
COUNTERFACTUAL TRAINING: TEACHING MODELS PLAUSIBLE AND ACTIONABLE EXPLANATIONS

A PREPRINT

Patrick Altmeyer 

Faculty of Electrical Engineering, Mathematics and Computer Science
Delft University of Technology

p.altmeyer@tudelft.nl

Arie van Deursen

Faculty of Electrical Engineering, Mathematics and Computer Science
Delft University of Technology

Cynthia C. S. Liem

Faculty of Electrical Engineering, Mathematics and Computer Science
Delft University of Technology

February 11, 2025

ABSTRACT

Counterfactual Explanations have emerged as a popular tool to explain predictions made by opaque machine learning models: they explain how factual inputs need to change in order for some fitted model to produce some desired output. Much existing research has focused on identifying explanations that are not only valid but also deemed plausible and desirable with respect to the underlying data and stakeholder requirements. Recent work has shown that under this premise, the task of learning plausible explanations is effectively reassigned from the model itself to the (post-hoc) counterfactual explainer. Building on that work, we propose a novel model objective that leverages counterfactuals during the training phase (ad-hoc) in order to minimize the divergence between learned representations and plausible explanations. Through extensive experiments, we demonstrate that our proposed methodology facilitates training models that inherently deliver plausible explanations while maintaining high predictive performance.

Keywords Counterfactual Explanations • Explainable AI • Representation Learning

1 Introduction

Today's prominence of artificial intelligence (AI) has largely been driven by advances in **representation learning**: instead of relying on features and rules that are carefully hand-crafted by humans, modern machine learning (ML) models are tasked with learning these representations from scratch, guided by narrow objectives such as predictive accuracy (I. Goodfellow, Bengio, and Courville 2016). Modern advances in computing have made it possible to provide such models with ever greater degrees of freedom to achieve that task, which has often led them to outperform traditionally more parsimonious models. Unfortunately, in doing so they also learn increasingly complex and highly sensitive representations that we can no longer easily interpret.

This trend towards complexity for the sake of performance has come under serious scrutiny in recent years. At the very cusp of the deep learning revolution, Szegedy et al. (2013) showed that artificial neural networks (ANN) are sensitive

23 to adversarial examples: counterfactuals of model inputs that yield vastly different model predictions despite being
 24 “imperceptible” in that they are semantically indifferent from their factual counterparts. Despite partially effective
 25 mitigation strategies such as **adversarial training** (I. J. Goodfellow, Shlens, and Szegedy 2014), truly robust deep
 26 learning (DL) remains unattainable even for models that are considered shallow by today’s standards (Kolter 2023).

27 Part of the problem is that high degrees of freedom provide room for many solutions that are locally optimal with
 28 respect to narrow objectives (Wilson 2020)¹. Based purely on predictive performance, these solutions may seem to
 29 provide compelling explanations for the data, when in fact they are based on purely associative, semantically mean-
 30 ingless patterns. This poses two related challenges: firstly, it makes these models inherently opaque, since humans
 31 cannot simply interpret what type of explanation the complex learned representations correspond to; secondly, even
 32 if we could resolve the first challenge, it is not obvious how to mitigate models from learning representations that
 33 correspond to meaningless and implausible explanations.

34 The first challenge has attracted an abundance of research on **explainable AI** (XAI) which aims to develop tools to
 35 derive explanations from complex model representations. This can mitigate a scenario in which we deploy opaque
 36 models and blindly rely on their predictions. On countless occasions, this scenario has already occurred in practice
 37 and caused real harm to people who were affected adversely and often unfairly by automated decision-making systems
 38 (ADMS) involving opaque models (O’Neil 2016). Effective XAI tools can aide us in monitoring models and providing
 39 recourse to individuals to turn adverse outcomes (e.g. “loan application rejected”) into positive ones (“application
 40 accepted”). Wachter, Mittelstadt, and Russell (2017) propose **counterfactual explanations** as an effective approach
 41 to achieve this: they explain how factual inputs need to change in order for some fitted model to produce some desired
 42 output, typically involving minimal perturbations.

43 To our surprise, the second challenge has not yet attracted any consolidated research effort. Specifically, there has
 44 been no concerted effort towards improving model **explainability**, which we define here as the degree to which learned
 45 representations correspond to explanations that are interpretable and deemed **plausible** by humans (see Definition 3.1).
 46 Instead, the choice has typically been to improve the capacity of XAI tools to identify the subset explanations that are
 47 both plausible and valid for any given model, independent of whether the learned representations are also compatible
 48 with implausible explanations (Altmeyer et al. 2024). Fortunately, recent findings indicate that explainability can arise
 49 as byproduct of regularization techniques aimed at other objectives such as robustness, generalization and generative
 50 capacity Altmeyer et al. (2024).

51 Building on these findings, we introduce **counterfactual training**: a novel regularization technique geared explicitly
 52 towards aligning model representations with plausible explanations. Our contributions are as follows:

- 53 • We discuss existing related work on improving models and consolidate it through the lens of counterfactual
 54 explanations (Section 2).
- 55 • We present our proposed methodological framework that leverages faithful counterfactual explanations during
 56 the training phase of models to achieve the explainability objective (Section 3).
- 57 • Through extensive experiments we demonstrate the counterfactual training improve model explainability
 58 while maintaining high predictive performance. We run ablation studies and grid searches to understand
 59 how the underlying model components and hyperparameters affect outcomes. (Section 4).

60 Despite limitations of our approach discussed in Section 5, we conclude that counterfactual training provides a practi-
 61 cal framework for researchers and practitioners interested in making opaque models more trustworthy Section 6. We
 62 also believe that this work serves as an opportunity for XAI researchers to reevaluate the premise of improving XAI
 63 tools without improving models.

64 2 Related Literature

65 To the best of our knowledge, our proposed framework for counterfactual training represents the first attempt to use
 66 counterfactual explanations during training to improve model explainability. In high-level terms, we define model
 67 explainability as the extent to which valid explanations derived for an opaque model are also deemed plausible with
 68 respect to the underlying data and stakeholder requirements. To make this more concrete, we follow Augustin, Meinke,
 69 and Hein (2020) in tying the concept of explainability to the quality of counterfactual explanations that we can
 70 generate for a given model. The authors show that counterfactual explanations—understood here as minimal input
 71 perturbations that yield some desired model prediction—are generally more meaningful if the underlying model is
 72 more robust to adversarial examples. We can make intuitive sense of this finding when looking at adversarial training
 73 (AT) through the lens of representation learning with high degrees of freedom: by inducing models to “unlearn”

¹For clarity: we follow standard ML convention in using “degrees of freedom” to refer to the number of parameters estimated from data.

74 representations that are susceptible to worst-case counterfactuals (i.e. adversarial examples), AT effectively removes
 75 some implausible explanations from the solution space.

76 2.1 Adversarial Examples are Counterfactual Explanations

77 This interpretation of the link between explainability through counterfactuals on one side, and robustness to adversarial
 78 examples on the other, is backed by empirical evidence. Sauer and Geiger (2021) demonstrate that using counterfactual
 79 images during classifier training improves model robustness. Similarly, Abbasnejad et al. (2020) argue that counterfactuals
 80 represent potentially useful training data in machine learning, especially in supervised settings where inputs may
 81 be reasonably mapped to multiple outputs. They, too, demonstrate the augmenting the training data of image classifi-
 82 cers can improve generalization. Teney, Abbasnejad, and Hengel (2020) propose an approach using counterfactuals
 83 in training that does not rely on data augmentation: they argue that counterfactual pairs typically already exist in train-
 84 ing datasets. Specifically, their approach relies on, firstly, identifying similar input samples with different annotations
 85 and, secondly, ensuring that the gradient of the classifier aligns with the vector between pairs of counterfactual inputs
 86 using the cosine distance as a loss function. In the natural language processing (NLP) domain, counterfactuals have
 87 similarly been used to improve models through data augmentation: Wu et al. (2021), propose *POLYJUICE*, a general-
 88 purpose counterfactual generator for language models. They demonstrate empirically that augmenting training data
 89 through *POLYJUICE* counterfactuals improves robustness in a number of NLP tasks. Luu and Inoue (2023) introduce
 90 Counterfactual Adversarial Training (CAT), which also aims at improving generalization and robustness of language
 91 models. Specifically, they propose to proceed as follows: firstly, they identify training samples that are subject to
 92 high predictive uncertainty; secondly, they generate counterfactual explanations for those samples; and, finally, they
 93 fine-tune the given language model on the augmented dataset that includes the generated counterfactuals.

94 There have also been several attempts at formalizing the relationship between counterfactual explanations (CE) and
 95 adversarial examples (AE). Pointing to clear similarities in how CE and AE are generated, Freiesleben (2022) makes
 96 the case for jointly studying the opaqueness and robustness problem in representation learning. Formally, AE can
 97 be seen as the subset of CE, for which misclassification is achieved (Freiesleben 2022). Similarly, Pawelczyk et
 98 al. (2022) show that CE and AE are equivalent under certain conditions and derive theoretical upper bounds on the
 99 distances between them.

100 Two recent works are closely related to ours in that they use counterfactuals during training with the explicit goal
 101 of affecting certain properties of post-hoc counterfactual explanations. Firstly, Ross, Lakkaraju, and Bastani (2024)
 102 propose a way to train models that are guaranteed to provide recourse for individuals to move from an adverse outcome
 103 to some positive target class with high probability. The approach proposed by Ross, Lakkaraju, and Bastani (2024)
 104 builds on adversarial training, where in this context susceptibility to targeted adversarial examples for the positive
 105 class is explicitly induced. The proposed method allows for imposing a set of actionability constraints ex-ante: for
 106 example, users can specify that certain features (e.g. *age*, *gender*, ...) are immutable. Secondly, Guo, Nguyen, and
 107 Yadav (2023) are the first to propose an end-to-end training pipeline that includes counterfactual explanations as part
 108 of the training procedure. In particular, they propose a specific network architecture that includes a predictor and CE
 109 generator network, where the parameters of the CE generator network are learnable. Counterfactuals are generated
 110 during each training iteration and fed back to the predictor network. In contrast to Guo, Nguyen, and Yadav (2023),
 111 we impose no restrictions on the neural network architecture at all.

112 2.2 Beyond Robustness

113 Improving the adversarial robustness of models is not the only path towards aligning representations with plausible
 114 explanations. In a work closely related to this one, Altmeyer et al. (2024) show that explainability can be improved
 115 through model averaging and refined model objectives. The authors propose a way to generate counterfactuals that
 116 are maximally **faithful** to the model in that they are consistent with what the model has learned about the underlying
 117 data. Formally, they rely on tools from energy-based modelling to minimize the divergence between the distribution
 118 of counterfactuals and the conditional posterior over inputs learned by the model. Their proposed counterfactual
 119 explainer, *ECCo*, yields plausible explanations if and only if the underlying model has learned representations that
 120 align with them. They find that both deep ensembles (Lakshminarayanan, Pritzel, and Blundell 2017) and joint energy-
 121 based models (JEMs) (Grathwohl et al. 2020) tend to do well in this regard.

122 Once again it helps to look at these findings through the lens of representation learning with high degrees of freedom.
 123 Deep ensembles are approximate Bayesian model averages, which are most called for when models are underspecified
 124 by the available data (Wilson 2020). Averaging across solutions mitigates the aforementioned risk of relying on a
 125 single locally optimal representations that corresponds to semantically meaningless explanations for the data. Previous
 126 work by Schut et al. (2021) similarly found that generating plausible (“interpretable”) counterfactual explanations is
 127 almost trivial for deep ensembles that have also undergone adversarial training. The case for JEMs is even clearer:
 128 they involve a hybrid objective that induces both high predictive performance and generative capacity (Grathwohl et al.

129 This is closely related to the idea of aligning models with plausible explanations and has inspired our proposed
 130 counterfactual training objective, as we explain in Section 3.

131 3 Counterfactual Training

132 Counterfactual training combines ideas from adversarial training, energy-based modelling and counterfactuals expla-
 133 nations with the explicit objective of aligning representations with plausible explanations that comply with user re-
 134 quirements. In the context of CE, plausibility has broadly been defined as the degree to which counterfactuals comply
 135 with the underlying data generating process (Poyiadzi et al. 2020; Guidotti 2022; Altmeyer et al. 2024). Plausibility
 136 is a necessary but insufficient condition for using CE to provide algorithmic recourse (AR) to individuals affected by
 137 opaque models in practice. This is because for recourse recommendations to be **actionable**, they need to not only
 138 result in plausible counterfactuals but also be attainable. A plausible CE for a rejected 20-year-old loan applicant, for
 139 example, might reveal that their application would have been accepted, if only they were 20 years older. Ignoring all
 140 other features, this complies with the definition of plausibility if 40-year-old individuals are in fact more credit-worthy
 141 on average than young adults. But of course this CE does not qualify for providing actionable recourse to the applicant
 142 since *age* is not a mutable feature. For our intents and purposes, counterfactual training aims at improving model ex-
 143 plainability by aligning models with counterfactuals that meet both desiderata, plausibility and actionability. Formally,
 144 we define explainability as follows:

145 **Definition 3.1** (Model Explainability). Let $\mathbf{M}_\theta : \mathcal{X} \mapsto \mathcal{Y}$ denote a supervised classification model that maps from the
 146 D -dimensional input space \mathcal{X} to representations $\phi(\mathbf{x}; \theta)$ and finally to the K -dimensional output space \mathcal{Y} . Assume that
 147 for any given input-output pair $\{\mathbf{x}, \mathbf{y}\}_i$ there exists a counterfactual $\mathbf{x}' = \mathbf{x} + \Delta : \mathbf{M}_\theta(\mathbf{x}') = \mathbf{y}^+ \neq \mathbf{y} = \mathbf{M}_\theta(\mathbf{x})$ where
 148 \mathbf{y}^+ denotes some target output. We say that \mathbf{M}_θ is **explainable** to the extent that faithfully generated counterfactuals
 149 are plausible (i.e. consistent with the data) and actionable. Formally, we define these properties as follows:

- 150 1. (Plausibility) $\int^A p(\mathbf{x}|\mathbf{y}^+) d\mathbf{x} \rightarrow 1$ where A is some small region around \mathbf{x}' .
 151 2. (Actionability) Permutations Δ are subject to actionability constraints.

152 We consider counterfactuals as faithful to the extent that they are consistent with what the model has learned about the
 153 input data. Let $p_\theta(\mathbf{x}|\mathbf{y}^+)$ denote the conditional posterior over inputs, then formally:

- 154 3. (Faithfulness) $\int^A p_\theta(\mathbf{x}|\mathbf{y}^+) d\mathbf{x} \rightarrow 1$ where A is defined as above.

155 The definitions of faithfulness and plausibility in Definition 3.1 are the same as in Altmeyer et al. (2024), with adapted
 156 notation. Actionability constraints in Definition 3.1 vary and depend on the context in which \mathbf{M}_θ is deployed. In this
 157 work, we focus on domain and mutability constraints for individual features x_d for $d = 1, \dots, D$. We limit ourselves
 158 to classification tasks for reasons discussed in Section 5.

159 3.1 Our Proposed Objective

160 To train models with high explainability as defined in Definition 3.1, we propose the following objective,

$$\text{yloss}(\mathbf{M}_\theta(\mathbf{x}), \mathbf{y}) + \lambda_{\text{div}} \text{div}(\mathbf{x}, \mathbf{x}', \mathbf{y}; \theta) + \lambda_{\text{adv}} \text{advloss}(\mathbf{M}_\theta(\mathbf{x}'), \mathbf{y}) \quad (1)$$

161 where $\text{yloss}(\cdot)$ denotes any conventional classification loss function (e.g. cross-entropy) that induces discriminative
 162 (predictive) performance. The two additional components in Equation 1 are explained in more detail below. For now,
 163 they can be sufficiently described as inducing explainability directly and indirectly by penalizing: 1) the contrastive
 164 divergence, $\text{div}(\cdot)$, between counterfactuals x' and observed samples x and, 2) the adversarial loss, $\text{advloss}(\cdot)$, with
 165 respect to counterfactuals. The tradeoff between the different components can be governed by adjusting the strengths
 166 of the penalties λ_{div} and λ_{adv} .

167 3.1.1 Directly Inducing Explainability through Contrastive Divergence

168 Grathwohl et al. (2020) observe that any classifier can be re-interpreted as a joint energy-based model (JEM)
 169 that learns to discriminate output classes conditional on inputs and generate inputs. They show that JEMs can be
 170 trained to perform well at both tasks by directly maximizing the joint log-likelihood factorized as $\log p_\theta(\mathbf{x}, \mathbf{y}) =$
 171 $\log p_\theta(\mathbf{y}|\mathbf{x}) + \log p_\theta(\mathbf{x})$. The first factor can be optimized using conventional cross-entropy as in Equation 1. To
 172 optimize $\log p_\theta(\mathbf{x})$ Grathwohl et al. (2020) minimize the contrastive divergence between samples drawn from $p_\theta(\mathbf{x})$
 173 and training observations, i.e. samples from $p(\mathbf{x})$.

174 A key empirical finding in Altmeyer et al. (2024) was that JEMs tend to do well with respect to the plausibility objec-
 175 tive in Definition 3.1. If we consider samples drawn from $p_\theta(\mathbf{x})$ as counterfactuals, this is an expected finding, because

176 the JEM objective effectively minimizes the divergence between the conditional posterior and $p(\mathbf{x}|\mathbf{y}^+)$. To generate
 177 samples, Grathwohl et al. (2020) rely on Stochastic Gradient Langevin Dynamics (SGLD) using an uninformative
 178 prior for initialization. This is where we depart from their methodology: instead of generating samples through SGLD,
 179 we propose using counterfactual explainers to generate counterfactuals for observed training samples. Specifically, we
 180 have

$$\text{div}(\mathbf{x}, \mathbf{x}', y; \theta) = \mathcal{E}_\theta(\mathbf{x}, y) - \mathcal{E}_\theta(\mathbf{x}', y) \quad (2)$$

181 where $\mathcal{E}_\theta(\cdot)$ denotes the energy function. We generate samples \mathbf{x}' by first randomly sampling the target class $y^+ \sim$
 182 $p(y)$ and then generating a counterfactual explanation for that target, similar to how conditional sampling is used to
 183 draw from $p_\theta(\mathbf{x})$ in Grathwohl et al. (2020). In particular, we set $\mathcal{E}_\theta(\mathbf{x}, y) = -\mathbf{M}_\theta(\mathbf{x})[y^+]$ where y^+ denotes the
 184 index of the target class.

185 Intuitively, the gradient of Equation 2 decreases the energy of observed training samples (positive samples) while at
 186 same time increasing the energy of counterfactuals (negative samples) (Du and Mordatch 2020). As the generated
 187 counterfactuals get more plausible (Definition 3.1) over the cause of training, these two opposing effects gradually
 188 balance each out (Lippe 2024).

189 The departure from SGLD allows us to tap into the vast repertoire of explainers that have been proposed in the literature
 190 to meet different desiderata. Typically, these methods facilitate the imposition of domain and mutability constraints,
 191 for example. In principle, any existing approach for generating counterfactual explanations is viable, so long as it does
 192 not violate the faithfulness condition. Like JEMs (Murphy 2022), counterfactual training can be considered as a form
 193 of contrastive representation learning.

194 3.1.2 Indirectly Inducing Explainability through Adversarial Robustness

195 Based on our analysis in Section 2, counterfactuals \mathbf{x}' can be repurposed as additional training samples (Luu and Inoue
 196 2023) or adversarial examples (Friesleben 2022; Pawelczyk et al. 2022). This leaves some flexibility with respect to
 197 the exact choice for $\text{advloss}(\cdot)$ in Equation 1. An intuitive functional form to use, though likely not the only reasonable
 198 choice, is inspired by adversarial training:

$$\text{advloss}(\mathbf{M}_\theta(\mathbf{x}'), \mathbf{y}; \varepsilon) = \begin{cases} \text{yloss}(\mathbf{M}_\theta(\mathbf{x}'), \mathbf{y}) & \text{if } \|\Delta\|_\infty \leq \varepsilon \\ 0 & \text{otherwise.} \end{cases} \quad (3)$$

199 Under this choice we treat the counterfactual \mathbf{x}' as an adversarial example iff it is imperceptible, i.e. the magnitude of
 200 the perturbation of any individual feature is upper-bounded at ε .

201 3.2 Encoding Actionability Constraints

202 Many existing counterfactual explainers support domain and mutability constraints out-of-the-box. In fact, both types
 203 of constraints can be implemented for any counterfactual explainer that relies on gradient descent in the feature space
 204 for optimization (Altmeyer, Deursen, et al. 2023). In this context, domain constraints can be imposed by simply
 205 projecting counterfactuals back to the specified domain, if the previous gradient step resulted in updated feature values
 206 that were out-of-domain. Mutability constraints can similarly be enforced by setting partial derivatives to zero to
 207 ensure that features are only mutated in the allowed direction, if at all.

208 Since actionability constraints are binding at test time, we should also impose them when generating \mathbf{x}' during each
 209 training iteration to align model representations with user requirements. Through their effect on \mathbf{x}' , both types of
 210 constraints influence model outcomes through Equation 2. Here it is crucial that we avoid penalizing implausibility
 211 that arises due to mutability constraints. For any mutability-constrained feature d this can be achieved by enforcing
 212 $\mathbf{x}[d] - \mathbf{x}'[d] := 0$ whenever perturbing $\mathbf{x}'[d]$ in the direction of $\mathbf{x}[d]$ would violate mutability constraints. Specifically,
 213 we set $\mathbf{x}[d] := \mathbf{x}'[d]$ if

- 214 1. Feature d is strictly immutable in practice.
- 215 2. We have $\mathbf{x}[d] > \mathbf{x}'[d]$ but feature d can only be decreased in practice.
- 216 3. We have $\mathbf{x}[d] < \mathbf{x}'[d]$ but feature d can only be increased in practice.

217 From a Bayesian perspective, setting $\mathbf{x}[d] := \mathbf{x}'[d]$ can be understood as assuming a point mass prior for $p(\mathbf{x})$ with
 218 respect to feature d . Intuitively, we think of this simply in terms ignoring implausibility costs with respect to immutable
 219 features, which effectively forces the model to instead seek plausibility with respect to the remaining features. This

220 in turn results in lower overall sensitivity to immutable features, which we demonstrate empirically for different
 221 classifiers in Section 4. Under certain conditions, this results holds theoretically²:

222 **Theorem 3.1** (Protecting Immutable Features). *Let $M_\theta(x) = \Theta x$ denote a linear classifier with $y \in \{1, \dots, K\} = \mathcal{K}$
 223 and $x \in \mathbb{R}^D$. If we assume multivariate Gaussian class densities with common diagonal covariance matrix $\Sigma_k = \Sigma$
 224 for all $k \in \mathcal{K}$, then protecting an immutable feature from the contrastive divergence penalty Equation 2 will result in
 225 lower classifier sensitivity to that feature, provided that at least one other feature is mutable.*

226 It is worth highlighting that Theorem 3.1 assumes independence of features. This raises a valid concern about the
 227 effect of protecting immutable features in the presence of proxy features that remain unprotected. We discuss this
 228 limitation in Section 5.

229 4 Experiments

230 4.1 Experimental Setup

231 4.2 Experimental Results

232 5 Discussion

- 233 1. Limited to classification models.
 234 2. Proxy attributes of immutable features.

235 6 Conclusion

236 References

- 237 Abbasnejad, Ehsan, Damien Teney, Amin Parvaneh, Javen Shi, and Anton van den Hengel. 2020. “Counterfactual
 238 Vision and Language Learning.” In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition
 239 (CVPR)*, 10041–51. <https://doi.org/10.1109/CVPR42600.2020.01006>.
- 240 Altmeyer, Patrick, Arie van Deursen, et al. 2023. “Explaining Black-Box Models Through Counterfactuals.” In
 241 *Proceedings of the JuliaCon Conferences*, 1:130. 1.
- 242 Altmeyer, Patrick, Mojtaba Farmanbar, Arie van Deursen, and Cynthia CS Liem. 2024. “Faithful Model Explanations
 243 Through Energy-Constrained Conformal Counterfactuals.” In *Proceedings of the AAAI Conference on Artificial
 244 Intelligence*, 38:10829–37. 10.
- 245 Augustin, Maximilian, Alexander Meinke, and Matthias Hein. 2020. “Adversarial Robustness on in-and Out-
 246 Distribution Improves Explainability.” In *European Conference on Computer Vision*, 228–45. Springer.
- 247 Du, Yilun, and Igor Mordatch. 2020. “Implicit Generation and Generalization in Energy-Based Models.” <https://arxiv.org/abs/1903.08689>.
- 248 Freiesleben, Timo. 2022. “The Intriguing Relation Between Counterfactual Explanations and Adversarial Examples.”
Minds and Machines 32 (1): 77–109.
- 249 Goodfellow, Ian J, Jonathon Shlens, and Christian Szegedy. 2014. “Explaining and Harnessing Adversarial Examples.”
<https://arxiv.org/abs/1412.6572>.
- 250 Goodfellow, Ian, Yoshua Bengio, and Aaron Courville. 2016. *Deep Learning*. MIT Press.
- 251 Grathwohl, Will, Kuan-Chieh Wang, Joern-Henrik Jacobsen, David Duvenaud, Mohammad Norouzi, and Kevin Swer-
 252 sky. 2020. “Your Classifier Is Secretly an Energy Based Model and You Should Treat It Like One.” In *International
 253 Conference on Learning Representations*.
- 254 Guidotti, Riccardo. 2022. “Counterfactual Explanations and How to Find Them: Literature Review and Benchmark-
 255 ing.” *Data Mining and Knowledge Discovery*, 1–55.
- 256 Guo, Hangzhi, Thanh H. Nguyen, and Amulya Yadav. 2023. “CounterNet: End-to-End Training of Prediction Aware
 257 Counterfactual Explanations.” In *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery
 258 and Data Mining*, 577–89. KDD ’23. New York, NY, USA: Association for Computing Machinery. <https://doi.org/10.1145/3580305.3599290>.
- 259 Hastie, Trevor, Robert Tibshirani, and Jerome Friedman. 2009. *The Elements of Statistical Learning*. Springer New
 260 York. <https://doi.org/10.1007/978-0-387-84858-7>.
- 261 Kolter, Zico. 2023. “Keynote Addresses: SaTML 2023 .” In *2023 IEEE Conference on Secure and Trustworthy
 262 Machine Learning (SaTML)*, xvi–. Los Alamitos, CA, USA: IEEE Computer Society. [https://doi.org/10.1109/Sa-TML54575.2023.00009](https://doi.org/10.1109/Sa-

 263 TML54575.2023.00009).
- 264 Lakshminarayanan, Balaji, Alexander Pritzel, and Charles Blundell. 2017. “Simple and Scalable Predictive Uncer-
 265 tainty Estimation Using Deep Ensembles.” *Advances in Neural Information Processing Systems* 30.

²For the proof, see the supplementary appendix.

- 270 Lippe, Phillip. 2024. “UvA Deep Learning Tutorials.” <https://uvadlc-notebooks.readthedocs.io/en/latest/>.
- 271 Luu, Hoai Linh, and Naoya Inoue. 2023. “Counterfactual Adversarial Training for Improving Robustness of Pre-
272 Trained Language Models.” In *Proceedings of the 37th Pacific Asia Conference on Language, Information and
273 Computation*, 881–88.
- 274 Murphy, Kevin P. 2022. *Probabilistic Machine Learning: An Introduction*. MIT Press.
- 275 O’Neil, Cathy. 2016. *Weapons of Math Destruction: How Big Data Increases Inequality and Threatens Democracy*.
276 Crown.
- 277 Pawelczyk, Martin, Chirag Agarwal, Shalmali Joshi, Sohini Upadhyay, and Himabindu Lakkaraju. 2022. “Exploring
278 Counterfactual Explanations Through the Lens of Adversarial Examples: A Theoretical and Empirical Analysis.”
279 In *Proceedings of the 25th International Conference on Artificial Intelligence and Statistics*, edited by Gustau
280 Camps-Valls, Francisco J. R. Ruiz, and Isabel Valera, 151:4574–94. Proceedings of Machine Learning Research.
281 PMLR. <https://proceedings.mlr.press/v151/pawelczyk22a.html>.
- 282 Poyiadzi, Rafael, Kacper Sokol, Raul Santos-Rodriguez, Tijl De Bie, and Peter Flach. 2020. “FACE: Feasible and
283 Actionable Counterfactual Explanations.” In *Proceedings of the AAAI/ACM Conference on AI, Ethics, and Society*,
284 344–50.
- 285 Ross, Alexis, Himabindu Lakkaraju, and Osbert Bastani. 2024. “Learning Models for Actionable Recourse.” In
286 *Proceedings of the 35th International Conference on Neural Information Processing Systems*. NIPS ’21. Red
287 Hook, NY, USA: Curran Associates Inc.
- 288 Sauer, Axel, and Andreas Geiger. 2021. “Counterfactual Generative Networks.” <https://arxiv.org/abs/2101.06046>.
- 289 Schut, Lisa, Oscar Key, Rory Mc Grath, Luca Costabello, Bogdan Sacaleanu, Yarin Gal, et al. 2021. “Generating
290 Interpretable Counterfactual Explanations By Implicit Minimisation of Epistemic and Aleatoric Uncertainties.” In
291 *International Conference on Artificial Intelligence and Statistics*, 1756–64. PMLR.
- 292 Szegedy, Christian, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus.
293 2013. “Intriguing Properties of Neural Networks.” <https://arxiv.org/abs/1312.6199>.
- 294 Teney, Damien, Ehsan Abbasnedjad, and Anton van den Hengel. 2020. “Learning What Makes a Difference from
295 Counterfactual Examples and Gradient Supervision.” In *Computer Vision–ECCV 2020: 16th European Confer-
296 ence, Glasgow, UK, August 23–28, 2020, Proceedings, Part x 16*, 580–99. Springer.
- 297 Wachter, Sandra, Brent Mittelstadt, and Chris Russell. 2017. “Counterfactual Explanations Without Opening the Black
298 Box: Automated Decisions and the GDPR.” *Harv. JL & Tech.* 31: 841. <https://doi.org/10.2139/ssrn.3063289>.
- 299 Wilson, Andrew Gordon. 2020. “The Case for Bayesian Deep Learning.” <https://arxiv.org/abs/2001.10995>.
- 300 Wu, Tongshuang, Marco Tulio Ribeiro, Jeffrey Heer, and Daniel Weld. 2021. “Polyjuice: Generating Counterfactuals
301 for Explaining, Evaluating, and Improving Models.” In *Proceedings of the 59th Annual Meeting of the Associa-
302 tion for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing
303 (Volume 1: Long Papers)*, edited by Chengqing Zong, Fei Xia, Wenjie Li, and Roberto Navigli, 6707–23. Online:
304 Association for Computational Linguistics. <https://doi.org/10.18653/v1/2021.acl-long.523>.

305 **A Notation**

- 306 • y^+ : The target class and also the index of the target class.
- 307 • y^- : The non-target class and also the index of non-the target class.
- 308 • \mathbf{y}^+ : The one-hot encoded output vector for the target class.
- 309 • θ : Model parameters (unspecified).
- 310 • Θ : Matrix of parameters.

311 **B Technical Details of Our Approach**

312 **B.1 Protecting Mutability Constraints with Linear Classifiers**

313 In Section 3.2 we explain that to avoid penalizing implausibility that arises due to mutability constraints, we impose a
 314 point mass prior on $p(\mathbf{x})$ for the corresponding feature. We argue in Section 3.2 that this approach induces models to
 315 be less sensitive to immutable features and demonstrate this empirically in Section 4. Below we derive the analytical
 316 results in Theorem 3.1.

317 *Proof.* Let d_{mtbl} and d_{immtbl} denote some mutable and immutable feature, respectively. Suppose that $\mu_{y^-, d_{\text{immtbl}}} <$
 318 $\mu_{y^+, d_{\text{immtbl}}}$ and $\mu_{y^-, d_{\text{mtbl}}} > \mu_{y^+, d_{\text{mtbl}}}$, where $\mu_{k,d}$ denotes the conditional sample mean of feature d in class k . In words,
 319 we assume that the immutable feature tends to take lower values for samples in the non-target class y^- than in the
 320 target class y^+ . We assume the opposite to hold for the mutable feature.

321 Assuming multivariate Gaussian class densities with common diagonal covariance matrix $\Sigma_k = \Sigma$ for all $k \in \mathcal{K}$, we
 322 have for the log likelihood ratio between any two classes $k, m \in \mathcal{K}$ (Hastie, Tibshirani, and Friedman 2009):

$$\log \frac{p(k|\mathbf{x})}{p(m|\mathbf{x})} = \mathbf{x}^\top \Sigma^{-1} (\mu_k - \mu_m) + \text{const} \quad (4)$$

323 By independence of x_1, \dots, x_D , the full log-likelihood ratio decomposes into:

$$\log \frac{p(k|\mathbf{x})}{p(m|\mathbf{x})} = \sum_{d=1}^D \frac{\mu_{k,d} - \mu_{m,d}}{\sigma_d^2} x_d + \text{const} \quad (5)$$

324 By Equation 5 we have the $\theta_{y^-, d_{\text{immtbl}}} - \theta_{y^+, d_{\text{immtbl}}} < 0$ and $\theta_{y^-, d_{\text{mtbl}}} - \theta_{y^+, d_{\text{mtbl}}} > 0$, where $\theta_{k,d} = \Theta[k, d]$ denotes the
 325 coefficient on feature d for class k .

326 Let \mathbf{x}' denote some randomly chosen individual from class y^- and let $y^+ \sim p(y)$ denote the randomly chosen target
 327 class. Then the partial derivative of the contrastive divergence penalty Equation 2 with respect to coefficient $\theta_{y^+, d}$ is
 328 equal to

$$\frac{\partial}{\partial \theta_{y^+, d}} (\text{div}(\mathbf{x}, \mathbf{x}', \mathbf{y}; \theta)) = \frac{\partial}{\partial \theta_{y^+, d}} ((-\mathbf{M}_\theta(\mathbf{x})[y^+]) - (-\mathbf{M}_\theta(\mathbf{x}')[y^+])) = x'_d - x_d \quad (6)$$

329 and equal to zero everywhere else.

330 Since $\mu_{y^-, d_{\text{immtbl}}} < \mu_{y^+, d_{\text{immtbl}}}$ we are more likely to have $x'_{d_{\text{immtbl}}} - x_{d_{\text{immtbl}}} < 0$ than vice versa at initialization. Similarly,
 331 we are more likely to have $x'_{d_{\text{mtbl}}} - x_{d_{\text{mtbl}}} > 0$ by $\mu_{y^-, d_{\text{mtbl}}} > \mu_{y^+, d_{\text{mtbl}}}$.

332 This implies that if we do not protect feature d_{immtbl} , the contrastive divergence penalty will push down on $\theta_{y^-, d_{\text{immtbl}}}$
 333 thereby exacerbating the existing effect $\theta_{y^-, d_{\text{immtbl}}} - \theta_{y^+, d_{\text{immtbl}}} < 0$. In words, not protecting the immutable feature
 334 would have the undesirable effect of making the classifier more sensitive to this feature, in that it would be more likely
 335 to predict class y^- as opposed to y^+ for lower values of d_{immtbl} .

336 By the same rationale, the contrastive divergence penalty can generally be expected to push up on $\theta_{y^-, d_{\text{mtbl}}}$ exacerbating
 337 $\theta_{y^-, d_{\text{mtbl}}} - \theta_{y^+, d_{\text{mtbl}}} > 0$. In words, this has the effect of making the classifier more sensitive to the mutable feature, in
 338 that it would be more likely to predict class y^- as opposed to y^+ for higher values of d_{mtbl} .

339 Thus, our proposed approach of protecting feature d_{immtbl} has the net affect of decreasing the classifier's sensitivity
 340 to the immutable feature relative to the mutable feature (i.e. no change in sensitivity for d_{immtbl} relative to increased
 341 sensitivity for d_{mtbl}). \square

342 We provide an illustrative example in Example B.1.

343 **Example B.1** (Prediction of Consumer Credit Default). Suppose now that d_{immutb} represents an individual's age and
 344 d_{mtbl} represents an individual's existing level of credit card debt. Assume that these two features are independent, the
 345 class conditional densities are Gaussian and we use no other features to predict the risk of individuals defaulting on a
 346 consumer loan using a linear classifier. We have simulated this scenario using synthetic data in Figure A1.

347 In panel (a) of Figure A1, we have trained the linear classifier using counterfactual training, treating both features as
 348 mutable. The linear decision boundary is roughly equally sensitive to both age and existing levels of debt.

349 Conversely, in panel (b) of Figure A1, we have trained the same classifier using counterfactual training, but this time
 350 treating age as an immutable feature. The result is a new decision boundary that has tilted in favor of higher sensitivity
 351 to the mutable feature (existing debt) and lower sensitivity to the immutable feature (age).

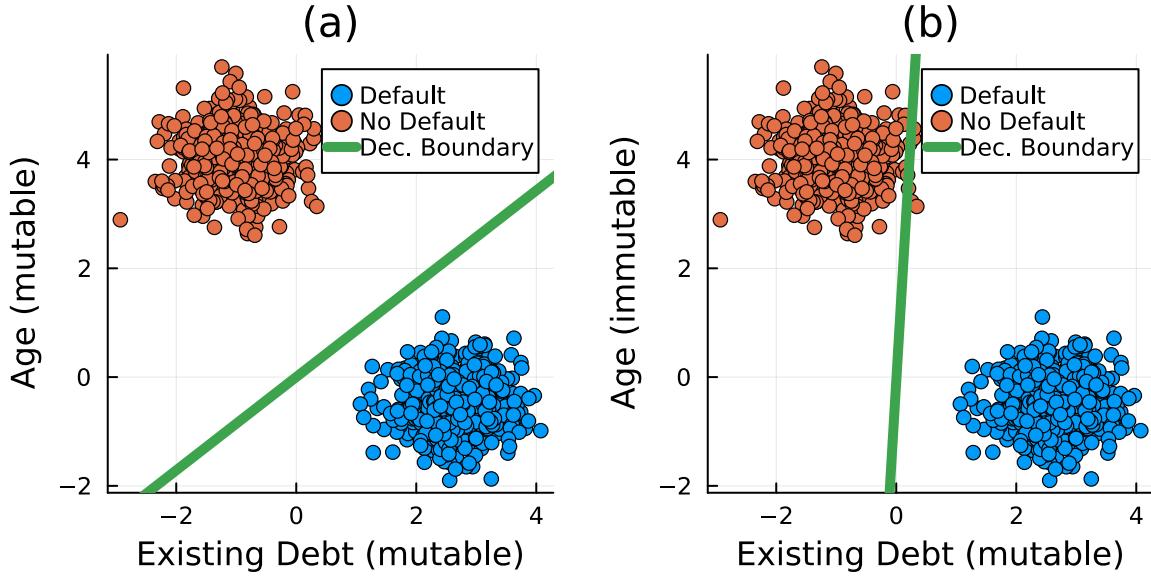


Figure A1: Visual illustration of the effect of imposing mutability constraints. See Example B.1 for details.

⚠ Warning

@Cynthia, @Arie, I have tentatively phrased the above in terms of a theorem and proof. This is something I've so far shied away from because I feel a bit out of my depth when it comes to mathematical proofs. The above makes intuitive sense to me, but I don't know for sure if it's correct.

352

353 C Detailed Results

354 C.1 Initial Grid Search

355 For the initial round of experiments we

356 C.1.1 Generator Parameters

357 The hyperparameter grids for the first investigation of the effect of generator parameters are shown in Parameters C.1
 358 and Parameters C.2.

359 **Parameters C.1** (Training Phase).

- 360 • Generator Parameters:

- 361 – λ_{cost} : 0.0, 0.001, 0.1
- 362 – λ_{div} : 0.01, 0.05, 0.1, 0.5, 1.0, 5.0, 10.0, 15.0
- 363 – Learning Rate: 1.0

- 364 – Maximum Iterations: 20, 50, 100
 365 – Optimizerimizer: sgd
 366 • Generator: ecco, generic, omni, revise
 367 • Training Parameters:
 368 – Objective: full, vanilla

369 **Parameters C.2** (Evaluation Phase).

- 370 • Counterfactual Parameters:
 371 – Convergence: max_iter
 372 – Maximum Iterations: 100
 373 – No. Individuals: 100
 374 – No. Runs: 5
 375 • Generator Parameters:
 376 – λ_{cost} : 0.0
 377 – λ_{div} : 0.1, 0.5, 1.0, 5.0, 10.0, 20.0
 378 – Learning Rate: 1.0
 379 – Maximum Iterations: 50
 380 – Optimizerimizer: sgd

381 **C.1.1.1 Linearly Separable**

- 382 • **Energy Penalty** (Table A1): *ECCo* generally does yield better results than *Vanilla* for higher choices of the
 383 energy penalty (10,15) during training. *Generic* performs poorly across the board. *Omni* seems to have an
 384 anchoring effect, in that it never performs terribly but also never as good as the best *ECCo* results. *REVISE*
 385 performs poorly across the board.
 386 • **Cost** (Table A2): Results for all generators (except *Omni*) are quite bad, which can likely be attributed to
 387 extremely bad results for some choices of the **Energy Penalty** (results here are averaged). For *ECCo* and
 388 *Generic*, higher cost values generally lead to worse results.
 389 • **Maximum Iterations**: No clear patterns recognizable, so it seems that smaller choices are ok.
 390 • **Validity**: *ECCo* almost always valid except for very low values during training and high values at evaluation
 391 time. *Generic* often has poor validity.
 392 • **Accuracy**: Seems largely unaffected.

Table A1: Results for Linearly Separable data by energy penalty.

Objective	$\lambda_{\text{div}}(\text{train})$	Generator	Value	Std
full	0.01	<i>ECCo</i>	$-9.91 \cdot 10^{11}$	$2.25 \cdot 10^{12}$
full	0.01	<i>Generic</i>	$-5.71 \cdot 10^{17}$	$1.3 \cdot 10^{18}$
full	0.01	Omniscient	-2.54	0.116
full	0.01	<i>REVISE</i>	-15.6	13.2
vanilla	0.01	<i>ECCo</i>	-4.28	3.52
vanilla	0.01	<i>Generic</i>	-4.45	3.47
vanilla	0.01	<i>Omniscient</i>	-5.12	4.46
vanilla	0.01	<i>REVISE</i>	-4.91	4.24
full	0.05	<i>ECCo</i>	$-5.63 \cdot 10^5$	$1.28 \cdot 10^6$
full	0.05	<i>Generic</i>	$-8.35 \cdot 10^{17}$	$1.9 \cdot 10^{18}$
full	0.05	Omniscient	-2.53	0.114
full	0.05	<i>REVISE</i>	-15	12.6
vanilla	0.05	<i>ECCo</i>	-4.4	3.66
vanilla	0.05	<i>Generic</i>	-4.38	3.48
vanilla	0.05	<i>Omniscient</i>	-5.25	4.62
vanilla	0.05	<i>REVISE</i>	-4.94	4.22
full	0.1	<i>ECCo</i>	$-6.74 \cdot 10^5$	$1.53 \cdot 10^6$
full	0.1	<i>Generic</i>	$-1.72 \cdot 10^{11}$	$3.9 \cdot 10^{11}$

Continuing table below.

Objective	$\lambda_{\text{div}}(\text{train})$	Generator	Value	Std
full	0.1	Omniscient	-2.56	0.124
full	0.1	<i>REVISE</i>	-15.6	13.2
vanilla	0.1	<i>ECCo</i>	-4.28	3.52
vanilla	0.1	<i>Generic</i>	-4.45	3.48
vanilla	0.1	<i>Omniscient</i>	-5.12	4.46
vanilla	0.1	<i>REVISE</i>	-4.91	4.25
full	0.5	<i>ECCo</i>	-11.8	9.83
full	0.5	<i>Generic</i>	$-1.06 \cdot 10^{18}$	$2.42 \cdot 10^{18}$
full	0.5	Omniscient	-2.54	0.123
full	0.5	<i>REVISE</i>	-15	12.6
vanilla	0.5	<i>ECCo</i>	-4.4	3.65
vanilla	0.5	<i>Generic</i>	-4.38	3.48
vanilla	0.5	<i>Omniscient</i>	-5.25	4.61
vanilla	0.5	<i>REVISE</i>	-4.95	4.22
full	1	<i>ECCo</i>	-11.5	11.1
full	1	<i>Generic</i>	$-1.71 \cdot 10^{11}$	$3.88 \cdot 10^{11}$
full	1	Omniscient	-2.59	0.117
full	1	<i>REVISE</i>	-15.7	13.3
vanilla	1	<i>ECCo</i>	-4.28	3.51
vanilla	1	<i>Generic</i>	-4.44	3.47
vanilla	1	<i>Omniscient</i>	-5.11	4.46
vanilla	1	<i>REVISE</i>	-4.91	4.25
full	5	<i>ECCo</i>	-3.99	3.12
full	5	<i>Generic</i>	$-4.88 \cdot 10^{17}$	$1.11 \cdot 10^{18}$
full	5	Omniscient	-2.53	0.117
full	5	<i>REVISE</i>	-14.6	12.1
vanilla	5	<i>ECCo</i>	-4.4	3.65
vanilla	5	<i>Generic</i>	-4.38	3.48
vanilla	5	<i>Omniscient</i>	-5.25	4.61
vanilla	5	<i>REVISE</i>	-4.95	4.22
full	10	ECCo	-2.31	0.735
full	10	<i>Generic</i>	$-1.7 \cdot 10^{11}$	$3.86 \cdot 10^{11}$
full	10	<i>Omniscient</i>	-2.53	0.117
full	10	<i>REVISE</i>	-15.5	13
vanilla	10	<i>ECCo</i>	-4.28	3.51
vanilla	10	<i>Generic</i>	-4.44	3.47
vanilla	10	<i>Omniscient</i>	-5.12	4.46
vanilla	10	<i>REVISE</i>	-4.91	4.24
full	15	ECCo	-2.01	0.488
full	15	<i>Generic</i>	$-4.91 \cdot 10^{17}$	$1.12 \cdot 10^{18}$
full	15	<i>Omniscient</i>	-2.53	0.116
full	15	<i>REVISE</i>	-14.4	11.7
vanilla	15	<i>ECCo</i>	-4.4	3.65
vanilla	15	<i>Generic</i>	-4.38	3.48
vanilla	15	<i>Omniscient</i>	-5.25	4.6
vanilla	15	<i>REVISE</i>	-4.95	4.23

Table A2: Results for Linearly Separable data by cost penalty.

Objective	$\lambda_{\text{cost}}(\text{train})$	Generator	Value	Std
full	0	<i>ECCo</i>	$-5.32 \cdot 10^3$	$1.21 \cdot 10^4$
full	0	<i>Generic</i>	$-1.03 \cdot 10^{18}$	$2.34 \cdot 10^{18}$
full	0	Omniscient	-2.64	0.125
full	0	<i>REVISE</i>	-15.4	12.9

Continuing table below.

Objective	$\lambda_{\text{cost}}(\text{train})$	Generator	Value	Std
vanilla	0	<i>ECCo</i>	-4.34	3.58
vanilla	0	<i>Generic</i>	-4.41	3.48
vanilla	0	<i>Omniscient</i>	-5.18	4.54
vanilla	0	<i>REVISE</i>	-4.93	4.23
full	0.001	<i>ECCo</i>	-362	811
full	0.001	<i>Generic</i>	$-2.65 \cdot 10^{17}$	$6.03 \cdot 10^{17}$
full	0.001	Omniscient	-2.49	0.115
full	0.001	<i>REVISE</i>	-15.5	13
vanilla	0.001	<i>ECCo</i>	-4.34	3.58
vanilla	0.001	<i>Generic</i>	-4.41	3.48
vanilla	0.001	<i>Omniscient</i>	-5.18	4.53
vanilla	0.001	<i>REVISE</i>	-4.93	4.23
full	0.1	<i>ECCo</i>	$-3.72 \cdot 10^{11}$	$8.46 \cdot 10^{11}$
full	0.1	<i>Generic</i>	$-4.49 \cdot 10^{14}$	$1.02 \cdot 10^{15}$
full	0.1	Omniscient	-2.5	0.112
full	0.1	<i>REVISE</i>	-14.6	12.2
vanilla	0.1	<i>ECCo</i>	-4.34	3.58
vanilla	0.1	<i>Generic</i>	-4.41	3.48
vanilla	0.1	<i>Omniscient</i>	-5.18	4.54
vanilla	0.1	<i>REVISE</i>	-4.93	4.24

393 C.1.1.2 Moons

- 394 • **Energy Penalty** (Table A3): *ECCo* consistently yields better results than *Vanilla*, except for very low choices
395 of the energy penalty during training for which it performs abysmal. *Generic* performs quite badly across
396 the board for high enough choices of the energy penalty at evaluation time. *Omni* has small positive effect.
397 *REVISE* performs poorly across the board.
- 398 • **Cost (distance penalty)**: *Generic* generally does better for higher values, while *ECCo* does better for lower
399 values.
- 400 • **Maximum Iterations**: No clear patterns recognizable, so it seems that smaller choices are ok.
- 401 • **Validity**: *ECCo* generally achieves full validity except for very low choices the energy penalty during training
402 and high choices at evaluation time. *Generic* performs poorly for high choices of the energy penalty during
403 evaluation.
- 404 • **Accuracy**: Largely unaffected although *ECCo* suffers a bit for very low choices the energy penalty during
405 training. *REVISE* suffers a lot in general (around 10 percentage points).

Table A3: Results for Moons data by energy penalty.

Objective	$\lambda_{\text{div}}(\text{train})$	Generator	Value	Std
full	0.01	<i>ECCo</i>	$-2.8 \cdot 10^{22}$	$6.39 \cdot 10^{22}$
full	0.01	<i>Generic</i>	$-4.89 \cdot 10^{30}$	$1.11 \cdot 10^{31}$
full	0.01	Omniscient	-4.74	5.08
full	0.01	<i>REVISE</i>	-572	$1.25 \cdot 10^3$
vanilla	0.01	<i>ECCo</i>	-15.5	17.3
vanilla	0.01	<i>Generic</i>	-10.9	11.9
vanilla	0.01	<i>Omniscient</i>	-12.7	14.4
vanilla	0.01	<i>REVISE</i>	-11.2	13
full	0.05	<i>ECCo</i>	$-1.55 \cdot 10^{16}$	$3.52 \cdot 10^{16}$
full	0.05	<i>Generic</i>	$-2.22 \cdot 10^{20}$	$5 \cdot 10^{20}$
full	0.05	Omniscient	-4.41	4.48
full	0.05	<i>REVISE</i>	$-1.04 \cdot 10^3$	$2.3 \cdot 10^3$
vanilla	0.05	<i>ECCo</i>	-15.5	17.2
vanilla	0.05	<i>Generic</i>	-11.7	12.8
vanilla	0.05	<i>Omniscient</i>	-12.4	14.1

Continuing table below.

Objective	$\lambda_{\text{div}}(\text{train})$	Generator	Value	Std
vanilla	0.05	<i>REVISE</i>	-11.3	13.1
full	0.1	<i>ECCo</i>	$-3.41 \cdot 10^3$	$7.73 \cdot 10^3$
full	0.1	<i>Generic</i>	$-5.22 \cdot 10^{30}$	$1.19 \cdot 10^{31}$
full	0.1	Omniscient	-4.78	5.12
full	0.1	<i>REVISE</i>	-288	594
vanilla	0.1	<i>ECCo</i>	-15.5	17.2
vanilla	0.1	<i>Generic</i>	-10.9	11.9
vanilla	0.1	<i>Omniscient</i>	-12.7	14.4
vanilla	0.1	<i>REVISE</i>	-11.3	13.1
full	0.5	<i>ECCo</i>	-7.09	7.51
full	0.5	<i>Generic</i>	$-1.11 \cdot 10^{31}$	$2.53 \cdot 10^{31}$
full	0.5	Omniscient	-4.58	4.83
full	0.5	<i>REVISE</i>	$-1.19 \cdot 10^3$	$2.64 \cdot 10^3$
vanilla	0.5	<i>ECCo</i>	-15.5	17.2
vanilla	0.5	<i>Generic</i>	-11.7	12.8
vanilla	0.5	<i>Omniscient</i>	-12.4	14.1
vanilla	0.5	<i>REVISE</i>	-11.3	13.1
full	1	<i>ECCo</i>	-6.06	6.33
full	1	<i>Generic</i>	$-1.58 \cdot 10^{33}$	$3.59 \cdot 10^{33}$
full	1	Omniscient	-4.66	4.89
full	1	<i>REVISE</i>	$-1.16 \cdot 10^3$	$2.59 \cdot 10^3$
vanilla	1	<i>ECCo</i>	-15.5	17.3
vanilla	1	<i>Generic</i>	-10.9	11.9
vanilla	1	<i>Omniscient</i>	-12.7	14.4
vanilla	1	<i>REVISE</i>	-11.3	13.1
full	5	ECCo	-2.57	2.07
full	5	<i>Generic</i>	$-1.17 \cdot 10^{28}$	$2.66 \cdot 10^{28}$
full	5	<i>Omniscient</i>	-4.29	4.31
full	5	<i>REVISE</i>	-530	$1.16 \cdot 10^3$
vanilla	5	<i>ECCo</i>	-15.5	17.2
vanilla	5	<i>Generic</i>	-11.7	12.7
vanilla	5	<i>Omniscient</i>	-12.4	14.1
vanilla	5	<i>REVISE</i>	-11.3	13.1
full	10	ECCo	-1.76	0.974
full	10	<i>Generic</i>	$-1.54 \cdot 10^{33}$	$3.51 \cdot 10^{33}$
full	10	<i>Omniscient</i>	-4.44	4.56
full	10	<i>REVISE</i>	$-1.52 \cdot 10^3$	$3.4 \cdot 10^3$
vanilla	10	<i>ECCo</i>	-15.5	17.3
vanilla	10	<i>Generic</i>	-10.9	11.9
vanilla	10	<i>Omniscient</i>	-12.7	14.4
vanilla	10	<i>REVISE</i>	-11.3	13.1
full	15	ECCo	-1.37	0.365
full	15	<i>Generic</i>	$-5.32 \cdot 10^{28}$	$1.21 \cdot 10^{29}$
full	15	<i>Omniscient</i>	-4.34	4.38
full	15	<i>REVISE</i>	-473	$1.03 \cdot 10^3$
vanilla	15	<i>ECCo</i>	-15.5	17.2
vanilla	15	<i>Generic</i>	-11.7	12.8
vanilla	15	<i>Omniscient</i>	-12.4	14.1
vanilla	15	<i>REVISE</i>	-11.3	13.1

406 C.1.1.3 Circles

- 407 • **Energy Penalty** (Table A4): *ECCo* consistently yields better results than *Vanilla*, though primarily for low to
408 medium choices of the energy penalty ($<= 5$) during training. The same goes for *Generic*, which sometimes
409 outperforms *ECCo* (for small energy penalty at evaluation time). *Omni* does alright for lower energy penalty

410 at evaluation time, but loses out for higher choices. *REVISE* performs poorly across the board (except very
 411 low choices at evaluation time).

- 412 • **Cost (distance penalty):** *ECCo* and *Generic* generally achieve the best results when no cost penalty is used
 413 during training. Both *Omni* and *REVISE* are largely unaffected.
- 414 • **Maximum Iterations:** *ECCo* consistently yields better results for higher numbers of iterations. *Generic*
 415 generally does best for a medium number (50). *Omni* is sometimes invalid (???).
- 416 • **Validity:** *ECCo* tends to outperform its *Vanilla* counterpart, though primarily for low to medium choices of
 417 the energy penalty (<=5) during training and evaluation. *Vanilla* typically worse across the board.
- 418 • **Accuracy:** Mostly unaffected, but *REVISE* again consistently some deterioration and *ECCo* deteriorates for
 419 high choices of energy penalty during training, reflecting other outcomes above.

Table A4: Results for Circles data by energy penalty.

Objective	$\lambda_{\text{div}}(\text{train})$	Generator	Value	Std
full	0.01	ECCo	-1.26	0.423
full	0.01	<i>Generic</i>	-1.49	0.71
full	0.01	<i>Omniscient</i>	-5.21	5.25
full	0.01	<i>REVISE</i>	$-2.71 \cdot 10^{26}$	$6.37 \cdot 10^{26}$
vanilla	0.01	<i>ECCo</i>	-9.33	7.34
vanilla	0.01	<i>Generic</i>	-8.89	6.88
vanilla	0.01	<i>Omniscient</i>	-8.67	6.87
vanilla	0.01	<i>REVISE</i>	-8.65	6.8
full	0.05	<i>ECCo</i>	-1.29	0.397
full	0.05	Generic	-1.21	0.356
full	0.05	<i>Omniscient</i>	-5.08	5.09
full	0.05	<i>REVISE</i>	$-5.91 \cdot 10^{27}$	$1.36 \cdot 10^{28}$
vanilla	0.05	<i>ECCo</i>	-9.35	7.32
vanilla	0.05	<i>Generic</i>	-8.85	6.87
vanilla	0.05	<i>Omniscient</i>	-8.7	6.96
vanilla	0.05	<i>REVISE</i>	-8.52	6.76
full	0.1	ECCo	-1.2	0.383
full	0.1	<i>Generic</i>	-1.5	0.735
full	0.1	<i>Omniscient</i>	-5.17	5.23
full	0.1	<i>REVISE</i>	$-3.06 \cdot 10^{26}$	$7.7 \cdot 10^{26}$
vanilla	0.1	<i>ECCo</i>	-9.33	7.32
vanilla	0.1	<i>Generic</i>	-8.88	6.86
vanilla	0.1	<i>Omniscient</i>	-8.69	6.9
vanilla	0.1	<i>REVISE</i>	-8.68	6.81
full	0.5	ECCo	-1.12	0.217
full	0.5	<i>Generic</i>	-1.21	0.352
full	0.5	<i>Omniscient</i>	-5.09	5.12
full	0.5	<i>REVISE</i>	$-5.97 \cdot 10^{27}$	$1.37 \cdot 10^{28}$
vanilla	0.5	<i>ECCo</i>	-9.35	7.3
vanilla	0.5	<i>Generic</i>	-8.89	6.92
vanilla	0.5	<i>Omniscient</i>	-8.68	6.93
vanilla	0.5	<i>REVISE</i>	-8.53	6.75
full	1	ECCo	-1.1	0.163
full	1	<i>Generic</i>	-1.49	0.726
full	1	<i>Omniscient</i>	-5.16	5.2
full	1	<i>REVISE</i>	$-3.09 \cdot 10^{26}$	$7.22 \cdot 10^{26}$
vanilla	1	<i>ECCo</i>	-9.34	7.36
vanilla	1	<i>Generic</i>	-8.86	6.85
vanilla	1	<i>Omniscient</i>	-8.7	6.9
vanilla	1	<i>REVISE</i>	-8.69	6.85
full	5	<i>ECCo</i>	-1.75	0.154
full	5	Generic	-1.21	0.363

Continuing table below.

Objective	$\lambda_{\text{div}}(\text{train})$	Generator	Value	Std
full	5	<i>Omniscient</i>	-5.14	5.16
full	5	<i>REVISE</i>	$-1.1 \cdot 10^{28}$	$2.5 \cdot 10^{28}$
vanilla	5	<i>ECCo</i>	-9.36	7.32
vanilla	5	<i>Generic</i>	-8.88	6.91
vanilla	5	<i>Omniscient</i>	-8.7	6.93
vanilla	5	<i>REVISE</i>	-8.52	6.73
full	10	<i>ECCo</i>	$-1.02 \cdot 10^6$	$2.32 \cdot 10^6$
full	10	Generic	-1.49	0.702
full	10	<i>Omniscient</i>	-5.13	5.16
full	10	<i>REVISE</i>	$-3.74 \cdot 10^{26}$	$9.09 \cdot 10^{26}$
vanilla	10	<i>ECCo</i>	-9.31	7.33
vanilla	10	<i>Generic</i>	-8.87	6.86
vanilla	10	<i>Omniscient</i>	-8.7	6.89
vanilla	10	<i>REVISE</i>	-8.69	6.83
full	15	<i>ECCo</i>	$-3.31 \cdot 10^{13}$	$7.54 \cdot 10^{13}$
full	15	Generic	-1.22	0.37
full	15	<i>Omniscient</i>	-5.2	5.23
full	15	<i>REVISE</i>	$-9.01 \cdot 10^{27}$	$2.06 \cdot 10^{28}$
vanilla	15	<i>ECCo</i>	-9.38	7.34
vanilla	15	<i>Generic</i>	-8.86	6.87
vanilla	15	<i>Omniscient</i>	-8.69	6.96
vanilla	15	<i>REVISE</i>	-8.51	6.73