
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 3, 2025

ABSTRACT

Counterfactual Explanations (CE) 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 desirable with respect to the underlying data and stakeholder requirements. Recent work has shown that under this premise, the task of learning desirable 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 desirable explanations. Through extensive experiments, we demonstrate that our proposed methodology facilitates training models that inherently deliver desirable 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 AIs 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 AIs 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, I. J. Goodfellow, Shlens, and Szegedy ([2014](#)) showed that artificial neural

23 networks (ANN) are sensitive to adversarial examples (AE): counterfactuals of model inputs that yield vastly different
 24 model predictions despite being semantically indifferent from their factual counterparts. Despite partially effective
 25 mitigation strategies such as **adversarial training**, truly robust deep learning (DL) remains unattainable even for
 26 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](#)). 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 undesirable 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 involving opaque models ([O'Neil 2016](#)). Effective XAI tools can aide us in monitoring models and providing recourse
 39 to affected individuals ([Wachter, Mittelstadt, and Russell 2017](#)).

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

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

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

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

60 2 Related Literature

61 2.1 Background on Counterfactual Explanations

62 ([Wachter, Mittelstadt, and Russell 2017](#); [Joshi et al. 2019](#); [Altmeyer et al. 2024](#))

63 2.2 Learning Representations

64 For example, joint-energy models

65 2.3 Generalization and Robustness

66 Sauer and Geiger ([2021](#)) generate counterfactual images for MNIST and ImageNet through independent mechanisms
 67 (IM): each IM learns class-conditional input distributions over a specific lower-dimensional, semantically meaningful
 68 factor, such as *texture*, *shape* and *background*. They demonstrate that using these generated counterfactuals during
 69 classifier training improves model robustness. Similarly, Abbasnejad et al. ([2020](#)) argue that counterfactuals represent
 70 potentially useful training data in machine learning, especially in supervised settings where inputs may be reasonably
 71 mapped to multiple outputs. They, too, demonstrate the augmenting the training data of image classifiers can improve
 72 generalization.

73 Tenev, Abbasnejad, and Hengel (2020) propose an approach using counterfactuals in training that does not rely on
 74 data augmentation: they argue that counterfactual pairs typically already exist in training datasets. Specifically, their
 75 approach relies on, firstly, identifying similar input samples with different annotations and, secondly, ensuring that the
 76 gradient of the classifier aligns with the vector between pairs of counterfactual inputs using the cosine distance as a loss
 77 function (referred to as *gradient supervision*) (*this might be useful for our task as well*). In the natural language pro-
 78 cessing (NLP) domain, counterfactuals have similarly been used to improve models through data augmentation: Wu et
 79 al. (2021), propose POLYJUICE, a general-purpose counterfactual generator for language models. They demonstrate
 80 empirically that augmenting training data through POLYJUICE counterfactuals improves robustness in a number of
 81 NLP tasks.

82 **2.4 Link to Adversarial Training**

83 From this perspective, adversarial training induces models to “unlearn” representations that are susceptible to the
 84 semantically most meaningless explanations—adversarial examples.

85 Freiesleben (2022) propose two definitional differences between Adversarial Examples (AE) and Counterfactual Ex-
 86 planations (CE): firstly, and more importantly according to the authors, the term AE implies missclassification, which
 87 is not the case for CE (*this might be a useful notion for use to distinguish between adversarials and explanations*
 88 *during training*); secondly, they argue that closeness plays a more critical role in the context of CE but confess that
 89 even counterfactuals that are not close might be relevant explanations. Pawelczyk et al. (2022) show that CE and AE
 90 are equivalent under certain conditions and derive upper bounds on the distances between them.

91 **2.5 Closely Related**

92 Guo, Nguyen, and Yadav (2023) are the first to propose end-to-end training pipeline that includes counterfactual ex-
 93 planations as part of the training procedure. In particular, they propose a specific network architecture that includes
 94 a predictor and CE generator network (*akin a GAN?*), where the parameters of the CE generator network are learn-
 95 able. Counterfactuals are generated during each training iteration and fed back to the predictor network (*here we are*
 96 *aligned*). In contrast, we impose no restrictions on the neural network architecture at all. (*to ensure the one-hot en-*
 97 *coding of categorical features is maintained, they simple use softmax (might be interesting for CE.jl)*) Interestingly,
 98 the authors find that their approach is sensitive to the choice of the loss function: only MSE seems to lead to good
 99 performance. They also demonstrate theoretically, that the objective function is difficult to optimize due to divergent
 100 gradients and suffers from poor adversarial robustness. (*because partial gradients with respect to the classification*
 101 *loss component and the counterfactual validity component point in opposite directions*). To mitigate these issues,
 102 the authors use block-wise gradient descent: they first update with respect to classification loss and then use a second
 103 update with respect to the other loss components (*this might be useful for our task as well*). Ross, Lakkaraju, and
 104 Bastani (2024) propose a way to train models that are guaranteed to provide recourse for individuals with high proba-
 105 bility. The approach builds on adversarial training (*here we are aligned*), where in this context adversarial examples
 106 are actively encouraged to exist, but only target attacks with respect to the positive class. The proposed method allows
 107 for imposing a set of actionable recourse ex-ante: for example, users can impose mutability constraints for features
 108 (*here we are aligned*). (*To solve their objective function more efficiently, they use a first-order Taylor approximation*
 109 *to approximate the recourse loss component (might be applicable in our case)*)

110 Luu and Inoue (2023) introduce Counterfactual Adversarial Training (CAT) with intention of improving generalization
 111 and robustness of language models. Specifically, they propose to proceed as follows: firstly, identify training samples
 112 that are subject to high predictive uncertainty (entropy); secondly, generate counterfactual explanations for those
 113 samples; and, finally, finetune the model on the augmented dataset that includes the generated counterfactuals.

114 **3 Counterfactual Training**

115 **4 Experiments**

116 **4.1 Experimental Setup**

117 **4.2 Experimental Results**

118 **5 Discussion**

119 **6 Conclusion**

120 **References**

121 Abbasnejad, Ehsan, Damien Teney, Amin Parvaneh, Javen Shi, and Anton van den Hengel. 2020. “Counterfactual
 122 Vision and Language Learning.” In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition*
 123 (*CVPR*), 10041–51. <https://doi.org/10.1109/CVPR42600.2020.01006>.

- 124 Altmeyer, Patrick, Mojtaba Farmanbar, Arie van Deursen, and Cynthia CS Liem. 2024. “Faithful Model Explanations
 125 Through Energy-Constrained Conformal Counterfactuals.” In *Proceedings of the AAAI Conference on Artificial*
 126 *Intelligence*, 38:10829–37. 10.
- 127 Augustin, Maximilian, Alexander Meinke, and Matthias Hein. 2020. “Adversarial Robustness on in-and Out-
 128 Distribution Improves Explainability.” In *European Conference on Computer Vision*, 228–45. Springer.
- 129 Freiesleben, Timo. 2022. “The Intriguing Relation Between Counterfactual Explanations and Adversarial Examples.”
 130 *Minds and Machines* 32 (1): 77–109.
- 131 Goodfellow, Ian J, Jonathon Shlens, and Christian Szegedy. 2014. “Explaining and Harnessing Adversarial Examples.”
 132 <https://arxiv.org/abs/1412.6572>.
- 133 Goodfellow, Ian, Yoshua Bengio, and Aaron Courville. 2016. *Deep Learning*. MIT Press.
- 134 Guo, Hangzhi, Thanh H. Nguyen, and Amulya Yadav. 2023. “CounterNet: End-to-End Training of Prediction Aware
 135 Counterfactual Explanations.” In *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery*
 136 *and Data Mining*, 577–89. KDD ’23. New York, NY, USA: Association for Computing Machinery. <https://doi.org/10.1145/3580305.3599290>.
- 138 Joshi, Shalmali, Oluwasanmi Koyejo, Warut Vigitbenjaronk, Been Kim, and Joydeep Ghosh. 2019. “Towards Realistic
 139 Individual Recourse and Actionable Explanations in Black-Box Decision Making Systems.” <https://arxiv.org/abs/1907.09615>.
- 141 Kolter, Zico. 2023. “Keynote Addresses: SaTML 2023 .” In *2023 IEEE Conference on Secure and Trustworthy*
 142 *Machine Learning (SaTML)*, xvi–. Los Alamitos, CA, USA: IEEE Computer Society. <https://doi.org/10.1109/SaTML54575.2023.00009>.
- 144 Luu, Hoai Linh, and Naoya Inoue. 2023. “Counterfactual Adversarial Training for Improving Robustness of Pre-
 145 Trained Language Models.” In *Proceedings of the 37th Pacific Asia Conference on Language, Information and*
 146 *Computation*, 881–88.
- 147 O’Neil, Cathy. 2016. *Weapons of Math Destruction: How Big Data Increases Inequality and Threatens Democracy*.
 148 Crown.
- 149 Pawelczyk, Martin, Chirag Agarwal, Shalmali Joshi, Sohini Upadhyay, and Himabindu Lakkaraju. 2022. “Exploring
 150 Counterfactual Explanations Through the Lens of Adversarial Examples: A Theoretical and Empirical Analysis.”
 151 In *Proceedings of the 25th International Conference on Artificial Intelligence and Statistics*, edited by Gustau
 152 Camps-Valls, Francisco J. R. Ruiz, and Isabel Valera, 151:4574–94. Proceedings of Machine Learning Research.
 153 PMLR. <https://proceedings.mlr.press/v151/pawelczyk22a.html>.
- 154 Ross, Alexis, Himabindu Lakkaraju, and Osbert Bastani. 2024. “Learning Models for Actionable Recourse.” In
 155 *Proceedings of the 35th International Conference on Neural Information Processing Systems*. NIPS ’21. Red
 156 Hook, NY, USA: Curran Associates Inc.
- 157 Sauer, Axel, and Andreas Geiger. 2021. “Counterfactual Generative Networks.” <https://arxiv.org/abs/2101.06046>.
- 158 Schut, Lisa, Oscar Key, Rory Mc Grath, Luca Costabello, Bogdan Sacaleanu, Yarin Gal, et al. 2021. “Generating
 159 Interpretable Counterfactual Explanations By Implicit Minimisation of Epistemic and Aleatoric Uncertainties.” In
 160 *International Conference on Artificial Intelligence and Statistics*, 1756–64. PMLR.
- 161 Teney, Damien, Ehsan Abbasnejad, and Anton van den Hengel. 2020. “Learning What Makes a Difference from
 162 Counterfactual Examples and Gradient Supervision.” In *Computer Vision–ECCV 2020: 16th European Conference*,
 163 *Glasgow, UK, August 23–28, 2020, Proceedings, Part x 16*, 580–99. Springer.
- 164 Wachter, Sandra, Brent Mittelstadt, and Chris Russell. 2017. “Counterfactual Explanations Without Opening the Black
 165 Box: Automated Decisions and the GDPR.” *Harv. JL & Tech.* 31: 841. <https://doi.org/10.2139/ssrn.3063289>.
- 166 Wilson, Andrew Gordon. 2020. “The Case for Bayesian Deep Learning.” <https://arxiv.org/abs/2001.10995>.
- 167 Wu, Tongshuang, Marco Tulio Ribeiro, Jeffrey Heer, and Daniel Weld. 2021. “Polyjuice: Generating Counterfactuals
 168 for Explaining, Evaluating, and Improving Models.” In *Proceedings of the 59th Annual Meeting of the Association*
 169 *for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing*
 170 *(Volume 1: Long Papers)*, edited by Chengqing Zong, Fei Xia, Wenjie Li, and Roberto Navigli, 6707–23. Online:
 171 Association for Computational Linguistics. <https://doi.org/10.18653/v1/2021.acl-long.523>.

172 **A Training Details**

173 **A.1 Initial Grid Search**

174 For the initial round of experiments we

175 **A.1.1 Generator Parameters**

176 The hyperparameter grids for the first investigation of the effect of generator parameters are shown in Parameters A.1
177 and Parameters A.2.

178 **Parameters A.1 (Training Phase).**

- 179 • Generator Parameters:
 - 180 – λ_{cost} : 0.0, 0.001, 0.1
 - 181 – λ_{div} : 0.01, 0.05, 0.1, 0.5, 1.0, 5.0, 10.0, 15.0
 - 182 – Learning Rate: 1.0
 - 183 – Maximum Iterations: 20, 50, 100
 - 184 – Optimizerimizer: sgd
- 185 • Generator: ecco, generic, omni, revise
- 186 • Training Parameters:
 - 187 – Objective: full, vanilla

188 **Parameters A.2 (Evaluation Phase).**

- 189 • Counterfactual Parameters:
 - 190 – Convergence: max_iter
 - 191 – Maximum Iterations: 100
 - 192 – No. Individuals: 100
 - 193 – No. Runs: 5
- 194 • Generator Parameters:
 - 195 – λ_{cost} : 0.0
 - 196 – λ_{div} : 0.1, 0.5, 1.0, 5.0, 10.0, 20.0
 - 197 – Learning Rate: 1.0
 - 198 – Maximum Iterations: 50
 - 199 – Optimizerimizer: sgd

200 **A.1.1.1 Linearly Separable**

- 201 • **Energy Penalty** (Table A1): *ECCo* generally does yield better results than *Vanilla* for higher choices of the energy penalty (10,15) during training. *Generic* performs poorly accross the board. *Omni* seems to have an anchoring effect, in that it never performs terribly but also never as good as the best *ECCo* results. *REVISE* performs poorly across the board.
- 202 • **Cost** (Table A2): Results for all generators (except *Omni*) are quite bad, which can likely be attributed to extremely bad results for some choices of the **Energy Penalty** (results here are averaged). For *ECCo* and *Generic*, higher cost values generally lead to worse results.
- 203 • **Maximum Iterations**: No clear patterns recognizable, so it seems that smaller choices are ok.
- 204 • **Validity**: *ECCo* almost always valid except for very low values during training and high values at evaluation time. *Generic* often has poor validity.
- 205 • **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

Continuing table below.

Objective	$\lambda_{\text{div}}(\text{train})$	Generator	Value	Std
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>	$-1.72 \cdot 10^{11}$	$3.9 \cdot 10^{11}$
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

Continuing table below.

Objective	$\lambda_{\text{div}}(\text{train})$	Generator	Value	Std
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
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

A.1.1.2 Moons

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

Continuing table below.

Objective	$\lambda_{\text{div}}(\text{train})$	Generator	Value	Std
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

225 A.1.1.3 Circles

- 226 • **Energy Penalty** (Table A4): *ECCo* consistently yields better results than *Vanilla*, though primarily for low to
 227 medium choices of the energy penalty ($<=5$) during training. The same goes for *Generic*, which sometimes
 228 outperforms *ECCo* (for small energy penalty at evaluation time). *Omni* does alright for lower energy penalty
 229 at evaluation time, but loses out for higher choices. *REVISE* performs poorly across the board (except very
 230 low choices at evaluation time).
- 231 • **Cost (distance penalty)**: *ECCo* and *Generic* generally achieve the best results when no cost penalty is used
 232 during training. Both *Omni* and *REVISE* are largely unaffected.
- 233 • **Maximum Iterations**: *ECCo* consistently yields better results for higher numbers of iterations. *Generic*
 234 generally does best for a medium number (50). *Omni* is sometimes invalid (???).
- 235 • **Validity**: *ECCo* tends to outperform its *Vanilla* counterpart, though primarily for low to medium choices of
 236 the energy penalty ($<=5$) during training and evaluation. *Vanilla* typically worse across the board.
- 237 • **Accuracy**: Mostly unaffected, but *REVISE* again consistently some deterioration and *ECCo* deteriorates for
 238 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

Continuing table below.

Objective	$\lambda_{\text{div}}(\text{train})$	Generator	Value	Std
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
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