ReLax: An Efficient and Scalable Recourse Explanation Benchmarking Library using JAX

Anonymous Author(s)
Affiliation
Address
email

Abstract

Despite the progress made in the field of algorithmic recourse, current research practices remain constrained, largely restricting benchmarking and evaluation of recourse methods to medium-sized datasets (approximately 50k data points) due to the severe runtime overhead of recourse generation. This constraint impedes the pace of research development in algorithmic recourse and raises concerns about the scalability of existing methods. To mitigate this problem, we propose ReLax, a JAX-based benchmarking library, designed for efficient and scalable recourse explanations. ReLax supports a wide range of recourse methods and datasets and offers performance improvements of at least two orders of magnitude over existing libraries. Notably, we demonstrate that ReLax is capable of benchmarking real-world datasets of up to 10M data points, roughly 200 times the scale of current practices, without imposing prohibitive computational costs. ReLax is fully open-sourced and can be accessed at https://github.com/BirkhoffG/jax-relax.

1 Introduction

The field of algorithmic recourse and counterfactual (CF) explanation[1] [46, 43, 34, 25] gains increasing attention from the research community as recourse explanations are often favored by human end-users by providing a contrastive case to individuals adversely impacted by algorithm-driven decisions. For instance, recourse methods can provide suggestions for loan applicants who have been rejected by a bank's ML algorithm, or provide actionable recommendations for teachers engaging with students teetering on the edge of school dropout.

Numerous recourse explanation methods have been recently proposed [46, 34, 43, 42, 48, 19, 24, 45, 40]. However, despite the progress made, current research practices often restrict the evaluation of recourse explanation methods on medium-sized datasets (with under 50k data points). This constraint primarily stems from the excessive runtime overhead of recourse generation by the existing open-source recourse libraries [36, 34, 27]. For instance, as shown in Figure 1, the CARLA library [36], a popular recourse explanation library, requires roughly 30 minutes to benchmark the adult dataset containing ∼32,000 data points. At this speed, it would take CARLA approximately 15 hours to benchmark a dataset with one million samples, and nearly one week to benchmark a dataset with a scale of 10 million. As a result, this severe runtime overhead hinders the large-scale analysis of recourse explanations, impedes the pace of research development of new recourse methods, and raises concerns about the scalability of existing methods being deployed in data-intensive ML applications.

[1] It is worth noting that counterfactual explanation [46], algorithmic recourse [43], and contrastive explanation [12] share close connections [45, 40], which leads us to use these terms interchangeably.

Submitted to the Workshop "XAI in Action: Past, Present, and Future Applications" at the 37th Conference on Neural Information Processing Systems (NeurIPS 2023). Do not distribute.

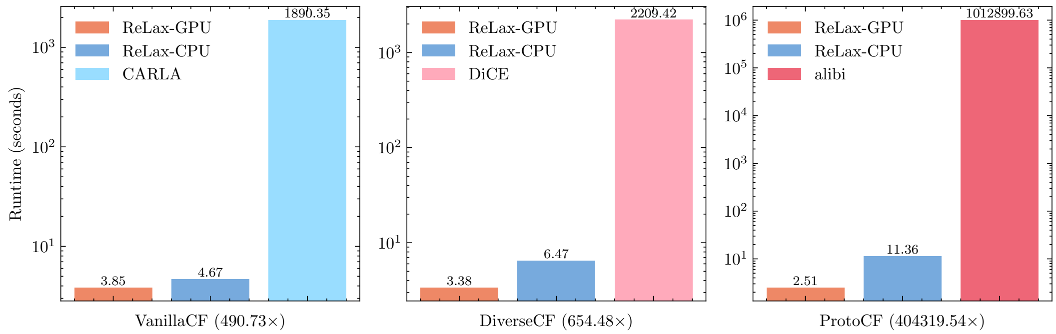


Figure 1: Runtime comparison of benchmarking the *adult* dataset between ReLax and three open-source recourse librarires (CARLA [36], DiCE [34], and alibi [27]). ReLax outperforms existing libraries with at least two orders of magnitude speed-up in recourse generations.

Contributions In this paper, we present ReLax (Recourse Explanation Library using Jax), an efficient and scalable benchmarking library for recourse and counterfactual explanations. We show that by leveraging language primitives such as vectorization, parallelization, and JIT compilation in JAX [9, 16], ReLax achieves over two orders of magnitude speed up than existing libraries (as shown in Figure 1). Notably, we demonstrate that ReLax is capable of benchmarking real-world with 10 million data points, roughly 200 times the scale of current research practices, without imposing prohibitive computational costs. Our primary contributions are summarized as follows:

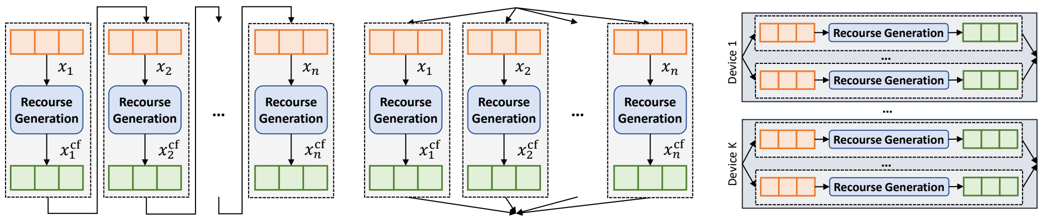
- (Fast and Scalable System) We propose ReLax, the *first* JAX-based library for recourse explanation, enabling efficient and scalable recourse generation. ReLax is at least two order-of-magnitudes faster than the existing recourse explanation libraries (shown in Figure 1). We further demonstrate that ReLax can real-world datasets of up to 10M data points with a reasonable amount of computational cost.
- (Comprehensive set of Methods) ReLax supports a diverse set of recourse methods and datasets. Notably, we implement eight recourse explanation methods in JAX ranging from non-parametric, semi-parametric, and parametric recourse explanation methods. In addition, we include 14 medium-sized datasets, and one large-scale dataset.
- (Extensive Experiments) We perform comprehensive experiments on both medium-sized and large-sized datasets. Our experimental results present an open research challenge in optimally balancing the trade-off between cost and invalidity.
- (Open-sourced System) We have made ReLax fully open-sourced at https://github.com/BirkhoffG/jax-relax, allowing for the reproduction of our experiments and facilitating rapid and scalable benchmarking for newly proposed recourse methods.

2 ReLax: Towards Efficient and Scalable Recourse Benchmarking

In this section, we provide an overview of ReLax's design. First, We introduce the preliminaries of recourse explanations. Next, we delve into the design of ReLax to enable efficient and scalable recourse explanation benchmarking. Finally, We describe the complete benchmarking process.

2.1 Preliminaries & Problem Formulation

We consider an ML model denoted as $f : \mathbb{R}^d \to [0, 1]$, which is trained on a set of N input data points represented as $D = (x_1, y_1), ..., (x_N, y_N)$, and predict a binary label. Given an input instance x and the ML model f, a recourse explanation method finds a counterfactual example (or recourse) x^{cf} that leads to the ML model providing the opposite prediction compared to the original instance x (i.e., $f(x^{\text{cf}}; \theta) = 1 - f(x; \theta)$), while ensuring a minimal cost of change (i.e., the "distance" $c(x, x^{\text{cf}})$ between x and x^{cf} is minimized). To generate valid recourse explanations, existing methods can be broadly classified into three categories: *non-parametric*, *semi-parametric*, and *parametric* methods.



(a) Sequential recourse generation: Generate recourse explanations one after another.

(b) Vectorized recourse generation: Vectorization for efficiency on a single device.

(c) Parallelized recourse generation: Utilizing multiple computing devices (e.g., GPUs) at scale.

Figure 2: Illustration of three recourse generation processes supported in `ReLax`. (a) Sequential generation strategy generates each recourse explanation one after another, which can be prohibitively slow. (b) Vectorized generation strategy enables modern hardware to perform SIMD, which considerably reduces the runtime overhead for large datasets. (c) Parallelized generation strategy distributes data to multiple devices (e.g., multiple GPUs) for benchmarking at scale.

Non-parametric methods [46, 43, 34, 44, 25, 42] generate recourse explanations x^{cf} by independently solving the underlying optimization problem for every single input instance x:

$$x^{\text{cf}} = \text{argmin}_{x^{\text{cf}}} \ \mathcal{L}\big(f(x^{\text{cf}}), 1 - f(x)\big) + \lambda \cdot c(x, x^{\text{cf}}) \tag{1}$$

where the first part of Eq. 1 maximizes the validity to ensure that the generated recourse x^{cf} gets an opposite prediction to x. The second part of Eq. 1 minimizes the cost of change (or distance) between x and x^{cf}. Finally, λ balances the trade-off between the two objective terms.

Parametric methods [31, 19, 48, 35, 18] aim to train a parametric model $g : \mathbb{R}^d \to \mathbb{R}^d$ parametrized by θ_g to generate recourse explanations in an amortized manner (i.e., $x^{\text{cf}} = g(x; \theta_g)$). Parametric methods optimize the parameter θ_g of a global model via a learning problem:

$$\text{argmin}_{\theta_g} \ \mathcal{L}\big(f(g(x; \theta_g)), 1 - f(x)\big) + \lambda \cdot c(x, g(x; \theta_g)) \tag{2}$$

Importantly, the parametric methods generate the recourse explanation without the need to solve computationally intensive optimization problems during the inference stage.

Semi-parametric methods [44, 37, 22, 2] employ a similar approach to parametric methods by training a parametric model $g(\cdot; \theta_g)$. However, they incorporate an additional step to optimize for recourse explanations, akin to non-parametric methods. Typically, semi-parametric methods involve two stages: (i) First, they train a data model to fit the distribution of the training data, and (ii) subsequently search for recourse explanations x^{cf} utilizing the learned data model.

Remark Regardless of recourse methods, the generation of recourse explanations is sample-independent, i.e., $x^{\text{cf}}_{(i)} | f_\theta, x_{(i)} \perp\!\!\!\perp \{x^{\text{cf}}_{(1)} | f_\theta, x_{(1)}, ..., x^{\text{cf}}_{(n)} | f_\theta, x_{(n)}\}$. This implies that generating a specific recourse $x^{\text{cf}}_{(i)}$ is independent of the generation process for other recourses $x^{\text{cf}} \setminus x^{\text{cf}}_{(i)}$. As a result, it is feasible (and advantageous) to generate recourse explanations in parallel.

2.2 Efficiency and Scalability in `ReLax`

`ReLax` natively supports three recourse generation strategies, namely *sequential*, *vectorized*, and *parallelized* strategy, as illustrated in Figure 2. In addition, `ReLax` further enhances its performance by fusing inner recourse generation steps via the Just-In-Time (JIT) compilation. Together, `ReLax` ensures efficient and scalable performance across diverse data scales and complexities.

Sequential Recourse Generation. The sequential recourse generation strategy involves generating recourse explanations one after another, as illustrated in Figure 2a. Unfortunately, while widely used in existing recourse libraries [36, 34, 27] due to its simplicity in implementation, this strategy becomes computational inefficiency when generating recourse explanations for large-scale datasets (as we show in Table 4 in Section 3). In fact, applying the inefficient sequential strategy is one of the reasons for the slowdown observed in existing recourse libraries.

Vectorized Recourse Generation. To efficiently generate recourse explanations, `ReLax` implements the *vectorized* strategy; it takes advantage of modern hardware by applying the recourse generation operations to the entire dataset *at once* (rather than in an element-wise manner as used by the sequential generation strategy, shown in Figure 2b). This vectorized strategy can considerably accelerate recourse generation by enabling the ability of modern hardware (e.g., CPU, GPU) to perform Single Instruction on Multiple Data (SIMD) in parallel. As a result, the vectorized strategy enhances `ReLax`'s ability to efficiently process large datasets.

Parallelized Recourse Generation. In addition, `ReLax` supports *parallelized* strategy for benchmarking recourse explanation methods at scale. The *parallelized* strategy takes advantage of utilizing multiple computing devices (e.g., multiple GPUs) by splitting a dataset into multiple sub-datasets; each sub-dataset is simultaneously executed in different devices (illustrated in Figure 2c). This strategy allows for even larger-scale datasets to be efficiently processed.

Just-In-Time Compilation. Finally, `ReLax` fuses the inner recourse generation steps as fast low-level kernels via the just-in-time (JIT) compilation to hardware accelerators. The use of JIT compilation significantly improves computational speed and optimizes for reduced memory allocation, thereby ensuring an efficient and scalable recourse generation.

2.3 Benchmarking Details

Recourse Methods. `ReLax` implements eight state-of-the-art recourse methods using JAX including (i) three non-parametric methods (VanillaCF [46], DiverseCF [34], GrowingSphere [30]); (ii) two semi-parametric methods (ProtoCF [44], C-CHVAE [37], CLUE [2]); and (iii) two parametric methods (VAE-CF [31], CounterNet [19]). We provide more details in Appendix D.

Medium-Sized Datasets. In `ReLax`, we gather 14 binary-classification tabular datasets, as summarized in Table 1. fall within the category of medium-sized datasets (i.e., $N < 200,000$), covering a wide range of application domains, including financial, education, healthcare, sociology, etc. Further information on each dataset can be found in Appendix C.

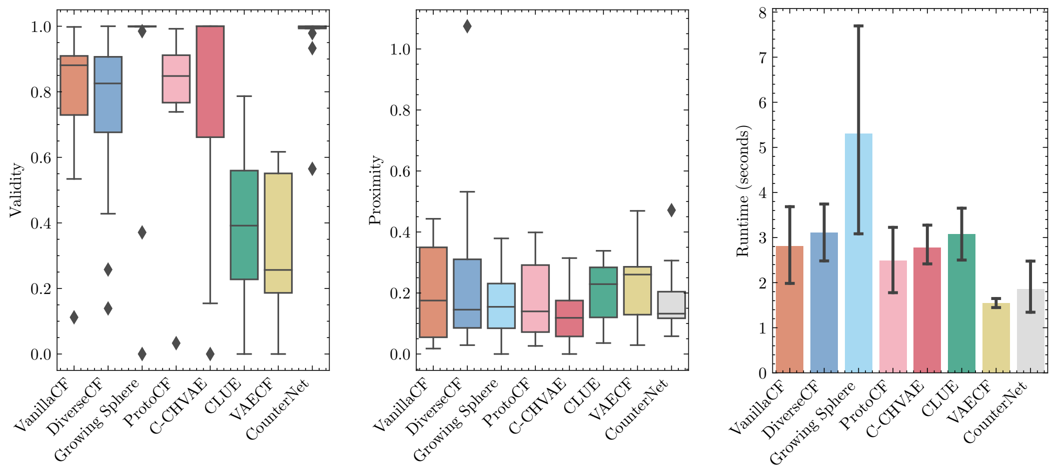
Large-Scale Datasets. In addition to 14 medium-sized datasets, we further benchmark over the forktable dataset [13] for predicting individuals' annual income. This US censuring dataset contains ~10 million data points. To our knowledge, this is the first attempt to benchmark a dataset at the scale of 10 million data points in the recourse explanation community.

Evaluation Metrics. We employ three metrics to evaluate recourse explanations: (i) *Validity*, which measures the fraction of valid recourse explanations x^{cf} with respect to the predictive model $f(\cdot;\theta)$. (ii) *Proximity*, which computes the l_1 distance between the input instance x and its corresponding recourse explanation x^{cf}. (iii) *Runtime*, which represents the total processing time required for generating recourse explanations on the entire testset. Additionally, we include two supplementary metrics, namely *sparsity* and *manifold distance*, and provide the flexibility for users to define their own evaluation metrics. For more details, please refer to Appendix E.

3 Results

Counterfactual Validity. Figure 3a compares the validity achieved by eight parametric methods on 14 medium-sized datasets. Among those eight methods, CounterNet and Growing Sphere achieve the best validity score with a near-perfect validity score on average. On the other hand, C-CHVAE, CLUE, and VAECF show either an unstable validity performance (i.e., a large variation of validity occurs in C-CHVAE), or fail to generate recourse with high validity (i.e., CLUE and VAECF achieve below 50% validity score). The unstable and deteriorated performance of these three methods might be attributed to the training of base VAE models, as these methods rely on a VAE model to generate recourse explanations. Without careful hyper-parameter tuning of the VAE model for each dataset, recourse methods that rely on a VAE model might lead to sub-optimal performance.

Proximity. Table 3b compares the proximity score achieved by all recourse explanation methods on 14 medium-sized datasets. Notably, C-CHVAE outperforms others by achieving the lowest proximity score, approximately 22% and 25% lower than the next best methods - Growing Sphere and CounterNet, respectively. Conversely, DiverseCF and VAECF lag behind in achieving the worst proximity, with their proximity scores standing ~96% and ~78% higher than C-CHVAE.



(a) Boxplot of validity on medium-size datasets. High validity is desirable.

(b) Boxplot of normalized proximity on medium-size datasets. Low proximity is preferable.

(c) Barplot of runtime on medium-size datasets for each recourse method. Low runtime is desirable.

Figure 3: Comparison of recourse method performance across 14 medium-sized datasets. It is desirable to achieve high validity, low proximity, and low runtime.

Cost-Invalidity Trade-off. We further analyze the counterfactual validity and proximity through the lens of the cost-invalidity tradeoff for 14 medium-size datasets. It is vital to ensure that the recourse explanation balances the trade-off between the cost of change (i.e., proximity) and the invalidation percentage (or invalidity, which is computed as 1 - *validity*). This trade-off is illustrated in Figure 4, which plots the average values of proximity against invalidity. We observe that there is no definitive winner in optimally balancing this cost-invalidity trade-off, as none of the recourse methods are positioned at the bottom left of the figure. For instance, while CounterNet exhibits the lowest invalidity, it only achieves a second-tier proximity value, paired with Growing Sphere and ProtoCF. In contrast, C-CHVAE achieves the lowest proximity score but only secures a third-tier invalidity score, on par with VanillaCF and ProtoCF. This analysis underscores the importance of considering both proximity and invalidity in recourse explanations, and presents an open challenge to the research community to devise methods that optimally balance this trade-off.

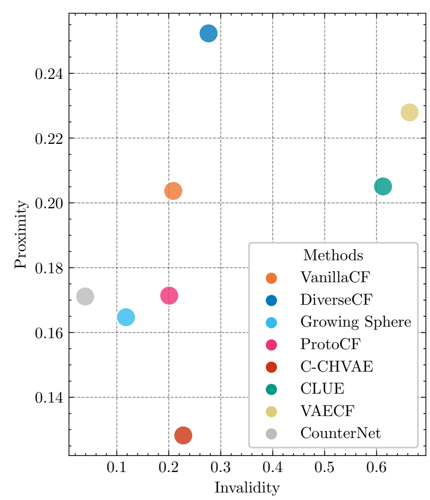


Figure 4: Illustration of the cost-invalidity trade-off across medium-sized datasets for each recourse method. Methods positioned at the bottom left are better.

Running Time. Figure 3c presents the average runtime (in seconds) required by different methods to generate recourse explanations for the entire testset for 14 medium-size datasets. Notably, CounterNet and VAECF, two parametric methods, outperform other methods by maintaining an average runtime of under 2 seconds. Furthermore, all recourse methods complete the entire recourse generation process within 10 seconds. This result underscores the high efficiency of `ReLax` in benchmarking recourse explanations.

Scaling to Large Datasets. We benchmark recourse explanation methods on the forktable dataset, which consists of ∼10 million data points. This benchmarking is conducted using both the vectorized strategy on one Nvidia V100 GPU, and the parallelized strategy on four V100 GPUs. Figure 6 shows the runtime for each recourse explanation method in benchmarking the forktable dataset by adopting the vectorized and parallelized strategies. First, `ReLax` is highly efficient in

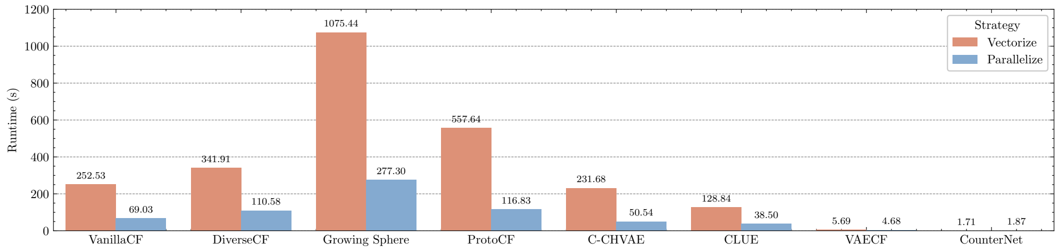


Figure 5: Runtime comparison of different recourse generation strategies on the forktable dataset.

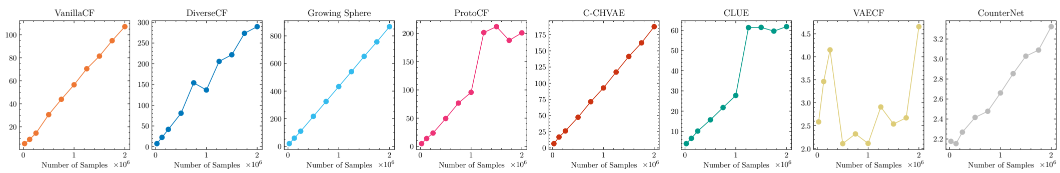


Figure 6: Scalability plot of recourse methods in ReLax on the forktable dataset. With an increasing number of samples, the runtime of each method increases linearly.

benchmarking the large-scale dataset, with the maximum runtime being under 30 minutes. On average, it takes non-parametric methods $\sim$556.7 seconds, semi-parametric methods $\sim$306.1 seconds, and parametric methods $\sim$3.7 seconds on a single GPU machine. In addition, the parallelized strategy cuts the runtime by roughly 4X, which demonstrates that ReLax's potential in benchmarking even larger datasets. *This result demonstrates that ReLax is the first recourse explanation library in benchmarking datasets with 10 million samples within a practical runtime.*

Scalability Analysis. We assess the scalability of ReLax across varieties of dataset sizes. Figure 6 plots the runtime of eight recourse methods on the forktable dataset, with sample sizes ranging from 25,000 to 2,500,000. Importantly, the recourse methods in ReLax exhibit linear time complexity, demonstrating the capability of processing million-sample datasets in less than half an hour. To the best of our knowledge, none of the recourse libraries at present are capable of efficiently handling datasets with over a million samples within a reasonable amount of time.

4 Conclusion & Future Work

In this paper, we present ReLax, an efficient and scalable recourse benchmarking system. Importantly, by leveraging the vectorized and parallelized generation strategies, and JIT compilation, ReLax achieves over two orders-of-magnitude speed-up in benchmarking recourse explanation than existing libraries. Through extensive experiments, we showcase the efficiency and scalability of the system by benchmarking across 14 medium-sized datasets and a ten-million-sized dataset. Furthermore, our experimental results present open research challenges in optimally balancing the trade-off between cost and invalidity. Our work lays a foundation for standardized benchmarking in the field of recourse explanations with special consideration on efficiency and scalability. We envision ReLax becoming an invaluable tool for researchers and ML practitioners aiming to develop, evaluate, compare, and analyze new recourse explanation methods.

Despite the notable advantages of ReLax, we acknowledge that there are still limitations that need to be addressed in future developments. Firstly, as JAX is a relatively new library, its ecosystem is still evolving, which may restrict the implementation of certain recourse methods. For instance, we were unable to implement the actionable recourse method [43] due to the absence of a linear programming solver in JAX. Additionally, the causal recourse method [25] is incompatible due to the lack of support for causal graphical models in JAX. Additionally, given the rapid progress in the field of recourse explanation, it is impractical to incorporate every existing recourse method into ReLax. Therefore, we take initiatives to open-source ReLax to engage with a wider open-source community to contribute new recourse methods. By collaborating with the open-source community, we aim to continue to grow ReLax to stay at the forefront of recourse explanation research and development.

References

[1] Altmeyer, P. (2022). CounterfactualExplanations.jl - a Julia package for Counterfactual Explanations and Algorithmic Recourse.

[2] Antoran, J., Bhatt, U., Adel, T., Weller, A., and Hernández-Lobato, J. M. (2021). Getting a {clue}: A method for explaining uncertainty estimates. In *International Conference on Learning Representations*.

[3] Asuncion, A. and Newman, D. (2007). Uci machine learning repository.

[4] Babuschkin, I., Baumli, K., Bell, A., Bhupatiraju, S., Bruce, J., Buchlovsky, P., Budden, D., Cai, T., Clark, A., Danihelka, I., Dedieu, A., Fantacci, C., Godwin, J., Jones, C., Hemsley, R., Hennigan, T., Hessel, M., Hou, S., Kapturowski, S., Keck, T., Kemaev, I., King, M., Kunesch, M., Martens, L., Merzic, H., Mikulik, V., Norman, T., Papamakarios, G., Quan, J., Ring, R., Ruiz, F., Sanchez, A., Schneider, R., Sezener, E., Spencer, S., Srinivasan, S., Stokowiec, W., Wang, L., Zhou, G., and Viola, F. (2020). The DeepMind JAX Ecosystem.

[5] Ben Hamner, dcthompson, J. (2012). Predicting a biological response.

[6] Bhatt, U., Xiang, A., Sharma, S., Weller, A., Taly, A., Jia, Y., Ghosh, J., Puri, R., Moura, J. M. F., and Eckersley, P. (2020). Explainable machine learning in deployment. In *Proceedings of the 2020 Conference on Fairness, Accountability, and Transparency*, FAT* '20, page 648–657, New York, NY, USA. Association for Computing Machinery.

[7] Binns, R., Van Kleek, M., Veale, M., Lyngs, U., Zhao, J., and Shadbolt, N. (2018). 'it's reducing a human being to a percentage' perceptions of justice in algorithmic decisions. In *Proceedings of the 2018 Chi conference on human factors in computing systems*, pages 1–14.

[8] BlastChar (2019). Telco customer churn.

[9] Bradbury, J., Frostig, R., Hawkins, P., Johnson, M. J., Leary, C., Maclaurin, D., Necula, G., Paszke, A., VanderPlas, J., Wanderman-Milne, S., and Zhang, Q. (2018). JAX: composable transformations of Python+NumPy programs.

[10] Cortez, P. and Silva, A. M. G. (2008). Using data mining to predict secondary school student performance.

[11] Dehghani, M., Gritsenko, A., Arnab, A., Minderer, M., and Tay, Y. (2022). Scenic: A jax library for computer vision research and beyond. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 21393–21398.

[12] Dhurandhar, A., Chen, P.-Y., Luss, R., Tu, C.-C., Ting, P., Shanmugam, K., and Das, P. (2018). Explanations based on the missing: Towards contrastive explanations with pertinent negatives. In *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, NIPS'18, page 590–601, Red Hook, NY, USA. Curran Associates Inc.

[13] Ding, F., Hardt, M., Miller, J., and Schmidt, L. (2021). Retiring adult: New datasets for fair machine learning. *Advances in Neural Information Processing Systems*, 34.

[14] Dua, D. and Graff, C. (2017). UCI machine learning repository.

[15] FICO (2018). Explainable machine learning challenge. https://community.fico.com/s/explainable-machine-learning-challenge.

[16] Frostig, R., Johnson, M. J., and Leary, C. (2018). Compiling machine learning programs via high-level tracing. *Systems for Machine Learning*, 4(9).

[17] Grin, L. (2023). Road safety data.

[18] Guo, H., Jia, F., Chen, J., Squicciarini, A., and Yadav, A. (2022). Rocoursenet: Distributionally robust training of a prediction aware recourse model. *arXiv preprint arXiv:2206.00700*.

[19] Guo, H., Nguyen, T., and Yadav, A. (2023). Counternet: End-to-end training of prediction aware counterfactual explanation. In *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD '23), August 6–10, 2023, Long Beach, CA, USA*.

[20] Hopkins, M., Reeber, E., Forman, G., and Suermondt, J. (1999). Spambase data set. *Hewlett-Packard Labs*, 1(7).

[21] Jagerman, R., Wang, X., Zhuang, H., Qin, Z., Bendersky, M., and Najork, M. (2022). Rax: Composable learning-to-rank using jax. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, page 3051–3060.

[22] Joshi, S., Koyejo, O., Vijitbenjaronk, W., Kim, B., and Ghosh, J. (2019). Towards realistic individual recourse and actionable explanations in black-box decision making systems. *arXiv preprint arXiv:1907.09615*.

[23] Kaggle (2018). Titanic - machine learning from disaster. https://www.kaggle.com/c/titanic/overview.

[24] Karimi, A.-H., Barthe, G., Schölkopf, B., and Valera, I. (2020). A survey of algorithmic recourse: definitions, formulations, solutions, and prospects. *arXiv preprint arXiv:2010.04050*.

[25] Karimi, A.-H., Schölkopf, B., and Valera, I. (2021). Algorithmic recourse: from counterfactual explanations to interventions. In *Proceedings of the 2021 ACM Conference on Fairness, Accountability, and Transparency*, pages 353–362.

[26] Kidger, P. (2021). *On Neural Differential Equations*. PhD thesis, University of Oxford.

[27] Klaise, J., Looveren, A. V., Vacanti, G., and Coca, A. (2021). Alibi explain: Algorithms for explaining machine learning models. *Journal of Machine Learning Research*, 22(181):1–7.

[28] Kohavi, R. and Becker, B. (1996). Uci machine learning repository: Adult data set.

[29] Kuzilek, J., Hlosta, M., and Zdrahal, Z. (2017). Open university learning analytics dataset. *Scientific data*, 4:170171.

[30] Laugel, T., Lesot, M.-J., Marsala, C., Renard, X., and Detyniecki, M. (2017). Inverse classification for comparison-based interpretability in machine learning. *arXiv preprint arXiv:1712.08443*.

[31] Mahajan, D., Tan, C., and Sharma, A. (2019). Preserving causal constraints in counterfactual explanations for machine learning classifiers. *arXiv preprint arXiv:1912.03277*.

[32] Mansouri, K., Ringsted, T., Ballabio, D., Todeschini, R., and Consonni, V. (2013). Quantitative structure–activity relationship models for ready biodegradability of chemicals. *Journal of chemical information and modeling*, 53(4):867–878.

[33] Miller, T. (2019). Explanation in artificial intelligence: Insights from the social sciences. *Artificial Intelligence*, 267:1–38.

[34] Mothilal, R. K., Sharma, A., and Tan, C. (2020). Explaining machine learning classifiers through diverse counterfactual explanations. In *Proceedings of the 2020 Conference on Fairness, Accountability, and Transparency*, pages 607–617.

[35] Nemirovsky, D., Thiebaut, N., Xu, Y., and Gupta, A. (2022). Countergan: Generating counterfactuals for real-time recourse and interpretability using residual gans. In *Uncertainty in Artificial Intelligence*, pages 1488–1497. PMLR.

[36] Pawelczyk, M., Bielawski, S., van den Heuvel, J., Richter, T., and Kasneci, G. (2021). Carla: A python library to benchmark algorithmic recourse and counterfactual explanation algorithms. *Advances in Neural Information Processing Systems Track on Datasets and Benchmarks*.

[37] Pawelczyk, M., Broelemann, K., and Kasneci, G. (2020). Learning model-agnostic counterfactual explanations for tabular data. In *Proceedings of The Web Conference 2020*, pages 3126–3132.

[38] Phan, D., Pradhan, N., and Jankowiak, M. (2019). Composable effects for flexible and accelerated probabilistic programming in numpyro. *arXiv preprint arXiv:1912.11554*.

[39] Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R. (2014). Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research*, 15(1):1929–1958.

[40] Stepin, I., Alonso, J. M., Catala, A., and Pereira-Fariña, M. (2021). A survey of contrastive and counterfactual explanation generation methods for explainable artificial intelligence. *IEEE Access*, 9:11974–12001.

[41] Subramani, P., Vadivelu, N., and Kamath, G. (2021). Enabling fast differentially private sgd via just-in-time compilation and vectorization. *Advances in Neural Information Processing Systems*, 34:26409–26421.

[42] Upadhyay, S., Joshi, S., and Lakkaraju, H. (2021). Towards robust and reliable algorithmic recourse. *arXiv preprint arXiv:2102.13620*.

[43] Ustun, B., Spangher, A., and Liu, Y. (2019). Actionable recourse in linear classification. In *Proceedings of the Conference on Fairness, Accountability, and Transparency*, pages 10–19.

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

[45] Verma, S., Dickerson, J., and Hines, K. (2020). Counterfactual explanations for machine learning: A review. *arXiv preprint arXiv:2010.10596*.

[46] Wachter, S., Mittelstadt, B., and Russell, C. (2017). Counterfactual explanations without opening the black box: Automated decisions and the gdpr. *Harv. JL & Tech.*, 31:841.

[47] Xu, B., Wang, N., Chen, T., and Li, M. (2015). Empirical evaluation of rectified activations in convolutional network. *arXiv preprint arXiv:1505.00853*.

[48] Yang, F., Alva, S. S., Chen, J., and Hu, X. (2021). Model-based counterfactual synthesizer for interpretation. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, KDD '21, page 1964–1974, New York, NY, USA. Association for Computing Machinery.

[49] Yeh, I.-C. and Lien, C.-h. (2009). The comparisons of data mining techniques for the predictive accuracy of probability of default of credit card clients. *Expert Systems with Applications*, 36(2):2473–2480.

[50] Zhang, K. and Fan, W. (2008). Forecasting skewed biased stochastic ozone days: analyses, solutions and beyond. *Knowledge and Information Systems*, 14:299–326.

A Relatex Work

Recourse Explanation Methods Recourse and counterfactual explanation methods concentrate on the generation of new instances that lead to contrastive predicted outcomes [46, 45, 24, 40]. Given their ability to provide actionable recourse, these explanations are often favored by human end-users [7, 33, 6]. We categorize prior work on recourse methods into *non-parametric methods* [46, 43, 34, 25, 42], which aim to find recourse explanations without involving parameterized models, *semi-parametric methods* [44, 37, 22, 2], which indirectly utilize parametric models to find recourse explanations, and *parametric methods* [48, 35, 31, 19, 18], which amortizedly apply parametric models (e.g., a neural network model) for recourse explanation generation. `ReLax` contains a diverse set of recourse explanation methods for comprehensive benchmarking.

Recourse Explanation Libraries To our knowledge, there exists several notable implementations and benchmarks for recourse explanation methods, including CARLA [36], DiCE [34], alibi [27], and CounterfactualExplanations.jl [1]. In particular, CARLA benchmarks 11 recourse explanation methods (mostly based on the implementations from the corresponding research labs) on two medium-size datasets. However, CARLA, along with other libraries, falls short when tasked with benchmarking larger datasets, as it imposes prohibitive computational costs due to ineffective hardware utilization. On the other hand, `ReLax` represents a more efficient and scalable alternative, which can benchmark large-scale datasets.

JAX Finally, we briefly review JAX as it is a central component of `ReLax`. JAX offers language primitives for automatic differentiation, JIT compilation to hardware accelerators, and function vectorization [9, 16]. JAX provides an ease-of-use API to compose computing systems while leveraging accelerators for performance. Due to its ease of use, JAX has been used in computer vision [11], probabilistic programming [38], differential equation [26], differential privacy [41], reinforcement learning [4], learning-to-rank [21], and many other fields. However, the adoption of JAX in recourse explanation, or explainable AI more generally, is absent. To address this gap, we introduce the *first* recourse explanation benchmarking library in the JAX ecosystem.

B API

The primary objective of `ReLax` is to facilitate the benchmarking of state-of-the-art recourse explanation methods on a large scale. We have meticulously designed the API of `ReLax` to prioritize ease of use and extensibility. Figure 7 illustrates the software design of `ReLax`, where the colored boxes represent the main modules, and the gray box represents the high-level functional APIs designed for benchmarking recourse explanations.

Tabular Data Module (i.e., `DataModule`) loads the tabular datasets and prepares the data for ML model training and recourse generation. Users can define features as continuous or categorical features. In addition, users can specify immutable features such that the recourse explanation methods will avoid modifying them during the process of recourse generation. Figure 8 shows an example of customizing the data loading process.

Furthermore, `ReLax` offers the flexibility to customize how recourse constraints are handled, including those introduced by categorical feature preprocessing and immutable features. Users can easily customize recourse constraints by overriding the `TabularDataModule.apply_constraints` method. Figure 9 provides a pseudo-implementation example for customizing recourse constraints. This design allows for recourse generation that satisfies user-defined constraints, such as causal constraints [25] or any other desired constraints.

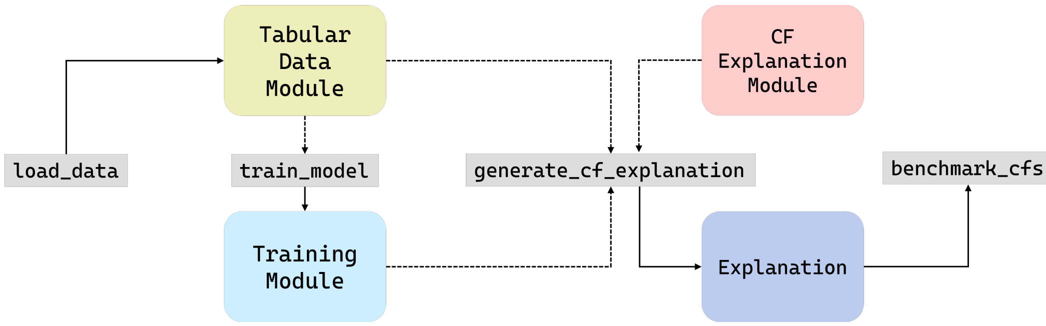


Figure 7: Overview of ReLax's design and APIs. The colored boxes represent the main modules, and the gray box represents the high-level functional APIs designed for loading data, training ML models, and benchmarking recourse explanations. The dashed arrows denote the inputs of the function, and the solid arrows denote the outputs of the function.

```python
from relax import DataModuleConfig, DataModule

data_config = DataModuleConfig(
    # The name of the dataset
    data_name="custom",
    # The directory of the data
    data_dir=".../custom.csv",
    # List all continuous variables
    continous_cols=[...],
    # List all categorical (discrete) variables
    discret_cols=[...],
    # List all immutable features that we do not wish to change
    imutable_cols=[...]
)

# Load the Data Module
datamodule = DataModule.from_config(data_config)
```

Figure 8: An example of customized data loading.

```python

class CustomizedDataModule(DataModule):
    def apply_constraints(
        self,
        x: jax.Array,
        cf: jax.Array,
        hard: bool
    ):
        # Override the method to apply customized constraints
        ...
```

Figure 9: Pseudo-implementation of customizing the recourse constraints.

In the Predictive Training Module, users can define the model structure and the optimization procedure. With the number of epochs and batch size defined, users can train the ML model by simply calling `train_model()`. In the Counterfactual Explanation module, users can choose implemented recourse methods and define the hyperparameters for the recourse explanation. With the predictive function and data as input, users can generate a counterfactual for each data instance by calling `generate_cf_explanations()`. Finally, users can use `benchmark_cfs()` to evaluate the quality of the recourse explanations with the standardized metrics. Figure 10 provides an example implementation of generating and benchmarking recourse explanations.

Table 1: Summary of the 14 medium-sized datasets used in ReLax.

Dataset	# Samples	# Continuous	# Categorical	# Immutable	Category
Adult [28]	32561	2	6	2	Sociology
HELOC [15]	10459	21	2	0	Finance
Credit [49]	30000	20	3	1	Finance
OULAD [29]	32593	23	8	2	Education
Student [10]	649	2	14	0	Education
Titanic [23]	891	2	25	2	Document
Cancer [14]	569	30	0	0	Healthcare
German [3]	1000	7	13	0	Finance
Spam [20]	4601	57	0	0	Computer
Ozone [50]	2534	72	0	0	Physical
QSAR [32]	1055	37	3	0	Life
Bioresponse [5]	3751	1776	0	0	Life
Churn [8]	7043	3	16	1	Business
Road [17]	111762	30	3	0	Sociology

```python
from relax.methods import VanillaCF
from relax import generate_cf_explanations

cf_exp = generate_cf_explanations(
    # Define the recourse method for generating recourses
    VanillaCF(),
    # Define the data module
    datamodule,
    # The predict function
    pred_fn,
    # The auxiliary prediction function
    pred_fn_args={ ... }
)

# Benchmark the recourse methods by returning metrics results
results = benchmark_cfs([cf_exp])
```

Figure 10: Pseudo-implementation of generating and benchmarking recourse explanations.

C Datasets

In ReLax, we gather 14 binary-classification tabular datasets that fall within the category of medium-sized datasets (i.e., $N < 200,000$), covering a wide range of application domains (as summarized in Table 1). Here, we provide further information on each medium-sized dataset:

- Adult [28] was extracted from the census bureau database from 1994, consisting of 32,561 instances. The classifier aims to determine whether an individual makes over 50K USD a year (Y=1) or not (Y=0) using demographic data.
- Credit [49] was obtained from real cardholders' credit risk data in Taiwan, consisting of 30,000 instances. The classifier uses historical payments to predict the default of payment (Y=1) or not (Y=0).
- HELOC [15] is an anonymized dataset of Home Equity Line of Credit (HELOC) applications made by real homeowners, with 10,459 instances. A HELOC is a line of credit typically offered by a bank as a percentage of home equity. The classifier uses information of the applicants to determine whether they will repay their HELOC account within 2 years (Y=1) or not (Y=0).
- OULAD [29] comprises 32,593 instances and is a subset of the 2013 and 2014 OU student data. It includes both demographic data and interaction data of the students. The classifier determines whether MOOC students drop out (Y=1) or not (Y=0), based on their online learning logs.

- Student [10] is a dataset of 649 instances compiled from two Portuguese secondary schools, encompassing reports of marks as well as social and school-related attributes for predicting whether a student will pass (Y=1) or fail (Y=0) the exam.
- Titanic [23] comprises passenger information from the Titanic accident. The classifier utilizes passenger information to determine whether a passenger survived the Titanic shipwreck (Y=1) or not (Y=0).
- Cancer [14] is collected from the Breast Cancer Wisconsin (Diagnostic) dataset comprises diagnostic information obtained from digitized images of fine needle aspirate (FNA) of breast mass tumors, with a total of 569 instances available for analysis. The classifier uses the description of characteristics of the cell nuclei to determine whether the tumor is malignant (Y=1) or benign (Y=0).
- German [3] contains information about credit applications from German banks, with a total of 1,000 instances. The classifier uses information of the applicants to predict whether an applicant is a good (Y=1) or bad (Y=0) credit risk.
- Spam [20] was created by Mark Hopkins, Erik Reeber, George Forman, and Jaap Suermondt at Hewlett-Packard Labs. The classifier uses frequency of words or characters in an Email to determine whether the Email is a spam (Y=1) or not (Y=0).
- Ozone [50] comprises meteorology and ozone data collected from 1998 to 2004 at the Houston, Galveston, and Brazoria (HGB) area. The classifier uses the meteorology data to predict whether a day is an high ozone day (Y=1) or not (Y=0).
- QSAR [32] was built in the Milano Chemometrics and QSAR Research Group. The classifier uses molecular descriptors to determine whether a chemical is ready (Y=1) or not ready (Y=0) biodegradable.
- Bioresponse [5] consists of molecular descriptors of molecules. The classifier aims to predict whether a molecule was seen to elicit a biological response (Y=1) or not (Y=0)
- Churn [8] contains information about a fictional telco company that provided home phone and Internet services to 7043 customers in California in Q3. The classifier aims to predict whether a customer left within the last month (Y=1) or not (Y=0) using multiple important demographics, as well as a Satisfaction Score, Churn Score, and Customer Lifetime Value (CLTV) index.
- Road [17] comprises detailed road safety data about the circumstances of personal injury road collisions in Great Britain from 1979, including the types of vehicles involved and the resulting casualties. The classifier uses the collision information to predict the sex of the involved driver, whether male (Y=1) or female (Y=0).

D Recourse Methods

As discussed in Section 2.3, `ReLax` implements eight state-of-the-art recourse methods. Here, we provide more details about each implemented method.

- **VanillaCF** [46] is a non-parametric post-hoc method that generates recourse explanations by optimizing counterfactual validity and proximity.
- **DiverseCF** [34] is a non-parametric method that optimizes counterfactual validity, proximity, and diversity.
- **Growing Spheres** [30] is a non-parametric method that applies a random search algorithm to generate valid recourses by generating samples around the input point x.
- **ProtoCF** [44] is a semi-parametric method that first trains an auto-encoder model to fit the training data distribution. Next, for each data point, it optimizes for validity, proximity, and data manifold with the support of the auto-encoder model.
- **C-CHVAE** [37] is a semi-parametric method that first trains a variational auto-encoder model to fit the training data distribution. Next, for each data point, it randomly perturbs the latent variables of the VAE model to find a valid recourse explanation.

- **CLUE** [2] is a semi-parametric method that first trains a variational auto-encoder model using the training dataset, then for each data point, it uses the gradient descent to find the latent variables that lead to the VAE model to output a valid recourse explanation.
- **VAECF** [31] is a parametric method that trains a VAE model to directly generate recourse explanations.
- **CounterNet** [19] is a parametric method that jointly trains a predictive network and counterfactual generator. The CF generator is optimized for counterfactual validity and proximity.

E Evaluation Metrics

Here, we provide formal definitions of the evaluation metrics used in `ReLax`.

Predictive Accuracy measures the accuracy of the predictive model defined as the fraction of correct predictive labels.

$$\texttt{Predictive-Accuracy} = \frac{\#|f(x) = y|}{n} \tag{3}$$

Validity refers to the proportion of input instances x for which CF explanation methods generate valid CF examples x^{cf}.

$$\texttt{Validity} = \frac{\#|f(x^{\text{cf}}) = 1 - y|}{n} \tag{4}$$

Proximity is measured by calculating the L_1 norm distance between x and x^{cf} and dividing it by the number of features.

$$\texttt{Proximity} = \frac{1}{nd} \sum_{i=1}^{n} \sum_{j=1}^{d} \| x_i^{(j)} - x_i^{\text{cf}(j)} \|_1 \tag{5}$$

Sparsity is defined by calculating the ratio of the number of feature changes between x and x^{cf} to the total number of features.

$$\texttt{Sparsity} = \frac{1}{nd} \sum_{i=1}^{n} \sum_{j=1}^{d} \| x_i^{(j)} - x_i^{\text{cf}(j)} \|_0 \tag{6}$$

Manifold distance is the L_1 distance between x^{cf} and its nearest neighbor (with $k = 1$) in the dataset.

$$\texttt{Manifold distance} = \frac{1}{n} \sum_{i=1}^{n} \| KNN(x_i^{\text{cf}}, \mathcal{D}) - x_i^{\text{cf}} \|_1 \tag{7}$$

F Additional Experimental Evaluations

F.1 Feature Processing

Handling Continuous & Categorical Features. To ensure fair benchmarking, `ReLax` employs consistent data preprocessing methods for each dataset and method, unless otherwise specified. First, `ReLax` normalizes all continuous features to the range of [0, 1] prior to training. Additionally, `ReLax` transforms all categorical features in each dataset into numeric features using one-hot encoding. During the optimization/training of recourse generation, `ReLax` applies a softmax function to each categorical feature. This softmax function guarantees that each categorical feature in the generated recourse explanations adheres to the one-hot encoding format, as the softmax output will sum up to 1. `ReLax` adopts this categorical normalization to all recourse explanation methods, unless explicitly specified (e.g., in the case of DiverseCF [34], which incorporates a penalty term to enforce adherence to the one-hot encoding format for categorical features).

Handling Immutable Features. To ensure the feasibility of generated recourse explanations, `ReLax` incorporates a mechanism to enforce immutable features, which are features that cannot be altered, to remain unchanged. This is achieved by projecting the corresponding features of each recourse explanation onto the feasible space. During the optimization or training of recourse generation, ReLax applies this projection to ensure that the generated recourse remains within the feasible space (i.e., $x^{\text{cf}} \leftarrow \mathbb{P}(x^{\text{cf}})$). During inference, ReLax enforces the immutability of features by ensuring that the set of immutable features remains unchanged. It is important to note that ReLax has the capability to handle and enforce user-defined constraints as well (See Section B for further details).

F.2 Experimental Settings

Datasets & Hyperparameter Settings As outlined in Section 2.3, ReLax contains 14 medium-sized datasets, and one large-size dataset. We split the dataset into a 75%:25% train-test split. The training set is used to train the predictive model and (semi-)parametric recourse methods. We use the test dataset to benchmark recourse explanations. For all the methods in ReLax, we use the default hyperparameters in the original paper for a fair comparison. See Appendix F for detailed settings.

Predictive Model For each dataset, we train a neural network model and use it as the target predictive model for all baselines. The predictive network contains multiple feed-forward layers; each feed-forward layer uses LeakyRelu activation functions [47] followed by a dropout layer [39]. Details about the model architecture and training for each dataset can be found in Appendix F.

Computational Recourses As described in Section 3, the main results of ReLax are obtained on either a single V100 GPU, or a machine with four GPUs. In addition, the runtime results of CARLA, DiCE, alibi, and ReLax-CPU in Figure 1 are obtained on a 16-core Intel CPU with 64 GB memory.

Hyperparameters of the Predictive Models Table 2 outlines the learning rate, batch size, and the model architecture to train the predictive model, which is a multi-layer perception. For each model, we train for 10 epochs and select the best model with the lowest validation loss.

Hyperparameters of Recourse Methods Here, we outline the hyperparameters used for recourse methods. For more details, please check our code base.

- **VanillaCF** [46]. We set the λ weight to 0.01 to balance the trade-off between proximity and validity. For the target loss, we use binary cross entropy with a learning rate of 0.001. To ensure convergence and avoid overfitting, we set the maximum number of steps to 1000.
- **DiverseCF** [34]. We set the λ_1 weight to 0.01 for proximity. DiverseCF supports finding multiple recourses for an input instance, so we choose to generate 5 recourses, and return the optimal one. We set the learning rate to 0.01 and maximum number of steps to 1000 similar to VanillaCF.
- **Growing Spheres** [30]. We set the maximum number of steps to 100, the number of generated candidate counterfactuals to 1000, and the step size to 0.05.
- **ProtoCF** [44]. For training the auto-encoder model, we set the dimensions of the encoding layer to $[50, 10]$ and the dimensions of the decoding layer to $[10, 50]$ with a learning rate of 0.03 and dropout rate of 0.3.
- **C-CHVAE** [37]. We train the VAE model for 10 epochs using a batch size of 128. The encoding layers of the VAE model are set to [20, 16, 14, 12], and the decoding layer to [12, 14, 16, 20]. During the inference stage, we set the maximum number of steps to 100, the number of generated candidate counterfactuals to 300, and the step size to 0.1.
- **CLUE** [2]. We train the VAE model for 10 epochs using a batch size of 128 and a learning rate of 0.001. The encoding layers of the VAE model are set to [20, 16, 14, 12], and the decoding layer to [12, 14, 16, 20]. During the inference stage, we set the maximum number of steps to 500, and the step size to 0.01.
- **VAECF** [31]. We train the VAE model for 10 epochs using a batch size of 128 and a learning rate of 0.001. The encoding layers of the VAE model are set to [20, 16, 14, 12], and the

decoding layer to [12, 14, 16, 20]. We set the dropout rate to 0.1. Finally, the number of samples is set to 50, and regularization for validity is set to 42.0.

- **CounterNet** [19]. We set the $\lambda_1 = 1.0$, $\lambda_2 = 0.2$, $\lambda_3 = 0.1$ for balancing the $\mathcal{L}_1$, $\mathcal{L}_2$, $\mathcal{L}_3$. We set the dropout rate to 0.3, and the learning rate to 0.003.

F.3 Additional Results

In Table 2, we show the predictive accuracy of the predictive model for each dataset. In Table 3, we present the full performance results of recourse methods on 14 medium-sized datasets.

Table 2: Hyperparameters, architectures, and predictive accuracy of the predictive models for each dataset.

Dataset	**Learning Rate**	**Batch Size**	**Dims.**	**Accuracy**
Adult	.003	256	[29, 50, 10]	.824
HELOC	.003	256	[35,50,10]	.703
OULAD	.001	256	[127,50,10]	.927
Credit	.003	256	[33,50,10]	.813
Cancer	.003	32	[30,50,10]	.909
Student	.003	32	[85,50,10]	.902
Titanic	.003	64	[57,50,10]	.816
German	.004	64	[61,50,10]	.756
Spam	.003	256	[57,50,10]	.934
Ozone	.003	256	[72,50,10]	.934
QSAR	.004	128	[44,50,10]	.848
Bioresponse	.005	256	[1776,50,10]	.788
Churn	.003	256	[46,50,10]	.806
Road	.004	128	[35,50,10]	.751

Table 3: Evaluation of recourse methods on 14 medium-sized datasets.

Dataset	**VanillaCF**		**DiverseCF**		**ProtoCF**		**CounterNet**		**C-CHVAE**		**CLUE**		**Growing Sphere**		**VAE-CF**	
	Val.	Prox.	Val.	Prox.	Val.	Prox.	Val.	Prox.	Val.	Prox.	Val.	Prox.	Val.	Prox.	Val.	Prox.
Adult	.897	6.730	.658	3.414	.764	6.547	.996	5.383	.182	.889	.182	4.704	1.0	5.630	.182	7.951
HELOC	.711	3.309	.826	2.947	.848	3.703	1.0	4.852	1.0	4.145	.787	4.870	1.0	4.655	.617	8.981
OULAD	.707	5.436	.882	14.65	.767	3.367	.999	9.388	1.0	8.258	.540	9.462	1.0	11.35	.506	11.23
Credit	.907	3.908	.974	.959	.876	3.557	1.0	3.864	.155	.505	.418	3.750	1.0	4.376	.155	5.173
Cancer	.965	2.734	.923	9.661	.972	2.059	.993	3.516	1.0	9.433	.608	9.973	1.0	5.396	.608	7.926
Student	.865	21.23	.258	15.40	.798	17.11	1.0	17.89	1.0	15.08	.252	21.05	1.0	14.98	.252	22.60
Titanic	.910	23.69	.139	4.820	.919	17.87	.565	7.176	.996	9.091	.220	16.57	1.0	19.16	.251	18.17
German	.900	21.44	.836	1.55	.848	19.49	.996	13.23	1.0	14.22	.148	2.64	1.0	14.84	.164	21.69
Spam	.986	1.018	.915	1.769	.992	1.682	1.0	3.322	1.0	3.143	.365	2.048	.998	3.449	.365	1.660
Ozone	.112	31.91	1.0	77.36	.033	23.15	1.0	33.95	0.0	0.0	0.0	15.15	0.0	0.0	0.0	8.616
QSAR	.784	15.14	.731	18.67	.739	7.552	1.0	5.350	1.0	7.430	.261	13.90	.985	12.26	.261	12.75
Bioresponse	.998	34.01	.735	944.6	.978	48.61	.994	20.3	1.0	21.7	.566	465.1	.371	36.26	.566	132.6
Churn	.804	17.90	.825	12.67	.889	18.34	.933	14.09	.893	11.81	.571	12.12	1.0	17.43	.199	21.58
Road	.534	1.483	.428	3.087	.767	2.861	.979	4.875	.584	2.862	.509	2.962	1.0	2.898	.584	7.984

F.4 Empirical Findings on the Large-Scale Dataset

In this section, we benchmark recourse explanation methods on the forktable dataset, which consists of ~10 million data points. This benchmarking is conducted using both the vectorized strategy on one Nvidia V100 GPU, and the parallelized strategy on four V100 GPUs. To our knowledge, `ReLax` is the *first* to benchmark datasets with 10 million samples within a practical runtime.

Cost-Invalidity Trade-Off We analyze the validity and proximity of the large dataset by plotting the cost-invalidity tradeoff. Figure 11 plots the proximity against the invalidity of the forktable dataset. We observe a similar pattern as to the result in benchmarking the medium-sized datasets. Similar to Figure 4, we observe that there is no definitive winner in optimally balancing this cost-invalidity trade-off. This result reiterates the difficulty of balancing both proximity and validity in recourse explanations.

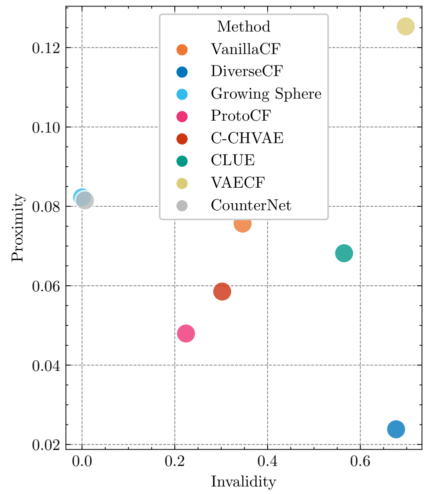


Figure 11: Illustration of the cost-invalidity trade-off on the forktable dataset. Methods at the bottom right are preferable.

Table 4: Ablation study on the adult dataset assessing the impact of JIT compilation and vectorized strategy within ReLax. OOM indicates out-of-memory with the same setting. Missing entries represent bugs or runtime crashes. Both JIT compilation and the vectorized strategy are highly effective to reduce the runtime, with an average reduction of ∼84.1X and ∼4,754.2X, respectively.

		Methods							
JIT	**Vectorization**	**VanillaCF**	**DiverseCF**	**ProtoCF**	**Sphere**	**C-CHVAE**	**CLUE**	**VAE-CF**	**CounterNet**
	✓	200.01	847.08	388.80	OOM	40.02	191.91	8.04	3.46
✓		15791.77	19956.06	22467.60				9.69	4291.98
✓	✓	3.85	3.38	2.51	10.04	3.42	3.39	1.62	1.79

F.5 Ablation Study

We conduct two ablation studies to underscore the importance of JIT compilation and the vectorized strategy in accelerating recourse generation. First, we first disable the JIT compilation to evaluate its impact. Moreover, we highlight the significance of the vectorized strategy by running the sequential generation strategy. Table 4 presents the runtime comparison of eight recourse methods in ReLax with these two ablations on the adult datasets. Crucially, disabling JIT compilation results in an average slowdown of ∼84.1X slower on average, which in turn, underscores the importance of JIT compilation. Furthermore, running the sequential generation strategy leads to a dramatic increase in runtime in an average slowdown of ∼4754.2X. This result emphasizes the limitations of sequential generation strategies (commonly used in existing recourse libraries), and the importance of vectorization in speeding up the recourse generation.

F.6 Comparison with CARLA

We conducted an experiment with VanillaCF on the adult dataset using the CARLA library [36]. Table 5 presents the validity and proximity results for the adult dataset. However, it is crucial to note that the results of ReLax and CARLA cannot be directly compared due to CARLA's limitations in handling multi-class categorical features. CARLA only supports binary categorical features, whereas ReLax is capable of handling multi-valued categorical features (see Section F.1).

Table 5: Results of VanillaCF on the adult dataset from CARLA.

Dataset	**VanillaCF**	
	Val.	Prox.
Adult	0.7893	1.149