1}{N}\sum_{i=1}^{N} \kappa_i$ for N queries.

Plausibility

We assess the plausibility of a latent $\mathbf{z}$ via an anomaly score formed from the observational difference between its decoding and that instance's reconstruction, following the observation that autoencoders act to denoise anomalous inputs [28]:

$$\text{anom}(\mathbf{z}) = \text{odiff}(dec(\mathbf{z}), dec(enc(dec(\mathbf{z}))))$$

This measures the inverse of plausibility in our experiments.[1] Using the hypothesis that autoencoders denoise their inputs, we approximate the data likelihood gradient with the reconstruction loss between the current latent's reconstruction and that scene's reconstruction using the same model:

$$\nabla_{\mathbf{z}} p(dec(\mathbf{z})) \approx -\nabla_{\mathbf{z}} ||dec(\mathbf{z}) - dec(enc(dec(\mathbf{z})))||$$

[1] We did experiment with One-Class SVMs, but performance on a SC2 Assault scene labeled for anomalous scenes was poor in comparison with the autoencoder approach.

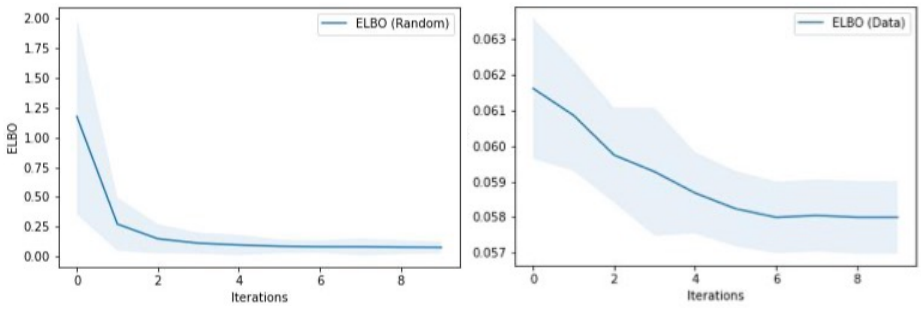

Fig. 2. Evidence lower bound (ELBO) loss against number of round trips for the input, starting from random points (left) and from data-drawn instances (right). The ELBO lower bounds the log-likelihood of the input (as the loss decreases, the input likelihood increases).

We verified the appropriateness of using this approach to increase the plausibility of generated counterfactuals by having our VAE repeatedly encode and decode its own reconstructions. Figure 2 shows the mean and standard deviation of the ELBO loss[2] of the input at each step of the recurrence. The figure shows the curve for 1000 scenes sampled from our StarCraft II minigame data (described in the following sections) and against random latents, using the encoder and decoder for that minigame. The ELBO, in non-loss form, lower bounds the model's log-likelihood of the data and in both cases decreases in ELBO loss gives an increase in instance likelihood, with the greatest impact at the first step. Repeating this procedure with 1000 randomly sampled latents also gives the same result. We note that while likelihoods from deep generative models may not be sufficiently calibrated for outlier detection [38], we are not attempting to estimate a distribution, but instead are looking to increase the likelihood—hence the plausibility—of the reconstructions during counterfactual generation. Indeed, recent work has found that deep network confidence assessments measure how familiar a model is with the features of a scene [6], matching our need to reconstruct using elements the model is more familiar with.

Counterfactual Methods

Figure 1 (right) illustrates our counterfactual generation architecture featuring the three methods we investigate in this work. The first draws a suitable example, the Nearest Unlike Neighbor (NUN), from previously observed examples. The second uses a traversal in the jointly trained latent space between the query and the NUN example. By stopping the traversal when the counterfactual criterion is met, this interpolated counterfactual is more proximal to the query (requires fewer feature edits). The third approach uses gradient information provided by the outcome predictors to perform a directed search in the latent space.

[2] ELBO loss drawn from our training setup, with a KL scaling term of $\beta = 10^{-5}$.

Nearest Unlike Neighbors

The Nearest Unlike Neighbor (NUN) is an example drawn from a library of observed instances that is similar to the query, but has an outcome that meets the counterfactual criterion [15]. It is a reliable way to obtain valid and plausible counterfactuals and serves as a baseline for comparison in our experiments. We draw the NUN $\langle \mathbf{x}_q, \mathbf{y}_{NUN} \rangle$ from the VAE's training instances, minimizing the observational distance while meeting the counterfactual criterion:

$$\mathbf{x}_{NUN} = \arg\min_{\mathbf{x}_c} \text{odiff}(x_q, x_c) \quad \text{s.t. } \kappa_i = 1.$$

Latent Interpolation

NUNs can ensure plausibility, but may not be sufficiently proximal to the query. Several studies have generated counterfactuals for tabular data by interpolating between the query and the NUN [15, 39]. As we are using a generative model, we can perform a similar interpolation in the latent space by interpolating linearly between the latent encodings of the query $\mathbf{z}_q$ and the NUN $\mathbf{z}_{NUN}$ to obtain the interpolated latent representation $\mathbf{z}_\iota$. The scaling factor α is sampled from 0 to 1, set to the first point where the counterfactual criterion is first satisfied, i.e., $\kappa_i(\sigma_i(\mathbf{z}_\iota)) = 1$. If $\alpha = 1$, we consider the interpolation to have failed to produce a valid counterfactual, as the result is the NUN. If a point was found, we then update $\mathbf{z}_\iota$ with a plausibility adjustment, with the magnitude λ selected by a grid search along the unit direction of the gradient for the point with the lowest anomaly score.

$$\mathbf{z}_\iota = \alpha(\mathbf{z}_{NUN} - \mathbf{z}_q) + (1 - \alpha)\mathbf{z}_q, \quad \alpha \in [0, 1]$$
$$\mathbf{z}_\iota = \mathbf{z}_\iota + \lambda \nabla_{\mathbf{z}} p(dec(\mathbf{z}_\iota))$$

Iterative Gradient Updates

Instead of relying on interpolating toward a concrete example, we can simply follow the gradient signal from the desired outcome predictor to shift it in the desired direction. We then apply a plausibility adjustment to shift the latent to a higher likelihood state:

$$\mathbf{z} = \mathbf{z} + s\lambda_1 \nabla_{\mathbf{z}} y^{(i)}$$
$$\mathbf{z} = \mathbf{z} + \lambda_2 \nabla_{\mathbf{z}} p(dec(\mathbf{z})).$$

where $s \in \{-1, 1\}$ is the desired sign of the change, with scaling terms λ_1, λ_2.[3] We iterate this update until the counterfactual criterion $\kappa_i(\sigma_i(z))$ is satisfied or a maximum number of steps is reached [4]. We note that the gradient update over a latent space trained only for reconstruction, without the plausibility adjustment, is equivalent XGEMS [13] and similar methods in the literature.

[3] Set to $\lambda_1 = 5, \lambda_2 = 1$, tuned over the training set.
[4] This was arbitrarily set to 1000 in our experiments.

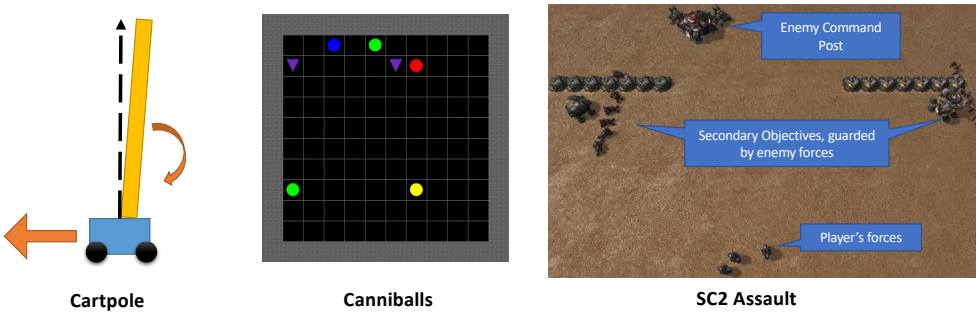

Fig. 3. Environments used in counterfactual generation experiments.

Experiment

We now describe the RL environments used, along with the model, training, and counterfactual query setup. We follow with results detailing counterfactual quality across three environments, and show the effect of the plausibility adjustment. We then show how the joint training helps improve counterfactual quality in comparison with a reconstruction only latent. Specific details of the model, training, errors, parameters, and code and data release are given in the Technical Appendix.

Environments

In order to assess generalizability of our counterfactual methods, we conducted experiments in three different reinforcement learning environments: Cartpole [3], Canniballs [33], and a custom minigame in the StarCraft II Learning Environment [36]. Figure 3 illustrates the environments.

Cartpole (left) is a two-dimensional physics simulation, where the agent has to balance a pole on a cart by moving left and right. Reward is given for each timestep the pole remains upright and balanced, with episodes ending when the pole falls over or the cart veers too far from its origin. Observed state consist of four continuous parameters: cart velocity and position, pole angle and angular velocity.

Canniballs (center) [33] is a gridworld game designed to exercise multiple subgoals in a highly stochastic environment. The player controls the red ball, and reward is earned for consuming weaker entities in the game, with a penalty applied for stalling or being consumed. Episodes end when the player is consumed or after a fixed number of steps. All game entities have a strength level, including the player, who can only consume entities weaker than itself. Strength is built up by consuming different entity types (colored balls and triangles), where balls have their own behavior, such as random movement, bouncing across the field, or chasing the player. Observations are in the form of a set of categorical spatial feature layers.

StarCraft II[5] (right) is a multiplayer real-time strategy game that features a variety of unit and building types. Each unit type has strengths and weaknesses, and part of

[5] https://StarCraft2.com

the strategy is to employ the best units to win the game. Buildings provide unique capabilities and can be destroyed or seized. For our experiments, we developed a custom scenario designed to exercise complex decision-making. The agent takes the part of one of the players, and is rewarded for destroying enemy forces, seizing secondary objectives, and destroying the enemy's command post. Capturing a secondary objective provides the player with reinforcements, which can be used to avoid obstacles. The observation space is spatial, but contains multiple layers containing both numerical and categorical data, and is significantly more complex than the two other environments. For the remainder of this paper, we will refer to this scenario as *SC2 Assault*.

Reinforcement Learning

Both Cartpole and Canniballs were trained using the RLLib framework [19]. For the SC2 Assault agent, we used a V-trace [7] agent trained using the Reaver toolkit [29]. This was implemented in the StarCraft II Learning Environment via the PySC2 interface [36], using a subset of the full action set that is focused on movement and attacks for each type of unit. Having trained the RL agents, we produced 1000 episodes for each environment using the trained policy. This resulted in $189{,}674$ frames for Cartpole, $136{,}671$ for Canniballs, and $213{,}407$ for SC2 Assault.

For the outcome variables used to form counterfactual queries, we based our approach on the concept of interestingness elements [32, 31], corresponding to numeric measures that allow highlighting meaningful and potentially explanatory situations as an RL agent interacts with its environment. Each measure is derived from data representing the agent's internal state, such as the value function estimate, V, the action value function Q (depending on the architecture), the action distribution, and others. For these experiments, we used the following interestingness variables as outcomes for exploring counterfactuals:

- **Value**: The value function estimate, measuring the expected discounted cumulative reward at any given state.
- **Confidence**: The action execution certainty of the agent, where we use a measure of statistical dispersion that relies on the entropy of the policy's action distribution.
- **Riskiness**: The margin between highest- and lowest-valued outcomes from taking an action, representing the perceived tolerance for mistakes in the environment.

Model

We now describe the VAE used to construct the surrogate model from agent trajectories. For the Cartpole agent, we used MLP encoders and decoders over the vector. Canniballs and SC2 Assault use spatial features, for which we used a convolutional architecture encoder and decoder. The VAE itself differs from the standard hierarchical model by having all convolutional layers feed into the latent, and using a linear transform after the latent prior to the decoding.

For each environment, we used 95% of the recorded trajectories for training and the remainder for testing. Test mean-squared error for normalized predictions was under 0.1 across the full range of $[-1, 1]$.

We now detail the three major experiments that form the core of our contributions. The first compares gradient-driven counterfactuals across several RL environments. The second examines how plausibility adjustments can improve the likelihood of a generated example and reduce the number of anomalous counterfactuals. Finally, we demonstrate the effectiveness of the jointly training latent space on the quality of counterfactuals. Equivalence to baselines from literature are marked when appropriate.

Counterfactual Query Setup

For each interestingness variable i and sign of change s, we sampled 100 individual instances from the set of recorded trajectories. Each instance $\langle \mathbf{x}_q, \mathbf{y}_q \rangle$ was filtered so there is sufficient margin in variable i for a valid counterfactual, e.g., $-1 \leq \mathbf{y}_q^{(i)} + s\epsilon \leq 1$. For the value function, ϵ was two times the standard deviation. The other variables are in the range $[-1, 1]$, and we set $\epsilon = 0.5$. From our inventory of three interestingness variables and two signs of change (increasing or decreasing their value), we experimented with a total of six combinations (600 queries) for each counterfactual generation method and environment.

Table 1. Counterfactual methods for each environment, with microaveraged statistics across counterfactual quality measures. † indicates significant improvement in observational difference against the NUN baseline. * indicates significant improvement in observational difference and anomaly scores against the XGEMS baseline. Best values, including ties, in each domain are bolded.

Method	Obs Diff	Anom Score	Valid CFs
Cartpole			
NUN	1.28 ± 0.71	0.11 ± 0.03	$\mathbf{1.00}$
InterpPt	$\mathbf{0.86 \pm 0.82}^{\dagger}$	$\mathbf{0.07 \pm 0.02}$	0.50
Grad	$0.99 \pm 0.89^{\dagger}$	0.31 ± 0.54	0.98
Canniballs			
NUN	1754.91 ± 89.31	$\mathbf{0.00 \pm 0.00}$	0.67
InterpPt	$\mathbf{5.46 \pm 4.10}^{\dagger}$	0.20 ± 0.57	0.67
Grad	$18.94 \pm 20.88^{\dagger}$	12.56 ± 17.13	$\mathbf{0.99}$
SC2 Assault			
NUN	1746.12 ± 573.25	$\mathbf{7.52 \pm 3.88}$	$\mathbf{1.00}$
InterpPt	$1234.94 \pm 880.92^{\dagger}$	30.38 ± 24.87	0.67
*Grad**	$\mathbf{83.36 \pm 141.25}^{\dagger}$	33.73 ± 45.29	0.97
InterpPt -Pls	1224.06 ± 879.17	48.26 ± 75.52	0.69
*Grad -Pls**	$\mathbf{82.44 \pm 136.11}$	34.41 ± 46.62	0.97
SC2 Assault, Reconstruction-only Latent			
InterpPt	881.94 ± 770.84	72.00 ± 70.78	0.44
Grad	124.17 ± 100.85	83.07 ± 44.34	0.81
xGEMs	123.34 ± 97.66	83.12 ± 44.95	0.81

Results

We compared the following methods across the three RL environments:

- Drawing the Nearest Unlike Neighbor from the training set (NUN), which is used as a baseline.
- Latent Interpolation to the NUN, stopping at the first point where κ_i is met (InterpPt).
- Using Iterative Gradient Update to perturb the latent until κ_i is met (Grad).

Table 1 reports the micro-averaged mean and standard deviation of the observational differences, anomaly scores[6], and fraction of valid counterfactual queries for each method against the given query combinations for the three environments. Only values from valid counterfactuals were used to compute these measures. Significance tests are conducted using a two sample t-test with $\alpha = 0.01$. As expected, drawing from a memory of actual instances (NUN) produces the least anomalous and most plausible counterfactuals. However, both latent-based approaches produce counterfactuals that with significantly lower observational differences (more proximal) across all three domains. An in-depth analysis of proximity across all three domains is given in the Technical Appendix.

The Gradient method produced valid counterfactuals for most queries, missing at most 3% overall. The InterpPt method generated counterfactuals about 67% of the time across all three environments, with the remainder requiring full traversal to the NUN. We examined the impact of a plausible scene gradient adjustment on counterfactuals for the SC2 Assault minigame environment, showing the Latent Interpolation and Gradient methods without the plausibility adjustment (InterpPt -Pls, Grad -Pls). We find the plausibility adjustment significantly reduces the anomaly score without impacting the observational difference for InterpPt. However, we find no significant difference in anomaly scores with Gradient.

To relate anomaly scores to a concrete number of anomalous scenes, we tuned a threshold on the VAE reconstruction scores to detect labeled anomalous SC2 Assault scenes, achieving a test accuracy of 95% over a baseline guess of 66% (plausible)[7]. Out of 600 queries, InterpPt produced 4 anomalous counterfactuals, compared to 20 without the plausibility adjustment. The Gradient method produced 46 with the adjustment, and 48 without.

Impact of Joint Training

We tested our hypothesis that joint training of input reconstruction and outcome prediction leads to better counterfactuals, as approaches in the literature trained these two tasks sequentially (see Sec. 2). Using the SC2 Assault task, we trained the VAE model with just the reconstruction objective using an otherwise similar setup. We then trained the outcome predictors given the latents produced by the reconstruction-only model,

[6] Observational difference and anomaly score are the inverses of proximity and plausibility, so lower scores indicate better performance.

[7] Model and training are detailed in the Technical Appendix

achieving prediction errors comparable to those of the full model. We then re-ran the same set of experiments using the reconstruction-only latent and predictors. Results are presented at the bottom half of Table 1. Here we see that counterfactuals generated from the reconstruction-only latent space produced considerably more anomalous counterfactuals, with fewer valid counterfactuals. We note that the Gradient approach without the plausibility adjustment over the Reconstruction-Only latent is equivalent to XGEMS [13]. In comparison, Gradient derived counterfactuals over the joint space have significantly lower observational differences and anomaly scores, with and without the plausibility adjustment.

In the joint latent, the Interpt and Gradient methods produced a combined total of 50 concrete anomalies, whereas their equivalents from the reconstruction-only latent gave a total of 176 anomalies. XGEMS itself gave 107 anomalies, in contrast to its equivalent in the joint latent, which had 48.

Counterfactual Analysis

We now present an analysis of counterfactuals drawn from the SC2 Assault minigame task. We first present an overview of how the minigame scenes are structured for the agent.

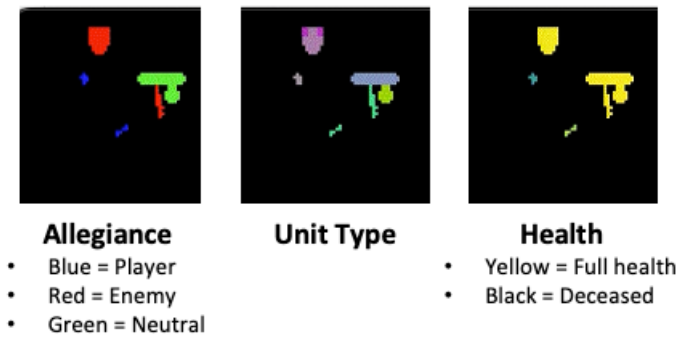

Fig. 4. The SC2 Assault task uses multiple spatial frames to represent different attributes of the units and structures in the game. The leftmost grid represents the allegiance of the unit. The middle encodes the type of unit or structure, while the rightmost describes the health status of that unit.

Figure 4 describes the observation format. Each scene is described by three spatial *semantic frames* representing different semantic information. The *Allegiance* of the unit determines which faction the unit belongs to. *Unit Type* details the specific type of unit or building at that location. For simplicity, the following analyses will highlight significant unit types directly. Finally, *Health* shows the relative health of the unit using a scale of bright yellow representing full health to black representing no health.

For these analyses, we examine counterfactuals where the value function estimate is either increased by two standard deviations from a sampled low-value scene, or from

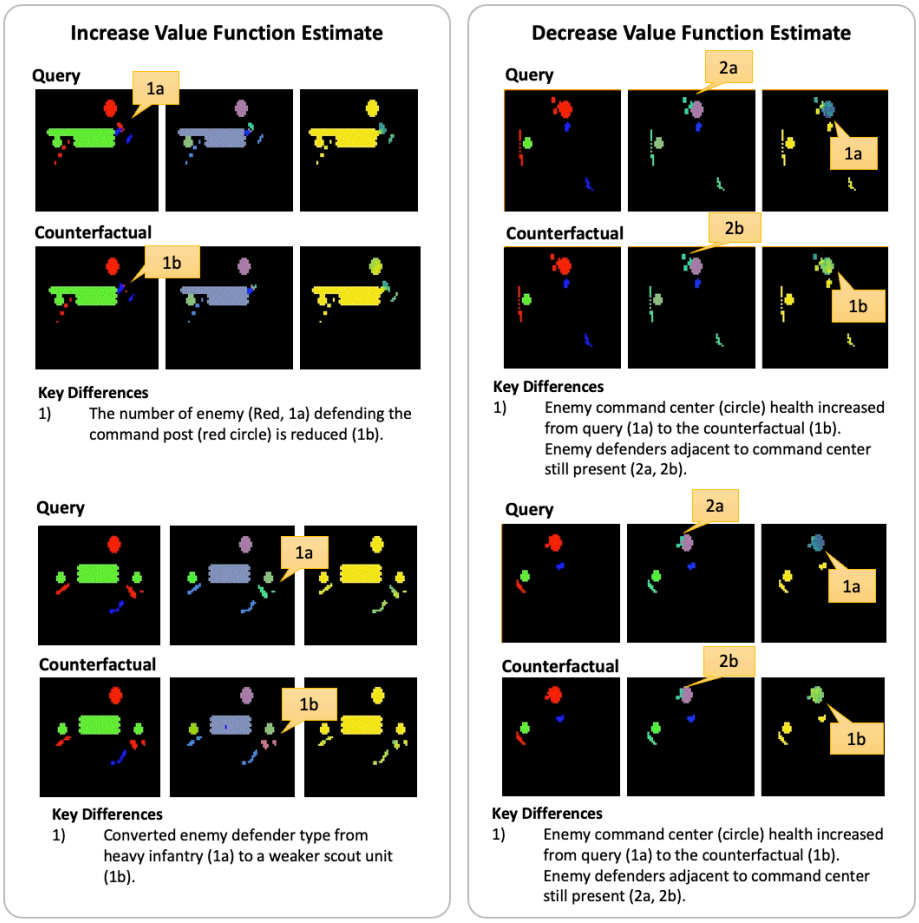

Fig. 5. Four example counterfactuals generated using gradient-based walks in the latent space. The left column shows low value function estimate query scenes with corresponding higher valued counterfactuals, the right with high value function estimates and lower valued counterfactuals. For each example, the top row shows the semantic frames (allegiance, unit type, health) for the query scene, the middle row shows its higher valued counterfactual, and the bottom row shows the highlighted key differences. Key differences are derived by analyzing the differences in each spatially significant semantic frame.

decreased by the same amount from a high-value one. Because the observation space is composed of semantic frames, with each corresponding to a concept such as unit type or allegiance, the differences between a query scene and its counterfactual directly map to understandable and meaningful changes. This can be used to identify types of changes associated with counterfactuals along an outcome variable; for example, converting enemy defenders into weaker units results in increased value-function estimates. While this is specific to the SC2 Assault task, similar analyses may be extended to other domains via pre-trained detectors.

Figure 5 shows four sampled query scenes and their counterfactuals generated by gradient perturbation with the plausibility adjustment in place. The left column for low value queries to high value counterfactuals, with the opposite on the right. We find the generated counterfactuals are both proximal to the query and with minimal artifacts. Deltas in the semantic frames between the queries and counterfactuals show numerous explanations for factors that can increase the value-estimate. For example, reducing the number of enemy defenders (top left) or attacking when the enemy defenders are of a weaker type (bottom left) increase the odds of success. On the other hand, situations where the enemy command center is at full health, but still has a full complement of defenders reduces the value function estimate, or decreases the agent's perceived ability to destroy the enemy command post.

Figure 6 illustrates the impact of the plausibility adjustment. For a low-valued starting query, the gradient-based counterfactual with the plausibility adjustment displays less noise, and a lower anomaly score, than one run to the same value.

The sequence of feature changes in the scene between the query and a counterfactual also provides information about which features bolster an outcome in the query. Figure 7 gives an example a low value-function query and a high valued NUN drawn from the library. Four samples are drawn from the latent trajectory from the query (left) to the NUN (right). The top row shows the scene reconstruction, restricted to the unit allegiance frame for clarity. The bottom shows the value function estimate (blue) and the anomaly score for a reconstruction at that point in the latent traversal (red). The differences between scene features between each of the sampled points illustrates the change in the value function resulting from those feature changes. In addition, the schedule of feature edits can identify correlated features and their impact on the value function.

Discussion and Future Work

We presented a latent space that jointly contains information about observations and outcome variables that permits unconditioned sampling. From this, we presented two methods for obtaining counterfactuals: the first employing interpolations between the query and case-based instances drawn from a memory, the second using gradient updates to iteratively update the query to reflect the desired outcomes. We show that the joint latent approach can produce more valid counterfactuals that are also more plausible. We also show that reconstruction error as a proxy for the likelihood gradient can help improve the plausibility of counterfactuals in certain cases. We followed with an assessment of a sampling of generated counterfactuals, demonstrating the ability of the method create meaningful and plausible examples.

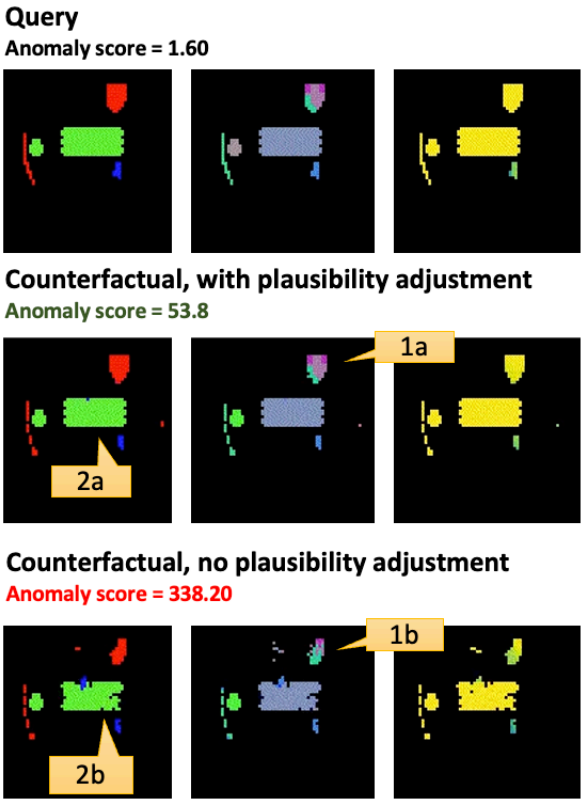

Fig. 6. The impact of plausibility adjustments: The top row shows the query scene, along with its computed anomaly score. The middle row shows a gradient-based counterfactual for increasing the value, with adjustments in place, the bottom rows shows the equivalent-valued counterfactual without the adjustments. These adjustments preserve the spatial structure of the command structure and defending units (1a vs. 1b) as well as the center set of obstacles and friendly Blue unit placement (2a vs. 2b).

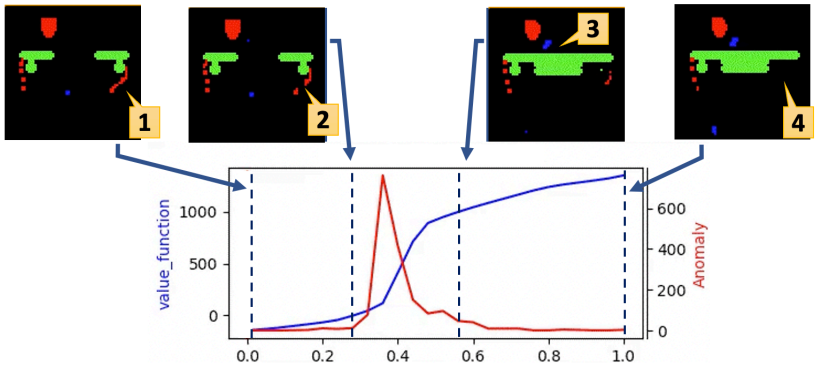

Fig. 7. Traversal in the latent space between the a low-value query (1) and high-value nearest unlike neighbor counterfactual (4). The top row shows the reconstructed scene from the latent, restricted to unit Allegiance for clarity, while the bottom shows the value estimate and the anomaly score for that point in the traversal. The traversal provides a sensitivity analysis showing both the value of the feature and its importance in the original query

Future areas of investigation include a closer examination of these methods in contrast to feature-level adversarial methods. A major concern about that class of methods is that the counterfactuals they generate may be imperceptible to humans, as their perturbations are fuzzing attacks that minimize feature-level changes. While latent space traversal methods can take steps to ensure a minimal amount of feature-level differences, future work should include stronger assurances for preventing the generation of counterfactuals that are imperceptibly different from the query. Our approach allows gradient-based adjustments to have a latent meet different criteria, such as improving data-likelihood. This can be tailored to include adjustments reflecting feature edits corresponding to actionable elements the agent or human operator has control over, such as the disposition of friendly forces in the SC2 Assault environment.

Another area of possible improvement would be the use of classifier-free guidance to improve the quality of the counterfactuals, as gradient-based signals from discriminative classifiers may not be sufficient to capture the shape of certain outcome variable distributions [10]. Use of this method in conditional latent diffusion was also found to improve generated imagery [30].

Finally, we note that this work, like many others, looks at intrinsic measures of counterfactual quality. Proper extrinsic evaluations of how counterfactuals can improve a meaningful task remains to be addressed. One possible avenue would be to use counterfactuals to improve the examples used for machine-teaching and tutoring applications [34]. We are also investigating the use of directed counterfactuals to warn decision-makers of likely or dangerous possible scenarios. In addition, they may also act as a source of additional weak evidence for observational assessments of the causal link between features and outcomes.

Acknowledgements

This material is based upon work supported by the Defense Advanced Research Projects Agency (DARPA) under Contract No. HR001119C0112. Any opinions, findings and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the DARPA.

References

1. Agarap, A.F.: Deep learning using rectified linear units (relu). arXiv preprint arXiv:1803.08375 (2018)
2. Bowman, S.R., Vilnis, L., Vinyals, O., Dai, A.M., Józefowicz, R., Bengio, S.: Generating sentences from a continuous space. CoRR **abs/1511.06349** (2015), http://arxiv.org/abs/1511.06349
3. Brockman, G., Cheung, V., Pettersson, L., Schneider, J., Schulman, J., Tang, J., Zaremba, W.: Openai gym. arXiv preprint arXiv:1606.01540 (2016)
4. Byrne, R.M.J.: Counterfactuals in explainable artificial intelligence (xai): Evidence from human reasoning. In: Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI-19. pp. 6276–6282. International Joint Conferences on Artificial Intelligence Organization (7 2019). https://doi.org/10.24963/ijcai.2019/876, https://doi.org/10.24963/ijcai.2019/876
5. Dathathri, S., Madotto, A., Lan, J., Hung, J., Frank, E., Molino, P., Yosinski, J., Liu, R.: Plug and play language models: A simple approach to controlled text generation. CoRR **abs/1912.02164** (2019), http://arxiv.org/abs/1912.02164
6. Dietterich, T.G., Guyer, A.: The familiarity hypothesis: Explaining the behavior of deep open set methods (2022). https://doi.org/10.48550/ARXIV.2203.02486, https://arxiv.org/abs/2203.02486
7. Espeholt, L., Soyer, H., Munos, R., Simonyan, K., Mnih, V., Ward, T., Doron, Y., Firoiu, V., Harley, T., Dunning, I., Legg, S., Kavukcuoglu, K.: IMPALA: scalable distributed deep-rl with importance weighted actor-learner architectures. CoRR **abs/1802.01561** (2018), http://arxiv.org/abs/1802.01561
8. Heusel, M., Ramsauer, H., Unterthiner, T., Nessler, B., Klambauer, G., Hochreiter, S.: Gans trained by a two time-scale update rule converge to a nash equilibrium. CoRR **abs/1706.08500** (2017), http://arxiv.org/abs/1706.08500
9. Higgins, I., Matthey, L., Pal, A., Burgess, C., Glorot, X., Botvinick, M., Mohamed, S., Lerchner, A.: beta-vae: Learning basic visual concepts with a constrained variational framework (2016)
10. Ho, J., Salimans, T.: Classifier-free diffusion guidance (2022)
11. Ioffe, S., Szegedy, C.: Batch normalization: Accelerating deep network training by reducing internal covariate shift. In: Bach, F., Blei, D. (eds.) Proceedings of the 32nd International Conference on Machine Learning. Proceedings of Machine Learning Research, vol. 37, pp. 448–456. PMLR, Lille, France (07–09 Jul 2015), https://proceedings.mlr.press/v37/ioffe15.html
12. Jeanneret, G., Simon, L., Jurie, F.: Diffusion models for counterfactual explanations. In: Asian Conference on Computer Vision (2022)
13. Joshi, S., Koyejo, O., Kim, B., Ghosh, J.: xgems: Generating examplars to explain black-box models. CoRR **abs/1806.08867** (2018), http://arxiv.org/abs/1806.08867
14. Karras, T., Laine, S., Aittala, M., Hellsten, J., Lehtinen, J., Aila, T.: Analyzing and improving the image quality of stylegan. CoRR **abs/1912.04958** (2019), http://arxiv.org/abs/1912.04958
15. Keane, M.T., Smyth, B.: Good counterfactuals and where to find them: A case-based technique for generating counterfactuals for explainable AI (XAI). CoRR **abs/2005.13997** (2020), https://arxiv.org/abs/2005.13997
16. Khorram, S., Fuxin, L.: Cycle-consistent counterfactuals by latent transformations. 2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) pp. 10193–10202 (2022)

17. Kingma, D.P., Welling, M.: Auto-encoding variational bayes (2013). https://doi.org/10.48550/ARXIV.1312.6114, https://arxiv.org/abs/1312.6114
18. Klys, J., Snell, J., Zemel, R.: Learning latent subspaces in variational autoencoders. Advances in neural information processing systems **31** (2018)
19. Liang, E., Liaw, R., Nishihara, R., Moritz, P., Fox, R., Gonzalez, J., Goldberg, K., Stoica, I.: Ray rllib: A composable and scalable reinforcement learning library. CoRR **abs/1712.09381** (2017), http://arxiv.org/abs/1712.09381
20. Lundberg, S.M., Lee, S.: A unified approach to interpreting model predictions. CoRR **abs/1705.07874** (2017), http://arxiv.org/abs/1705.07874
21. Miller, T.: Explanation in artificial intelligence: Insights from the social sciences. CoRR **abs/1706.07269** (2017), http://arxiv.org/abs/1706.07269
22. Moore, J., Hammerla, N., Watkins, C.: Explaining deep learning models with constrained adversarial examples. CoRR **abs/1906.10671** (2019), http://arxiv.org/abs/1906.10671
23. Mothilal, R.K., Sharma, A., Tan, C.: Explaining machine learning classifiers through diverse counterfactual explanations. CoRR **abs/1905.07697** (2019), http://arxiv.org/abs/1905.07697
24. Nalisnick, E., Matsukawa, A., Teh, Y.W., Gorur, D., Lakshminarayanan, B.: Do deep generative models know what they don't know? (2018). https://doi.org/10.48550/ARXIV.1810.09136, https://arxiv.org/abs/1810.09136
25. Nguyen, A., Yosinski, J., Bengio, Y., Dosovitskiy, A., Clune, J.: Plug & play generative networks: Conditional iterative generation of images in latent space. CoRR **abs/1612.00005** (2016), http://arxiv.org/abs/1612.00005
26. Olson, M.L., Khanna, R., Neal, L., Li, F., Wong, W.: Counterfactual state explanations for reinforcement learning agents via generative deep learning. CoRR **abs/2101.12446** (2021), https://arxiv.org/abs/2101.12446
27. Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., Desmaison, A., Kopf, A., Yang, E., DeVito, Z., Raison, M., Tejani, A., Chilamkurthy, S., Steiner, B., Fang, L., Bai, J., Chintala, S.: Pytorch: An imperative style, high-performance deep learning library. In: Wallach, H., Larochelle, H., Beygelzimer, A., d'Alché-Buc, F., Fox, E., Garnett, R. (eds.) Advances in Neural Information Processing Systems 32, pp. 8024–8035. Curran Associates, Inc. (2019), http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf
28. Pomerleau, D.A.: Input reconstruction reliability estimation. In: Hanson, S., Cowan, J., Giles, C. (eds.) Advances in Neural Information Processing Systems. vol. 5. Morgan-Kaufmann (1992), https://proceedings.neurips.cc/paper/1992/file/b7bb35b9c6ca2aee2df08cf09d7016c2-Paper.pdf
29. Ring, R.: Reaver: Modular deep reinforcement learning framework. https://github.com/inoryy/reaver (2018)
30. Rombach, R., Blattmann, A., Lorenz, D., Esser, P., Ommer, B.: High-resolution image synthesis with latent diffusion models. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2022), https://github.com/CompVis/latent-diffusionhttps://arxiv.org/abs/2112.10752
31. Sequeira, P., Gervasio, M.: Interestingness elements for explainable reinforcement learning: Understanding agents' capabilities and limitations. Artificial Intelligence **288**, 103367 (2020). https://doi.org/https://doi.org/10.1016/j.artint.2020.103367
32. Sequeira, P., Yeh, E., Gervasio, M.: Interestingness Elements for Explainable Reinforcement Learning through Introspection. In: Joint Proceedings of the ACM IUI 2019 Workshops. p. 7. ACM (Mar 2019)
33. Showalter, S.: Cameleon canniballs environment. https://github.com/SRI-AIC/cameleon#environments (2021)

34. Simard, P.Y., Amershi, S., Chickering, D.M., Pelton, A.E., Ghorashi, S., Meek, C., Ramos, G.A., Suh, J., Verwey, J., Wang, M., Wernsing, J.: Machine teaching: A new paradigm for building machine learning systems. CoRR **abs/1707.06742** (2017), http://arxiv.org/abs/1707.06742
35. Szegedy, C., Inc, G., Zaremba, W., Sutskever, I., Inc, G., Bruna, J., Erhan, D., Inc, G., Goodfellow, I., Fergus, R.: Intriguing properties of neural networks. In: In ICLR (2014)
36. Vinyals, O., Ewalds, T., Bartunov, S., Georgiev, P., Vezhnevets, A.S., Yeo, M., Makhzani, A., Küttler, H., Agapiou, J.P., Schrittwieser, J., Quan, J., Gaffney, S., Petersen, S., Simonyan, K., Schaul, T., van Hasselt, H., Silver, D., Lillicrap, T.P., Calderone, K., Keet, P., Brunasso, A., Lawrence, D., Ekermo, A., Repp, J., Tsing, R.: Starcraft II: A new challenge for reinforcement learning. CoRR **abs/1708.04782** (2017), http://arxiv.org/abs/1708.04782
37. Wachter, S., Mittelstadt, B.D., Russell, C.: Counterfactual explanations without opening the black box: Automated decisions and the GDPR. CoRR **abs/1711.00399** (2017), http://arxiv.org/abs/1711.00399
38. Wang, Z., Dai, B., Wipf, D., Zhu, J.: Further analysis of outlier detection with deep generative models (2020). https://doi.org/10.48550/ARXIV.2010.13064, https://arxiv.org/abs/2010.13064
39. Wiratunga, N., Wijekoon, A., Nkisi-Orji, I., Martin, K., Palihawadana, C., Corsar, D.: Discern: Discovering counterfactual explanations using relevance features from neighbourhoods. CoRR **abs/2109.05800** (2021), https://arxiv.org/abs/2109.05800
40. Yang, F., Liu, N., Du, M., Hu, X.: Generative counterfactuals for neural networks via attribute-informed perturbation. CoRR **abs/2101.06930** (2021), https://arxiv.org/abs/2101.06930

Technical Supplement

Model

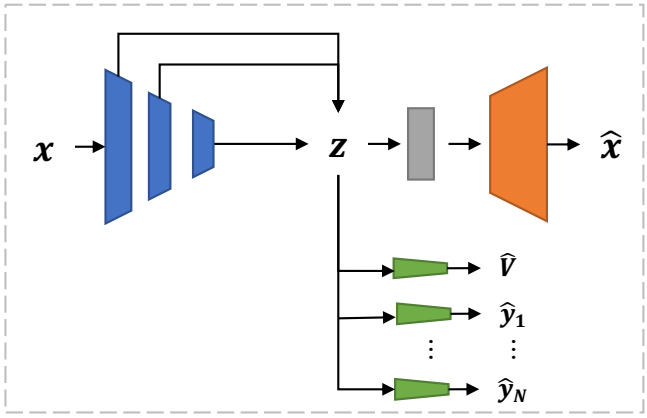

Fig. 8. The modified Variational Autoencoder used in our work. Changes include a multi-level latent incorporating information from each convolution layer, additional prediction targets off the latent z, and an additional affine translation layer prior to decoding.

Figure 8 illustrates the modified Variational Autoencoder (VAE) [17] used to create the joint observation and outcome latent. The encoder encodes input observations x to a latent z. To improve reconstruction quality, the latent is passed through an affine transformation to interpret the latent prior to decoding, as proposed in StyleGAN 2 [14]. This finally passed to a decoder, which provides the reconstruction $\hat{x}$.

The architecture of the encoder and decoders are matched with the nature of the observations in specific domain. For Cartpole [3], encoders were composed of three sequential blocks, each consisting a linear layer followed by a batch normalization [11] with a rectified linear activation [1]. Cartpole's feature space is a four dimensional vector, consisting of the cart position and velocity, and pole angle and rotation rate.

Both Canniballs [33] and the Starcraft 2 Learning Environment [36] SC2 Assault minigame used 2D convolutions instead of linear layers. Canniballs' observation space consisted of four 12×12 spatial arrays, representing player, enemy, food, and obstacle locations. The SC2 Assault observation space consisted of three 64×64 spatial feature arrays, representing allegiance of each unit, their types, and their health. Models for these environments used a sequence of four blocks consisting of a 2D convolution, batch norm, and rectified linear activation. Convolutions employed a kernel size of 4, with Canniballs using a stride of 1 while SC2 Assault used 2. Decoders for all three domains mirrored the encoding arrangement, with transpose equivalents for 2D convolutions. The dimensionality of the latent z was 256 for all three environments, and the affine post-latent transform was implemented as a linear layer size 256.

Following the dense connectivity pattern in [**?**], the latent z is composed of information from each convolution layer in the encoder. The intent is to capture lower level information in the latent.

For predicting the Interestingness outcome variables, we used a sequence of three linear layers (sizes 128, 64, 32) with rectified linear activations, with a final single node layer for regressing the standardized target variable value, with mean and standard-deviations estimated from the training data.

For the reconstruction-only latent model of the SC2 Assault environment, the standalone predictors for the Interestingness Variables used the same architecture as one used for the joint training.

Training

All of our models and code were implemented in PyTorch [27] and trained on a NVIDIA GeForce RTX 3090. Training was conducted to 1.1×10^9 steps. The variational autoencoder was also tuned with a fixed beta schedule going from $\beta = 0$ to 10^{-5} [2], allowing the encoders to learn a feature set prior to increasing the weight on the KL term for the priors. Optimization was conducted using Adam with a learning rate of 1×10^{-3}.

To support a wide range of domains, our VAE supports inputs from multiple encoders and decoders. This is particularly important for StarCraft II [36] representations, which consist of multiple spatial semantic maps. Each encoder can either be MLPs or convolutional, depending on the part of the input it captures. The outputs of the encoders are concatenated and passed to a MLP for inferring the mean and standard deviations. To improve the ability to reconstruct smaller details, convolutional encoders assemble a combined latent by constructing individual latents from each encoder layer.

For the Interestingness Variables, standardized outcome variable values were targeted during training, using a mean-squared error loss for training the outcome variable predictors. Reconstruction losses were dependent upon the environment and the observational feature types. Cartpole observations consisted of four standardized real valued variables, and were targeted using mean-square error. Canniballs' four spatial arrays were binary-valued, and used binary cross entropy losses for each layer. Finally, SC2 Assault consisted of two categorical spatial arrays, using cross entropy loss, and one real-valued using mean-squared error with values normalized to the 0-1 range.

Table 2 shows respective losses at the end of training smoothed via exponential moving average. The last column shows the losses for training the reconstruction-only latent, for the SC2 Assault environment.

Loss	**Cartpole**	**Canniballs**	**SC2 Assault**	**SC2 Recon-only**
Recon	3.6×10^{-6}	8.2×10^{-5}	3.1×10^{-3}	3.3×10^{-3}
	Interestingness Variables			
Value	3.6×10^{-5}	1.4×10^{-3}	1.0×10^{-3}	2.6×10^{-3}
Confidence	7.9×10^{-6}	8.1×10^{-6}	6.3×10^{-4}	1.6×10^{-3}
Riskiness	1.9×10^{-4}	1.4×10^{-9}	7.6×10^{-6}	1.4×10^{-5}

Table 2. Validation losses, smoothed using exponential moving average, by environment. Last column lists losses for the reconstruction-only model trained on the SC2 Assault environment. All Interestingness Variable losses are reported in mean-squared error against standardized values. Reconstruction (recon) scores are summed over the losses for each feature layer comprising each environment, with categorical losses represented by cross entropy and regression losses by mean-square error against normalized values.

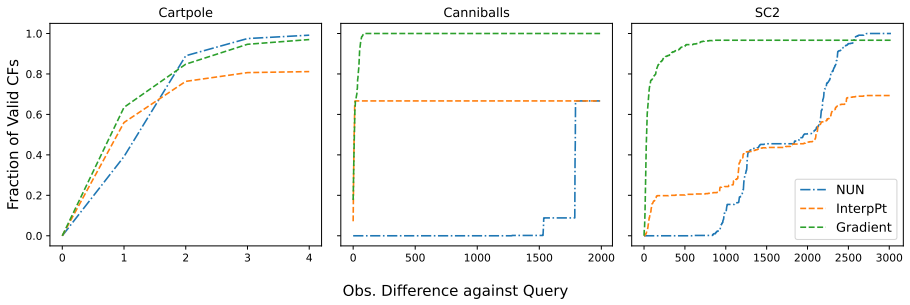

Fig. 9. Number of valid counterfactuals (CFs, y-axis) at or below the observational difference to the query (x-axis) for each environment. In all environments, the latent-based approaches (Interpolation and Gradient) were able to produce counterfactuals not present in the memory (NUN) and that are more proximal to the query.

Counterfactual Proximity Analysis and Unseen Scenes

In certain use cases, there is an upper bound on the distance between the query and a valid counterfactual before they are deemed unrelatable [15]. To assess this, we examined the proximity of valid counterfactuals to the query for each method. Figure 9 shows the cumulative fraction of valid counterfactuals whose observational differences from the query are at or below the given level, where "valid" is binary a binary pass/fail metric. We find that the latent space methods (InterpPt and Gradient) were able to produce more valid counterfactuals compared to retrieving the closest counterfactual from memory (NUN). These are also unseen scenes, as they would otherwise have been selected as counterfactuals from memory. This effect is more pronounced in the Canniballs and SC2 Assault environments, as their observation spaces are significantly more complex than Cartpole's.

Anomalous Scene Detector

To arrive at a quantitative estimate of the number of actual anomalous scenes in the SC2 Assault minigame, we took 1392 pairs of real instances and sampled the reconstruction between them, giving us 4176 scenes. We then labeled these as anomalous or not, based on mistakes such as partially reconstructed units or duplicates of unique structures. Earlier experimentation found anomalous scenes derived from randomly sampled latents were easily distinguishable, while those from latent interpolations between latents for real scenes proved extremely difficult to classify. As a result, we focused on separating plausible versus anomalous scenes derived from interpolated latents. We scored all of the instances using the VAE based anomaly score, which computed the observational difference between the latent's reconstruction and its re-encoded reconstruction. The scores for 2196 randomly selected training scenes were used to tune a threshold maximizing anomalous scene detection accuracy, giving a training accuracy of 96% and a test accuracy of 95% on the remaining 1980 test instances, over a baseline guess (plausible) of 66%.

Code Release

The necessary code and data files will be made open-source and will be available at from our Github page, https://github.com/imago.