

Article

Not peer-reviewed version

Interpretable Deep Prototype-Based Neural Networks: Can a 1 Look Like a 0?

[Esteban García-Cuesta](#)*, [Daniel Manrique](#), [Radu Constantin Ionescu](#)

Posted Date: 29 July 2025

doi: 10.20944/preprints202507.2389.v1

Keywords: Prototype-Based Networks; Interpretable AI; Robustness of Explanations; Neural Networks




Preprints.org is a free multidisciplinary platform providing preprint service that is dedicated to making early versions of research outputs permanently available and citable. Preprints posted at Preprints.org appear in Web of Science, Crossref, Google Scholar, Scilit, Europe PMC.

Copyright: This open access article is published under a Creative Commons CC BY 4.0 license, which permit the free download, distribution, and reuse, provided that the author and preprint are cited in any reuse.

Disclaimer/Publisher's Note: The statements, opinions, and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions, or products referred to in the content.

Article

Interpretable Deep Prototype-Based Neural Networks: Can a 1 Look Like a 0?

Esteban García-Cuesta * , Daniel Manrique  and Radu Constantin Ionescu

Departamento de Inteligencia Artificial, ETSI Informáticos, Universidad Politécnica de Madrid, Spain

* Correspondence: esteban.garcia@upm.com; Tel.: +34-91-067-2887

Abstract

Prototype-Based Networks (PBNs) are inherently interpretable architectures that facilitate understanding of model outputs by analyzing the activation of specific neurons—referred to as prototypes—during the forward pass. The learned prototypes serve as transformations of the input space into a latent representation that more effectively encapsulates the main characteristics shared across data samples, thereby enhancing classification performance. Crucially, these prototypes can be decoded and projected back into the original input space, providing direct interpretability of the features learned by the network. While this characteristic marks a meaningful advancement toward the realization of fully interpretable artificial intelligence systems, our findings reveal that prototype representations can be deliberately or inadvertently manipulated without compromising the superficial appearance of explainability. In this study, we present a series of empirical investigations that substantiate this phenomenon and introduce it as a structural paradox inherent to the architecture itself, which may pose a significant robustness concern for explainable AI methodologies.

Keywords: prototype-based networks; interpretable AI; robustness of explanations; neural networks

1. Introduction

Model interpretability is a critical requirement in the development of machine learning models intended to represent aspects of real-world phenomena. Numerous techniques and studies have sought to enhance interpretability, primarily through post hoc analyses or by employing surrogate models that are inherently more interpretable. Despite these efforts, a fundamental challenge persists: establishing a clear correspondence between the knowledge acquired by the model and observable elements of reality. In an effort to address this issue, a Prototype Deep Neural Network (PDNN) was introduced in [1]. Prototype learning is a subclass of intrinsically interpretable methods, derived from the classical paradigm of case-based reasoning known as prototype classification. In this context, a prototype is not constrained to a single observation from the training set, but rather may constitute a generalized representation formed from a combination of multiple samples in the learned latent space [2]. The prototype learning framework typically incorporates an autoencoder in conjunction with a prototype classifier layer. The autoencoder generates a compressed latent representation of the input data, while the prototype layer identifies representative points within this space that both resemble specific inputs and are indicative of particular classes. Each learned prototype can subsequently be visualized through the decoder, enabling interpretability of the learned features, and can be traced back to the output softmax layer, thereby providing insight into class associations. Moreover, this architectural design supports a form of case-based reasoning by identifying the training sample(s) most similar to a given prototype, further enhancing the transparency and accountability of the model's decision-making process.

In this context, while the interpretability of the method has received considerable attention, the reliability and validity of the learned prototypes have often been overlooked. For relatively simple datasets, such as handwritten digits [3], prototype validation is typically performed by reconstructing

and visualizing the prototypes in the input space, followed by a qualitative assessment of whether the prototypes visually resemble the intended classes. However, this raises a critical question: can we confidently assert that a prototype which appears to represent, for instance, the digit '1', is indeed interpreted as such by the deep neural network (DNN)? In this study, we address this question through a series of experiments, including the development of a novel Deep Prototype-Based Network. Our results demonstrate that visual similarity does not necessarily align with the model's internal representation, highlighting a potential gap between human interpretability and model-driven classification.

2. State of the Art

[4] were the first to show that deep networks can reason through learned prototypes while remaining end-to-end trainable. Prototype-based neural networks offer a hybrid approach that combines the power of deep learning with the interpretability of case-based reasoning. Its architecture integrates a convolutional autoencoder with a prototype-based classification layer, allowing the model to explain its predictions through comparisons with prototypical training examples. Building upon this foundational work, [5] proposed ProtoPNet, a model that popularized the "this-looks-like-that" paradigm through patch-to-prototype visualizations. The training methodology incorporated techniques such as prototype projection and class-specific weight regularization, achieving competitive performance across fine-grained classification tasks involving bird, car, and dermatological image datasets, while preserving interpretable, image-level justifications.

Two primary strategies have been proposed to overcome the representational limitations inherent in single-layer prototype-based models. First, [6] introduced ProtoTree, a differentiable decision tree architecture in which internal nodes correspond to learned prototypes. This structure enables an input to traverse only $O(\log K)$ nodes, producing compact and interpretable rule chains. Second, explicit prototype hierarchies [7] organize low-level visual parts into increasingly abstract semantic concepts, facilitating multi-granular reasoning and enhancing robustness to out-of-distribution inputs.

Hybrid approaches have also been developed to enable reasoning beyond raw pixel representations. Concept Bottleneck Models (CBMs) [8] structure the prediction pipeline around human-interpretable attributes, allowing users to intervene and correct misclassified concepts at inference time. In a complementary direction, the Deep k-Nearest Neighbours (DkNN) algorithm [9] enhances arbitrary classifiers by incorporating layer-wise nearest neighbor retrieval, thereby quantifying model uncertainty through non-conformity scores. Both methods underscore the utility of leveraging intermediate representations, particularly similarity structures within hidden layers, without compromising predictive accuracy, a principle that underpins the multi-layer prototype architecture proposed in this work. Additionally, Capsule Networks [10] implement dynamic routing mechanisms to capture part-to-whole relationships, while part-based R-CNNs [11] localize discriminative object regions.

The Deep Prototype Network (DPN) introduced in this work adopts a compositional reasoning framework akin to prior hierarchical or part-based models, but implements this structure through a deliberately layered architecture rather than relying on post hoc clustering of prototypes. This design choice is motivated by the principle that successive prototype layers should progressively capture increasingly abstract semantic concepts or larger spatial extents, thereby enabling richer part-to-whole reasoning directly within the network's forward pass.

3. Deep Prototype-Based Network Architecture

Figure 1 illustrates the proposed Deep Prototype-Based Network (DPBN) architecture, which extends Li et al. [2018]'s notation, cost terms, and design philosophy, but stacks several prototype layers in a feedforward chain rather than using a single one. This architecture performs a back-and-forth data flow that couples representational capacity with prototype-based interpretability.

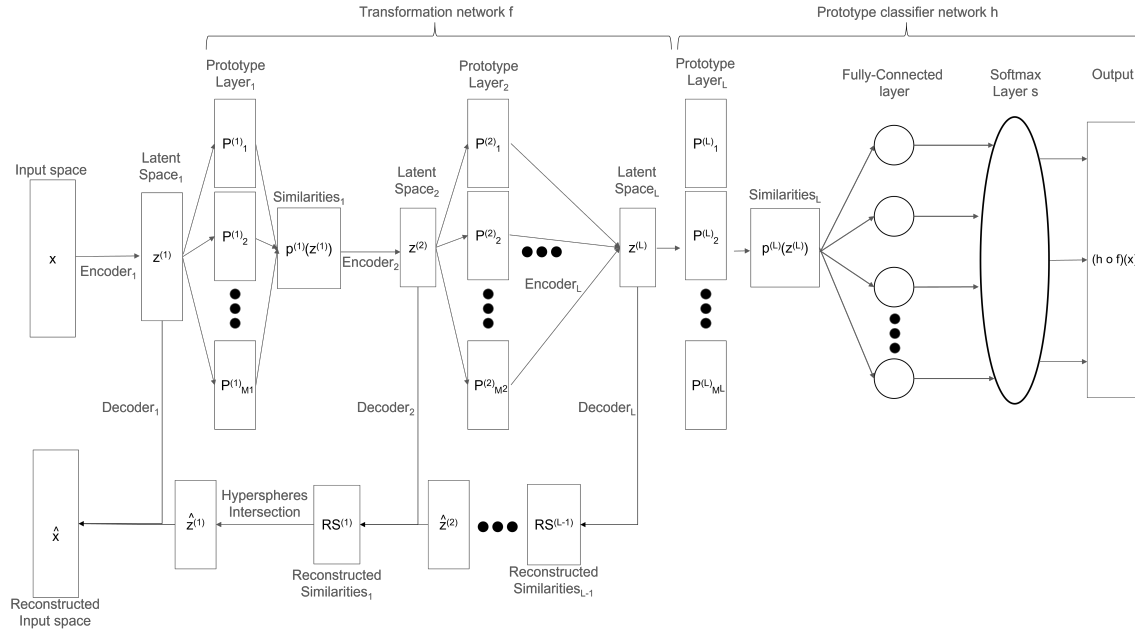


Figure 1. DPBN architecture.

Let $\mathcal{D} = \{(x_i, y_i)\}_{i=1}^N \subset \mathcal{X} \times \{1, \dots, K\}$, $\mathcal{X} \subset \mathbb{R}^p$, denote a finite training set sampled from an underlying distribution \mathbb{P}_{XY} . All spaces are Euclidean and, therefore, equipped with the canonical inner product $\langle \cdot, \cdot \rangle$ and the induced norm $\|\cdot\|_2$. The uppercase calligraphic symbols note sets or spaces, and the superscripts in parentheses index the depth $\ell \in \{1, \dots, L\}$. The set of all network parameters is denoted as $\Theta = \{\theta_{f(1)}, \{\theta_{e(\ell)}, \theta_{d(\ell)}\}_{\ell=2}^L, \{P_j^{(\ell)}\}, W\}$.

3.1. The Forward Encoding Flow

The upper part of Figure 1 shows that the forward flow begins in the input space, followed by the first encoder, which maps the input data to a latent space. Then, the transformation network f comprises $L-1$ stacked triples consisting of a prototype layer, the calculation of a similarity vector, and an encoder for the following layer. Finally, the prototype classifier network h takes as input the last prototype layer (L), calculates the similarity vector, and passes the results through a linear mapping with a subsequent softmax layer that produces the estimated classification distribution of an input.

The input space domain \mathcal{X} is a subset of a p -dimensional real vector space. A sample is denoted by $x \in \mathcal{X}$. No further structure is assumed, except for the differentiability of all subsequent mappings defined on \mathcal{X} .

The first layer of the DPBN is the first encoder, which projects input data onto a lower-dimensional manifold:

$$f^{(1)} : \mathcal{X} \longrightarrow \mathcal{Z}_1, \quad \mathcal{Z}_1 = \mathbb{R}^{q_1}, \quad q_1 \ll p, \\ f^{(1)}(x) = z^{(1)}.$$

The mapping $f^{(1)}$ is usually implemented using a convolutional neural network that ends with a fully connected layer of width q_1 . The parameters associated with this mapping are collectively denoted by $\theta_{f(1)}$ and form part of the general parameter set.

The next layer is the first prototype layer $\mathcal{P}^{(1)}$,

$$\mathcal{P}^{(1)} = \{P_j^{(1)}\}_{j=1}^{M_1} \subset \mathcal{Z}_1,$$

and comprises a finite set of prototypes. The value $M_1 \in \mathbb{N}$ is a fixed hyperparameter a priori, while prototypes are learned during the training process. Given a latent representation $z^{(1)}$, its similarity

vector is calculated using the squared Euclidean distance to all prototypes. It is collected in the similarity vector

$$p^{(1)}(z^{(1)}) = \left(d_1^{(1)}(z^{(1)}), \dots, d_{M_1}^{(1)}(z^{(1)})\right)^\top \in \mathbb{R}^{M_1}, d_j^{(1)}(z^{(1)}) = \|z^{(1)} - P_j^{(1)}\|_2^2,$$

which defines M_1 differentiable functions $d_j^{(1)} : \mathcal{Z}_1 \rightarrow \mathbb{R}^+$, thereby enabling the application of gradient-based optimization methods without requiring any modifications.

The next layer is the second encoder. The resulting similarity vector $p^{(1)}(z^{(1)})$ computed in the previous layer, which does not contain any positional ordering, is compressed by a learned *distance encoder* to obtain the second layer representation.

$$e^{(2)} : \mathbb{R}^{M_1} \longrightarrow \mathcal{Z}_2, \quad \mathcal{Z}_2 = \mathbb{R}^{q_2},$$

$$z^{(2)} = e^{(2)}(p^{(1)}(z^{(1)})).$$

The function $e^{(2)}$ is implemented as a multilayer perceptron, with its parameters collectively denoted by $\theta_{e^{(2)}}$. The representation $z^{(2)}$ can be expressed as $z^{(2)} = e^{(2)} \circ p^{(1)} \circ f^{(1)}(x)$ using the composition of functions.

Similarly to the first layer, for every subsequent layer $\ell \in \{2, \dots, L\}$, there is an encoder that generates the latent space \mathcal{Z}_ℓ , followed by a prototype set $\mathcal{P}^{(\ell)} = \{P_j^{(\ell)}\}_{j=1}^{M_\ell} \subset \mathcal{Z}_\ell$, then a similarity vector $p^{(\ell)}(z^{(\ell)})$, and finally an encoder for the next layer $\ell + 1$:

$$p^{(\ell)}(z^{(\ell)}) = \left(d_1^{(\ell)}(z^{(\ell)}), \dots, d_{M_\ell}^{(\ell)}(z^{(\ell)})\right)^\top \in \mathbb{R}^{M_\ell},$$

$$e^{(\ell+1)} : \mathbb{R}^{M_\ell} \longrightarrow \mathcal{Z}_{\ell+1}, \quad z^{(\ell+1)} = e^{(\ell+1)}(p^{(\ell)}(z^{(\ell)})).$$

We denote as $\Phi^{(\ell)}(x) = e^{(\ell)} \circ p^{(\ell-1)} \circ e^{(\ell-1)} \circ \dots \circ p^{(1)} \circ f^{(1)}(x) = z^{(\ell)}$ the composite mapping that transforms x into $z^{(\ell)}$.

To provide the final classification decision after passing through all layers of the DPBN, the final similarity vector $p^{(L)}(z^{(L)}) \in \mathbb{R}^{M_L}$ is fed into a linear map $W \in \mathbb{R}^{K \times M_L}$ to obtain

$$\eta(x) = W p^{(L)}(z^{(L)}) \in \mathbb{R}^K.$$

The subsequent softmax activation function produces the estimated class distribution $\hat{y}(x)$, where $\hat{c}(x) = \arg \max \hat{y}(x)$ is the predicted class.

A summary of the forward flow from the input vector x to the predicted class distribution is as follows:

$$x \xrightarrow{f^{(1)}} z^{(1)} \xrightarrow{\mathcal{P}^{(1)}} p^{(1)}(z^{(1)}) \xrightarrow{e^{(2)}} z^{(2)} \xrightarrow{\mathcal{P}^{(2)}} \dots \xrightarrow{e^{(L)}} z^{(L)} \xrightarrow{\mathcal{P}^{(L)}} p^{(L)}(z^{(L)}) \xrightarrow{W+Softmax} \hat{y}.$$

3.2. The Backward Decoding Flow

What distinguishes prototype-based networks, and in particular the proposed DPBN, from regular neural networks is the presence of prototypes that, through their reconstruction back into the input space, offer visual explanations for their predictions. However, this visualization is not straightforward in the case of the DPBN. The procedure to visualize deeper prototypes introduces a novel approach called the hypersphere intersection (HS-Int) algorithm, which transforms a reconstructed similarity vector at deep ℓ , $RS^{(\ell)}$ in Figure 1 or $\tilde{p}^{(\ell)}$, into a reconstructed code $\tilde{z}^{(\ell)}$.

The lower part of Figure 1 shows the backward decoding flow that reconstructs the deepest code $z^{(L)}$ back into the input space. This reconstruction process performs $L-1$ decoding and hypersphere-

intersection pairs until obtaining the first reconstructed code $\tilde{z}^{(1)}$. Finally, $\tilde{z}^{(1)}$ is decoded back into \tilde{x} .

The hypersphere intersection algorithm is based on previous results from algebraic and geodesy/GPS positioning [12–14], as well as nonlinear least-squares optimization (Levenberg-Marquardt) [15,16] and the Moore-Penrose generalized inverse matrix theory [17,18].

For any depth $\ell \geq 2$, the decoder produces the radius vector

$$\hat{p}^{(\ell-1)} = (\hat{\rho}_1^{(\ell-1)}, \dots, \hat{\rho}_{M_{\ell-1}}^{(\ell-1)})^\top, \quad \hat{\rho}_j^{(\ell-1)} \geq 0,$$

which, together with the prototype set $\mathcal{P}^{(\ell-1)} = \{P_j^{(\ell-1)}\}_{j=1}^{M_{\ell-1}} \subset \mathcal{Z}_{\ell-1}$, defines $M_{\ell-1}$ hyperspheres in the ambient space $\mathcal{Z}_{\ell-1}$ as follows:

$$\mathcal{S}_j = \{z \in \mathcal{Z}_{\ell-1} : \|z - P_j\|_2^2 = \hat{\rho}_j^2\},$$

where $P_j = P_j^{(\ell-1)}$ and $\hat{\rho}_j^2 = (\hat{\rho}_j^{(\ell-1)})^2$ to simplify the notation. Thus, the objective becomes to recover a point $z^* \in \bigcap_{j=1}^{M_{\ell-1}} \mathcal{S}_j$, or, if the intersection is empty, the solution to the minimization problem

$$z^* = \min_{z \in \mathcal{Z}_{\ell-1}} \{F(z)\}, \quad F(z) = \sum_{j=1}^{M_{\ell-1}} (\|z - P_j\|_2^2 - \hat{\rho}_j^2). \quad (1)$$

We draw on previous studies in this field to design the algorithm in three steps. The first is to perform linear reduction by applying Bancroft's algebraic difference trick [12] to obtain the linear system:

$$Az = b, \quad A \in \mathbb{R}^{(M_{\ell-1}-1) \times q_{\ell-1}}, \quad b \in \mathbb{R}^{M_{\ell-1}-1}.$$

Its least-squares solution is $z_p = A^+b$, where A^+ is the Moore-Penrose pseudoinverse. Every other solution differs from z_p by an element of the null space of A :

$$z = z_p + N\alpha, \quad N \in \mathbb{R}^{q_{\ell-1} \times d}, \quad \alpha \in \mathbb{R}^d, \quad d = q_{\ell-1} - \text{rank}(A),$$

where $\text{rank}(A)$ represents the rank of matrix A , N stacks an orthonormal basis of $\ker A$, with $\ker A$ being the kernel of matrix A , so d equals the number of remaining degrees of freedom and α are their coordinates. If $d = 0$, the linear part has isolated at most one candidate; otherwise, the quadratic constraint of the next step must fix α (or show that there is no exact intersection).

The second step inserts the above-mentioned parameterization $z = z_p + N\alpha$ into the first sphere equation $\|z - P_1\|_2^2 = \hat{\rho}_1^2$, which yields

$$q(\alpha) = \|z_p + N\alpha - P_1\|_2^2 - \hat{\rho}_1^2 = 0.$$

If $d = 0$, no free parameters remain, and q reduces to a constant; the unique linear candidate z_p is an exact intersection; otherwise, no actual solution exists.

If $d = 1$, then $q(\alpha)$ is an ordinary quadratic in one variable. Its roots have closed forms tabulated by Norrdine [14, Alg. 2], producing up to two exact points, which we collect in $\mathcal{Z}_{\text{exact}}$.

Lastly, if $d \geq 2$, then it means that the same equation defines at most a $(d-1)$ -dimensional quadric in the α space. Therefore, we launch a Levenberg-Marquardt search [16] from $\alpha = 0$ to find a feasible α (or certify that none exists), following Norrdine's least squares recipe [14, Alg. 1].

Finally, any α that satisfies $q(\alpha) = 0$ is mapped through $z = z_p + N\alpha$ and added to $\mathcal{Z}_{\text{exact}}$ in the third step. However, if no exact intersection is found, ($\mathcal{Z}_{\text{exact}} = \emptyset$), we minimize the nonlinear least-squares cost

$$F(z) = \frac{1}{2} \sum_{j=1}^{M_{\ell-1}} (\|z - P_j\|_2^2 - \hat{\rho}_j^2)^2.$$

The optimization process uses the Levenberg-Marquardt-damped Gauss-Newton scheme [15,16], which smoothly interpolates between gradient descent and the Gauss-Newton step via an adaptive damping factor. After convergence, we obtain z^* . The local uniqueness is verified with the Jacobian rank criterion of Abel-Chaffee [13]. The algorithm finally returns $\tilde{z}^{(\ell-1)} = z^*$ together with the uniqueness flag.

Once the HS-Int algorithm is described, we can define the decoding cascade, which starts from $z^{(L)}$, applies a decoding step

$$\hat{p}^{(\ell-1)} = d^{(\ell)}(z^{(\ell)}),$$

and then applies the HS-Int algorithm

$$\tilde{z}^{(\ell-1)} = \text{HS-Int}(\mathcal{P}^{(\ell-1)}, \hat{p}^{(\ell-1)})$$

to $\ell = L, L-1, \dots, 2$. Note that the decoding step transforms a code from a latent space into a reconstructed similarity vector. Then, the HS-Int algorithm takes the reconstructed similarities vector and the set of prototypes from the corresponding prototype layer to generate the reconstructed code for the previous layer. Finally, $\tilde{x} = d^{(1)}(z^{(1)}) \in \mathcal{X}$ is the global reconstruction against which the reconstruction term in (2) is measured. This cascade enforces explicit geometric consistency between successive latent spaces, endowing every latent point with a deterministic pre-image in the input domain.

The reconstruction pathway can be summarized as

$$z^{(L)} \xrightarrow{d^{(L)}} \hat{p}^{(L-1)} \xrightarrow{\text{HSInt}} \tilde{z}^{(L-1)} \xrightarrow{d^{(L-1)}} \dots \tilde{z}^{(1)} \xrightarrow{d^{(1)}} \tilde{x}.$$

The proposed DPBN provides expressive capacity and interpretability. Every depth $\ell \geq 2$ applies the composition $e^{(\ell)} \circ p^{(\ell-1)}$, which is Lipschitz continuous when $e^{(\ell)}$ employs bounded activation functions, endowing the network with a hierarchical piecewise-smooth warping of the latent metric. Yet interpretability is preserved: each prototype $P_j^{(\ell)}$ is deterministically mapped to the input domain by the reconstruction chain, which yields a visual or otherwise perceptible exemplar representing that prototype at its innate semantic granularity.

3.3. Guided Prototype Learning

Together with the pipelines for classification and decoding, it is crucial to define the joint learning objective of the network architecture.

Let $\mathcal{L}_{\text{CE}}(\hat{y}, y) = -y \log \hat{y}$ be the cross-entropy between the prediction \hat{y} and the ground truth y . With $\{z_i^{(\ell)}\}_{i=1}^B$ referring to the latent codes of a mini-batch of size B , we define the layer-wise prototype coverages as:

$$R_1^{(\ell)} = \frac{1}{M_\ell} \sum_{j=1}^{M_\ell} \min_{i=1, \dots, B} \|P_j^{(\ell)} - z_i^{(\ell)}\|_2^2, \quad R_2^{(\ell)} = \frac{1}{B} \sum_{i=1}^B \min_{j=1, \dots, M_\ell} \|z_i^{(\ell)} - P_j^{(\ell)}\|_2^2.$$

Similarly, with a mini-batch of size B , we establish the loss functions of the autoencoders for the image and the distances as:

$$\mathcal{L}_{\text{img}}^{\text{AE}} = \frac{1}{B} \sum_{i=1}^B \|\tilde{x}_i - x_i\|_2^2, \quad \mathcal{L}_{\text{dist}}^{\text{AE}} = \frac{1}{B} \sum_{i=1}^B \sum_{\ell=2}^L \|\tilde{d}_i^{(\ell)} - d_i^{(\ell)}\|_2^2.$$

As such, for positive scalars acting as regularization terms λ_{R_1} , λ_{R_D} , $\lambda_{R_1}^{(\ell)}$, and $\lambda_{R_2}^{(\ell)}$, the total loss reads

$$\mathcal{L} = \frac{1}{B} \sum_{i=1}^B \mathcal{L}_{\text{CE}}(\hat{y}(x_i), y_i) + \lambda_{R_1} \mathcal{L}_{\text{img}}^{\text{AE}} + \lambda_{R_D} \mathcal{L}_{\text{dist}}^{\text{AE}} + \sum_{\ell=1}^L \left(\lambda_{R_1}^{(\ell)} R_1^{(\ell)} + \lambda_{R_2}^{(\ell)} R_2^{(\ell)} \right). \quad (2)$$

All model parameters Θ are trained by stochastic gradient descent on (2), with gradients obtained via automatic differentiation through the analytic Jacobian of HS-Int whenever the exact intersection is unique; otherwise, the sub-gradient of the best-fit residual is used.

4. Experiments and Results

In this section, we conduct a systematic evaluation of the proposed Deep Prototype-Based Network (DPBN), complemented by a series of targeted architectural variants. This analysis aims to demonstrate the model's capabilities while also addressing a key challenge: the tension between visually interpretable prototype reasoning and the internal mechanisms of class assignment within the network.

The first is a baseline experiment to assess performance on the MNIST dataset ([3]) and to examine the prototype representations obtained from the proposed DPBN architecture (see Figure 1). The second experiment is a model distillation study in which we investigate whether the prototypes learned by the deepest layer of a four-layer Deep Prototype-Based Network (DPBN) can be effectively transferred to a significantly shallower model, without compromising either predictive performance or interpretability. The third experiment addresses a binary classification task designed to demonstrate that the network can produce accurate predictions and visualizations consistent with the class "0", while its internal explanatory mechanisms align more closely with class "1".

For each experimental configuration, we report both classification accuracy and a novel validation metric introduced in this study—the Normalized Negative Entropy (NNE) score. Quantitative findings are further supported by qualitative analyses, comprising visualizations of reconstructed prototypes and learned weight matrices. Collectively, the results elucidate the impact of each architectural choice on the reliability, interpretability, and overall performance of DPBNs.

All the experiments are conducted using the MNIST ([3]) benchmark dataset. The original training set, comprising 60,000 images, is randomly partitioned into 90% for training and 10% for validation. The official test set of 10,000 images is held out for evaluation. All images are resized to 28×28 pixels and normalized to the $[0, 1]$ range using min-max scaling.

4.1. NNE Score

Prototype-based networks typically conclude with a linear classifier, where the weight matrix $W \in \mathbb{R}^{K \times M_L}$ maps prototype-sample distances to class logits. To assess the interpretability of this final classification layer, we introduce the Normalized Negative Entropy (NNE) score. This metric interprets the prototype-to-class weight matrix W as an information-theoretic channel. After negating W , a row-wise softmax is applied to obtain a class probability distribution for each prototype. The Shannon entropy of each distribution is then computed, inverted, normalized by $\log K$, and averaged across all prototypes. The resulting NNE score lies within the interval $[0, 1]$, attaining a maximum value of 1 only when each prototype exhibits a one-hot (i.e., perfectly class-specific) voting pattern.

In contrast to geometric alignment measures—such as the B-Cos transform [19], which enhances the fidelity of saliency maps by maximizing cosine similarity between inputs and classifier weights, or the support-vector-based alignment strategy employed by WASUP [20]—the NNE metric exclusively captures the distributional sharpness of the classification head, providing a complementary perspective on model interpretability.

Because large *negative* distances should yield a high vote, we first negate the weight matrix and apply a row-wise softmax:

$$P = \sigma(-W^\top) \in [0, 1]^{M_L \times K}, \quad \sum_{k=1}^K P_{ik} = 1 \quad \forall i.$$

Each row p_i can now be read as a categorical distribution over classes for prototype i . As such, for a row p_i we can compute its Shannon entropy

$$H(p_i) = - \sum_{k=1}^K p_{ik} \log p_{ik}, \quad (3)$$

and normalise it by the maximum possible entropy $\log K$ to obtain a confidence score

$$\tilde{H}(p_i) = 1 - \frac{H(p_i)}{\log K}, \quad \tilde{H}(p_i) \in [0, 1]. \quad (4)$$

$\tilde{H}(p_i) = 1$ when p_i is a one-hot vector (prototype decisively represents one class) and $\tilde{H}(p_i) = 0$ when p_i is uniform (prototype conveys no class information).

Finally, we average over all P prototypes:

$$\text{NNE}(W) = \frac{1}{P} \sum_{i=1}^P \tilde{H}(p_i). \quad (5)$$

$\text{NNE} \in [0, 1]$ is therefore *higher* for more interpretable weight matrices.

We validated the metric with two extreme cases:

Matrix	NNE
Identity \mathbf{I}_K (<i>perfectly sharp</i>)	1.000
Uniform $(\frac{1}{K}\mathbf{1})$ (<i>fully diffuse</i>)	0.000

It complements structural sparsity metrics such as global/local size and Hoyer-Square norms in ProtoS-ViT [21] and the “transparent-head complexity” of Shallow-ProtoPNet [22] by quantifying decisiveness even when the head is dense. Where neuron-level Class-Selectivity Indices normalize entropy over *activations* [23], NNE is the first to apply an entropy normalization directly to prototype voting weights, making it agnostic to input distribution and layer depth. Recent analyses of classifier weight spectra, enabled by bilinear multilayer perceptrons (MLPs) [24], underscore the value of interpreting the classification head as an information-theoretic channel. In this context, the NNE score provides a concise, task-size-invariant metric that aligns with and supports this perspective.

The proposed metric will be employed throughout this work to evaluate the quality of the classification weight matrix across all experiments, serving as a means to assess and compare the reliability and class specificity of the learned prototypes.

4.2. Experiment 1: Baseline Architecture

The objective of this experiment is twofold: to evaluate the performance of the proposed architecture on the MNIST dataset and to analyze the structure of the resulting prototype representations. As an initial step, we consider the canonical Deep Prototype-Based Network (DPBN), which closely follows the configuration introduced by [4], with minor modifications to facilitate subsequent ablation studies. The input pipeline preserves the original 28×28 luminance pixel values of the MNIST digits, thereby avoiding information loss during preprocessing.

Within the network, layer 1 comprises compact convolutional encoder-decoder module that projects each input image into a 40-dimensional latent vector and reconstructs it back into pixel space. The encoder employs a fixed stride pattern of (2,2,2,2) and a sequence of convolutional filters (32,32,32,10), progressively reducing the spatial resolution from 28×28 to 2×2 over four downsampling stages. This architectural design ensures that each spatial location in the final feature map corresponds to a roughly square receptive field in the input image, a property leveraged in later experiments to interpret prototype activations in a spatially meaningful manner.

The flattened feature map is subsequently projected into a latent representation of dimension $q_1 = 15$ via a fully connected layer. The choice of $q_1 < 40$ is deliberate, as it constrains the image

autoencoder to learn a non-trivial, low-dimensional manifold rather than approximating the identity mapping. This serves as an implicit regularization mechanism for the initial prototype search space. Within this space, $M_1 = 20$ prototypes are allocated, resulting in a prototype-to-latent ratio of $M_1/q_1 = 1.33$. Empirical observations from preliminary experiments indicated that this ratio is sufficient to support a near one-to-one correspondence between digit classes and prototypes, while avoiding the introduction of redundant representations.

The subsequent layers (layers 2 through 4) share an identical structural design, differing from the initial layer primarily in that their encoders are implemented as multilayer perceptrons (MLPs) operating on similarity vectors rather than raw image data. Specifically, each distance vector produced by layer l is projected into a latent space of dimension $q_{l+1} = 10$, then decoded back to its original dimensionality M_l , and subsequently passed to the next prototype layer of equal size. In each layer, the prototypes $p_j^{(l)}$ are initialized from a uniform distribution $\mathcal{U}(0, 1)$ and jointly optimized alongside the corresponding autoencoder weights. Importantly, we enforce the condition $M_l \geq q_l$ at every depth to ensure that the resulting distance matrices are overcomplete -a necessary property for the hypersphere-intersection solver used to reconstruct prototype representations across layers.

The coefficients we set to $\lambda_{R_1} = \lambda_{R_D} = \lambda_{R_1} = \lambda_{R_2} = 0.5$ based on a coarse log-grid search aimed at balancing classification accuracy with reconstruction fidelity. Model training was performed using the Adam optimizer with a fixed learning rate of $1e-4$. An early stopping criterion with a patience of 100 epochs was employed, terminating training when validation accuracy failed to improve over 100 consecutive epochs. On the MNIST dataset, the model converged to a test accuracy of 98.05% after approximately 850 epochs, thereby establishing a strong performance upper bound for the ablation studies that follow.

Interpretability, however, diminishes with increasing network depth. As illustrated in Figure 2, reconstructed prototypes from all four layers are visually compared. Prototypes in the first layer exhibit clearly defined glyphs, each distinctly associated with a specific digit class. This observation confirms that the autoencoder, in conjunction with the R_1/R_2 terms regularization terms, effectively aligns prototypes with the underlying data manifold. By the layer 2, the prototypes lose fine-grained details and become more abstract, although they still retain rough class-specific contours. In layers 3 and 4, interpretability degrades substantially: the reconstructed shapes converge into nearly indistinguishable forms, differentiated only by subtle intensity variations and with weakly class-informative interpretable patterns. Despite this loss in human interpretability, the softmax-normalized weight matrix yields a Normalized Negative Entropy (NNE) score of 0.26, indicating that the classifier maintains a coherent and class-specific internal representation, even as the prototypes themselves become less visually discernible. This observation is particularly significant, as it indicates that the final-layer prototypes retain sufficient information to support accurate classification while simultaneously integrating multiple visual patterns. This behavior is a consequence of the decreasing dimensionality across successive layers, which compels the network to fuse lower-level features. Accordingly, the process can be interpreted as a hierarchical compositional mechanism that preserves discriminative attributes across layers.

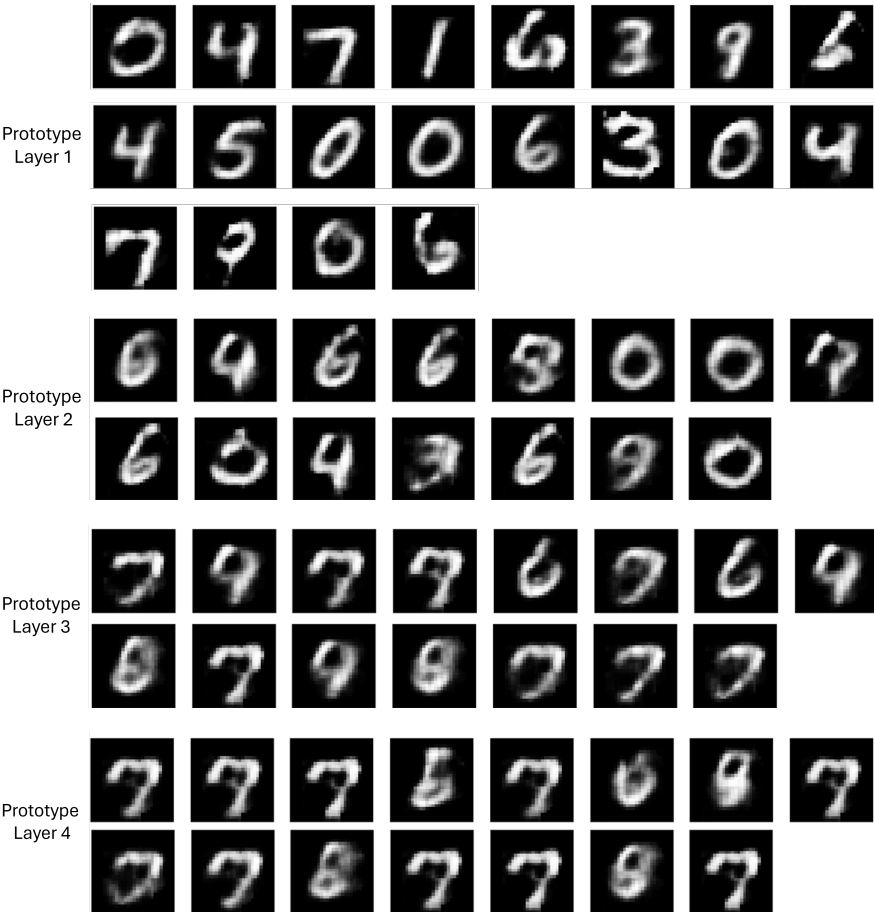


Figure 2. Reconstructions of prototypes reveal that the uppermost panel (layer 1) exhibits digit-specific structures, while deeper layers (from top to bottom) tend to converge toward patterns that are less interpretable by humans due to their combination.

4.3. Experiment 2: Model Distillation

In this experiment we investigate whether the prototypes learned by the deepest layer of a four-layer DPBN could be effectively transferred to a significantly shallower model without compromising either predictive accuracy or interpretability. Specifically, the prototype layer extracted from the fourth layer of the reference DPBN was copied into a single-layer DPBN, where it remained fixed during training. Only the autoencoder and the classification head were subsequently trained. Qualitative comparisons between the original layer-4 prototypes and those reconstructed by the distilled model, along with the associated weight matrices W_1 and W_2 , are shown in Figure 3 and Figure 4.

Despite the substantial reduction in network depth, the distilled model achieved an accuracy statistically indistinguishable from that of the baseline. Moreover, its weight matrix maintained coherence, with a NNE of 0.18. To assess whether this performance could be attributed to the information encoded in the transferred prototypes, we repeated the experiment using randomly initialized prototype vectors, which were likewise frozen during training. Contrary to our initial hypothesis, the network again attained comparable accuracy and produced sharply defined reconstructed prototypes. This suggests that the raw numerical values of the prototypes, in isolation, contribute little semantic content. Rather, interpretability appears to arise from the dynamics of the autoencoder and the spatial configuration of the prototypes within the latent space. Notably, the geometry of this space retains semantic structure: prototypes corresponding to visually similar digits, such as 8 and 9, are positioned closely, whereas those representing semantically distant digits, such as 2 and 9, are more widely separated. We postulate that this latent space organization underlies the richer structure observed in the weight matrix when training begins with learned or trainable prototypes as opposed to random ones. These findings motivated the follow-up experiment described in the subsequent section, in which the network was

trained end-to-end using randomly initialized, frozen prototypes. This design aimed to isolate the contribution of latent space geometry from the influence of prototype initialization.

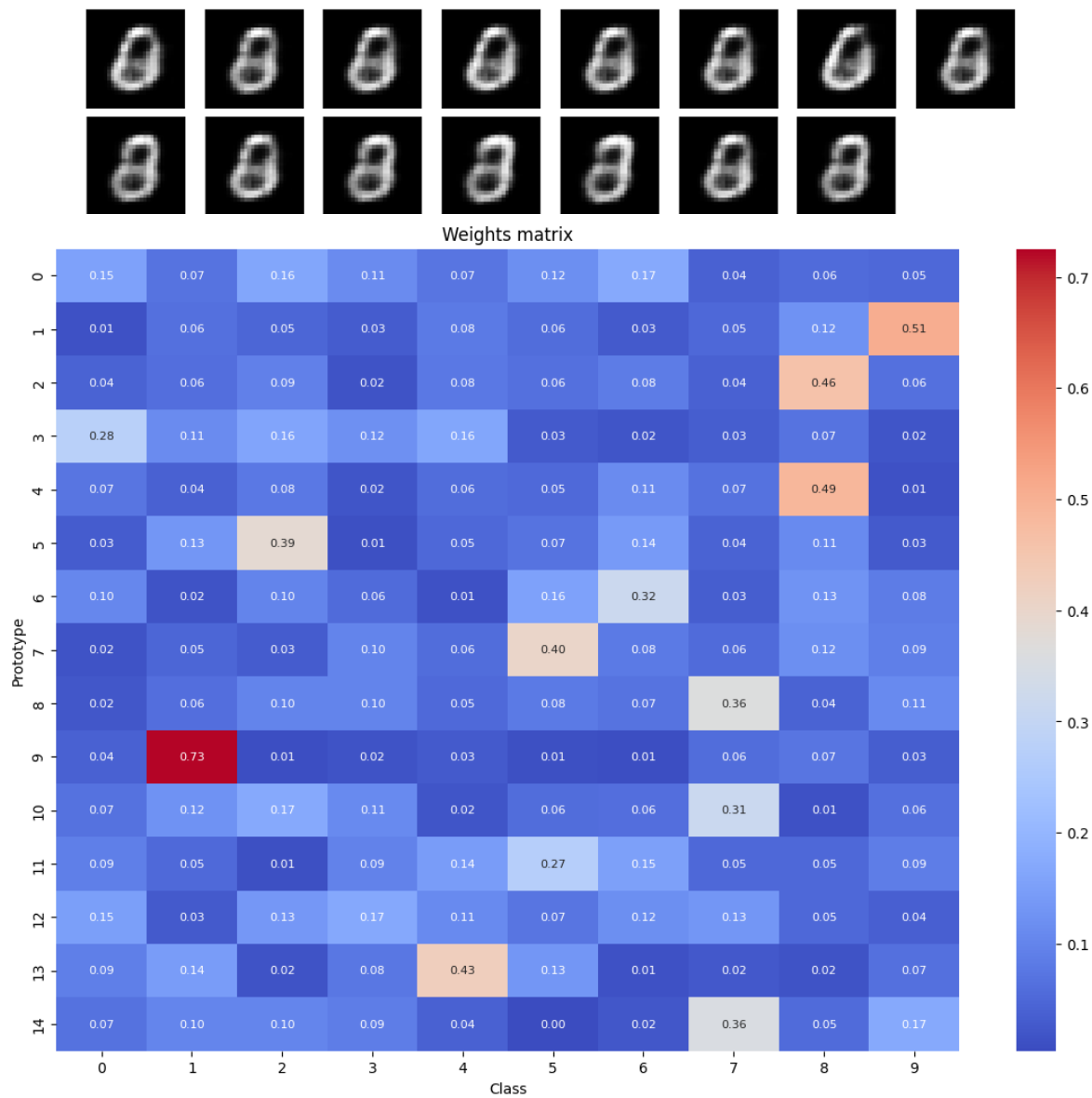


Figure 3. Prototypes derived from the final layer in Experiment 2, presented alongside the corresponding weight matrix.

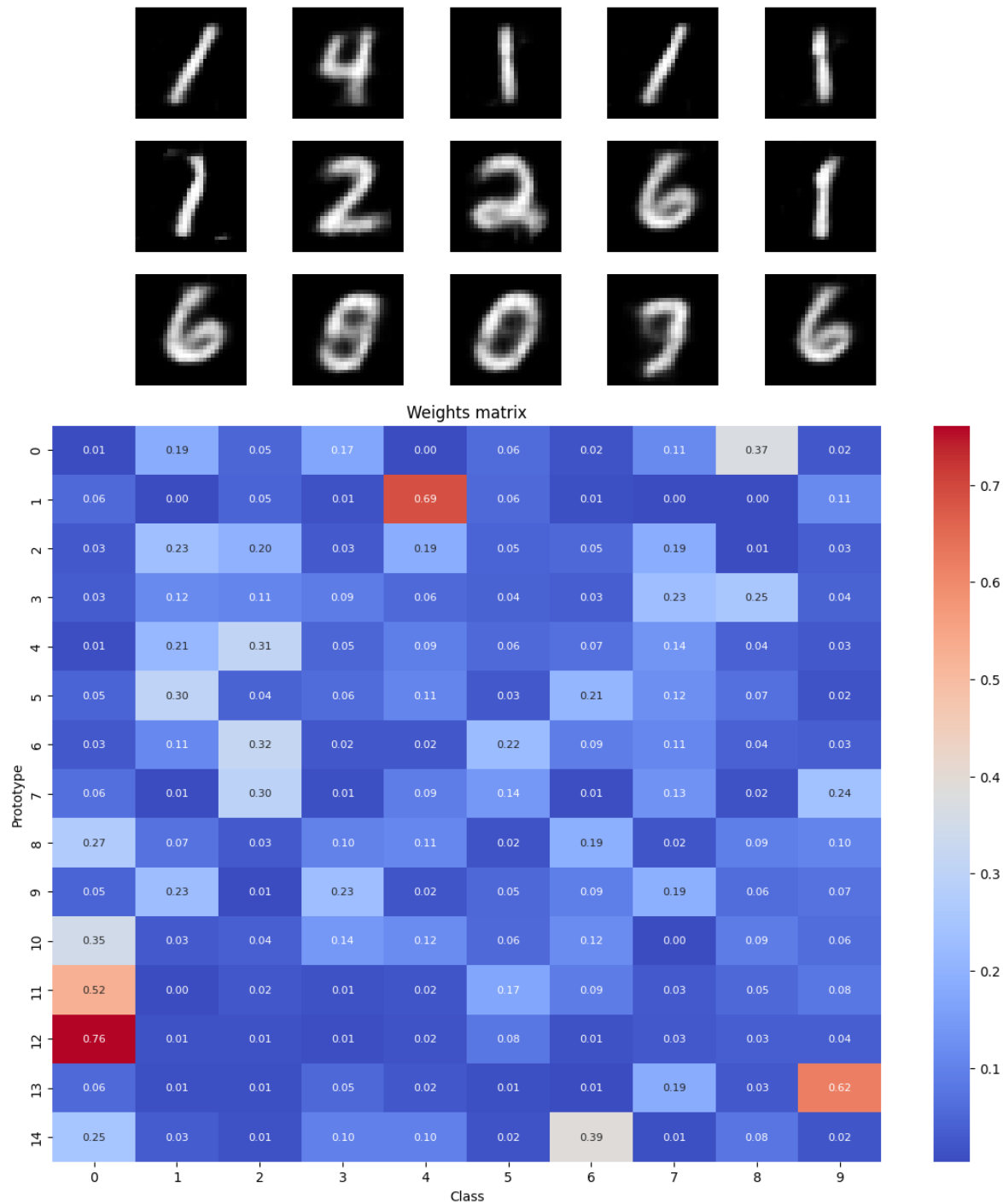


Figure 4. Prototypes obtained from those transferred from a 4-layer Deep Prototype-Based Network (DPBN) in Experiment 2, shown together with the corresponding weight matrix.

4.4. Experiment 3: Can a 1 Look Like a 0?

In this experiment we investigate whether the conclusions drawn in the previous multiclass setting generalize to the simpler binary task of distinguishing the digit “0” from all other handwritten digits. To construct the training corpus, all images corresponding to class “0” from the MNIST training split were retained, while samples from the remaining classes were randomly downsampled to ensure that the alternative class (denoted other) contained an equal number of instances. The resulting dataset was thus balanced, with 80% allocated for training, 10% for validation, and the remaining 10% for testing. All target labels were transformed into binary form using the indicator function $y \mapsto 1[y \neq 0]$.

We retained the deep prototype architecture of the previous experiments, but specified a four-layer prototype hierarchy $(15, 20) \rightarrow (10, 15) \rightarrow (10, 15) \rightarrow (10, 4)$, where each pair denotes the latent dimensionality, and the number of prototypes. The network was trained for 370 epochs with Adam (learning rate = 10^{-4}) and early stopping after 100 epochs without validation improvement. The highest validation accuracy achieved was 99.49%, with a final test accuracy of 99.48%. Despite this strong predictive performance, the interpretability metrics were less encouraging. Specifically, the softmax-normalized weight matrix of the final classification layer exhibited a normalized negative entropy (NNE) of 0.14, indicating a relatively flat association between prototypes and class labels. Furthermore, two of the four prototypes reconstructed from the deepest layer resembled noisy, amorphous “clouds” (see Figure 5) rather than well-defined digit patterns, and the weight matrix (see Figure 6) deviated substantially from a one-hot configuration.

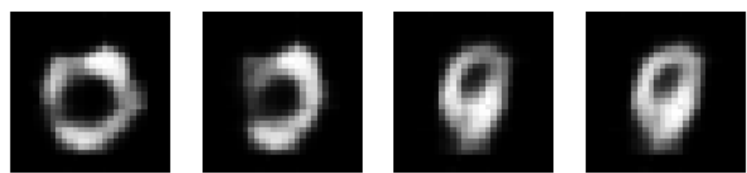


Figure 5. Reconstructed prototypes from the last layer showing a low degree of interpretability.

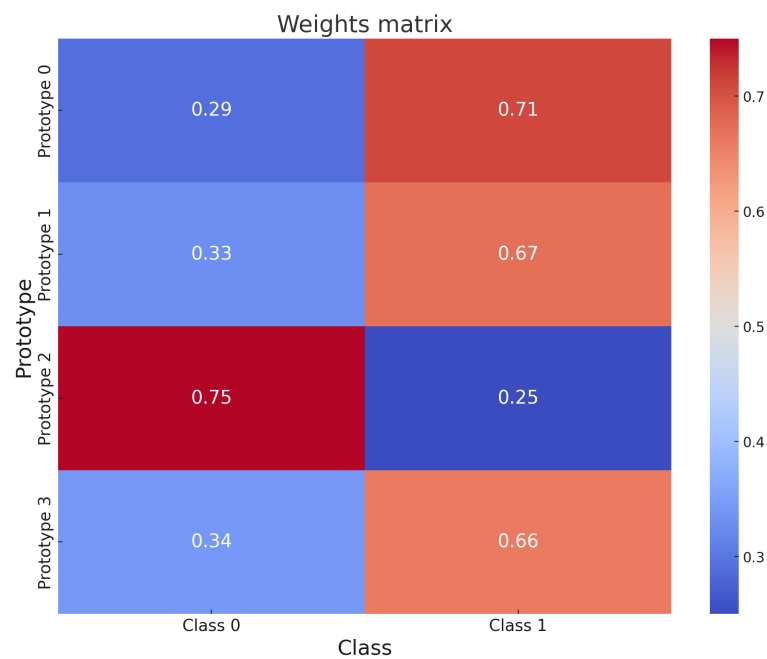


Figure 6. Softmax-normalised weight matrix of the deep prototype network. Rows correspond to prototypes, columns to the two classes. Prototype #4 (bottom row) is visually a digit “0” yet assigns the bulk of its weight to the *other* class.

To examine whether the latent geometry captured by the deep model could be more explicit, we distilled the last-layer prototypes into a shallow DPBN comprising a single layer, following the same procedure as in Experiment 2. The prototypes were held fixed during training, while only the autoencoder and classification weights were updated. Training converged after 210 epochs, achieving an accuracy of 99.83%, marginally surpassing that of the original deep model. These results support the hypothesis that a well-structured set of prototypes can facilitate more efficient representation learning, even when the prototypes themselves lack of clear visual interpretability.

A detailed examination of the distilled model revealed an noteworthy behavior. Prototype #4, whose reconstruction clearly depicts a digit “0” (see Figure 7), is associated by the weight matrix with the “others” class with a probability of 66 %. An analysis of the twelve test images with latent

encodings closest to this prototype (Figure 8) confirms that all correspond to well-formed instances of the digit “0”, each correctly classified as class “0”. Consequently, the prototype provides clear visual evidence of the digit “0” while functionally contributing to the decision for the opposing class. Because the mapping between prototypes and classes is learned independently of the visual characteristics of the prototypes, users lacking access to the weight matrix may be misled into believing that the model’s predictions for the “others” class are based on archetypal non-zero digits, when in fact the decisive signal is anchored in a visually canonical zero.

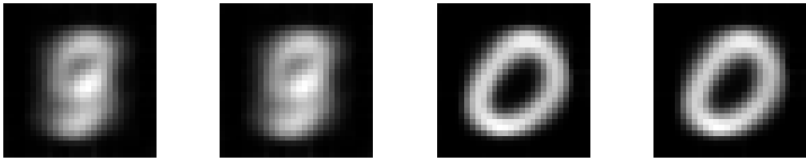


Figure 7. Reconstructed deepest-layer prototypes after training. The last prototype (right-most) is a clear “0” despite its functional association with the *other* class.

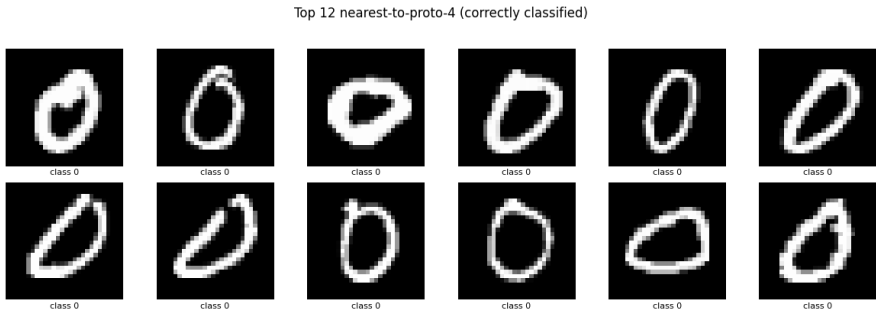


Figure 8. Twelve test images nearest to Prototype #4 in latent space, all correctly labelled as class “0”.

The binary classification setting thus amplifies a concern previously suggested by the multiclass experiments: prototype visualizations alone are insufficient to establish model trustworthiness. A prototype may function as a decoy, simultaneously presenting a compelling, human-interpretable exemplar while driving predictions for an unexpected class. Given that the distance vectors input to the linear classification head are high-dimensional, manual verification of class–prototype associations is impractical, and summary metrics such as normalized negative entropy (NNE) reflect only global sharpness rather than individual inconsistencies.

5. Conclusions

In this study, we introduced a novel prototype-based neural network architecture incorporating multiple prototype layers. We evaluated its performance and interpretability using the MNIST dataset, demonstrating both its representational capabilities and highlighting critical limitations concerning interpretability. Results demonstrate the network’s capacity to integrate and represent patterns across successive prototype layers. However, we observed a progressive decline in the visual interpretability of prototypes at deeper layers, complicating human understanding of the learned representations. Importantly, the decline in visual clarity at deeper prototype layers did not compromise the model’s predictive performance, which remained consistently high—indicating that the final prototype layer retained strong discriminative capacity despite reduced interpretability. A distilled shallow model, constructed by reusing the final prototype layer from the original deep prototype-based network (DPBN) as a fixed component, achieved a slight improvement in classification accuracy (99.83%). However, it exposed a critical flaw—prototypes visually representing “0” contributed internally to decisions for the “1” class. Hence, visualization of prototypes alone are insufficient for model transparency and reliable interpretability, and it is required to align visual and functional behaviors. Accordingly, we argue that prototype-based explanations within the “this-looks-like-that” paradigm necessitate additional safeguards—either through training constraints that enforce alignment between

visual and functional semantics or via post-hoc auditing tools capable of identifying contradictory associations. A comprehensive investigation into such reliability guarantees is reserved for future work.

Author Contributions: Conceptualization, E.G-C.; methodology, E.G-C. and D.M.; software, R.I.; validation, E.G-C., D.M., and R.I.; formal analysis, E.G-C. and D.M.; investigation, E.G-C and R.I. and D.M.; data curation, R.I.; writing—original draft preparation, E.G-C. and R.I.; writing—review and editing, E.G-C and D.M.; visualization, R.I.; supervision, E.G-C.; All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Conflicts of Interest: The authors declare no conflicts of interest.

References

1. Li, O.; Liu, H.; Chen, C.; Rudin, C. Deep Learning for Case-based Reasoning through Prototypes: A Neural Network that Explains its Predictions. *CoRR* **2017**, *abs/1710.04806*, [1710.04806].
2. Ras, G.; Xie, N.; Van Gerven, M.; Doran, D. Explainable deep learning: A field guide for the uninitiated. *Journal of Artificial Intelligence Research* **2022**, *73*, 329–396.
3. Deng, L. The mnist database of handwritten digit images for machine learning research. *IEEE Signal Processing Magazine* **2012**, *29*, 141–142.
4. Li, O.; Liu, H.; Chen, C.; Rudin, C. Deep Learning for Case-Based Reasoning through Prototypes: A Neural Network that Explains Its Predictions. In Proceedings of the Proceedings of the 32nd AAAI Conference on Artificial Intelligence, 2018. ArXiv preprint arXiv:1710.04806 (v2, Nov. 2017).
5. Chen, C.; Li, O.; Tao, D.; Barnett, A.; Rudin, C.; Su, J. This Looks Like That: Deep Learning for Interpretable Image Recognition. In Proceedings of the NeurIPS, 2019.
6. Nauta, M.; van Bree, R.; Seifert, C. Neural Prototype Trees for Interpretable Fine-Grained Image Recognition. In Proceedings of the CVPR, 2021.
7. Gulshad, S.; Long, T.; van Noord, N. Hierarchical Explanations for Video Action Recognition. In Proceedings of the 2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW), 2023, pp. 3703–3708. <https://doi.org/10.1109/CVPRW59228.2023.00379>.
8. Koh, P.W.; Nguyen, T.; Tang, Y.S.; Musmann, S.; Pierson, E.; Kim, B.; Liang, P. Concept Bottleneck Models. In Proceedings of the ICML, 2020.
9. Papernot, N.; McDaniel, P. Deep k-Nearest Neighbors: Towards Confident, Interpretable and Robust Deep Learning. In Proceedings of the ICML, 2018.
10. Sabour, S.; Frosst, N.; Hinton, G.E. Dynamic Routing Between Capsules. In Proceedings of the NeurIPS, 2017.
11. Zhang, N.; Donahue, J.; Girshick, R.; Darrell, T. Part-based R-CNNs for Fine-grained Category Detection. In Proceedings of the ECCV, 2014.
12. Bancroft, S. An Algebraic Solution of the GPS Equations. *IEEE Transactions on Aerospace and Electronic Systems* **1985**, *AES-21*, 56–59.
13. Abel, J.; Chaffee, J. Existence and Uniqueness Analysis for the GPS Equations. *IEEE Transactions on Aerospace and Electronic Systems* **1991**.
14. Norrdine, A. An Algebraic Solution to the Multilateration Problem. In Proceedings of the Proc. International Conference on Indoor Positioning and Indoor Navigation (IPIN), 2012, pp. 1–6.
15. Levenberg, K. A Method for the Solution of Certain Non-Linear Problems in Least Squares. *Quarterly of Applied Mathematics* **1944**, *2*, 164–168.
16. Marquardt, D.W. An Algorithm for Least-Squares Estimation of Nonlinear Parameters. *Journal of the Society for Industrial and Applied Mathematics* **1963**, *11*, 431–441.
17. Moore, E.H. On the Reciprocal of the General Algebraic Matrix. *Bulletin of the American Mathematical Society* **1920**, *26*, 394–395.
18. Penrose, R. A Generalized Inverse for Matrices. *Proceedings of the Cambridge Philosophical Society* **1955**, *51*, 406–413.
19. Bohle, M.; Singh, N.; Fritz, M.; Schiele, B. B-Cos Alignment for Inherently Interpretable CNNs and Vision Transformers. *IEEE transactions on pattern analysis and machine intelligence* **2024**, PP. <https://doi.org/10.1109/TPAMI.2024.3355155>.

20. Wolf, T.N.; Kavak, E.; Bongratz, F.; Wachinger, C. SIC: Similarity-Based Interpretable Image Classification with Neural Networks, 2025, [[arXiv:cs.CV/2501.17328](#)].
21. Turbé, H.; Bjelogrić, M.; Mengaldo, G.; Lovis, C. ProtoS-ViT: Visual foundation models for sparse self-explainable classifications, 2024, [[arXiv:cs.CV/2406.10025](#)].
22. Singh, G.; Frizzo Stefenon, S.; Yow, K.C. The shallowest transparent and interpretable deep neural network for image recognition. *Scientific Reports* **2025**, *15*, 13940. <https://doi.org/10.1038/s41598-025-92945-2>.
23. Leavitt, M.L.; Morcos, A. Selectivity considered harmful: evaluating the causal impact of class selectivity in DNNs, 2020, [[arXiv:cs.LG/2003.01262](#)].
24. Pearce, M.T.; Doms, T.; Rigg, A.; Oramas, J.M.; Sharkey, L. Bilinear MLPs enable weight-based mechanistic interpretability. *arXiv preprint arXiv:2410.08417* **2025**. Analyzes bilinear MLP weight-spectrum for mechanistic interpretability.

Disclaimer/Publisher's Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.