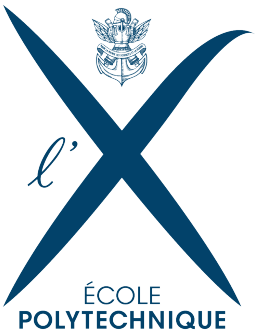
Interpretable Counterfactual Explanations Guided by Prototypes

Arnaud Van Looveren **Janis Klaise**

Prepared by:

MOHAMED JALAL BAIM

The full code implementation is available on GitHub:
github.com/Jalalbaim/Interpretable-Counterfactual-Guided-by-Prototypes



INSTITUT
POLYTECHNIQUE
DE PARIS

Abstract

Counterfactual explanations aim to answer the question: "What minimal change to the input would alter the model's prediction?"

In this report, we'll study the paper *Interpretable Counterfactual Explanations Guided by Prototypes*, which proposes a model-agnostic method that improves the interpretability and computational efficiency of counterfactual explanations by introducing class prototypes into the optimization process.

This report provides a summary, a mathematical analysis, a reproduction of the experiments, as well as additional proposals (K-Means for prototype selection, CIFAR-10 experiments, pixel selection using XAI methods), and a critical evaluation of the proposed method within the broader landscape of Explainable AI (XAI).

Table of Contents

1 Introduction

The ubiquity of machine-learning models in sensitive aspects of modern society makes it critical for the end user to be able to understand how a given decision was reached by the model.

In the context of eXplainable Artificial Intelligence (XAI), we aim to provide human-understandable explanations for machine learning models, especially when these models are deployed in high-stakes decision-making domains.

Among the different explanation paradigms, counterfactual explanations have gained significant attention due to their intuitive nature. A counterfactual explanation answers the question: *"What minimal change to the input would alter the model's prediction?"* [9].

Actually, there is some contrasts with feature attribution methods such as LIME [4] and SHAP [3], which quantify the contribution of each feature to a prediction.
While attribution methods highlight importance, they do not necessarily provide a concrete alternative scenario that would change the outcome. Here comes **Counterfactual explanations**, which are therefore often described as contrastive explanations, answering the question "Why output A instead of B?" [1].

However, generating high-quality counterfactuals is non-trivial. A naive optimization approach may produce unrealistic or adversarial-like examples that satisfy the classifier but do not correspond to plausible data points.

The main challenge is to be able to generate explanations that statisfies: the classifier prediction, the plausibility and also the interpretability.

1.1 Motivation of the Paper

The paper *Interpretable Counterfactual Explanations Guided by Prototypes* addresses limitations in existing counterfactual generation methods. The authors argue that a satisfactory counterfactual explanation should satisfy four key properties:

1. It must achieve the target prediction.
2. It should be sparse, modifying as few features as possible.
3. It should be interpretable, meaning it lies close to the training data distribution, especially within the target class manifold.
4. It must be plausible, and remains in-distribution.

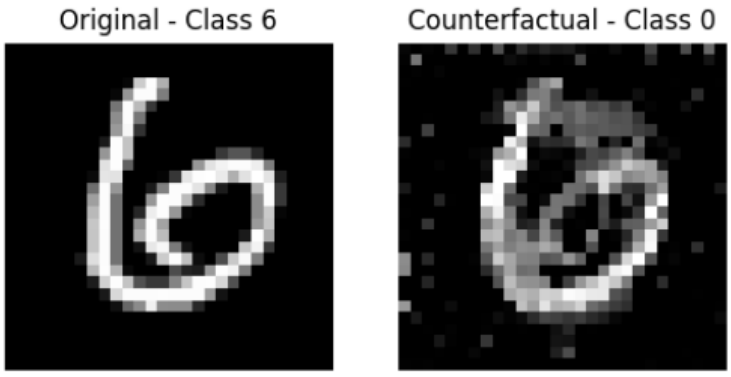


Figure 1: Illustraion of a counterfactual generation. The original image belongs to class 6, and the counterfactual belongs to class 0.

2 Related works

2.1 Standard approaches

Counterfactual instances—synthetic data points engineered from real inputs to alter a model's prediction, have emerged as a powerful alternative to feature attribution methods such as LIME [4] and SHAP [3].

Several optimization-based approaches have been proposed. Wachter et al. [9] formulate counterfactual generation as the minimization of an objective function combining prediction error and a scaled L_1 norm to promote sparsity.

Laugel et al. [2] instead propose a heuristic search procedure based on growing hyperspheres around the original instance until a class change occurs.

Moreover, in black-box settings, the number of model evaluations required during optimization scales either with input dimensionality or with the number of sampled candidates, leading to significant computational overhead. Dhurandhar et al. [1] introduce contrastive explanations that identify minimal sets of features to change or preserve; however, these approaches still rely heavily on prediction-based optimization.

2.2 Contributions

While prior approaches such as [9] focus on prediction change and sparsity, they do not explicitly enforce that counterfactuals lie near the data manifold of the target class. As a result, generated instances may resemble adversarial perturbations rather than meaningful alternatives.

Other works introduce reconstruction losses via autoencoders to encourage realism [1], but these approaches do not directly guide the search toward representative target-class examples.

To address both **interpretability and computational efficiency**, the authors propose a prototype-guided counterfactual generation strategy. Instead of pushing an instance away from its original class, the method explicitly brings it toward a representative prototype of the target class.

By introducing the prototype-based loss term, the optimization is guided toward realistic regions of the feature space. Furthermore, this approach enables the removal of the prediction loss term in certain settings, reducing computational cost.

This idea situates the work at the intersection of counterfactual explanations, prototype-based learning [7], and interpretability. As such, the paper contributes both algorithmically and interpretability within XAI.

3 Prototype-Guided Counterfactuals

A counterfactual instance x^{cf} is obtained by slightly modifying an initial sample x_0 through a minimal perturbation δ, such that $x^{cf} = x_0 + \delta$. This modification should induce a change in the model's prediction toward a target class, while ensuring that x^{cf} remains close to the real data distribution of that class (i.e., in-distribution).

To operationalize this idea, the authors formulate counterfactual generation as a global optimization problem: find the smallest perturbation δ that flips the prediction to the target class while preserving plausibility with respect to the data distribution.

3.1 Definition of Class Prototypes

A prototype is intended to represent a typical example of a given class. Rather than using a single arbitrary training instance, the method constructs a representative point that captures the local structure of the class distribution.

3.1.1 Original approach

When an encoder $ENC(\cdot)$ is available (for instance, the encoder of an autoencoder trained on the data), prototypes are defined in the latent space rather than in the original feature space.

Let $\mathcal{X}_i$ denote the set of training instances predicted as class i. The encoder maps each instance $x \in \mathbb{R}^d$ to a latent representation $ENC(x) \in \mathbb{R}^E$, where typically $E < d$.

For a given input x_0, the prototype of class i is constructed as the mean encoding of the K nearest latent representations belonging to class i:

$$proto_i = \frac{1}{K} \sum_{k=1}^{K} ENC(x_{ik}), \tag{1}$$

where the instances x_{ik} are selected based on increasing Euclidean distance to $ENC(x_0)$ in latent space.

The main point from working in latent space is that the encoder is assumed to capture meaningful structure of the data manifold. As a result, distances in latent space reflect semantic similarities rather than raw Euclidean distances in the image space.

3.1.2 K-d Trees class representations

When an autoencoder (AE) is not available, the authors propose an alternative prototype selection procedure in the original feature space.

After labeling a representative training set using the predictive model, each class i is represented by a separate k-d tree built from instances with label i. For each k-d tree with class $j \neq t_0$, we compute the Euclidean distance between x_0 and the k-nearest neighbor in that tree, denoted $x_{j,k}$.

The closest $x_{j,k}$ across all classes $j \neq t_0$ is then selected as the class prototype proto_j.

3.1.3 KMeans

Another idea, which is not mentionned in the paper, is to run K-Means locally (per class) in the latent space to define prototypes. Concretely, for each class we apply the K-Means algorithm in the latent space and use the resulting centroids as prototypes. This yields multiple prototypes per class, which can capture different modes of variation (i.e., different behaviors) within the same class.

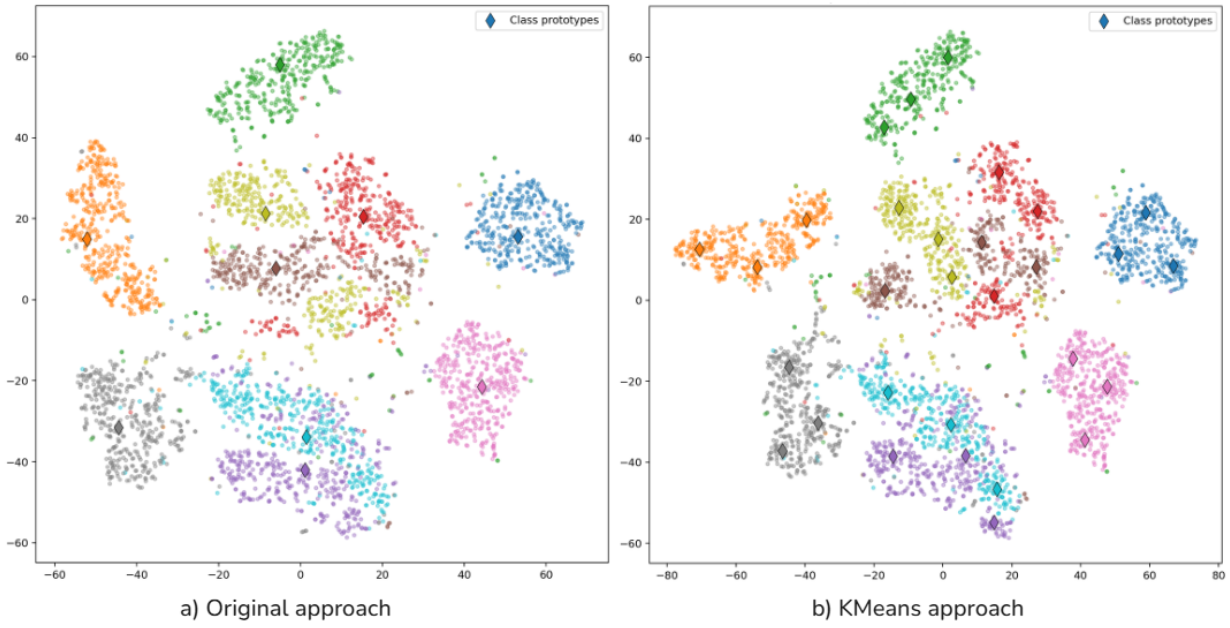


Figure 2: Visualization of the autoencoder's latent space on the MNIST dataset (4,000 samples) using t-SNE. (a) shows the original approach described in the paper, where some class prototypes are placed far from the corresponding class distribution. (b) shows our approach with K-Means ($K = 3$), which yields better results by defining prototypes within the class distribution.

3.2 Prototype Selection

Once prototypes are defined for each class, the next step is to select the most appropriate target prototype. Let t_0 be the predicted class of the original instance x_0. The method selects the closest prototype belonging to a class different from the original one.

$$j = \arg\min_{i \neq t_0} \|ENC(x_0) - proto_i\|_2. \tag{2}$$

This selection ensures that the optimization moves toward the *nearest plausible alternative* rather than an arbitrary target class. Which corresponds to identifying the most accessible counterfactual class in the latent space.

3.3 The optimization objective

The generation of counterfactual explanations is formulated as a continuous optimization problem over the perturbation δ.

The classical formulation of counterfactual generation can be written as:

$$\min_{\delta} \; c \cdot L_{\text{pred}}(x_0, \delta) + \beta\|\delta\|_1 + \|\delta\|_2^2 + L_{AE}(x_0 + \delta), \tag{3}$$

where each term plays a specific role.

Prediction Loss L_{pred}. The prediction loss enforces that the perturbed samples changes class.

$$L_{\text{pred}} = \max\left(f_{t_0}(x_0 + \delta) - \max_{i \neq t_0} f_i(x_0 + \delta), -\kappa\right), \tag{4}$$

where:

- t_0 is the original predicted class,
- $f_i(x)$ is the predicted probability for class i,
- $\kappa \geq 0$ is a confidence margin.

L_1, L_2 Regularizations (Sparsity Term, Stability Term). The L_1 and L_2 norms of the perturbation:

$$\|\delta\|_1 = \sum_{k=1}^{d} |\delta_k| \qquad \|\delta\|_2^2 = \sum_{k=1}^{d} \delta_k^2 \tag{5}$$

L_1 encourages sparse changes, while L_2 penalizes large deviations and smooths the optimization landscape. Together, the L_1 and L_2 terms form an elastic-net type regularization that balances sparsity and smoothness.

Autoencoder Reconstruction Loss L_{AE}. Using an $AE(\cdot)$ trained on the data distribution, this loss encourages plausibility.

$$L_{AE} = \gamma\|x_0 + \delta - AE(x_0 + \delta)\|_2^2, \tag{6}$$

This term is motivated by the following assumption: if an autoencoder (AE) is trained on the data distribution, it should reconstruct with high fidelity any sample that remains in-distribution, and it should reconstruct poorly samples that are out-of-distribution.

Therefore, the reconstruction loss penalizes perturbed instances that move too far from the learned data manifold.

3.4 Prototype Loss Term

To integrate prototypes into the optimization process, the authors introduce an additional loss term:

$$L_{\text{proto}} = \theta \|ENC(x_0 + \delta) - proto_j\|_2^2, \tag{7}$$

This loss encourages the perturbed instance to move toward a dense and representative region of the target class, and provides a clear direction in the latent space

The full optimization objective becomes:

$$\min_{\delta} \; c \cdot L_{\text{pred}} + \beta \|\delta\|_1 + \|\delta\|_2^2 + L_{AE} + L_{\text{proto}}. \tag{8}$$

To sum up, prototype-guided optimization, pulls the instance toward the prototype of a target-class neighborhood. This transforms the search from boundary-crossing into manifold-following.

Importantly, once the optimization is strongly guided by the prototype term, the prediction loss can in some cases be removed, as the attraction toward a representative target-class region is often sufficient to induce a class change.

Overall, the prototype-guided formulation integrates prediction change, sparsity, and in-distribution-ness through, addressing the main weaknesses of standard counterfactual optimization.

3.5 Algorithm

Algorithm 1 Counterfactual Guided by Prototypes generation (CGP) algorithm

Require: Parameters $\beta, \theta, c, \kappa, \gamma$, AE, training samples $\mathcal{X} = \{x_1, \ldots, x_n\}$, original sample x_0.

Prediction Step:

Compute predicted class of x_0:

$$t_0 = \arg\max_i f_{\text{pred}}(x_0)$$

Label training samples using f_{pred}:

$$\mathcal{X}_i = \{x \in \mathcal{X} \mid \arg\max f_{\text{pred}}(x) = i\}$$

Prototype Construction:

for each class i **do**

Encode samples in latent space:

$$z = ENC(x), \quad x \in \mathcal{X}_i$$

Define class prototypes:

$$proto_i = \frac{1}{K} \sum_{k=1}^{K} ENC(x_{ik})$$

end for

Target Prototype Selection:

Choose nearest prototype different from t_0:

$$j = \arg\min_{i \neq t_0} \|ENC(x_0) - proto_i\|_2$$

Optimization Step:

Solve:

$$\delta^* = \arg\min_{\delta \in \mathcal{X}} \; cL_{\text{pred}} + \beta \|\delta\|_1 + \|\delta\|_2^2 + L_{AE} + L_{\text{proto}}$$

Return:

$$x^{cf} = x_0 + \delta^*$$

4 Experimental Setup

4.1 Dataset

The experiments are conducted on the MNIST handwritten digit dataset, which consists of grayscale images 28×28 of handwritten digits from 0 to 9.The dataset contains 60,000 training images and 10,000 test images.

Due to computational limitations, I could not reproduce the experiments on the other datasets considered in the paper (Breast Cancer Wisconsin and Adult/Census), so I focused on MNIST.

4.2 Models

Two neural architectures are used in the experiments: a classifier and an autoencoder $AE(\dot{})$.

Classifier. The digit classifier is a convolutional neural network composed of:

- Two convolutional layers with ReLU activations,
- Each followed by 2×2 max-pooling,
- A fully connected layer,
- A final softmax output layer over the 10 digit classes.

The network achieves high classification accuracy on the test set, ensuring that generated counterfactuals reflect meaningful decision boundary crossings rather than model instability.

Autoencoder. An autoencoder is trained to compress the inputs into a latent representation and then reconstruct the input from this latent representation. It consists of:

- An encoder composed of same convolutional layers as the classifier.
- A symmetric decoder that reconstructs the image from the latent code.

The encoder $ENC(\cdot)$ is used to construct class prototypes in latent space. The full autoencoder $AE(\cdot)$ is used to evaluate reconstruction-based interpretability metrics.

Additionally, class-specific autoencoders AE_i are trained using only images from digit class i. These models allow evaluation of whether a generated counterfactual better aligns with the target-class distribution than with the original class.

4.3 Evaluation Metrics

Counterfactual explanations are evaluated using two metrics:e

4.3.1 IM1: Class-Specific Reconstruction Ratio

The first metric measures how well a counterfactual aligns with the target class manifold compared to the original class manifold.

- AE_i be an autoencoder trained only on images of class i,
- AE_{t_0} be an autoencoder trained on the original class.

For a counterfactual x^{cf} with target class i, IM1 is defined as:

$$IM1 = \frac{\|x^{cf} - AE_i(x^{cf})\|_2^2}{\|x^{cf} - AE_{t_0}(x^{cf})\|_2^2 + \varepsilon}. \tag{9}$$

- If $IM1 < 1$, the counterfactual is reconstructed better by the target-class autoencoder than by the original-class autoencoder.
- Lower values indicate better class-conditional alignment.

This metric evaluates whether the counterfactual lies closer to the data manifold of the target class rather than that of the original class.

4.3.2 IM2: Reconstruction Similarity

The second metric evaluates how consistent the reconstruction of a counterfactual is when using a target-class autoencoder versus a global autoencoder trained on all classes.

It is defined as:

$$IM2 = \frac{\|AE_i(x^{cf}) - AE(x^{cf})\|_2^2}{\|x^{cf}\|_1 + \varepsilon}. \tag{10}$$

- A small IM2 value indicates that the target-class autoencoder reconstructs the counterfactual similarly to the global autoencoder.
- This suggests that the counterfactual is consistent with the global data distribution while also matching the target class.

4.4 Results

In this part, we run the CGP algorithm using the following hyperparameter settings:

Table 1: Hyperparameters used for the original method.

c	β	θ	κ	γ	K	lr	max_iter	Method
1.0	0.1	200	0.01	100	K_{1O}	10^{-2}	5000	`Original`

The method used for prototype definition is: Original or KMeans

4.5 MNIST Results

Qualitative results. Figure 3 illustrates representative counterfactual examples on MNIST together with a 2D latent projection of the encoded data manifold.

In the first example (9→4), the counterfactual changes the digit structure (loop and stroke) to match a typical "4", rather than adding pixel-level noise. In latent space, the point moves clearly from the class-9 cluster toward the class-4 prototype and ends in a high-density target region.

In the second example (1→8), the thin vertical stroke is transformed into two loops, consistent with the morphology of an "8". Again, the latent trajectory shifts from the original cluster toward the target prototype, indicating a semantic (not adversarial) displacement.

The experiment in Table 2 analyzes the impact of L_{proto} on the counterfactual search process, using an encoder to define prototypes with $K = 10$. We also investigate the importance of the L_{AE} and L_{pred} terms in the presence of L_{proto}. We compare counterfactuals obtained with the following objective functions:

$$\begin{aligned}
A &= c\,L_{\text{pred}} + \beta\,\|\delta\|_1 + \|\delta\|_2^2 \\
B &= c\,L_{\text{pred}} + \beta\,\|\delta\|_1 + \|\delta\|_2^2 + L_{AE} \\
C &= c\,L_{\text{pred}} + \beta\,\|\delta\|_1 + \|\delta\|_2^2 + L_{\text{proto}} \\
D &= c\,L_{\text{pred}} + \beta\,\|\delta\|_1 + \|\delta\|_2^2 + L_{AE} + L_{\text{proto}} \\
E &= \beta\,\|\delta\|_1 + \|\delta\|_2^2 + L_{\text{proto}} \\
F &= \beta\,\|\delta\|_1 + \|\delta\|_2^2 + L_{AE} + L_{\text{proto}}
\end{aligned}$$

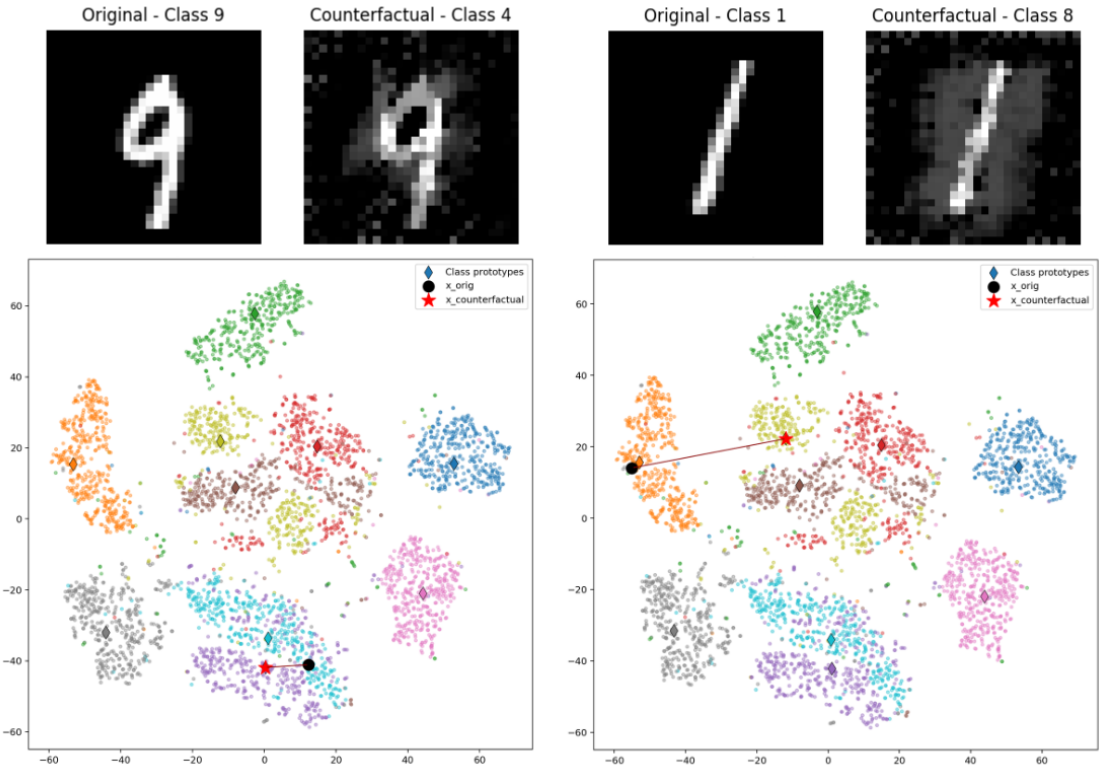


Figure 3: Results of the algorithm run on two different samples from the MNIST dataset. The first row shows x_{orig} and the corresponding x_{cf} generated toward the targeted class. The second row shows the generation process in latent space and how x_{orig} moves from the original class to the targeted class to create x_{cf}.

For each of the ten classes, we randomly sample 10 instances (compared to 50 in the original paper) from the test set and generate counterfactuals using 3 random seeds (same as in the paper). This yields a total of 1800 counterfactuals per objective (1500x6 in the paper).

Table 2: Quantitative results for objective functions A–F (mean ± standard deviation). Lower is better for time, gradient updates, IM1, and IM2.

Method	Time (s)	Gradient updates	IM1	IM2
A	2.082 ± 0.489	475.270 ± 107.980	1.044 ± 0.522	0.321 ± 0.280
B	2.267 ± 0.959	403.070 ± 169.560	1.687 ± 3.655	0.279 ± 0.272
C	1.899 ± 1.046	366.130 ± 201.394	1.072 ± 0.512	0.262 ± 0.215
D	1.439 ± 1.260	230.190 ± 201.960	1.260 ± 1.271	0.280 ± 0.241
E	1.511 ± 0.824	371.570 ± 203.153	1.070 ± 0.511	0.262 ± 0.215
F	1.228 ± 1.062	233.310 ± 202.081	1.204 ± 1.156	0.276 ± 0.241

Table 2 summarizes the findings for the speed and interpretability measures.

Speed. Time & Gradient updates columns in Table 2 reports the mean runtime and number of gradient updates required to generate a valid counterfactual for each objective, with standard deviations indicating variability across runs.
For objective A, a substantial fraction of the compute time is spent tuning c to balance L_{pred} against elastic-net regularization.

Objective B adds L_{AE} mainly to enforce plausibility (in-distribution behavior) rather than to accelerate convergence, and its speed gains are inconsistent.

Adding L_{proto} (objective C) markedly reduces both runtime and iterations compared to A, highlighting the benefit of prototype guidance.

Combining L_{AE} and L_{proto} (objective D) further improves efficiency and yields the fastest convergence among the differentiable objectives.

For black-box models, numerical gradients for L_{pred} can be a bottleneck because they require many prediction queries, scaling with input dimensionality.

Objectives E and F remove L_{pred}, resulting in a great speed-up over the other variants.

Quantitative interpretability. Interpretability is evaluated using IM1 and IM2, which quantify how well a counterfactual aligns with the target-class manifold. Lower values indicate better class-specific consistency.

Objective A shows the highest IM1, meaning that prediction-driven optimization with elastic-net regularization yields sparse but weakly class-consistent counterfactuals. Adding L_{AE} (B) slightly improves IM2 but remains unstable, indicating that global reconstruction alone does not ensure class-specific interpretability.

In contrast, objectives with L_{proto} (C–F) consistently lower IM1 and IM2, confirming that prototype guidance improves alignment with the target-class distribution. Removing L_{pred} (E, F) has little impact, showing that L_{proto} is the main driver of quantitative interpretability.

4.6 Visual interpretability

Figure 4 shows counterfactual examples on the first row and their reconstructions using AE on the second row for different loss functions.

The counterfactuals generated with A or B are sparse but remain visually close to the original digit and remain in the original class. Their reconstructions confirm that these samples still lie near the manifold of the original class rather than forming a clean representation of the target class.

In contrast, methods including the prototype loss L_{proto} (C–F) produce counterfactuals that show more clearly a typical instance of the target digit (3 in this case), with smoother and more coherent structural changes.

The corresponding AE reconstructions are sharper and class-consistent, illustrating that prototype guidance leads to visually and distributionally more interpretable counterfactuals.

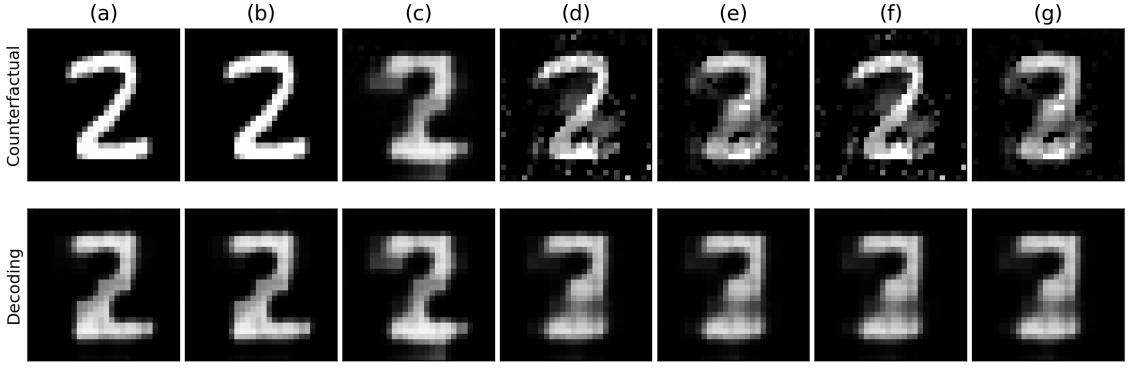


Figure 4: (a) Shows the original instance, (b) to (g) on the first row illustrate counterfactuals generated by using loss functions A to F. (b) to (g) on the second row show the reconstructed counterfactuals using AE.

4.7 Zoom on Losses

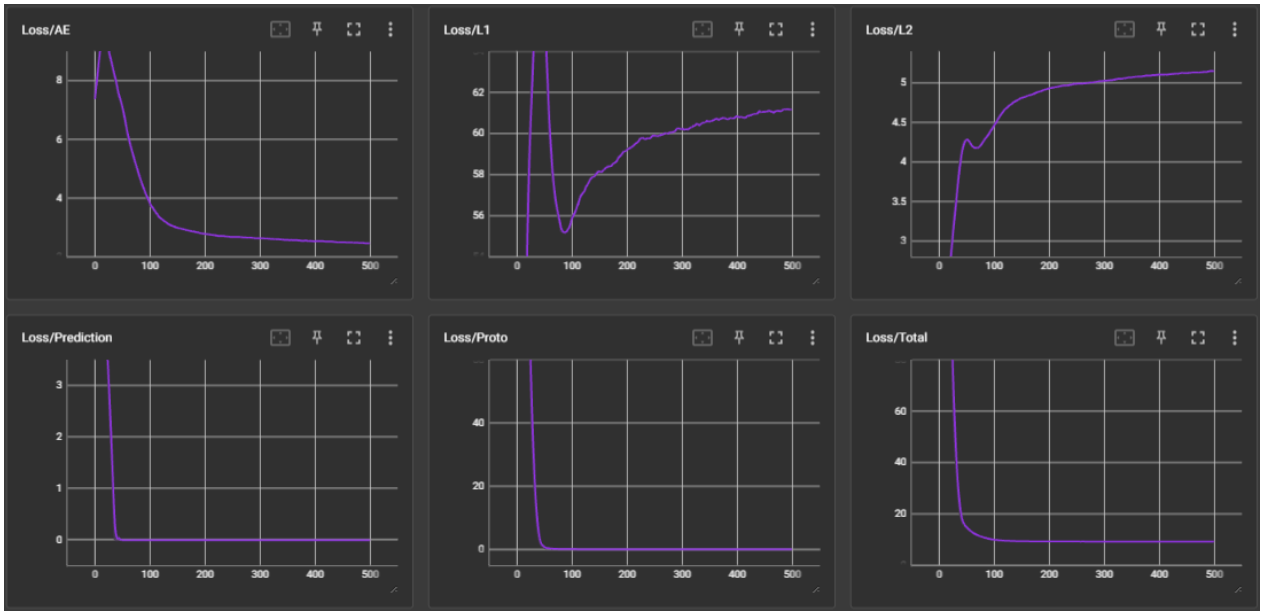


Figure 5: Evolution of Losses: L_{pred}, $L1$, $L2$, L_{AE}, L_{proto} for the image 9→4.

Figure 5 illustrates the evolution of each loss term as well as the total loss. The latter drops sharply during the first iterations, indicating that the optimization quickly moves the counterfactual toward a feasible solution.

Both the prediction L_{pred} and prototype losses L_{proto} rapidly converge to (near) zero, showing that the class change is achieved early and that the instance is efficiently guided toward the target prototype.

The autoencoder loss L_{AE} steadily decreases and then stabilizes, suggesting improved alignment with the target class data manifold.

In contrast, the L_1 and L_2 terms increase over the iterations, reflecting the trade-off between sparsity and manifold alignment as the perturbation magnitude grows to satisfy the prototype and reconstruction constraints.

5 Additional experiments

5.1 CIFAR-10

To extend the paper's results, we propose evaluating the method on a more challenging dataset such as CIFAR-10.

The paper demonstrates strong performance on MNIST, but MNIST is relatively simple (grayscale, single channel, 28×28). A CIFAR-10 study would better test robustness on natural images since it is more challenging, suggesting 32×32, RGB images.

The goal of this part is to scale the method to higher-resolution, colored images and to test its generalizability on real natural images.

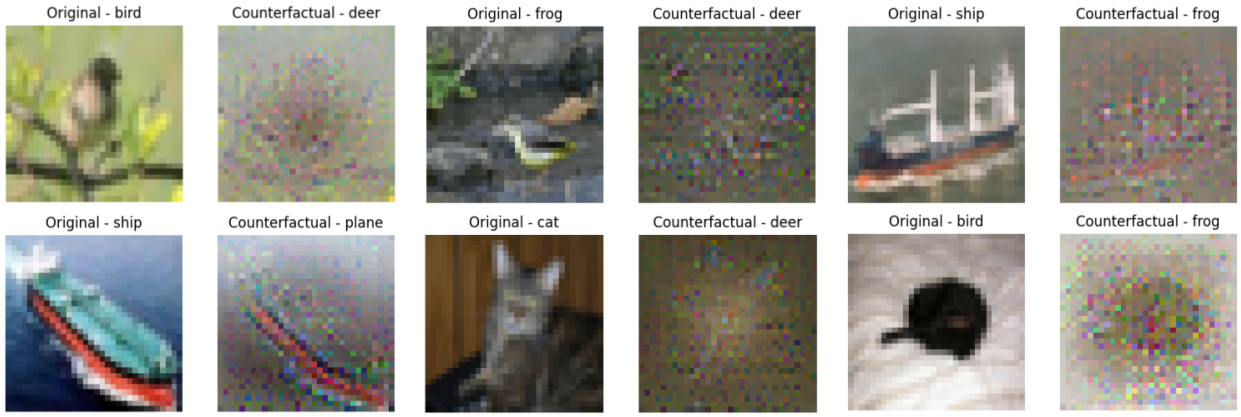


Figure 6: Results on CIFAR10 dataset.

As shown in Figure 6, the algorithm's output is not very compromising, the counterfactual generated, are simply high fraquency noise, more similar to adversarial attacks than a semantically coherent transformation of the image.

As a conclusion, we can state that scaling the method from MNIST to CIFAR-10 reveals important limitations in its robustness and generalization. While the approach remains effective on low-dimensional, grayscale digit data, it does not naturally extend to more complex natural images without additional constraints or stronger generative priors.

In summary, this experiment suggests that the method does not scale beyond MNIST. Since it fails on CIFAR-10, it is unlikely to succeed on even more challenging datasets such as ImageNet or Caltech101, which involve higher-resolution images (256×256, 512×512); see Figure 7.

Figure 7: Example results on Caltech101.

5.2 Pixel selection

In this part, we introduce visual explanations to test whether the algorithm can recover the original distribution after masking some pixels and reconstructing from the perturbed image.

The goal of this part is to show that the proposed method can faithfully recover the original distribution using only the objective function in Equation 8.

Let's assume we have a test image and an attribution method (e.g., Saliency[6], Grad-CAM[5], Integrated Gradients[8]). The method outputs a heatmap in which the most important pixels for the model's prediction have higher values. If we mask the top-k most important pixels, we obtain a perturbed image with missing pixels. This perturbed image is misclassified and no longer associated with the original class. Using the proposed method, we then try to move it back toward the original prototype. Intuitively, the method should re-fill the masked pixels to recover the original shape while moving closer to the original prototype.

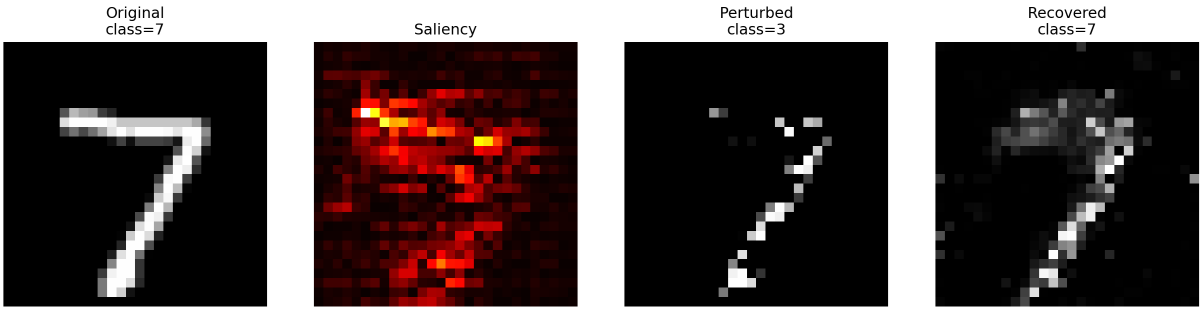


Figure 8: Pixel selection experiment. From left to right: original image (class 7), saliency heatmap, perturbed image after masking top-k salient pixels (misclassified as class 3), and recovered image after optimization toward the original prototype (class 7).

Observations. Figure 8 illustrates the full pipeline. The saliency map highlights the horizontal stroke and upper junction of the digit as the most influential regions for predicting class 7. After masking these top-k pixels, the perturbed image loses critical structural components and is misclassified as class 3. This confirms that the attribution method correctly identifies pixels that are essential for the classifier's decision.

When applying the proposed optimization procedure with the original class prototype as target, the recovered image regains the missing horizontal stroke and restores the global structure of the digit. The reconstruction is not a simple denoising effect: the optimization selectively reintroduces class-consistent patterns that align with the prototype in latent space. The final image is again classified as class 7, demonstrating that the method successfully moves the perturbed sample back toward the original data manifold.

6 Conclusion

This report examined *Interpretable Counterfactual Explanations Guided by Prototypes*, revisited its formulation, and reproduced key experiments while proposing extensions.

We clarified the role of the prototype loss in steering optimization toward class-consistent regions and summarized the speed and interpretability trade-offs across objective variants. On MNIST, prototype-guided objectives reduced runtime and improved manifold alignment compared with prediction-only baselines, supporting the paper's central claims.

We then explored extensions. The K-Means prototype strategy yielded more representative class anchors in latent space, reducing cases where nearest-neighbor prototypes fall outside the class distribution.

In contrast, scaling to *CIFAR-10* exposed limitations: the method tended to produce high-frequency, adversarial-like artifacts rather than semantically coherent counterfactuals, indicating that additional generative constraints or stronger priors are needed for complex natural images. The *Caltech-101* example further underscores these scalability challenges.

Finally, the pixel-selection experiment showed that when salient regions are masked, the optimization can recover class-consistent structure and return the sample toward the original prototype, suggesting the method is effective at reconstruction in low-dimensional settings.

Overall, prototype guidance improves interpretability and efficiency on simple datasets, but robustness and realism degrade on higher-resolution, multi-class image data, pointing to clear directions for future work.

References

[1] Amit Dhurandhar, Pin-Yu Chen, Ronny Luss, Chun-Chen Tu, PaiShun Ting, Karthikeyan Shanmugam, and Payel Das. Explanations based on the missing: Towards contrastive explanations with pertinent negatives. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 31, pages 592–603, 2018.

[2] Thibault Laugel, Marie-Jeanne Lesot, Christophe Marsala, Xavier Renard, and Marcin Detyniecki. Comparison-based inverse classification for interpretability in machine learning. In *Information Processing and Management of Uncertainty in Knowledge-Based Systems*, pages 100–111. Springer, 2018.

[3] Scott M Lundberg and Su-In Lee. A unified approach to interpreting model predictions. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 30, 2017.

[4] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. "why should i trust you?": Explaining the predictions of any classifier. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*, pages 1135–1144. ACM, 2016. doi: 10.1145/2939672.2939778.

[5] Ramprasaath R. Selvaraju, Abhishek Das, Ramakrishna Vedantam, Michael Cogswell, Devi Parikh, and Dhruv Batra. Grad-cam: Visual explanations from deep networks via gradient-based localization. *International Journal of Computer Vision*, 128:336–359, 2016.

[6] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *CoRR*, abs/1409.1556, 2014. URL https://api.semanticscholar.org/CorpusID:14124313.

[7] Jake Snell, Kevin Swersky, and Richard Zemel. Prototypical networks for few-shot learning. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 30, 2017.

[8] Mukund Sundararajan, Ankur Taly, and Qiqi Yan. Axiomatic attribution for deep networks. In Doina Precup and Yee Whye Teh, editors, *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pages 3319–3328. PMLR, 06–11 Aug 2017. URL https://proceedings.mlr.press/v70/sundararajan17a.html.

[9] Sandra Wachter, Brent Mittelstadt, and Chris Russell. Counterfactual explanations without opening the black box: Automated decisions and the gdpr. *Harvard Journal of Law & Technology*, 31(2), 2018.