

The Polytopal Neural Network

A. Emilie J. Wedenborg
Technical University of Denmark

Anders V. Nørskov
Technical University of Denmark

Teresa Dorszewski
Technical University of Denmark

Kristoffer Wickstrøm
UiT The Arctic University of Norway

Morten Mørup
Technical University of Denmark

Abstract

Understanding how deep neural networks process information remains a central challenge. Existing interpretability methods often compromise structural fidelity, rely on prespecified corpora, or explain models post-hoc. We propose Polytopal Neural Networks (PNNs), a framework that extracts distinct layer-wise aspects by enforcing a polytope-based structure that is used directly in subsequent information processing. We scale our approach using learned corpus representations and an amortized simplex inference procedure and highlight how the framework also gives a direct route to vector quantized (VQ) training. In PNNs, observations are explicitly described by their alignment with layer-specific aspects. Empirical results show that imposing polytopal constraints on neural network representations preserves meaningful structures in the latent space with minimal degradation in performance, favorable compressed representations when compared to VQ representations in unsupervised learning, while also providing a performant new approach to VQ deep learning training. Our findings suggest that deep networks can enforce interpretable polytope-based representations, offering a principled path toward more transparent AI systems with minimal performance compromise.

1 INTRODUCTION

The increasing deployment of deep learning systems in scientific, industrial, and societal contexts has intensified the demand for methods that make their internal computations transparent and trustworthy [Longo

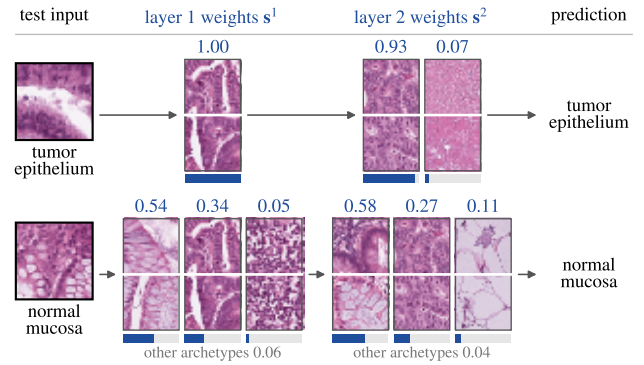


Figure 1: Two PathMNIST test images traced through two PNN layers ($K = 10$). Each layer passes a convex combination of archetypes to the next layer. Numbers denote mixture weights; each archetype is illustrated by its two nearest training images.

et al., 2024]. Yet, many existing approaches fail to explain how neural networks structure and transform information across layers. A substantial body of explainable AI (XAI) research has focused on post-hoc analyses of input-output behavior, including saliency-based attribution methods and activation visualizations [Bach et al., 2015, Selvaraju et al., 2017, Uendes et al., 2025]. Although such techniques can identify influential input regions, they offer limited insight into the internal mechanisms directing the representation learning within the network. This limitation is especially noticeable given consistent empirical evidence that deep networks develop structured, hierarchical representations, where early layers encode low-level attributes such as color and texture, while deeper layers assemble these primitives into higher-order concepts and object-level abstractions [Bau et al., 2017, Dorszewski et al., 2025, Vielhaben et al., 2024]. Despite their central role in learning and inference, these latent representations remain largely unconstrained and difficult to interpret, restricting their usefulness

for mechanistic understanding, model validation, and scientific discovery.

We introduce Polytopal Neural Networks (PNNs), a class of architectures in which latent representations are explicitly constrained to lie on a polytope by use of simplex constraints (i.e., imposing non-negativity and sum-to-one constraints). This induces a structured, interpretable latent geometry in which each representation is expressed as a convex combination of learned archetypes defining the corners of the polytope (Figure 1). Crucially, this constraint is enforced during training and shapes the network’s internal computations, rather than serving as a post-hoc explanation mechanism [Wedenborg et al., 2026]. PNNs are related to corpus-based explanation methods that express predictions as mixtures of exemplars [Crabbe et al., 2021]. However, these approaches rely on a fixed, user-defined corpus and aim to explain individual predictions after training. In contrast, PNNs learn the archetypal corpus end-to-end and enforce simplex constraints directly on the latent space, ensuring that predictions are intrinsically consistent with the interpretable polytope representations imposed at every layer.

We use interpretability to mean the extent to which a model’s internal representations and computations can be understood in terms of identifiable components and their contributions. An explanation is intrinsic when those components are part of the computation producing the output, rather than estimated afterward. It is local when it describes an individual input and global when it describes structure shared across inputs. PNNs provide intrinsic local explanations through each input’s mixture weights over archetypes, and global structure through the shared archetypes that define each layer’s polytope. This guarantees a faithful description of the constrained representations.

PNN’s use of archetypes as reference profiles is motivated by geometric accounts of cognition based on similarity and betweenness [Gärdenfors and Williams, 2001], and by evidence that contrastive categorization can favor extreme, idealized representatives along distinguishing dimensions [Davis and Love, 2010]. This framing connects the PNN to classical Archetypal Analysis (AA) [Cutler and Breiman, 1994, Mørup and Hansen, 2012], which models data as mixtures of extremal points lying on the convex hull. AA based models are well known for their interpretability, as both the archetypes and the mixture coefficients allow intuitive explanations. Integrating AA constraints into deep architectures, PNNs bridge the gap between interpretable, prototype-based modeling and high-capacity representation learning, ensuring the results remain strictly grounded in the original data.

The proposed framework also relates to vector quantization (VQ) methods, most notably vector quantized variational autoencoders (VQ-VAEs) [van den Oord et al., 2017], which replaces continuous latent manifolds with finite representations to promote compression, disentanglement, and efficient generative modeling. Both VQ-VAE and PNNs reduce latent complexity by constraining representations to a limited set of learned components. However, their mechanisms differ: VQ-VAE relies on hard nearest-neighbor assignments, exponential moving average (EMA) updates, and auxiliary optimization heuristics such as the straight-through estimator and commitment losses [van den Oord et al., 2017], whereas PNNs employ continuous convex combinations of archetypes, avoiding discrete assignment operations while preserving a compact and interpretable latent structure.

In summary, PNNs combine the strengths of Archetypal Analysis, Vector Quantization, and deep representation learning. Each layer decomposes activations into convex mixtures of learned archetypes, forming a sequence of sequential polytopes that jointly explain and reconstruct the model’s internal representations. This polytopal structure enforces geometric consistency and provides an intrinsic explanation of each prediction through archetypal coefficients.

Specifically, our main contributions are:

- (i) **Polytopal representation learning.** A layer-wise framework that constrains latent features to lie within learned polytopes, with AA constrained representations enforced exactly at every constrained layer.
- (ii) **Integrated interpretability.** The coordinates used to explain a representation are also the coordinates that the next layer receives.
- (iii) **Scalable inference.** A batched learning framework with (a) a compact corpus learned jointly with the network and updated from mini-batches and (b) amortized simplex inference.
- (iv) **A direct optimization route for VQ.** Replacing the simplex with one-hot coordinates reduces the PNN to a VQ model whose codebook is grounded in the corpus and is trained without EMA updates, a commitment loss, or a straight-through estimator.

2 RELATED WORK

Archetypal Analysis (AA), introduced by Cutler and Breiman [1994], has been incorporated into deep learning frameworks, including sparse autoencoder-based models [Fel et al., 2025a] and other deep architectures [Keller et al., 2019, 2021, Milite et al., 2025, Wieser

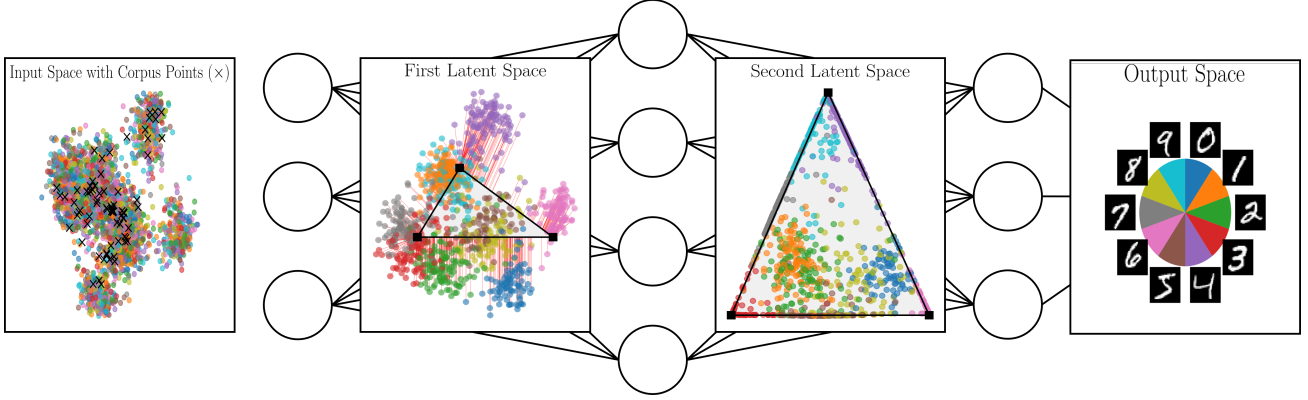


Figure 2: **The Polytopal Neural Network:** Points in the input space (left) are used to learn data driven corpus points ($\times$) used to span the polytopes in the layers of the PNN. Each layer maps the observations into a low-dimensional simplex-structured latent space (polytope), where convex combinations of the encoded corpus points define the layer specific archetypal vertices. A subsequent latent transformation refines the geometric organization forming a refined polytope. Finally, points are decoded into the output space (right), where the resulting representation exhibits improved separation and structure. Colored points indicate different data groups, and black vertices denote learned archetypes.

et al., 2025]. Most existing non-linear AA approaches place archetypes only at a single latent bottleneck of an autoencoder, and several enforce the archetypal structure only through a penalty [Keller et al., 2019, 2021, Wieser et al., 2025], so the archetypes act as a geometric prior rather than a strict constraint, limiting interpretability and consistency across layers.

Early work such as AAnet proposed learning nonlinear archetypal spaces using deep autoencoders that map data into a latent simplex, demonstrating improved recovery of extremal structure in nonlinear domains [van Dijk et al., 2019].

Recent research highlights AA as a tool for explainable representation learning, from stabilizing sparse autoencoders [Fel et al., 2025a] to identifying task-relevant concepts in vision transformers [Fel et al., 2025b]. Post hoc probing confirms that stable archetypes exist within pretrained networks [Wedenborg et al., 2026] but leaves the network unchanged. Deep AA architectures, in turn, place archetypes at a single bottleneck. There remains a need for architectures that enforce simplex-constrained representations throughout the network depth with explanations fully consistent with the networks’ information processing.

An alternative paradigm learns prototypical summaries jointly with the model. Prototype- and clustering-based approaches encourage representation space to organize around learned centroids or parts, facilitating concept-level explanations [Chen et al., 2024, Liang et al., 2023, Wang et al., 2023, Zhou and Wang, 2024]. ProtoPNet [Chen et al., 2019] assigns prototypes to classes, while PIP-Net [Nauta et al.,

2023] learns prototype–class associations through supervised classification after self-supervised pretraining. The PNN instead constrains intermediate representations to convex combinations of K archetypes without class assignments, using only the task loss. This constraint applies at multiple classifier depths, in unlabeled autoencoders, and to tokens from pretrained backbones. We therefore compare against the same unconstrained architecture and other representation bottlenecks: projection layers, vector quantization, and the Dirichlet VAE. Centroid-oriented representations emphasize typical instances and are effective for class- or part-centered interpretations. Notably, centroid-based representations are also explored in VQ-VAEs [van den Oord et al., 2017]. By contrast, AA targets extremal data geometry, decomposing space into vertices of a convex polytope and expressing each point as a convex mixture of these vertices [Cutler and Breiman, 1994, Hastie et al., 2009, Mørup and Hansen, 2012]. This shift from central tendency (centroids) to extremal representation (archetypes) yields complementary interpretive advantages where archetypes capture boundary phenomena and diverse modes of variation that centroids can obscure.

PNNs more closely resemble the Dirichlet VAE [Joo et al., 2020] in which the latent representation is projected onto the standard simplex with an associated Dirichlet prior imposed to characterize the continuum within the standard simplex. The PNN notably differs here by imposing a learned polytope as opposed to a fixed projection onto the standard simplex. The use of subspace projections of latent representations has also been widely adopted in deep learning using linear pro-

jection layers, see also [Hawkins-Hooker et al., 2018] and the recent generalization to arbitrary higher-order feature spaces [Morimoto and Huang, 2025]. Whereas these procedures can produce lower-dimensional subspaces they do not promote explainable polytopal latent representations.

3 METHODS

Notation. Data and representations are observations as rows throughout. $\mathbf{X} \in \mathbb{R}^{N \times M}$ holds N observations. At layer ℓ , $\mathbf{Z}^\ell \in \mathbb{R}^{N \times d_\ell}$ holds the representations before projection, $\mathbf{C}^\ell \in \mathbb{R}^{K \times N}$ is row-stochastic, and $\mathbf{S}^\ell \in \mathbb{R}^{N \times K}$ is row-stochastic. A row-stochastic matrix has non-negative entries and rows that sum to one. The archetypes are $\mathbf{A}^\ell = \mathbf{C}^\ell \mathbf{Z}^\ell \in \mathbb{R}^{K \times d_\ell}$ and the projected representations are $\mathbf{R}^\ell = \mathbf{S}^\ell \mathbf{A}^\ell$.

3.1 Archetypal Analysis and the Polytopal Neural Network

AA approximates the convex hull of the data by a polytope with K vertices, the archetypes, which are convex combinations of the data [Mørup and Hansen, 2012]. It solves

$$\begin{aligned} \min_{\mathbf{C}, \mathbf{S}} \|\mathbf{X} - \mathbf{S}\mathbf{C}\mathbf{X}\|_F^2 \\ \text{s.t. } \mathbf{c}_k \geq 0, \mathbf{c}_k \mathbf{1} = 1 \forall k, \mathbf{s}_n \geq 0, \mathbf{s}_n \mathbf{1} = 1 \forall n, \end{aligned} \quad (1)$$

where $\mathbf{c}_k$ and $\mathbf{s}_n$ are the k th row of $\mathbf{C}$ and the n th row of $\mathbf{S}$. The rows of $\mathbf{C}\mathbf{X}$ are the archetypes and $\mathbf{S}$ projects each observation onto the polytope they span. Archetypes are therefore idealized, data-grounded extremes, and each observation has a transparent decomposition in terms of these extremes.

A deep network applies L layers $f_{\theta^\ell}^\ell$ and is trained by minimizing $\mathcal{L}(\mathbf{Y}, f_\theta(\mathbf{X}))$ with $f_\theta = f_{\theta^L}^L \circ \dots \circ f_{\theta^1}^1$. Supervised learning uses labels as $\mathbf{Y}$, and an autoencoder uses the inputs themselves. The PNN inserts a projection operator $\Delta_{\mathbf{C}^\ell}^\ell$ after the chosen layers,

$$\begin{aligned} f_\theta(\mathbf{X}) = f_{\theta^L}^L \circ [\Delta_{\mathbf{C}^{L-1}}^{L-1} \circ f_{\theta^{L-1}}^{L-1}] \\ \circ \dots \circ [\Delta_{\mathbf{C}^1}^1 \circ f_{\theta^1}^1](\mathbf{X}), \end{aligned} \quad (2)$$

and learns $\{\theta^\ell, \mathbf{C}^\ell\}$ subject to $\mathbf{C}^\ell$ being row-stochastic. With $\mathbf{Z}^\ell$ the output of layer ℓ before projection, $\Delta_{\mathbf{C}^\ell}^\ell(\mathbf{Z}^\ell) = \mathbf{S}^\ell \mathbf{C}^\ell \mathbf{Z}^\ell$, where

$$\mathbf{S}^\ell = \arg \min_{\mathbf{S}} \|\mathbf{Z}^\ell - \mathbf{S}\mathbf{C}^\ell \mathbf{Z}^\ell\|_F^2 \quad \text{s.t. } \mathbf{s}_n \geq 0, \mathbf{s}_n \mathbf{1} = 1 \forall n. \quad (3)$$

This is a convex quadratic program (QP) per sample. We parameterize $\mathbf{C}^\ell = \text{softmax}(\tilde{\mathbf{C}}^\ell)$ row-wise with unconstrained logits $\tilde{\mathbf{C}}^\ell$ and train all parameters by back-propagation, treating $\mathbf{S}^\ell$ as fixed when

differentiating with respect to $\mathbf{C}^\ell$ (Appendix B.1). An exact solve by an active-set method costs up to $\mathcal{O}(K^3)$ per sample. Sequential minimal optimization (SMO) [Wedenborg and Mørup, 2025] reduces a full pass to $\mathcal{O}(K^2)$ by updating two coordinates at a time in closed form. Storing $\mathbf{X}$, every $\mathbf{Z}^\ell$, and $\mathbf{C}^\ell$ still costs $\mathcal{O}(NM + N \sum_\ell d_\ell + NLK)$ in memory.

3.2 Scaling the PNN

Scaling Memory Usage by Batched Streaming Updates of Data-Driven Corpus: We decouple the archetypes from the full dataset by learning a corpus of $N^c \ll N$ points, $\mathbf{X}^c = \mathbf{C}^c \mathbf{X}$ with row-stochastic $\mathbf{C}^c \in \mathbb{R}^{N^c \times N}$. The corpus is encoded by the same layers as the data, $\mathbf{Z}^{c,\ell}$, and defines the archetypes $\mathbf{A}^\ell = \mathbf{C}^\ell \mathbf{Z}^{c,\ell}$ with $\mathbf{C}^\ell \in \mathbb{R}^{K \times N^c}$. Let $\tilde{c}_{ki}$ be the logits of corpus row k . For numerical stability we subtract the row maximum of all N observations, $w_{ki} = \exp(\tilde{c}_{ki} - \max_j \tilde{c}_{kj})$, which leaves the softmax unchanged. For a mini-batch B and its complement $\neg B$, corpus row k is

$$\mathbf{x}_k^c = \frac{\sum_{i \in B} w_{ki} \mathbf{x}_i + \sum_{i \in \neg B} w_{ki} \mathbf{x}_i}{\sum_{i \in B} w_{ki} + \sum_{i \in \neg B} w_{ki}}. \quad (4)$$

The batch sums carry gradients. The non-batch sums are computed from the current logits without gradients, in chunks. A row outside the batch therefore receives no gradient on that step, but any update it received while in an earlier batch is reflected the next time the statistics are recomputed. Eq. (4) is the exact weighted average at every step, and gradient memory scales with $|B|$ and N^c rather than with N resulting in a memory usage of $\mathcal{O}(N^c |B| + N^c M + |B| M + N^c \sum_\ell d_\ell + N^c LK)$.

Amortized simplex inference. Although exact inference solves Eq. (3) for every sample in every forward pass at cost of $\mathcal{O}(K^3)$ reduced by SMO to $\mathcal{O}(K^2)$ this can still be prohibitive. We therefore instead learn a map g_ϕ^ℓ on the archetype correlations $\mathbf{h}_n = \mathbf{z}_n (\mathbf{A}^\ell)^\top \in \mathbb{R}^K$,

$$\hat{\mathbf{s}}_n = \text{softmax}(g_\phi^\ell(\mathbf{h}_n)), \quad \hat{\mathbf{r}}_n = \hat{\mathbf{s}}_n \mathbf{A}^\ell. \quad (5)$$

The input of g_ϕ^ℓ is a layer normalization of $\mathbf{h}_n$ together with $\log \text{std}(\mathbf{h}_n)$, because the projection depends on the scale of $\mathbf{h}_n$, which layer normalization removes. The task loss is computed through $\hat{\mathbf{r}}_n$, but g_ϕ^ℓ is never trained on it. It is trained only on the projection residual $\|\mathbf{z}_n - \hat{\mathbf{s}}_n \mathbf{A}^\ell\|^2$ with $\mathbf{z}_n$ and $\mathbf{A}^\ell$ detached, so it cannot deform the encoder, the head, the corpus, or the archetypes. It is fitted before the archetypes are trained and refit every ten epochs as the polytope moves. At test time the amortized coordinates can be refined by K rounds of batched SMO warm-started at $\hat{\mathbf{s}}_n$ (Appendix B.1).

3.3 Theoretical Properties

For a fixed polytope, let $A \in \mathbb{R}^{K \times d}$ hold the archetypes as columns, $G = AA^\top$, $h = Az$, and $\Delta^{K-1} = \{s \in \mathbb{R}^K \mid s \geq 0, \mathbf{1}^\top s = 1\}$. Expanding $\frac{1}{2}\|z - A^\top s\|^2$ and dropping the term independent of s gives

$$\Phi(h) = \arg \min_{s \in \Delta^{K-1}} \frac{1}{2} s^\top G s - h^\top s. \quad (6)$$

Lemma 1. *If G is positive definite, the minimizer is unique and Φ is a continuous piecewise affine map from $\mathbb{R}^K$ to Δ^{K-1} .*

G is positive definite if and only if A has full row rank, which requires $K \leq d$.

Theorem 1. *If G is positive definite, there is a finite ReLU network f_θ such that $\Phi(h) = \text{sparsemax}(f_\theta(h))$ for all $h \in \mathbb{R}^K$.*

Proofs are in Appendix A. The theorem establishes exact representability for a fixed polytope with sparsemax; we use softmax (Table 11). For CNNs and AEs, we refit the amortizer every ten epochs and apply K SMO rounds per forward pass. Appendix B.2 evaluates projection fidelity.

3.4 Token-level PNN.

For pretrained vision backbones we place the polytopal layer on the spatial tokens of the last feature map. Each token $\mathbf{z}_{n,t}$, $t = 1, \dots, T$, of image n is layer-normalized and linearly projected to 256 dimensions, and all tokens share one polytope $\mathbf{A}$. The amortizer assigns each token its own coordinates $\mathbf{s}_{n,t} \in \Delta^{K-1}$, and a linear classifier acts on the token mean of the projections,

$$\mathbf{y}_n = \left(\frac{1}{T} \sum_t \mathbf{s}_{n,t} \mathbf{A} \right) \mathbf{W}^\top + \mathbf{b}. \quad (7)$$

The pooled representation therefore stays in the polytope, with coordinates $\bar{\mathbf{s}}_n = \frac{1}{T} \sum_t \mathbf{s}_{n,t}$. Because the head is affine, the logits equal the mean of the per-token logits $\mathbf{s}_{n,t} \mathbf{A} \mathbf{W}^\top + \mathbf{b}$, so every token makes an exact, additive contribution to the prediction. Correspondingly, the corpus consists of individual (image, token) feature rows.

3.5 Generalizations of PNN

Interestingly, the developed PNN methodology naturally reduces to alternative well-known modeling procedures by changing the constraints used in (3).

Vector Quantization (VQ): Whereas PNNs approximate latent representations via convex combinations of learned archetypes, VQ-VAEs use a finite set of learnable embedding vectors. Unlike PNNs, where

archetypes are learned via gradient-based optimization, the VQ-VAE codebook vectors do not receive gradients directly from the loss. To enable backpropagation through the non-differentiable quantization step, the Straight-Through Estimator (STE) [Bengio et al., 2013] is employed, passing gradients from the codebook reconstruction $\mathbf{R}^\ell$ to $\mathbf{Z}^\ell$ unchanged. The resulting training objective consists of the task loss and a commitment loss,

$$\mathcal{L} = \mathcal{L}_{\text{task}} + \beta \sum_\ell \|\mathbf{Z}^\ell - \text{sg}[\mathbf{R}^\ell]\|_F^2, \quad (8)$$

where $\text{sg}[\cdot]$ is the stop gradient and the commitment term encourages encoder outputs to remain close to their assigned codebook entries and limits latent volume [van den Oord et al., 2017], see also Appendix G.3.1 for further details. Interestingly, the PNN framework provides a direct gradient-based optimization procedure for VQ deep learning by defining $\mathbf{R}^\ell$ as the solution to

$$\begin{aligned} & \underset{\mathbf{S}^\ell}{\text{minimize}} \quad \|\mathbf{Z}^\ell - \mathbf{S}^\ell \mathbf{C}^\ell \mathbf{Z}^{c,\ell}\|_F^2 \\ & \text{subject to} \quad \mathbf{s}_n^\ell \in \{0, 1\}^K, \quad \mathbf{s}_n^\ell \mathbf{1} = 1, \quad \forall n. \end{aligned} \quad (9)$$

$\mathbf{s}_n$ is a one-hot vector indicating centroid assignment. Notably, the optimal $\mathbf{C}$ used to define the cluster centroids by minimizing the least squares error is accordingly given by $\mathbf{C} = (\mathbf{S}^{\ell\top} \mathbf{S}^\ell)^{-1} \mathbf{S}^{\ell\top}$, provided every centroid is assigned at least one corpus point. Empirically we found this to always be true, but standard VQ-VAE measures to prevent codebook collapse are also applicable to the PNN-VQ. The equation corresponds to defining the centroids as the average of the observations of the learned corpus $\mathbf{Z}^{c,\ell}$ assigned to the centroid thus satisfying the non-negativity and sum to one constraints making this optimization consistent with the PNN formulation. Consequently, this approach can be directly implemented using the existing constraints on $\mathbf{C}^\ell$ based on the softmax reparameterization, whereas $\mathbf{s}_n$, instead of solved iteratively, can be estimated directly with one non-zero element p by

$$(\mathbf{s}_n^\ell)_p = 1 \text{ where } p = \underset{k}{\text{argmin}} \|\mathbf{z}_n^\ell - \mathbf{c}_k \mathbf{Z}^{c,\ell}\|_2^2. \quad (10)$$

Thereby, the PNN provides a simple gradient-based training objective without the need to impose a commitment loss. The encoder in the PNN-VQ learns through the corpus features $\mathbf{Z}^{c,\ell}$, and the layers after the projection learn through $\mathbf{R}^\ell$.

Projection Layers (PL): The use of Projection Layers (PL) in which each layer’s activation is projected onto a subspace before further processing is well established within deep learning to perform dimensionality reduction and compression, see also Hawkins-Hooker et al. [2018] and the recent generalization to

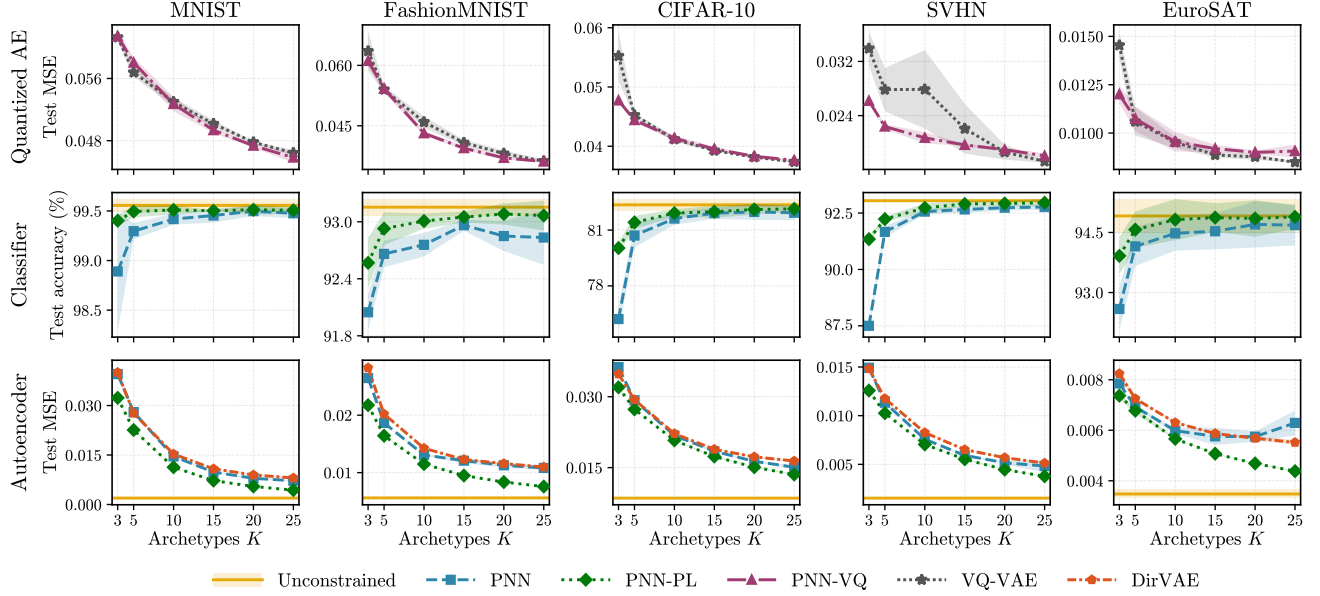


Figure 3: Performance as a function of the number of components across datasets. Shaded regions indicate $\pm$ one standard deviation over runs. **Top row:** VQ reconstruction MSE, with improved reconstruction quality reflected by lower values. **Middle row:** Classification accuracy. **Bottom row:** Autoencoder reconstruction MSE.

arbitrary higher order feature spaces Morimoto and Huang [2025]. Such projection layers can naturally be implemented within the PNN framework by imposing unconstrained representations

$$\begin{aligned} & \text{minimize} \quad \|\mathbf{Z}^\ell - \mathbf{S}^\ell \mathbf{C}^\ell \mathbf{Z}^{c,\ell}\|_F^2 \\ & \text{subject to} \quad \mathbf{s}_n^\ell \in \mathbb{R}^K \quad \forall n. \end{aligned} \quad (11)$$

Consequently, $\mathbf{s}_n$ is trivially given as the closed-form solution to the normal equation

$$\mathbf{s}_n^\ell = \mathbf{z}_n^\ell \mathbf{Z}^{c,\ell\top} \mathbf{C}^{\ell\top} (\mathbf{C}^\ell \mathbf{Z}^{c,\ell} \mathbf{Z}^{c,\ell\top} \mathbf{C}^{\ell\top})^{-1}. \quad (12)$$

whereas $\mathbf{C}^\ell$ is optimized unconstrained without the softmax mapping. We will denote the above generalizations PNN-VQ and PNN-PL, respectively.

4 RESULTS AND DISCUSSION

Experimental setup. We evaluate PNNs on image classification and reconstruction, with additional experiments on tabular classification. The main image benchmarks are MNIST [LeCun et al., 2010], FashionMNIST [Xiao et al., 2017], CIFAR-10 [Krizhevsky, 2009], SVHN [Netzer et al., 2011], and EuroSAT [Helber et al., 2019]. These experiments use a CNN classifier with polytopal layers after its two fully connected hidden layers and a convolutional autoencoder (AE) with a polytopal bottleneck. We further evaluate the PNN-CNN on PathMNIST, DermaMNIST, and BloodMNIST from MedMNIST v2 [Yang et al.,

2023]. For larger backbones and token representations, we evaluate ImageNet-100 [Tian et al., 2020] and ImageNet-1k [Deng et al., 2009] using ConvNeXt-T [Liu et al., 2022], initialized from ImageNet-1k weights and fine-tuned on each benchmark. The PNN projects each spatial token onto a shared polytope before averaging the resulting representations and applying a linear classifier. We also evaluate a PNN head on the pooled features of a frozen compact vision transformer [Hassani et al., 2022] on CIFAR-10. Additional MLP results on Digits [Alpaydin and Kaynak, 1998], Iris [Fisher, 1936], Wine [Aeberhard and Forina, 1992], and Breast Cancer [Wolberg et al., 1993] are reported in Appendix G.1.

The main CNN and AE experiments compare PNN with the corresponding unconstrained networks and the PNN-VQ and PNN-PL variants, enabling comparison between polytopal, subspace-based, and codebook-based representations. The AE comparisons additionally include VQ-VAE [van den Oord et al., 2017] and a variational autoencoder with a Dirichlet bottleneck [Joo et al., 2020]. The MedMNIST and larger-backbone experiments compare PNNs with their corresponding unconstrained models. Dataset splits, architectures, training procedures, and experiment-specific settings are provided in Appendix E.

We first evaluate the proposed scalable inference framework by comparing mini-batched training with full-data (non-batched) updates. Mini-batching with

batch-wise corpus updates (Section 3.2) achieves accuracy comparable to full-data training (see Appendix Table 7). We also evaluate the proposed amortized simplex inference procedure and find that with periodic realignment this serves as a fast and accurate replacement for the full QP projection (Appendix B.2).

To validate our direct training strategy for PNN-VQ, we compare it to standard VQ training with a commitment loss [van den Oord et al., 2017]. As shown in Figure 3, PNN-VQ achieves comparable, and in several cases lower test mean-squared-error (MSE), with detailed results reported in Table 12.

We next evaluate classification with a CNN backbone. In Figure 3, the PNN generally approaches the accuracy of the unconstrained CNN as the number of archetypes increases. The encoder dimension remains fixed, while larger values of K allow the constrained representation to use more archetypes. The largest improvements occur when moving away from the smallest archetype counts, although the remaining gap to the unconstrained model varies across datasets.

This comparison extends to the three MedMNIST datasets. With $K = 25$, the PNN achieves test accuracies of 89.1% on PathMNIST, 77.4% on DermaMNIST, and 98.1% on BloodMNIST, compared with 90.4%, 78.6%, and 98.3% for the corresponding unconstrained CNNs (Table 9). Increasing K from 10 to 25 improves the mean accuracy on PathMNIST and DermaMNIST, while BloodMNIST remains at 98.1%.

We evaluate the proposed method in an unsupervised setting using an AE. Reconstruction quality is measured using MSE, with a single PNN layer structuring the bottleneck under the PNN constraints. As shown in Figure 3, the PNN-AE shows similar tendencies as the PNN-CNN and generally performs competitively to the less constrained variants. PNN-AE generally achieves lower reconstruction MSE than DirVAE. The unconstrained AE and PNN-PL generally achieve lower MSE than PNN-AE, indicating a reconstruction cost associated with the convex constraint that generally shrinks with K . We also compare softmax and sparsemax as the amortizer output map (Table 11). Across MNIST, FashionMNIST, and CIFAR-10, the two variants achieve similar reconstruction MSE under the same refinement protocol.

Token-level PNNs on pretrained ConvNeXt-T retain most of the unconstrained model’s accuracy (Table 1). On ImageNet-100, accuracy varies little for $K \geq 100$, with overlapping error bars. Full results are in Appendix F.

Next, the PNN module is added to a frozen compact transformer where PNN acts on the pooled represen-

Table 1: Larger backbones and label spaces. Accuracy (%) of the unconstrained model and of the PNN, and the share of unconstrained accuracy the PNN retains. IN-100 and IN-1k are ImageNet-100 and ImageNet-1k, whose backbones are fine-tuned. Full results in Appendix F.

Setting	Classes	Unconstr.	K	PNN	Kept (%)
IN-100, ConvNeXt-T	100	94.6 $\pm$ 0.1	25	91.7 $\pm$ 0.5	96.9
IN-1k, ConvNeXt-T	1000	81.2 $\pm$ 0.0	128	78.6 $\pm$ 0.2	96.9
CIFAR-10, compact ViT	10	94.47 $\pm$ 0.21	10	94.46 $\pm$ 0.23	100.0

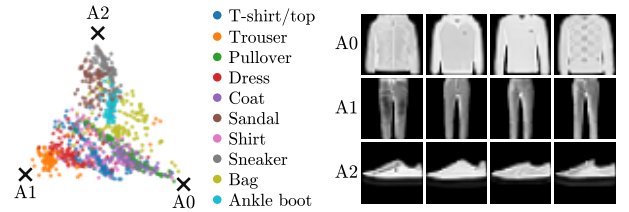


Figure 4: PNN-AE ($K = 3$) on FashionMNIST. Left: latent space projected via MDS; crosses indicate archetypes. Right: the four training images with the largest weight on each archetype, showing coherent groups (upper body, lower body, footwear).

tation. On CIFAR-10, its PNN head achieves 94.46% test accuracy with $K = 10$, compared to 94.47% for the unconstrained classifier (Table 1).

Figure 4 shows the latent space of a PNN-AE with $K = 3$ on FashionMNIST using multidimensional scaling, together with the four training images with the largest weight on each archetype. The displayed examples form recognizable groups of lower-body clothing, upper-body clothing, and footwear. This gives a qualitative illustration of how the learned polytope organizes the representations in this model.

The token-based ImageNet models allow the same archetypal representation to be inspected spatially. Figure 5 illustrates how individual tokens are represented using a shared polytope before their reconstructed features are averaged for classification. Figure 6 shows token predictions and the spatial weights of selected archetypes for individual images. The associated training examples provide visual references for interpreting these archetypes.

Each token is represented by its coordinates inside the polytope. The image-level margin can therefore be described as a sum of archetype scores, each weighted by the archetype’s share of the image (Figure 6). The snake example is described by two archetypes. Most of the tokens fall near an archetype whose training images show a small snake within a scene. The head tokens are primarily described by an archetype whose

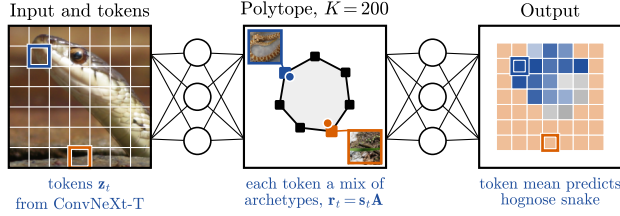


Figure 5: Token-level PNN on ImageNet-100 with ConvNeXt-T and 200 archetypes. Each spatial token is represented as a convex combination of shared archetypes. A linear classifier reads the mean of the reconstructed token features. Two archetypes associated with the predicted and competing classes are illustrated using training images. The outlined tokens have the largest weights on the displayed archetypes.

training images shows the hognose snake scale pattern. From the token votes we see that the head tokens have higher weight which influences the predictions. Unlike prototype networks [Chen et al., 2019, Nauta et al., 2023], which score patches by similarity to learned prototypes, the PNN represents each token as a point in the polytope spanned by the archetypes, so the decomposition uses the token’s own coordinates and requires no prototype-specific losses.

From a computational perspective, PNNs add per-iteration cost through corpus-based archetype construction and simplex inference. This cost depends on the batch size, representation dimension, corpus size, number of archetypes, and number of tokens. The inference procedure differs across experiments. The CNN and AE use the amortized simplex inference procedure followed by K SMO refinement rounds during training and evaluation. The ImageNet models use the amortizer without this refinement in the task path. The larger-scale experiments demonstrate that the construction can be used with spatial token representations, while its computational overhead depends on the chosen inference procedure. Costs are further examined in Appendix E.7. From an explainability perspective, PNNs expose the archetypal decomposition of a representation used in the forward computation. The FashionMNIST and ImageNet examples show how this decomposition can be inspected at the level of an entire input or individual spatial tokens. The archetypes are learned without predefined concept annotations, and representative training examples provide a way to examine their visual meaning. For the ImageNet model’s linear classifier, the archetype contributions can also be traced directly to the class scores. With nonlinear downstream networks, the coordinates describe the intermediate representation, and their effect on the output must be assessed through that subsequent computation. We test this by

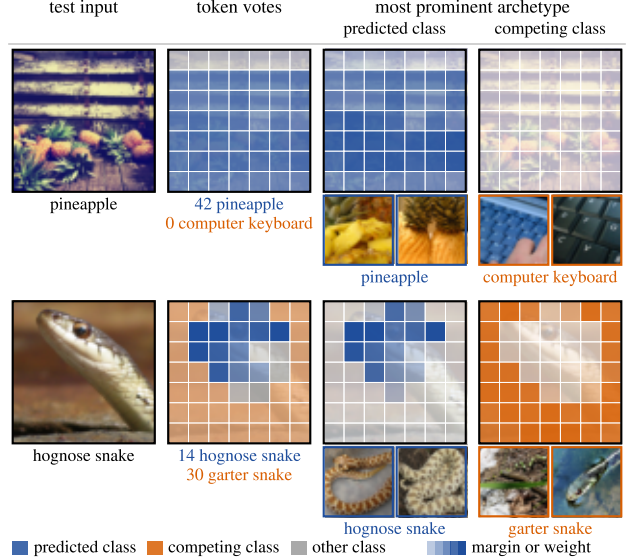


Figure 6: Token predictions and archetype weights for two ImageNet-100 images. Each row shows the input, token predictions, and spatial weights of selected archetypes associated with the predicted and competing classes. Training examples are used to illustrate the archetypes. Blue denotes the predicted class, orange the competing class, and gray other classes. The counts below the token-prediction panels show how many tokens favor each displayed class. The image prediction depends on the averaged class scores.

removing the archetype with the largest simplex coordinate, renormalizing the remaining coordinates, and compared this with removing a random archetype. For the CNN, this changes more predictions than removing a random archetype (Figures 8 and 9).

5 CONCLUSION

We have introduced the Polytopal Neural Network, a novel architecture that prioritizes interpretability and trustworthiness in deep learning by rigorously enforcing AA constrained latent representations. Through extensive evaluation across multiple network architectures and diverse datasets, we have demonstrated that PNNs produce inherently interpretable predictions with semantically meaningful explanations. Unlike post-hoc explanation methods, PNNs provide a principled guarantee that the explanations reflect the actual decision-making process learned during training, addressing a fundamental challenge in XAI. Our analysis reveals that PNNs can be understood as a continuous quantization method, establishing a natural connection to vector quantization approaches. In this framework, the learned archetypes form a semantically meaningful codebook that enables both effective

classification and transparent reasoning.

References

- Stefan Aeberhard and M. Forina. Wine. UCI Machine Learning Repository, 1992. DOI: 10.24432/C5PC7J.
- E. Alpaydin and C. Kaynak. Optical recognition of handwritten digits. UCI Machine Learning Repository, 1998. DOI: 10.24432/C50P49.
- Raman Arora, Amitabh Basu, Poorya Mianjy, and Anirbit Mukherjee. Understanding deep neural networks with rectified linear units. *CoRR*, abs/1611.01491, 2016. URL <http://arxiv.org/abs/1611.01491>.
- Sebastian Bach, Alexander Binder, Grégoire Montavon, Frederick Klauschen, Klaus-Robert Müller, and Wojciech Samek. On pixel-wise explanations for non-linear classifier decisions by layer-wise relevance propagation. *PloS one*, 10(7):e0130140, 2015.
- David Bau, Bolei Zhou, Aditya Khosla, Aude Oliva, and Antonio Torralba. Network dissection: Quantifying interpretability of deep visual representations. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 6541–6549, 2017.
- Yoshua Bengio, Nicholas Léonard, and Aaron Courville. Estimating or propagating gradients through stochastic neurons for conditional computation. *arXiv preprint arXiv:1308.3432*, 2013.
- Chaofan Chen, Oscar Li, Daniel Tao, Alina Barnett, Cynthia Rudin, and Jonathan K Su. This looks like that: deep learning for interpretable image recognition. *Advances in neural information processing systems*, 32, 2019.
- Guikun Chen, Xia Li, Yi Yang, and Wenguan Wang. Neural clustering based visual representation learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5714–5725, 2024.
- Jonathan Crabbe, Zhaozhi Qian, Fergus Imrie, and Mihaela van der Schaar. Explaining Latent Representations with a Corpus of Examples. *Advances in Neural Information Processing Systems*, 34:12154–12166, 12 2021.
- Adele Cutler and Leo Breiman. Archetypal Analysis. *Technometrics*, 36(4):338–347, 11 1994. ISSN 0040-1706. doi: 10.1080/00401706.1994.10485840.
- Tyler Davis and Bradley C. Love. Memory for category information is idealized through contrast with competing options. *Psychological Science*, 21(2):234–242, 2010. doi: 10.1177/0956797609357712. URL <https://doi.org/10.1177/0956797609357712>. PMID: 20424052.
- Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pages 248–255, 2009. doi: 10.1109/CVPR.2009.5206848.
- Teresa Dorszewski, Lenka Tětková, Robert Jenssen, Lars Kai Hansen, and Kristoffer Knutsen Wickstrøm. From colors to classes: Emergence of concepts in vision transformers. *arXiv preprint arXiv:2503.24071*, 2025.
- Thomas Fel, Ekdeep Singh Lubana, Jacob S. Prince, Matthew Kowal, Victor Boutin, Isabel Papadimitriou, Binxu Wang, Martin Wattenberg, Demba E. Ba, and Talia Konkle. Archetypal SAE: Adaptive and stable dictionary learning for concept extraction in large vision models. In Aarti Singh, Maryam Fazel, Daniel Hsu, Simon Lacoste-Julien, Felix Berkenkamp, Tegan Maharaj, Kiri Wagstaff, and Jerry Zhu, editors, *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pages 16543–16572. PMLR, 13–19 Jul 2025a.
- Thomas Fel, Binxu Wang, Michael Lepori, Matthew Kowal, Andrew Lee, Randall Balestrieri, Sonia Joseph, Ekdeep Lubana, Talia Konkle, Demba Ba, and Martin Wattenberg. Into the rabbit hull: From task-relevant concepts in dino to minkowski geometry, 10 2025b.
- R. A. Fisher. Iris. UCI Machine Learning Repository, 1936. DOI: 10.24432/C56C76.
- Peter Gärdenfors and Mary-Anne Williams. Reasoning about categories in conceptual spaces. In *Proceedings of the 17th International Joint Conference on Artificial Intelligence - Volume 1, IJCAI’01*, page 385–392, San Francisco, CA, USA, 2001. Morgan Kaufmann Publishers Inc. ISBN 1558608125.
- Ali Hassani, Steven Walton, Nikhil Shah, Abulikemu Abuduweili, Jiachen Li, and Humphrey Shi. Escaping the big data paradigm with compact transformers, 2022. URL <https://arxiv.org/abs/2104.05704>.
- Trevor Hastie, Robert Tibshirani, and Jerome Friedman. *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. Springer Series in Statistics. Springer New York, NY, 2nd edition, 2009. ISBN 978-0-387-84857-0. doi: 10.1007/978-0-387-84858-7.
- Alex Hawkins-Hooker, Henry Kenlay, and John Reid. Projection layers improve deep learning models of regulatory dna function. *BioRxiv*, page 412734, 2018.

- Patrick Helber, Benjamin Bischke, Andreas Dengel, and Damian Borth. Eurosat: A novel dataset and deep learning benchmark for land use and land cover classification. *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, 12(7):2217–2226, 2019. doi: 10.1109/JSTARS.2019.2918242.
- Weonyoung Joo, Wonsung Lee, Sungrae Park, and Il-Chul Moon. Dirichlet variational autoencoder. *Pattern Recognition*, 107:107514, 2020.
- Sebastian Keller, Maxim Samarin, Fabricio Arend Torres, Mario Wieser, and Volker Roth. Learning extremal representations with deep archetypal analysis. *International Journal of Computer Vision*, 129:1–16, 04 2021. doi: 10.1007/s11263-020-01390-3.
- Sebastian Mathias Keller, Maxim Samarin, Mario Wieser, and Volker Roth. Deep archetypal analysis. In Gernot A. Fink, Simone Frintrop, and Xiaoyi Jiang, editors, *Pattern Recognition*, pages 171–185, Cham, 2019. Springer International Publishing. ISBN 978-3-030-33676-9.
- Alex Krizhevsky. Learning multiple layers of features from tiny images. Technical report, University of Toronto, 2009. CIFAR-10 dataset.
- Yann LeCun, Corinna Cortes, and C. J. C. Burges. Mnist handwritten digit database. *ATT Labs [Online]*. Available: <http://yann.lecun.com/exdb/mnist/>, 2010.
- James Liang, Yiming Cui, Qifan Wang, Tong Geng, Wenguan Wang, and Dongfang Liu. Clusterformer: clustering as a universal visual learner. *Advances in neural information processing systems*, 36:64029–64042, 2023.
- Zhuang Liu, Hanzi Mao, Chao-Yuan Wu, Christoph Feichtenhofer, Trevor Darrell, and Saining Xie. A ConvNet for the 2020s. In *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 11966–11976, Los Alamitos, CA, USA, June 2022. IEEE Computer Society. doi: 10.1109/CVPR52688.2022.01167. URL <https://doi.ieeecomputersociety.org/10.1109/CVPR52688.2022.01167>.
- Luca Longo, Mario Brcic, Federico Cabitza, Jaesik Choi, Roberto Confalonieri, Javier Del Ser, Riccardo Guidotti, Yoichi Hayashi, Francisco Herrera, Andreas Holzinger, Richard Jiang, Hassan Khosravi, Freddy Lecue, Gianclaudio Malgieri, Andrés Páez, Wojciech Samek, Johannes Schneider, Timo Speith, and Simone Stumpf. Explainable artificial intelligence (xai) 2.0: A manifesto of open challenges and interdisciplinary research directions. *Information Fusion*, 106:102301, 2024. ISSN 1566-2535. doi: <https://doi.org/10.1016/j.inffus.2024.102301>.
- André F. T. Martins and Ramón F. Astudillo. From softmax to sparsemax: a sparse model of attention and multi-label classification. In *Proceedings of the 33rd International Conference on International Conference on Machine Learning - Volume 48, ICML’16*, page 1614–1623. JMLR.org, 2016.
- Salvatore Milite, Giulio Caravagna, and Andrea Sottoriva. Midaa: deep archetypal analysis for interpretable multi-omic data integration based on biological principles. *Genome Biology*, 26(1):90, 2025.
- Toshinari Morimoto and Su-Yun Huang. Tensorprojection layer: A tensor-based dimension reduction method in deep neural networks. *Neurocomputing*, page 131695, 2025.
- Morten Mørup and Lars Kai Hansen. Archetypal analysis for machine learning and data mining. *Neurocomputing*, 80:54–63, 3 2012. ISSN 0925-2312. doi: 10.1016/J.NEUCOM.2011.06.033.
- Meike Nauta, Jörg Schlötterer, Maurice van Keulen, and Christin Seifert. Pip-net: Patch-based intuitive prototypes for interpretable image classification. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2744–2753, June 2023.
- Yuval Netzer, Tao Wang, Adam Coates, Alessandro Bissacco, Bo Wu, and Andrew Y. Ng. Svhn: The street view house numbers dataset. <http://ufldl.stanford.edu/housenumbers/>, 2011.
- Ramprasaath R Selvaraju, Michael Cogswell, Abhishek Das, Ramakrishna Vedantam, Devi Parikh, and Dhruv Batra. Grad-cam: Visual explanations from deep networks via gradient-based localization. In *Proceedings of the IEEE international conference on computer vision*, pages 618–626, 2017.
- Yonglong Tian, Dilip Krishnan, and Phillip Isola. Contrastive multiview coding. In *Computer Vision – ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XI*, page 776–794, Berlin, Heidelberg, 2020. Springer-Verlag. ISBN 978-3-030-58620-1. doi: 10.1007/978-3-030-58621-8_45. URL https://doi.org/10.1007/978-3-030-58621-8_45.
- Buelent Uendes, Shujian Yu, and Mark Hoogenboom. Start smart: Leveraging gradients for enhancing mask-based XAI methods. In *The Thirteenth International Conference on Learning Representations*, 2025.

- Aaron van den Oord, Oriol Vinyals, and Koray Kavukcuoglu. Neural discrete representation learning. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, NIPS’17, page 6309–6318, Red Hook, NY, USA, 2017. Curran Associates Inc. ISBN 9781510860964.
- David van Dijk, Daniel Burkhardt, Matthew Amodio, Alex Tong, Guy Wolf, and Smita Krishnaswamy. Finding archetypal spaces using neural networks. *arXiv preprint arXiv:1901.09078*, 2019.
- Johanna Vielhaben, Dilyara Bareeva, Jim Berend, Wojciech Samek, and Nils Strodthoff. Beyond scalars: Concept-based alignment analysis in vision transformers. *arXiv preprint arXiv:2412.06639*, 2024.
- Wenguan Wang, Cheng Han, Tianfei Zhou, and Dongfang Liu. Visual recognition with deep nearest centroids. In *International Conference on Learning Representations (ICLR)*, 2023.
- A. Emilie J. Wedenborg and Morten Mørup. Archetypal analysis for binary data. In *ICASSP 2025 - 2025 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 1–5, 2025. doi: 10.1109/ICASSP49660.2025.10888386.
- Anna Emilie Jennow Wedenborg, Teresa Dorszewski, Lars Kai Hansen, Kristoffer Knutsen Wickstrøm, and Morten Mørup. Explaining latent representations of neural networks with archetypal analysis. In Hyeongji Kim, Adín Ramírez Rivera, and Benjamin Ricaud, editors, *Proceedings of the 7th Northern Lights Deep Learning Conference (NLDL)*, volume 307 of *Proceedings of Machine Learning Research*, pages 448–468. PMLR, 06–08 Jan 2026. URL <https://proceedings.mlr.press/v307/wedenborg26a.html>.
- Mario Wieser, Daniel Siegismund, and Stephan Steigele. Revisiting deep archetypal analysis for phenotype discovery in high content imaging. In *2025 IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, pages 3802–3811. IEEE, 2025.
- William Wolberg, Olvi Mangasarian, Nick Street, and W. Street. Breast cancer wisconsin (diagnostic). UCI Machine Learning Repository, 1993. DOI: 10.24432/C5DW2B.
- Han Xiao, Kashif Rasul, and Roland Vollgraf. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms, 2017.
- Jiancheng Yang, Rui Shi, Donglai Wei, Zequan Liu, Lin Zhao, Bilian Ke, Hanspeter Pfister, and Bingbing Ni. Medmnist v2 - a large-scale lightweight benchmark for 2d and 3d biomedical image classification. *Scientific Data*, 10(1):1–10, 2023. ISSN 2052-4463. doi: 10.1038/s41597-022-01721-8.
- Tianfei Zhou and Wenguan Wang. Prototype-based semantic segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 46(10):6858–6872, 2024.

The Polytopal Neural Network

Supplementary Materials

A PROOFS

We use the notation of Section 3.3. The archetype matrix $A \in \mathbb{R}^{K \times d}$ is fixed, $G = AA^\top$, and $\Delta^{K-1} = \{s \in \mathbb{R}^K \mid s \geq 0, \mathbf{1}^\top s = 1\}$. Write

$$F_h(s) = \frac{1}{2} s^\top G s - h^\top s. \quad (13)$$

A.1 Proof of Lemma 1

Since G is positive definite, F_h is strictly convex. The simplex is compact and convex, so a minimizer exists and is unique for every $h \in \mathbb{R}^K$.

Fix a nonempty index set $\mathcal{J} \subseteq \{1, \dots, K\}$ and set $s_{\mathcal{J}^c} = 0$. The stationarity and sum-to-one conditions on these coordinates are

$$\begin{pmatrix} G_{\mathcal{J}\mathcal{J}} & \mathbf{1} \\ \mathbf{1}^\top & 0 \end{pmatrix} \begin{pmatrix} s_{\mathcal{J}} \\ \lambda \end{pmatrix} = \begin{pmatrix} h_{\mathcal{J}} \\ 1 \end{pmatrix}. \quad (14)$$

Here λ is the multiplier for $\mathbf{1}^\top s = 1$. Because $G_{\mathcal{J}\mathcal{J}}$ is positive definite, this system is invertible. Its solution $s_{\mathcal{J}}(h)$ and $\lambda^{\mathcal{J}}(h)$ is therefore affine in h .

This candidate satisfies the remaining KKT conditions exactly when

$$s_{\mathcal{J}}^{\mathcal{J}}(h) \geq 0, \quad (Gs^{\mathcal{J}}(h) - h + \lambda^{\mathcal{J}}(h)\mathbf{1})_{\mathcal{J}^c} \geq 0. \quad (15)$$

These inequalities are affine in h and define a closed polyhedral region. On this region the candidate is the unique minimizer, so Φ is affine there.

Every minimizer has a nonempty support and satisfies the KKT conditions for that support. The regions therefore cover $\mathbb{R}^K$. There are finitely many index sets, and uniqueness ensures that the affine formulas agree wherever their regions overlap.

Finally, let $h_m \rightarrow h$. Compactness of the simplex ensures that every subsequence of $\Phi(h_m)$ has a convergent further subsequence. Passing to the limit in

$$F_{h_m}(\Phi(h_m)) \leq F_{h_m}(s) \quad \text{for every } s \in \Delta^{K-1} \quad (16)$$

shows that every such limit minimizes F_h . Uniqueness forces the limit to equal $\Phi(h)$. Thus $\Phi(h_m) \rightarrow \Phi(h)$, establishing continuity. Hence Φ is continuous piecewise affine. $\square$

A.2 Proof of Theorem 1

By Lemma 1, each coordinate of Φ is a continuous piecewise affine function with finitely many pieces. Every such scalar function admits an exact finite ReLU representation [Arora et al., 2016]. Representing each coordinate and stacking the resulting networks gives a finite ReLU network f_θ satisfying

$$f_\theta(h) = \Phi(h) \quad \text{for all } h \in \mathbb{R}^K. \quad (17)$$

Sparsemax is the Euclidean projection onto the probability simplex [Martins and Astudillo, 2016],

$$\text{sparsemax}(v) = \arg \min_{s \in \Delta^{K-1}} \frac{1}{2} \|s - v\|^2. \quad (18)$$

It therefore leaves every point of Δ^{K-1} unchanged. Since $\Phi(h) \in \Delta^{K-1}$,

$$\text{sparsemax}(f_\theta(h)) = \text{sparsemax}(\Phi(h)) = \Phi(h) \quad \text{for all } h \in \mathbb{R}^K. \quad (19)$$

This proves the theorem. $\square$

B ON THE SIMPLEX PROJECTION

B.1 Quadratic Scaling of Projections via. SMO-Updates

Instead of the QP based estimation producing the full gradient of $\mathbf{C}^\ell$ accounting also for the $\mathbf{S}^\ell$ dependencies on $\mathbf{C}^\ell$ as given by (20):

$$\mathbf{r}_n^\top = (\mathbf{C}^\ell \mathbf{z}_n^\ell)^\top (\mathbf{C}^\ell \mathbf{Z}^\ell \mathbf{z}_n^{\ell\top} + \lambda \mathbf{I})^\top (\mathbf{C}^\ell \mathbf{Z}^\ell \mathbf{Z}^{\ell\top} \mathbf{C}^{\ell\top} + \lambda \mathbf{I})^{-1}_{\mathcal{P}, \mathcal{P}}, \quad (20)$$

we can reduce the gradient computation of $\mathbf{C}^\ell$ to only account for the direct influence that $\mathbf{C}^\ell$ exerts on $\mathbf{r}_n$ keeping $\mathbf{s}_n$ fixed. Consequently, we propagate gradients only through the expression $\mathbf{r}_n^\ell = \mathbf{s}_n^\ell \mathbf{C}^\ell \mathbf{Z}^\ell$, treating $\mathbf{s}_n^\ell$ as constant. This avoids having gradient computations rely on any cubic in scaling matrix inversion.

To further reduce the cubic complexity of estimating $\mathbf{s}_n$ we exploit the recently proposed sequential minimal optimization (SMO) procedure for AA Wedenborg and Mørup [2025]. The SMO procedure only considers two coefficients (p, q) at a time. Defining

$$\sigma_i^\ell = s_{i,p}^\ell + s_{i,q}^\ell, \quad s_{i,p}^\ell = \sigma_i^\ell \alpha_i^\ell, \quad s_{i,q}^\ell = \sigma_i^\ell (1 - \alpha_i^\ell), \quad (21)$$

the problem reduces to a one-dimensional quadratic function in $\alpha_i^\ell \in [0, 1]$ redistributing the mass σ_i^ℓ between the two elements. Importantly, this has a simple closed-form solution given by

$$\alpha_i^{\star\ell} = \text{clip}_{[0,1]} \left(-\frac{\sigma_i(H_{qp}^\ell - H_{qq}^\ell) + d_{i,p}^\ell - d_{i,q}^\ell}{\sigma_i(H_{pp}^\ell - 2H_{pq}^\ell + H_{qq}^\ell)} \right), \quad (22)$$

where $\mathbf{H}^\ell = (\mathbf{C}^\ell \mathbf{Z}^\ell)(\mathbf{C}^\ell \mathbf{Z}^\ell)^\top$, $\mathbf{d}_i^\ell = -\mathbf{z}_i^\ell (\mathbf{C}^\ell \mathbf{Z}^\ell)^\top + \mathbf{s}_i^\ell \mathbf{H}^\ell$.

Pairs with $\sigma_i = 0$ are skipped, as they contain no weight.

Pairs (p, q) are iterated in random order, optionally multiple times per batch in which it was found in Wedenborg and Mørup [2025] that convergence could be achieved with complexity $\mathcal{O}(K^2)$. All updates are performed outside the computational graph such that gradients are propagated only through archetype parameters $\mathbf{C}^\ell$, not through $\mathbf{S}$ and can be implemented trivially in parallel across the considered observations. This yields substantial speedups while preserving high-quality simplex-constrained projections.

B.2 Fidelity of the Amortizer

To test the fidelity of the Amortizer we replace the deployed amortizer with the exact SMO projection at test time. Figure 7 shows the resulting test MSE against the amortized test MSE. All 150 configurations lie on the diagonal. The amortizer therefore does not materially change what the trained network computes.

C INTERPRETABILITY OF THE POLYTOPE

To test the interpretability of the PNN, we test how the output depends on the archetypes, if the archetypes are useful extreme configurations, removing the top archetype should hurt performance more than removing archetypes at random. We therefore take every trained PNN-AE and remove one archetype from the deployed coordinates of 1000 test images. The remaining coordinates are renormalized and decoded by the same decoder. Figures 8 and 9 shows both increases against K .

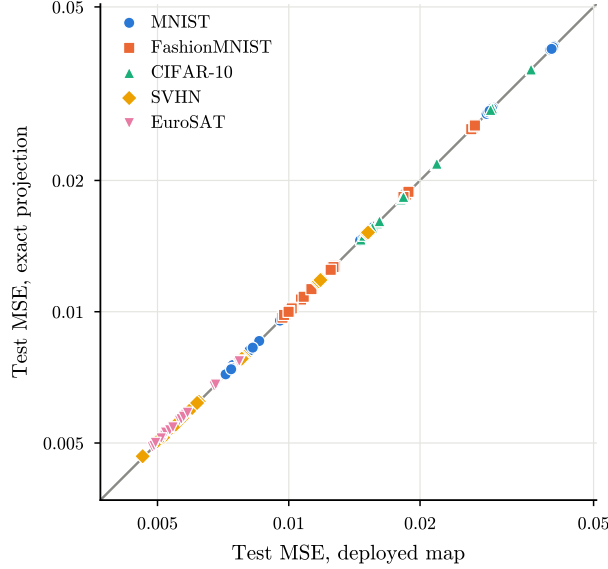


Figure 7: Test MSE of the PNN-AE with the exact SMO projections vs. the Amortized version.

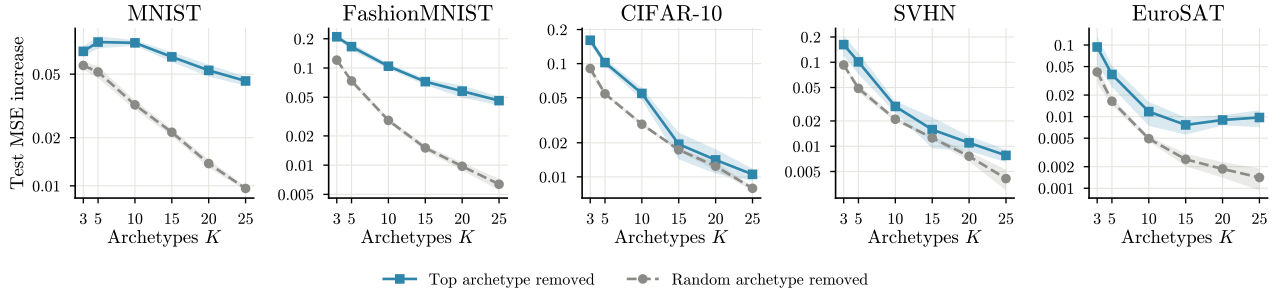


Figure 8: Share of PNN-CNN test predictions that change when the highest-weight archetype is removed and when a single archetype is removed at random, against the number of archetypes K . Mean $\pm$ standard deviation for five seeds.

D TOKEN-WISE REPRESENTATIONS

To make the method applicable to larger scale models with larger backbones we implement a token based version of the PNN. It works by applying the projection along the feature dimension and applying it to each token of a sequence. A patch embedding produces $\mathbf{Z} = [\mathbf{z}_1, \dots, \mathbf{z}_T] \in \mathbb{R}^{T \times d}$, and each token is projected onto the same polytope, $\mathbf{r}_t = \mathbf{s}_t \mathbf{A}$ with $\mathbf{A} = \mathbf{C} \mathbf{Z}^c$. Stacking gives $\mathbf{R} = \mathbf{S} \mathbf{A}$ with $\mathbf{S} \in \mathbb{R}^{T \times K}$. In ImageNet, each corpus element is the cached feature vector of one training image and token. These vectors are recomputed from the current backbone every epoch, and column generation updates corpus membership. The same corpus is shared across positions.

E EXPERIMENTAL DETAILS

E.1 Data

MNIST, FashionMNIST, CIFAR-10 and SVHN use their official test splits. We hold out 10% of each training split for validation. EuroSAT uses a 70/10/20 training, validation and test split. The CNN splits are stratified by class, while the AE splits are unstratified. Each run uses its training seed for the split. MNIST and FashionMNIST images are 28×28 , CIFAR-10 and SVHN images are 32×32 , and EuroSAT images are 64×64 .

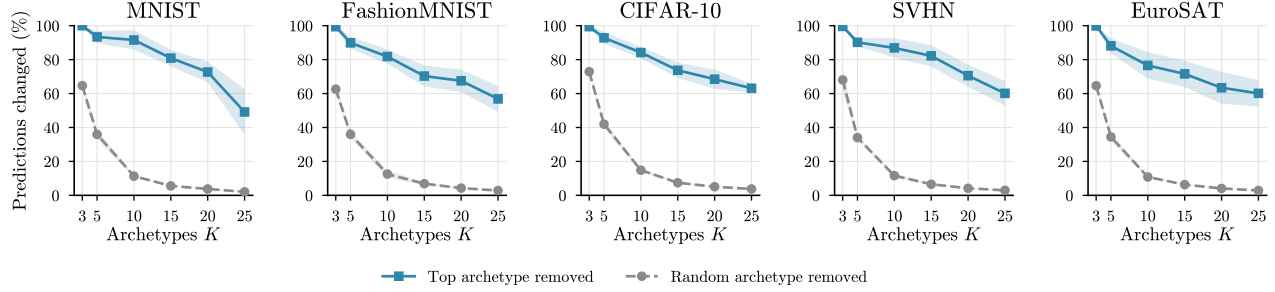


Figure 9: Increase in test MSE of the PNN-AE when the highest-weight archetype is removed and when a single archetype is removed at random, against the number of archetypes K . Mean $\pm$ standard deviation for five seeds.

Classifier inputs use fixed channel normalization. CIFAR-10, SVHN and EuroSAT training batches use reflected random crops with padding 4 and horizontal flips with probability 0.5. MNIST and FashionMNIST use reflected random crops with padding 2 and no flips.

Other experiments. PathMNIST, DermaMNIST and BloodMNIST from MedMNIST v2 [Yang et al., 2023] use their official splits at 64×64 and the classifier augmentation above. ImageNet-100 [Tian et al., 2020] is the class subset of ImageNet-1k [Deng et al., 2009]. Both train on random resized crops with flips and evaluate on a centre crop. The frozen transformer uses CIFAR-10 with 5000 training images held out for validation.

E.2 Architectures

The image encoder has three strided convolutions, $\text{Conv}_{3 \times 3, s=2}(C, 32)$, $\text{Conv}_{3 \times 3, s=2}(32, 64)$ and $\text{Conv}_{3 \times 3, s=2}(64, 128)$, each followed by group normalization with 8, 16 and 32 groups and a ReLU. The feature map is flattened and linearly projected to $\mathbf{z} \in \mathbb{R}^{64}$. The classifier applies $\mathbf{h}_1 = \text{ReLU}(W_1 \mathbf{z})$, a PNN projection $\tilde{\mathbf{h}}_1$, $\mathbf{h}_2 = \text{ReLU}(W_2 \tilde{\mathbf{h}}_1)$, a second PNN projection $\tilde{\mathbf{h}}_2$, and a linear output layer. The first site forms archetypes from their encoded features. Before the second site, the corpus passes through the first PNN projection and the second fully connected layer, as the input images do. Each site forms its archetypes from the corpus features entering that site. The autoencoders use the same encoder, a PNN bottleneck, and a mirrored transposed-convolution decoder with group normalization, ReLU and a sigmoid output.

Other experiments. The MedMNIST classifier is the CNN above with a 128-dimensional latent space. On ImageNet, ConvNeXt-T start from ImageNet-1k weights. Their spatial tokens are projected to 256 dimensions by a PCA-initialized linear layer, and the PNN projects every token (Appendix D). The reconstructions are averaged over tokens before a linear classifier. The unconstrained model applies the same projection to the pooled feature, and the width control uses 128 dimensions. The frozen-head experiment places the PNN after the attention pooling of a compact transformer [Hassani et al., 2022] and initializes its output layer from the backbone classifier.

E.3 Model Variants

PNN. The amortizer of Eq. (5) has two hidden layers of width $\max(8K, 128)$ and ReLU activations. The CNN and AE refine its detached output with K SMO rounds during both training and evaluation.

PNN-VQ. One-hot coordinates by nearest archetype, with the reconstruction or task loss only. In the autoencoder, $\mathbf{C}$ is initialized from k -means clusters of the encoded corpus, using a softened form of $(\mathbf{S}^\top \mathbf{S})^{-1} \mathbf{S}^\top$ from Section 3.5 with 95% of each row of $\mathbf{C}$ spread uniformly over its cluster and the remaining 5% spread uniformly over the other corpus points, and is then learned by the task gradient.

PNN-PL. Unconstrained $\mathbf{C}$ and least squares coordinates.

VQ-VAE. A codebook updated by an exponential moving average, the straight-through estimator, and the commitment loss $\beta \|\mathbf{Z} - \text{sg}[\mathbf{R}]\|_F^2$ with $\beta = 0.25$ [van den Oord et al., 2017]. The codebook is initialized by

k -means++ and unused entries are reinitialized.

DirVAE. The encoder outputs positive concentrations. Training uses the inverse Gamma CDF approximation and normalizes the samples onto the simplex [Joo et al., 2020]. Evaluation uses the posterior mean. The loss adds $\beta_{\text{KL}} D^{-1} \sum_k \text{KL}(\text{Gamma}(\alpha_k, 1) \parallel \text{Gamma}(1, 1))$ to mean pixel MSE, where D counts pixel values and $\beta_{\text{KL}} = 0.01$.

E.4 Training Protocol

The main CNN and AE grids use $K \in \{3, 5, 10, 15, 20, 25\}$ and seeds $\{42, 123, 456, 789, 1024\}$. Runs use AdamW, cosine learning-rate schedules and gradient-norm clipping. Table 2 lists epoch limits and initial learning rates. Early stopping can shorten each phase.

Phase 1 loads the unconstrained network trained with the same seed. Before Phase 2, AA on up to 20,000 encoded training images selects one anchor image per corpus point. The anchor starts with weight 0.25 and the remaining weight is spread uniformly over the training set. The archetypes are initialized on this corpus. Phase 2 trains the archetype weights and output layer, with the encoder frozen. The CNN also freezes its two inner fully connected layers. Phase 3 updates the task network after an initial period with the encoder frozen. The corpus logits are trained by the task loss through the mini-batch rows of Eq. (4).

Amortizer fitting. The AE fits the amortizer by minimizing $\|z - \hat{\mathbf{S}}\mathbf{A}\|^2$, normalized by the mean squared feature magnitude. CNN warm starts and refits minimize coordinate MSE against targets computed with $2K$ batched SMO rounds. Features, archetypes and targets are detached during fitting. Between refits, each training step also takes one amortizer step on the normalized projection residual $|z - \hat{\mathbf{S}}\mathbf{A}|^2 / |\mathbf{z}|^2$, with $\mathbf{z}$ and $\mathbf{A}$ detached. This step updates only the amortizer, and the task loss never updates it.

Refits run every ten epochs and after Phase 2. The AE also refits after Phase 3. Its final refit is retained only if the measured projection residual on training data does not increase. The CNN has no post-Phase-3 refit.

Table 2: Hyperparameters of the main CNN and AE grids. Epoch counts are maxima and learning rates are initial values.

	CNN classifier	Autoencoder
Latent dimension	64	64
Corpus size	2048	2048
Baseline batch size	256	2048
Phase 2 and Phase 3 batch size	512	512
Unconstrained baseline, epochs	200	200
Phase 2 epochs	60	100
Phase 3 epochs	300	200
Backbone frozen at the start of Phase 3	30 epochs	20 epochs
Learning rate, unconstrained baseline	10^{-3}	10^{-3}
Learning rate, archetypes in Phase 2	$3 \cdot 10^{-3}$	$5 \cdot 10^{-3}$
Learning rate, output layer in Phase 2	$5 \cdot 10^{-4}$	$2 \cdot 10^{-3}$
Learning rate in Phase 3, encoder / output / archetypes	$10^{-4} / 2 \cdot 10^{-4} / 10^{-3}$	$10^{-4} / 2 \cdot 10^{-4} / 10^{-3}$
Learning rate, CNN inner layers in Phase 3	10^{-4}	–
Gradient clipping	2.0	1.0
Amortizer epochs, warm start / refit after Phase 2 / after Phase 3	200 / 100 / 0	200 / 100 / 100
Amortizer refit interval / maximum epochs per refit	10 / 30	10 / 30
Initial amortizer temperature τ	1	1
Learning rate, corpus logits, Phase 2 / Phase 3	$2 \cdot 10^{-2} / 5 \cdot 10^{-3}$	$10^{-2} / 10^{-3}$

Task objective and adaptation. The task loss is pixel MSE for the AE and cross-entropy for the CNN, with no latent AA penalty. After periodic amortizer refits, two epochs adapt the output layer at learning rate 10^{-3} . The AE also allows up to 50 decoder-only adaptation epochs before Phase 3. The layer-count and simplex-map ablations use seeds $\{42, 123, 456\}$ and $K \in \{3, 10, 25\}$. Their comparison arms use the same seeds.

E.5 Training beyond the main grids

Table 3 summarizes the MedMNIST, ImageNet and frozen-transformer experiments. Appendices E.1 and E.2 describe their data and architectures. They share the PNN layer, AA corpus selection, column generation and an amortizer that the task loss does not update. They differ in backbone, optimizer, coordinate map and reported metric.

Table 3: Settings of the experiments beyond the main CNN and AE grids. Unconstrained models use the same backbone, data and optimization as the PNN of their column. The ImageNet-100 PNN has three seeds and its unconstrained model five.

	MedMNIST	ImageNet-100	ImageNet-1k	Frozen head
Backbone	two-site CNN	ConvNeXt-T	ConvNeXt-T	compact ViT
Backbone initialization	baseline, same seed	ImageNet-1k	ImageNet-1k	CIFAR-10, same seed
Input resolution	64×64	224×224	224×224	32×32
PNN site	two FC sites	every token	every token	pooled feature
PNN input dimension	128	256	256	256
K	10, 25	25, 50, 100, 150, 200	128, 256	3, 5, 10, 20, 40
Corpus	2048 images	3200 token rows	32,768 token rows	2048 feature rows
Seeds	5	3	2	5
Optimizer	AdamW	SGD	SGD	AdamW
Batch size	256	512	512	512
Epochs	80 + 160	2 + 30	2 + 30	80
Amortizer hidden width	$\max(8K, 128)$	1024	1024	$\max(8K, 128)$
Coordinate map	K refinement rounds	amortizer only	amortizer only	K refinement rounds
Reported split	test	validation	validation	test
Checkpoint	best validation	Final epoch	Final epoch	best validation

MedMNIST. Training follows the main CNN protocol with the changes in Table 3. Phase 2 and Phase 3 run for up to 80 and 160 epochs, the encoder stays frozen for the first 20 Phase 3 epochs, and the post-Phase-2 amortizer refit is limited to 50 epochs.

ImageNet. Two epochs with the backbone frozen are followed by 30 epochs of fine-tuning with SGD, learning rate 0.02, a backbone multiplier of 0.1, momentum 0.9, weight decay 10^{-4} , label smoothing 0.1 and cosine decay. As in the main grids, AA selects the corpus from training tokens and column generation updates it. The amortizer is fitted to projected-gradient targets and refitted when its arg-max agreement falls below 0.85. It is deployed without SMO refinement.

Frozen transformer head. Each backbone is trained for 160 epochs with mixup, CutMix, RandAugment and random erasing. The head trains for 80 epochs with AdamW, learning rate 10^{-3} and cosine decay, using K refinement rounds. Its amortizer is refitted to $2K$ -round SMO targets every ten epochs, when column generation also updates the corpus.

The released code records every remaining setting.

E.6 Choice of K

We define K as the number of archetypes that span the convex polytope approximating the latent representation manifold. This hyperparameter controls the geometric expressivity of the model, meaning that smaller values of K impose a stronger convex constraint, leading to a more compact and interpretable representation, whereas larger values increase the capacity to capture fine-grained variations in the data. Thus, K governs a trade-off between compression and fidelity. From a geometric perspective, increasing K refines the polytope approximation of the underlying data manifold, enabling the network to capture higher-order structures while maintaining interpretability through the archetypal basis. We recommend systematically assessing the impact of the choice of K on performance as assessed in Figure 3 and for ease of interpretation select the smallest K with sufficient performance. We presently used the same number of archetypes for each PNN layer. Whereas this can be

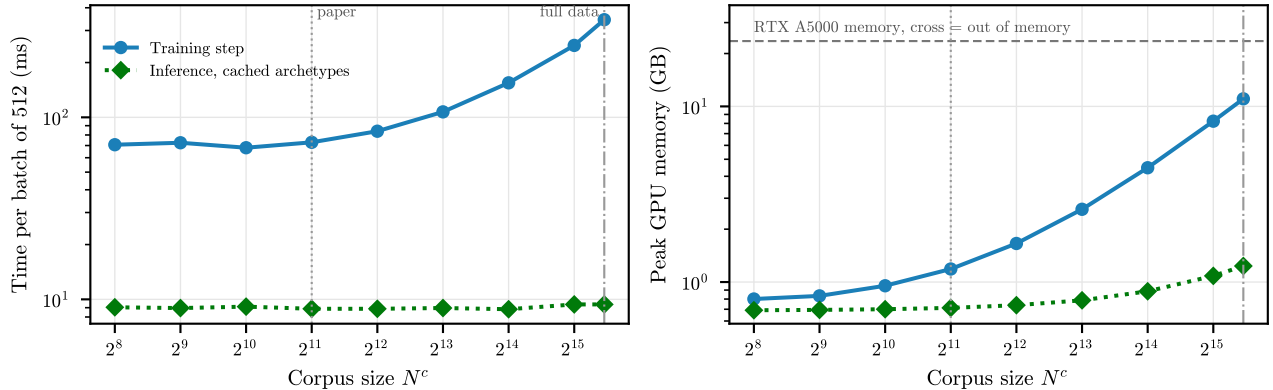


Figure 10: Cost of the PNN-CNN (CIFAR-10, $K = 10$) against the corpus size N^c , on one NVIDIA RTX A5000. Left: time per batch of 512 for a Phase-3 training step and for inference with the archetypes cached. Right: peak GPU memory of the same. Using the full training set as the corpus ($N^c = 45,000$) costs $4.7\times$ the training time and $9.3\times$ the memory of the learned corpus ($N^c = 2048$); inference time does not depend on N^c .

defined for each layer separately, tuning separate values of K for each layer requires the fitting of many models and future work should investigate efficient search strategies for identifying suitable layer specific values of K .

E.7 Training costs

Table 4 shows that training time increases with K , while memory changes only slightly. As expected PNN-PL and PNN-VQ are cheaper than PNN at the same K . Figure 10 shows that a smaller corpus reduces training time and memory.

Table 4: Measured training cost of the CNN models on one NVIDIA RTX A5000. Each model trains for 100 epochs from its seed-42 checkpoint with its own training step.

	Unconstrained	PNN $K = 3$	PNN $K = 10$	PNN $K = 25$	PNN-PL $K = 25$	PNN-VQ $K = 25$
<i>Training time (minutes per 100 epochs)</i>						
MNIST	5.3	10.3	14.9	24.8	7.7	6.7
FMNIST	5.3	10.3	14.9	25.5	7.8	6.6
CIFAR-10	4.5	9.0	12.8	21.1	6.6	5.9
SVHN	6.5	14.2	18.8	31.4	9.7	8.5
EuroSAT	2.1	5.4	7.0	10.5	4.2	3.7
<i>Peak GPU memory in training (GB)</i>						
MNIST	0.24	0.60	0.61	0.64	0.52	0.52
FMNIST	0.24	0.60	0.61	0.64	0.52	0.52
CIFAR-10	0.70	1.18	1.19	1.22	1.08	1.08
SVHN	1.12	1.60	1.61	1.65	1.51	1.51
EuroSAT	1.31	3.22	3.22	3.26	2.84	2.84

F LARGER BACKBONES

Table 5 reports ImageNet-100 with ConvNeXt-T, and Table 6 reports ImageNet-1k with ConvNeXt-T. Experimental details for these runs can be found in Appendix E.

G ADDITIONAL EXPERIMENTAL RESULTS

This appendix provides comprehensive experimental results and implementation details to support reproducibility. We organize the material as follows: Section G.1 presents additional MLP classifier results; Section G.2

Table 5: ImageNet-100 at 224×224 with a fine-tuned ConvNeXt-T. Top-1 accuracy (%) on the 5,000 validation images at the final epoch, mean $\pm$ std.

	Unconstrained	$K=25$	$K=50$	$K=100$	$K=150$	$K=200$
ConvNeXt-T	94.6 $\pm$ 0.1	91.7 $\pm$ 0.5	92.0 $\pm$ 0.3	91.9 $\pm$ 0.7	92.5 $\pm$ 0.2	92.0 $\pm$ 0.7

Table 6: ImageNet-1k with a fine-tuned ConvNeXt-T. Top-1 accuracy (%) on the 50,000 validation images at the final epoch, mean $\pm$ std over two seeds.

	Top-1	Seeds
Unconstrained, 256-d linear projection	81.2 $\pm$ 0.0	2
Unconstrained, 128-d linear projection	81.1 $\pm$ 0.1	2
PNN, $K = 128$	78.6 $\pm$ 0.2	2
PNN, $K = 256$	78.9 $\pm$ 0.1	2

provides CNN classifier analysis and visualizations; Section G.3 reports autoencoder reconstruction experiments.

G.1 Multi-Layer Perceptron Classifiers

G.1.1 Tabular Dataset Performance

We evaluate PNN-constrained MLPs on four tabular benchmark datasets: Iris, Wine, Breast Cancer, and Digits. These datasets vary in dimensionality (4-64 features), sample size (150-5,620 instances), and task complexity (2-10 classes), enabling systematic assessment of polytopal constraints under diverse conditions.

Table 8 presents the mean test accuracy as a function of the number of archetypes K for each dataset. Across all benchmarks, PNN-MLP approaches the performance of the unconstrained baseline as K increases, with minimal degradation even at low archetype counts. Notably, PNN-MLP consistently outperforms both PNN-VQ-MLP and PNN-PL-MLP variants, demonstrating that soft convex combinations better preserve discriminative information compared to hard assignments (VQ) or unconstrained projections (PL). The Wine and Breast Cancer datasets exhibit particularly strong performance, achieving near-baseline accuracy with as few as 5 archetypes, suggesting that the decision boundaries in these tasks align well with polytopal geometry.

G.1.2 Validation of Scalable Training Procedures

To validate our scalable training framework (3.2), we compare mini-batched updates against full-batch optimization on three small datasets where exact computation is tractable. Table 7 reports test accuracy for three projection solvers: CVX (generic convex solver), QP (quadratic programming with linear constraints), and SMO (our Sequential Minimal Optimization approach).

The results demonstrate that: (1) mini-batched training achieves equivalent accuracy to full-batch methods, validating our corpus streaming approach; (2) SMO-based updates match the performance of exact QP solvers while avoiding cubic-complexity matrix inversions; and (3) the simplified gradient computation (treating $\mathbf{S}$ as constant) introduces negligible bias. These findings confirm that our scalability optimizations preserve model quality while enabling application to large-scale datasets.

G.1.3 Training Phase Ablation

Table 8 decomposes the performance across the three training phases described in E.4: Pretrained (Phase 1, backbone only), Polytope (Phase 2, polytope only), and Full (Phase 3, end-to-end).

Several patterns emerge from this ablation. First, for standard PNN-MLP, the Polytope phase stays within 4 points of the Pretrained accuracy on Breast Cancer, Iris and Wine for $K \geq 5$, but reaches only 35.7 to 83.3% on Digits.

Second, PNN-PL consistently maintains pretrained performance in Phase 2, confirming that unconstrained sub-

Table 7: Test accuracy (%) comparison between batched and non-batched training across optimization methods. Results show mean $\pm$ std. Batched training with SMO achieves statistically equivalent performance to exact methods while enabling scalability.

	K	CVX		QP		SMO		AMORTIZED	
		BATCH	$\neg$ BATCH	BATCH	$\neg$ BATCH	BATCH	$\neg$ BATCH	BATCH	$\neg$ BATCH
Breast	5	96.1 $\pm$ 2.3	97.1 $\pm$ 1.3	96.4 $\pm$ 1.5	96.4 $\pm$ 1.8	95.4 $\pm$ 4.1	96.8 $\pm$ 1.5	96.3 $\pm$ 2.2	96.3 $\pm$ 1.8
	10	97.3 $\pm$ 1.2	97.1 $\pm$ 1.2	96.4 $\pm$ 1.2	97.2 $\pm$ 1.4	96.7 $\pm$ 1.1	97.0 $\pm$ 1.2	96.7 $\pm$ 1.7	96.3 $\pm$ 1.7
	25	97.4 $\pm$ 1.0	96.9 $\pm$ 1.5	96.5 $\pm$ 0.0	97.4 $\pm$ 2.0	96.9 $\pm$ 2.5	96.9 $\pm$ 1.5	96.0 $\pm$ 0.7	96.3 $\pm$ 1.4
Iris	5	93.7 $\pm$ 4.0	94.7 $\pm$ 2.8	94.3 $\pm$ 3.5	95.0 $\pm$ 3.6	93.7 $\pm$ 4.0	93.0 $\pm$ 5.1	94.0 $\pm$ 5.3	84.7 $\pm$ 15.1
	10	95.3 $\pm$ 2.8	95.3 $\pm$ 2.8	95.3 $\pm$ 2.8	95.3 $\pm$ 2.8	95.7 $\pm$ 2.7	96.0 $\pm$ 3.1	94.0 $\pm$ 3.9	94.0 $\pm$ 3.9
	25	95.0 $\pm$ 3.2	95.0 $\pm$ 3.2	95.3 $\pm$ 3.2	95.3 $\pm$ 2.8	94.7 $\pm$ 2.8	95.3 $\pm$ 2.8	93.3 $\pm$ 4.7	94.0 $\pm$ 3.9
Wine	5	98.3 $\pm$ 2.3	97.5 $\pm$ 3.3	98.3 $\pm$ 2.3	97.2 $\pm$ 3.2	98.1 $\pm$ 2.3	97.2 $\pm$ 3.2	97.2 $\pm$ 1.8	97.8 $\pm$ 2.1
	10	98.1 $\pm$ 2.3	98.3 $\pm$ 2.3	97.5 $\pm$ 2.4	97.5 $\pm$ 2.4	97.2 $\pm$ 2.3	98.1 $\pm$ 2.3	98.9 $\pm$ 1.4	98.3 $\pm$ 2.2
	25	100.0 $\pm$ 0.0	98.6 $\pm$ 2.0	98.6 $\pm$ 2.0	98.6 $\pm$ 2.0	97.2 $\pm$ 0.0	98.6 $\pm$ 2.0	98.3 $\pm$ 2.2	97.8 $\pm$ 2.1

space projections impose negligible information loss. The near-perfect preservation (e.g., 97.8% on Wine in the Polytope phase for every $K \geq 5$) demonstrates that the learned subspaces capture the essential structure of the pretrained features.

Third, PNN-VQ exhibits substantial performance degradation in Phase 2, particularly at low K values. For the Breast Cancer dataset with $K = 3$, PNN-VQ drops to 86.0% compared to the 96.3% pretrained baseline, indicating that hard discrete assignments cannot faithfully represent the continuous structure of pretrained features without co-adaptation. However, the performance recovers somewhat as K increases, suggesting that a sufficiently large codebook can approximate the required representational capacity. PNN-VQ-MLP stays competitive on Breast Cancer, Iris and Wine at larger K .

Finally, Phase 3 end-to-end training (Full) provides modest but consistent improvements for standard PNN. This validates our three-phase protocol: pretraining establishes a strong initialization, polytope-only optimization verifies representational capacity, and joint fine-tuning enables full co-adaptation when necessary.

G.2 CNN Experiments

Table 9 applies the PNN-CNN to PathMNIST, DermaMNIST and BloodMNIST from MedMNIST v2 [Yang et al., 2023], with $K \in \{10, 25\}$ and five seeds, more information in Appendix E.

Table 10 compares one constrained site with two under an otherwise identical configuration on MNIST, FashionMNIST and CIFAR-10.

G.3 AE Experiments

From Figure 11 we see how the size of the polytope affects the quality of the reconstruction in the datasets. A larger polytope means less compression and therefore more expressive power which results in better reconstruction.

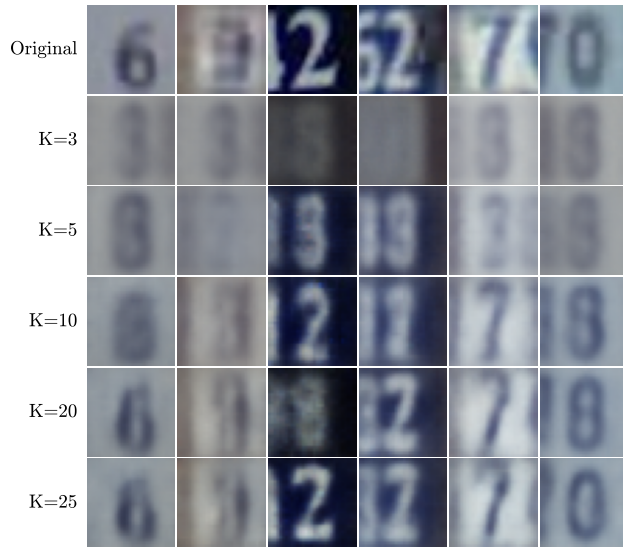
From Figure 12 shows how each of the models reconstruct across datasets. We see that the PNN visually offers competitive reconstruction quality.

In table 11 we further compared using a softmax instead of a sparsemax in the amortizer. Both methods have similar mean MSE.

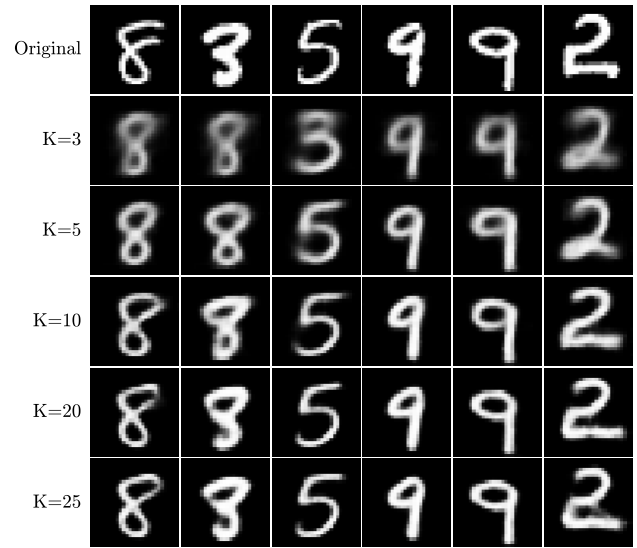
G.3.1 Vector Quantization

To benchmark polytopal soft-clustering against discrete hard-clustering, we compare to the VQ-VAE. VQ-VAE discretizes the latent space using a finite codebook of embeddings $\mathbf{E}^\ell$.

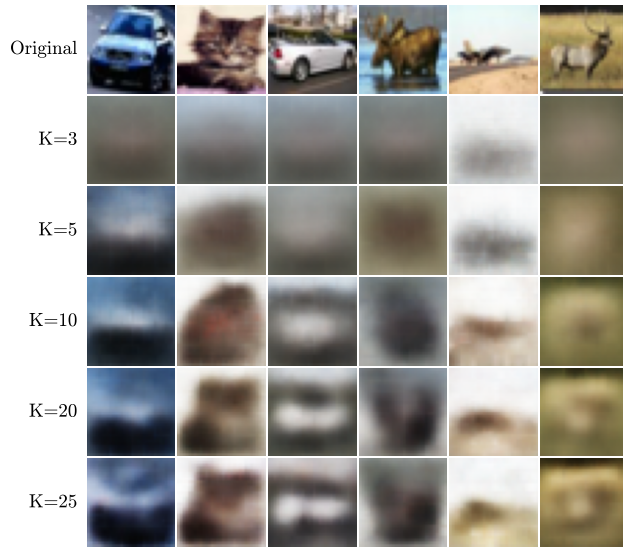
The codebook is updated using an exponential moving average (EMA) scheme rather than direct gradient descent. Gradients are propagated through the quantization operation using the Straight-Through Estimator



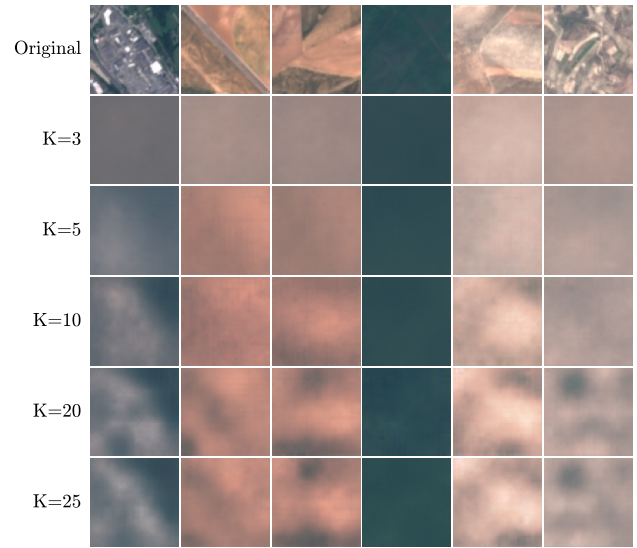
(a) SVHN



(b) MNIST



(c) CIFAR-10



(d) EuroSAT

Figure 11: PNN-AE reconstruction by K for SVHN, MNIST, CIFAR-10, EuroSAT

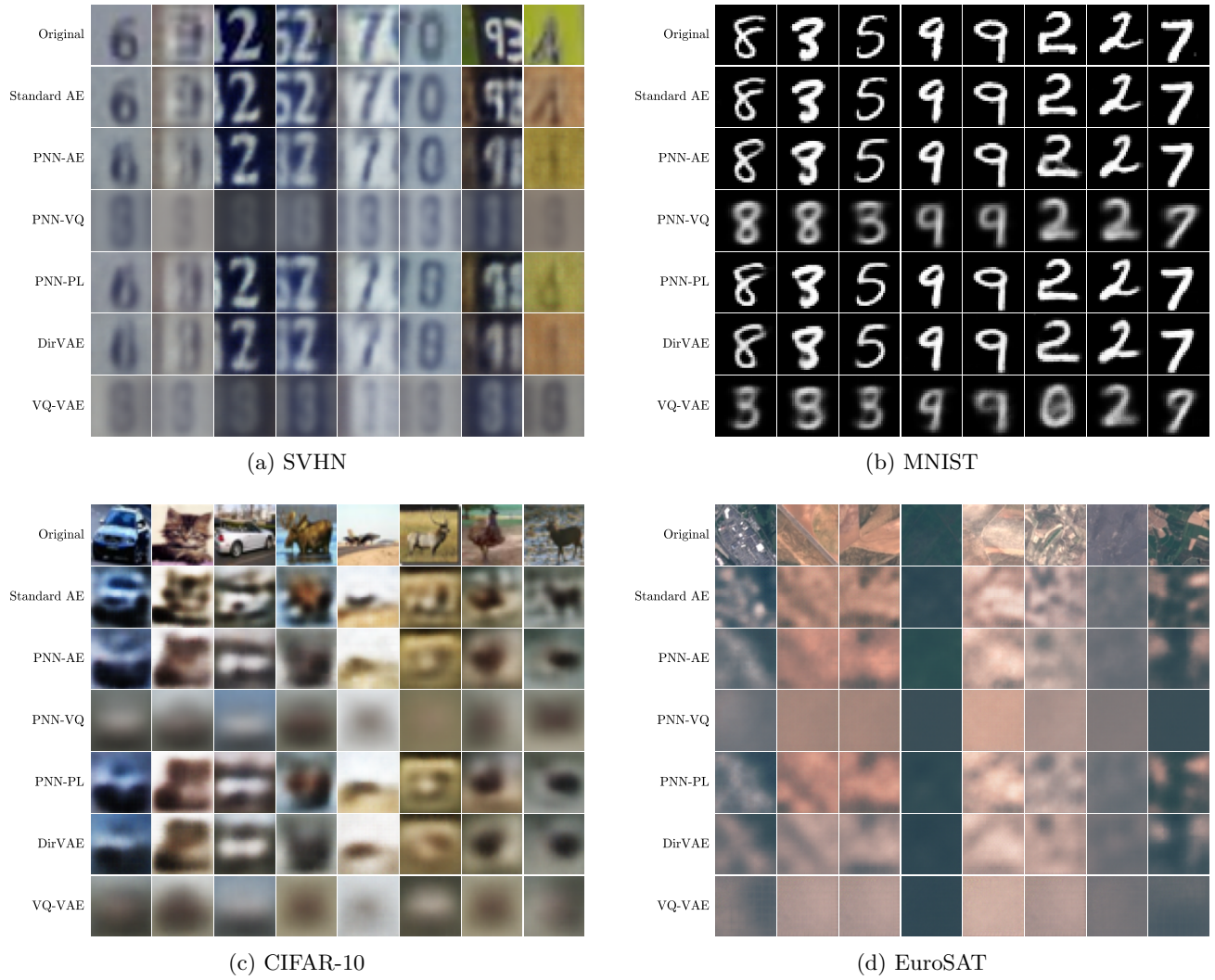


Figure 12: Reconstruction by model for $K = 25$ for SVHN, MNIST, CIFAR-10 and EuroSAT.

Table 8: MLP test classification accuracy (%) across training phases for PNN variants

	K	PRETRAINED	PNN-PL-MLP		PNN-VQ-MLP		PNN-MLP	
			POLYTOPE	FULL	POLYTOPE	FULL	POLYTOPE	FULL
Breast	3	96.3±2.3	96.3±2.3	96.0±2.9	86.0±13.9	85.6±13.7	85.5±15.6	92.0±10.5
	5	96.3±2.3	96.5±2.2	96.1±2.6	93.3±4.7	93.0±4.6	97.1±2.0	96.8±1.9
	7	96.3±2.3	96.3±2.3	96.3±2.3	89.5±6.4	89.3±6.4	96.2±2.0	95.7±2.6
	10	96.3±2.3	96.3±2.3	96.3±2.3	93.2±3.4	92.6±3.1	96.9±2.2	96.8±1.9
	15	96.3±2.3	96.3±2.3	96.3±2.3	94.4±1.8	92.6±5.1	97.0±1.9	96.9±2.1
	20	96.3±2.3	96.3±2.3	96.1±2.6	94.2±3.0	93.9±2.5	97.4±1.8	96.2±1.7
	25	96.3±2.3	96.3±2.3	96.1±2.6	95.8±1.7	94.0±2.7	96.7±1.7	96.3±1.4
Digits	3	93.5±1.5	74.4±5.3	85.6±3.0	18.2±8.3	19.6±9.5	19.2±7.8	23.8±11.5
	5	93.5±1.5	91.9±1.9	92.9±2.0	15.7±6.0	18.7±7.8	35.7±14.1	45.0±15.9
	7	93.5±1.5	93.3±1.6	93.6±1.8	22.1±10.2	23.7±8.7	48.9±14.9	59.3±12.2
	10	93.5±1.5	93.5±1.5	93.7±2.2	20.6±7.6	25.8±4.0	54.0±17.9	64.0±18.6
	15	93.5±1.5	93.5±1.5	93.8±2.1	30.2±8.4	30.7±9.0	77.1±4.0	80.7±3.5
	20	93.5±1.5	93.5±1.5	93.7±2.2	26.9±13.7	31.9±10.4	74.4±14.9	81.4±8.0
	25	93.5±1.5	93.4±1.5	93.8±2.1	40.7±16.3	44.1±13.4	83.3±6.1	86.7±4.9
Iris	3	94.0±4.3	94.7±3.8	94.7±3.8	85.3±11.4	85.3±11.4	83.0±12.1	89.0±9.2
	5	94.0±4.3	94.7±3.8	94.7±3.8	85.3±7.7	85.3±7.7	90.0±13.0	93.3±5.2
	7	94.0±4.3	94.0±4.3	94.7±3.8	92.7±4.3	92.7±4.3	90.3±9.2	93.7±5.1
	10	94.0±4.3	94.0±4.3	94.7±3.8	88.7±1.8	88.7±1.8	94.3±4.2	94.3±4.2
	15	94.0±4.3	94.0±4.3	94.7±3.8	92.0±4.5	92.0±4.5	93.3±5.4	93.0±6.0
	20	94.0±4.3	94.0±4.3	94.0±4.3	93.3±4.1	93.3±4.1	94.0±4.7	93.0±4.0
	25	94.0±4.3	94.0±4.3	94.0±4.3	92.7±4.3	93.3±4.7	93.3±5.2	94.0±5.4
Wine	3	97.8±2.3	97.2±2.0	97.8±2.3	71.1±19.1	71.1±19.1	85.8±16.1	88.6±14.7
	5	97.8±2.3	97.8±2.3	97.8±2.3	91.7±5.2	91.1±4.6	97.2±2.6	97.5±1.6
	7	97.8±2.3	97.8±2.3	97.8±2.3	97.8±3.6	97.8±3.6	97.8±3.2	97.5±2.4
	10	97.8±2.3	97.8±2.3	97.8±2.3	96.1±2.5	95.0±2.3	98.1±1.9	98.3±1.4
	15	97.8±2.3	97.8±2.3	98.3±2.5	96.7±3.0	97.2±2.8	97.2±2.6	96.9±3.1
	20	97.8±2.3	97.8±2.3	98.3±2.5	96.7±3.6	97.2±3.9	96.9±2.8	97.2±2.6
	25	97.8±2.3	97.8±2.3	98.3±2.5	97.8±3.0	97.8±3.0	98.1±2.3	98.1±2.3

(STE), passing gradients from quantized latents $\mathbf{R}^\ell$ to encoder outputs $\mathbf{Z}^\ell$ unchanged.

The training objective is

$$\mathcal{L} = \mathcal{L}_{\text{task}} + \beta \sum_{\ell} \|\mathbf{Z}^\ell - \text{sg}[\mathbf{R}^\ell]\|_F^2, \quad (23)$$

where the commitment term encourages encoder outputs to remain close to their assigned codebook vectors. To prevent codebook collapse, $\mathbf{E}^\ell$ is initialized via greedy k-means++ on the full training set, and unused embeddings are periodically reinitialized.

A direct comparison between the PNN-VQ and the VQ-VAE can be seen in table 12.

Table 9: CNN test accuracy (%) on three MedMNIST datasets at 64×64 , mean $\pm$ std over five seeds. Each PNN run starts from the unconstrained network of the same seed and uses the deployed coordinate map.

	Classes	Unconstrained	PNN $K = 10$	PNN $K = 25$
PathMNIST	9	90.4 $\pm$ 1.2	88.5 $\pm$ 1.5	89.1 $\pm$ 1.5
DermaMNIST	7	78.6 $\pm$ 0.7	76.4 $\pm$ 0.6	77.4 $\pm$ 0.7
BloodMNIST	8	98.3 $\pm$ 0.0	98.1 $\pm$ 0.2	98.1 $\pm$ 0.1

Table 10: CNN test accuracy (%) with one constrained site or two, deployed coordinates (the amortizer followed by K rounds of warm-started batched SMO), mean $\pm$ std over the same three seeds for every column. The configuration is otherwise identical.

	K	Unconstrained	One site	Two sites
MNIST	3	99.6 $\pm$ 0.0	98.4 $\pm$ 1.3	91.7 $\pm$ 12.8
	10		99.4 $\pm$ 0.0	99.4 $\pm$ 0.1
	25		99.5 $\pm$ 0.0	99.5 $\pm$ 0.0
FMNIST	3	93.2 $\pm$ 0.1	92.1 $\pm$ 0.1	91.6 $\pm$ 0.6
	10		92.8 $\pm$ 0.3	92.8 $\pm$ 0.2
	25		93.0 $\pm$ 0.1	92.7 $\pm$ 0.1
CIFAR-10	3	82.3 $\pm$ 0.3	76.9 $\pm$ 0.7	75.7 $\pm$ 0.6
	10		81.6 $\pm$ 0.4	81.5 $\pm$ 0.5
	25		81.8 $\pm$ 0.4	82.2 $\pm$ 0.4

Table 11: PNN-AE reconstruction MSE with a softmax or sparsemax amortizer output. Values are mean $\pm$ std over the same three seeds.

	K	Softmax	Sparsemax
MNIST	3	0.0399 $\pm$ 0.0003	0.0400 $\pm$ 0.0005
	10	0.0147 $\pm$ 0.0005	0.0148 $\pm$ 0.0001
	25	0.0073 $\pm$ 0.0001	0.0075 $\pm$ 0.0003
FMNIST	3	0.0266 $\pm$ 0.0002	0.0268 $\pm$ 0.0003
	10	0.0127 $\pm$ 0.0001	0.0127 $\pm$ 0.0001
	25	0.0099 $\pm$ 0.0002	0.0100 $\pm$ 0.0004
CIFAR-10	3	0.0363 $\pm$ 0.0001	0.0363 $\pm$ 0.0001
	10	0.0218 $\pm$ 0.0000	0.0218 $\pm$ 0.0000
	25	0.0152 $\pm$ 0.0004	0.0157 $\pm$ 0.0007

Table 12: Reconstruction MSE of a standard VQ-VAE (EMA codebook, commitment loss, straight-through estimator) and of PNN-VQ (direct gradient updates of a corpus-grounded codebook, reconstruction loss only), mean $\pm$ std over five seeds.

	K	VQ-VAE	PNN-VQ
MNIST	3	0.0613 $\pm$ 0.0005	0.0614 $\pm$ 0.0006
	5	0.0568 $\pm$ 0.0006	0.0580 $\pm$ 0.0007
	10	0.0530 $\pm$ 0.0006	0.0527 $\pm$ 0.0009
	15	0.0502 $\pm$ 0.0005	0.0493 $\pm$ 0.0004
	20	0.0478 $\pm$ 0.0001	0.0474 $\pm$ 0.0003
	25	0.0464 $\pm$ 0.0004	0.0458 $\pm$ 0.0007
FMNIST	3	0.0635 $\pm$ 0.0049	0.0611 $\pm$ 0.0013
	5	0.0540 $\pm$ 0.0008	0.0542 $\pm$ 0.0010
	10	0.0459 $\pm$ 0.0014	0.0431 $\pm$ 0.0005
	15	0.0408 $\pm$ 0.0008	0.0394 $\pm$ 0.0005
	20	0.0381 $\pm$ 0.0008	0.0370 $\pm$ 0.0004
	25	0.0362 $\pm$ 0.0004	0.0361 $\pm$ 0.0003
CIFAR-10	3	0.0552 $\pm$ 0.0041	0.0477 $\pm$ 0.0004
	5	0.0452 $\pm$ 0.0004	0.0444 $\pm$ 0.0003
	10	0.0412 $\pm$ 0.0005	0.0414 $\pm$ 0.0005
	15	0.0393 $\pm$ 0.0001	0.0395 $\pm$ 0.0004
	20	0.0382 $\pm$ 0.0002	0.0383 $\pm$ 0.0001
	25	0.0374 $\pm$ 0.0002	0.0376 $\pm$ 0.0002
SVHN	3	0.0339 $\pm$ 0.0025	0.0262 $\pm$ 0.0004
	5	0.0278 $\pm$ 0.0032	0.0224 $\pm$ 0.0004
	10	0.0279 $\pm$ 0.0058	0.0207 $\pm$ 0.0008
	15	0.0221 $\pm$ 0.0037	0.0197 $\pm$ 0.0002
	20	0.0186 $\pm$ 0.0011	0.0190 $\pm$ 0.0003
	25	0.0172 $\pm$ 0.0002	0.0181 $\pm$ 0.0003
EuroSAT	3	0.0145 $\pm$ 0.0007	0.0120 $\pm$ 0.0005
	5	0.0106 $\pm$ 0.0007	0.0107 $\pm$ 0.0006
	10	0.0095 $\pm$ 0.0003	0.0096 $\pm$ 0.0005
	15	0.0089 $\pm$ 0.0001	0.0092 $\pm$ 0.0002
	20	0.0088 $\pm$ 0.0002	0.0090 $\pm$ 0.0001
	25	0.0085 $\pm$ 0.0000	0.0091 $\pm$ 0.0003