CERTIFIED COUNTERFACTUALS FOR NEURAL NETWORKS

Anonymous authors
Paper under double-blind review

ABSTRACT

Counterfactual explanations aim to identify minimal changes that alter a model's prediction. Unlike adversarial perturbations, however, these changes should correspond to plausible, interpretable, and actionable modifications of the input. Existing approaches broadly fall into two categories: data-based and model-based methods. Data-based methods search among reference points, favoring plausibility but potentially sacrificing proximity and disclosing individual reference points. Model-based methods require query-time model access and generally provide no formal guarantee that the target prediction persists throughout a neighborhood of the returned point or that the point lies in a well-supported region of the input space. We introduce CertCF, a counterfactual explanation method for neural network classifiers that bridges these two families by linking counterfactual generation with neural network verification. CertCF uses a reference dataset to construct a certified under-approximation of the target-class preimages as a union of convex regions, and uses this structure at query time to enable fast, plausible, and valid-by-design counterfactual search. Additionally, it can guarantee bounded robustness to input perturbations and restrict changes to a designated subset of features. We evaluate CertCF on seven tabular datasets commonly used in the counterfactual explanations literature, comparing it against four representative methods, and show that CertCF achieves 100% validity and strong robustness while remaining competitive on the main quality metrics.

1 INTRODUCTION

Machine learning models increasingly inform decisions that affect individuals, making the interpretation of their predictions essential. This need has motivated the field of explainable AI, which develops methods aimed at making model predictions more understandable.

Explainability methods can be distinguished along two complementary dimensions. The first concerns scope: global methods describe a model's general behavior, whereas local methods explain a particular prediction. The second concerns the source of interpretability: intrinsic approaches rely on models that are understandable by design, whereas post-hoc approaches construct explanations for an already-trained model (Guidotti et al., 2018). Local post-hoc explanations therefore provide information about a specific prediction without requiring the predictive model itself to be interpretable.

Within this taxonomy, counterfactual explanations constitute a prominent family of local post-hoc methods (Wachter et al., 2018). Given an input and its prediction, they identify how the input would need to change for the model to produce a desired alternative outcome, thereby answering the question: *What would need to change to obtain a different prediction?* The desired outcome is formalized as a condition on the model output. For example, if a credit-scoring model predicts that a loan application will be rejected, a counterfactual explanation may identify changes in the applicant's financial attributes under which the model would instead predict approval. This can help the applicant interpret the adverse prediction through concrete changes and, when those changes are feasible and actionable, use them as guidance toward a more favorable model outcome.

From an optimization perspective, counterfactual generation resembles a targeted adversarial attack, since both seek a nearby input producing a specified alternative output (Pawelczyk et al., 2022). Their goals are nevertheless different: adversarial attacks exploit model sensitivity, whereas counterfactuals seek meaningful alternatives for explanation or recourse. In the lending example, feasibility excludes

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

inputs such as a negative number of credit accounts. Actionability permits recommendations such as reducing current debt but excludes altering age or past defaults. Plausibility favors profiles resembling those observed among applicants, while sparsity favors changing only a few attributes. Finally, robustness requires the target prediction to persist under small measurement errors, such as a minor discrepancy in reported income (Jiang et al., 2024b). Because these properties partly compete, no method can optimize all of them simultaneously.

Two broad families can be distinguished according to what they require at query time: data-based and model-based methods. Data-based methods rely on precomputed model predictions (reference points) and retrieval structures, avoiding model access at query time (Poyiadzi et al., 2020). Their query-time cost is therefore independent of model size and depends instead on the stored data and retrieval procedure. Directly returning reference points favors plausibility but may sacrifice proximity and disclose personal information.

Model-based methods instead explore the input space by repeatedly evaluating or differentiating through the model at query time (Laugel et al., 2017; Mothilal et al., 2020). This permits direct optimization of proximity and easier inclusion of additional constraints, but makes query-time cost dependent on both model complexity and the search procedure. Although each returned candidate can be validated by evaluating the model, conventional model-based methods provide no formal guarantee throughout a neighborhood of the returned candidate. Verification-based variants can establish formal validity or robustness guarantees, but generally encode the network and solve a new verification or optimization problem for every query (Karimi et al., 2020; Jiang et al., 2024a). Since exact verification of ReLU networks is NP-complete, repeatedly solving these query-specific problems can become computationally prohibitive in practice (Katz et al., 2017).

This raises two related questions. First, how can the cost of formal verification be shifted offline and amortized across many queries? Second, how can this enable model-size-independent online counterfactual search over continuous regions with certified robustness, without requiring stored reference points to be returned?

CertCF answers these questions by performing formal verification during its offline phase and storing the results as a reusable certified atlas. The model and a reference dataset are used to select reference points as anchors, around which LiRPA constructs certified regions as convex under-approximations of the corresponding target-class preimages (Xu et al., 2020; Zhang et al., 2025). At query time, CertCF retrieves nearby certified regions from the atlas for the target class and projects the query onto each of them to find the closest point satisfying feasibility and actionability constraints. Online computation therefore depends on the number and geometry of the retrieved regions rather than on model size, and requires neither further model evaluations nor access to the full reference dataset. Because the projection operates over continuous certified regions, CertCF does not require returning a reference point, thereby reducing the direct exposure of stored examples. Membership in a certified region guarantees target validity, while shrinking the region inward by δ before projection certifies robustness to all L_p perturbations of radius δ.

Our main contributions are: (i) we introduce CertCF, a counterfactual generation method that transforms certified regions obtained from preimage under-approximations into a reusable geometric search space, amortizing neural network verification across multiple queries; (ii) we formulate online generation as constrained projection onto retrieved certified regions, requiring no further model evaluations. For a fixed input space, its query-time cost is independent of model size. This formulation provides target validity by construction, supports feasibility and actionability constraints, and can certify robustness to bounded input perturbations; (iii) we evaluate CertCF against four counterfactual baselines on seven tabular datasets, returning target-valid counterfactuals for all 4,773 queries while achieving strong empirical robustness and plausibility. Ablations assess CertCF's internal mechanisms, while additional experiments evaluate its scalability.

2 Background & Related Work

Counterfactual reasoning has long been studied through possible-world semantics, in which a counterfactual is evaluated relative to worlds that are closest to the actual one (Lewis, 2013). A foundational formulation for automated decision systems was later provided by Wachter et al. (2018),

who cast counterfactual explanation as finding a minimal change to an input that produces a desired alternative outcome without requiring access to the model's internal structure.

Definition 1 (Counterfactual Explanation) *Let $f : \mathcal{X} \to \mathbb{R}^m$ be a classifier that assigns a score $f_c(x)$ to each class $c \in \{1, \ldots, m\}$, with predicted class $\hat{y}(x) = \arg\max_c f_c(x)$. Given a query $x_q \in \mathcal{X}$ and a desired target class $t \neq \hat{y}(x_q)$, a counterfactual explanation is a solution*

$$x_{cf} \in \arg\min_{x \in \mathcal{X}} d(x_q, x) \qquad \textit{subject to} \qquad \hat{y}(x) = t, \tag{1}$$

where d measures the cost of changing x_q into x.

Counterfactual generation has since been studied across diverse settings, including linear classification (Ustun et al., 2019), image classification (Thiagarajan et al., 2021), and graph-structured prediction (Ma et al., 2022).

2.1 COUNTERFACTUALS

Following the query-time distinction introduced earlier, data-based methods construct counterfactuals from reference points or from structures built over them. The simplest approach, which we refer to as Nearest Neighbor (NN), returns the nearest reference point assigned to the target class. This guarantees validity with respect to the model predictions stored for the data and favors plausibility, but restricts the search to reference points, potentially sacrificing proximity and disclosing sensitive information. FACE (Poyiadzi et al., 2020) instead constructs a neighborhood graph over the data and searches for low-cost paths through high-density regions, explicitly prioritizing feasible and actionable transitions over direct endpoint proximity. Its results can, however, be highly sensitive to the graph construction, connectivity parameters, and density and confidence thresholds.

Among model-based methods, Growing Spheres (Laugel et al., 2017) samples expanding shells around the query and subsequently removes superfluous feature changes, requiring only forward pass predictions but explicitly modeling neither data density nor feasibility. DiCE (Mothilal et al., 2020) uses gradients to jointly optimize target-class loss, proximity, and diversity under user-defined feature constraints, requiring iterative optimization per query and enforcing target validity through a soft loss rather than a formal constraint.

Generative methods learn representations intended to favor plausible counterfactuals. C-CHVAE (Pawelczyk et al., 2020) performs a query-specific search in a VAE latent space, whereas VCNet (Guyomard et al., 2022) and CounteRGAN (Nemirovsky et al., 2021) train generators for direct, low-latency generation. Their results depend on the learned representation and provide no certified validity or robustness guarantee.

Verification-based methods replace soft validity objectives with formal constraints. MACE (Karimi et al., 2020) encodes the model, distance, and optional plausibility and diversity requirements as logical formulae, then binary-searches the distance threshold through SAT or SMT solving. This provides pointwise validity and approximate distance optimality, but requires repeated solver calls over the full encoded model for every query, making generation expensive and increasingly difficult to scale as model complexity grows.

Robust counterfactual methods address different sources of uncertainty (Jiang et al., 2024b). PROBE (Pawelczyk et al., 2023) and CROCO (Guyomard et al., 2023) model noisy implementation of recommended changes, whereas Maragno et al. (Maragno et al., 2024) construct target-valid regions under bounded feature perturbations; unlike CertCF, these methods perform query-specific optimization over the predictive model. PROPLACE (Jiang et al., 2024a) and APΔS (Marzari et al., 2026) instead target bounded model-parameter changes, a complementary robustness notion; PROPLACE additionally requires training data and repeated MILP solves for every query.

A related line of formal-XAI research explains predictions by identifying minimal feature subsets. An abductive explanation identifies feature values sufficient to preserve the current prediction, whereas a contrastive explanation identifies features whose modification can change it (Ignatiev et al., 2019; Marques-Silva & Ignatiev, 2022; Bassan & Katz, 2023). Their verification queries may also return prediction-changing inputs, as in VeriX (Wu et al., 2023), but these are not optimized for a specified target class, proximity, feasibility, or robustness. Computing subset-minimal explanations entails repeated verification and is computationally hard (Barceló et al., 2020; Bassan et al., 2026);

direct per-query adaptation can therefore become prohibitive as input dimensionality and model complexity grow. Network abstraction and refinement (Bassan et al., 2025) and LiRPA-based abstract perturbation domains (Boumazouza et al., 2026) improve scalability, but a substantial gap to efficient counterfactual generation remains.

2.2 Preimage Approximation

Many neural-network verification methods begin with a specified input region and determine whether all corresponding outputs satisfy a desired property. Preimage analysis instead asks which inputs produce a specified target output.

Definition 2 (Restricted preimage) *Given a neural network $f : \mathbb{R}^d \to \mathbb{R}^m$, an output set $O \subseteq \mathbb{R}^m$, and a bounded input region $C \subseteq \mathbb{R}^d$, the preimage of O restricted to C is*

$$f_C^{-1}(O) = \{x \in C : f(x) \in O\}. \tag{2}$$

Because the exact preimage can be non-convex and difficult to represent, preimage methods construct sound under- or over-approximations of it. An under-approximation contains only inputs guaranteed to satisfy the property, which makes it suitable for constructing certified counterfactual search regions.

Linear Relaxation-based Perturbation Analysis (LiRPA) derives affine bounds on neural-network outputs over a bounded input region (Zhang et al., 2018; Xu et al., 2020). It does so by replacing nonlinear activations with sound linear relaxations and propagating the resulting bounds through the network. In particular, for a scalar output specification g, backward LiRPA can produce an affine lower bound

$$a^\top x + b \le g(x), \qquad \forall x \in C. \tag{3}$$

PREMAP (Zhang et al., 2025) uses these bounds to under-approximate preimages of polyhedral output sets. It expresses each output constraint as a scalar specification and intersects the input region with the halfspaces on which the corresponding affine lower bounds are nonnegative. The result is a sound under-approximation and is convex whenever the input region is convex. PREMAP can tighten this approximation by partitioning input regions or splitting nonlinear neurons. CertCF instead applies the construction locally around multiple reference anchors and stores the resulting certified regions as a reusable atlas.

To our knowledge, no existing counterfactual method reuses certified continuous input regions across queries without online model or reference-dataset access. CertCF addresses this gap by storing locally certified regions as an atlas for targeted counterfactual search.

3 Method

We consider a trained classifier $f : \mathcal{X} \to \mathbb{R}^m$ that assigns a score $f_c(x)$ to each class $c \in \{1, \dots, m\}$, with predicted class $\hat{y}(x) = \arg\max_c f_c(x)$. Given a query $x_q \in \mathcal{X}$ and a target class $t \neq \hat{y}(x_q)$, our objective is to generate a counterfactual assigned to class t that is close to x_q and satisfies the required feasibility and actionability constraints. Figure 1 summarizes the execution flow of CertCF, which we describe below by following its offline and online phases in order.

3.1 Offline Phase: Certified-Atlas Construction

The offline phase uses the classifier and a reference dataset to construct a certified atlas: a collection of certified regions whose union under-approximates the preimage of each target class. The reference dataset is a representative collection of inputs used to locate the atlas anchors and initialize their local regions. It may coincide with the training data or be provided separately, and it does not require ground-truth labels because points are grouped according to the classifier predictions. Let $\mathcal{D}_{\text{ref}} = \{x_i\}_{i=1}^n$ denote this dataset, and let $I_t = \{i : \hat{y}(x_i) = t\}$ index its points predicted as class t. Each reference point serves as an anchor for the class predicted by the classifier.

For each anchor x_i associated with class t, CertCF initializes a local L_p ball whose radius is proportional to the distance from the nearest reference point assigned to a different class:

$$\epsilon_i = \alpha \min_{j \notin I_t} \|x_i - x_j\|_p, \tag{4}$$

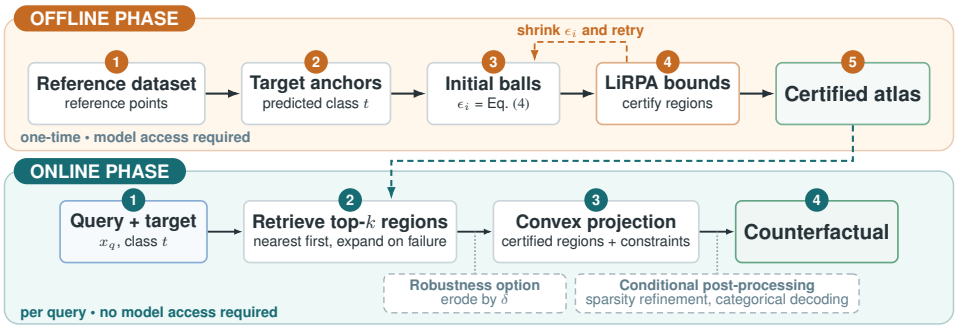


Figure 1: CertCF constructs its certified atlas offline, then answers queries by retrieving and projecting onto stored regions without model access. Dashed boxes denote conditional operations.

where $\alpha > 0$ is a scaling hyperparameter and $p \in [1, \infty]$ specifies the certification norm. The resulting ball $\mathcal{B}_p(x_i, \epsilon_i)$ can be viewed as an initial geometric guess of the local target-class preimage around x_i and determines the input region supplied to LiRPA. Equation (4) is only a data-dependent initialization: it does not estimate the distance to the decision boundary or certify the entire ball.

To certify class t, we express the target prediction through one-vs-all score margins. Let $u \in \mathbb{R}^m$ denote a generic vector of classifier scores. Given a strictly positive margin $\gamma > 0$, define the target output set as

$$O_t^\gamma = \{u \in \mathbb{R}^m : u_t - u_c \geq \gamma,\ \forall c \neq t\}\,. \tag{5}$$

Let $C_i = \mathcal{B}_p(x_i, \epsilon_i)$. Applying the CROWN-based LiRPA procedure (Zhang et al., 2018; Xu et al., 2020) over C_i yields affine lower bounds $a_{ic}^\top x + b_{ic} \leq f_t(x) - f_c(x)$ for every $x \in C_i$ and $c \neq t$. CertCF uses these bounds to construct the local certified region

$$\mathcal{P}_i = \left\{x \in C_i : a_{ic}^\top x + b_{ic} \geq \gamma,\ \forall c \neq t\right\}. \tag{6}$$

Figure 2 illustrates the resulting intersection in two dimensions.

By soundness of the LiRPA bounds, $\mathcal{P}_i \subseteq C_i \cap f^{-1}(O_t^\gamma)$, so every point in $\mathcal{P}_i$ has target-class margin at least $\gamma > 0$ and is therefore guaranteed to be assigned to class t.

LiRPA looseness may produce a nonempty certified region that does not contain its anchor; this does not invalidate the certificate. CertCF nevertheless requires anchor inclusion to retain a known valid feature configuration, particularly for one-hot categorical features. If $x_i \notin \mathcal{P}_i$, it retries at most R times with $\epsilon_i \leftarrow \beta\epsilon_i$, $\beta \in (0, 1)$, and retains the first region containing x_i, or excludes the anchor.

Let $J_t \subseteq I_t$ denote the indices of the anchors successfully certified within the retry budget. We define the class-t atlas and its geometric coverage as

$$\mathcal{A}_t = \{(x_i, \mathcal{P}_i) : i \in J_t\}\,, \tag{7}$$

$$\mathcal{U}_t = \bigcup_{i \in J_t} \mathcal{P}_i \subseteq f^{-1}(O_t^\gamma). \tag{8}$$

The atlas stores each region with its anchor for query-time retrieval. Because it contains finitely many local under-approximations, $\mathcal{U}_t$ is sound but incomplete: failure to find a counterfactual in the atlas does not imply that none exists in the full target-class preimage. Once constructed, the atlas supports online search without the classifier or full reference dataset. Certificates are independent across anchors, enabling data-parallel construction; Appendix F analyzes its scalability, and Algorithm 1 gives the complete procedure.

3.2 Online Phase: Counterfactual Search

The online phase aims to find a certified counterfactual for a query x_q and target class t by searching the stored atlas, without accessing the classifier or the reference dataset. Let $\mathcal{C}(x_q) \subseteq \mathcal{X}$ denote the admissible set induced by domain-level feasibility constraints and any query-specific actionability constraints. Let $q \in [1, \infty]$ denote the norm used to measure query-to-counterfactual distance. Both

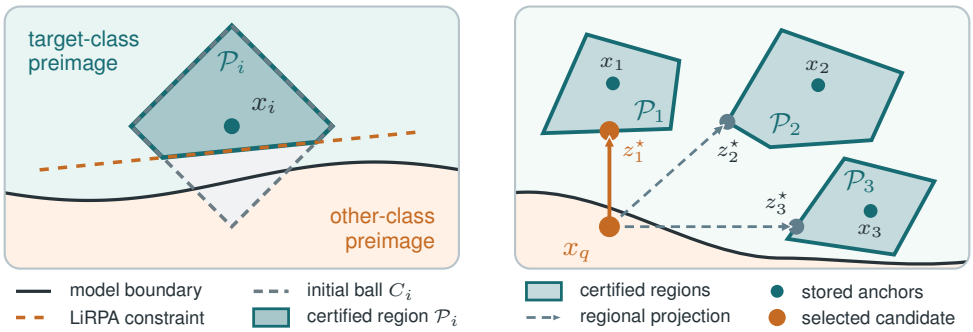


Figure 2: CertCF geometry in two dimensions. (a) LiRPA clips the initial L_1 ball C_i into the certified region $\mathcal{P}_i$. (b) CertCF projects x_q independently onto retrieved regions and selects $z_1^\star$. Model boundaries are illustrative and are not accessed online.

the certification norm p and the projection norm q are user-selectable and need not coincide. The ideal atlas search solves

$$x_{\mathrm{cf}}^\star \in \arg\min_{x \in \mathcal{U}_t \cap \mathcal{C}(x_q)} \|x - x_q\|_q. \tag{9}$$

Solving Equation (9) over every region can be expensive. CertCF therefore orders the entries of $\mathcal{A}_t$ by the distance $\|x_i - x_q\|_q$ between x_q and their anchors, and initially considers the k nearest regions. Let $K_t^k(x_q) \subseteq J_t$ denote their indices. Anchor distance is used only for retrieval: candidates are obtained by projecting onto the associated regions rather than by returning the anchors themselves. If none of the first k regions yields a feasible candidate, CertCF examines the remaining regions in increasing anchor-distance order until a candidate is found or the atlas is exhausted.

If a robustness radius $\delta \geq 0$ is requested, CertCF replaces each retrieved region $\mathcal{P}_i$ with its L_p erosion

$$\mathcal{P}_i^\delta = \{x \in \mathcal{P}_i : \mathcal{B}_p(x, \delta) \subseteq \mathcal{P}_i\}. \tag{10}$$

For $\delta = 0$, the region is unchanged. For $\delta > 0$, every point in $\mathcal{P}_i^\delta$ remains inside $\mathcal{P}_i$ under any perturbation Δx satisfying $\|\Delta x\|_p \leq \delta$, and therefore retains the target prediction throughout that neighborhood. For the region in Equation (6), the erosion has a closed form: a worst-case L_p perturbation can increase the distance from the anchor by δ and decrease each affine margin by $\delta\|a_{ic}\|_{p^*}$, as follows from the triangle and Hölder inequalities, respectively. Therefore,

$$\mathcal{P}_i^\delta = \left\{x : \|x - x_i\|_p \leq \epsilon_i - \delta, \quad a_{ic}^\top x + b_{ic} \geq \gamma + \delta\|a_{ic}\|_{p^*},\ \forall c \neq t\right\}, \tag{11}$$

where p^* is the dual norm of p; the eroded region is empty when $\delta > \epsilon_i$.

For each retrieved region, CertCF enforces feasibility and actionability by solving a constrained projection. To preserve convexity, continuous constraints are imposed directly, while discrete feature requirements are relaxed during the solve. Let $\mathcal{C}_{\mathrm{cvx}}(x_q)$ denote the resulting convex admissible set. For tabular data, feature bounds and continuous actionability constraints are imposed directly, each mutable one-hot block is relaxed to the probability simplex, and immutable blocks remain fixed. For every $i \in K_t^k(x_q)$, the projected candidate is

$$z_i^\star \in \arg\min_{x \in \mathcal{P}_i^\delta \cap \mathcal{C}_{\mathrm{cvx}}(x_q)} \|x - x_q\|_q. \tag{12}$$

Figure 2(b) illustrates these independent regional projections. This is a convex optimization problem, and projections onto different regions are independent, allowing them to be evaluated concurrently. Appendix F analyzes the resulting online scalability and parallel implementation. Regions for which the constraints are infeasible produce no candidate. The same projection is applied to each additional region examined during fallback. Because the simplex relaxation may produce fractional categorical values, CertCF decodes them into valid one-hot assignments and retains only those that remain within $\mathcal{P}_i^\delta$ and satisfy $\mathcal{C}(x_q)$. A region produces no candidate if none of its decoded assignments passes these checks, ensuring that decoding does not compromise feasibility or certification.

For the L_1 projection used in our experiments, CertCF optionally applies an iteratively reweighted group-L_1 refinement to encourage changes in fewer original features; Appendix G gives the complete update. Let $\mathcal{G}$ contain one singleton group per numerical feature and one group per categorical block. Candidates are compared using

$$S_\lambda(x; x_q) = \|x - x_q\|_1 + \lambda \sum_{g \in \mathcal{G}} \frac{\|x_g - x_{q,g}\|_1}{\|x_g - x_{q,g}\|_1 + \eta}, \tag{13}$$

where $\lambda \geq 0$ controls sparsity and $\eta > 0$ keeps the fractions finite. The second term approximates the number of changed groups; reported proximity remains the unpenalized L_1 distance.

Among the decoded candidates that pass all checks, CertCF returns the one with the smallest selection score. This score is $S_\lambda(x; x_q)$ when sparsity refinement is enabled and $\|x - x_q\|_q$ otherwise. If the initial top-k set yields no candidate, the expansion procedure described above continues; CertCF reports failure only when the atlas is exhausted without recovering an admissible candidate.

Every returned counterfactual lies in $\mathcal{P}_i^\delta \cap \mathcal{C}(x_q)$, guaranteeing target validity, satisfaction of the imposed feasibility and actionability constraints, and, when $\delta > 0$, robustness. The top-k heuristic affects only optimality: an unexamined region may contain a lower-scoring candidate. Searching the complete atlas removes this approximation, whereas sparsity refinement uses Equation (13) instead of pure distance. Appendix D evaluates the heuristic, and Algorithm 2 gives the complete procedure.

4 Experimental Evaluation

Experimental setup. We evaluate CertCF on seven binary tabular datasets commonly used in the counterfactual-explanation literature: Adult (Becker & Kohavi, 1996), COMPAS (Angwin et al., 2016), German Credit (Hofmann, 1994), HELOC (FICO, 2018), Give Me Some Credit (Credit Fusion & Cukierski, 2011), Lending Club (Gupta, 2019; Arora & Kaur, 2020), and Wisconsin Breast Cancer (Wolberg et al., 1993). We focus on binary datasets because counterfactual generation ultimately evaluates a binary target condition: whether the desired outcome is satisfied or not. For multiclass models, a desired class defines the target outcome, while multiple desired classes can be handled by searching the union of their class-specific atlases. Numerical features are standardized and categorical features are one-hot encoded. Unless stated otherwise, we set $p = q = 1$, using the L_1 norm for both offline certification and online projection. For online search, L_1 is aligned with the sparsity objective because it tends to concentrate changes on fewer coordinates, whereas L_∞ may distribute them across many features. We retain at most $10{,}000$ reference points per class and randomly select at most 500 anchors per predicted class for atlas construction. All methods are evaluated using the same trained classifier and the same $4{,}773$ queries across the seven datasets.

Baselines. We compare CertCF with four representative counterfactual methods: Nearest Neighbor (NN), FACE (Poyiadzi et al., 2020), Growing Spheres (GS) (Laugel et al., 2017), and DiCE (Mothilal et al., 2020). NN and FACE represent data-based approaches, respectively using direct retrieval and graph-based search, whereas GS and DiCE represent model-based approaches based on black-box sampling and gradient optimization. This selection covers the principal query-time families considered in Section 2.1 without requiring separately trained generative models. Each method returns one counterfactual per query; complete configurations are reported in Appendix A.

Evaluation protocol and metrics. Unless stated otherwise, metrics are computed in the standardized, one-hot encoded representation. Validity is the fraction of queries for which the target class is reached. Over target-valid counterfactuals, proximity is the mean L_1 distance and sparsity is the mean number of changed coordinates. Empirical robustness measures target-class retention under sampled numerical and categorical perturbations and is distinct from CertCF's formal robustness guarantee. Plausibility is evaluated using a novelty-mode Local Outlier Factor score (Breunig et al., 2000), with smaller values indicating greater data support. Reference-data alignment is assessed for methods that generate new points through the relative proximity ratio ρ, which compares their distance to the nearest target-class reference point with the natural nearest-neighbor distance within that class; $\rho \approx 1$ is preferred. NN and FACE are excluded because they return reference points directly. Motivated by decision-tree representations of transparent and consistent counterfactual actions (Kanamori et al., 2022), our interpretability proxy is the normalized test error of a depth-3 decision tree trained to map

Method	Validity (%) ↑	L_1 prox. ↓	Sparsity ↓	Emp. rob. (%) ↑	Interp. ↓	Time (ms) ↓	Plaus. ↓	RPR (1 best)
Certified atlas								
CertCF	**100.00** *	7.647	8.03 *	**94.34** *	**1.44** *	345.9	**0.065** *	**0.456**
Data-based								
NN	**100.00** *	6.918 *	10.70	90.34	1.83	**0.343** *	0.067	–
FACE	98.06	9.334	12.42	91.26	1.66	2403.2	0.101	–
Model-based								
GS	99.62	**2.565**	**3.72**	78.59	2.12	28.0	0.236	1.930
DiCE	93.17	4.369	6.20	93.30	2.44	1630.0	0.259	2.946

Table 1: Dataset-averaged benchmark results. Bold indicates the best overall mean (ties included); * indicates the best mean among methods requiring no model access at query time (CertCF, NN, and FACE). Arrows indicate the preferred direction; RPR is reported only for methods that generate new points and is best when closest to 1. Bootstrap 95% confidence intervals and per-dataset results are reported in Appendix E.

queries to their counterfactuals. Query time is the mean wall-clock latency per query; DiCE timing uses a separate 20-query, batch-size-one run. Metrics are first computed within each dataset and then averaged equally across datasets. Uncertainty is summarized by percentile-bootstrap 95% confidence intervals across datasets. Full definitions are provided in Appendix E.

Main benchmark. Table 1 reports the dataset-averaged results, while Appendix E provides uncertainty estimates and per-dataset metrics. Among methods requiring no model access at query time, CertCF achieves the best mean sparsity, ties NN at 100% validity, and outperforms FACE in proximity. Across all methods, it also obtains the best mean empirical robustness, plausibility, and interpretability-proxy scores, with lower latency than DiCE and FACE. GS and DiCE retain the strongest proximity and sparsity, but CertCF narrows the sparsity gap, changing 8.03 coordinates on average versus 6.20 for DiCE, and is the only method to reach the target class for all $4{,}773$ queries without relying on direct reference-point retrieval. CertCF therefore moves toward the advantages of model-based search while retaining certified validity and model-independent online operation.

Figure 3(a) condenses these results into model-oriented and data-support aggregates. The model-oriented axis aggregates validity, proximity, sparsity, and empirical robustness, whereas the data-support axis aggregates interpretability and plausibility. Validity and robustness are scaled by their natural $[0, 100]$ ranges, while the remaining metrics are min–max normalized across methods after reversing those for which lower is better. We then average scores within each family and standardize the two aggregates across methods. Query time is excluded because it measures computational cost, while RPR is excluded because it is undefined for NN and FACE. NN and FACE favor data-support properties, whereas GS and DiCE favor model-oriented properties. CertCF attains the highest data-support score while moving toward the model-oriented regime, placing it between the two families.

New-point distributional alignment. Figure 3(b) compares the nearest target-class reference distances of newly generated counterfactuals with the leave-one-out same-class reference distribution. CertCF best matches the reference distribution, with an Earth Mover distance of 0.100, compared with 0.644 for GS and 0.873 for DiCE. NN and FACE are excluded because they return reference points directly. CertCF therefore remains substantially closer to the empirical support than the model-based alternatives without directly retrieving stored examples.

Ablation results. Ablations across the same seven datasets support CertCF's main design choices (Appendix D). At $\alpha = 0.2$, adaptive shrinkage certifies all attempted anchors, compared with a 56.0% macro-averaged yield without it. Projection onto the uncertified initial neighborhoods reaches only 82.3% validity; targeted PGD restores full empirical validity but takes 13.18 seconds per query, versus 0.031 for CertCF, without providing a formal regional guarantee. Finally, top-5 retrieval recovers the exhaustive-search optimum for 99.57% of queries with a mean L_1 gap below 7×10^{-4}.

Scalability results. Across 25 FCNN architectures, increasing model size from 562 to 272,130 parameters raises optimized atlas-construction time from 0.102 to 1.416 seconds, while median

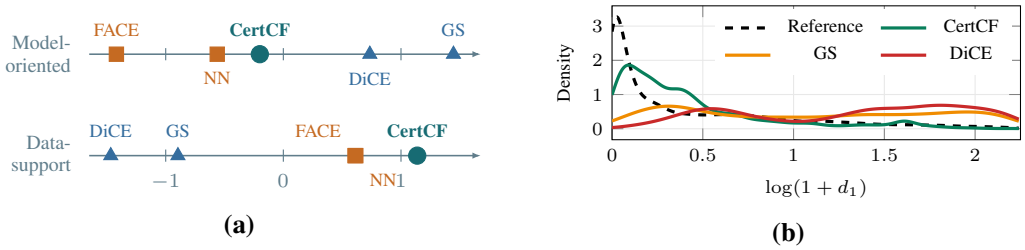


Figure 3: Qualitative comparison of counterfactual methods. (a) Equal-weight aggregates of normalized model-oriented and data-support metrics, standardized across methods; positions are descriptive. (b) Nearest-reference distance distributions for methods generating new points; the dashed curve reports leave-one-out same-class reference distances, and curves are clipped at the 99th percentile.

query latency remains near 87.5 milliseconds and all 25,000 queries remain valid (Appendix F). The CIFAR-10 stress test shows the same separation: increasing ReLU activations from 188,416 to 532,480 raises construction time from 41.79 to 65.61 seconds, while query latency remains near 0.44 seconds and all 300 queries remain valid. Model complexity therefore affects offline certification, whereas online generation depends on the retrieved regions and input dimension.

Limitations. CertCF's finite atlas under-approximates the target-class preimage, so feasible or closer counterfactuals may exist outside its regions. Increasing coverage through additional anchors or tighter regions raises the amortized construction and storage costs, although construction is parallelizable (Appendix F). The main benchmark evaluates empirical robustness with erosion disabled (Appendix C); enabling erosion provides a formal guarantee for a prescribed $\delta > 0$ but reduces the feasible search space. Moreover, quality evaluation is limited to unconstrained binary tabular tasks: the CIFAR-10 experiment (Appendix F.3) tests scalability only, not perceptual plausibility or semantic actionability. Finally, all evaluated classifiers use ReLU activations; other architectures require sound and sufficiently tight LiRPA bounds for their operators.

5 Conclusion and Future Work

We introduced CertCF, a counterfactual explanation method that transforms formal neural-network verification into a reusable certified atlas. CertCF constructs certified under-approximations of target-class preimages offline and searches them online through constrained projection, requiring neither further classifier evaluations nor access to the full reference dataset. Every returned counterfactual is target-valid by construction, while region erosion can additionally provide a formal robustness guarantee for a prescribed L_p perturbation radius.

Across seven binary tabular datasets, CertCF returned target-valid counterfactuals for all 4,773 queries and achieved the best overall mean empirical robustness, plausibility, and interpretability-proxy scores. Among methods requiring no model access at query time, it also obtained the best mean sparsity without relying on direct reference-point retrieval. These results position CertCF between the two main families: it retains the model-independent online operation of data-based methods, but replaces their discrete search over reference points with continuous projection over certified regions. This moves its proximity and sparsity toward those of model-based search, although the model-based methods retain an advantage on these objectives. The ablations support the certification, adaptive-shrinkage, and retrieval mechanisms, while the scalability experiments confirm that increasing model size affects atlas construction rather than online query latency.

Future work could improve atlas coverage and construction cost through adaptive anchor selection, region refinement, and compression. Structural or causal constraints could strengthen plausibility and actionability, while probabilistic preimage bounds (Marzari et al., 2025) could support different confidence levels. Certified latent-space regions may extend the approach to high-dimensional domains, provided that the representation and classifier can be certified jointly. More broadly, CertCF shows how formal verification can be compiled into reusable geometric structures for efficient counterfactual generation with formal guarantees.

AI USE STATEMENT

Generative AI tools were used to assist with literature discovery, manuscript drafting and editing, LaTeX formatting and figures, experimental code, and the interpretation and presentation of results. All AI-assisted content, code, calculations, claims, and citations were reviewed and verified by the authors, who take responsibility for the final content of this work.

REFERENCES

Julia Angwin, Jeff Larson, Surya Mattu, and Lauren Kirchner. Machine bias: There's software used across the country to predict future criminals. and it's biased against blacks. *ProPublica*, 2016. URL https://www.propublica.org/article/machine-bias-risk-assessments-in-criminal-sentencing.

Nisha Arora and Pankaj Deep Kaur. A bolasso based consistent feature selection enabled random forest classification algorithm: An application to credit risk assessment. *Applied Soft Computing*, 86:105936, 2020. doi: 10.1016/j.asoc.2019.105936. URL https://doi.org/10.1016/j.asoc.2019.105936.

Pablo Barceló, Mikaël Monet, Jorge Pérez, and Bernardo Subercaseaux. Model interpretability through the lens of computational complexity. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 33, pp. 15487–15498, 2020.

Shahaf Bassan and Guy Katz. Towards formal XAI: Formally approximate minimal explanations of neural networks. In *Proceedings of the 29th International Conference on Tools and Algorithms for the Construction and Analysis of Systems (TACAS)*, pp. 187–207, 2023. URL https://arxiv.org/abs/2210.13915.

Shahaf Bassan, Yizhak Yisrael Elboher, Tobias Ladner, Matthias Althoff, and Guy Katz. Explaining, fast and slow: Abstraction and refinement of provable explanations, 2025. URL https://arxiv.org/abs/2506.08505.

Shahaf Bassan, Xuanxiang Huang, and Guy Katz. Unifying formal explanations: A complexity-theoretic perspective. In *International Conference on Learning Representations (ICLR)*, 2026.

Barry Becker and Ronny Kohavi. Adult. UCI Machine Learning Repository, 1996. DOI: https://doi.org/10.24432/C5XW20.

Ryma Boumazouza, Raya Elsaleh, Melanie Ducoffe, Shahaf Bassan, and Guy Katz. FAME: Formal abstract minimal explanation for neural networks. In *Proceedings of the 14th International Conference on Learning Representations (ICLR)*, 2026. URL https://arxiv.org/abs/2603.10661.

Markus M. Breunig, Hans-Peter Kriegel, Raymond T. Ng, and Jörg Sander. LOF: Identifying density-based local outliers. In *Proceedings of the 2000 ACM SIGMOD International Conference on Management of Data*, SIGMOD '00, pp. 93–104. ACM, 2000. doi: 10.1145/335191.335388.

Credit Fusion and Will Cukierski. Give me some credit: Kaggle credit scoring dataset. Kaggle dataset, 2011. URL https://www.kaggle.com/competitions/givemesomecredit/data. Accessed: 2026-04-23.

FICO. Explainable machine learning challenge. https://community.fico.com/s/explainable-machine-learning-challenge, 2018. Accessed: 2018.

Riccardo Guidotti, Anna Monreale, Salvatore Ruggieri, Franco Turini, Dino Pedreschi, and Fosca Giannotti. A survey of methods for explaining black box models, 2018. URL https://arxiv.org/abs/1802.01933.

Sparsh Gupta. Lending Club Loan Dataset 2007–2011. Kaggle dataset, 2019. URL https://www.kaggle.com/datasets/imsparsh/lending-club-loan-dataset-2007-2011. Accessed: 2026-05-03.

Victor Guyomard, Françoise Fessant, Thomas Guyet, Tassadit Bouadi, and Alexandre Termier. Vcnet: A self-explaining model for realistic counterfactual generation, 2022. URL https://arxiv.org/abs/2212.10847.

Victor Guyomard, Françoise Fessant, Thomas Guyet, Tassadit Bouadi, and Alexandre Termier. Generating robust counterfactual explanations. In *Machine Learning and Knowledge Discovery in Databases: Research Track*, volume 14171 of *Lecture Notes in Computer Science*, pp. 394–409. Springer Nature Switzerland, 2023. doi: 10.1007/978-3-031-43418-1_24.

Hans Hofmann. Statlog (German Credit Data). UCI Machine Learning Repository, 1994. DOI: https://doi.org/10.24432/C5NC77.

Alexey Ignatiev, Nina Narodytska, and Joao Marques-Silva. Abduction-based explanations for machine learning models. In *Proceedings of the 33rd AAAI Conference on Artificial Intelligence*, pp. 1511–1519, 2019. doi: 10.1609/aaai.v33i01.33011511.

Junqi Jiang, Jianglin Lan, Francesco Leofante, Antonio Rago, and Francesca Toni. Provably robust and plausible counterfactual explanations for neural networks via robust optimisation, 2024a. URL https://arxiv.org/abs/2309.12545.

Junqi Jiang, Francesco Leofante, Antonio Rago, and Francesca Toni. Robust counterfactual explanations in machine learning: A survey, 2024b. URL https://arxiv.org/abs/2402.01928.

Kentaro Kanamori, Takuya Takagi, Ken Kobayashi, and Yuichi Ike. Counterfactual explanation trees: Transparent and consistent actionable recourse with decision trees. In *Proceedings of the 25th International Conference on Artificial Intelligence and Statistics*, volume 151 of *Proceedings of Machine Learning Research*, pp. 1846–1870. PMLR, 2022. URL https://proceedings.mlr.press/v151/kanamori22a.html.

Amir-Hossein Karimi, Gilles Barthe, Borja Balle, and Isabel Valera. Model-agnostic counterfactual explanations for consequential decisions, 2020. URL https://arxiv.org/abs/1905.11190.

Guy Katz, Clark Barrett, David Dill, Kyle Julian, and Mykel Kochenderfer. Reluplex: An efficient smt solver for verifying deep neural networks, 2017. URL https://arxiv.org/abs/1702.01135.

Thibault Laugel, Marie-Jeanne Lesot, Christophe Marsala, Xavier Renard, and Marcin Detyniecki. Inverse classification for comparison-based interpretability in machine learning, 2017. URL https://arxiv.org/abs/1712.08443.

D. Lewis. *Counterfactuals*. Wiley, 2013. ISBN 9781118696415. URL https://books.google.it/books?id=bCvnk3JMvfAC.

Fei Tony Liu, Kai Ming Ting, and Zhi-Hua Zhou. Isolation forest. In *Proceedings of the 2008 Eighth IEEE International Conference on Data Mining*, ICDM '08, pp. 413–422, USA, 2008. IEEE Computer Society. ISBN 9780769535029. doi: 10.1109/ICDM.2008.17. URL https://doi.org/10.1109/ICDM.2008.17.

Jing Ma, Ruocheng Guo, Saumitra Mishra, Aidong Zhang, and Jundong Li. Clear: Generative counterfactual explanations on graphs, 2022. URL https://arxiv.org/abs/2210.08443.

Donato Maragno, Jannis Kurtz, Tabea E. Röber, Rob Goedhart, Ş. Ilker Birbil, and Dick den Hertog. Finding regions of counterfactual explanations via robust optimization. *INFORMS Journal on Computing*, 36(5):1316–1334, 2024. doi: 10.1287/ijoc.2023.0153.

Joao Marques-Silva and Alexey Ignatiev. Delivering trustworthy AI through formal XAI. In *Proceedings of the 36th AAAI Conference on Artificial Intelligence*, pp. 12342–12350, 2022.

Luca Marzari, Manuele Bicego, Ferdinando Cicalese, and Alessandro Farinelli. On the probabilistic learnability of compact neural network preimage bounds, 2025. URL https://arxiv.org/abs/2511.11656.

Luca Marzari, Francesco Leofante, Ferdinando Cicalese, and Alessandro Farinelli. Probabilistically robust counterfactual explanations under model changes. *Artificial Intelligence*, 351:104459, 2026. ISSN 0004-3702. doi: 10.1016/j.artint.2025.104459. URL https://www.sciencedirect.com/science/article/pii/S000437022500178X.

Ramaravind K. Mothilal, Amit Sharma, and Chenhao Tan. Explaining machine learning classifiers through diverse counterfactual explanations. In *Proceedings of the 2020 Conference on Fairness, Accountability, and Transparency*, FAT* '20, pp. 607–617. ACM, January 2020. doi: 10.1145/3351095.3372850. URL http://dx.doi.org/10.1145/3351095.3372850.

Daniel Nemirovsky, Nicolas Thiebaut, Ye Xu, and Abhishek Gupta. Countergan: Generating realistic counterfactuals with residual generative adversarial nets, 2021. URL https://arxiv.org/abs/2009.05199.

Martin Pawelczyk, Klaus Broelemann, and Gjergji Kasneci. Learning model-agnostic counterfactual explanations for tabular data. In *Proceedings of The Web Conference 2020*, WWW '20, pp. 3126–3132. ACM, April 2020. doi: 10.1145/3366423.3380087. URL http://dx.doi.org/10.1145/3366423.3380087.

Martin Pawelczyk, Chirag Agarwal, Shalmali Joshi, Sohini Upadhyay, and Himabindu Lakkaraju. 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*, volume 151 of *Proceedings of Machine Learning Research*, pp. 4574–4594. PMLR, 2022. URL https://proceedings.mlr.press/v151/pawelczyk22a.html.

Martin Pawelczyk, Teresa Datta, Johannes van den Heuvel, Gjergji Kasneci, and Himabindu Lakkaraju. Probabilistically robust recourse: Navigating the trade-offs between costs and robustness in algorithmic recourse. In *International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=sC-PmTsiTB.

Rafael Poyiadzi, Kacper Sokol, Raul Santos-Rodriguez, Tijl De Bie, and Peter Flach. Face: Feasible and actionable counterfactual explanations. In *Proceedings of the AAAI/ACM Conference on AI, Ethics, and Society*, AIES '20, pp. 344–350. ACM, February 2020. doi: 10.1145/3375627.3375850. URL http://dx.doi.org/10.1145/3375627.3375850.

Jayaraman J. Thiagarajan, Vivek Sivaraman Narayanaswamy, Deepta Rajan, Jia Liang, Akshay Chaudhari, and Andreas Spanias. Designing counterfactual generators using deep model inversion. In *Advances in Neural Information Processing Systems*, volume 34, 2021. URL https://proceedings.neurips.cc/paper_files/paper/2021/hash/8ca01ea920679a0fe3728441494041b9-Abstract.html.

Berk Ustun, Alexander Spangher, and Yang Liu. Actionable recourse in linear classification. In *Proceedings of the Conference on Fairness, Accountability, and Transparency*, FAT* '19, pp. 10–19. ACM, January 2019. doi: 10.1145/3287560.3287566. URL http://dx.doi.org/10.1145/3287560.3287566.

Verified Intelligence. auto_lirpa: Automatic linear relaxation based perturbation analysis for neural networks. https://github.com/Verified-Intelligence/auto_LiRPA, 2026. Version 0.7.2.

Sandra Wachter, Brent Mittelstadt, and Chris Russell. Counterfactual explanations without opening the black box: Automated decisions and the gdpr, 2018. URL https://arxiv.org/abs/1711.00399.

William Wolberg, Olvi Mangasarian, Nick Street, and W. Street. Breast cancer wisconsin (diagnostic). UCI Machine Learning Repository, 1993. URL https://archive.ics.uci.edu/dataset/17/breast+cancer.

Min Wu, Haoze Wu, and Clark Barrett. VeriX: Towards verified explainability of deep neural networks. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 36, 2023. URL https://arxiv.org/abs/2212.01051.

Kaidi Xu, Zhouxing Shi, Huan Zhang, Yihan Wang, Kai-Wei Chang, Minlie Huang, Bhavya Kailkhura, Xue Lin, and Cho-Jui Hsieh. Automatic perturbation analysis for scalable certified robustness and beyond, 2020. URL https://arxiv.org/abs/2002.12920.

Huan Zhang, Tsui-Wei Weng, Pin-Yu Chen, Cho-Jui Hsieh, and Luca Daniel. Efficient neural network robustness certification with general activation functions, 2018. URL https://arxiv.org/abs/1811.00866.

Xiyue Zhang, Benjie Wang, Marta Kwiatkowska, and Huan Zhang. Premap: A unifying preimage approximation framework for neural networks, 2025. URL https://arxiv.org/abs/2408.09262.

A HYPERPARAMETER CONFIGURATION

For each method, we report the main hyperparameters used in the benchmark and their final values in Table 2. We focus on the small set of parameters that most directly affect the quality–runtime trade-off. The only exception is FACE, whose behavior depends strongly on the graph radius ϵ. Values that are too small produce fragmented graphs with isolated nodes, while values that are too large yield overly dense graphs. As a result, FACE does not admit a single robust setting across datasets, so ϵ is chosen separately for each dataset and reported in Table 3.

Method	Hyperparameter	Value
CertCF	`alpha`	0.20
	`sparsity_lambda`	1.0
	`k_per_class`	500
	`query_k_candidates`	5
	`adaptive_eps_shrink_factor`	0.5
	`adaptive_eps_max_shrinks`	8
	`sparsity_reweight_iters`	2
DiCE	`diversity_weight`	0.0
	`proximity_weight`	1.0
	`min_iter`	50
FACE	`tp`	0.5
	`epsilon`	Dataset-specific; see Table 3
Growing Spheres	`max_radius`	50.0
	`radius_step`	0.25
	`n_samples_in_layer`	1000

Table 2: Main hyperparameters used in the benchmark. Parameters not shown are kept at their default values.

Dataset	FACE ϵ	Dataset	FACE ϵ
Adult	2.9	Give Me Some Credit	1.1
COMPAS	1.5	Lending Club	2.6
German Credit	3.6	Wisconsin Breast Cancer	4.5
HELOC	3.3		

Table 3: Dataset-specific graph-radius values used for FACE.

The values in Table 2 were chosen to give each method a stable and competitive configuration without performing dataset-specific tuning, except for the FACE graph radius. For CertCF, we set $\alpha = 0.20$ because the ablation in Appendix D shows that, once adaptive shrinkage is enabled, increasing α beyond moderate values has little effect on proximity while preserving stable certified coverage. We set `sparsity_lambda` to 1.0 to favor sparse counterfactuals, accepting the small proximity cost observed in the sparsity ablation. Tuning these hyperparameters separately for each dataset could further improve the reported metrics. However, we deliberately use a single global CertCF configuration to show that the method remains competitive without dataset-specific tuning.

The execution settings used for the reported tabular timings are stated separately because they affect scheduling rather than the optimization objective. Atlas construction uses LiRPA batches of at most 128 anchors, eight CPU workers for the initial-radius computation, and two class-level LiRPA workers. Online generation uses up to eight worker processes for the independent top-5 projections, with one numerical-library thread per process. The process pool is initialized before the measured query loop. The batched and parallel implementation is described and evaluated in Appendix F.4.

Unless stated otherwise, CertCF uses classification margin $\gamma = 10^{-4}$, sparsity constant $\eta = 10^{-3}$, and a 120-second timeout per query. Continuous projections are formulated in CVXPY and solved with CLARABEL, accepting only an `optimal` status, with at most 100 iterations and tolerances 10^{-3}, 10^{-5}, and 10^{-8} for the absolute duality gap, relative duality gap, and feasibility, respectively. One-hot decoding first tests the largest relaxed entry in each block, then uses a beam of width 8 that

considers the three most promising categories per branch and permits at most 32 solver calls before falling back to exact search.

All experiments were executed on a machine equipped with an Intel i7-14700K CPU, 32 GB of DDR5 RAM, and an NVIDIA RTX 4070 SUPER GPU. The complete implementation and experiment configurations are available in the accompanying anonymous repository.[1]

For DiCE, we disable the diversity term by setting `diversity_weight` to 0.0, since the benchmark evaluates a single counterfactual per query rather than a diverse set. The proximity weight is kept at 1.0 to retain the standard validity–proximity trade-off, and `min_iter` is set to 50 to give the gradient optimization enough iterations before early stopping.

For FACE, the original paper does not prescribe a general rule for selecting the graph hyperparameters. We therefore use a graph-connectivity heuristic. For each dataset, we construct radius-neighbor graphs on up to $10{,}000$ training points per class using the L_2 distance used by FACE, and scan $\epsilon \in \{0.1, 0.2, \ldots, 5.0\}$. We select the smallest ϵ such that the largest connected component contains at least 90% of the nodes, the fraction of isolated nodes is at most 5%, and the average degree lies between 5 and 100. This avoids fragmented graphs while preventing excessively dense graphs, which would make shortest-path search expensive and reduce the method to a near-nearest-neighbor search. The resulting dataset-specific values are reported in Table 3.

For Growing Spheres, we use a maximum radius of 50.0 and a radius step of 0.25 so that the expanding search explores a sufficiently large neighborhood without making each shell too coarse. The number of samples per layer is set to 1000, which gives a reasonable compromise between search resolution and runtime, especially in higher-dimensional tabular spaces.

B Classifier Training

Each tabular dataset is randomly split into 80% training, 10% validation, and 10% test data using seed 42. Numerical scaling statistics and, where needed, numerical imputation values are fitted on the training split. The categorical one-hot vocabularies are fitted on the complete cleaned dataset to keep encoded columns consistent across splits.

For each dataset, we train a fully connected neural network (FCNN) with two hidden layers of 64 and 32 neurons, ReLU activations, and dropout ($p = 0.2$). To mitigate class imbalance, we minimize weighted cross-entropy with weights inversely proportional to class frequencies. We use AdamW with weight decay 10^{-4}, batch size 128, and seed 42. A cosine schedule decays the learning rate from 5×10^{-3} to 10^{-6} over 100 epochs. Training runs for the full 100 epochs, after which we retain the checkpoint attaining the lowest validation loss.

C Robustness

Figure 4 evaluates the empirical stability of the returned counterfactuals under random perturbations. The horizontal axis gives the standard deviation σ of the Gaussian noise added to numerical features, while the vertical axis gives the probability of flipping each categorical OHE block to another category. Each cell reports the percentage of perturbed counterfactuals that remain classified as the target class, averaged across datasets; higher values are better.

In this experiment, we do not enable CertCF's erosion mechanism. Hence, the values in the heatmap should be read as post-hoc empirical robustness, not as the certified robustness guarantee of Section 3. If erosion were enabled for a bounded perturbation set matching the chosen robustness radius, CertCF would return only counterfactuals whose entire perturbation neighborhood lies inside a certified target-class polytope, yielding 100% robustness by construction for all perturbations inside that set.

CertCF is the most robust method over most of the grid and degrades smoothly as the perturbation strength increases. DiCE is also competitive and becomes slightly better in the largest-perturbation cells, while NN and FACE show intermediate stability. Growing Spheres is the least robust method, with a sharp drop as soon as numerical perturbations increase. This suggests that CertCF coun-

[1] Anonymous code repository.

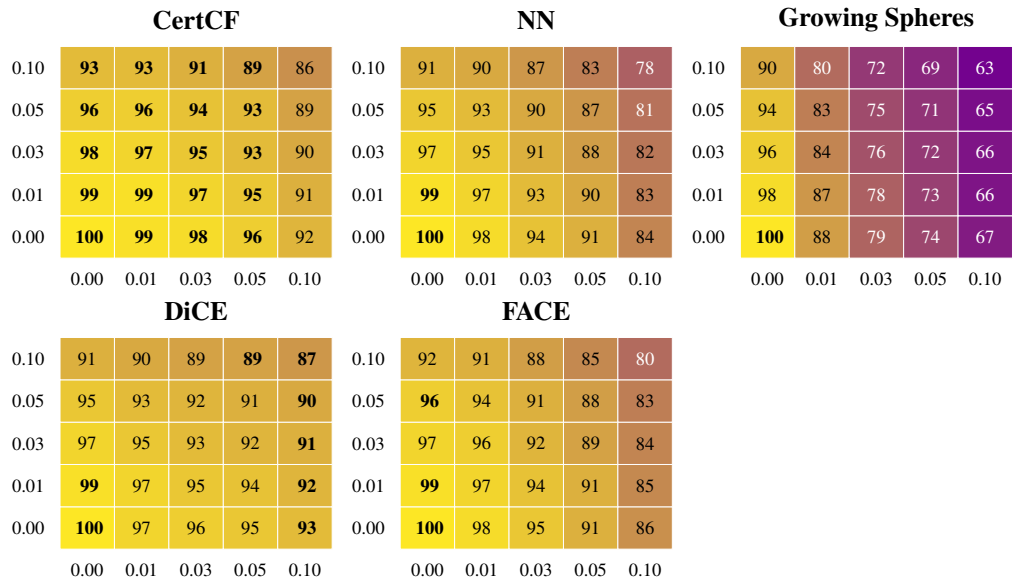


Figure 4: Empirical robustness under random perturbations, averaged across datasets. Columns vary the standard deviation σ of the Gaussian noise applied to numerical features, while rows vary the probability of changing each categorical one-hot block. Each cell reports the percentage of perturbed counterfactuals that retain the target prediction. CertCF's erosion mechanism is disabled in this experiment.

terfactuals tend to lie in more stable target-class regions, rather than merely crossing the decision boundary.

D Ablation Study

All ablation experiments use the same seven tabular datasets, trained classifiers, and preprocessing pipeline as the main evaluation in Section 4. Results are computed separately for each dataset and then macro-averaged, giving every dataset equal weight. Unless the corresponding component is being varied, we use seed 42, L_1 certification and projection, radius multiplier $\alpha = 0.2$, and top-5 region retrieval. We examine adaptive shrinkage, the radius multiplier, the sparsity penalty, the LiRPA backend, the certification mechanism, the projection-objective norm, and the retrieval heuristic.

Adaptive shrinkage. Figure 5 compares atlas construction with and without adaptive shrinkage for $\alpha \in \{0.01, 0.05, \ldots, 0.95, 0.99\}$, using at most 500 randomly sampled anchors per predicted class. Without shrinkage, the macro-averaged certification yield decreases from 99.9% at $\alpha = 0.01$ to 56.0% at the default $\alpha = 0.2$ and 5.4% at $\alpha = 0.99$. Moreover, from $\alpha = 0.3$ onward, at least one dataset loses all certified anchors for one of its classes. Adaptive shrinkage instead certifies every attempted anchor and preserves a complete binary atlas on all seven datasets throughout the tested range. At $\alpha = 0.2$, only 44.0% of anchors require contraction, while the macro-averaged median final-to-initial radius ratio remains 78.6%. Adaptive shrinkage therefore prevents large initial neighborhoods from compromising atlas coverage while retaining their requested size whenever certification already succeeds.

Radius multiplier. Figure 6 reports the macro-averaged L_0 sparsity and L_1 proximity across the seven tabular datasets as α varies under different sparsity penalties. Increasing α yields modest improvements in L_1 proximity up to approximately $\alpha = 0.3$, after which the curves largely plateau. At fixed λ, L_0 sparsity varies little with α and is governed primarily by the sparsity penalty. Thus, CertCF is not particularly sensitive to the precise value of α within the tested range; the default $\alpha = 0.2$ already lies near the observed performance plateau.

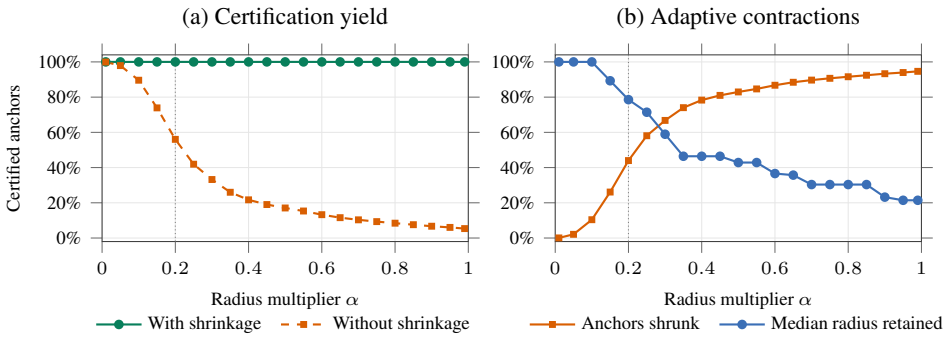


Figure 5: Effect of adaptive shrinkage across the seven tabular datasets. The left panel reports the macro-averaged fraction of attempted anchors yielding a certified region, with and without shrinkage. With shrinkage enabled, the right panel reports the fraction of anchors requiring contraction and the macro-averaged median ratio between the final and initial radii. The dotted line marks the default radius multiplier $\alpha = 0.2$.

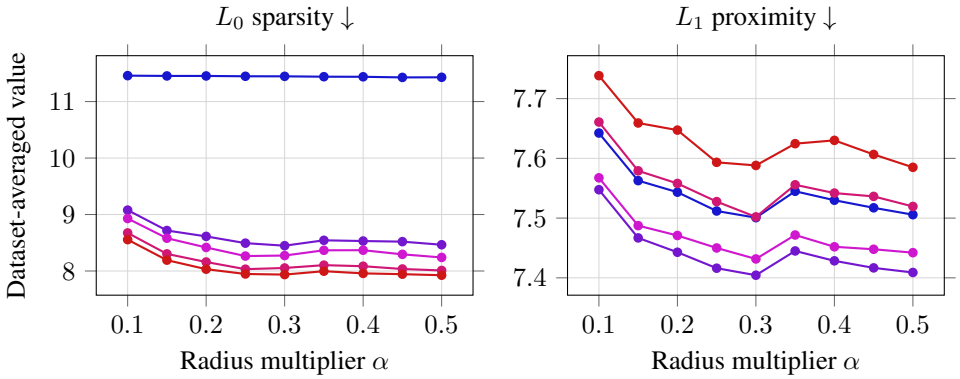


Figure 6: Macro-averaged effect of the radius multiplier α across the seven tabular datasets under sparsity penalties $\lambda \in \{0, 0.1, 0.2, 0.5, 1.0\}$, ordered from blue to red.

Sparsity penalty. Figure 7 reports the macro-averaged effect of the group-aware sparsity penalty λ across the seven tabular datasets. For every tested value of α, introducing a small penalty sharply improves L_0 sparsity: at the default $\alpha = 0.2$, increasing λ from 0 to 0.1 reduces the average number of changed features from 11.45 to 8.61, approximately 25%. Larger penalties yield diminishing sparsity gains, reaching 8.03 at $\lambda = 1$, while progressively trading L_1 proximity for sparsity. We therefore use $\lambda = 1$ in the main evaluation to favor sparse counterfactuals while accepting a modest proximity cost.

LiRPA backend. We compare standard CROWN, optimized CROWN, and α-CROWN using 300 prediction-balanced anchors, up to 200 queries per dataset, and three independent repetitions, with the sparsity penalty disabled. All backends certify every selected anchor and achieve full counterfactual validity. Optimized CROWN and α-CROWN increase the mean retained fraction of the final regions from 0.975 to 0.984, but this small geometric improvement does not improve counterfactual proximity or sparsity. In contrast, their macro-averaged offline build time is more than ten times that of standard CROWN. We therefore use standard CROWN throughout the main evaluation.

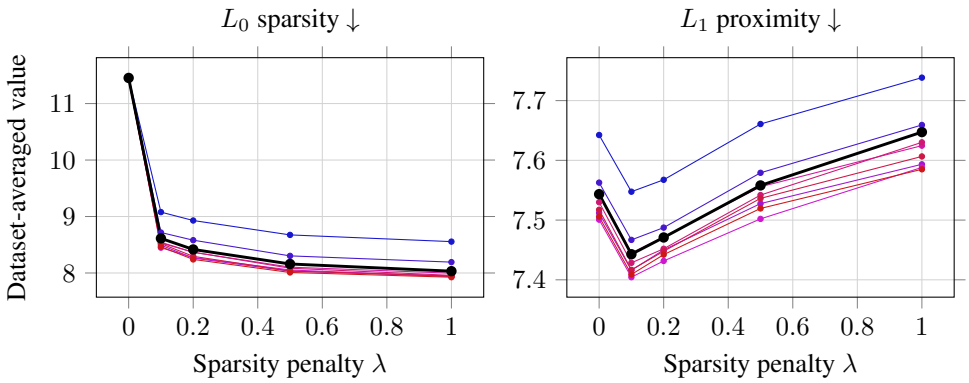


Figure 7: Macro-averaged effect of the group-aware sparsity penalty λ across the seven tabular datasets. Curves correspond to $\alpha \in \{0.10, 0.15, \ldots, 0.50\}$; the thicker black curve denotes the default $\alpha = 0.20$. The remaining curves range from blue ($\alpha = 0.10$) to red ($\alpha = 0.50$).

Backend	Build (s) ↓	Retained ↑	Validity (%) ↑	L_1 ↓	L_0 ↓
CROWN	**0.137**	0.975	100.0	**7.942**	**11.639**
Optimized CROWN	1.406	0.984	100.0	7.962	11.652
α-CROWN	1.403	0.984	100.0	7.962	11.652

Table 4: Effect of the LiRPA backend, macro-averaged across the seven tabular datasets. Build times are first averaged over three repetitions within each dataset. The retained fraction estimates, using 1,024 uniformly sampled points, the portion of each final L_1 ball contained in its certified polytope.

Certification mechanism. We isolate the contribution of replacing the initial neighborhoods from Equation (4) with LiRPA-certified regions. Across the seven tabular datasets, all methods use the same reference data, up to 1,000 prediction-balanced anchors, up to 500 queries per dataset, $\alpha = 0.2$, and top-5 retrieval, with the sparsity penalty disabled. *Anchor-Ball* projects each query directly onto the retrieved initial L_1 balls without evaluating the classifier. *Anchor-PGD* instead applies targeted projected gradient descent within the same balls, optimizing candidates against the classifier using 200 steps for each $\beta \in \{0.1, 1, 10, 100, 1000\}$ and three initializations. CertCF replaces the initial balls with adaptively refined LiRPA-certified polytopes before projection and requires no model evaluation during online generation.

Table 5 reports results computed separately for each dataset and then macro-averaged. Success is the fraction of queries for which a method returns a counterfactual assigned to the target class, while runtime is the mean online generation time per query. The remaining metrics are evaluated on the 2,266 shared-success queries for which all three methods return a target-valid counterfactual. Mean L_1 distance measures the distance between each query and its counterfactual.

Empirical robustness is evaluated using ten perturbations for every combination of numerical-noise standard deviation $\sigma \in \{0, 0.01, 0.03, 0.05, 0.10\}$ and categorical-flip probability in the same set, for a total of 250 perturbations per counterfactual. Target retention is the fraction of these perturbations that preserve the target prediction, whereas all-250 retention is the fraction of counterfactuals for which every perturbation preserves it. The post-hoc certified L_1 radius is the largest radius identified by a 14-step binary search over $[0, 5]$ for which LiRPA certifies the target prediction throughout the corresponding neighborhood. Plausibility is measured using $\log_{10}(\max\{-s_{\mathrm{LOF}}, 10^{-12}\})$ and the Isolation Forest score; smaller values of the former and larger values of the latter indicate greater support from the reference-data distribution (Breunig et al., 2000; Liu et al., 2008).

Metric	Anchor-Ball	Anchor-PGD	CertCF
Success rate (%) ↑	82.3	**100.0**	**100.0**
Mean query–counterfactual L_1 ↓	**6.954**	7.078	7.372
Mean query time (s) ↓	0.0312	13.1788	**0.0308**
Post-hoc certified L_1 radius ↑	0.314	0.323	**0.387**
Mean $\log_{10}(\mathrm{LOF})$ ↓	0.0570	0.0575	**0.0537**
Isolation Forest score ↑	-0.4250	-0.4256	$\mathbf{-0.4238}$
Mean target retention (%) ↑	91.5	91.9	**93.3**
All 250 retained (%) ↑	20.0	20.7	**25.2**

Table 5: Ablation of the certification mechanism, macro-averaged across the seven tabular datasets. Except for success and runtime, metrics are evaluated on shared-success queries. The certified radius is computed post hoc using LiRPA. Arrows indicate the preferred direction.

The initial balls alone are insufficient to guarantee validity, reaching the target class for only 82.3% of queries. Anchor-PGD restores empirical validity but provides no formal region guarantee and requires approximately 13.18 seconds per query. CertCF also achieves full validity while remaining over 400 times faster than Anchor-PGD. On shared-success queries, it incurs an approximately 4.2% L_1 proximity cost relative to Anchor-PGD, but yields the largest certified radius, the best plausibility scores, and the strongest empirical retention. Because erosion is disabled in this ablation, CertCF guarantees target validity but does not prescribe a nonzero robustness radius during generation; the certified radii reported in the table are computed post hoc.

Projection-objective norm. We isolate the norm used by the query-time projection. Both variants use the same L_1-certified atlas, the same 4,773 benchmark queries across the seven tabular datasets, and no sparsity penalty; only the projection objective changes between L_1 and L_2. Metrics are computed separately within each dataset and then macro-averaged. Distances are measured between each query and its counterfactual, while sparsity counts changed encoded coordinates.

Table 6: Effect of the query-time projection norm, macro-averaged across the seven tabular datasets. Both variants use the same L_1-certified atlas and achieve full target validity.

Objective	Validity (%) ↑	Mean L_1 ↓	Mean L_2 ↓	Changed coordinates ↓
L_1	**100.0**	**7.1483**	2.4080	**11.2065**
L_2	**100.0**	7.3901	**2.2238**	11.7975

As expected, each objective achieves the best proximity under its corresponding norm. The L_2 objective distributes changes across more encoded coordinates, whereas the L_1 objective produces more concentrated changes. We therefore use L_1 in the main tabular evaluation. This experiment changes only the projection objective; the certification and robustness norm remains L_1 in both variants. Although CertCF also supports an L_∞ projection objective, we do not use it in the tabular evaluation because minimizing the largest coordinate change may distribute modifications across many features, making it less aligned with sparse counterfactuals.

Top-k heuristic. To isolate the top-k retrieval heuristic, we disable fallback and compare search over the first $k \in \{1, \ldots, 7\}$ retrieved regions with exhaustive search over the same certified atlas. We use up to 200 queries from each of the seven tabular datasets and disable the sparsity penalty. Let $D^\star(x)$ denote the minimum feasible L_1 distance obtained by exhaustive search and $D_k(x)$ the distance obtained with top-k retrieval. For queries on which both searches succeed, the absolute optimality gap is

$$\Delta_k(x) = \max\{0, D_k(x) - D^\star(x)\}. \tag{14}$$

We regard the exhaustive optimum as recovered when

$$D_k(x) \le D^\star(x) + 10^{-5} + 10^{-6}\max\{1, D^\star(x)\}. \tag{15}$$

A failed top-k search counts as a miss. The mean gap averages Δ_k over all queries for which both searches succeed, whereas the conditional gap averages it only over shared-success queries that miss

the exhaustive optimum. The latter therefore measures the severity of the approximation when a miss actually occurs, independently of how often misses occur. Each metric is first computed within each dataset and then macro-averaged across datasets. For the conditional gap, datasets with no qualifying miss do not contribute to the average.

Table 7: Top-k retrieval compared with exhaustive certified-atlas search, macro-averaged across the seven tabular datasets. Both gaps are measured in L_1 distance. The conditional gap includes only shared-success misses; runtime is the macro-average of per-dataset median query times.

k	Success (%) ↑	Exact recovery (%) ↑	Mean gap ↓	Conditional gap ↓	Median time (ms) ↓
1	98.86	88.72	0.01920	0.14659	9.06
2	99.36	96.33	0.00763	0.15366	13.45
3	99.71	97.91	0.00294	0.15889	17.45
4	99.79	98.05	0.00264	0.12983	21.38
5	99.93	99.57	0.00067	0.14145	25.30
6	100.00	99.71	0.00055	0.13110	28.27
7	100.00	99.79	0.00054	0.18818	32.17

Both the empirical miss probability and the mean optimality gap decrease approximately exponentially over the evaluated range. Least-squares fits in log space give

$$\widehat{p}_{\mathrm{miss}}(k) \approx 0.176 \exp(-0.664k), \qquad R^2_{\log} = 0.956, \tag{16}$$

$$\overline{\Delta}_{L_1}(k) \approx 0.0263 \exp(-0.624k), \qquad R^2_{\log} = 0.933. \tag{17}$$

These descriptive fits correspond to reductions of approximately 48.5% in miss probability and 46.4% in mean gap for each additional retrieved region. They summarize the observed range and should not be interpreted as universal scaling laws.

Retrieving five regions captures most of the available benefit: it succeeds with a macro-averaged success rate of 99.93% and a macro-averaged exact recovery rate of 99.57%, while keeping the mean L_1 gap below 7×10^{-4}. Its macro-averaged median query time is 25.3 ms, compared with 1075.8 ms for exhaustive search. Increasing k beyond five yields only small improvements at additional computational cost, supporting the default $k = 5$ used in the main experiments. The conditional gap is not monotonic in k: as retrieval improves, progressively fewer misses remain, and their mean is determined by a small set of harder queries. This does not contradict the simultaneous decrease in miss frequency and overall mean gap.

E PER-DATASET METRICS

Let Q denote the benchmark query set and $S_m \subseteq Q$ the queries for which method m returns a target-valid counterfactual. Metrics are computed separately within each dataset; Table 9 does not average across datasets. Unless stated otherwise, distances and changed-coordinate counts are computed in the transformed representation, in which numerical features are standardized and categorical features are one-hot encoded.

Validity, proximity, and sparsity. Validity is $|S_m|/|Q|$ and is therefore computed over all queries. For the successful queries in S_m, proximity is the mean transformed-space distance $\|x_{cf} - x_q\|_1$. Sparsity is the mean number of transformed coordinates satisfying $|(x_{cf})_j - (x_q)_j| > 10^{-6}$. Larger validity and smaller proximity and sparsity are preferred.

Empirical robustness. For every successful counterfactual, target-class retention is evaluated on the Cartesian grid $G \times G$, where $G = \{0, 0.01, 0.03, 0.05, 0.10\}$ specifies, respectively, the numerical-noise standard deviation and categorical-block flip probability. Ten perturbations are sampled per grid cell. The reported score is the mean target-retention percentage over the 25 grid cells and S_m, with larger values preferred. This is a finite perturbation test and is distinct from CertCF's optional formal erosion guarantee.

Plausibility and reference-data alignment. A novelty-mode Local Outlier Factor model with 20 neighbors is fitted on the transformed training set. For each successful counterfactual, plausibility is scored as $\log_{10}(\max\{-s_{\mathrm{LOF}}(x_{cf}), 10^{-12}\})$, where s_{LOF} is `score_samples`; the reported value is the mean over S_m, and smaller values are preferred (Breunig et al., 2000). For methods that generate new points, reference-data alignment is measured by the relative proximity ratio (RPR) ρ: the mean L_1 distance of counterfactuals to their nearest target-class training point divided by the mean leave-one-out L_1 nearest-neighbor distance among same-class training points:

$$\rho = \frac{\mathbb{E}[d(x_{cf}, x_{cf}^{\mathrm{1NN}})]}{\mathbb{E}[d(x, x^{\mathrm{1NN}})]}. \tag{18}$$

Here, $\rho \approx 1$ is ideal. The numerator and denominator are computed within each dataset over S_m and the training set, respectively; values closest to one are preferred. RPR is not reported for NN and FACE because they return reference points directly. For Figure 3(b), we additionally pool these distances across datasets after applying $\log(1 + d_1)$ and use a common 99th-percentile cutoff. Distributional alignment is measured on the same transformed and clipped values using the Earth Mover distance; smaller values indicate a closer match to the reference distribution.

Interpretability proxy. A depth-3 multi-output decision-tree regressor is trained to map x_q to x_{cf} using a 70/30 split, minimum leaf size 5, and random state 0. The mean test L_1 prediction error is normalized by the mean successful nearest-neighbor-baseline L_1 distance within the same dataset, with smaller values preferred. This measures the consistency of the query-to-counterfactual mapping with a simple surrogate; it does not directly measure human understanding.

Query time. Query time is the mean stored wall-clock time over all query rows. DiCE uses a separate 20-query, batch-size-one timing run. Smaller values are preferred.

CertCF successfully returned a counterfactual for all $4{,}773$ evaluated queries: $1{,}000/1{,}000$ on Adult, $617/617$ on COMPAS, $100/100$ on German Credit, $1{,}000/1{,}000$ on HELOC, $1{,}000/1{,}000$ on Give Me Some Credit, $1{,}000/1{,}000$ on Lending Club, and $56/56$ on Wisconsin Breast Cancer, corresponding to a 0% failure rate.

To quantify uncertainty across datasets, we resample the seven dataset-level results with replacement and recompute their unweighted mean. Table 8 reports percentile 95% confidence intervals from 20,000 bootstrap resamples using seed 123. These intervals describe cross-dataset variability for the fixed trained models and query sets; they do not measure variability from retraining the classifiers.

Method	Validity (%)	L_1 proximity	Sparsity	Emp. robustness (%)
CertCF	[100.00, 100.00]	[3.986, 11.595]	[4.50, 12.18]	[91.82, 97.09]
NN	[100.00, 100.00]	[3.114, 11.408]	[5.26, 18.01]	[87.45, 93.63]
FACE	[96.56, 99.20]	[4.200, 15.101]	[6.77, 19.17]	[87.75, 94.77]
GS	[99.11, 100.00]	[1.147, 4.474]	[2.19, 5.64]	[74.34, 81.64]
DiCE	[88.97, 97.20]	[2.586, 6.303]	[4.69, 7.90]	[89.17, 97.44]

Method	Interp. proxy	Query time (s)	Plausibility	RPR
CertCF	[1.01, 1.99]	[0.0845, 0.7854]	[0.015, 0.137]	[0.284, 0.690]
NN	[1.20, 2.55]	[0.000120, 0.000674]	[0.031, 0.111]	–
FACE	[1.06, 2.36]	[0.7303, 4.4395]	[0.013, 0.245]	–
GS	[1.60, 2.73]	[0.0087, 0.0497]	[0.104, 0.408]	[1.488, 2.481]
DiCE	[1.62, 3.33]	[1.2064, 2.1048]	[0.129, 0.458]	[1.905, 4.072]

Table 8: Percentile-bootstrap 95% confidence intervals for the dataset-averaged benchmark metrics. Each interval is computed from 20,000 resamples of the seven dataset-level results. RPR values closest to one are preferred and are reported only for methods that generate new points.

Dataset	Method	Validity ↑	L_1 ↓	L_0 ↓	Robust. ↑	DT err. ↓	Query (s) ↓	$\log_{10}(\text{LOF})$ ↓	RPR → 1
Adult	CertCF	**1.000**	7.181	6.90	92.4	**1.165**	0.330	**0.014**	**0.219**
	NN	**1.000**	4.974	5.73	88.3	1.720	**0.001**	0.086	–
	Growing Spheres	0.991	**1.420**	**1.67**	83.2	2.070	0.078	0.155	2.510
	DiCE	0.902	3.675	5.18	91.5	2.466	2.211	0.203	3.254
	FACE	0.990	8.511	9.24	**92.6**	1.671	7.617	0.039	–
COMPAS	CertCF	**1.000**	1.681	1.82	**91.6**	**1.546**	0.087	0.263	0.442
	NN	**1.000**	1.171	2.06	85.5	2.294	**0.000**	**0.172**	–
	Growing Spheres	**1.000**	**0.951**	**1.81**	78.1	2.428	0.005	0.384	**1.473**
	DiCE	0.843	2.421	3.55	90.6	4.107	1.191	0.814	4.943
	FACE	0.995	1.398	2.58	84.0	2.211	0.708	0.514	–
German Credit	CertCF	**1.000**	9.186	8.72	92.4	1.489	1.621	0.011	0.305
	NN	**1.000**	9.809	10.31	90.7	1.591	**0.000**	0.010	–
	Growing Spheres	**1.000**	**2.834**	**3.03**	78.7	1.809	0.063	0.032	**1.228**
	DiCE	0.880	6.951	8.65	85.8	1.899	2.700	0.034	1.317
	FACE	0.970	14.396	14.74	**92.8**	**1.370**	0.006	**0.007**	–
Give Me Some Credit	CertCF	**1.000**	1.342	2.99	89.6	**2.951**	0.042	**0.098**	**1.094**
	NN	**1.000**	0.723	4.95	86.2	3.652	**0.000**	0.113	–
	Growing Spheres	**1.000**	**0.228**	**2.84**	66.9	3.715	0.001	0.115	3.336
	DiCE	0.996	0.869	5.96	**98.5**	4.050	1.114	0.252	5.243
	FACE	0.992	0.822	5.28	86.0	3.428	3.405	0.113	–
HELOC	CertCF	**1.000**	11.267	11.90	98.8	**0.721**	0.079	**0.008**	0.353
	NN	**1.000**	9.632	15.16	94.3	0.899	**0.000**	0.027	–
	Growing Spheres	**1.000**	**3.391**	5.61	79.8	1.290	0.011	0.690	**1.527**
	DiCE	0.919	4.604	**4.13**	**99.9**	1.279	0.834	0.186	1.660
	FACE	0.940	8.841	16.75	92.7	0.814	1.611	0.012	–
Lending Club	CertCF	**1.000**	5.496	5.64	**95.6**	**1.431**	0.160	0.007	**0.554**
	NN	**1.000**	3.877	6.72	89.0	2.060	**0.000**	0.024	–
	Growing Spheres	**1.000**	**1.543**	**2.33**	80.6	2.263	0.011	0.104	1.684
	DiCE	0.982	3.238	5.75	87.0	2.320	1.628	0.184	2.621
	FACE	0.995	7.136	8.37	90.8	1.556	3.469	**0.003**	–
Wisconsin Breast Cancer	CertCF	**1.000**	17.379	18.25	**100.0**	0.796	0.102	0.056	0.225
	NN	**1.000**	18.239	30.00	98.4	0.611	**0.000**	0.037	–
	Growing Spheres	0.982	**7.589**	**8.73**	82.9	1.240	0.026	0.168	1.747
	DiCE	**1.000**	8.823	10.16	99.8	0.933	1.732	0.141	**1.583**
	FACE	0.982	24.231	30.00	99.9	**0.602**	0.006	**0.018**	–

Table 9: Evaluation of counterfactual explanation methods across seven benchmark datasets. Best values within each dataset are shown in bold. Arrows denote the direction of improvement (↑ higher is better, ↓ lower is better); RPR values closest to 1 are preferred and are reported only for methods that generate new points.

F Computational Scalability

We evaluate computational scalability by separating offline precomputation from online counterfactual search. Most comparison methods generate recourse entirely on demand, whereas CertCF and FACE incur an offline cost to construct reusable structures that amortize over subsequent queries.

Both phases of CertCF are naturally data parallel. During offline construction, anchors or anchor batches can be partitioned across CPU workers, GPUs, or compute nodes, since their LiRPA certificates are computed independently before the resulting regions are collected into the atlas. At query time, independent queries can be distributed for throughput, while the candidate-region projections of a single query can be evaluated concurrently to reduce latency. With at least k available workers, the projection component of a top-k query is determined by the slowest candidate projection, plus retrieval, dispatch, and result-reduction overhead. CertCF therefore admits horizontal scaling in both phases, with practical performance bounded primarily by available compute and memory.

We organize the empirical analysis along four axes: the original tabular benchmark (Appendix F.1), controlled FCNN architecture scaling (Appendix F.2), image-domain scalability stress tests (Appendix F.3), and batched and parallel implementation speedups (Appendix F.4).

F.1 ORIGINAL TABULAR BENCHMARK

Offline build phase. CertCF computes LiRPA certificates around target-class anchors and stores the resulting certified polytopes, whereas FACE constructs a graph over the training data and assigns density-weighted edge costs.

Method	Mean (s)	Median (s)	Min (s)	Max (s)
CertCF	0.3816	0.4372	0.1942	0.4765
FACE	3080.35	1350.40	16.48	14249.92

Table 10: Offline build time for methods with a reusable offline structure. Statistics summarize one construction time per dataset. CertCF uses the optimized offline implementation described in Appendix F.4.

Across the seven datasets, CertCF requires 0.3816 s on average and at most 0.4765 s to construct its atlas. FACE averages 3,080.35 s and reaches 14,249.92 s, making its mean measured build time approximately $8{,}073\times$ larger. Both are one-time costs that can be amortized over subsequent queries.

Online query phase. CertCF projects each query onto a small set of retrieved certified polytopes. FACE searches its precomputed graph, Nearest Neighbor ranks target-class training points, Growing Spheres samples candidates, and DiCE solves a gradient-based optimization problem.

Method	Mean (s)	Median (s)	Min (s)	Max (s)
CertCF	0.3459	0.0967	0.0180	6.7460
NN	0.0003	0.0003	0.0000	0.0072
Growing Spheres	0.0280	0.0181	0.0008	0.2182
DiCE	1.6300	1.6618	0.2137	3.6040
FACE	2.4032	2.5368	0.0019	10.6751

Table 11: Online query time across datasets. The mean and median columns average the corresponding dataset-level statistics, so each dataset contributes equally; the minimum and maximum are taken over all query rows. CertCF uses up to eight parallel projection workers. DiCE is evaluated with batch size one to reflect the one-query-at-a-time setting.

Nearest Neighbor and Growing Spheres have the lowest mean query times, but do not provide a certified target-class region. CertCF averages 0.3459 s per query, compared with 1.6300 s for DiCE and 2.4032 s for FACE, corresponding to approximately $4.7\times$ and $6.9\times$ lower mean latency, respectively. Its 0.0967 s median indicates that the mean is influenced by a smaller number of more expensive queries.

F.2 CONTROLLED FCNN ARCHITECTURE SCALING

Synthetic benchmark. To isolate network complexity from input dimension and dataset effects, we use a balanced binary dataset generated with `make_classification`. It contains 32 numerical features, all informative, with two clusters per class, class separation 1.5, and 1% symmetric label noise. Using seed 42, we generate 10,000 training, 1,000 validation, and 1,000 test samples. Features are standardized using training-set statistics.

We train 25 ReLU FCNNs by combining depths $D \in \{1, 2, 3, 4, 5\}$ with uniform hidden widths $W \in \{16, 32, 64, 128, 256\}$. We construct an atlas of 1,000 certified regions for every model and use $k = 3$ at query time. Across the grid, test accuracy remains between 97.8% and 98.3%, and all 25,000 queries return a target-valid counterfactual.

We evaluate both the original serial implementation and the optimized implementation described in Appendix F.4. Serial build time is the original measurement for each architecture. Optimized build time is the mean over three repetitions using eight CPU workers for the initial radii and LiRPA batches of at most 512 anchors. Serial and parallel query times are evaluated on the same 1,000 queries per

architecture. Both include the reweighted-L_1 sparsity refinement used in the main benchmark, with $\lambda = 1$ and two reweighting iterations. Figure 8 reports all 25 architectures and shows the distinct offline and online scaling behavior before and after these optimizations.

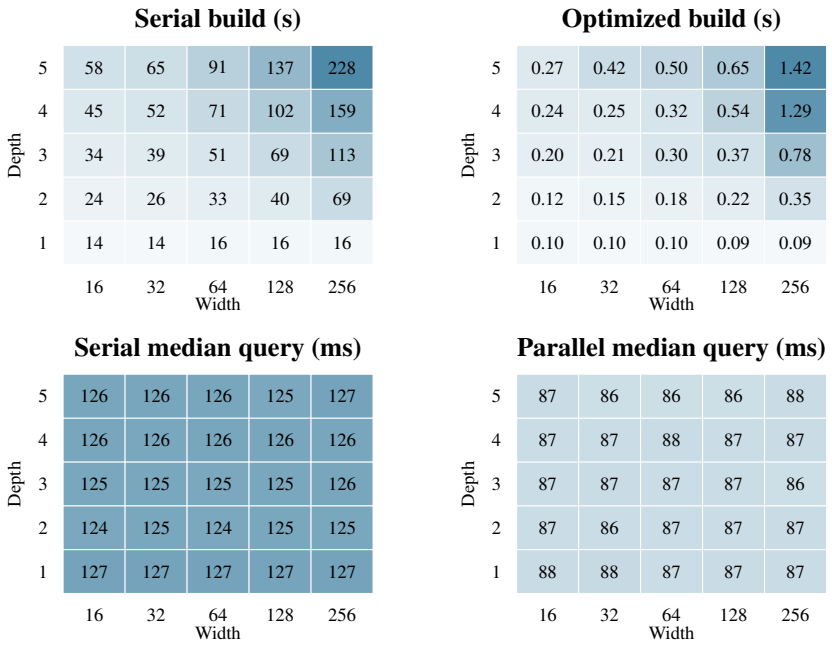


Figure 8: Complete network-complexity grid before and after implementation optimization. Top: serial and optimized atlas construction. Bottom: serial and parallel median query latency. Optimized build cells are means over three repetitions; all displayed values are rounded. Build colors are normalized separately within each panel. Query colors use a common standardized scale centered at the joint mean, with the palette spanning two joint standard deviations on either side. Network size strongly affects offline construction, whereas query latency remains stable across the grid.

Table 12 compares the two endpoints of the grid. Increasing the network from 16 to 1,280 hidden neurons and from 562 to 272,130 parameters raises optimized atlas-construction time from 0.102 to 1.416 s. In contrast, median online query time is nearly unchanged within each implementation. At these endpoints, batching accelerates atlas construction by $132.7\times$ and $161.0\times$, while parallel projection reduces median query latency by approximately $1.45\times$.

		Atlas build (s)			Median query (ms)		
(D, W)	Parameters	Serial	Optimized	Speedup	Serial	Parallel	Speedup
$(1, 16)$	562	13.590	0.102 ± 0.007	$132.7\times$	127.269	87.525	$1.454\times$
$(5, 256)$	272,130	227.965	1.416 ± 0.020	$161.0\times$	127.460	87.576	$1.455\times$

Table 12: Serial and optimized runtime at the endpoints of the controlled FCNN grid. Each build constructs 1,000 certified regions; optimized build time is mean $\pm$ sample standard deviation over three repetitions. Query columns report medians over the same 1,000 queries per architecture.

Over this finite grid, the serial and optimized build times are described by

$$\widehat{T}_{\text{serial}} = 5.70 + 7.82D + 0.1713(D-1)W \quad \text{s}, \tag{19}$$

$$\widehat{T}_{\text{optimized}} = 0.0619 + 0.0210D + 0.001238(D-1)W \quad \text{s}. \tag{20}$$

Their leave-one-architecture-out predictive R^2 values are 0.992 and 0.951, respectively. These are empirical fits for the evaluated architectures, certification protocol, hardware, and seed, not asymptotic LiRPA complexity bounds.

Figure 9 compares held-out predictions with the observed measurements and shows how both fitted build times change with width at each depth.

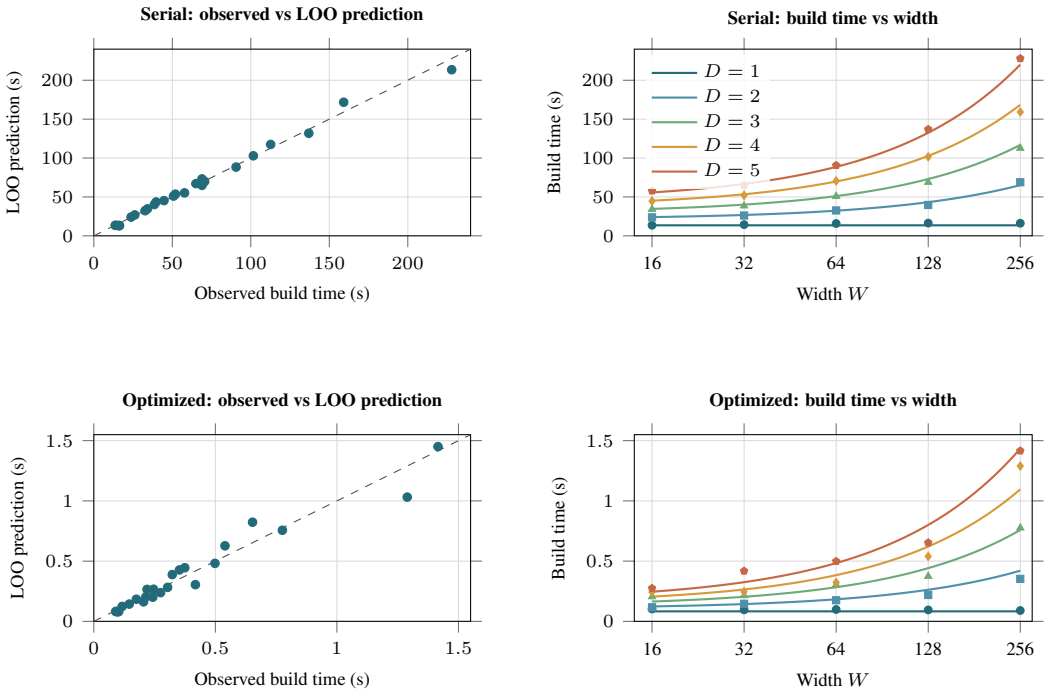


Figure 9: Validation and interpretation of the serial and optimized empirical atlas-build-time models. Left: held-out predictions against observed times; dashed lines denote perfect prediction. Right: observed times (markers) and fitted predictions (lines) as width increases, separated by network depth. Optimized observations are means over three repetitions.

F.3 Image-domain scalability stress tests

The experiment below stress-tests the computational scalability of CertCF on multiclass convolutional networks. On CIFAR-10, we evaluate both offline atlas construction and targeted online generation using networks with more than 10^5 ReLU activations. This establishes technical applicability at this scale, but does not measure LiRPA relaxation tightness or perceptual counterfactual quality. Natural images occupy a low-dimensional manifold embedded in a high-dimensional input space, whereas the current method constructs and projects onto convex regions directly in pixel space. Consequently, validity and certification do not by themselves imply perceptual plausibility, semantic actionability, or on-manifold image recourse.

CIFAR-10 ResNets. We use the CIFAR-10 dataset and pretrained ResNet-20, ResNet-32, and ResNet-56 checkpoints from `pytorch-cifar-models`.[2] For every network, we select 10,000 training images balanced by true class and use all of them as anchors, producing 10,000 certified regions. The 100 test queries contain ten images per true class and are correctly classified by all three networks. For each query, the target is the true class of its nearest L_1 support image belonging to a different class. We use L_1 certification and projection, $\alpha = 0.2$, and $k = 3$, without sparsity refinement or robustness erosion.

[2]Architecture and checkpoints from `https://github.com/chenyaofo/pytorch-cifar-models`, commit `786c162`, distributed under the BSD-3-Clause license.

Network	Parameters	ReLU activations	Accuracy	Build (s)	Median query (s)
ResNet-20	272,474	188,416	92.60%	41.79 ± 0.50	0.4340
ResNet-32	466,906	303,104	93.53%	49.59 ± 0.16	0.4368
ResNet-56	855,770	532,480	94.38%	65.61 ± 0.29	0.4356

Table 13: CertCF scaling on pretrained CIFAR-10 ResNets. Each optimized build constructs 10,000 certified regions; times are mean $\pm$ sample standard deviation over three repetitions using eight workers for the initial radii and LiRPA batches of at most 32 anchors. Median parallel query time is computed on the same 100 test queries for every network.

Across the evaluated architectures, the optimized build time grows less than linearly with the number of ReLU activations: a $2.83\times$ increase in activations corresponds to a $1.57\times$ increase in build time. Query time again remains essentially constant across network sizes because the classifier is not evaluated after the atlas has been constructed. Its approximately 0.44-second median is higher than in the tabular grid because each projection operates in the 3,072-dimensional pixel space. During atlas construction, peak allocated CUDA memory increases from 2.26 to 6.17 GiB, while peak process RSS remains between 4.60 and 4.69 GiB. Across the three networks, CertCF returns a target-valid counterfactual for all 300 queries.

F.4 Batched and parallel implementation

We optimize both phases of CertCF by exploiting the independent subproblems they contain. Offline, initial-radius computation and LiRPA certification decompose across anchors; online, projections decompose across retrieved regions. We therefore use CPU parallelism and GPU batching during atlas construction, and process-level parallelism for query-time projection.

Offline atlas construction. To compute the initial radii from Equation (4), we split the anchors into class-specific chunks. Each CPU worker computes the distance from its assigned anchors to all support points predicted outside their class and returns the minimum for each anchor. The resulting radii are then restored to the original anchor order.

For FCNNs, batched LiRPA retains a separate radius for every anchor. We divide the anchors into batches of at most the configured size. After each LiRPA call, certified anchors are removed from the batch. The radii of the remaining anchors are reduced, and only those anchors are evaluated again. Batching therefore changes the evaluation schedule but not the region, certification, or adaptive-shrink rule. In auto-LiRPA v0.7.2 (Verified Intelligence, 2026), however, batched certification of CNNs requires a single scalar perturbation radius, so a batch cannot directly use distinct per-anchor radii. This is a restriction of the library version used in our experiments that could be addressed in future versions. To batch CNN anchors on the GPU, we first sort same-class anchors by radius. We then group anchors with similar radii, requiring the largest radius in each batch to be at most 5% greater than the smallest. LiRPA is evaluated at the largest radius in each group. The resulting bounds remain sound for every original ball because each is contained in the larger batch ball. An anchor that is not certified by this conservative bound is recomputed at its exact radius before the usual adaptive-shrink procedure.

We first evaluate the parallel computation of the initial radii in isolation. On the 10,000 CIFAR-10 anchors, its mean runtime over three repetitions decreases from 82.152 s with one CPU worker to 26.923 s with eight workers. Table 14 reports the batch-size sweep used to choose the implementation settings. The FCNN case is the largest controlled-grid model, with five hidden layers of width 256 and 1,000 atlas anchors. The CNN case is ResNet-20 with 10,000 anchors; its initial-radius computation is fixed at eight CPU workers so that this part of the sweep isolates LiRPA batching. For the FCNN, build time continues to decrease up to batch size 512, which contains all 500 anchors of one target class in the initial wave. For ResNet-20, the mean $\pm$ standard-deviation ranges for batch sizes 32, 64, and 128 overlap. Although 64 has the lowest observed mean, batch size 32 is only 3.3% slower and uses 38.7% less peak GPU memory. We consequently select 512 for the controlled FCNN grid and 32 for the CNN stress test.

Workload	Batch size	Build (s)	LiRPA (s)	GPU (MiB)	Speedup
FCNN ($D = 5, W = 256$)	1	295.80 ± 16.01	295.80 ± 16.01	13.1	$1.00\times$
	8	44.14 ± 0.88	44.14 ± 0.88	22.4	$6.70\times$
	16	24.28 ± 0.44	24.27 ± 0.44	34.6	$12.18\times$
	32	12.52 ± 0.15	12.51 ± 0.15	59.0	$23.63\times$
	64	6.83 ± 0.14	6.83 ± 0.14	107.7	$43.29\times$
	128	3.81 ± 0.07	3.80 ± 0.07	205.2	$77.69\times$
	256	2.25 ± 0.01	2.25 ± 0.01	400.2	$131.20\times$
	512	$\mathbf{1.52 \pm 0.15}$	$\mathbf{1.51 \pm 0.15}$	778.0	$194.66\times$
ResNet-20	1	276.88 ± 2.19	250.84 ± 2.63	443.1	$1.00\times$
	8	61.53 ± 0.47	35.44 ± 0.36	1,337.9	$4.50\times$
	16	47.13 ± 0.38	21.02 ± 0.23	1,634.2	$5.88\times$
	32	$\mathbf{41.79 \pm 0.50}$	$\mathbf{15.23 \pm 0.12}$	**2,311.2**	$6.62\times$
	64	40.47 ± 1.20	13.90 ± 0.22	3,768.3	$6.84\times$
	128	42.09 ± 0.46	15.41 ± 0.45	6,539.3	$6.58\times$

Table 14: LiRPA batch-size sweep. Build and LiRPA times are reported as mean $\pm$ sample standard deviation over three repetitions. Speedup is relative to the mean at batch size one within the same workload. GPU denotes peak allocated CUDA memory. Bold entries denote the selected settings.

All batch sizes preserve the set of certified anchors, final radii, and shrink decisions in the FCNN sweep. For ResNet-20, the selected batch size 32 produces 403 radius batches and certifies all 10,000 centers without exact-radius fallback or adaptive shrink. Relative to the fully serial implementation, it reduces LiRPA time from 262.853 s to 15.231 ± 0.122 s and complete atlas-construction time from 339.028 s to 41.795 ± 0.500 s, corresponding to speedups of $17.26\times$ and $8.11\times$, respectively. Peak allocated GPU memory increases from 443.1 to 2,311.2 MiB.

Online counterfactual queries. CertCF solves the projection problems associated with the retrieved top-k regions concurrently using at most eight worker processes, matching the eight performance cores of the evaluation CPU, and selects the final candidate by distance, resolving ties in anchor order. With W workers and per-projection cost P, the idealized projection latency is

$$O\left(\left\lceil \frac{k}{W} \right\rceil P\right), \tag{21}$$

plus process and solver overhead; the total computational work remains approximately $O(kP)$. This optimization reduces single-query latency, which we consider a more relevant objective in realistic deployment settings than maximizing throughput by batching many queries. However, the parallel implementation can also be used to process multiple queries concurrently, with each query assigned to a separate worker.

Table 15 reports representative paired measurements. The synthetic experiment uses a separate benchmark with a depth-3, width-128 FCNN, a certified atlas, and 1,000 test queries, extending the evaluation to $k \in \{1, \ldots, 8\}$ and 1, 2, 4, and 8 workers; the table shows the most parallel case. Across its 24,000 parallel configurations, the returned distance and selected anchor exactly match the serial result. The tabular and CIFAR reruns preserve success and target validity for all 4,773 and 300 queries, respectively. On the tabular benchmark, 4,772 of 4,773 counterfactuals agree within 10^{-6}; one solver near-tie selects a different valid region and changes the global mean L_1 distance by less than 2×10^{-5}.

Workload	Statistic	Serial	Parallel	Speedup
Synthetic, $k = 8$	Median	24.597 ms	6.124 ms	$4.02\times$
Seven tabular datasets, $k = 5$	Total	1,207.54 s	832.43 s	$1.45\times$
CIFAR-10 ResNets, $k = 3$	Total	322.46 s	134.73 s	$2.39\times$

Table 15: Representative serial and parallel online-query measurements. The parallel implementation uses up to eight worker processes and one numerical thread per process.

The smaller aggregate speedup on the tabular datasets is partly due to work that remains serial. When reweighted sparsity refinement is enabled, or when categorical features must be decoded,

the region selected by the parallel workers is solved once more in the main process to reduce numerical differences between processes. This additional serial work, together with the relatively short projections for the $k = 5$ regions, limits the aggregate speedup. We quantify the cost of sparsity refinement below.

Cost of optional sparsity refinement. To isolate the cost of sparsity refinement, we repeat the complete controlled FCNN grid using the same 25 architectures and 1,000 paired queries per architecture. Both configurations use $k = 3$ and up to eight parallel projection workers. The complete configuration applies the reweighted-L_1 refinement with $\lambda = 1$ and two reweighting iterations; the ablated configuration disables this optional step.

Configuration	Success	Median (ms)	P95 (ms)	Mean L_0	Mean L_1	Mean L_2
Without refinement	100%	**4.020**	**6.420**	32.00	**23.48**	**5.58**
With refinement	100%	87.043	172.051	**22.12**	23.76	5.93

Table 16: Ablation of the optional sparsity refinement over all 25,000 paired queries in the controlled FCNN grid. Both configurations preserve 100% success and differ only in whether reweighted-L_1 refinement is enabled.

Without refinement, the parallel convex projection takes 4.020 ms at the median and slightly improves mean L_1 and L_2 proximity by 1.17% and 5.89%, respectively. However, every returned counterfactual changes all 32 features. Enabling refinement reduces mean L_0 from 32.00 to 22.12, a 30.9% reduction, while increasing median latency by $21.65\times$. This additional cost comprises the reweighted projection solves for every retrieved region and the complete refinement repeated for the winning region in the main process. Thus, the approximately 4 ms result characterizes the parallel projection path, whereas the complete query time includes the optional computation used to obtain more sparse counterfactuals.

G Complete CertCF Algorithm

Algorithms 1 and 2 summarize the two phases of CertCF. The first constructs the reusable certified atlas for a target class. The second answers a query using the precomputed atlas, without further network evaluations.

Algorithm 1 Offline certified-atlas construction

Require: Model f, target class t, reference dataset $\mathcal{D}_{\text{ref}}$, scale factor α, shrink factor β, maximum retries R, classification margin γ
Ensure: Certified atlas $\mathcal{A}_t$
$\mathcal{A}_t = \emptyset$
Predict the label $\hat{y}_i$ of every $x_i \in \mathcal{D}_{\text{ref}}$
$I_t = \{i : \hat{y}_i = t\}$
for each $i \in I_t$ **do**
  $x_j =$ nearest point in $\mathcal{D}_{\text{ref}}$ such that $\hat{y}_j \neq t$
  $\epsilon_i = \alpha \|x_i - x_j\|_p$ {Equation (4)}
  $\mathcal{P}_i = \text{NONE}$
  for $r = 0, \ldots, R$ **do**
    $C_i = \mathcal{B}_p(x_i, \epsilon_i)$
    $B_i = \text{RunLiRPA}(f, C_i, t, \gamma)$
    $P = C_i \cap \text{TargetClassHalfspaces}(B_i, \gamma)$
    if P is nonempty and $x_i \in P$ **then**
      $\mathcal{P}_i = P$
      break
    end if
    $\epsilon_i = \beta \epsilon_i$
  end for
  if $\mathcal{P}_i \neq \text{NONE}$ **then**
    Add $(x_i, \mathcal{P}_i)$ to $\mathcal{A}_t$
  end if
end for
return $\mathcal{A}_t$

Optional sparsity refinement. For the L_1 projection, let $\mathcal{G}$ contain one singleton group per numerical feature and one group per categorical block. Starting from $z_i^{(0)} = z_i^\star$, iteration r updates

$$w_g^{(r)} = 1 + \frac{\lambda}{\|z_{i,g}^{(r-1)} - x_{q,g}\|_1 + \eta}, \tag{22}$$

$$z_i^{(r)} \in \arg\min_{x \in \mathcal{P}_i^\delta \cap \mathcal{C}_{\mathrm{cvx}}(x_q)} \sum_{g \in \mathcal{G}} w_g^{(r)} \|x_g - x_{q,g}\|_1, \tag{23}$$

where $\lambda \geq 0$ controls the sparsity penalty and $\eta > 0$ keeps the weights finite. Groups that remain close to their query values receive larger weights, encouraging subsequent solves to concentrate changes in fewer groups. Each iterate is passed through the same decode-and-certify step and compared using Equation (13).

Algorithm 2 Online counterfactual query

Require: Query x_q, target atlas $\mathcal{A}_t$, number of initial anchors k, admissible set $\mathcal{C}(x_q)$, convex constraints $\mathcal{C}_{\mathrm{cvx}}(x_q)$, robustness radius δ, selection score s
Ensure: Certified counterfactual or FAILURE

Order the regions in $\mathcal{A}_t$ by anchor distance from x_q
$K =$ first $\min(k, |\mathcal{A}_t|)$ regions; $U =$ remaining regions
while K is nonempty **do**
  $V = \emptyset$
  for all $(x_i, \mathcal{P}_i) \in K$ **in parallel do**
    $P = \mathcal{P}_i$
    if $\delta > 0$ **then**
      $P = \mathrm{Erode}(\mathcal{P}_i, \delta)$
    end if
    $z_i = \mathrm{ClosestPoint}(x_q, P, \mathcal{C}_{\mathrm{cvx}}(x_q))$
    if the projection is infeasible **then**
      continue
    end if
    if sparsity refinement is enabled **then**
      $z_i = \mathrm{RefineForSparsity}(z_i, x_q, P, \mathcal{C}_{\mathrm{cvx}}(x_q))$
    end if
    $x_i^{cf} = \mathrm{DecodeAndCertify}(z_i, P, \mathcal{C}(x_q))$
    if $x_i^{cf} \neq$ NONE **then**
      Add $(i, x_i^{cf}, s(x_i^{cf}))$ to V
    end if
  end for
  if $V \neq \emptyset$ **then**
    $x_{\mathrm{best}} =$ first minimizer of $s(x_i^{cf})$ in anchor order
    return x_{best}
  end if
  if U is empty **then**
    return FAILURE
  end if
  $K =$ next nearest region in U; remove that region from U
end while
return FAILURE

The online phase first evaluates the k nearest target-class anchors and returns the best certified candidate under the active selection score. Without sparsity refinement, this score is query-to-counterfactual distance; otherwise, it is Equation (13). If no candidate is feasible, the method progressively examines the remaining regions in nearest-anchor order. This heuristic does not guarantee the global optimum over the complete atlas when the initial top-k set succeeds. It does preserve validity, because every returned candidate must pass the certified-region check. We use $k = 5$ in the reported experiments. The DECODEANDCERTIFY step first tries direct one-hot decoding, then bounded beam search, and finally exact decoding if the earlier attempts fail; every decoded candidate is checked against the selected certified region. Parallel results are reduced in anchor order for deterministic tie-breaking. When process workers are used together with sparsity refinement or categorical decoding, the winning region is solved again in the parent process to reduce process-local numerical variation. The offline radius batching described in Appendix F.4 changes only how LiRPA calls are scheduled; the certified regions returned by Algorithm 1 retain the same soundness requirement.