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

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

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

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

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

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

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

## 2 Related Literature

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

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

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

### 2.1 Adversarial Examples are Counterfactual Explanations

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

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

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

### 2.2 Beyond Robustness

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

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

2020). This is closely related to the idea of aligning models with plausible explanations and has inspired our proposed counterfactual training objective, as we explain in Section 3.

# 3 Counterfactual Training

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

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

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

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

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

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

## 3.1 Our Proposed Objective

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

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

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

### 3.1.1 Directly Inducing Explainability through Contrastive Divergence

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

A key empirical finding in Altmeyer et al. (2024) was that JEMs tend to do well with respect to the plausibility objective in Definition 3.1. If we consider samples drawn from $p_\theta(\mathbf{x})$ as counterfactuals, this is an expected finding, because the JEM objective effectively minimizes the divergence between the conditional posterior and $p(\mathbf{x}|\mathbf{y}^+)$. To generate samples, Grathwohl et al. (2020) rely on Stochastic Gradient Langevin Dynamics (SGLD) using an uninformative prior for initialization. This is where we depart from their methodology: instead of generating samples through SGLD, we propose using counterfactual explainers to generate counterfactuals for observed training samples. Specifically, we have

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

where $\mathcal{E}_\theta(\cdot)$ denotes the energy function. We generate samples $\mathbf{x}'$ by first randomly sampling the target class $\mathbf{y}^+ \sim p(\mathbf{y})$ and then generating a counterfactual explanation for that target, similar to how conditional sampling can be used for JEMs (Grathwohl et al. 2020). Intuitively, the gradient of Equation 2 decreases the energy of observed training samples (positive samples) while at same time increasing the energy of counterfactuals (negative samples) (Du and Mordatch 2020). As the generated counterfactuals get more plausible (Definition 3.1) over the cause of training, these two opposing effects gradually balance each out (Lippe 2024).

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

#### 3.1.2 Indirectly Inducing Explainability through Adversarial Robustness

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

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

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

### 3.2 Encoding Actionability Constraints

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

Since both types of constraints directly affect $\mathbf{x}'$, they also influence model outcomes through Equation 2.

Mutability constraints

## 4 Experiments

### 4.1 Experimental Setup

### 4.2 Experimental Results

## 5 Discussion

## 6 Conclusion

## References

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

Altmeyer, Patrick, Arie van Deursen, et al. 2023. "Explaining Black-Box Models Through Counterfactuals." In *Proceedings of the JuliaCon Conferences*, 1:130. 1.

Altmeyer, Patrick, Mojtaba Farmanbar, Arie van Deursen, and Cynthia CS Liem. 2024. "Faithful Model Explanations Through Energy-Constrained Conformal Counterfactuals." In *Proceedings of the AAAI Conference on Artificial Intelligence*, 38:10829–37. 10.

Augustin, Maximilian, Alexander Meinke, and Matthias Hein. 2020. "Adversarial Robustness on in-and Out-Distribution Improves Explainability." In *European Conference on Computer Vision*, 228–45. Springer.

Du, Yilun, and Igor Mordatch. 2020. "Implicit Generation and Generalization in Energy-Based Models." https://arxiv.org/abs/1903.08689.

Freiesleben, Timo. 2022. "The Intriguing Relation Between Counterfactual Explanations and Adversarial Examples." *Minds and Machines* 32 (1): 77–109.

Goodfellow, Ian J, Jonathon Shlens, and Christian Szegedy. 2014. "Explaining and Harnessing Adversarial Examples." https://arxiv.org/abs/1412.6572.

Goodfellow, Ian, Yoshua Bengio, and Aaron Courville. 2016. *Deep Learning*. MIT Press.

Grathwohl, Will, Kuan-Chieh Wang, Joern-Henrik Jacobsen, David Duvenaud, Mohammad Norouzi, and Kevin Swersky. 2020. "Your Classifier Is Secretly an Energy Based Model and You Should Treat It Like One." In *International Conference on Learning Representations*.

Guidotti, Riccardo. 2022. "Counterfactual Explanations and How to Find Them: Literature Review and Benchmarking." *Data Mining and Knowledge Discovery*, 1–55.

Guo, Hangzhi, Thanh H. Nguyen, and Amulya Yadav. 2023. "CounterNet: End-to-End Training of Prediction Aware Counterfactual Explanations." In *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, 577–89. KDD '23. New York, NY, USA: Association for Computing Machinery. https://doi.org/10.1145/3580305.3599290.

Kolter, Zico. 2023."Keynote Addresses: SaTML 2023 ." In *2023 IEEE Conference on Secure and Trustworthy Machine Learning (SaTML)*, xvi–. Los Alamitos, CA, USA: IEEE Computer Society. https://doi.org/10.1109/SaTML54575.2023.00009.

Lakshminarayanan, Balaji, Alexander Pritzel, and Charles Blundell. 2017. "Simple and Scalable Predictive Uncertainty Estimation Using Deep Ensembles." *Advances in Neural Information Processing Systems* 30.

Lippe, Phillip. 2024. "UvA Deep Learning Tutorials." https://uvadlc-notebooks.readthedocs.io/en/latest/.

Luu, Hoai Linh, and Naoya Inoue. 2023. "Counterfactual Adversarial Training for Improving Robustness of Pre-Trained Language Models." In *Proceedings of the 37th Pacific Asia Conference on Language, Information and Computation*, 881–88.

Murphy, Kevin P. 2022. *Probabilistic Machine Learning: An Introduction*. MIT Press.

O'Neil, Cathy. 2016. *Weapons of Math Destruction: How Big Data Increases Inequality and Threatens Democracy*. Crown.

Pawelczyk, Martin, Chirag Agarwal, Shalmali Joshi, Sohini Upadhyay, and Himabindu Lakkaraju. 2022. "Exploring Counterfactual Explanations Through the Lens of Adversarial Examples: A Theoretical and Empirical Analysis." In *Proceedings of the 25th International Conference on Artificial Intelligence and Statistics*, edited by Gustau Camps-Valls, Francisco J. R. Ruiz, and Isabel Valera, 151:4574–94. Proceedings of Machine Learning Research. PMLR. https://proceedings.mlr.press/v151/pawelczyk22a.html.

Poyiadzi, Rafael, Kacper Sokol, Raul Santos-Rodriguez, Tijl De Bie, and Peter Flach. 2020. "FACE: Feasible and Actionable Counterfactual Explanations." In *Proceedings of the AAAI/ACM Conference on AI, Ethics, and Society*, 344–50.

Ross, Alexis, Himabindu Lakkaraju, and Osbert Bastani. 2024. "Learning Models for Actionable Recourse." In *Proceedings of the 35th International Conference on Neural Information Processing Systems*. NIPS '21. Red Hook, NY, USA: Curran Associates Inc.

Sauer, Axel, and Andreas Geiger. 2021. "Counterfactual Generative Networks." https://arxiv.org/abs/2101.06046.

Schut, Lisa, Oscar Key, Rory Mc Grath, Luca Costabello, Bogdan Sacaleanu, Yarin Gal, et al. 2021. "Generating Interpretable Counterfactual Explanations By Implicit Minimisation of Epistemic and Aleatoric Uncertainties." In *International Conference on Artificial Intelligence and Statistics*, 1756–64. PMLR.

Szegedy, Christian, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. 2013. "Intriguing Properties of Neural Networks." https://arxiv.org/abs/1312.6199.

Teney, Damien, Ehsan Abbasnedjad, and Anton van den Hengel. 2020. "Learning What Makes a Difference from Counterfactual Examples and Gradient Supervision." In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part x 16*, 580–99. Springer.

Wachter, Sandra, Brent Mittelstadt, and Chris Russell. 2017. "Counterfactual Explanations Without Opening the Black Box: Automated Decisions and the GDPR." *Harv. JL & Tech.* 31: 841. https://doi.org/10.2139/ssrn.3063289.

Wilson, Andrew Gordon. 2020. "The Case for Bayesian Deep Learning." https://arxiv.org/abs/2001.10995.

Wu, Tongshuang, Marco Tulio Ribeiro, Jeffrey Heer, and Daniel Weld. 2021. "Polyjuice: Generating Counterfactuals for Explaining, Evaluating, and Improving Models." In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing*

*(Volume 1: Long Papers)*, edited by Chengqing Zong, Fei Xia, Wenjie Li, and Roberto Navigli, 6707–23. Online: Association for Computational Linguistics. https://doi.org/10.18653/v1/2021.acl-long.523.

# A Training Details

## A.1 Initial Grid Search

For the initial round of experiments we

### A.1.1 Generator Parameters

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

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

- Generator Parameters:
  - $\lambda_{\text{cost}}$: `0.0, 0.001, 0.1`
  - $\lambda_{\text{div}}$: `0.01, 0.05, 0.1, 0.5, 1.0, 5.0, 10.0, 15.0`
  - Learning Rate: `1.0`
  - Maximum Iterations: `20, 50, 100`
  - Optimizerimizer: `sgd`
- Generator: `ecco, generic, omni, revise`
- Training Parameters:
  - Objective: `full, vanilla`

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

- Counterfactual Parameters:
  - Convergence: `max_iter`
  - Maximum Iterations: `100`
  - No. Individuals: `100`
  - No. Runs: `5`
- Generator Parameters:
  - $\lambda_{\text{cost}}$: `0.0`
  - $\lambda_{\text{div}}$: `0.1, 0.5, 1.0, 5.0, 10.0, 20.0`
  - Learning Rate: `1.0`
  - Maximum Iterations: `50`
  - Optimizerimizer: `sgd`

#### A.1.1.1 Linearly Separable

- **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.
- **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.
- **Maximum Iterations**: No clear patterns recognizable, so it seems that smaller choices are ok.
- **Validity**: *ECCo* almost always valid except for very low values during training and high values at evaluation time. *Generic* often has poor validity.
- **Accuracy**: Seems largely unaffected.

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

| Objective | $\lambda_{\text{div}}$(train) | Generator | Value | Std |
|---|---|---|---|---|
| full | 0.01 | *ECCo* | $-9.91 \cdot 10^{11}$ | $2.25 \cdot 10^{12}$ |
| full | 0.01 | *Generic* | $-5.71 \cdot 10^{17}$ | $1.3 \cdot 10^{18}$ |
| **full** | **0.01** | **Omniscient** | **-2.54** | **0.116** |
| full | 0.01 | *REVISE* | -15.6 | 13.2 |

Continuing table below.

| Objective | $\lambda_{\text{div}}$(train) | Generator | Value | Std |
|---|---|---|---|---|
| vanilla | 0.01 | *ECCo* | -4.28 | 3.52 |
| vanilla | 0.01 | *Generic* | -4.45 | 3.47 |
| vanilla | 0.01 | *Omniscient* | -5.12 | 4.46 |
| vanilla | 0.01 | *REVISE* | -4.91 | 4.24 |
| full | 0.05 | *ECCo* | $-5.63 \cdot 10^{5}$ | $1.28 \cdot 10^{6}$ |
| full | 0.05 | *Generic* | $-8.35 \cdot 10^{17}$ | $1.9 \cdot 10^{18}$ |
| **full** | **0.05** | **Omniscient** | **-2.53** | **0.114** |
| full | 0.05 | *REVISE* | -15 | 12.6 |
| vanilla | 0.05 | *ECCo* | -4.4 | 3.66 |
| vanilla | 0.05 | *Generic* | -4.38 | 3.48 |
| vanilla | 0.05 | *Omniscient* | -5.25 | 4.62 |
| vanilla | 0.05 | *REVISE* | -4.94 | 4.22 |
| full | 0.1 | *ECCo* | $-6.74 \cdot 10^{5}$ | $1.53 \cdot 10^{6}$ |
| full | 0.1 | *Generic* | $-1.72 \cdot 10^{11}$ | $3.9 \cdot 10^{11}$ |
| **full** | **0.1** | **Omniscient** | **-2.56** | **0.124** |
| full | 0.1 | *REVISE* | -15.6 | 13.2 |
| vanilla | 0.1 | *ECCo* | -4.28 | 3.52 |
| vanilla | 0.1 | *Generic* | -4.45 | 3.48 |
| vanilla | 0.1 | *Omniscient* | -5.12 | 4.46 |
| vanilla | 0.1 | *REVISE* | -4.91 | 4.25 |
| full | 0.5 | *ECCo* | -11.8 | 9.83 |
| full | 0.5 | *Generic* | $-1.06 \cdot 10^{18}$ | $2.42 \cdot 10^{18}$ |
| **full** | **0.5** | **Omniscient** | **-2.54** | **0.123** |
| full | 0.5 | *REVISE* | -15 | 12.6 |
| vanilla | 0.5 | *ECCo* | -4.4 | 3.65 |
| vanilla | 0.5 | *Generic* | -4.38 | 3.48 |
| vanilla | 0.5 | *Omniscient* | -5.25 | 4.61 |
| vanilla | 0.5 | *REVISE* | -4.95 | 4.22 |
| full | 1 | *ECCo* | -11.5 | 11.1 |
| full | 1 | *Generic* | $-1.71 \cdot 10^{11}$ | $3.88 \cdot 10^{11}$ |
| **full** | **1** | **Omniscient** | **-2.59** | **0.117** |
| full | 1 | *REVISE* | -15.7 | 13.3 |
| vanilla | 1 | *ECCo* | -4.28 | 3.51 |
| vanilla | 1 | *Generic* | -4.44 | 3.47 |
| vanilla | 1 | *Omniscient* | -5.11 | 4.46 |
| vanilla | 1 | *REVISE* | -4.91 | 4.25 |
| full | 5 | *ECCo* | -3.99 | 3.12 |
| full | 5 | *Generic* | $-4.88 \cdot 10^{17}$ | $1.11 \cdot 10^{18}$ |
| **full** | **5** | **Omniscient** | **-2.53** | **0.117** |
| full | 5 | *REVISE* | -14.6 | 12.1 |
| vanilla | 5 | *ECCo* | -4.4 | 3.65 |
| vanilla | 5 | *Generic* | -4.38 | 3.48 |
| vanilla | 5 | *Omniscient* | -5.25 | 4.61 |
| vanilla | 5 | *REVISE* | -4.95 | 4.22 |
| **full** | **10** | **ECCo** | **-2.31** | **0.735** |
| full | 10 | *Generic* | $-1.7 \cdot 10^{11}$ | $3.86 \cdot 10^{11}$ |
| full | 10 | *Omniscient* | -2.53 | 0.117 |
| full | 10 | *REVISE* | -15.5 | 13 |
| vanilla | 10 | *ECCo* | -4.28 | 3.51 |
| vanilla | 10 | *Generic* | -4.44 | 3.47 |
| vanilla | 10 | *Omniscient* | -5.12 | 4.46 |
| vanilla | 10 | *REVISE* | -4.91 | 4.24 |
| **full** | **15** | **ECCo** | **-2.01** | **0.488** |
| full | 15 | *Generic* | $-4.91 \cdot 10^{17}$ | $1.12 \cdot 10^{18}$ |
| full | 15 | *Omniscient* | -2.53 | 0.116 |

Continuing table below.

| **Objective** | $\lambda_{\text{div}}$ **(train)** | **Generator** | **Value** | **Std** |
|---|---|---|---|---|
| full | 15 | *REVISE* | -14.4 | 11.7 |
| vanilla | 15 | *ECCo* | -4.4 | 3.65 |
| vanilla | 15 | *Generic* | -4.38 | 3.48 |
| vanilla | 15 | *Omniscient* | -5.25 | 4.6 |
| vanilla | 15 | *REVISE* | -4.95 | 4.23 |

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

| **Objective** | $\lambda_{\text{cost}}$ **(train)** | **Generator** | **Value** | **Std** |
|---|---|---|---|---|
| full | 0 | *ECCo* | $-5.32 \cdot 10^{3}$ | $1.21 \cdot 10^{4}$ |
| full | 0 | *Generic* | $-1.03 \cdot 10^{18}$ | $2.34 \cdot 10^{18}$ |
| **full** | **0** | **Omniscient** | **-2.64** | **0.125** |
| full | 0 | *REVISE* | -15.4 | 12.9 |
| vanilla | 0 | *ECCo* | -4.34 | 3.58 |
| vanilla | 0 | *Generic* | -4.41 | 3.48 |
| vanilla | 0 | *Omniscient* | -5.18 | 4.54 |
| vanilla | 0 | *REVISE* | -4.93 | 4.23 |
| full | 0.001 | *ECCo* | -362 | 811 |
| full | 0.001 | *Generic* | $-2.65 \cdot 10^{17}$ | $6.03 \cdot 10^{17}$ |
| **full** | **0.001** | **Omniscient** | **-2.49** | **0.115** |
| full | 0.001 | *REVISE* | -15.5 | 13 |
| vanilla | 0.001 | *ECCo* | -4.34 | 3.58 |
| vanilla | 0.001 | *Generic* | -4.41 | 3.48 |
| vanilla | 0.001 | *Omniscient* | -5.18 | 4.53 |
| vanilla | 0.001 | *REVISE* | -4.93 | 4.23 |
| full | 0.1 | *ECCo* | $-3.72 \cdot 10^{11}$ | $8.46 \cdot 10^{11}$ |
| full | 0.1 | *Generic* | $-4.49 \cdot 10^{14}$ | $1.02 \cdot 10^{15}$ |
| **full** | **0.1** | **Omniscient** | **-2.5** | **0.112** |
| full | 0.1 | *REVISE* | -14.6 | 12.2 |
| vanilla | 0.1 | *ECCo* | -4.34 | 3.58 |
| vanilla | 0.1 | *Generic* | -4.41 | 3.48 |
| vanilla | 0.1 | *Omniscient* | -5.18 | 4.54 |
| vanilla | 0.1 | *REVISE* | -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 abismal. *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 | $\lambda_{\text{div}}$(train) | Generator | Value | Std |
|---|---|---|---|---|
| full | 0.01 | *ECCo* | $-2.8 \cdot 10^{22}$ | $6.39 \cdot 10^{22}$ |
| full | 0.01 | *Generic* | $-4.89 \cdot 10^{30}$ | $1.11 \cdot 10^{31}$ |
| **full** | **0.01** | **Omniscient** | **-4.74** | **5.08** |
| full | 0.01 | *REVISE* | -572 | $1.25 \cdot 10^{3}$ |
| vanilla | 0.01 | *ECCo* | -15.5 | 17.3 |
| vanilla | 0.01 | *Generic* | -10.9 | 11.9 |
| vanilla | 0.01 | *Omniscient* | -12.7 | 14.4 |
| vanilla | 0.01 | *REVISE* | -11.2 | 13 |
| full | 0.05 | *ECCo* | $-1.55 \cdot 10^{16}$ | $3.52 \cdot 10^{16}$ |
| full | 0.05 | *Generic* | $-2.22 \cdot 10^{20}$ | $5 \cdot 10^{20}$ |
| **full** | **0.05** | **Omniscient** | **-4.41** | **4.48** |
| full | 0.05 | *REVISE* | $-1.04 \cdot 10^{3}$ | $2.3 \cdot 10^{3}$ |
| vanilla | 0.05 | *ECCo* | -15.5 | 17.2 |
| vanilla | 0.05 | *Generic* | -11.7 | 12.8 |
| vanilla | 0.05 | *Omniscient* | -12.4 | 14.1 |
| vanilla | 0.05 | *REVISE* | -11.3 | 13.1 |
| full | 0.1 | *ECCo* | $-3.41 \cdot 10^{3}$ | $7.73 \cdot 10^{3}$ |
| full | 0.1 | *Generic* | $-5.22 \cdot 10^{30}$ | $1.19 \cdot 10^{31}$ |
| **full** | **0.1** | **Omniscient** | **-4.78** | **5.12** |
| full | 0.1 | *REVISE* | -288 | 594 |
| vanilla | 0.1 | *ECCo* | -15.5 | 17.2 |
| vanilla | 0.1 | *Generic* | -10.9 | 11.9 |
| vanilla | 0.1 | *Omniscient* | -12.7 | 14.4 |
| vanilla | 0.1 | *REVISE* | -11.3 | 13.1 |
| full | 0.5 | *ECCo* | -7.09 | 7.51 |
| full | 0.5 | *Generic* | $-1.11 \cdot 10^{31}$ | $2.53 \cdot 10^{31}$ |
| **full** | **0.5** | **Omniscient** | **-4.58** | **4.83** |
| full | 0.5 | *REVISE* | $-1.19 \cdot 10^{3}$ | $2.64 \cdot 10^{3}$ |
| vanilla | 0.5 | *ECCo* | -15.5 | 17.2 |
| vanilla | 0.5 | *Generic* | -11.7 | 12.8 |
| vanilla | 0.5 | *Omniscient* | -12.4 | 14.1 |
| vanilla | 0.5 | *REVISE* | -11.3 | 13.1 |
| full | 1 | *ECCo* | -6.06 | 6.33 |
| full | 1 | *Generic* | $-1.58 \cdot 10^{33}$ | $3.59 \cdot 10^{33}$ |
| **full** | **1** | **Omniscient** | **-4.66** | **4.89** |
| full | 1 | *REVISE* | $-1.16 \cdot 10^{3}$ | $2.59 \cdot 10^{3}$ |
| vanilla | 1 | *ECCo* | -15.5 | 17.3 |
| vanilla | 1 | *Generic* | -10.9 | 11.9 |
| vanilla | 1 | *Omniscient* | -12.7 | 14.4 |
| vanilla | 1 | *REVISE* | -11.3 | 13.1 |
| **full** | **5** | **ECCo** | **-2.57** | **2.07** |
| full | 5 | *Generic* | $-1.17 \cdot 10^{28}$ | $2.66 \cdot 10^{28}$ |
| full | 5 | *Omniscient* | -4.29 | 4.31 |
| full | 5 | *REVISE* | -530 | $1.16 \cdot 10^{3}$ |
| vanilla | 5 | *ECCo* | -15.5 | 17.2 |
| vanilla | 5 | *Generic* | -11.7 | 12.7 |
| vanilla | 5 | *Omniscient* | -12.4 | 14.1 |
| vanilla | 5 | *REVISE* | -11.3 | 13.1 |
| **full** | **10** | **ECCo** | **-1.76** | **0.974** |
| full | 10 | *Generic* | $-1.54 \cdot 10^{33}$ | $3.51 \cdot 10^{33}$ |
| full | 10 | *Omniscient* | -4.44 | 4.56 |
| full | 10 | *REVISE* | $-1.52 \cdot 10^{3}$ | $3.4 \cdot 10^{3}$ |
| vanilla | 10 | *ECCo* | -15.5 | 17.3 |

Continuing table below.

| Objective | $\lambda_{\text{div}}$(train) | Generator | Value | Std |
|---|---|---|---|---|
| vanilla | 10 | *Generic* | -10.9 | 11.9 |
| vanilla | 10 | *Omniscient* | -12.7 | 14.4 |
| vanilla | 10 | *REVISE* | -11.3 | 13.1 |
| **full** | **15** | **ECCo** | **-1.37** | **0.365** |
| full | 15 | *Generic* | $-5.32 \cdot 10^{28}$ | $1.21 \cdot 10^{29}$ |
| full | 15 | *Omniscient* | -4.34 | 4.38 |
| full | 15 | *REVISE* | -473 | $1.03 \cdot 10^{3}$ |
| vanilla | 15 | *ECCo* | -15.5 | 17.2 |
| vanilla | 15 | *Generic* | -11.7 | 12.8 |
| vanilla | 15 | *Omniscient* | -12.4 | 14.1 |
| vanilla | 15 | *REVISE* | -11.3 | 13.1 |

#### A.1.1.3 Circles

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

Table A4: Results for Circles data by energy penalty.

| Objective | $\lambda_{\text{div}}$(train) | Generator | Value | Std |
|---|---|---|---|---|
| **full** | **0.01** | **ECCo** | **-1.26** | **0.423** |
| full | 0.01 | *Generic* | -1.49 | 0.71 |
| full | 0.01 | *Omniscient* | -5.21 | 5.25 |
| full | 0.01 | *REVISE* | $-2.71 \cdot 10^{26}$ | $6.37 \cdot 10^{26}$ |
| vanilla | 0.01 | *ECCo* | -9.33 | 7.34 |
| vanilla | 0.01 | *Generic* | -8.89 | 6.88 |
| vanilla | 0.01 | *Omniscient* | -8.67 | 6.87 |
| vanilla | 0.01 | *REVISE* | -8.65 | 6.8 |
| full | 0.05 | *ECCo* | -1.29 | 0.397 |
| **full** | **0.05** | **Generic** | **-1.21** | **0.356** |
| full | 0.05 | *Omniscient* | -5.08 | 5.09 |
| full | 0.05 | *REVISE* | $-5.91 \cdot 10^{27}$ | $1.36 \cdot 10^{28}$ |
| vanilla | 0.05 | *ECCo* | -9.35 | 7.32 |
| vanilla | 0.05 | *Generic* | -8.85 | 6.87 |
| vanilla | 0.05 | *Omniscient* | -8.7 | 6.96 |
| vanilla | 0.05 | *REVISE* | -8.52 | 6.76 |
| **full** | **0.1** | **ECCo** | **-1.2** | **0.383** |
| full | 0.1 | *Generic* | -1.5 | 0.735 |
| full | 0.1 | *Omniscient* | -5.17 | 5.23 |
| full | 0.1 | *REVISE* | $-3.06 \cdot 10^{26}$ | $7.7 \cdot 10^{26}$ |
| vanilla | 0.1 | *ECCo* | -9.33 | 7.32 |
| vanilla | 0.1 | *Generic* | -8.88 | 6.86 |
| vanilla | 0.1 | *Omniscient* | -8.69 | 6.9 |

Continuing table below.

| Objective | $\lambda_{\text{div}}$(train) | Generator | Value | Std |
|---|---|---|---|---|
| vanilla | 0.1 | *REVISE* | -8.68 | 6.81 |
| **full** | **0.5** | **ECCo** | **-1.12** | **0.217** |
| full | 0.5 | *Generic* | -1.21 | 0.352 |
| full | 0.5 | *Omniscient* | -5.09 | 5.12 |
| full | 0.5 | *REVISE* | $-5.97 \cdot 10^{27}$ | $1.37 \cdot 10^{28}$ |
| vanilla | 0.5 | *ECCo* | -9.35 | 7.3 |
| vanilla | 0.5 | *Generic* | -8.89 | 6.92 |
| vanilla | 0.5 | *Omniscient* | -8.68 | 6.93 |
| vanilla | 0.5 | *REVISE* | -8.53 | 6.75 |
| **full** | **1** | **ECCo** | **-1.1** | **0.163** |
| full | 1 | *Generic* | -1.49 | 0.726 |
| full | 1 | *Omniscient* | -5.16 | 5.2 |
| full | 1 | *REVISE* | $-3.09 \cdot 10^{26}$ | $7.22 \cdot 10^{26}$ |
| vanilla | 1 | *ECCo* | -9.34 | 7.36 |
| vanilla | 1 | *Generic* | -8.86 | 6.85 |
| vanilla | 1 | *Omniscient* | -8.7 | 6.9 |
| vanilla | 1 | *REVISE* | -8.69 | 6.85 |
| full | 5 | *ECCo* | -1.75 | 0.154 |
| **full** | **5** | **Generic** | **-1.21** | **0.363** |
| full | 5 | *Omniscient* | -5.14 | 5.16 |
| full | 5 | *REVISE* | $-1.1 \cdot 10^{28}$ | $2.5 \cdot 10^{28}$ |
| vanilla | 5 | *ECCo* | -9.36 | 7.32 |
| vanilla | 5 | *Generic* | -8.88 | 6.91 |
| vanilla | 5 | *Omniscient* | -8.7 | 6.93 |
| vanilla | 5 | *REVISE* | -8.52 | 6.73 |
| full | 10 | *ECCo* | $-1.02 \cdot 10^{6}$ | $2.32 \cdot 10^{6}$ |
| **full** | **10** | **Generic** | **-1.49** | **0.702** |
| full | 10 | *Omniscient* | -5.13 | 5.16 |
| full | 10 | *REVISE* | $-3.74 \cdot 10^{26}$ | $9.09 \cdot 10^{26}$ |
| vanilla | 10 | *ECCo* | -9.31 | 7.33 |
| vanilla | 10 | *Generic* | -8.87 | 6.86 |
| vanilla | 10 | *Omniscient* | -8.7 | 6.89 |
| vanilla | 10 | *REVISE* | -8.69 | 6.83 |
| full | 15 | *ECCo* | $-3.31 \cdot 10^{13}$ | $7.54 \cdot 10^{13}$ |
| **full** | **15** | **Generic** | **-1.22** | **0.37** |
| full | 15 | *Omniscient* | -5.2 | 5.23 |
| full | 15 | *REVISE* | $-9.01 \cdot 10^{27}$ | $2.06 \cdot 10^{28}$ |
| vanilla | 15 | *ECCo* | -9.38 | 7.34 |
| vanilla | 15 | *Generic* | -8.86 | 6.87 |
| vanilla | 15 | *Omniscient* | -8.69 | 6.96 |
| vanilla | 15 | *REVISE* | -8.51 | 6.73 |