Counterfactual Explanation and Causal Inference In Service of Robustness in Robot Control

Simón C. Smith and Subramanian Ramamoorthy
Institute of Perception, Action and Behaviour
School of Informatics
The University of Edinburgh
Edinburgh, UK
{artificialsimon, s.ramamoorthy}@ed.ac.uk

***Abstract*—We propose an architecture for training generative models of counterfactual conditionals of the form, *'can we modify event A to cause B instead of C?'*, motivated by applications in robot control. Using an 'adversarial training' paradigm, an image-based deep neural network model is trained to produce small and realistic modifications to an original image in order to cause user-defined effects. These modifications can be used in the design process of image-based robust control - to determine the ability of the controller to return to a working regime by modifications in the input space, rather than by adaptation. In contrast to conventional control design approaches, where robustness is quantified in terms of the ability to reject noise, we explore the space of counterfactuals that might cause a certain requirement to be violated, thus proposing an alternative model that might be more expressive in certain robotics applications. So, we propose the generation of counterfactuals as an approach to explanation of black-box models and the envisioning of potential movement paths in autonomous robotic control. Firstly, we demonstrate this approach in a set of classification tasks, using the well known MNIST and CelebFaces Attributes datasets. Then, addressing multi-dimensional regression, we demonstrate our approach in a reaching task with a physical robot, and in a navigation task with a robot in a digital twin simulation.**

***Index Terms*—counterfactual conditionals, causal inference, model explainability, state envisioning, controller robustness**

I. Introduction

A robust autonomous system is one that can cope with perturbations and uncertainty in the state space [1]. These systems are designed to maintain a specified level of performance despite disturbances in the input, or perturbations in parameters, typically assumed bounded by a well-defined set. In realistic high-dimensional robotic systems, determining these limits tends to be highly non-trivial. In robotic systems that make use of machine learning methods such as deep neural networks (DNN), it is difficult to establish properties of behaviour outside the support of the distribution of the training data [2]. Moreover, even minimal carefully crafted disturbances (keeping the input close in statistical terms to the original data distribution), known as adversarial attacks, can already produce undesirable behaviour in DNNs [3]. In applications involving persistently autonomous and unmanned vehicles, difficult configurations of the robot-environment may only appear after several hundreds or thousands of hours of normal functioning. So, training such autonomous systems to be robust calls for alternate approaches to characterising and utilising the bounds of normal operation.

In the present work, we propose a generative model (called *cGen*) that produces *minimal* and *realistic* counterfactual 'disturbances' to a known input. The cGen counterfactual generator is used as a means to characterise robot control robustness. We do this via the level of modification to the environment that causes the controller to cease to be operational with respect to some requirement. In a sense, this model is used to compute the equivalent of stability boundaries for dynamical systems, but focussed here on issues arising from the perception-action loops. By identifying such counterfactual modifications to a specific environment configuration where the controller fails to achieve the required goal, we are in a position to add to the training set additional configurations *hallucinated* by the generative model. We use the term hallucination as a way to describe the generation of data points that are close to the original distribution of the training data but are not explicitly present in them [4], [5]. One use of such counterfactuals is as a way to explain classification decisions by DNNs. In this case, the counterfactuals make evident the required modification (disturbance) to an image to be classified in a user-defined category, differently from the original one. In contrast to work on adversarial attacks that focus attention on misclassification with imperceptibly small disturbances, our counterfactuals include a target class and close-to-the-original data distribution modifications.

The training regime of cGen is based on the Generative Adversarial Network (GAN) approach [6]. For image classification, we train a generative model that takes input from the training data and modify it to be classified as a user-defined class. The generative model has two main objectives. Firstly, to modify the original input to fool the classifier (discriminator) to classify it as belonging to the target class.

This work was supported by EPSRC funding for the ORCA Hub (EP/R026173/1, 2017-2021).

Secondly, the modification has to be as small a possible, i.e. the distance from the original and the modified image has to be minimal [7]. The loss function of the generator is a linear combination of these two features. Following the usual GAN training pattern, we train the classifier to discriminate between unmodified target class images and the ones modified by the model. When addressing robot control regression problems, we include an extra module - the predictor (controller) of the system. The predictor takes the modified image (counterfactual) as input and produces real-valued output that can represent motor commands, plans, end-effector positions, etc. In this architecture for regression, the generator also includes the loss of the target controller in its loss function. The classifier now behaves closely to the role of the discriminator in a regular GAN, by discriminating between modified and non-modified images. Note that the predictor is fixed during all the steps of the cGen training. In this stage, we are focused on finding the optimal counterfactuals rather than improving the controller.

In causal inference, counterfactuals are related to the idea that "if event A had not occurred, event B would not have occurred". Counterfactuals define a causal dependency between them and the outcome of applying it (or not) to the actual event. This is a more powerful definition than the correlation between cause and effect. Causal dependencies allow us to be certain that the counterfactual induces the results rather than mere co-occurrence. These characteristics make counterfactuals a better candidate to assess the robustness of a system compared to other utility functions like prediction accuracy or performance.

To evaluate this proposed approach, we study three cases. First, we use cGen in image classification tasks. In this case, the counterfactuals *explain* the difference between the two classes. The induced disturbance makes explicit the features present in the image that the classifier uses to predict a specific category. In the second case, we study the causal *modification* needed to solve a robot control problem. Two experiments are carried out, one on a physical PR2 robot with an end-effector planning problem, and the other a Model Predictive Control (MPC) problem involving a simulated Husky robot. In the third case, we study our proposed counterfactual robustness measure between MPC controllers with different degrees of generalisation.

II. Related Work

Methods for robust controller design, such as H^∞ loop-shaping [8], have been successfully used to design robust controllers for autonomous robotic control [9], manipulators [10] and navigation [11], where external perturbations are suppressed by weighting the system transfer function appropriately. Similarly, sliding mode control [12] has been used with robotic manipulators and navigation [13]–[15], in order to have the system *slide* along a suitable sub-manifold defined by multiple discrete control modes. More foundationally, the notion of Lyapunov stability based control [16], has been used also in multi-agent coordination [17], path following [18] and hybrid models of bipedal walking [19], [20], where the system is guaranteed to stay near an equilibrium point if the solution is near the equilibrium.

In control theory, one uses notions such as gain/phase margin and the combination of both of them as disk margin to measure the robustness of such controllers. These measurements are typically defined in a frequency domain associated with the single-input single-output (SISO) or multiple-input multiple-output (MIMO) systems' feedback loop [21]. These margins indicate the limits of disturbances before the system loses maintenance of performance criteria or even stability. Thus, in general, controllers with larger margins are more robust than controllers with smaller margins. Given detailed models and when we have access to models such as the transfer function, we can derive these margins numerically or graphically.

However, such approaches are not well suited to the design of adaptive systems such as those using DNN-based control [22]. Here, we propose a model-agnostic measurement approach for robustness based on the distance from a perturbed state to the closest state within the working limits of a robotic visually grounded controller. Even in data-driven systems, the sought state may not exist in the data. Thus, we use a generative model for data hallucination.

Data hallucination or 'imagination' relates to the generation of data points that are plausible in the domain, but are not part of the original training data set. For example, in image generation GAN methods, a generative model is able to produce images that are indistinguishable from the original set by a discriminator. Conditional-GAN and style transfer [23], [24] are examples of image generation based on feature transformation. Usually, GAN-based methods measure the quality of the generator by computing the entropy of the conditional probability of the generated image belonging to the training set and the entropy of the marginal probability of the generated images [4], [5]. Along the same lines, modification of the latent space in Variational Autoencoders (VAE) [25] has been studied to interpolate features in images in combinations that do not appear in the training data [26], [27]. For control tasks, the authors in [28] use data hallucination by modifying the latent space in a VAE to find tools that can be used to reach a goal in 3D space. Within Reinforcement Learning (RL), data hallucination has been used as a way to find plausible future states of the environment to train agents without having to take the action in the real environment [29], [30] and for imagining predictable and sensitive future states [31]. Differently from model-based RL, where a forward model is used to predict the next state based on the actual state and action, this class of RL methods use internal models to generate rollouts where an imagined reward is used to improve the policy. In regret based strategies [32], an agent compares the actual action to all the other actions that did not take, answering "what could be the value if the agent had tried another action?".

Causal inference enables determination of cause-effect connections [33] in order to be able to make explainable predictions about the world. Causal models provide a framework for modelling causation in addition to statistical relationships [34], [35]. A useful representation in causal models is that of a directed acyclic graph representing the cause of an effect as its parent nodes [36], [37]. The richer structure of causal learning compared to statistical learning allows for a better understanding of the underlying cause-effect properties of a system [38]. This deeper understanding is exploited to explain the behaviour of black-box models [39], [40] and for interpretability [41]. As a specific case of causal inference, counterfactuals have been used to study fairness, accountability and transparency in machine learning [7], [42]. In these cases, the authors use tabular data and low-dimensionality images, where a random search is performed directly over the state variables or over a set of features. In high-dimensional data streams such as with RGB images, counterfactual analysis has been used to explain classifiers by taking salient blobs of a target class image and placing it in the original class image [43]; In [44], an in-filling blob is generated following the original data distribution to change the classification of a DNN. Most of these approaches represent counterfactuals as part of an optimisation problem for a particular instance of the state. In our approach, we train a generative model to produce the counterfactuals for any arbitrary data point.

III. Counterfactual Generator

We use counterfactual inference in three different use cases. First, in the case of a visual robot control, we use it to find the smallest required modification to the input state that allows the robot to solve a control task. Consider an image-based controller under adversarial conditions that prevent the robot from reaching its objective. In this scenario, the desired counterfactual will be the minimal and realistic modification to the image (scene) that allows the robot to indeed achieve the task. For a realistic modification, we mean a change in the state that could plausibly be performed in the real environment. For example, adversarial attacks can be a significant distribution change in the input, but imperceptible to the human eye [3]. On the contrary, the counterfactuals that we consider are changes that have a meaning in the domain, e.g. the configuration of obstacles in a navigation task. Our proposed method is able to generate a solution based on the actual input state, and to provide an evaluation of the efficiency of the solution. We define the relative robustness of a model with respect to another by comparing the distance from the original state to the counterfactually modified state. A lesser distance implies that the counterfactual induces a smaller modification of the original input. Thus, a smaller counterfactual distance implies that the controller is more efficient to recover from adversarial attacks. The generation of counterfactuals is analogous to adaptive methods in configuration space [45]. In these approaches, the robot imagines a solution, via an internal model, for an adversarial attack and then applies the solution to its own configuration. In our case, the counterfactual is in the state space, where an external stakeholder has to apply the modification to the environment.

Now, we present the architecture of our counterfactual generative model for classification model explanation. In this approach, we want to find the minimal modification to the input that will make the classifier change the prediction from the original class to a target class. Given an input $\boldsymbol{x} \in \mathbb{R}^n$ with n dimensions, we want to find an $\boldsymbol{x'} \in \mathbb{R}^n$ that is the closest to the original input that has the highest probability of belonging to the target class $t_c \in \mathbb{N}$:

$$\min_{\boldsymbol{x'}} d_g(\boldsymbol{x}, \boldsymbol{x'}) + d_c(C(\boldsymbol{x'}), t_c), \tag{1}$$

where $d_g \colon \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}$ and $d_c \colon \mathbb{R} \times \mathbb{R} \to \mathbb{R}$ are distance functions on the input space and in the class space respectively, and $C \colon \mathbb{R}^n \to \mathbb{R}$ predicts the probability of the input belonging to t_c.

We propose a generator model $G \colon \mathbb{R}^n \to \mathbb{R}^n$ that takes any arbitrary input and generates the modified version of the input:

$$\boldsymbol{x'} = G(\boldsymbol{x}, \boldsymbol{\theta}) \tag{2}$$

where $\boldsymbol{\theta} \in \mathbb{R}^m$ represents the m parameters of the generator that are trained to minimising the cGen loss function L_{cgen}:

$$\begin{aligned} L_g(\boldsymbol{x}, \boldsymbol{x'}) &= d_g(\boldsymbol{x}, \boldsymbol{x'}) \\ L_c(\boldsymbol{x}, t_c) &= d_c(C(\boldsymbol{x}), t_c) \\ L_{\text{cgen}}(\boldsymbol{x}, \boldsymbol{x'}) &= L_g(\boldsymbol{x}, \boldsymbol{x'}) + L_c(\boldsymbol{x}, t_c) \end{aligned} \tag{3}$$

Following the family of generators GAN, we replace the GAN generator for a counterfactual generator with an auto-encoder architecture. The counterfactual generator optimises Eq. 3 rather than the usual reconstruction loss. Fig. 1 shows the architecture of the classification cGen. Compared to a regular GAN, the cGen takes as input the original value $\boldsymbol{x}$ rather than a randomised input sampled from a probabilistic latent space. The binary classifier is trained to predict if $\boldsymbol{x}$ belongs to the target class t_c. Since the generator has the same architecture as an auto-encoder, we can pre-train it with the original class data subset and a reconstruction loss.

We train the cGen with a GAN mechanism, optimising the generator and the classifier in consecutive epochs. The generator is trained to find the parameters that minimise the weighted multi-variable loss from Eq. 3 as:

$$L_{\text{cgen}}(\boldsymbol{x}, \boldsymbol{x'}) = (1-\alpha) L_g(\boldsymbol{x}, \boldsymbol{x'}) + \alpha L_c(\boldsymbol{x'}, t_c), \tag{4}$$

with $\alpha \in [0, 1]$. The first term on the RHS is the generator loss. The second term on the RHS represents the classifier loss. The idea is that the generator preserves as much as possible the original input (as an auto-encoder does), but also modifying the input, so it is classified as the target class. A value of $\alpha = 1$ generates counterfactuals closer to the target class without any regard for the original input. A value of $\alpha = 0$ will generate counterfactuals identical to the original input without considering the target class. The weights of the classifier are frozen during the training of the generator. On alternating epochs, the classifier is trained to discriminate images belonging to the target class data set or from the generator.

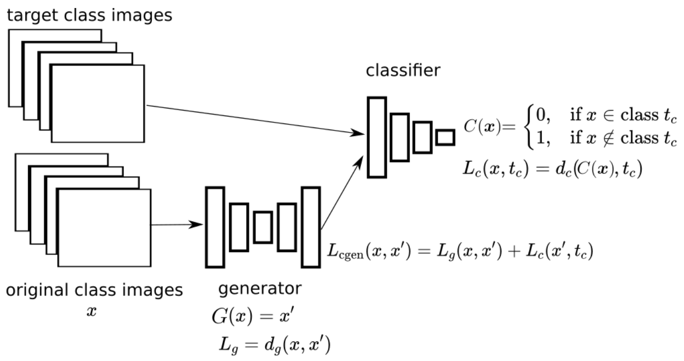

target class images
classifier
$C(x) = \begin{cases} 0, & \text{if } x \in \text{class } t_c \\ 1, & \text{if } x \notin \text{class } t_c \end{cases}$
$L_c(x, t_c) = d_c(C(x), t_c)$
$L_{\text{cgen}}(x, x') = L_g(x, x') + L_c(x', t_c)$
original class images
x
generator
$G(x) = x'$
$L_g = d_g(x, x')$

Fig. 1: The cGen architecture for classification. The generator and the classifier are implemented as deep convolutional neural networks.

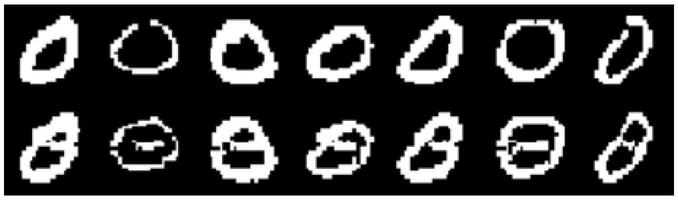


Fig. 2: The top row shows MNIST number in the original class 0. The bottom row shows the counterfactuals for target class 8. Note how the shape of the 0 is maintained in the counterfactual. Balance weight $\alpha = 0.8$.

A. Classification results

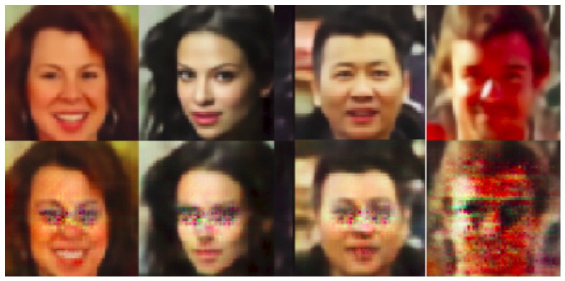
We test the classification version of the cGen with the MNIST handwritten digits and celebrity faces dataset [46].

1) MNIST: We implement the classifier, encoder and decoder part of the generator with four convolutional/deconvolutional layers, and a fully-connected output layer with sigmoid activation. The distance functions d_g and d_c (Eq. 3) were implemented as mean squared error. We pre-trained the generator with the original class subset and the classifier with the training data set from MNIST as a binary classifier for class number 8.

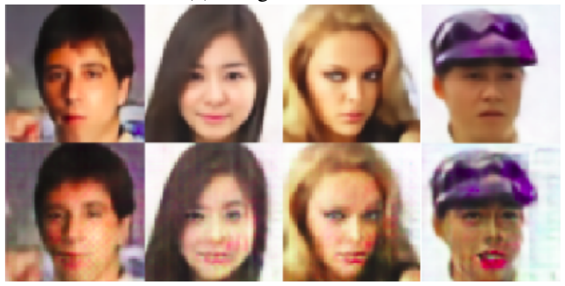
Fig. 2 shows a set of results after the training phase for the original class number 0. The top row shows original class images, and in the same column on the bottom row, the generated output for each image. The examples shown are taken from the validation set, not seen during training. Note that the counterfactuals were obtained from the same generator, on a forward pass for each original image.

2) CelebA: For the celebrities faces dataset, we use a similar implementation adjusting input and kernel sizes from the MNIST implementation. We test the counterfactual generator with two experiments. One for the original class "no sunglasses" and target class "sunglasses". The other experiment is for classes "no smile" and "smile". Following the same steps as with the previous dataset, after pre-training the generator, we run the cGen optimisation.

Fig. 3 shows the results for the celebrity faces experiment. For the sunglasses category, the counterfactuals (bottom row)



(a) Sunglasses class



(b) Smile class

Fig. 3: The top row shows the images of celebrities in the original class. The bottom row shows the counterfactuals for the target class. $\alpha = 0.8$.

show noise in the position where sunglasses are usually located. The last example in the figure shows two black circles in the eyes positions, but also adds noise to the rest of the face. For the smile category, more subtle modifications are made to the lips and the area surrounding it.

IV. cGen for Regression

With a view to applying this methodology to robot control tasks, we now consider the use of the cGen for real-valued predictions.

We add a new component to the cGen architecture. The new component is a predictor function $P\colon \mathbb{R}^n \to \mathbb{R}^m$, with m as the prediction dimension. Fig. 4 shows this architecture. In this setup, the generator includes a new component in the L_{cgen} loss function, the predictor loss L_p. The role of the predictor is to evaluate the counterfactual with respect to a goal prediction. Now, the classifier acts like a regular GAN discriminator. The classifier is used to keep the output of the generator as close as possible to training data set distribution. Minimising the classifier loss accounts for a coherent counterfactual with respect to the training data.

The predictor is the model that calculates the regression. For example, the predictor can be the policy of an agent, a weather forecast, or any type of real-valued function. The new component of the cGen loss captures the prediction error:

$$L_p(\boldsymbol{x'}, t_r) = d_p(P(\boldsymbol{x'}), t_r), \tag{5}$$

where $P(\cdot)$ is the predictor model, $t_r \in \mathbb{R}^m$ a target value and $d_p\colon \mathbb{R}^m \times \mathbb{R}^m \to \mathbb{R}$ a distance function between regression targets. The new cGen loss function for regression is:

$$L_{\text{cgen}}(\boldsymbol{x}, \boldsymbol{x'}) = \alpha L_g(\boldsymbol{x}, \boldsymbol{x'}) + \beta L_c(\boldsymbol{x'}, t_c) + \gamma L_p(\boldsymbol{x'}, t_r). \tag{6}$$

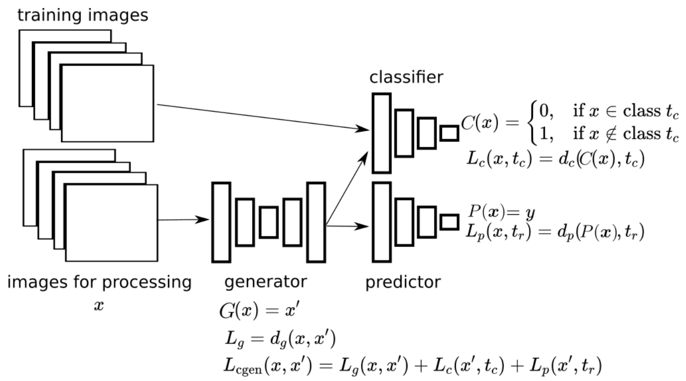
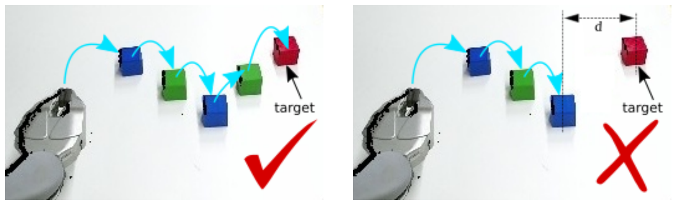

training images
classifier
$C(x) = \begin{cases} 0, & \text{if } x \in \text{class } t_c \\ 1, & \text{if } x \notin \text{class } t_c \end{cases}$
$L_c(x, t_c) = d_c(C(x), t_c)$
$P(x) = y$
$L_p(x, t_r) = d_p(P(x), t_r)$
images for processing
x
generator
predictor
$G(x) = x'$
$L_g = d_g(x, x')$
$L_{\text{cgen}}(x, x') = L_g(x, x') + L_c(x', t_c) + L_p(x', t_r)$

Fig. 4: Regression cGen architecture. The target class t_c of the classifier corresponds to $\boldsymbol{x}$ being part of the training dataset. The predictor includes a target regression t_r.



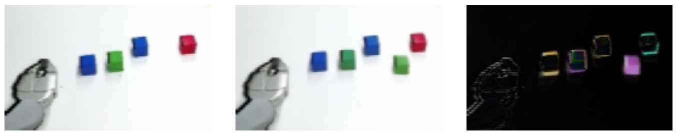
(a) Successful control (b) Failed control

Fig. 5: Image (a) shows a successful scenario where the end effector can reach the target object. The cyan arrows represent the path that the end effector will follow to reach the target (red object). Image (b) shows a scenario where the controller is unable to find a path between the green and red object as the distance d between them is larger than the δ threshold.

Note that t_c is not a target class as in the classification case. Now, t_c targets whether $\boldsymbol{x}$ is part of the training data.

A. *Counterfactuals for Robot Control*

To test the regression cGen architecture, we implement autonomous control on a PR2 robot. Fig. 5a shows the movement of the end effector of the left arm of the robot. From an initial position on the left side of the scene, the end effector transits over the objects to arrive at the red object. The movement from one object to the next is conditional on the distance to the next object. A movement can only be performed if a distance d to the next object is smaller than a set threshold δ. If the distance d to the next object is larger than the threshold δ, the end effector is unable to arrive at the target, and has to remain in the same position failing the objective. This restriction can be seen as a safety measure or domain-specific constraint. Fig. 5b shows a failed execution as the distance from the last visited object to the red cube is larger than δ. We would like to find a counterfactual, i.e. a minimal and realistic modification to the image, wherein the robot does indeed arrive at the target. Valid solutions can include the adjustment of the position of an object, e.g. move the target closer, or adding new objects to fill in the gap between objects.



(a) Original (b) Counterfactual (c) Difference

Fig. 6: (a) is the original input of a failed controller. The end effector is only able to move to the third object (from left to right), missing the target. (b) The counterfactual shows the addition of a new object to the scene. With that modification, the controller can reach the target. (c) is the difference between the original image and the counterfactual. The pink blob represents the new object in the scene. Some reconstruction noise is present around the objects in the original image.

We implement the visual processing based on standard (and fairly classical) computer vision techniques, including thresholding and edge detection. However, a controlled based on this input is not differentiable. Thus, we are not able to use it directly as the predictor in the cGen architecture. To overcome this problem, we trained a neural network with data pairs generated by the original controller. In this network, the input is the image used by the controller, and the output is the distance of the last achievable position of the end effector. If the end effector reaches the target, the output is 0. If the end effector does not reach the target, the output is the distance between the last position of the end effector and the next object. We collected image samples from the on-board camera of the robot. Using data augmentation techniques, we trained a convolutional DNN as a differentiable approximator of the original controller.

Similar to the classification cGen, we can pre-train the generator, but not the classifier (as now it depends on generated images). We use adversarial training for cGen, minimising Eq. 6, with fixed weights for the predictor and the classifier. After one epoch training the generator, we train the classifier with real images and images from the generator. After a short iterative search, the meta-parameters where set to $\alpha = 0.8$, $\beta = 0.1$ and $\gamma = 0.1$. The final results from the generator are passed trough an auto-encoder trained on the original images. This step reduces the noise in the counterfactual making it easier to compare to the original image.

Fig. 6a shows the original image of a failed controller where the robot is unable to reach the target object (red cube). Fig. 6b shows how the generator adds an object (green cube) in the gap between the target (red cube) and the previous object (blue cube). Fig. 6c highlights the difference between the two images. Fig. 6b and 6c serve as an explanation of what modification to the original scene are needed to have a successful controller. Now, we can configure the scene following the counterfactual (adding the green cube) and run the controller. With this modification, the end effector is able to reach the target (not shown on the images).

Fig. 7 shows more counterfactuals. The top rows are the original scenarios where the controller fails. The bottom rows

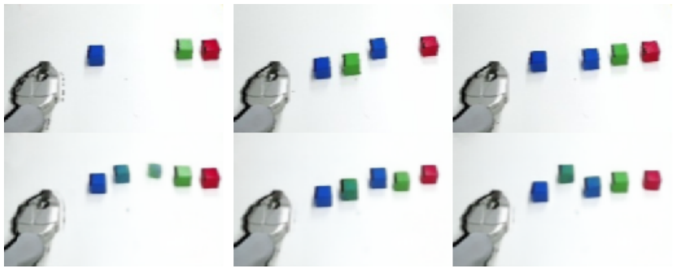


Fig. 7: Other results for regression cGen. The top row shows original images, counterfactuals shown in the bottom row. The generator can add more than one object in the scene, so the controller arrives at the goal. Also, objects are added in other positions to close the gap.

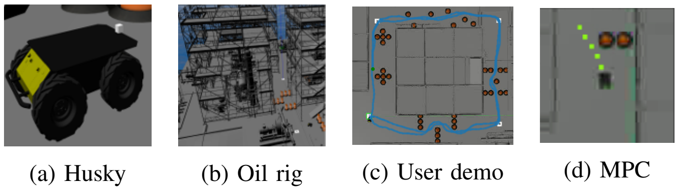


Fig. 8: Learning from demonstration: A user tele-operates a Husky robot (a) on an oil platform digital twin (b) for a surveillance and inspection task (c). The user navigates the robot around the four corners of the platform (blue lines). The system learns from a local top-view camera centred over the robot (d) the future position of the robot (yellow dots).

are the counterfactuals. Note that in the first scenario, the counterfactual adds two objects to fill the gap. The counterfactual were executed by the controller successfully.

B. Model Predictive Control

Now, we test our approach in a high-dimensional regression robotic control problem. In a Learning from Demonstration (LfD) setup, we train a controller to predict the future position of a tele-operated robot in a surveillance task (Fig. 8). The controller learns to predict the future positions in a set of demonstrations provided by the operator. The task for the operator is stated as: starting from the bottom-left corner (Fig. 8c), visit all the corners of the platform in a clockwise direction and return to the starting point. The future position that the controller has to learn to predict is a 10-dimensional vector with the x- and y-axis position of five pairs in the next 5 seconds in intervals of 1 second (Fig. 8d, yellow dots). After training, the learned controller can be used in a model predictive control paradigm to autonomously control the robot.

For this experiment, we modify the cGen architecture for regression presented in the previous section. We modify the generator model from an autoencoder to its variational counterpart [25]. With this modification, we can directly optimise the generator over its latent space Z rather than modify all the parameters of the neural network. First, we pre-train the generator to minimise the reconstruction error. Then, we can directly optimise the cGen loss (Eq. 6) with stochastic gradient

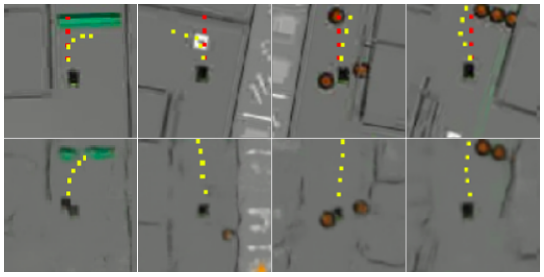


Fig. 9: Counterfactuals generated by the cGen architecture. The top row shows the original input with the model predictive control in yellow dots and the goal in red dots. The bottom row shows the counterfactuals and their predictions.

descend over the latent variables $z_i \in Z$, with $i = \{1, 2, \dots, l\}$ and l the number of latent variables. This modification of the architecture has two main advantages. First, the optimisation process of the generator is linear with respect to the size of the latent space ($\mathcal{O}(l)$), thus speeding up the training time. Second, the counterfactual generated images are an interpolation of the latent space, rendering images closer to the original training set. The downside of this architecture change is that the decoder and encoder modules of the generator need to be pre-trained and the counterfactual search has to be performed over each instance of an input image $\boldsymbol{x}$, and one regression goal t_r.

Fig. 9 shows counterfactuals for a set of scenarios taken from the demonstrations with a goal that is different from the prediction of the controller. The top row shows the original scenarios with the model predictive control as the yellow dots. The straight line of red dots represents the regression goal for the counterfactuals t_r. The bottom row shows the counterfactuals for the original input. In the first figure, the generator changes the green barrier for two barriers with a separation in the middle. Note the counterfactual predictive control (bottom row yellow dots) is closer to the goal (top row red dots), but is not an exact solution for the required goal. This difference is due to the fact that there is no scenario in the demonstration that includes a green barrier only on the right side of the hallway. Such a solution is discarded by a higher loss in the discriminator. For the rest of the scenarios, the counterfactuals include addition and removal of obstacles and rotation of the scene.

V. CGen for Robustness Analysis

As discussed earlier, robust control design considers how well a controller maintains performance criteria when facing external perturbations. One controller is considered to be more robust than another if it can control the plant under larger perturbations than the other controller. In this section, we use counterfactual analysis to obtain an indicator of the efficiency of a controller to return to a working regime after it is affected by perturbations. The idea is that when a system faces a perturbation or an adversarial attack, then instead of adapting, e.g. via changes in the configuration space, the counterfactual analysis is able to find a solution in the state space. For example in the oil platform scenario, where the MPC is based on images taken from a top-view camera, a failure to arrive at a destination goal can be due to obstacles in the path of the robot. This scenario can be solved by manipulating the objects in the scene (not necessarily by the robot, instead by requesting human help). In this case, for different controllers, we can measure the quality of the counterfactual solution by the number of modifications to the scene induced by the cGen and by the distance between the desired goal and the one obtained with the counterfactual image.

To measure the difference between controllers, we trained three controllers under different demonstrations. For the three controllers, the surveillance task remains the same as in Sec. IV-B, but with different obstacle complexity. The first controller (a) is trained in scenarios with two types of obstacles: orange cones and green barriers. The second controller (b) is trained from demonstrations with scenarios with cones but no barriers. The third controller (c) is trained in scenarios without obstacles. We define a set of goals for the MPC to predict movements of the robot in a range of $[-45, 45]$ degrees from the vertical line.

As a baseline, we measure the robustness of the controllers to random noise in the input data. We inject noise μ with an increasing gain η until the difference between the non-modified prediction and the modified one is larger than a set value ε: $||p(\boldsymbol{x}) - p(\boldsymbol{x} + \eta\mu)||^2 < \varepsilon$. If the difference is larger than ε, the controller has arrived to its limit to reject perturbations. Fig. 10 shows that the sensitivity to noise is similar for the three controllers, i.e. they all share similar robustness to noise perturbation. To measure the average size of counterfactual modifications required by each controller to return to a working regime, we use cGen to generate counterfactuals over a set of 10 scenarios with different goals. Fig. 11 shows the average losses for the three controllers. The counterfactual cGen loss (Eq. 6) shows that controller a has the lowest loss among the three controllers. This is confirmed by the generator and predictor loss. These two low values indicate that the counterfactuals are close to the original image and also produce a prediction close to the goals. Controller b shows a worst generator loss but a similar prediction loss compared to controller a. This behaviour can be explained by the missing training data including the green barrier, but still maintaining a good level of sensitivity to the input. The counterfactuals for the controller b, when faced in a scenario with previously unseen obstacles (barriers), result in larger modification of the scene to allow the model to make a closer prediction to the goal. In this case, the counterfactual not only have to modify (or remove) the barrier; it also has to add cones to guide the controller to the goal. Controller c shows the worst counterfactual performance among the three controllers. The generator and prediction loss are both high. We explain this behaviour as that the controller does not respond to obstacles on the input image –as it did not experience them during training–. Also, the counterfactuals will tend to remove all the object from the scene rather than applying smaller modifications.

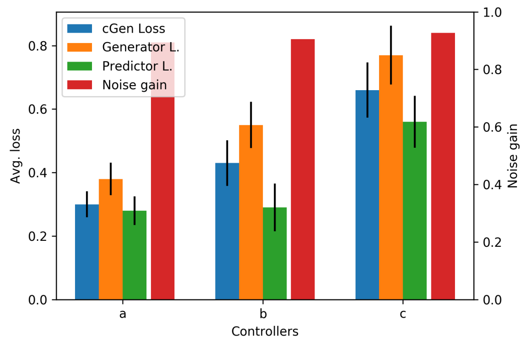


Fig. 10: Three controllers trained under different levels of complexity of demonstration. All controllers have a similar response to perturbations (η as the noise gain and $\varepsilon = 0.1$). Controller a was trained with full demonstrations. Controller b worst performance compared to controller a is explained by the unseen obstacles during training. Controller c has the worst performance among the controllers. The lack of obstacles during training does not allow the controller to produce more complex behaviour.

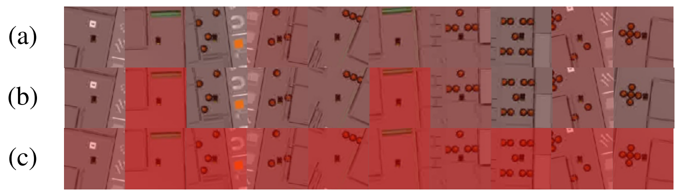


Fig. 11: Performance of the controller in different scenarios. Strength of the red colour indicates a higher counterfactual loss. Controller b has a larger loss on scenarios with a barrier. Controller c performs comparatively bad in all the scenarios.

Fig. 11 shows the counterfactual loss for each scenario for controller a, b and c. In the image, a stronger red colour means a higher value average loss for all the goals. Comparing controller a and b, we can see that they have similar values for the loss in most of the scenarios. Still, for both of the scenarios where a green barrier is present, controller b has a worst result. Controller c has a lower performance in most on the scenarios, where only the first scenario without obstacles is comparable to the other two controllers.

VI. Conclusion

In this work, we present an approach to characterising relative robustness in image-based robotic control. We present an architecture to train a counterfactual generative model that seeks to produce small, yet close-to-the-original distribution, modifications to a known input during the process of solving classification and regression problems. We evaluate this approach in robot control tasks, after first demonstrating the principle in object classification. Our proposed robustness measure takes into consideration both the controller and the environment, relating it to embodiment theory [47].

REFERENCES

[1] K. Zhou and J. C. Doyle, *Essentials of robust control*. Prentice hall Upper Saddle River, NJ, 1998, vol. 104.

[2] C. Zhang, S. Bengio, M. Hardt, B. Recht, and O. Vinyals, "Understanding deep learning requires rethinking generalization," *arXiv preprint arXiv:1611.03530*, 2016.

[3] C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. Goodfellow, and R. Fergus, "Intriguing properties of neural networks," *arXiv preprint arXiv:1312.6199*, 2013.

[4] T. Salimans, I. Goodfellow, W. Zaremba, V. Cheung, A. Radford, and X. Chen, "Improved techniques for training GANs," in *Advances in neural information processing systems*, 2016, pp. 2234–2242.

[5] M. Heusel, H. Ramsauer, T. Unterthiner, B. Nessler, and S. Hochreiter, "GANs trained by a two time-scale update rule converge to a local nash equilibrium," in *Advances in neural information processing systems*, 2017, pp. 6626–6637.

[6] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," in *Advances in neural information processing systems*, 2014, pp. 2672–2680.

[7] S. Wachter, B. Mittelstadt, and C. Russell, "Counterfactual explanations without opening the black box: Automated decisions and the GPDR," *Harv. JL & Tech.*, vol. 31, p. 841, 2017.

[8] K. Glover and D. McFarlane, "A loop shaping design procedure using H_∞- synthesis," *IEEE Transactions on Automatic Control*, vol. 37, no. 6, pp. 759–769, 1992.

[9] M. La Civita, G. Papageorgiou, W. Messner, and T. Kanade, "Design and flight testing of a high-bandwidth h-infinity loop shaping controller for a robotic helicopter," in *AIAA Guidance, Navigation, and Control Conference and Exhibit*, 2002, p. 4836.

[10] J. S. Yeon and J. H. Park, "Practical robust control for flexible joint robot manipulators," in *2008 IEEE international conference on robotics and automation*. IEEE, 2008, pp. 3377–3382.

[11] G. Rigatos and P. Siano, "A new nonlinear h-infinity feedback control approach to the problem of autonomous robot navigation," *Intelligent Industrial Systems*, vol. 1, no. 3, pp. 179–186, 2015.

[12] A. S. Zinober, "Deterministic control of uncertain systems," in *Proceedings. ICCON IEEE International Conference on Control and Applications*. IEEE, 1989, pp. 645–650.

[13] S. Yu, X. Yu, B. Shirinzadeh, and Z. Man, "Continuous finite-time control for robotic manipulators with terminal sliding mode," *Automatica*, vol. 41, no. 11, pp. 1957–1964, 2005.

[14] L. Wang, T. Chai, and L. Zhai, "Neural-network-based terminal sliding-mode control of robotic manipulators including actuator dynamics," *IEEE Transactions on Industrial Electronics*, vol. 56, no. 9, pp. 3296–3304, 2009.

[15] J. Guldner and V. I. Utkin, "Sliding mode control for gradient tracking and robot navigation using artificial potential fields," *IEEE Transactions on Robotics and Automation*, vol. 11, no. 2, pp. 247–254, 1995.

[16] A. S. Zinober, *Variable structure and Lyapunov control*. Springer, 1994, vol. 193.

[17] P. Ogren, M. Egerstedt, and X. Hu, "A control Lyapunov function approach to multi-agent coordination," in *Proceedings of the 40th IEEE Conference on Decision and Control (Cat. No. 01CH37228)*, vol. 2. IEEE, 2001, pp. 1150–1155.

[18] S. Park, J. Deyst, and J. P. How, "Performance and Lyapunov stability of a nonlinear path following guidance method," *Journal of guidance, control, and dynamics*, vol. 30, no. 6, pp. 1718–1728, 2007.

[19] A. D. Ames, K. Galloway, K. Sreenath, and J. W. Grizzle, "Rapidly exponentially stabilizing control Lyapunov functions and hybrid zero dynamics," *IEEE Transactions on Automatic Control*, vol. 59, no. 4, pp. 876–891, 2014.

[20] S. Ramamoorthy and B. Kuipers, "Qualitative hybrid control of dynamic bipedal walking." in *Robotics: Science and Systems.*, 2006.

[21] J. D. Blight, R. Lane Dailey, and D. Gangsaas, "Practical control law design for aircraft using multivariable techniques," *International Journal of Control*, vol. 59, no. 1, pp. 93–137, 1994.

[22] S.-Y. Chu and C.-C. Teng, "Tuning of pid controllers based on gain and phase margin specifications using fuzzy neural network," *Fuzzy sets and systems*, vol. 101, no. 1, pp. 21–30, 1999.

[23] K. Sricharan, R. Bala, M. Shreve, H. Ding, K. Saketh, and J. Sun, "Semi-supervised conditional GANs," *arXiv preprint arXiv:1708.05789*, 2017.

[24] L. A. Gatys, A. S. Ecker, and M. Bethge, "Image style transfer using convolutional neural networks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 2414–2423.

[25] D. P. Kingma and M. Welling, "Auto-encoding variational Bayes," *arXiv preprint arXiv:1312.6114*, 2013.

[26] M. F. Mathieu, J. J. Zhao, J. Zhao, A. Ramesh, P. Sprechmann, and Y. LeCun, "Disentangling factors of variation in deep representation using adversarial training," in *Advances in neural information processing systems*, 2016, pp. 5040–5048.

[27] D. Berthelot, C. Raffel, A. Roy, and I. Goodfellow, "Understanding and improving interpolation in autoencoders via an adversarial regularizer," *arXiv preprint arXiv:1807.07543*, 2018.

[28] Y. Wu, S. Kasewa, O. Groth, S. Salter, L. Sun, O. P. Jones, and I. Posner, "Imagine that! leveraging emergent affordances for tool synthesis in reaching tasks," *arXiv preprint arXiv:1909.13561*, 2019.

[29] S. Racanière, T. Weber, D. Reichert, L. Buesing, A. Guez, D. Jimenez Rezende, A. Puigdomènech Badia, O. Vinyals, N. Heess, Y. Li, R. Pascanu, P. Battaglia, D. Hassabis, D. Silver, and D. Wierstra, "Imagination-augmented agents for deep reinforcement learning," in *Advances in Neural Information Processing Systems 30*, I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, Eds. Curran Associates, Inc., 2017, pp. 5690–5701. [Online]. Available: http://papers.nips.cc/paper/7152-imagination-augmented-agents-for-deep-reinforcement-learning.pdf

[30] G. Kalweit and J. Boedecker, "Uncertainty-driven imagination for continuous deep reinforcement learning," in *Conference on Robot Learning*, 2017, pp. 195–206.

[31] S. C. Smith and J. M. Herrmann, "Homeokinetic reinforcement learning," in *Partially Supervised Learning*, F. Schwenker and E. Trentin, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, pp. 82–91.

[32] J. N. Lee, M. Laskey, A. K. Tanwani, A. Aswani, and K. Goldberg, "A dynamic regret analysis and adaptive regularization algorithm for on-policy robot imitation learning," in *International Workshop on the Algorithmic Foundations of Robotics*. Springer, 2018, pp. 212–227.

[33] J. Pearl *et al.*, "Causal inference in statistics: An overview," *Statistics surveys*, vol. 3, pp. 96–146, 2009.

[34] J. Pearl, *Causality*. Cambridge university press, 2009.

[35] S. Nair, Y. Zhu, S. Savarese, and L. Fei-Fei, "Causal induction from visual observations for goal directed tasks," *arXiv preprint arXiv:1910.01751*, 2019.

[36] P. O. Hoyer, D. Janzing, J. M. Mooij, J. Peters, and B. Schölkopf, "Nonlinear causal discovery with additive noise models," in *Advances in Neural Information Processing Systems 21*, D. Koller, D. Schuurmans, Y. Bengio, and L. Bottou, Eds. Curran Associates, Inc., 2009, pp. 689–696. [Online]. Available: http://papers.nips.cc/paper/3548-nonlinear-causal-discovery-with-additive-noise-models.pdf

[37] N. Shajarisales, D. Janzing, B. Schölkopf, and M. Besserve, "Telling cause from effect in deterministic linear dynamical systems," in *International Conference on Machine Learning*, 2015, pp. 285–294.

[38] J. Peters, D. Janzing, and B. Schölkopf, *Elements of causal inference: foundations and learning algorithms*. MIT press, 2017.

[39] D. Alvarez-Melis and T. S. Jaakkola, "A causal framework for explaining the predictions of black-box sequence-to-sequence models," *arXiv preprint arXiv:1707.01943*, 2017.

[40] S. Singla, B. Pollack, J. Chen, and K. Batmanghelich, "Explanation by progressive exaggeration," 2019.

[41] J. Kim and J. Canny, "Interpretable learning for self-driving cars by visualizing causal attention," in *Proceedings of the IEEE international conference on computer vision*, 2017, pp. 2942–2950.

[42] A. Van Looveren and J. Klaise, "Interpretable counterfactual explanations guided by prototypes," *arXiv preprint arXiv:1907.02584*, 2019.

[43] Y. Goyal, Z. Wu, J. Ernst, D. Batra, D. Parikh, and S. Lee, "Counterfactual visual explanations," *arXiv preprint arXiv:1904.07451*, 2019.

[44] C.-H. Chang, E. Creager, A. Goldenberg, and D. Duvenaud, "Explaining image classifiers by counterfactual generation," *arXiv preprint arXiv:1807.08024*, 2018.

[45] J. Bongard, V. Zykov, and H. Lipson, "Resilient machines through continuous self-modeling," *Science*, vol. 314, no. 5802, pp. 1118–1121, 2006.

[46] Z. Liu, P. Luo, X. Wang, and X. Tang, "Large-scale celebfaces attributes (celeba) dataset," *Retrieved August*, vol. 15, p. 2018, 2018.

[47] A. Clark and M. A. Boden, "Being there: Putting brain, body, and world together again," *Nature*, vol. 386, no. 6622, pp. 237–237, 1997.