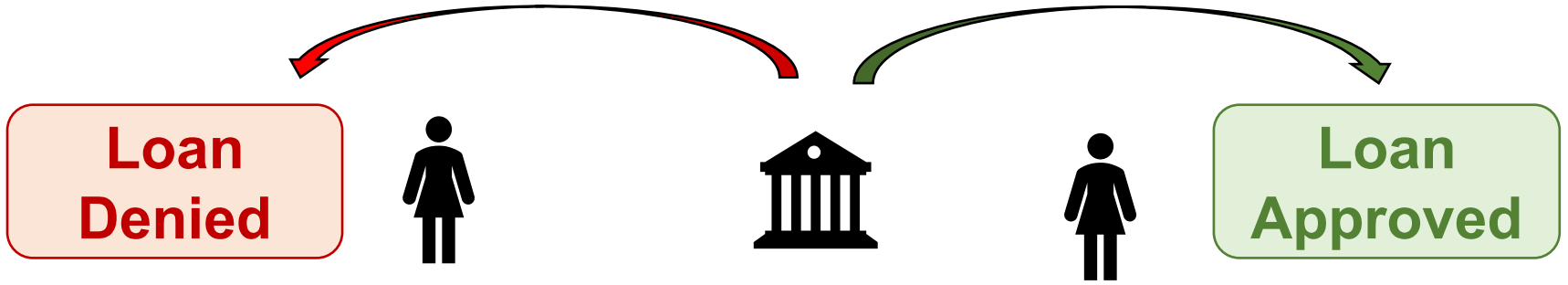
CounterNet: End-to-End Training of Prediction Aware Counterfactual Explanations

Hangzhi Guo[1] Thanh H. Nguyen[2] Amulya Yadav[1]

[1]The Pennsylvania State University

[2]University of Oregon

Counterfactual Explanation (Recourse)

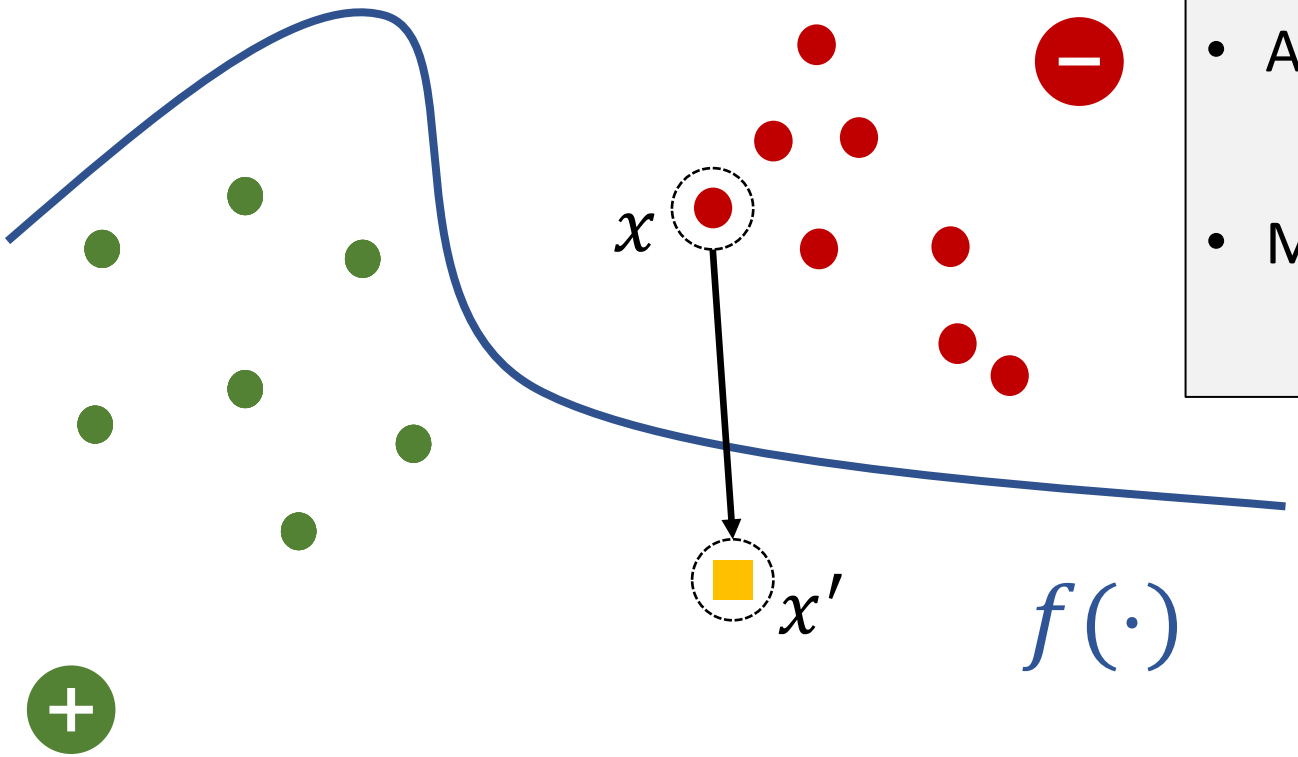


Feature	Value
Age	24
Amount	$20,000
Duration	36
FICO	600
Income	2,500

Counterfactual Explanation

Value
24
$16,000 (– $4,000)
36
650 (+ 50)
2,500

Counterfactual Explanation (Recourse)



Criteria of a good counterfactual explanation

- A valid counterfactual

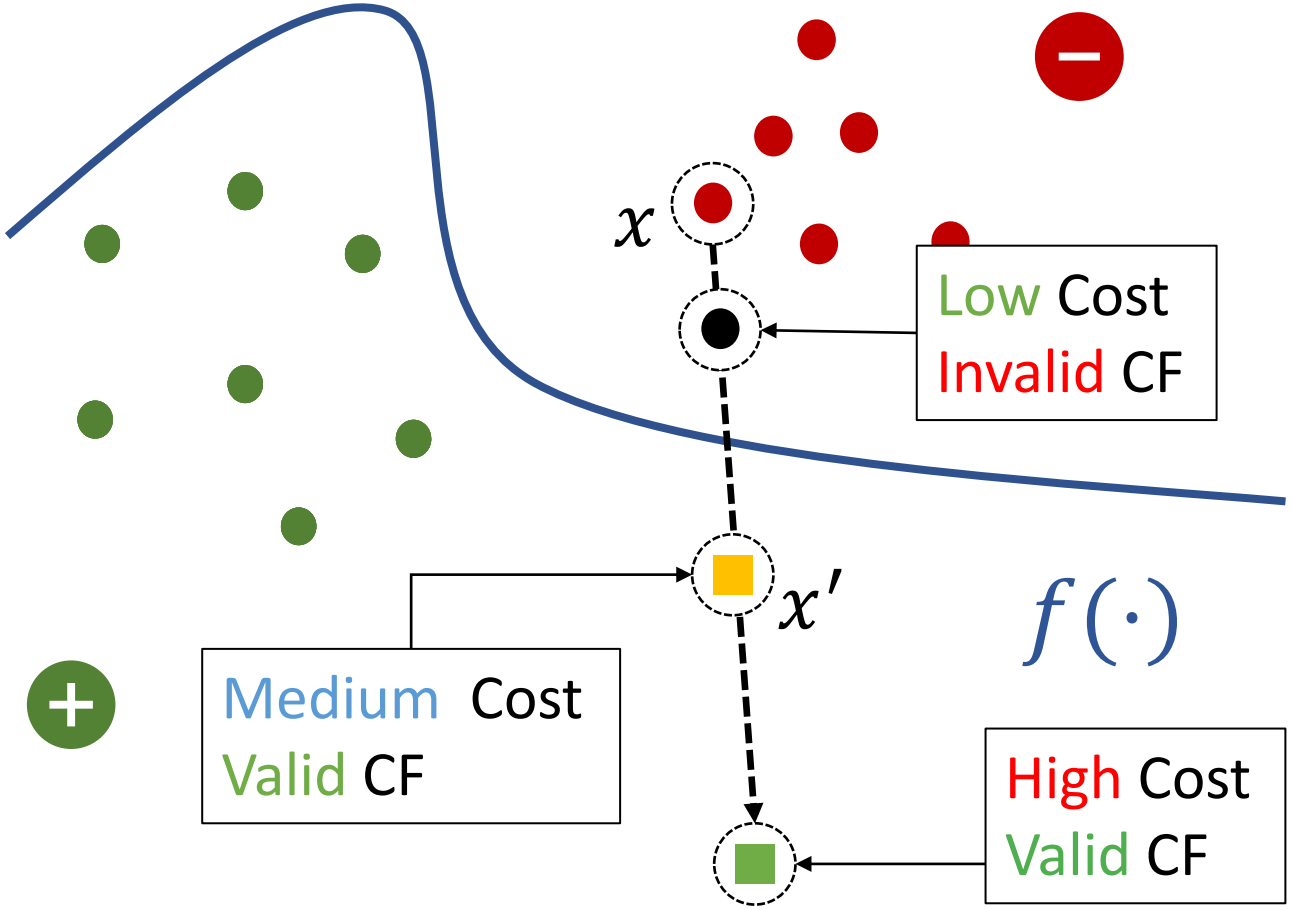
$$f(x') = y^{\text{target}}$$

- Minimal cost of changes

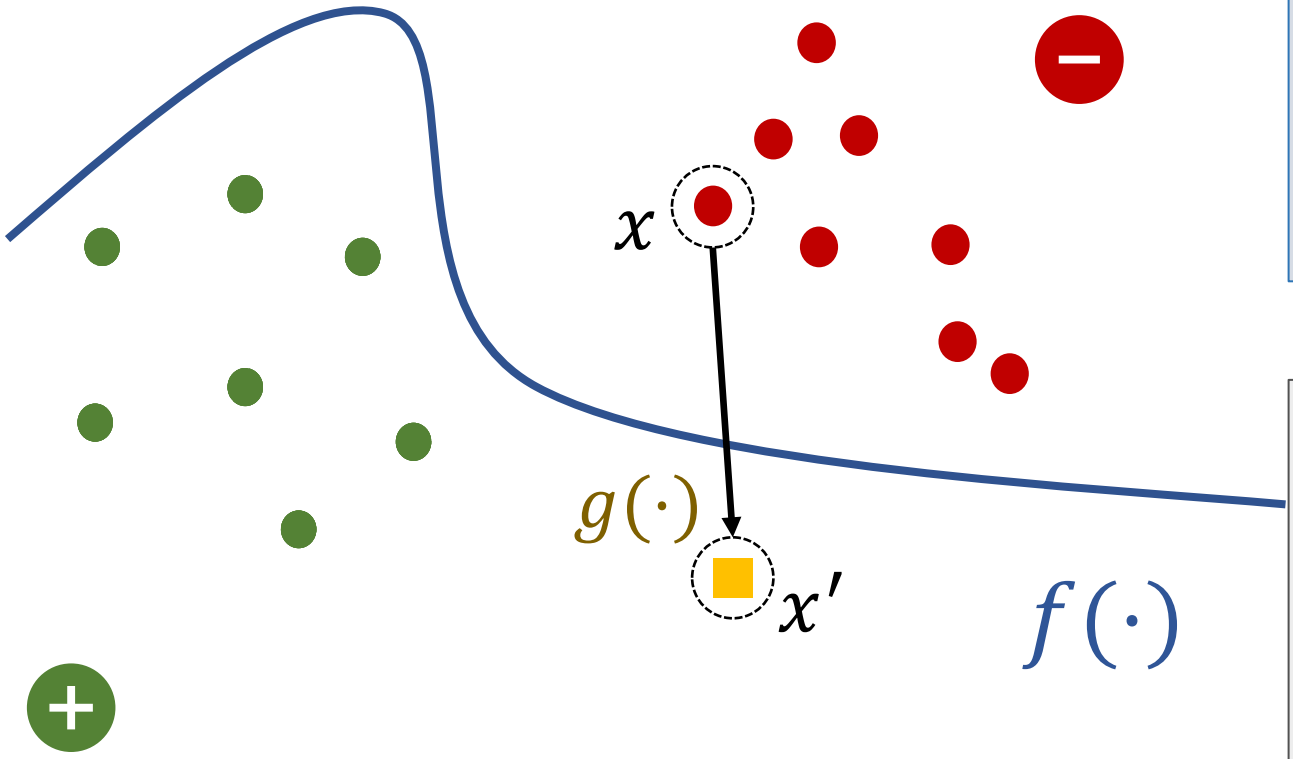
$$\min c(x, x')$$

Balancing the **cost-invalidity trade-off** is important.

Cost-Invalidity Trade-Off



Existing Methods



Non-parametric Post-hoc Methods:

$$\min_{x'} \underbrace{\mathcal{L}\left(f\left(x^{\text{cf}}\right), y^{\text{target}}\right)}_{\text{validity}} + \lambda \cdot \underbrace{c(x, x')}_{\text{cost of changes}}$$

Parametric Post-hoc Methods:

$$\min_{\theta_g} \underbrace{\mathcal{L}(f(x'), y^{\text{target}})}_{\text{validity}} + \lambda \cdot \underbrace{c(x, x')}_{\text{cost of changes}}$$

where $x' = g_{\theta_g}(x)$

Existing Methods

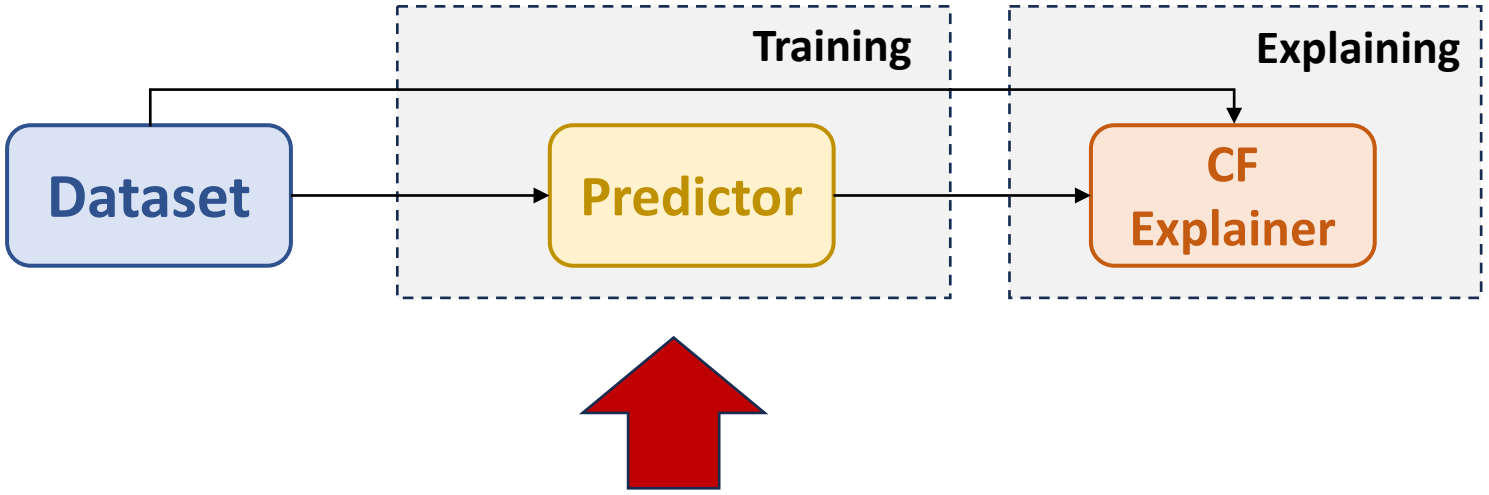
Non-parametric Post-hoc Methods:

$$\min_{\theta_g} \underbrace{\mathcal{L}(f(x^{\text{cf}}), y^{\text{target}})}_{\text{validity}} + \lambda \cdot \underbrace{c(x, x^{\text{cf}})}_{\text{cost of changes}}$$

where $x^{\text{cf}} = g_{\theta_g}(x)$

Existing methods are **post-hoc** (*explain **after** predict*).

Existing Methods are Post-Hoc

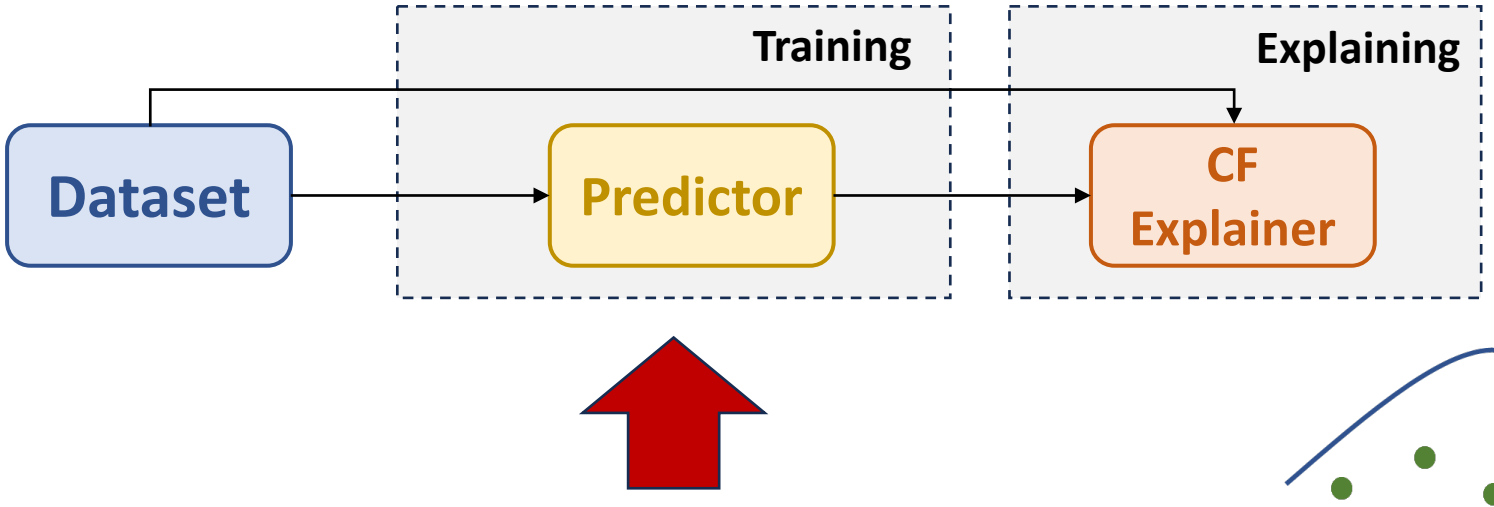


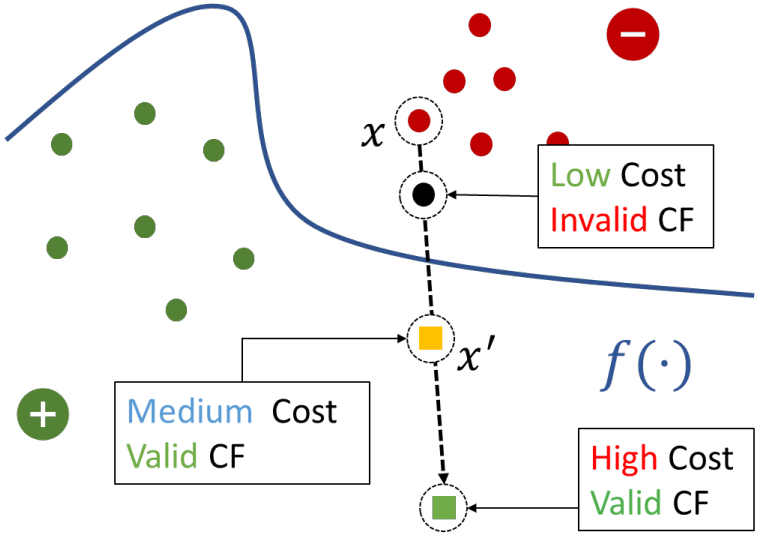
Assuming **black-box** models

EU-GDPR enforces the *"Right to Explanation"*

ML developers want to provide explanations along with predictions.

Existing Methods are Post-Hoc

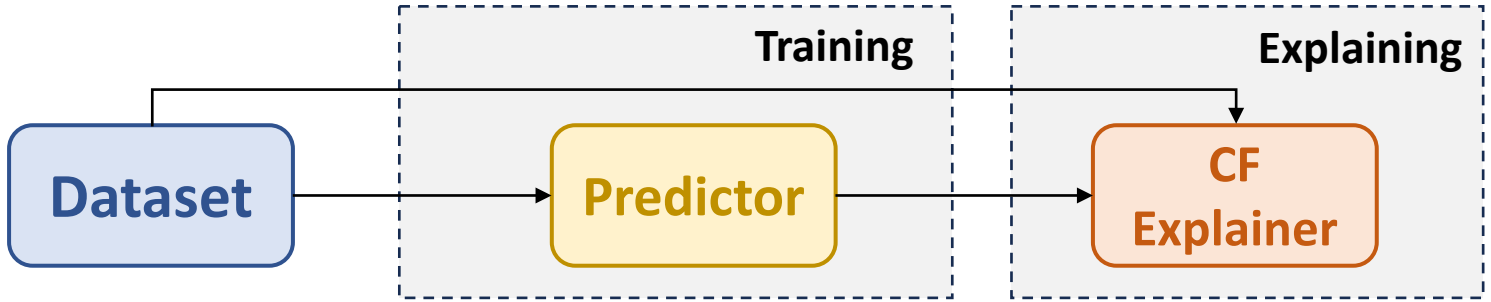




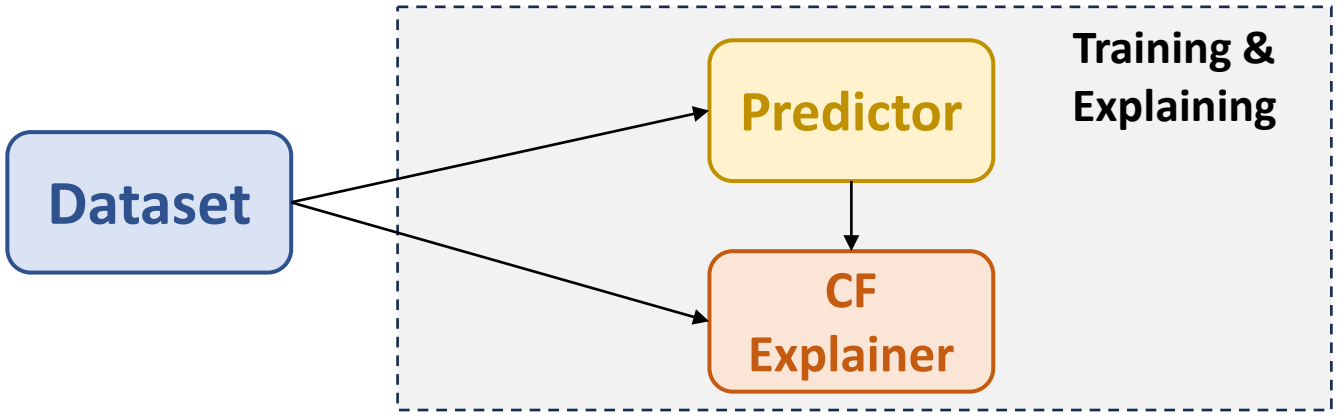
Assuming **black-box** models becomes overly limiting

- Does not properly balance the cost-invalidity trade-off.
- Runs prohibitively slow.

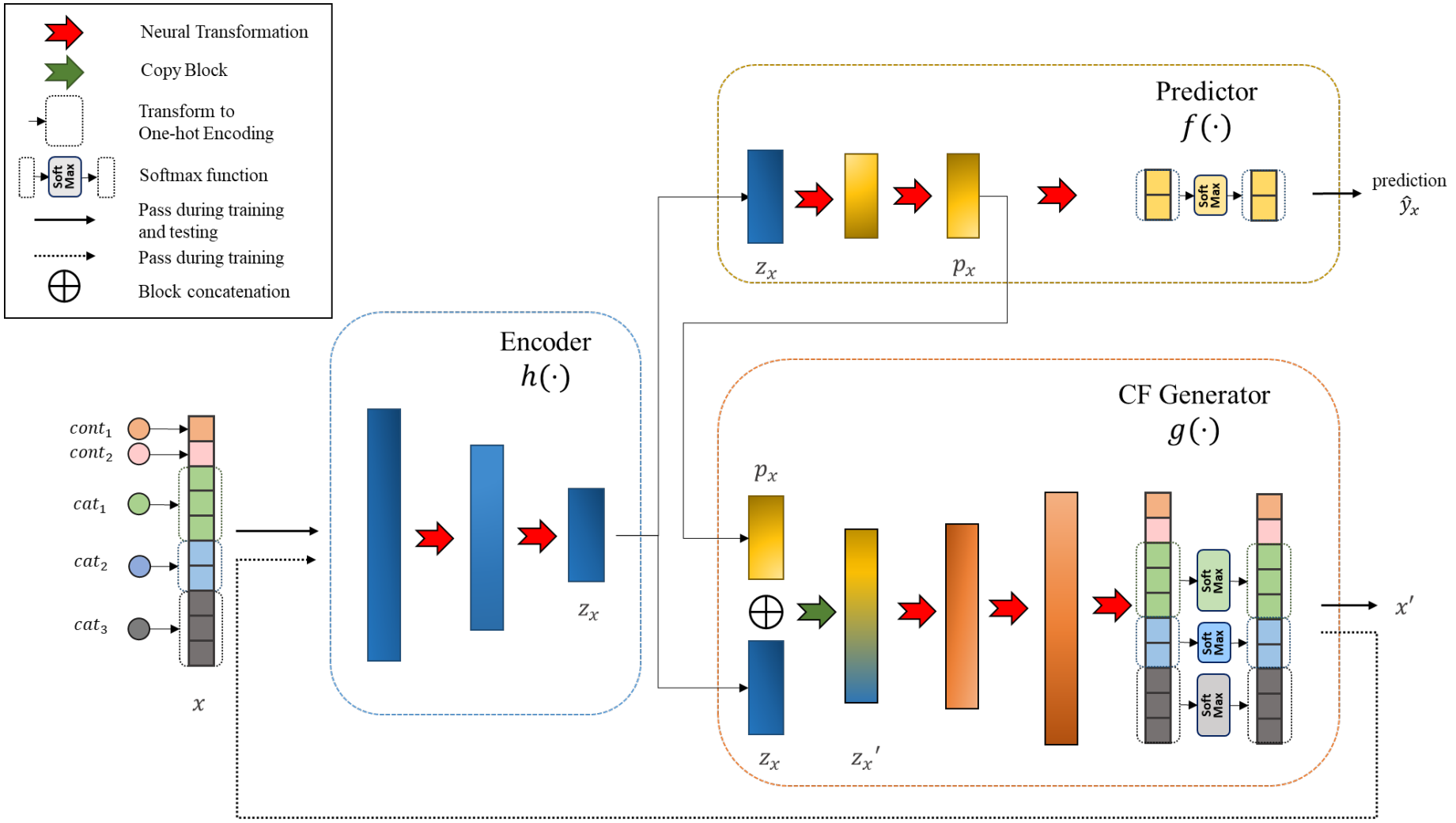
Existing Methods are Post-Hoc



Our End-to-End Approach: CounterNet



Architecture



Objective Function

$$\mathcal{L}_1 = \frac{1}{n}\sum_{i=1}^{n}\left(y_i - \hat{y}_{x_i}\right)^2$$

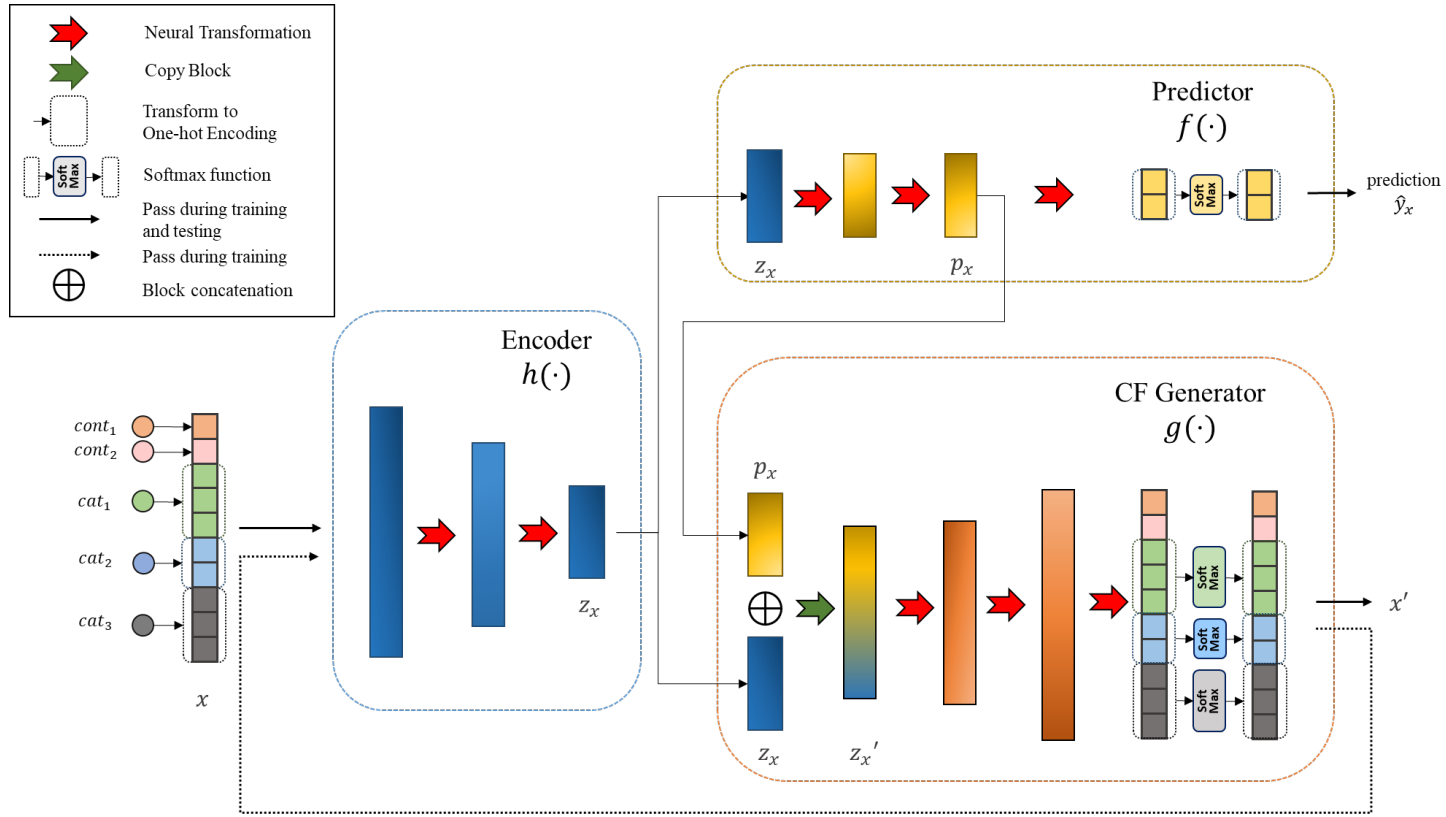
→ Prediction Loss

$$\mathcal{L}_2 = \frac{1}{n}\sum_{i=1}^{n}\left(\hat{y}_{x_i} - \left(1 - \hat{y}_{x_i'}\right)\right)^2$$

→ Validity Loss

$$\mathcal{L}_3 = \frac{1}{n}\sum_{i=1}^{n}(x_i - x_i')^2$$

→ Change of Cost Loss

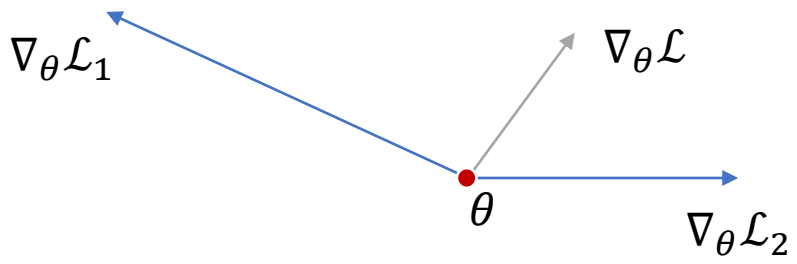


$$\arg\min_{\theta} \lambda_1 \cdot \mathcal{L}_1 + \lambda_2 \cdot \mathcal{L}_2 + \lambda_3 \cdot \mathcal{L}_3$$

Two Issues in Training

Issue ONE: Poor Convergence

LEMMA 3.1 (DIVERGENT GRADIENT PROBLEM). *Let* $\mathcal{L}_1 = \|y - \hat{y}_x\|_2$, *and* $\mathcal{L}_2 = \|\hat{y}_x - (1 - \hat{y}_{x'})\|_2$, *assuming that* $x' \to x$, $0 < \hat{y}_x < 1$, y *is a binary label, and* $|\hat{y}_x - y| < 0.5$, *then* $\nabla\mathcal{L}_1 \cdot \nabla\mathcal{L}_2 < 0$. *(See proof in Appendix A.1)*

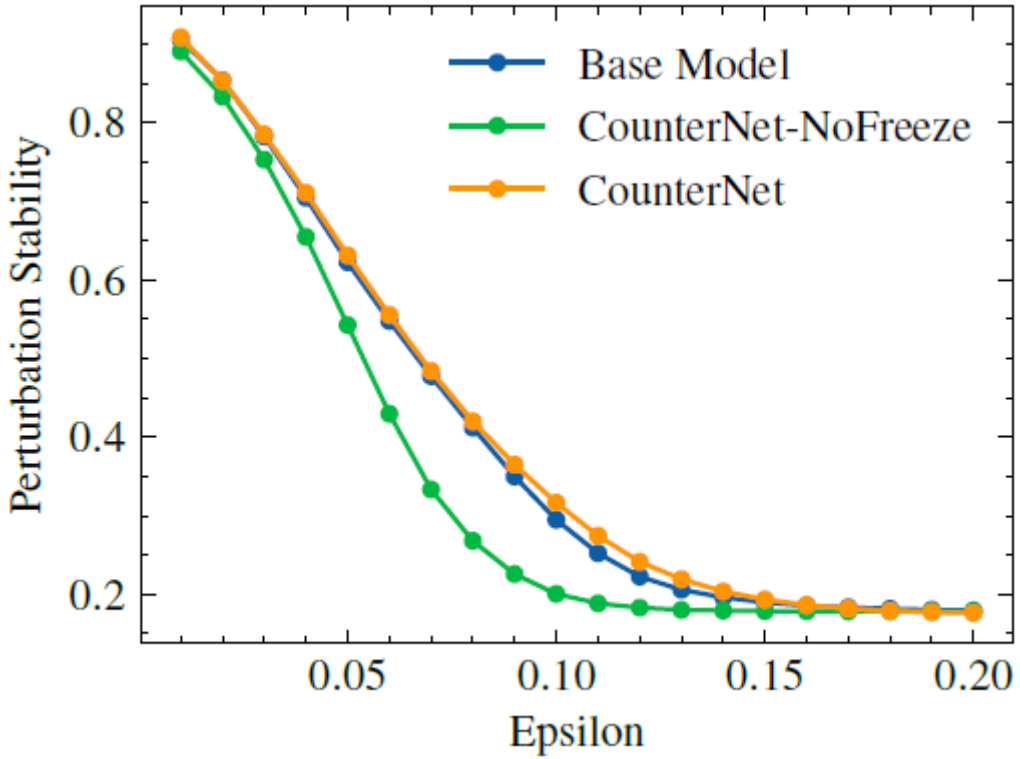


- Divergent gradient of $\nabla_\theta\mathcal{L}_1$ and $\nabla_\theta\mathcal{L}_2$ leads to training instability.

Issue TWO: Adversarial Robustness

- Optimizing validity loss $\mathcal{L}_2$ w.r.t. predictors θ_f leads to decreased adversarial robustness

LEMMA 3.2 (LIPSCHITZ CONTINUITY). *Suppose f is a locally Lipschitz continuous function parameterized by θ_f, then it satisfies $|f_{\theta_f}(x) - f_{\theta_f}(x')| \leq K \|x - x'\|_2$, where the Lipschitz constant of f is $K = \sup_{x' \in \mathbb{B}(x,\epsilon)} \{\|\nabla f_{\theta_f}(x')\|_2\}$. Let $\mathcal{L}_2 = \left\| f_{\theta_f}(x) - (1 - f_{\theta_f}(x')) \right\|_2$, assuming that $x' \to x$, $0 < f_{\theta_f}(\cdot) < 1$, $f_{\theta_f}(x) \to y$, and y is a binary label, then minimizing $\mathcal{L}_2$ w.r.t. θ_f increases the Lipschitz constant K. (See proof in Appendix A.2)*

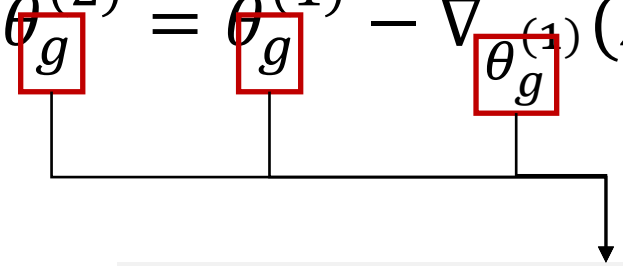


Block-Wise Gradient Descent

For each batch of m data points $\{x_{(i)}, y_{(i)}\}_m$,

① $\theta^{(1)} = \theta^{(0)} - \nabla_{\theta^{(0)}}(\lambda_1 \cdot \mathcal{L}_1)$

② $\theta_g^{(2)} = \theta_g^{(1)} - \nabla_{\theta_g^{(1)}}(\lambda_2 \cdot \mathcal{L}_2 + \lambda_3 \cdot \mathcal{L}_3)$



🔨 Issue One

- Distribute to two stages

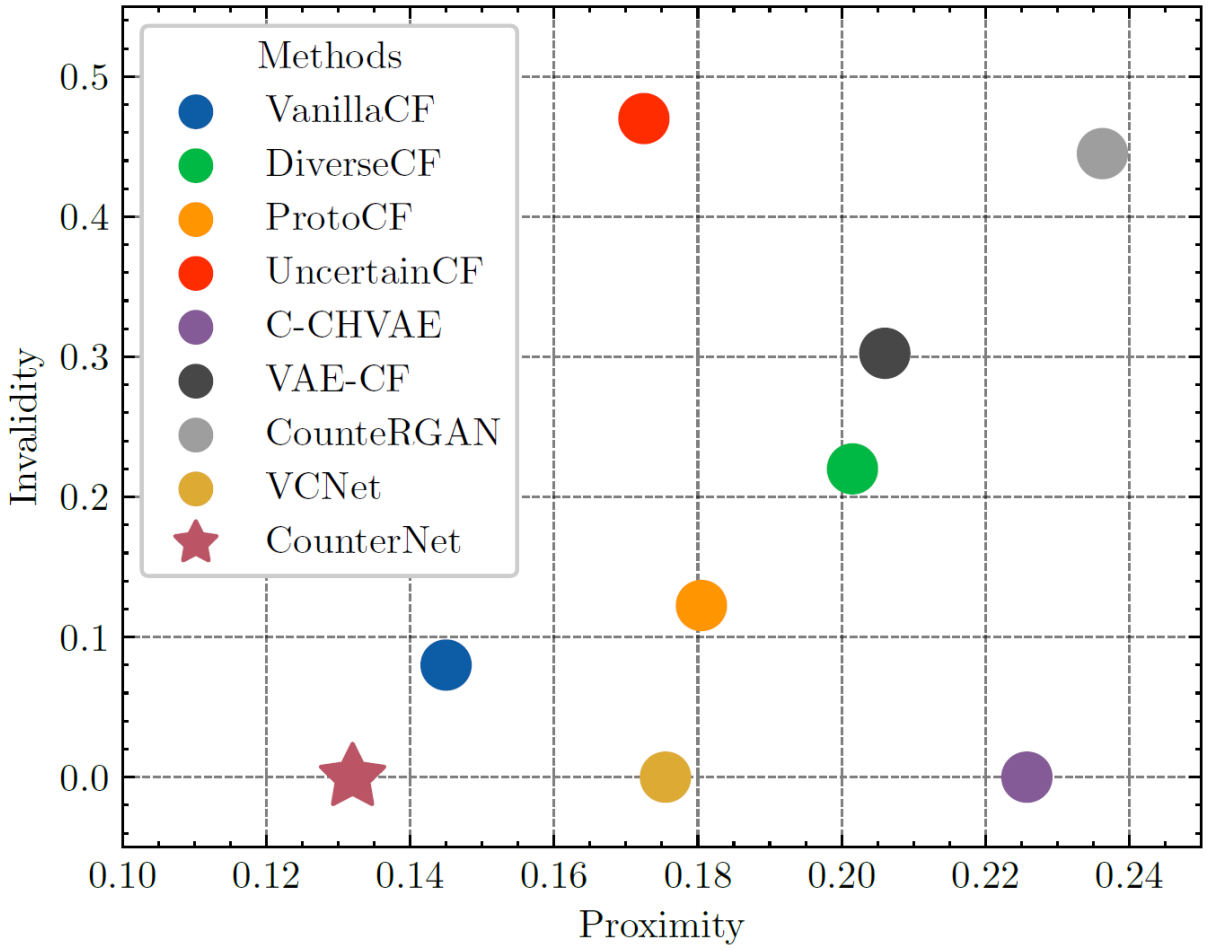
🔨 Issue Two

- Only optimizing θ_g at Stage ②

CounterNet is best-performing method

Method	Adult				Credit				HELOC				OULAD			
	Val.	Prox.	Spar.	Man.	Val.	Prox.	Spar.	Man.	Val.	Prox.	Spar.	Man.	Val.	Prox.	Spar.	Man.
VanillaCF	0.76	.202	**.556**	*0.57*	0.92	**.123**	.841	0.59	**1.00**	.154	.883	0.71	**1.00**	.101	.762	1.30
DiverseCF	0.54	.276	.662	1.16	**1.00**	.264	.918	1.68	0.90	.149	*.434*	1.34	0.68	.117	**.565**	2.51
ProtoCF	0.59	.250	.648	0.62	0.92	.197	.855	0.82	**1.00**	.168	.805	*0.56*	**1.00**	.107	.754	1.46
UncertainCF	0.36	.307	.713	1.23	0.62	.155	**.217**	0.80	0.55	.130	**.161**	0.94	0.59	.098	.734	2.23
C-CHVAE	**1.00**	.281	.721	0.94	**1.00**	.357	.853	1.85	**1.00**	.155	.790	0.81	**1.00**	.110	.797	2.11
VAE-CF	0.66	.287	.734	1.03	0.13	.201	.756	0.62	**1.00**	.221	.893	1.04	**1.00**	.115	.586	2.19
CounteRGAN	0.78	.327	.698	2.21	0.39	.260	.687	2.03	**1.00**	.271	.509	2.23	0.43	.087	.587	2.15
VCNet	**1.00**	.291	.755	**0.19**	**1.00**	.162	.939	**0.16**	**1.00**	.154	.786	**0.39**	**1.00**	.095	.903	1.33
CounterNet	**1.00**	**.196**	*.644*	0.64	**1.00**	*.132*	.912	*0.56*	**1.00**	**.125**	.740	*0.56*	**1.00**	**.075**	.725	**0.87**

CounterNet is best-performing method



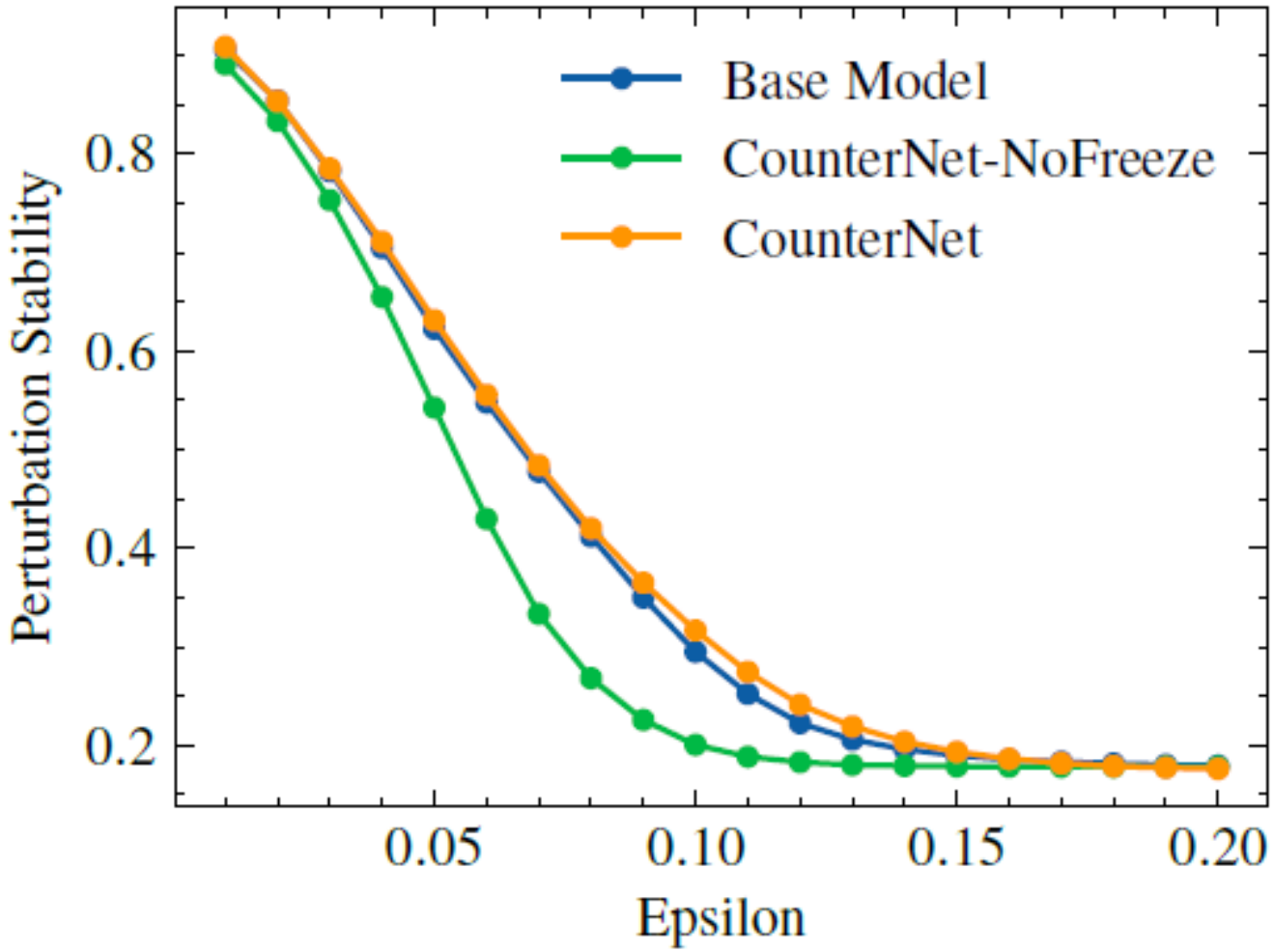
CounterNet runs faster than other baselines

Method	Adult	Credit	HELOC	OULAD
VanillaCF	1432.09	1358.26	1340.42	1705.93
DiverseCF	4685.39	3898.43	3921.72	5478.17
ProtoCF	2348.21	2056.01	1956.71	2823.29
UncertainCF	379.95	60.80	7.91	6.81
C-CHVAE	3.28	568.28	2.68	4.79
VAE-CF	1.72	1.28	1.48	1.84
CounteRGAN	1.96	1.77	1.59	2.40
VCNet	1.39	1.23	1.13	1.81
CounterNet	**0.64**	**0.39**	**0.44**	**0.79**

CounterNet matches predictive accuracy

Dataset	Base Model	CounterNet
Adult	0.831	0.828
Credit	0.813	0.819
HELOC	0.717	0.716
OULAD	0.934	0.929

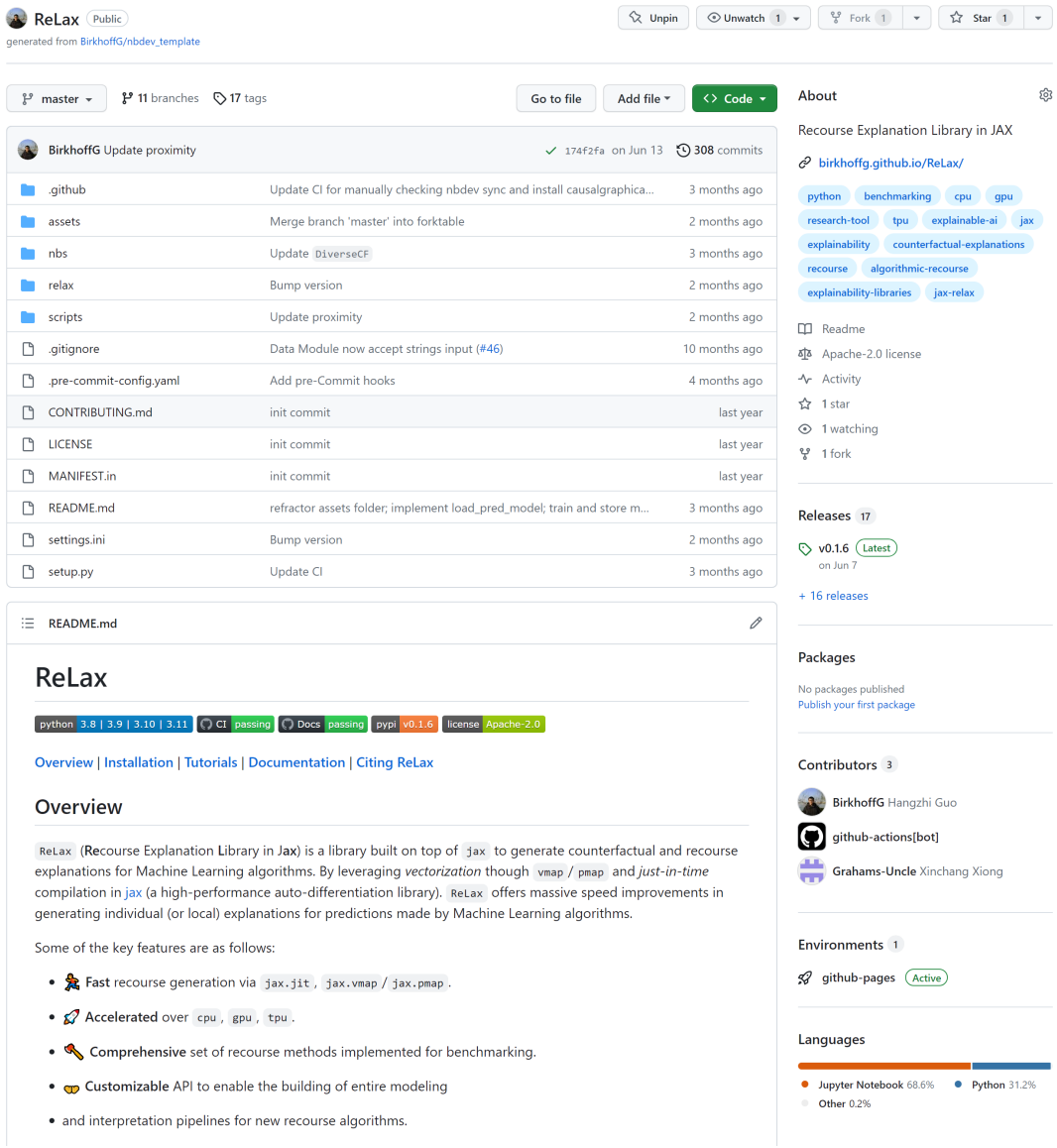
CounterNet does not suffer from increased adversarial vulnerability



Key Insights

- *Post-hoc explainability can be sub-optimal and overly limiting in counterfactual explanations.*
- *CounterNet represents a first step towards developing end-to-end counterfactual explanation system*
  - *Distribution shift, diversity, causality, other data modality...*

GitHub: https://github.com/BirkhoffG/ReLax



ReLax Public

generated from BirkhoffG/nbdev_template

Unpin | Unwatch 1 | Fork 1 | Star 1

master | 11 branches | 17 tags | Go to file | Add file | Code

BirkhoffG Update proximity — 174f2fa on Jun 13 — 308 commits

.github	Update CI for manually checking nbdev sync and install causalgraphica...	3 months ago
assets	Merge branch 'master' into forktable	2 months ago
nbs	Update DiverseCF	3 months ago
relax	Bump version	2 months ago
scripts	Update proximity	2 months ago
.gitignore	Data Module now accept strings input (#46)	10 months ago
.pre-commit-config.yaml	Add pre-Commit hooks	4 months ago
CONTRIBUTING.md	init commit	last year
LICENSE	init commit	last year
MANIFEST.in	init commit	last year
README.md	refractor assets folder; implement load_pred_model; train and store m...	3 months ago
settings.ini	Bump version	2 months ago
setup.py	Update CI	3 months ago

README.md

ReLax

python 3.8 | 3.9 | 3.10 | 3.11 | CI passing | Docs passing | pypi v0.1.6 | license Apache-2.0

Overview | Installation | Tutorials | Documentation | Citing ReLax

Overview

ReLax (Recourse Explanation Library in Jax) is a library built on top of jax to generate counterfactual and recourse explanations for Machine Learning algorithms. By leveraging *vectorization* though vmap / pmap and *just-in-time* compilation in jax (a high-performance auto-differentiation library). ReLax offers massive speed improvements in generating individual (or local) explanations for predictions made by Machine Learning algorithms.

Some of the key features are as follows:

- **Fast** recourse generation via jax.jit, jax.vmap / jax.pmap.
- **Accelerated** over cpu, gpu, tpu.
- **Comprehensive** set of recourse methods implemented for benchmarking.
- **Customizable** API to enable the building of entire modeling
- and interpretation pipelines for new recourse algorithms.

About

Recourse Explanation Library in JAX

birkhoffg.github.io/ReLax/

python | benchmarking | cpu | gpu | research-tool | tpu | explainable-ai | jax | explainability | counterfactual-explanations | recourse | algorithmic-recourse | explainability-libraries | jax-relax

Readme | Apache-2.0 license | Activity | 1 star | 1 watching | 1 fork

Releases 17

v0.1.6 Latest on Jun 7

+ 16 releases

Packages

No packages published

Publish your first package

Contributors 3

BirkhoffG Hangzhi Guo

github-actions[bot]

Grahams-Uncle Xinchang Xiong

Environments 1

github-pages Active

Languages

Jupyter Notebook 68.6% | Python 31.2% | Other 0.2%

Check out ReLax,
our new open-source *recourse explanation* library at GitHub.

GitHub: https://github.com/BirkhoffG/counternet

Thank you!

Hangzhi Guo

✉

hangz@psu.edu

Thanh H. Nguyen

✉

thanhhng@cs.uoregon.edu

Amulya Yadav

✉

amulya@psu.edu